Instance-Based Process Matching Using Event-Log Information

Han van der Aa[1(✉)], Avigdor Gal[2], Henrik Leopold[1], Hajo A. Reijers[1], Tomer Sagi[3], and Roee Shraga[2]

[1] Department of Computer Sciences, VU University Amsterdam, Amsterdam, The Netherlands
{j.h.vander.aa,h.leopold,h.a.reijers}@vu.nl

[2] Faculty of Industrial Engineering and Management, Technion – Israel Institute of Technology, Haifa, Israel
avigal@technion.ac.il, shraga89@tx.technion.ac.il

[3] Hewlett Packard Labs, Guttwirt Industrial Park, Technion City, Haifa, Israel
ts.tomersagi@gmail.com

Abstract. Process model matching provides the basis for many process analysis techniques such as inconsistency detection and process querying. The matching task refers to the automatic identification of correspondences between activities in two process models. Numerous techniques have been developed for this purpose, all share a focus on process-level information. In this paper we introduce *instance-based process matching*, which specifically focuses on information related to instances of a process. In particular, we introduce six similarity metrics that each use a different type of instance information stored in the event logs associated with processes. The proposed metrics can be used as standalone matching techniques or to complement existing process model matching techniques. A quantitative evaluation on real-world data demonstrates that the use of information from event logs is essential in identifying a considerable amount of correspondences.

Keywords: Process model matching · Event logs · Process similarity

1 Introduction and Motivation

Process models have been established as a means to design, analyze, and improve information systems [7]. The creation, utilization, and evolution of such models is supported by a manifold of concepts and techniques that offer, for instance, re-use driven modeling support, harmonization of model variants, model-based system validation, and effective management of model repositories. Many of these techniques share a reliance on the identification of correspondences between entities of different models, also termed *process model matching* [13]. The accuracy and, therefore, usefulness of techniques supporting the creation, utilization, and evolution of models is highly dependent on the correctness and completeness of the process model matching outcome.

In recent years, a plethora of works have addressed process model matching [1,3]. Growing alongside related fields such as *ontology alignment* and *schema matching* [15], process model matching offers innovation through the use of process-oriented information in the matching task. Existing process model matching techniques focus mainly on process information described by process models themselves. In this work we present the first matching technique that uses an important additional resource: *event logs*. Such logs offer valuable information on attributes, event durations, and other aspects that specifically relate to the *observed execution* of processes, rather than their *specification*. We propose and evaluate six new matching techniques that use event-log information and evaluate their contribution to the effective matching of processes. These techniques aim to identify correspondences that cannot be identified by just considering process model information.

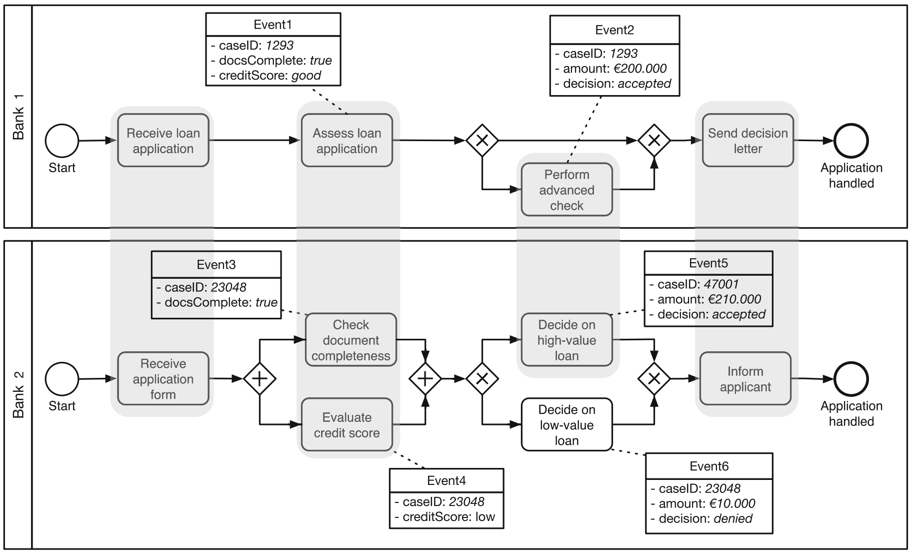

Fig. 1. Two process models and their correspondences

To illustrate the usefulness of event-log information for process model matching, consider two process models, M_1 and M_2, which depict two (simplified) processes to handle loan applications. Also consider their respective sets of activities $\mathcal{A}_1$ and $\mathcal{A}_2$. Figure 1 illustrates these models, M_1 at the top and M_2 at the bottom, and highlights their *correspondences*, i.e. the activities that represent similar behavior.

