Report: The Process Model Matching Contest 2013

Ugur Cayoglu[1](✉), Remco Dijkman[2], Marlon Dumas[3], Peter Fettke[4,5], Luciano García-Bañuelos[3], Philip Hake[4,5], Christopher Klinkmüller[6], Henrik Leopold[7], André Ludwig[6], Peter Loos[4,5], Jan Mendling[8], Andreas Oberweis[1], Andreas Schoknecht[1], Eitam Sheetrit[9], Tom Thaler[4,5], Meike Ullrich[1], Ingo Weber[10,11], and Matthias Weidlich[9]

[1] Institute of AppliedInformatics and Formal Description Methods (AIFB), Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
`ugur.cayoglu@kit.edu`

[2] Eindhoven University of Technology, Eindhoven, The Netherlands
`r.m.dijkman@tue.nl`

[3] University of Tartu, Tartu, Estonia
{`marlon.dumas,luciano.garcia`}`@ut.ee`

[4] Institute for Information Systems (IWi), DFKI, Saarbrücken, Germany
{`Tom.Thaler,Philip.Hake,Peter.Fettke,Peter.Loos`}`@iwi.dfki.de`

[5] Saarland University, Saarbrücken, Germany

[6] Information Systems Institute, University of Leipzig, Leipzig, Germany
{`klinkmueller,ludwig`}`@wifa.uni-leipzig.de`

[7] Humboldt-Universität zu Berlin, Arcata, Germany
`henrik.leopold@wiwi.hu-berlin.de`

[8] Wirtschaftsuniversität Wien, Vienna, Austria
`jan.mendling@wu.ac.at`

[9] Technion - Israel Institute of Technology, Haifa, Israel
{`eitams,weidlich`}`@tx.technion.ac.il`

[10] Software Systems Research Group, NICTA, Sydney, Australia
`ingo.weber@nicta.com.au`

[11] School of Computer Science & Engineering, University of New South Wales, Kensington, Australia

Abstract. Process model matching refers to the creation of correspondences between activities of process models. Applications of process model matching are manifold, reaching from model validation over harmonization of process variants to effective management of process model collections. Recently, this demand led to the development of different techniques for process model matching. Yet, these techniques are heuristics and, thus, their results are inherently uncertain and need to be evaluated on a common basis. Currently, however, the BPM community lacks established data sets and frameworks for evaluation. The Process Model Matching Contest 2013 aimed at addressing the need for effective evaluation by defining process model matching problems over published data sets.

This paper summarizes the setup and the results of the contest. Besides a description of the contest matching problems, the paper comprises short descriptions of all matching techniques that have been

DOI: 10.1007/978-3-319-06257-0_35, © Springer International Publishing Switzerland 2014

submitted for participation. In addition, we present and discuss the evaluation results and outline directions for future work in this field of research

Keywords: Process matching · Model alignment · Contest · Matching evaluation

1 Introduction

Business process models allow for managing the lifecycle of a business process, from its identification over its analysis, design, implementation, and monitoring [1]. A process model captures the activities of a business process along with their execution dependencies. Process model matching is concerned with supporting the creation of an alignment between process models, i.e., the identification of correspondences between their activities.

In recent years, many techniques building on process model matching have been proposed. Examples include techniques for the validation of a technical implementation of a business process against a business-centered specification model [2], delta-analysis of process implementations and a reference model [3], harmonization of process variants [4,5], process model search [6–8], and clone detection [9]. Inspired by the field of schema matching and ontology alignment, cf., [10,11], this demand led to the development of different techniques for process model matching. Yet, these techniques are heuristics and, thus, their results are inherently uncertain and need to be evaluated on a common basis. Currently, the BPM community lacks established data sets and frameworks for evaluation.

In this paper, we report on the setup and results of the Process Model Matching Contest 2013. It was organized as part of the 4th International Workshop on Process Model Collections: Management and Reuse (PMC-RM 13) that took place on August 26, 2013, at the 11th International Conference on Business Process Management in Beijing, China. The Contest Co-Chairs were Henrik Leopold and Matthias Weidlich.

