Similarity Search of Business Process Models

Marlon Dumas, Luciano García-Bañuelos*
University of Tartu
Estonia
marlon.dumas@ut.ee, lgarcia@ut.ee

Remco Dijkman
Eindhoven University of Technology
The Netherlands
r.m.dijkman@tue.nl

Abstract

Similarity search is a general class of problems in which a given object, called a query object*, is compared against a collection of objects in order to retrieve those that most closely resemble the query object. This paper reviews recent work on an instance of this class of problems, where the objects in question are business process models. The goal is to identify process models in a repository that most closely resemble a given process model or a fragment thereof.*

1 Introduction

As organizations reach higher levels of Business Process Management (BPM) maturity, they tend to accumulate considerable amounts of business process models – reportedly in the hundreds or thousands in the case of multinational companies. These models constitute a valuable asset to support business analysis and system design activities. In organizations with high degrees of BPM maturity, such process models are centrally managed in dedicated process model repositories that provide advanced browsing and search features.

In this paper we review recent developments pertaining to one particular search feature over process model repositories, namely similarity search [3]. In this context, similarity search is defined as follows: given a process model P (the *query*) and a collection of process models C, retrieve the models in C that are most similar to P and rank them according to their degree of similarity. Similarity search is relevant in the context of model maintenance. For example, before adding a model to a repository one needs to check that a similar model does not already exist so as to prevent duplication. Similarly, in the context of company mergers, process analysts need to find overlapping processes across the merged companies in order identify opportunities for consolidation.

Similarity search queries are defined with respect to a similarity measure between pairs of process models. The similarity between pairs of process models can be measured on the basis of three complementary aspects of process models: (i) the labels attached to tasks, events and other model elements; (ii) their graph structure; (iii) their execution semantics. The next three sections discuss a number of similarity search techniques classified according to these three criteria. We then summarize the results of an experimental evaluation covering a representative subset of these techniques. Finally, we outline some interesting open research directions.

Copyright 2009 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.
Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*On leave from Autonomous University of Tlaxcala, Mexico, with funding from the European Regional Development Fund.

2 Label Similarity

One way of measuring the similarity between a pair of process models is by first computing an alignment between these models, that is, a relation between elements in one model and elements in the other model. For example, consider the two process models in Figure 1, captured using the Business Process Modeling (BPMN) notation.[1] A possible alignment is the one that matches task "Order" in the first model with the task with the same label in the second model, and task "Verify Invoice" with "Verification Invoice". Given such an alignment, the similarity between two models can be defined as a ratio between the size of the alignment, and the size of the process models. For example, we could define a similarity measure as follows: $\frac{2\times|A|}{|P|+|P'|}$, where A is an alignment and $|P|$ and $|P'|$ denote the number of compared model elements in models P and P' respectively. In the above example, this formula gives 0.66 if we only compare tasks (gateways are ignored).

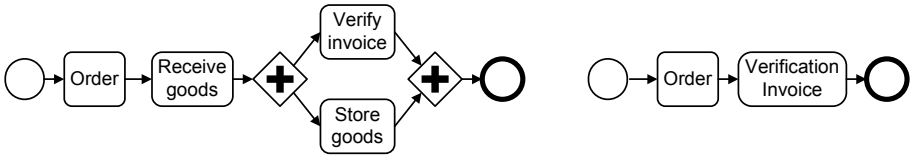

Figure 1: Sample pair of process models

Note that "Verify Invoice" does not perfectly match "Verification Invoice". So instead of giving a weight of one to this match, we should give it a weight equal to the similarity between these two labels (a number between zero and one). The resulting similarity measure is then $\frac{2\cdot\Sigma_{(n,m)\in A}\mathrm{Sim}_l(n,m)}{|P_1|+|P_2|}$ where Sim_l is a similarity measure between pairs of model elements. The similarity between model elements can be computed from their labels using syntactic similarity measures, semantic measures, or a combination of both. Syntactic measures are based on string-edit distance, n-gram, morphological analysis (stemming), and stop-word elimination techniques, whereas semantic techniques are based on synonym and other semantic relations captured in thesauri (e.g. Wordnet).[2] For example, "Verify Invoice" and "Verification Invoice" have a high syntactic similarity because they are almost identical after stemming, whereas "Verify Invoice" and "Check Invoice" have a high semantic similarity since "Verify" and "Check" are synonyms.