In process model matching, we wish to automatically identify these correspondences between $\mathcal{A}_1$ and $\mathcal{A}_2$. By analyzing the labels of the activity, some correspondences can be identified in a straightforward manner, such as the correspondence between `receive loan application` and `receive application form`. However, the label-based identification of other correspondences is not as straightforward, if at all possible. Consider the `assess loan application` activity in M_1 and the activities `check document completeness` and `evaluate credit score` from M_2. For the correspondences between these activities, there is no obvious syntactic or semantic relation for the contents of their labels. This makes it difficult to recognize their similarity based on textual analysis. However, the events associated with these activities provide valuable information about their similarity. `Event1` in M_1 includes attributes that describe the completeness of the filed documents and the credit score. Events `Event3` and `Event4` in M_2 are each also associated with one of these attributes. This similarity between event attributes provides a strong indication of relation between the activities, which could not be derived without considering event information.

In other cases, the names of attributes that are associated with events, by themselves, do not suffice to distinguish among potential correspondences. For example, the `decide on low-value loan` (a_m) and `decide on high-value loan` (a_n) activities from model M_2 both have events that contain an `amount` attribute. Therefore, the attribute names are not sufficient to determine which of these corresponds to the `perform advanced check` activity (a_o) in M_1. However, by analyzing the values associated with these attributes throughout an event log, this could be achieved. For instance, if events corresponding to a_n and a_o are always associated with amounts above €200.000, while a_m always has a lower amount, the correspondence between a_n and a_o can be asserted.

The main contribution of the paper is in the introduction of six conceptual notions of similarity between event classes. These *similarity notions* cover different aspects of process information stored in event logs, ranging from similarity in execution times to data-based similarity. We also discuss operationalizations of the similarity notions into *similarity measures*. In particular, we define one specific similarity measure for each of the six notions and reflect on alternative ways to operationalize them. We also offer a full-scale instance-based process model matching tool, which builds on an existing tool for schema matching.

The remainder of the paper is organized as follows. Section 2 introduces preliminary notions relevant to event logs and process model matching. Section 3 describes our proposed six similarity measures. The quantitative evaluation in Sect. 4 considers the performance of these individual similarity measures for matching, as well as their composition in matching ensembles. We discuss related work in Sect. 5 and conclude the paper in Sect. 6.

2 Preliminaries

This section introduces notions relevant to the matching techniques we present in this paper. In particular, we define event logs and process matching concepts.

An *event log* $\mathcal{L}$ comprises a set of *traces*, each representing an execution of a single process instance. Each trace $t = \langle e_1, \ldots, e_n \rangle \in \mathcal{L}$ consists of a sequence of events. We use $\mathcal{E}$ to denote the finite set of *event classes* that occur in a log. An occurrence of an event $e \in t$ for any trace $t \in \mathcal{L}$ corresponds to a specific event class, *i.e.*, $e \in E_i$ for $E_i \in \mathcal{E}$. For the purposes of this paper, we assume that for a process model M with an activity set $\mathcal{A}$ each activity $a \in \mathcal{A}$ corresponds to exactly one event class $E \in \mathcal{E}$ and vice versa. Therefore, without loss of generality, we shall refer to *activities* and *event classes* interchangeably.

We formally define *process model matching* based upon notions from [11]. For any pair of event class sets $\{\mathcal{E}_1, \mathcal{E}_2\}$, a matching task creates an $n \times n'$ *similarity matrix* $\mathcal{M}(\mathcal{E}_1, \mathcal{E}_2)$ over $\mathcal{E}_1 \times \mathcal{E}_2$. Each $M_{i,j}$ in the matrix represents a degree of similarity, usually a real number in $[0, 1]$, between the i-th event class in $\mathcal{E}_1$ and the j-th event class in $\mathcal{E}_2$. The matching task often consists of sequential steps in which different classes of matchers are applied. Here, it is important to distinguish between three classes of matchers: (i) first line matchers, (ii) ensemble matchers, and (iii) decision makers. A first line matcher (1LM) establishes a similarity matrix by directly analyzing sets of event classes, $\{\mathcal{E}_1, \mathcal{E}_2\}$. For any pair of event classes $E_1 \in \mathcal{E}_1$ and $E_2 \in \mathcal{E}_2$, each 1LM produces a score $[0, 1]$ that quantifies the similarity between E_1 and E_2 by comparing the instances of these classes according to a certain characteristic. Ensemble matchers and decision makers are both specific types of so-called *second line matchers* (2LMs). A 2LM establishes a similarity matrix from an input of one or more other similarity matrices. *Ensemble matchers* are 2LMs that combine the results of multiple 1LMs into a single similarity matrix, for example by computing a weighted average of the similarity matrices. Lastly, *decision makers* take a non-binary similarity matrix (with values in the range [0,1]), as created by a 1LM or an ensemble matcher, and convert it into a binary matrix (with values in $\{0, 1\}$). For example, if we know that each $E_1 \in \mathcal{E}_1$ corresponds to at most one event class in $\mathcal{E}_2$, a decision maker can be used to select the event class E_2 with the highest similarity scores for each E_1. We refer to this selected pair as a *correspondence* between E_1 and E_2.