The Process Model Matching Contest (PMMC) 2013 addresses the need for effective evaluation of process model matching techniques. The main goal of the PMMC is the comparative analysis of the results of different techniques. By doing so, it further aims at providing an angle to assess strengths and weaknesses of particular techniques and at outlining directions for improving process model matching. Inspired by the Ontology Alignment Evaluation Initiative (OAEI)[1], the PMMC was organized as a controlled, experimental evaluation. Two process model matching problems were defined and published with respective data sets. Then, participants were asked to send in their result files with the identified correspondences along with a short description of the matching technique. The evaluation of these results was conducted by the Contest Co-Chairs.

There have been seven submission to the contest covering diverse techniques for addressing the problem of process model matching. All submissions provided

[1] http://oaei.ontologymatching.org

reasonable results and could, therefore, be included in the evaluation and this paper. For each submitted matching technique, this paper contains an overview of the matching approach, details on the specific techniques applied, and pointers to related implementations and evaluations.

We are glad that the contest attracted interest and submissions from a variety of research groups. We would like to thank all of them for their participation.

The remainder of this paper is structured as follows. The next section gives details on the process model matching problems of the PMMC 2013. Section 3 features the short descriptions of the submitted matching approaches. Section 4 presents the evaluation results. Based on these results, Sect. 5 outlines directions for future work in process model matching before Sect. 6 concludes the paper.

2 Data Sets

The contest includes two sets of process model matching problems:

- **University Admission Processes (UA):** This set contains process models representing the admission processes of nine German universities. All models contain English text only. The models have been created by different modelers using varying terminology and capturing activities at different levels of granularity. All models are available as Petri-nets in the PNML format and shall be matched pairwise. Further, for eight out of the 36 model pairs, we also provide a gold standard alignment for initial evaluation.
- **Birth Registration Processes (BR):** This set comprises nine models of birth registration processes in Germany, Russia, South Africa, and the Netherlands. Four models were created by graduate students at the HU Berlin and five of the models stem from a process analysis in Dutch municipalities. Again, all models contain only English text, are available as Petri-nets in the PNML format, and shall be matched pairwise to obtain 36 alignments.

Table 1 gives an overview of the main characteristics of the two data sets. In addition to the minimum, maximum, and average number of labeled transitions per model, it shows the total and average number of simple and complex

Table 1. Characteristics of test data sets

Characteristic	UA	BR
No. of labeled Transitions (min)	11	9
No. of labeled Transitions (max)	44	25
No. of labeled Transitions (avg)	22	17.9
No. of 1:1 Correspondences (total)	345	348
No. of 1:1 Correspondences (avg)	9.6	9.7
No. of 1:n Correspondences (total)	83	171
No. of 1:n Correspondences (avg)	2.3	4.75

correspondences. From the numbers, we can learn that both model sets particularly differ with regard to the number of complex correspondences. While the admission models only contain an average of 2.3 complex correspondences per model, the birth certificate models contain 4.75. Consequently, we expect the birth certificate set to represent the more challenging sample.

3 Matching Approaches

In this section, we give an overview of the participating process model matching approaches. In total, seven matching techniques participated in the process model matching contest. Table 2 gives an overview of the participating approaches and the respective authors. In the following subsections, we provide a brief technical overview of each matching approach.

Table 2. Overview of participating approaches

No.	Approach	Authors
1	Triple-S: A Matching Approach for Petri Nets on Syntactic, Semantic and Structural Level	Cayoglu, Oberweis, Schoknecht, Ullrich
2	Business Process Graph Matching	Dijkman, Dumas, García-Bañuelos
3	RefMod-Mine/NSCM - N-Ary Semantic Cluster Matching	Thaler, Hake, Fettke, Loos
4	RefMod-Mine/ESGM - Extended Semantic Greedy Matching	Hake, Thaler, Fettke, Loos
5	Bag-of-Words Similarity with Label Pruning	Klinkmüller, Weber, Mendling, Leopold, Ludwig
6	PMLM - Process Matching Using Positional Language Models	Weidlich, Sheetrit
7	The ICoP Framework: Identification of Correspondences between Process Models	Weidlich, Dijkman, Mendling

