Probabilistic Optimization of Semantic Process Model Matching

Henrik Leopold[1], Mathias Niepert[2], Matthias Weidlich[3], Jan Mendling[4], Remco Dijkman[5], and Heiner Stuckenschmidt[2]

[1] Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany
`henrik.leopold@wiwi.hu-berlin.de`
[2] Universität Mannheim, 68159 Mannheim, Germany
`mathias|heiner@informatik.uni-mannheim.de`
[3] Technion - Israel Institute of Technology, Technion City, 32000 Haifa, Israel
`weidlich@tx.technion.ac.il`
[4] Wirtschaftsuniversität Wien, Augasse 2-6, 1090 Vienna, Austria
`jan.mendling@wu.ac.at`
[5] Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands
`r.m.dijkman@tue.nl`

Abstract. Business process models are increasingly used by companies, often yielding repositories of several thousand models. These models are of great value for business analysis such as service identification or process standardization. A problem is though that many of these analyses require the pairwise comparison of process models, which is hardly feasible to do manually given an extensive number of models. While the computation of similarity between a pair of process models has been intensively studied in recent years, there is a notable gap on automatically matching activities of two process models. In this paper, we develop an approach based on semantic techniques and probabilistic optimization. We evaluate our approach using a sample of admission processes from different universities.

1 Introduction

Business process models are increasingly used by companies for documentation purposes. A process documentation initiative stores an extensive amount of process models in a centralized process repository. This amount can easily rise to several thousand models in large enterprises. Due to the size of such companies, process modeling is often conducted by decentralized teams. A consistent and systematic documentation of processes is often achieved by defining guidelines. However, typically none of the team members has detailed insight into the entire set of process models stored in the repository.

The availability of a detailed documentation of a company's business processes bears a lot of potential for business analysis, such as process standardization, compatibility analysis, or business service identification. *Process model matching*, realized by tools called *matchers*, is a prerequisite for such analyses. It defines which activities in one process model correspond to which activities in another model. Such matches are required, for example, to determine which activities can be merged when deriving standard processes from a collection of processes. It is also needed to judge behavior compatibility or equivalence, and to query a collection of business process models for a certain process or process fragment. The importance of such questions is reflected by recent contributions on computing similarity for pairs of process models, e.g. [1,2,3,4,5,6].

In this paper, we address process model matching with semantic matching techniques and probabilistic optimization. The approach comprises two steps. First, match hypotheses are generated based on automatically annotated activity labels. We rely on a semantic interpretation of a activity labels, whereas existing work [7,8] (despite a notable exception [9]) is limited to syntactical similarity assessment. Second, match constraints are derived based on behavioral relations of process models. Those constraints are used for guiding the matching with a probabilistic model, whereas existing work directly leverages the model structure or execution semantics [7,8]. The evaluation of our approach with admission processes from nine different universities shows that the novel conceptual basis for process model matching indeed improves performance. In particular, we are able to show that match results are more stable over different levels of process model heterogeneity. Besides the definition of the matcher, our contribution is a comparative analysis of the strengths and weaknesses of classical matchers and semantic matching with probabilistic optimization. As such, we provide valuable insights for advancing the field of process model matching.

Against this background, the paper is structured as follows. Section 2 illustrates the problem of matching process models. Section 3 presents a matcher that incorporates the generation of semantic match hypotheses based on automatically annotated activities and a probabilistic approach towards match optimization using behavioral constraints. Section 4 challenges our approach using a process model collection from practice. Section 5 reflects our contribution in the light of related work. Finally, Section 6 summarizes the findings.

2 Problem Illustration

This section illustrates the problem of matching process models. We present basic terminology and discuss the state of the art in finding matches.

Given two process models with sets of activities $\mathcal{A}_1$ and $\mathcal{A}_2$, matches between their activities are captured by a relation $match : \mathcal{P}(\mathcal{A}_1) \times \mathcal{P}(\mathcal{A}_2)$. An element $(A_1, A_2) \in match$ defines that the set of activities A_1 matches the set of activities A_2, i.e., they represent the same behavior in the organization. If $|A_1| = 1$ and $|A_2| = 1$, we call the match an *elementary match*. Otherwise, we speak of a *complex match*. For convenience, we introduce a relation $map : \mathcal{A}_1 \times \mathcal{A}_2$, which defines the relations between individual activities as induced by $match$, $map = \{(a_1, a_2) | (A_1, A_2) \in match, a_1 \in A_1, a_2 \in A_2\}$.

Figure 1 shows admissions processes from two different universities. We highlighted matches by gray boxes around the activities, e.g., activity *Check formal requirements* of University A corresponds to activity *Check documents*

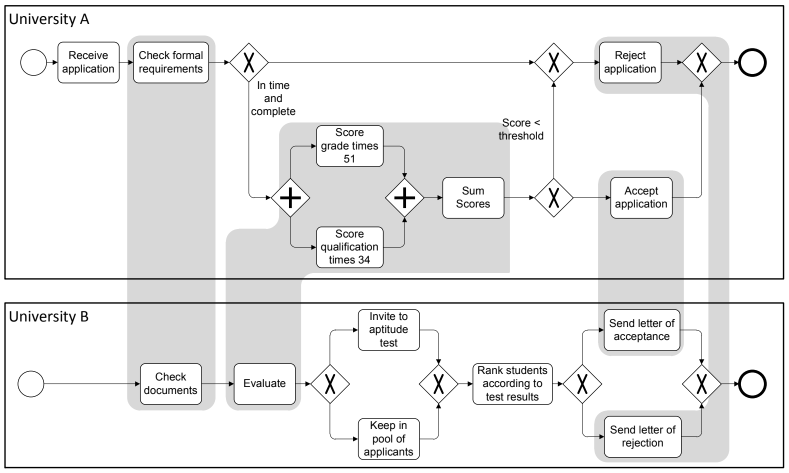

Fig. 1. Example of a business process models with matches.