3 Event-Class Similarity

This section describes how information contained in event logs can be utilized to identify correspondences among event classes. We describe six conceptual *notions* of similarity, which together provide a complete coverage of the prominent types of information contained in event logs: *ordering*, *frequencies*, *timestamps*, and *data attributes*. We consider one similarity notion for each of the first three types and, due to its versatility, three different similarity notions related to the *data attributes* associated with events. To illustrate the operationalization of these similarity notions, we introduce a corresponding similarity *measure* for each of them. Each measure produces a value in the range [0, 1], where a higher score indicates a stronger similarity. These measures can be used as 1LMs, where the similarity scores obtained by the measures are used to populate a similarity matrix. The measures that we introduce can be applied without imposing any assumptions on the data. Furthermore, we also reflect on alternative measures that typically depend on certain assumptions or are computationally more complex.

3.1 Positional Similarity

The underlying idea of positional similarity is that if two event classes occur at similar stages in the execution of a process, they are more likely to be similar. For example, the final event in M_1, `Send decision letter`, is more likely to be similar to `Inform applicant`, which occurs at the end of M_2, than to `Receive application form`, which occurs at the start of the process.

Similarity Measure. We define a *relative position* (RP) measure that quantifies the average position at which events of a certain event class occur in traces. To account for varying trace sizes, we consider the position of an event relative to the length of a trace. Specifically, we use p_e to denote the relative position of an event e in a trace, *e.g.*, for $t = \langle a, b, c \rangle$, $p_a = 1/3$, $p_b = 2/3$ and $p_c = 3/3$. Using $\bar{p}_E$ to denote the average p_e over all instances $e \in E$, Eq. 1 provides the RP measure.

$$RP(E_1, E_2) = 1 - |\bar{p}_{E_1} - \bar{p}_{E_2}| \tag{1}$$

Alternatives. Comparing the position of events in traces provides a basic measure for structural or behavioral similarity. Techniques for process model matching exist that use more advanced similarity measures, for example those base on *graph edit distance* or *behavioral relations* [4]. Such measures can also be adapted to work on graph structures or behavioral relations derived from event-log information. Such derivation is done by techniques that automatically derive process models from event logs, i.e. so-called *process discovery* techniques.

3.2 Occurrence Similarity

The frequency with which events of a certain event class occur can provide useful information regarding its similarity to other event classes. For example, if two event classes E_1 and E_2 each occur only rarely in an event log, then E_1 and E_2 both correspond to some exceptional action, hinting at their potential similarity. In the running example, for instance, it can be expected that the majority of loan requests will be for amounts below €200,000. This means that occurrences of the `perform advanced check` and `decide on high-value loan` are relatively rare. Therefore, comparing the frequencies with which event classes occur can be a useful similarity indicator. Furthermore, the consideration of frequencies can also be used to identify a lack of similarity, for instance between event classes that occur only once per trace and those that occur multiple times.

Similarity Measure. We define a measure $FREQ$ which compares the average number of occurrences of event classes per trace. We let $\bar{f}_E$ denote this average for an event class E, and use Eq. 2 to formalize $FREQ$. Because it is possible that $\bar{f}_E > 1$, this measure is normalized to ensure a confidence score in $[0, 1]$.

$$FREQ(E_1, E_2) = 1 - |\frac{|\bar{f}_{E_1} - \bar{f}_{E_2}|}{\max(\bar{f}_{E_1}, \bar{f}_{E_2})}| \tag{2}$$

Alternatives. An alternative way to evaluate occurrence similarity is to consider the fraction of traces in which an event class occurs, rather than the average number of occurrences per trace. Furthermore, it is possible to perform statistical tests rather than compare averages. We reflect on these tests in Sect. 3.3.

3.3 Duration Similarity

The time it takes to execute activities can serve as an indicator that provides useful hints regarding their similarity. In our running example it can be expected that activities that check loans with amounts over €200,000 are extensive and, therefore, consume a significant amount of time. By contrast, the communication of the decision to the applicant can very well be automated, resulting in negligible durations. Such a considerable difference in durations can be an important indicator for dissimilarity.

Similarity Measure. A straightforward similarity measure for durations can be obtained by comparing the average durations of an event class in a log. Using $\bar{d}_E$ to denote the average duration of events of class E, Eq. 3 provides a normalized measure that returns a score in $[0, 1]$.

$$DUR(E_1, E_2) = 1 - |\frac{|\bar{d}_{E_1} - \bar{d}_{E_2}|}{\max(\bar{d}_{E_1}, \bar{d}_{E_2})}| \tag{3}$$

Alternatives. Durations can vary significantly among occurrences of the same event class. An alternative is to use statistical tests, *e.g.*, the *t-test* or *Kolmogorov-Smirnov test* [16] to compare the statistical distribution of the durations for two event classes. To apply a statistical test certain preconditions have to be met. For example, the t-test requires data to be normally distributed. Another consideration to take into account is the cost of computing the similarity. For instance, the Kolmogorov-Smirnov test is computationally intensive, which can negatively affect its applicability to matching problems.