Variations of the above similarity measure have been proposed by Ehrig et al. [7] and Dijkman et al. [6]. In this latter work, the problem of finding a mapping between model elements is reduced to the linear assignment problem [9] for which efficient algorithms exist. In other work, label similarity measures are used in conjunction with structural or behavioural similarity measures (e.g. [8]).

3 Structural Similarity

Since process models are graphs, we can use graph matching as a basis for defining similarity measures. Specifically, given two process models, we can define their similarity as the opposite of their graph-edit distance [12] (i.e. one minus their normalized graph-edit distance). The graph-edit distance of two graphs is the minimum number of operations (insertions, deletions, substitutions) needed in order to transform one graph into the other. Unfortunately, the problem of computing the graph-edit distance is NP-hard. Thus, one needs to strike a tradeoff between computational complexity and accuracy. Below we review several possible tradeoffs.

[1] http://www.bpmn.org

[2] These linguistic matching techniques are also widely employed in the field of schema matching [14].

3.1 A* Algorithm

An exact method to compute the graph edit distance is by applying the A* search algorithm [12]. In the context of graph-edit distance calculation, the algorithm aims at constructing an alignment (mapping) between the nodes of the two graphs. Given a mapping, the graph-edit distance is calculated by considering that all non-mapped nodes have been deleted (or added), and that all pairs of nodes in the mapping correspond either to identical nodes or to node substitutions. The goal is to find the mapping with the smallest graph-edit distance. To this end, the A* graph matching algorithm starts with an empty mapping, which has a maximal edit distance since it implies that to go from one graph to the other, all nodes in one graph should be deleted and all nodes in the second graph should be added (no identical nodes and no substitutions). The algorithm incrementally constructs partial mappings of larger size, until it can no longer find a way of creating a larger mapping with a lower edit-distance. At each step, a number of new mappings are constructed by using the current partial mapping with the smallest edit-distance and adding new possible pairs to this mapping. It can be proved that this strategy leads to an optimal final mapping.

A major issue with this algorithm is the large number of partial mappings that must be maintained during the search – $O(m^n)$ in the worst case [12], where m and n are the numbers nodes in the two graphs that are being compared. Our own experiments have shown that this is problematic for process models with over 20 nodes [4]. We addressed this issue by forbidding pairs of nodes to be added to a mapping if their labels are too different from one another – but this breaks the optimality property of the algorithm.

In its basic form, the A* graph-edit distance algorithm only considers elementary graph-edit operations (deletion, insertion and substitution), meaning that only 1-to-1 node mappings are constructed. In [8] the set of graph-edit operations is extended with node splitting and node merging, i.e. N-to-M mappings are constructed. In [5], we observed that the number of partial mappings grows combinatorially when node splitting/merging operations are integrated and proposed an alternative two-step approach. First, the basic A-star algorithm is used to obtain a 1-to-1 mapping. In the second step, all deleted/inserted nodes are combined with adjacent nodes trying to improve the mapping. Combining a deleted node with a matched one is similar to node merging, while combining an inserted node with a matched one is similar to node splitting. In this way, the N-to-M mapping can be computed while maintaining the memory requirements of the 1-to-1 mapping approach.

3.2 Heuristic Search

Given the scalability limitations of exact graph matching techniques, several heuristics have been developed. In [4], we studied three such heuristics. The first one is a greedy heuristics, in which a mapping is created starting from an empty mapping and adding, at each iteration, the pair of most similar nodes that do not yet appear in the current mapping. The other two heuristics are closer to the A* graph matching algorithm in the sense that they explore several possible mappings (instead of just one current mapping as the greedy approach). However, these heuristics include a pruning function which is triggered when the number of currently considered mappings is larger than a given threshold. Only the most promising alignments are kept after a pruning phase.

3.3 Similarity Flooding

Given that process similarity can be measured on the basis of an alignment, the problem of similarity search can be related to alignment problems such as schema matching [14]. Similarity flooding is a graph matching technique that has been shown to yield good accuracy when applied to the problem of schema matching [11]. The idea behind similarity flooding is that a pair of nodes or edges of two graphs are similar when their adjacent elements are similar. The algorithm builds a matrix for similarity propagation which is updated iteratively by fixpoint computation. At the end, the matrix can be used to construct an alignment, e.g. construct a mapping with pairs of nodes whose similarity is greater than a given threshold. However, as stated in [11] the algorithm works fine for directed labelled graphs and degrades when edge labelling is uniform or absent.