of University B. Although the processes have the same goal, the organizational behavior is modeled differently. Different labels are used (e.g., *Accept application* versus *Send letter of acceptance*) and there are differences in the level of detail (e.g., *Evaluate* of University B is described in more detail for University A). In addition to being modeled differently, the behavior represented by the processes also differs. For example, at University B the *Evaluate* activity is mandatory, whereas at University A the matching activities can be skipped. Before these behavioral differences can be analyzed, however, matches between the activities have to be determined. The goal of *matchers*, such as the ones described in [8] and [7], is to detect such matches automatically.

A matching approach of particular interest is the ICoP framework [8]. It defines a generic architecture for assembling matchers along with reusable matching components. As such, it integrates several of the proposed matchers, e.g., the graph-based matcher presented in [7]. Following the ICoP architecture, the procedure for automatically detecting matches involves four kinds of matching components: *searchers* find potential matches between activities, *boosters* improve the quality of potential matches by combining them, *selectors* construct the actual mapping from potential matches, and *evaluators* evaluate the quality of an actual mapping with the purpose of finding the best mapping.

Matching components implemented for the ICoP framework leverage syntactic measures, such as string edit distance or vector-space scoring, to find match candidates. Selection and evaluation is guided by the structure of process models, e.g., utilizing the graph edit distance. An evaluation of the existing ICoP components showed that much improvement is still possible with respect to automatically detecting matches. Given the focus on syntactic measures of the existing components, approaches that relate activities based on the semantics of their labels can particularly be expected to improve matching performance.

3 Matching based on Semantics and Constraints

This Section introduces our approach for matching process models. It consists of four distinct phases. In the first phase we annotate the activities of the considered models with their semantic components such as action and business object. Afterwards, we use the annotated information in order generate match hypotheses for activity pairs. In the third we phase we compute behavioural constraints in order to properly incorporate control flow aspects into the matching process. Finally, in phase four, we determine the most likely match constellation by using the previously computed aspects as input for a Markov logic network.

3.1 Activity Label Annotation

The goal of the annotation phase is the enablement of a semantic matching of activities which goes beyond a simple string comparison. Therefore, we annotate each activity with the comprised components. As pointed out by [10], every activity label can be characterized by three components: an action, a business object on which the action is performed, and an optional additional information fragment which is providing further details. As an example consider the activity label *Forward Request to Insurance Department*. This label contains the action *forward*, the business object *request* and the additional information fragment *to Insurance Department*.

In order to accomplish the annotation of these components, we employ a technique introduced in prior work [11]. This technique builds on the insight that activity labels follow regular structures, so called label styles. The most frequent label styles are the verb-object and the action-noun style. Labels of the verb-object style are characterized by an imperative verb in the beginning which is followed by the business object and the additional fragment. Examples for verb-object label are *Calculate Costs for Production* or *Verify Invoice of Vendor*. While the action in verb-object labels is given as a verb, the action in action-noun labels is captured as a noun. As examples consider *Order Shipment to Customer* or *User Registration for ERP System*. These examples illustrate that the knowledge about the structural patterns of label styles can be used to properly derive action, business object and the additional fragment. Respectively, the annotation phase is subdivided into two main steps: the recognition of of the activity label style and the actual derivation and annotation of the label components.

Recognition of Label Style In the first step the correct label style is determined. This is impeded by two main challenges which are associated with activity labels: The lack of a rich sentence structure and the zero derivation ambiguity. The latter aspect refers to misinterpretations of words due to the fact that one syntactic word can be interpreted as verb and noun at the same time. Examples are *the plan* and *to plan* or *the transfer* and *to transfer*. In order to cope with these challenges we designed an algorithm which uses different stages of the label context to adequately determine its style. As a typical ambiguous example consider the activity label *Plan Data Transfer* from the SAP Reference Model. From analyzing the label in isolation it is not possible to decide about the label style. The activity could instruct to *plan* a *data transfer* or also to *transfer* a record of *plan data*. However, if we broaden the context and consider the whole process model collection, we can learn that many other processes from this collection deal with the business object *plan data*. Respectively, the label is classified as an action-noun label. If we encounter cases where the context of the process model collection is not sufficient to resolve the ambiguity, we use a word frequency list to probabilistically decide about the word class of the first word in the label. Depending on the result, it is accordingly categorized as a verb-object or action-noun label.

Derivation of Action and Business Object In the second step of the annotation phase action, business object and the additional fragment are determined and respectively annotated. As we already determined the label style in the previous step, this is straightforward. Making use of the structural knowledge about the label styles we e.g. know that a verb-object label begins with an imperative verb which is followed by the business object and the additional fragment. Hence, the verb-object label *Calculate Costs for Production* can be easily decomposed into the action *calculate*, the business object *costs* and the additional fragment *for production*. In the same vein, we derive these components from action-noun labels. We know that the action in action-noun labels is captured as a noun and positioned before the preposition which introduces the additional fragment. Accordingly, the action noun label *User Registration for ERP System* can be decomposed into the action *registration*, the business object *user* and the additional fragment *for ERP system*. By employing the lexical database WordNet [12], the verb *register* can be derived from the nominalized action *registration*. Having derived these components, the considered label can be annotated accordingly.