3.1 Triple-S: A Matching Approach for Petri Nets on Syntactic, Semantic and Structural Level

Overview. So far, a handful contributions have been made to the problem of process model matching. The Triple-S matching approach adheres to the KISS principle by avoiding complexity and *keeping it simple and stupid*. It combines similarity scores of independent levels as basis for a well-founded decision about matching transition pairs of different process models. The following three levels and scores are considered:

- **Syntactic level -** $SIM_{syn}(a, b)$: For the syntactic analysis of transition labels we perform two preprocessing steps: (1) tokenization and (2) stop word elimination. The actual analysis is based on the calculation of Levenshtein distances between each combination of tokens (i.e. words) from the labels of transitions a and b. The final syntactic score is the minimum distance over all tokens divided by the number of tokens, i.e. the minimum average distance between each token.
- **Semantic level -** $SIM_{sem}(a, b)$: Prior to analysis, we perform the same preprocessing steps as above mentioned. Subsequently, we apply the approach of Wu & Palmer [12] to calculate the semantic similarity between each token of labels of transitions a and b based on path length between the corresponding concepts. The final semantic score is the maximum average similarity, i.e. it is calculated in an analogous manner to the final syntactic score.
- **Structural level -** $SIM_{sem}(a, b)$: Here, we investigate the similarity of transitions a and b through a comparison of (i) the ratio of their in- and outgoing arcs and (ii) their relative position in the complete net.

These three scores are combined to the final score $SIM_{total}(a, b)$ which represents the matching degree between two transitions a and b from different process models. It is calculated according to the following formula:

$$SIM_{total}(a, b) = \omega_1 * SIM_{syn}(a, b) + \omega_2 * SIM_{sem}(a, b) + \omega_3 * SIM_{struc}(a, b)$$

The three parameters ω_1, ω_2 and ω_3 define the weight of each similarity level. A threshold value θ is used to determine whether transitions actually match, i.e. iff $SIM_{total} \geq \theta$, two transitions positively match.

Specific Techniques. Compared to [13], the Triple-S approach makes several adjustments. Firstly, stop words are eliminated and the Levenshtein distance is calculated on the level of single tokens instead of complete sentences. Secondly, for the semantic level an established NLP approach is introduced. Finally, on the structural level TripleS performs contextual analysis by investigating local similarity only.

Implementation. The Triple-S approach has been implemented using Java. For the calculation of the semantic score with the approach of Wu & Palmer, the *WS4J Java API*[2] has been used to query Princeton's English *WordNet* 3.0 lexical database [14]. Relative positions of transitions are calculated using the implementation of Dijkstra's algorithm by Vogella[3]. The code can be obtained from http://code.google.com/p/bpmodelmatching/wiki/Download?tm=4 under *GNU GPL v3* license.

[2] https://code.google.com/p/ws4j/

[3] http://www.vogella.com/articles/JavaAlgorithmsDijkstra/article.html

Evaluations. During our experiments we tried to approximate optimal results based on the gold standard examples. For the contest, we have used the following values: $\omega_1 = 0.45$, $\omega_2 = 0.3$, $\omega_3 = 0.25$ and $\theta = 0.6$. The Triple-S approach is currently developed as part of the ongoing SemReuse research project addressing business process model reuse. This contest on business process similarity presents a welcome possibility for first experiments. We are planning on refining the current measures for the individual levels, especially the semantic and structural level and improved detection of 1:n matches.

3.2 Business Process Graph Matching

Overview. Business process graph matching works by considering a business process as a labeled graph, wherein nodes correspond to tasks, events or gateways, and edges capture the flow of control between nodes in the process. Nodes are generally assumed to have a label, although gateways may be unlabeled.

Graph matching aims at computing a mapping between the nodes in the input graphs. In its most common form, the mapping relates one node in a graph to at most one node in the another graph (partial inductive mapping). The mapping induces a distance between the two graphs, which is usually calculated by adding the following components:

– the number of inserted nodes: nodes that appear in one graph, but not in the other (i.e.: nodes that are not part of the mapping);
– the sum of the distances between nodes that *are* part of the mapping based on their labels (e.g.: the nodes labeled 'receive request' and 'receiving request' are closer than the nodes labeled 'receive request' and 'reject request'); and
– the number of inserted edges: edges that appear in one graph, but not in the other.