Madushan et al. [10] have studied the application of similarity flooding for process model similarity search. In their work, process models are represented using an ontology-based notation in which process models are represented as graphs with labelled nodes and labelled edges. However, mainstream process modeling notations (e.g. BPMN) are such that edges have no labels, thus hindering the applicability of similarity flooding.

4 Behavioral Similarity

Process models define a part of the behavior of an organization. Therefore, another possibility for measuring their similarity is by measuring the similarity of their behavioral semantics. There are a number of behavioral semantics of business processes. For each of these behavioral semantics similarity metrics can be defined.

4.1 Comparison of traces

A simple way to define the execution semantics of a process model is in terms of the set of (completed) traces that it can accept. Assuming process models with finite sets of traces, we can define similarity measures in terms of this trace-based semantics. For example, we can define the similarity between two process models $P1$ and $P2$ as the ratio $\frac{2\times|T(P1)\cap T(P2)|}{|T(P1)\cup T(P2)|}$ where $T(P)$ is the set of traces generated by process P. However, this simple measure leads to unsatisfactory results. For example, the sets of traces of the two models shown in Figure 1 have an empty intersection. A more suitable alternative is to consider partial traces. Wombacher [16] studied the use of N-grams (partial traces consisting of n-items) as a basis to compare process models. Another technique based on a notion of "partially fitting traces" is presented in [1].

4.2 Simulation

A second way of defining behavioral semantics is in terms of a labelled transition system that captures all the states in which the process model can be, and all transitions that can cause the process model to change state. To determine if two process models are equivalent, we can then take the state-space of two process models and check if they can simulate one another (i.e. if they allow the same transitions in equivalent states). If the process models are not equivalent, we will find states which can be reached through the same sequence of transitions, and yet do not allow the same transitions. By counting such states, we can measure how dissimilar two process models are. This idea is applied by Nejati et al [13] in order to match statechart diagrams and could in principle be applied to process models.

4.3 Causal footprints

Trace- and state-based semantics aim to describe the behavior of a process as precisely as possible. However, their use can lead to performance problems due to large sets of traces and state explosion, while their level of precision is not required for measuring similarity. An approximation of the behavioral semantics of business processes would be sufficient for similarity measure. A possible approximation of this behavioral semantics is given by the concept of causal footprint [15].

A causal footprint of a business process is a triple (E, L_{lb}, L_{la}), where: E is the set of elements of the business process (e.g. tasks); $L_{lb} \subseteq \mathcal{P}(E) \times E$ is the set of look-back links, such that (lb, e) denotes that at least one element from lb must have occurred before e can occur; and $L_{la} \subseteq E \times \mathcal{P}(E)$ is the set of look-ahead links, such that (e, la) denotes that after e has occurred at least one element from la must occur. For example, if we identify the tasks in the example in figure 1 by the first letter of their label, a causal footprint for the leftmost process could be: $(\{O, R, V, S\}, \{(\{O\}, R), (\{R\}, V), (\{R\}, S)\}, \{(O, \{R\}), (R, \{V\}), (R, \{S\})\})\}$. Note that, if we add $(\{O\}, V)$ to the look-back links, the resulting causal footprint is still a valid footprint for the process. This illustrates that causal footprints are an approximate semantics.

The similarity search method based on causal footprints differs from other similar search methods in that it does not compute the similarity between each pair of process models. Instead, each process model in a collection is represented as a point in a vector space, and the problem of process model similarity search is reduced to the nearest-neighbour problem, that is, finding the nearest points to the query. This technique is often employed in the field of information retrieval for text documents. In the case of causal footprints, the dimensions (also known as the terms) of the vector space are the elements (e.g. tasks), the look ahead-links and the look-back links that appear in at least one business process in the collection. For a given business process, the values for the dimensions (also known as the weights of the terms) are determined based on the presence of the term in the process in question and the 'importance' of that term. Look-back and look-ahead links that consist of fewer elements are considered more important than those that consist of more elements.

5 Comparison

Table 1 summarises the results of an empirical evaluation of 8 similarity search techniques covering the three categories reviewed above. The evaluation involved 10 queries executed over a set of 100 process models. Details of the dataset and the evaluation method are given in [6, 4]. The table shows the mean average precision obtained for each technique across all 10 queries. Average precision is a measure commonly used to evaluate the quality of search techniques that return ranked lists of results [2]. The mean average precision for a given technique is the arithmetic mean of the average precisions obtained for each query using that technique.