3.2 Generation of Semantic Match Hypotheses

The generation of semantic match hypothesis builds on the previous annotation of activities. Goal of this phase is the computation of a similarity score for a given activity pair.

The general approach is the calculation of the semantic similarity between the actions, the business objects and the additional information fragments of the considered activity pair. Thereby, the term semantic similarity refers to the similarity between two components in the taxonomy WordNet [12]. There are many proposals how to use taxonomies for calculating the similarity between two concepts [13,14,15]. However, it also has been shown that the similarity measure introduced by Lin correlates best with human judgments [16]. Consequently, we employ the similarity measure of Lin to generate semantic match hypotheses for our process models. In order to calculate this semantic similarity between two labels l_1, l_2, we introduce three functions: a component similarity function

sim_c, a coverage function cov, and a label similarity function sim_l, combining the latter two to a final result.

The function sim_c calculates the semantic similarity between two label components l_{c_1} and l_{c_2}. In case, none or only one label contains the considered component, the similarity is determined with zero. Otherwise the result of the Lin measurement is returned.

$$sim_c(l_1, l_2) = \begin{cases} 0 & \text{if } l_{1_c} = \emptyset \vee l_{2_c} = \emptyset \\ Lin(l_{1_c}, l_{2_c}) & \text{if } l_{1_c} \neq \emptyset \wedge l_{2_c} \neq \emptyset \end{cases} \quad (1)$$

The coverage function cov is used to determine the number of components contained by a label l. Assuming that an activity label at least refers to an action, the result of cov always lies between 1 and 3. Formally, cov is defined as follows. Note that the index a denotes an action, bo the business object and add the additional information fragment.

$$cov(l) = \begin{cases} 1 & \text{if } l_a \neq \emptyset \wedge l_{bo} = \emptyset \wedge l_{add} = \emptyset \\ 2 & \text{if } l_a \neq \emptyset \wedge (l_{bo} \neq \emptyset \veebar l_{add} \neq \emptyset) \\ 3 & \text{if } l_a \neq \emptyset \wedge l_{bo} \neq \emptyset \wedge l_{add} \neq \emptyset \end{cases} \quad (2)$$

In order to combine the individual similarity results, we introduce the function sim_l. This function calculates the arithmetic mean of the similarity values for action, business object and the additional information. This is accomplished by dividing the sum of sim_a, sim_{bo} and sim_{add} by the maximum coverage among l_1 and l_2. As a result, we obtain the overall matching weight for two given labels.

$$sim_l(l_1, l_2) = \frac{sim_a(l_1, l_2) + sim_{bo}(l_1, l_2) + sim_{add}(l_1, l_2)}{\underset{l \, \in \, \{l_1, l_2\}}{\arg\max} \, cov(l)} \quad (3)$$

By calculating sim_l for every activity pair which can be combined from the considered process models, we obtain a set of match hypotheses. This set of hypotheses constitutes the first input for our probabilistic matching model.

3.3 Constraints Generation

Constraint satisfaction, also called second line matching [17], is often applied in schema and ontology matching as a means to guide the selection of matches. Here, constraints may relate to the general structure of matches (e.g., only 1:1 matches shall be considered), particular attribute pairs (e.g., a pair forms a matches or shall never be part of any match), or dependencies between different matches. We aim at matching such dependencies which are related to the execution semantics of process models. The intuition behind is that the order of processing described by one model is likely to coincide with the order of processing specified in a second model. Referring to the initial example in Figure 1, we see that in either model the activities related to check an application (e.g., *Check formal requirements* in University A and *Check documents* in University B) are preceding the activities related to taking a decision (e.g, *Reject application* and *Send letter of rejection*).

Also, activities for accepting an application are exclusive to those of rejection an application in either model.

There are different alternatives to formulate behavioral constraints for a process model. For the context of matching process models, a fine-grained formalization of constraints appears to be appropriate. Although we assume two models to show a rather consistent order of processing, slight deviations can always be expected and should have a minor impact on the matching process. Therefore, we consider a model that captures order constraints for the smallest possible conceptual entity, i.e., pairs of activities. Further, in many cases, the final matching will only be partial, meaning that activities of one model are without counterpart in the other model. This suggests to not rely on direct successorship of activities but on a notion that is insensitive of partial matchings.

Against this background, we capture behavioral constraints using a binary relation over activities, called weak order [18]. It holds between two activities a_1 and a_2 of a process model, if there exists an execution sequence in which a_1 occurs before a_2. By referring to the existence of a certain execution sequence, it captures the potential order of occurrence for activities. In the example, weak order holds between *Check formal requirements* and *Reject application* for University A, and between *Check documents* and *Send letter of rejection* for University B. Weak order implicitly allows to conclude on strict order between activities (if weak order holds only in one direction) and exclusiveness of activities (if weak order does not hold in any direction). Hence, exclusiveness of the activities representing acceptance and rejection of an application in either model is also covered.

Weak order of activities can be derived from the state space of a process model. For certain classes of models, however, the relation can also be derived directly from the structure. For models that incorporate only basic control flow routing, such as XOR and AND routing constructs, and that show soundness, i.e., the absence of behavioral anomalies such as deadlocks, the weak order relation is determined in low polynomial time to the size of the model [18].

3.4 Probabilistic Match Optimization

An instance of the process matching problem consists of the two processes, the match hypotheses with a-priori confidence values, and the behavioral profile relations holding between the activities. Statistical relational languages such as Markov logic [19] are a natural choice when uncertainty meets relational data. We will demonstrate that Markov logic is an excellent choice for a process matching framework as it is adaptable to different matching situations and allows fast prototyping of matching formulations.