The goal of a typical graph matching algorithm is to find the mapping with the smallest possible distance, also called as the graph-edit distance [15]. This is a computationally complex problem, because the space of possible mappings that need to be explored. Thus in practice, some pruning technique must be employed.

Specific Techniques. Graph matching algorithm can primarily be varied with respect to two points. The first variation point is the metric that is used for computing the weight of mapped nodes. The second variation point is the algorithm that is used to explore the space of possible mappings.

The two main classes of metrics to compute the weight of mapped nodes are syntactic metrics and semantic metrics. Syntactic metrics look at the label as a string of characters. For example, a typical syntactic metric between two labels is string-edit distance, which is the minimum number of character insertions, deletions and substitutions that must be performed to transform one string into another. Semantic metrics treat the label as a list or bag of words. A typical semantic similarity metric is based on matching the words of two given labels and defining a distance based on this matching. Words that are closer semantically (e.g. they are synonyms or share a hypernym) are more likely to be matched. The number of matches found and the strength of the matches then determines the similarity between the labels. Additional tweaks may be applied to deal with unlabeled nodes such as gateways.

Several algorithms can be envisioned to explore the space of possible mappings between two business process graphs. One is a greedy algorithm that, in each iteration, adds a mapping between two nodes that decreases the distance the most, until no such mapping can be found anymore. Another is based on search of the space of mappings based on the so-called A-star heuristics. We have investigated these alternatives in a number of papers [16,17].

Implementation. The graph matching approach to business process matching has been implemented both as part of the ICoP framework [18] and as part of version 5 of the tool ProM[4]. ProM is open source. ICoP is available on request. The tool uses WordNet to compute the semantic weights of node mappings.

Evaluations. We have evaluated several graph matching techniques on a collection of models extracted from the SAP R/3 reference model. The extracted collection consists of 100 so-called "document" models that simulate a repository of process models, and 10 so-called "query" models that simulate business process graphs that a user would be looking for. The goal is, given a query model, to rank the document models according to their similarity to the query model.

In this experiment, the aim was to test how close different techniques correspond to the "perceived similarity" of models as determined by a golden standard. The golden standard was constructed by asking a number of individuals to rate the similarity between pairs of process models in the collection (query model, document model) on a scale of 1 to 7.

In this respect, we found that a technique based on A-star achieves a higher mean average precision, which is a measure of ranking accuracy commonly used in information retrieval. The greedy algorithm comes relatively close to the A-star algorithm, while being faster.

3.3 RefMod-Mine/NSCM - N-Ary Semantic Cluster Matching

Overview. The approach for clustering business process model nodes consists of four components which are executed sequentially. First of all it conducts a *semantic error detection (1)*, where defects of modeling are being identified and automatically handled. After that, it uses all models as input for an *n-ary cluster matcher (2)*, which uses a *semantic similarity measure (3)* for pairwise node comparison. As a result of that cluster matching we get a set of clusters containing nodes of all considered models, which are being *extracted* to *binary complex matchings (4)*.

[4] http://www.processmining.org

Specific Techniques. *Semantic error detection.* While analyzing different business process models, we recognized the existence of model failures which leads to a misinterpretation of nodes during a process matching. Against that background, the main function of semantic error detection is the identification of wrong modeled transition nodes. Since the algorithm as well as the gold standard only matches transitions, this functionality checks whether the label suggests a node being a place or confirms it being a transition. Therefore the form and order of nouns and verbs of a label are being analyzed, which leads to the applicability only to English language models. The identified transitions are being marked as "ignore" and will not be considered in the following matching components.

N-Ary cluster matching. In contrast to existing matching techniques, the authors use an n-ary clustering instead of a binary matching. The nodes of all models are being pairwise compared using a semantic similarity measure. Since the cluster algorithm is agglomerative [1], it starts with clusters of size 1 (= transitions) and consolidates two transitions to a cluster if their similarity value passes a user-defined threshold. If two nodes are being clustered and both are already part of different clusters, the two clusters are being merged. Thus, the resulting clusters are hard and not fuzzy [19].