3.4 Attribute Name Similarity

The names of attributes provide insights into the data values used or created by events. These attribute names can be useful similarity indicators to identify correspondences. Their importance is demonstrated in the motivational scenario, where event classes that produce the same attributes (e.g. the `docsComplete` attribute) are recognized to be similar to each other.

Similarity Measure. We define an attribute name similarity measure $ATTR$, which determines the level of overlap in attribute names among the attribute sets associated with two event classes. To quantify this overlap, we adapt the well-known *inverse document frequency* (idf) and *cosine similarity* measures from the field of information retrieval [18]. The idf assigns weights to the occurrence of attribute names based on how common they are in a particular context, i.e. in a process. The underlying idea is that unique attributes, such as `docsComplete` in the motivational scenario, provide better indicators of similarity than common attributes. To compute the cosine similarity measure, we convert the attribute sets of event classes into weighted vector-based representations, denoted as $\mathbf{A_E}$. The weights in these vectors reflect the idf-score associated with a given attribute.

$$ATTR(E_1, E_2) = \frac{\mathbf{A_{E_1}} \cdot \mathbf{A_{E_2}}}{\parallel \mathbf{A_{E_1}} \parallel \parallel \mathbf{A_{E_2}} \parallel} \tag{4}$$

Alternatives. The ATTR measure only considers overlap in attributes with identical names. Numerous techniques exist that can be used to also quantify the similarity between non-identical attribute names [10]. Commonly applied measures include the Levenshtein distance [22] for *syntactic* similarity, which can be used to compute the string edit distance between attribute names. *Semantic* similarity measures can be used to recognize names with similar meanings, for instance those that use synonymous terms. The most commonly used tool to quantify semantic similarity is WordNet [2].

3.5 Attribute Value Similarity

The values of an attribute, associated with events of a given event class may provide insights into similarity beyond attribute name similarity. We have identified two general scenarios for this. First, an analysis of values can be useful to determine similarity in the context of opaque or unrelated attribute names. For instance, it is difficult to relate two attributes `month` and `m` based on their labels. By contrast, if both attributes are associated with numeric values in the range 1–12 (or even month names), their similarity becomes more apparent. Second, attribute value similarity can be used to disambiguate event classes that use the same attributes. The motivational scenario provides an example of this. The event classes `decide on high-value loan` and `decide on low-value loan` in M_2 both consider an `amount` attribute. Events of the former class are associated with a higher range of values than events of the latter. Therefore, by considering the attribute values, we can identify that the former event class is more likely to correspond to `perform advanced check` in M_1, which similarly occurs only for loan requests with a high amount.

Similarity Measure. To quantify attribute value similarity for two *individual* attributes, we rely on techniques from the research area of schema matching [8], where *content-based* matching, (direct comparison of sets of attribute values) is combined with *constraint-based* matching. The latter aims to extract constraints from a set of values, such as upper and lower bounds for numerical values. For brevity, we refrain from presenting explicitly the equations used in this method. After considering the similarity of individual attributes, the similarity values obtained in this manner can be used as weights to calculate the cosine similarity between two attribute sets. Using $VAL(A_E)$ to refer to the value sets of the attributes of an event class E, we compute the VAL measure as given by Eq. 5.

$$VAL(E_1, E_2) = \frac{\mathbf{VAL(A_{E_1})} \cdot \mathbf{VAL(A_{E_2})}}{\parallel \mathbf{VAL(A_{E_1})} \parallel \parallel \mathbf{VAL(A_{E_2})} \parallel} \tag{5}$$

Alternatives. Various alternative techniques exist to determine the similarity between two individual attributes, including identifying and comparing data types such as zip codes or geographical names, putting constraints on values, and identifying data patterns and distributions (cf. [15]).

3.6 Prerequisites Similarity

The input data used by an event can be an important indicator of event class similarity. Intuitively, this builds on the idea that the more similar the data that is used by event classes, the more similar their purpose. For example, in the motivational scenario, events of the classes `send decision letter` in M_1 and `inform applicant` in M_2 are the only ones to occur *after* an event has produced a value for the `decision` attribute. A challenge here is that the *XES-standard*[1] for event logs does not have an explicit notion of input data. Therefore, an event log can contain information on input data in two ways. First, input data elements of an event e might be part of the attribute set of e, as seen for the `amount` attribute of the `perform advanced check` event. In this case, similarity of inputs is already covered by the aforementioned attribute name and value similarity measures $ATTR$ and VAL. However, input data can also be derived from data attributes that were created *prior* to the execution of an event, which we operationalize next.