Markov Logic Networks Markov logic [19] is a first-order template language for log-linear models with binary variables. Log-linear models are parameterizations of undirected graphical models (Markov networks) which play an important role in the areas of reasoning under uncertainty [20] and statistical relational learning [21]. Log-linear models are also known as maximum-entropy models in the natural language processing community [22]. The *features* of a log-linear model can be complex allowing the user to incorporate prior knowledge about the importance of features of the data for classification. Moreover, within the framework of log-linear models users can specify *constraints* on the resulting classification. In the context of process matching, these constraints will allow us to punish inconsistent alignments.

A Markov network $\mathcal{M}$ is an undirected graph whose nodes represent a set of random variables $\mathbf{X} = \{X_1, ..., X_n\}$ and whose edges model direct probabilistic interactions between adjacent nodes. More formally, a distribution P is a log-linear model over a Markov network $\mathcal{M}$ if it is associated with:

- ○ a set of features $\{f_1(D_1), ..., f_k(D_k)\}$, where each D_i is a clique in $\mathcal{M}$ and each f_i is a function from D_i to $\mathbb{R}$,
- ○ a set of real-valued weights $w_1, ..., w_k$, such that

$$P(\mathbf{X} = \mathbf{x}) = \frac{1}{Z} \exp\left(\sum_{i=1}^{k} w_i f_i(D_i)\right),$$

where Z is a normalization constant [20].

A Markov logic network is a set of pairs (F_i, w_i) where each F_i is a first-order formula and each w_i a real-valued weight associated with F_i. With a finite set of constants C it defines a log-linear model over possible worlds $\{\mathbf{x}\}$ where each variable X_j corresponds to a ground atom and feature f_i is the number of true groundings (instantiations) of F_i with respect to C in possible world $\mathbf{x}$. Possible worlds are truth assignments to all ground atoms with respect to the set of constants C. We explicitly distinguish between weighted formulas and *deterministic* formulas, that is, formulas that always have to hold.

There are two common types of probabilistic inference tasks for a Markov logic network: Maximum a-posteriori (MAP) inference and marginal probability inference. The latter computes the posterior probability distribution over a subset of the variables given an instantiation of a set of evidence variables. MAP inference, on the other hand, is concerned with finding an assignment to the variables with maximal probability. Assume we are given a set $\mathbf{X}' \subseteq \mathbf{X}$ of instantiated variables and let $\mathbf{Y} = \mathbf{X} \setminus \mathbf{X}'$. Then, a most probable state of the ground Markov logic network is given by

$$\underset{\mathbf{y}}{\operatorname{argmax}} \sum_{i=1}^{k} w_i f_i(D_i).$$

Similar to previous work on matching ontologies with Markov logic[23,24], we can specify a set of hard and soft constraints that improve the overall matching results. Finding the most likely alignment then translates to computing the maximum a-posteriori state of the ground Markov logic network.

Markov Logic Formulation of Process Matching Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be the activities of two process models to be mapped, we describe each process model using their behavioral profiles in terms of weak order relations $wo_1 : \mathcal{A}_1 \times \mathcal{A}_1$

and $wo_2 : \mathcal{A}_2 \times \mathcal{A}_2$ between activities of the two process models. Furthermore, the mapping hypotheses are represented by a mapping relation $map : \mathcal{A}_1 \times \mathcal{A}_2$. In the Markov logic formulation, the relations wo_1 and wo_2 are modeled using observable predicates, that is, predicates whose ground state is known a-priori whereas the relation *map* is modeled using a hidden predicate. Hence, when an optimal alignment between activities in the two models is computed, we treat the the weak-order relations as observed and the mapping relation as a hidden predicate. For convenience, we also define the strict-order and the exclusiveness relation between activities of a process model as follows:

$$so_i(a_i, b_i) \Leftrightarrow wo_i(a_i, b_i) \wedge \neg wo_i(b_i, a_i)$$
$$ex_i(a_i, b_i) \Leftrightarrow \neg wo_i(a_i, b_i) \wedge \neg wo_i(b_i, a_i)$$

Using these relations, we can simply represent the constraints as a set of first-order formulas and add those to the Markov logic formulation. The knowledge base consists of the output of the base matcher encoded in terms of weighted atoms of the map relation acting as evidence plus two sets of atoms of the order relations mentioned above as static knowledge. The final result of the matching process is now computed by adding additional constraints and computing the a posteriori probability of the map atoms. We experimented with different types of constraints that have proven useful in the area of ontology matching and which we adapted to the case of process matching.

Cardinality It has been shown that restricting alignments to one-to-one matches typically leads to better results in ontology matching. In particular, because gold standard alignments in this area tend to be one-to-one. While this is clearly not the case for process matching, as processes are often described at different levels of granularity, the cardinality of the mapping relation is still an important constraint to avoid a too strong bias towards an alignment with too many erroneous matches. We experimented with different cardinality settings encoded using the following formulas.

$$|\{activitiy(a) | \exists b : map(a, b)\}| < n$$
$$|\{activitiy(b) | \exists a : map(a, b)\}| > m$$

Stability Stability is a constraint expressing that the structural properties of the aligned objects should be as identical as possible [**?**]. In particular, stability means that semantic relations that hold between two elements in one representation should also hold between the two elements in the other representation they are mapped to. In the case of process matching, we can "encourage" this notion of stability for the three order relations mentioned above, namely the weak order relation, strong order relation, and exclusiveness relation, by using the following implicitly universally quantified formulas where $a_i, b_i, i \in \{1, 2\}$, are activities in process model i.