Semantic similarity measure. The used similarity measure consists of three phases. The first phase splits node labels L into single words w_{i_L}, so that $split(L) = \{w_{1_L}, ..., w_{n_L}\}$. Stop words, like *the*, *is*, *at* as well as waste characters like additional spaces are being removed. The second phase computes the Porter Stem [20] $stem(w_{i_L})$ for each word and compares the stem sets of both labels. The number of stem matchings is being divided by the sum of all words.

$$sim(L_1, L_2) = \frac{|\{stem(w_{1_{L_1}}), ..., stem(w_{n_{L_1}})\} \cap \{stem(w_{1_{L_2}}), ..., stem(w_{m_{L_2}})\}|}{|split(L_1) + split(L_2)|}$$

If the resulting similarity value passes a user-defined threshold, the third phase checks the labels for antonyms using the lexical database WordNet [21] and checking the occurrence of negation words like not. Thus, that phase decides the similarity being 0 or $sim(L_1, L_2)$.

Binary matching extraction. The last component extracts binary matchings from the node clusters calculated by the n-ary cluster matcher. For each model pair all clusters are being scanned for the occurrence of nodes of both models. The containing node set of the first model is then being matched to the node set of the second model. Thus, the component returns a binary complex (N:M) matching for each model pair.

Implementation. The mentioned technique has been implemented in form of a php command line tool and can publicly checked out[5]. Next to the n-ary semantic cluster matching and other matching techniques, the research prototype is able to calculate node and process similarities from recent literature as well as analyzing models and matchings.

[5] https://refmodmine.googlecode.com/svn

Evaluations. To evaluate the approach, the authors analyzed the precision and recall values in case of the delivered admission models with the corresponding gold standard. After justifying the algorithm, the results leaded to a precision of 67 % and a recall of 34 %. Thereby, the threshold for semantic similarity was set to 60 %.

3.4 RefMod-Mine/ESGM - Extended Semantic Greedy Matching

Overview. In a first attempt dealing with the matching problem, a greedy matching [17] was implemented and evaluated based on precision and recall. Though a considerably high precision is achieved by this approach, only a low degree of recall is reached due to neglect of potential complex matches. To attain a higher recall and meet the demands of complex matches, the approach is extended.

The approach introduced here matches business process models pair-wisely based on the similarities of the process models' transitions. The result of the matching algorithm is a set of complex (N:M) transition matches between two process models. The matching is subdivided into three steps.

In the first step, a *pre-processing* of data is applied to the models. The second step consists in computing the similarity of all potential 1:1 transition matches of two models using a *word matching* technique. In a final step, a *heuristic grouping* of similar transitions from step 2 is conducted.

Specific Techniques. *Pre-Processing.* While evaluating precision and recall, the authors noticed that some transitions which seemed to represent process events rather than activities, had not been matched with regard to the gold-standard. Hence one step of the pre-processing is a heuristic filter which excludes such transitions from further matching steps.

Moreover, the labels of the transitions are split up into word sets according to split characters like whitespace or hyphen. After all non-word characters[6] have been removed from the word sets, stop words like *to*, *the*, and *is* are removed from the word sets.

Word Matching. Unlike most approaches, the computation of the transitions' similarity is accomplished applying the greedy matching technique [17] on business process models to transition labels. Therefore, at first, the similarity of the words of two labels is determined.

The computation of the similarity score sim_w of two words is based on dictionary lookups and a syntactic similarity measure [16]. In case the words represent synonyms or it exists a nominalization of one word that is synonymic to the other or vice versa, they receive a similarity score of 1. If the words or their nominalizations are considered antonyms, a similarity score of -1 is returned, otherwise they receive a syntactic similarity score based on Levenshtein's edit distance.

Let L be a label of Transition T that belongs to process model M and W a set of words of a label L. $sim_w(w_1, w_2)$ denotes the similarity of two words

[6] http://docs.oracle.com/javase/1.4.2/docs/api/java/util/regex/Pattern.html

$w_1 \in W_1$ and $w_2 \in W_2$. Furthermore, let $M_W : W_1 \rightarrow W_2$ be a partial injective mapping on the word sets W_1, W_2. Then $sim_L(L_1, L_2)$ denotes the similarity of two labels L_1, L_2.

$$sim_L(L_1, L_2) = \frac{\sum_{(w_1,w_2)\in M_W} sim_w(w_1, w_2)}{max(|W_1|, |W_2|)} \tag{1}$$

Heuristic Grouping. The subsequent grouping of transitions consists in adding all pairs which do not fall below a predefined similarity threshold t to the result. The following rules depict the heuristic grouping technique.