Similarity Measure. We define a measure $PREQ$ that determines the similarity of prerequisites based on the attributes associated with prior events. Specifically, given an event e_i that occurs at position i in a trace t, we define P_{e_i} as the union of all attribute sets A_{e_j} for $0 < j < i$. P_E then denotes all attributes contained in a set P_e for $e \in E$. The similarity between two prerequisites sets P_{E_1} and P_{E_2} is then computed in a similar manner as the $ATTR$ measure.

$$PREQ(E_1, E_2) = \frac{\mathbf{P_{E_1}} \cdot \mathbf{P_{E_2}}}{\parallel \mathbf{P_{E_1}} \parallel \parallel \mathbf{P_{E_2}} \parallel} \tag{6}$$

Alternatives. It is possible to consider the values of prerequisite attributes, rather than their names, as provided by the VAL measure, or by combining the two. Furthermore, alternative measures can consider two more factors in the similarity computation, namely frequency and proximity of prerequisite attributes. The *frequency* with which an attribute is created prior to the execution of an event $e \in E$ can be used to distinguish among mandatory and optional prerequisites. In the context of process matching, such a distinction was proposed by Sagi et al. [17]. The *proximity* between the creation of an attribute and an occurrence of an event can provide insights into their similarity. Intuitively, if an attribute is created by an event at index i in a trace, then this attribute is

[1] http://www.xes-standard.org/.

more likely to be a relevant prerequisite to its immediate sequel event e_{i+1} than it is to events that are further away. Frequency and proximity considerations can be integrated by adapting the weights of the elements of the vectors used by $PREQ$ accordingly.

4 Empirical Evaluation

This section presents an empirical evaluation that demonstrates the usefulness of event-log information for process matching. We evaluate the performance of the proposed event log-based matchers as a standalone tool. Specifically, we compare the correspondences obtained by automatic matching based on our 1LMs to a *gold standard* that contains the true correspondences between event classes. Our evaluation is based on real-world data, using a test collection of 105 event log pairs.

4.1 Test Collection

To perform the evaluation, we use data from the *BPI Challenge 2015* [5], which consists of real-world event data related to the handling of construction permit applications by five Dutch municipalities. The event data describe similar processes, while their actual implementation differs considerably. To obtain a sufficiently large collection of event logs to match, we split the event data into event logs, each relating to a different subprocess (on average 17 subprocesses per municipality). After removing the logs that contain less than five event classes (to avoid trivial matching tasks), we obtain a total of 57 event logs. We create pairs of event logs that relate to the same subprocess from different municipalities. This results in a total of 105 event log pairs.

Table 1. Characteristics of the test collection

Measure	Traces	Event classes	Total corr.	True corr.	Log overlap
Average	487.0	33.0	2,533.4	30.9	87.7%
Std.dev	353.6	40.7	6,246.3	36.5	10.5%
Minimum	8	5	15	3	50.0%
Maximum	1409	172	26,832	156	100.0%

Table 1 provides an overview of the test collection. The table illustrates the great diversity between the subprocesses. This can, for example, be seen in the number of event classes per log, which ranges from 5 to 172. The *true correspondences* reflect the actual correspondences between event classes from a pair of logs, also referred to as the *gold standard*. This gold standard directly follows from the traceability between the event classes in the logs of the different municipalities. The last column in the table describes the overlap in terms of the event classes of a log pair, *i.e.*, the fraction of event classes that appear in both logs. This measure indicates that, on average, 88% of the event classes in a log also appear in the gold standard. In the most extreme case, only 50% of the event classes from a log pair correspond to each other. Table 1 highlights the fact that even though the processes are similar across the five municipalities, considerable differences exist as well. The choice for this data collection is, furthermore, motivated by the lack of event logs associated with the collections typically used to evaluate matchers, i.e. the collections of the Process Model Matching Contests [1,3].

Note that in order to provide objective evaluation results, we hide all references to the names of event classes in this test collection. In particular, we hide the names and values of the following attributes: `concept:name`, `action_code`, `activityNameEN`, and `activityNameNL`.

4.2 Setup

To conduct the evaluation, we used the Ontobuilder Research Environment (ORE), an open source schema matching tool that enables researchers to run and evaluate matching experiments. We implemented the six 1LMs (Sect. 3) in ORE and made their implementation publicly available as part of the tool.[2]

As described in Sect. 2, establishing (exact) correspondences between the event class sets $\mathcal{E}_1, \mathcal{E}_2$ of a log pair requires a similarity matrix $\mathcal{M}(\mathcal{E}_1, \mathcal{E}_2)$ and a decision maker. Here, we obtain the similarity matrices in two different manners, resulting in a two-part evaluation. In the first part, we use each of the six 1LMs separately to construct $\mathcal{M}(\mathcal{E}_1, \mathcal{E}_2)$ based on a distinct similarity measure. This part of the evaluation provides insights into the performance of the *individual* 1LMs and into the characteristics of the test collection. In the second part, we use an ensemble matcher that combines the scores of the six similarity matrices into a single matrix. By evaluating this *matching ensemble*, we obtain insights into the combined performance of the matchers and their complementary nature. We further reflect on the way in which the measures complement each other by computing correlations among the individual similarity scores.