$$wo_i(a_i, b_i) \wedge \neg wo_j(a_j, b_j) \Rightarrow \neg(map(a_1, a_2) \wedge map(b_1, b_2)) \text{ with } i, j \in \{1, 2\}, i \neq j$$

$$so_i(a_i, b_i) \wedge \neg so_j(a_j, b_j) \Rightarrow \neg(map(a_1, a_2) \wedge map(b_1, b_2)) \text{ with } i, j \in \{1, 2\}, i \neq j$$

$$ex_i(a_i, b_i) \wedge \neg ex_j(a_j, b_j) \Rightarrow \neg(map(a_1, a_2) \wedge map(b_1, b_2)) \text{ with } i, j \in \{1, 2\}, i \neq j$$

Please note that these constraints do not need to be hard. Indeed, our empirical results have shown that in the process matching setting, constraints should be soft, making alignments that violate these respective constraints possible but less likely. Intuitively speaking, the more soft constraints an alignment violates, the less probable it is.

Coherence A weaker class of constraints are those that encourage *logical coherence* of the integrated model. More specifically, such constraints exclude conflicting combinations of semantic relations in the integrated model. In the case of process matching, coherence criteria can be formulated using order relations. The basic idea is that activities that are exclusive in one of the models should not be in a weak order or a strict-order relation in the other model.

$$so_i(a_i, b_i) \wedge ex_j(a_j, b_j) \Rightarrow \neg(map(a_1, a_2) \wedge map(b_1, b_2)) \text{ with } i, j \in \{1, 2\}, i \neq j$$

Another form of incoherence results when the strict order relations of aligned activities in the two models are inverted leading to a conceptual conflict in the merged process model. The constraint making alignments that cause this kind of incoherence less likely is sometimes referred to as 'criss-cross mappings' in the ontology matching setting [] and can be formalized as follows.

$$so_i(a_i, b_i) \wedge so_j(b_j, a_j) \Rightarrow \neg(map(a_1, a_2) \wedge map(b_1, b_2)) \text{ with } i, j \in \{1, 2\}, i \neq j$$

Note that coherence is a weaker form of stability that does not enforce the same semantic relation to hold, but only excludes incompatible semantic relations between mapped elements.

4 Evaluation

In this section we present an evaluation of the defined concepts. More specifically, Section 4.1 describes the sample of admission process models from different German universities that we use to that end. Section 4.2 summarizes the results for applying probabilistic match optimization using Markov logic networks. Section 4.3 compares the results of our optimized semantic matching approach with syntactic matching in ICoP. Furthermore, we discuss the results of the two approaches in terms of their strengths and weaknesses.

4.1 Study Admission Processes of Nine German Universities

Up until now, there is no commonly accepted sample available for testing matching algorithms for process models. Therefore, we created such a sample based on modeling projects of graduate students from Humboldt-Universität zu Berlin, Germany. These students participated in a research seminar on process modeling in three different semesters. The task of this seminar was to document the study admission process of a Germany university, and to compare the process with those of other student groups. This exercise yielded nine admission process models from different universities, which were created by different modelers using different terminology and capturing activities at different levels of granularity. All processes were modeled in BPMN, while the format analysis was conducted on a corresponding Petri net representation. The minimum number of activities in a process model was 10 ranging up to 44. On average, a process model has 21 activities in this sample.

The combination of those nine processes results in $9 * 8/2 = 36$ model pairs. In order to build our test sample, we involved three researchers in building the gold standard of the pairwise activity mappings. Matches were identified by two researchers independently, and the third researcher resolved those cases where different matches were proposed. We used the process models and the gold standard as input of two matching tools. We used the existing ICoP prototype for generating 1:1 matches, for short ICoP. For the approach presented in this paper, we implemented a separate prototype that incorporated different components for annotation [11], for constraint generation [18], TheBeast[6] for Markov logic networks [25], and the mixed integer programming solver Gurobi[7] to solve integer linear programs derived from the Markov logic networks. For short, we refer to this second prototype as Markov. Both of these matching prototypes were utilized to automatically generate matches between activities for each pair of process models. Those matches were compared with the matches defined in the gold standard. Using the gold standard, we can classify each proposed activity match as either true-positive (TP), true-negative (TN), false-positive (FP) or false-negative (FN). These sets provide the basis for calculating the *precision* (TP/(TP+FP)) and *recall* (TP/(TP+FN)) metrics. We will also report the F_1 measure, which is the harmonic mean of precision and recall (2∗precision∗recall/(precision+recall)).

4.2 Evaluation of Match Optimization

In this section, we investigate in how far the matching result benefits from the stability and coherence constraints as incorporated in the Markov prototype. To this end, we conducted experiments with different combinations of constraints. All experiments were conducted on a desktop PC with AMD Athlon Dual Core Processor 5400B with 2.6GHz and 1GB RAM. Our conjecture was that by the help of the constraints and the Markov logic network optimization we would improve precision without compromising recall too much. If so, the corresponding F_1 value should increase. Our base case is a configuration without any constraints, which yielded 0.079 precision, 0.572 recall, and an F_1 of 0.136.