Table 1: Mean average precision of representative search techniques (adapted from [4, 6])

Algorithm	**Mean avg. precision**	**Algorithm**	**Mean avg. precision**
Syntactic label sim.	0.8	A-star GM	0.86
Semantic label sim.	0.78	Sim. Flooding	0.56
Greedy GM	0.84	Causal Footprint	0.86
Heuristic GM	0.83	Text search engine	0.76

The table suggests that structural and behavioural techniques slightly outperform pure label-based ones. An exception is similarity flooding, which performs poorly, whereas it is known to have good performance in the context of schema matching. This can be explained by the fact that similarity flooding heavily relies on edge labels (in addition to node labels) whereas process models generally lack edge labels. The last row shows the result obtained by using a full-text search engine on the same set of queries. As expected, the average precision is lower than that obtained using any of the reviewed similarity search techniques (except similarity flooding).

6 Outlook

Existing process model similarity search techniques focus on process models composed of atomic tasks and connectors. Little attention has been paid to other process modelling constructs such as sub-process invocation and exception handlers. Perhaps more limiting is the fact that existing process model similarity search techniques tend to focus on the control-flow view of process models, neglecting data manipulation (e.g. data inputs/outputs) and resource allocation. Addressing this limitation is an avenue for further work.

As emphasized in this paper, the problem of similarity search of process models can be related to that of schema matching. Although some differences exist between these problems – particularly the general lack of edge labels in process models – there is an opportunity to transpose schema matching techniques to the process model similarity search problem. Several techniques reviewed in this paper are also found in automated schema matching tools. However, many other schema matching techniques have not yet been considered in the context of process model similarity search.

All process model similarity search techniques we know of employ linear search. In other words, the query model is compared to each model in the collection. An avenue for future work is to study the applicability and performance gains of graph indexing techniques [12] in the context of process model similarity search.

References

[1] W. van der Aalst, A.K. Alves de Medeiros, and A. Weijters. Process Equivalence: Comparing two process models based on observed behavior. In *Proc. of BPM 2006*, volume 4102 of *LNCS*, pages 129–144. Springer, 2006.

[2] C. Buckley and E.M. Voorhees. Evaluating evaluation measure stability. In *Proc. of the ACM SIGIR Conference*, pages 33–40, 2000.

[3] E. Chávez, G. Navarro, R.A. Baeza-Yates, and J.L. Marroquín. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, 2001.

[4] R. Dijkman, M. Dumas, and L. García-Bañuelos. Graph matching algorithms for business process model similarity search. In *Proc. of BPM 2009*, Ulm, Germany, September 2009.

[5] R. Dijkman, M. Dumas, L. García-Bañuelos, and Reina Käärik. Aligning business process models. In *Proc. of EDOC 2009*, Auckland, New Zealand, September 2009.

[6] R. Dijkman, M. Dumas, B. van Dongen, R. Käärik, and J. Mendling. Similarity of business process models: Metrics and evaluation. Working Paper 269, BETA Research School, Eindhoven, The Netherlands, 2009.

[7] M. Ehrig, A. Koschmider, and A. Oberweis. Measuring similarity between semantic business process models. In *Proc. of APCCM 2007*, pages 71–80, 2007.

[8] D. Grigori, J.C. Corrales, and M. Bouzeghoub. Behavioral matchmaking for service retrieval: Application to conversation protocols. *Inf. Syst.*, 33(7-8):681–698, 2008.

[9] R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38:325–340, 1987.

[10] Therani Madhusudan, J. Leon Zhao, and Byron Marshall. A case-based reasoning framework for workflow model management. *Data and Knowledge Engineering*, 50:87–115, 2004.

[11] S. Melnik, H. García-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm (extended technical report). Technical Report 2001-25, Stanford InfoLab, 2001.

[12] B. Messmer. *Efficient Graph Matching Algorithms for Preprocessed Model Graphs*. PhD thesis, University of Bern, Switzerland, 1995.

[13] S. Nejati, M. Sabetzadeh, M. Chechik, S. Easterbrook, and P. Zave. Matching and merging of statecharts specifications. In *Proc. of ICSE 2007*, pages 54–63, 2007.

[14] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.

[15] B. F. van Dongen, R. M. Dijkman, and J. Mendling. Measuring similarity between business process models. In *Proc. of CAiSE 2008*, volume 5074 of *LNCS*, pages 450–464. Springer, 2008.

[16] A. Wombacher. Evaluation of technical measures for workflow similarity based on a pilot study. In *Proc. of CoopIS 2006*, volume 4275 of *LNCS*, pages 255–272. Springer, 2006.