After obtaining a similarity matrix $\mathcal{M}(\mathcal{E}_1, \mathcal{E}_2)$ we apply a decision maker on $\mathcal{M}(\mathcal{E}_1, \mathcal{E}_2)$ to obtain a set of exact correspondences, to which we will refer to as $\mathcal{C}(\mathcal{E}_1, \mathcal{E}_2)$. In particular, we apply the *maximum weighted bipartite graph match* (MWBM) [12] to establish $\mathcal{C}(\mathcal{E}_1, \mathcal{E}_2)$. This decision maker is particularly well-suited in the context of the test collection, because it establishes 1:1 correspondences between event classes.

We use the well-known precision, recall, and F1 measures to compare the automatically obtained set of correspondences $\mathcal{C}$ to the set $\mathcal{G}$ of *actual* correspondences included in the gold standard. *Precision* (*pre*) here reflects the fraction of the correspondences obtained by the matching techniques that is also included in the gold standard, whereas recall (*rec*) represents the fraction of the correspondences in the gold standard that is correctly identi-

[2] https://bitbucket.org/tomers77/ontobuilder-research-environment.

fied by the matchers. The *F1* measure represents the *harmonic mean* of precision and recall. Equations 7, 8 and 9 formally define these measures.

$$pre = \frac{\mathcal{C} \cap \mathcal{G}}{\mathcal{C}} \quad (7) \qquad rec = \frac{\mathcal{C} \cap \mathcal{G}}{\mathcal{G}} \quad (8) \qquad F1 = \frac{2 * pre * rec}{pre + rec} \quad (9)$$

4.3 Results

Table 2 presents an overview of the results obtained by using the individual 1LMs and by a matching ensemble based on all six 1LMs. We will now elaborate on the results obtained through these two methods.

Table 2. Overview of the evaluation results

FLM	Precision	Recall	F1-score
RP	.24	.25	.25
FREQ	.14	.14	.14
DUR	.13	.11	.12
ATTR	.05	.04	.04
VAL	**.27**	**.27**	**.27**
PREQ	.09	.08	.08
Ensemble	**.38**	**.38**	**.38**

Matching Results. The results presented in Table 2 show that the performance varies greatly among the various 1LMs. The lowest performance results belong to *ATTR* and *PREQ* 1LMs, which both consider similarity based on *attribute names*. These 1LMs achieve $F1$-score of .04 and .08, respectively. A post-hoc analysis of the similarity matrices generated by these 1LMs shows that, indeed, attribute names provide little discriminatory power in the context of this particular test collection. In fact, most event classes are associated with identical or nearly identical sets of attributes, which results in a similarity score of 1.0 for the vast majority of event class pairs. By contrast, *VAL* achieves the highest results with an $F1$-score of 0.27. This shows that, as opposed to the names of attributes, attribute *values* provide a substantially better indicator of similarity. Furthermore, the performance of *RP* shows that the consideration of positional similarity also provides a relatively good indicator of similarity.

The last row of Table 2 presents the results obtained by an ensemble consisting of the six 1LMs. For this ensemble, we applied a naïve weighting scheme, in which we computed the average score of the six similarity measures. The results demonstrate that the ensemble greatly outperforms individual 1LMs, achieving an $F1$-score of .38. A one-sided paired t-test reveals that this result is statistically significant ($p < 0.05$) when compared to the best performing individual

1LM (*VAL*). The improved results of the ensemble illustrate that the six 1LMs are complementary to each other and can enhance each other's performance.

Top-*k* Results. The results shown so far indicate that the use of log data for process matching is a valid approach that can identify correspondences among activities by analyzing execution data. It is also clear that the use of log data alone does not suffice for achieving an industrial-strength matching tool. An $F1$-score of about 0.4 indicates a far from random correlation between the decisions made by the ensemble and the true correspondences. Still, it requires the support of other techniques to strengthen its performance. Existing process model-based matchers represent good candidates, because they use valuable process model information (*e.g.*, activity labels), which is purposefully not used by our log-based matchers. Because numerous model-based matchers exist, each with their own strengths and weaknesses, we leave for future research the best way to tackle the combination of log-based and model-based matching techniques. Here, we investigate the obtained results in more depth and determine to what extent the log-based techniques lend themselves well to process matching.

Identified correspondences can be incorrect because often an event class has multiple correspondences with equal or near equal similarity scores as the best candidates. The selection of a single, best correspondence then becomes an arbitrary selection among a handful of correspondences. This problem relates to the inherent issue of uncertainty in the matching task. Works on *matching monotonicity* [9] have found that this uncertainty prevents matchers from identifying a correct correspondence as the one with the highest similarity measure. However, these works argue that *good* matchers should contain the *correct correspondences* among the correspondences with the highest similarity scores, *i.e.*, in the so-called top-*k* matches. If they succeed in this, a good matcher positions a true correspondence high enough for a human observer to confirm it after scanning only a few possible correspondences. To test this, we check for each event class whether its correct correspondence occurs within the top 3 or top 5 correspondences with the highest similarity scores.