Table 1 summarizes the findings. The initial introduction of a 1:1 match cardinality constraint improves the results towards an F_1 of 0.27, with precision

[6] http://code.google.com/p/thebeast/
[7] http://www.gurobi.com/

Table 1. Precision, Recall, F_1 and processing time for different constraint types

configuration	precision	stddev.	recall	stddev.	F_1	stddev.	avg. time [s]
no constraints	0.079	0.033	0.572	0.205	0.136	0.052	1.1
cardinality 1:1	0.278	0.172	0.280	0.228	0.270	0.193	1.3
1-1 cardinality with							
weak order stability	0.421	0.217	0.263	0.170	0.315	0.182	109.2
strict order stability	0.354	0.216	0.304	0.236	0.316	0.216	41.5
exclusiveness stability	0.280	0.174	0.234	0.174	0.247	0.170	50.3
so-exclusiveness coherence	0.306	0.179	0.252	0.178	0.268	0.171	45.3
so-so coherence	0.342	0.195	0.317	0.226	0.318	0.197	16.7

Table 2. Precision, Recall, F_1 and processing time for Markov and ICoP

prototype	precision	stddev.	recall	stddev.	F_1	stddev.
Markov (weak order stability)	0.421	0.217	0.263	0.170	0.315	0.182
ICoP	0.506	0.309	0.255	0.282	0.294	0.253

and recall at roughly 0.28. We use this configuration to introduce the three types of stability constraints. It can be seen that both types of order constraints improve the match results, the F_1 rises to 0.315 and 0.316, respectively. Strict order coherence yields a comparable result. Exclusiveness-related stability and coherence prove to be less effective. The F_1 decreases due to a loss in recall.

These results suggest that order relations appear to be helpful in finding correct and ruling out incorrect matches. In comparison to the base case, the results improve from 0.136 to 0.315 for weak order stability in terms of the F_1 value. Compared to the case with only cardinality constraints ($F_1 = 0.27$), weak order stability yields a considerably better precision at the expense of a small loss in recall. This points to the potential of order constraints to inform automatic process matching.

4.3 Semantic versus Syntactic Matching

After having demonstrated the benefits of constraint optimization in the Markov prototype, this section aims to investigate in how far its usage of semantic match hypotheses advances beyond the syntactic match strategies of ICoP. We approach this question by considering the average precision, recall and F_1 measure for the admission process sample along with their standard deviation.

Table 2 provides the figures for comparing the Markov prototype and the existing ICoP prototype. It can be seen that ICoP achieves a better precision, but a weaker recall. However, the Markov prototype yields a better F_1 measure of 0.315 in comparison to 0.294. It is interesting to note that the Markov prototype achieves these results with a much lower standard deviation. The difference in standard deviation ranges from 0.071 up to 0.112. We might see in this difference an indication that the Markov prototype is more robust and less sensitive to specific characteristics of the process pair to be matched.

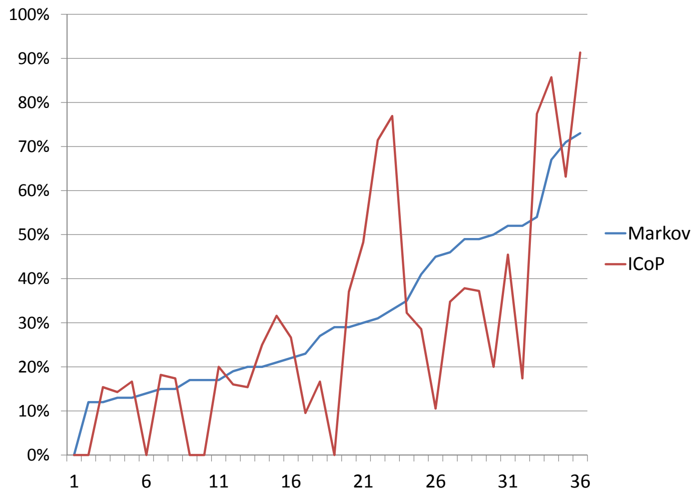

Fig. 2. F_1 measure of Markov and ICoP for the 36 match pairs, ordered by Markov result.

In order to understand which characteristics might favour one or the other approach, we plotted the F_1 measure for both as shown in Figure 2. For 20 of the 36 pairs Markov yielded better results, while ICoP was better in 16 cases. There are a few pairs with substantial difference: In three cases ICoP is better with a difference of more than 0.20, namely 0.234, 0.404, and 0.439. For four pairs, Markov is better with a difference of 0.290, 0.300, 0.345 and 0.346. We aim to illustrate the three classes of *comparable results*, *better ICoP*, and *better Markov* results by the help of three characteristic process model pairs.

Comparable Results: If comparable results are observed for both approaches, the resulting F_1 values remain in the lower range. The pair FU Berlin and TU Munich is one such example where Markov yields 0.20 and ICoP 0.25. The FU Berlin process has 21 activities are is described on a more fine-granular level at the TU Munich process with its 11 activities. There are seven 1:1 matches between these models and three 1:n matches. Eight activities of FU Berlin have no counterpart in the TU Munich process, and one Munich activity has no match. Both approaches suffer from the fact that both processes contain several activities that mention the same verb: the FU Berlin process has two activities with *to add* (*Add Certificate of Bachelor Degree* and *Add Certificate of German language*, four activities involving *to check* and three *send* activities; the TU Munich process has four *send* activities. ICoP provides one false-positive and eleven false-negatives; Markov has five false-positive and eleven false-negatives.

Table 3. Explorative results on relative strengths of Markov and ICoP

Match Pair	FU Berlin TU Munich	Cologne Frankfurt	Hohenheim Erlangen
Better Approach	Comparable	ICoP	Markov
Activities	21	10	25
	11	12	30
1:1 Match	7	6	6
1:n Match	3	0	4
No Match	9	10	28
ICoP False-Positives	1	1	3
Markov False-Positive	5	4	3
ICoP False-Negatives	11	1	17
Markov False-Negatives	11	4	10
ICoP F_1	0.25	0.77	0.17
Markov F_1	0.20	0.33	0.52

Better ICoP: ICoP yielded significantly better results for the match pair Cologne-Frankfurt (F_1 of 0.76 in comparison to 0.33 by Markov). The Cologne process has 10 activities, Frankfurt 12. There are six 1:1 matches and no 1:n matches. Four and six activities on each side, respectively, have no match partner. Five of these matches are syntactically equivalent, another being a substring of its match (*Acceptance* and *Send letter of Acceptance*). While the good performance of ICoP is no surprise, it is interesting that the semantic approach in Markov shows weak results. There are four false positive, which are semantically very close, but no match for this model pair (e.g. *Take Aptitude Test* and *Take Oral Exam*). As a consequence, the probabilistic optimizer penalizes some syntactically equal and correct matches. Markov could be improved by generating 100% confidence match hypotheses for syntactically identical activities. It is interesting to note that also the second case of superior performance of ICoP can be traced back to a great share of syntactically identical matches.

Better Markov: The processes for Hohenheim and Erlangen are much better matched by Markov than by ICoP. The two process models of this match pair have 25 and 30 activities, respectively. There are six 1:1 matches, four 1:n matches, and 28 activities without a match in the other model. While ICoP yields a low F_1 of 0.17, Markov achieves a respectable 0.52. ICoP only finds three correct matches, all being syntactically closely related (e.g. *Checking if complete* and *Check Application Complete*). It is interesting to find that Markov substantially benefits both from semantic match pairs and constraint optimization. Among others, the correct match *publishing the letters* and *send acceptance* is added by the help of the weak order stability and its semantic similarity. The weak order rule also helps to eliminate eight false matches including *Receiving the written applications* and *receive rejection.*

Table 3 summarizes the exploratory results on relative strengths of Markov and ICoP. The following three conclusions can be drawn from this evaluation, also from further investigation of the data. First, both approaches benefit from an increase in the number of 1:1 matches. The number of 1:1 matches is strongly correlated with the F_1 of both approaches for our sample with 0.646 and 0.637, respectively. Second, ICoP suffers from an increase in the number of not matched activities. We find a correlation of -0.143. Interestingly, there is no such correlation for Markov. Examples like the Hohenheim-Erlangen case suggest that the optimizer works well in filtering our unjustified match hypotheses based on weak order. Third, both approaches suffer from an increase in the number of 1:n matches. Interestingly, the decrease is much stronger for ICoP with a correlation of -0.461. For Markov, this correlation is only -0.166. Markov seems to benefit from semantic similarity in hypothesis generation, which turns out to be a remedy to some extent for representation on different levels of granularity. While these advantages of semantic matching appear to be stronger for larger models, there is the need to account for trivial matches that are syntactically the same. Markov has lost some share of its performance by not directly accepting such trivial matches. Nevertheless, it will be rather straight to account for such matches in Markov.

5 Related Work

The work presented in this paper mainly relates to two categories of related research, process model similarity and semantic matching.

Process model similarity techniques can be used to determine how similar two business process models are, usually measured on a scale from 0 to 1. There exists a variety of techniques that exploit textual information and the process model structure [2,1] or execution semantics [3,6]. An overview of these techniques is given in [1]. The relevance of process model similarity to process matching is twofold. First, often similarity techniques start by determining similarity of individual activities, which is clearly also of interest when determining matches. Second, similarity techniques often produce a mapping between activities as a byproduct of computing the similarity. The most important difference between similarity and matching is that, when computing the similarity between process models, a matching of lower quality is required than when the matching itself is the goal. Consequently, the similarity techniques are less advanced when it comes to determining matches. They mostly rely on simple (and fast) label comparison rather than semantic techniques to determine similarity of activities and neglect complex matches. There is one notable exception [9] that leverages synonyms from WordNet [26]. Our fine grained interpretation of activity labels, however, goes beyond the approach presented in [9].

Semantic matching has received considerable attention for schema and ontology matching, see [27,28,29]. In essence, semantic matching refers to the identification of relations (equivalence, more or less general, disjoint) between concepts, i.e., interpretations of schema or ontology entities [30]. Most prominently, the S-Match system [31] realized semantic matching by first interpreting labels and entities, which yields a set of concepts, before establishing relations between them. This approach heavily relies on external knowledge basis, such as WordNet [26]. Those are used to interpret single labels and derive concepts, but also to determine the semantic relations between concepts. Our approach for process model matching takes up these ideas: we interpret activity labels by extracting actions and business objects, i.e., concepts, to generate match hypothesis.

6 Conclusion

In this paper we presented a novel approach for automatically matching process models. The introduced technique combines semantic techniques and probabilistic optimization. The approach has been implemented as a prototype and tested using a set of 36 model pairs derived from 9 university admission processes. In addition, we compared the evaluation results with the performance of the previously introduced matching framework ICoP.

In general, the evaluation shows that our approach yields a higher overall F_1 value of 0.315 in comparison to 0.294 from ICoP. Furthermore, the lower standard deviation among the results indicates that the results of the Markov approach are much more stable. However, the detailed analysis of the matching pairs also revealed some specific strengths and weaknesses. First, we found out that ICoP has some problems with recognizing non-matched activities while Markov can handle this well. Further, Markov has a better ability to recognize matches on a different level of granularity. This clearly shows the advantage of the incorporation of semantic similarity hypotheses. However, while Markov is better able to recognize semantic relations, there are some cases where Markov actually fails to recognize trivial matches.

In future work we plan to improve our approach based on the identified weaknesses. Accordingly, we aim for addressing 1:n matches by adapting the current Markov logic implementation. Further, we intend to improve the calculation of the semantic match hypothesis. This may include the incorporation of domain ontologies or domain corpora which help to increase the accuracy of the calculated hypotheses.