Figure 2 presents results of the top-*k* analysis. For each matcher, we measure the recall of top-1, top-3, and top-5. As expected, the matching result improves significantly when the best correspondences are considered. This holds for all matchers, but with varying levels of success. The biggest gain is observed for the *RP* measure. There, the performance increases from a recall of 0.25 for top-1 to 0.69 and 0.78 for top-3 and top-5, respectively. As such, the *RP* measure performs nearly identically to the matching ensemble. The results indicate that all matchers show a monotonic behavior, though some more than others. It is interesting to see that while the top-1 performance of *RP* is worse than the one of VAL, a different picture is drawn for top-3 and top-5. There, *RP* surpasses VAL (in terms of Recall), performing just as well as the ensemble matcher. Finally, what is important to realize is that by considering the top-3 and top-5 scores users need to just evaluate approximately 4% and 7% of the total possible correspondences. These small fractions already enable the respective identification of close to 70% and 80% of the true correspondences.

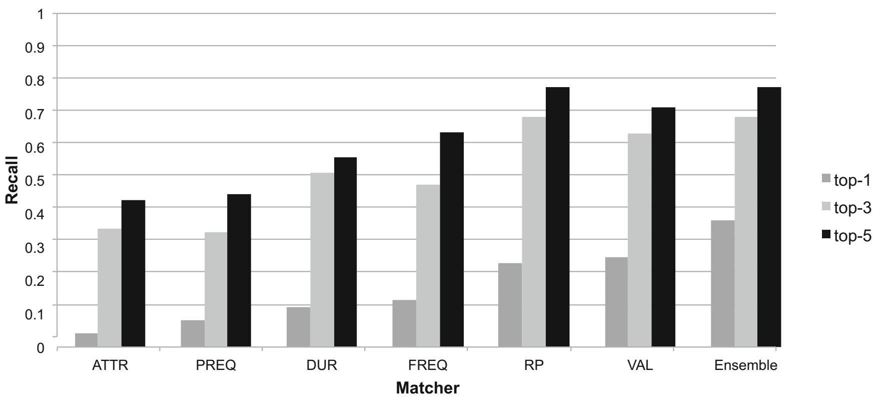

Fig. 2. Recall scores for top-k results

5 Related Work

The work presented in this paper relates to two main streams of research, namely process model matching and instance-based matching.

In the last few years, a plethora of *process model matching* approaches has been proposed [1,3]. Traditionally, they combine structural or behavioral properties with different types of textual similarity. Some rely on rather simplistic techniques such as the Levenshtein distance [20], others use WordNet for computing textual similarity [14]. Recognizing the limitations of many existing matchers in terms of performance, researchers recently started to explore alternative strategies. For instance, Klinkmüller et al. [13] improve matching results by incorporating user feedback. Weidlich et al. [21] used prediction techniques to select the most suitable matching technique for a given problem. In this work, we propose a new resource, event log data, to improve the matching results. Our experiments demonstrate that this indeed represents a promising direction.

Instance-based matching has been previously explored in the context of schema matching and the related field of ontology alignment. Engmann and Maßmann [8] used two methods to enhance their COMA++ matcher. The first, a constraint-based matcher, identifies the field types using a list of patterns and numerical constraints attempted over the instance data per attribute. An approach similar to our own *VAL* measure. Their second method applies to text-based fields taken from the same domain in which the string-similarity of all instances is compared and averaged. A similar approach is suggested by Wang et al. [19] which probe a Web query interfaces with keywords and then compare the vector-space similarity of the query result tokens. A similar approach is applied by Duan et al. [6], using *Locality Sensitive Hashing* (LSH) techniques to compare instances over very large ontologies. Zaiß et al. [23] use regular expressions to improve pattern identification of attribute domains. Our work differs from these works in that we make use of process-unique features to perform the matching task.

6 Conclusion

In this work we proposed instance-based process matching as a new element for the toolbox of matching process models. We introduced six 1LMs that assess the similarity of two event classes from different event logs. Each 1LM focuses on a different conceptual notion of similarity, resulting in a broad coverage of the process information stored in event logs. We demonstrated the usefulness of these similarity metrics through a quantitative evaluation using real-world data. The evaluation showed that by just considering the information specific to event logs, the introduced matchers can identify a considerable number of correspondences between event classes.

In future work, we set out to provide and test further operationalizations of the similarity concepts considered in this paper. Currently, we defined a single similarity metric for each of the six concepts. However, the majority of these concepts can be operationalized by implementing a variety of metrics, as discussed in Sect. 3. Furthermore, we strive to combine our event log-based matching techniques with traditional, model-based techniques for process model matching. By combining model-based matchers with the proposed log-based matchers, we will aim at achieving matching results that cannot be obtained by using either of these techniques alone.

References

1. Antunes, G., Bakhshandeh, M., Borbinha, J., Cardoso, J., Dadashnia, S., Francescomarino, C., Dragoni, M., Fettke, P., Gal, A., Ghidini, C., et al.: The process model matching contest 2015. In: 6th EMISA Workshop, pp. 127–155 (2015)
2. Budanitsky, A., Hirst, G.: Evaluating wordnet-based measures of lexical semantic relatedness. Comput. Linguist. **32**(1), 13–47 (2006)
3. Cayoglu, U., Dijkman, R.M., Dumas, M., Fettke, P., Garcıa-Banuelos, L., Hake, P., Klinkmüller, C., Leopold, H., Ludwig, A., Loos, P., et al.: The process model matching contest 2013. In: 4th International Workshop on Process Model Collections: Management and Reuse (PMC-MR 2013) (2013)
4. Dijkman, R.M., Dumas, M., Van Dongen, B., Käärik, R., Mendling, J.: Similarity of business process models: metrics and evaluation. Inf. Syst. **36**(2), 498–516 (2011)
5. Dongen, B.F.V.: BPI challenge 2015 (2015). https://doi.org/10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1
6. Duan, S., Fokoue, A., Hassanzadeh, O., Kementsietsidis, A., Srinivas, K., Ward, M.J.: Instance-based matching of large ontologies using locality-sensitive hashing. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012. LNCS, vol. 7649, pp. 49–64. Springer, Heidelberg (2012). doi:10.1007/978-3-642-35176-1_4
7. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer, Heidelberg (2012)
8. Engmann, D., Maßmann, S.: Instance matching with COMA++. In: BTW Workshops, pp. 28–37 (2007)
9. Gal, A., Anaby-Tavor, A., Trombetta, A., Montesi, D.: A framework for modeling and evaluating automatic semantic reconciliation. VLDB J. Int. J. Very Large Data Bases **14**(1), 50–67 (2005)

10. Gal, A., Weidlich, M.: Model matching - processes and beyond. In: Zdravkovic, J., Kirikova, M., Johannesson, P. (eds.) CAiSE 2015. LNCS, vol. 9097, pp. 525–526. Springer, Cham (2015). https://link.springer.com/book/10.1007/978-3-319-19069-3
11. Gal, A.: Uncertain schema matching. Synth. Lect. Data Manag. **3**(1), 1–97 (2011)
12. Galil, Z., Micali, S., Gabow, H.: An o(EV logV) algorithm for finding a maximal weighted matching in general graphs. SIAM J. Comput. **15**(1), 120–130 (1986)
13. Klinkmüller, C., Leopold, H., Weber, I., Mendling, J., Ludwig, A.: Listen to me: improving process model matching through user feedback. In: Sadiq, S., Soffer, P., Völzer, H. (eds.) BPM 2014. LNCS, vol. 8659, pp. 84–100. Springer, Cham (2014). doi:10.1007/978-3-319-10172-9_6
14. Leopold, H., Niepert, M., Weidlich, M., Mendling, J., Dijkman, R., Stuckenschmidt, H.: Probabilistic optimization of semantic process model matching. In: Barros, A., Gal, A., Kindler, E. (eds.) BPM 2012. LNCS, vol. 7481, pp. 319–334. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32885-5_25
15. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB J. **10**(4), 334–350 (2001)
16. Razali, N.M., Wah, Y.B., et al.: Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. J. Stat. Model. Anal. **2**(1), 21–33 (2011)
17. Sagi, T., Gal, A., Weidlich, M.: Measuring expected integration effort in service composition. In: 2014 IEEE International Conference on Services Computing (SCC), pp. 645–652. IEEE (2014)
18. Salton, G., McGill, M.J.: Introduction to modern information retrieval (1986)
19. Wang, J., Wen, J., Lochovsky, F., Ma, W.: Instance-based schema matching for web databases by domain-specific query probing. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases, vol. 30, pp. 408–419. VLDB Endowment (2004)
20. Weidlich, M., Dijkman, R., Mendling, J.: The ICoP framework: identification of correspondences between process models. In: Pernici, B. (ed.) CAiSE 2010. LNCS, vol. 6051, pp. 483–498. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13094-6_37
21. Weidlich, M., Sagi, T., Leopold, H., Gal, A., Mendling, J.: Predicting the quality of process model matching. In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 203–210. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40176-3_16
22. Yujian, L., Bo, L.: A normalized levenshtein distance metric. IEEE Trans. Pattern Anal. Mach. Intell. **29**(6), 1091–1095 (2007)
23. Zaiß, K., Schlüter, T., Conrad, S.: Instance-based ontology matching using regular expressions. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2008. LNCS, vol. 5333, pp. 40–41. Springer, Heidelberg (2008). doi:10.1007/978-3-540-88875-8_19