References

1. Dijkman, R., Dumas, M., van Dongen, B., Käärik, R., Mendling, J.: Similarity of business process models: Metrics and evaluation. Information Systems **36** (2010) 498–516
2. Grigori, D., Corrales, J.C., Bouzeghoub, M., Gater, A.: Ranking bpel processes for service discovery. IEEE T. Services Computing **3**(3) (2010) 178–192
3. Zha, H., Wang, J., Wen, L., Wang, C., Sun, J.: A workflow net similarity measure based on transition adjacency relations. Computers in Industry **61**(5) (2010) 463–471
4. Becker, M., Laue, R.: A comparative survey of business process similarity measures. Computers in Industry **63**(2) (2012) 148 – 167

5. Niemann, M., Siebenhaar, M., Schulte, S., Steinmetz, R.: Comparison and retrieval of process models using related cluster pairs. Computers in Industry **63**(2) (2012) 168 – 180
6. Kunze, M., Weidlich, M., Weske, M.: Behavioral Similarity – A Proper Metric. In Rinderle-Ma, S., Toumani, F., Wolf, K., eds.: Proceedings of the 9th International Conference on Business Process Management. BPM '11, Springer Heidelberg (2011) 166–181
7. Dijkman, R., Dumas, M., García-Bañuelos, L., Käärik, R.: Aligning business process models. In: IEEE International EDOC Conference 2009. (2009) 45–53
8. Weidlich, M., Dijkman, R.M., Mendling, J.: The icop framework: Identification of correspondences between process models. In Pernici, B., ed.: Advanced Information Systems Engineering, 22nd International Conference, CAiSE 2010, Hammamet, Tunisia, June 7-9, 2010. Proceedings. Volume 6051 of Lecture Notes in Computer Science., Springer (2010) 483–498
9. Ehrig, M., Koschmider, A., Oberweis, A.: Measuring similarity between semantic business process models. In Roddick, J., Hinze, A., eds.: Conceptual Modelling 2007, Proceedings of the Fourth Asia-Pacific Conference on Conceptual Modelling (APCCM 2007). Volume 67., Ballarat, Victoria, Australia, Australian Computer Science Communications (2007) 71–80
10. Mendling, J., Reijers, H.A., Recker, J.: Activity labeling in process modeling: Empirical insights and recommendations. Inf. Syst. **35**(4) (2010) 467–482
11. Leopold, H., Smirnov, S., Mendling, J.: On the refactoring of activity labels in business process models. Information Systems **37**(5) (2012) 443 – 459
12. Miller, G.A.: Wordnet: a lexical database for english. Commun. ACM 38 (11) (1995) 39–41
13. Lee, J.H., Kim, M.H., Lee, Y.J.: Information retrieval based on conceptual distance in is-a hierarchies. Journal of Documentation **49**(2) (June 1993) 188–207
14. Rada, R., Mili, H., Bicknell, E., Blettner, M.: Development and application of a metric on semantic nets. IEEE Transactions on Systems, Man, and Cybernetics **19**(1) (January 1989) 17–30
15. Resnik, P.: Using information content to evaluate semantic similarity in a taxonomy. In: 14th international joint conference on Artificial intelligence - Volume 1, Morgan Kaufmann Publishers Inc. (1995) 448–453
16. Shavlik, J.W., ed.: An Information-Theoretic Definition of Similarity. In Shavlik, J.W., ed.: Proceedings of the 15th International Conference on Machine Learning (ICML 1998), Morgan Kaufmann (1998)
17. Gal, A.: Uncertain Schema Matching. Synthesis Lectures on Data Management. Morgan & Claypool Publishers (2011)
18. Weidlich, M., Mendling, J., Weske, M.: Efficient consistency measurement based on behavioral profiles of process models. IEEE Trans. Software Eng. **37**(3) (2011) 410–429
19. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning **62**(1-2) (2006) 107–136
20. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press (2009)
21. Getoor, L., Taskar, B.: Introduction to Statistical Relational Learning. MIT Press (2007)
22. Manning, C.D., Schütze, H.: Foundations of statistical natural language processing. MIT Press (1999)

23. Niepert, M., Meilicke, C., Stuckenschmidt, H.: A Probabilistic-Logical Framework for Ontology Matching. In: Proceedings of the 24th AAAI Conference on Artificial Intelligence. (2010)
24. Noessner, J., Niepert, M.: CODI: Combinatorial Optimization for Data Integration–Results for OAEI 2010. In: Proceedings of the 5th Workshop on Ontology Matching. (2010)
25. Riedel, S.: Improving the accuracy and efficiency of MAP inference for Markov logic. In: Proc. of UAI-08. (2008) 468–475
26. Miller, G.: Wordnet: A lexical database for english. Commun. ACM **38**(11) (1995) 39–41
27. Euzenat, J., Shvaiko, P.: Ontology matching. Springer (2007)
28. Doan, A., Halevy, A.Y.: Semantic integration research in the database community: A brief survey. AI Magazine **26**(1) (2005) 83–94
29. Noy, N.F.: Semantic integration: A survey of ontology-based approaches. SIGMOD Record **33**(4) (2004) 65–70
30. Giunchiglia, F., Shvaiko, P.: Semantic matching. The Knowledge Engineering Review Journal **18**(3) (2003) 265–280
31. Giunchiglia, F., Yatskevich, M., Shvaiko, P.: Semantic matching: Algorithms and implementation. J. Data Semantics **9** (2007) 1–38