Let G be a set of transitions representing a group of transitions. Given a pair of transitions (T_1, T_2), which satisfies the threshold criterion $(sim_L(L_i, L_j) \geq t)$, a new group $G = \{T_1, T_2\}$ is added to the result set if neither T_1 nor T_2 belongs to any group. In case only one transition, either T_1 or T_2, is not represented in any group, this transition is added to the group the other transition belongs to. If T_1 belongs to group G_i and T_2 to group G_j, the groups G_i, G_j are replaced by the new group $G_n = G_i \cup G_j$.

Implementation. The matching approach is implemented in Java (jre6) and is embedded in RefMod-Mine, which is a tool set dedicated to the mining of reference models. The computation of the labels similarity largely relies on dictionary lookups. The underlying dictionary is the WordNet [21] database (v 3.0) and it is accessed via the RiTa.WordNet Java/Javascript API[7], which is free and open-source licensed under GPL.

Evaluations. The approach has been evaluated based on the partial gold standard provided. Therefore, the threshold for the grouping was set to 65 %.

3.5 Bag-of-Words Similarity with Label Pruning

Overview. The approach to process model matching discussed here is a subset of our previous paper [22]. While we explored various options before, herein we focus on the matching strategy that provided the most significant increase in match quality in our experiments. This technique solely considers activity labels, disregarding other information present in the process models such as events, process structure or behavior.

In a nutshell, the approach computes label similarity by (i) treating each label as a *bag of words* (a multi-set of words), (ii) applying word stemming (to transform, e.g., “evaluating” into “evaluate”) for better comparability, (iii) computing the similarity scores as per Levenshtein [23] and Lin [24] for each pair of words, (iv) pruning the multi-sets for both activity labels under comparison to be equal in the number of words, (v) computing an overall matching score for each activity pair, and (vi) selecting all activity pairs whose score is above a given threshold.

[7] RiTa.WordNet, http://www.rednoise.org/rita/wordnet/documentation/

Specific Techniques. For a detailed introduction of the overall approach we refer the reader to [22]. In the following we explain specific aspects of it and the configuration used in this paper.

One characteristic of the bag-of-words similarity is that it neglects the grammatical structure of the label. This is in contrast to [25] where the individual words of the labels are assigned with types; and words will only be compared if they belong to the same type. The rationale for neglecting label structure is that the brevity of labels makes it hard to deduce information like word forms. In this way, the bag-of-words similarity aims to offer a means to find matches like "reject applicant" vs. "send letter of rejection".

Furthermore, in case the two bags-of-words under comparison are different in size, the larger one is pruned to the size of the smaller one. Therefore, words with a small similarity score are removed from the larger set. This is done to better capture activity labels with a strong difference in specificity. For instance, "rank case" vs. "rank application on scale of 1 to 10" may have a very low average word similarity as the second label also contains information about a condition not present in the first label.

Finally, the decision to rely on a syntactical (Levenshtein) as well as a semantic (Lin) word similarity notion tries to lessen the weaknesses of both notions. While syntactical notions cannot account for a strong conceptual similarity of two words, a semantic notion struggles when spelling errors are present. However, there are still cases where this combination struggles.

Implementation. The technique is implemented in Java and part of the *Process Model Matching Tools for Java* (jpmmt)-project which aims at providing algorithms and measures for process model matching. The project is publicly available[8] under the terms of the MIT License[9].

Evaluations. In [22] we evaluated various configurations of the bag-of-words similarity with label pruning and its basic variant the bag-of-words similarity. These configurations included different pruning criteria and word similarity functions. In order to achieve comparability, we used the data set from [25] which includes the university admission processes and the corresponding matching standard also part of the data set of this matching contest. The evaluation showed that the technique has the potential to increase recall of process model matching compared to results yielded by the approaches introduced in [18,25].

Furthermore, we applied the technique in the context of *Business Process Querying* (BPQ). In [26] an approach to BPQ is presented that decomposes a BPMN-Q query [27] into a set of sub-queries. For these sub-queries corresponding process model fragments are determined within a process collection. Finally, these fragments are aggregated in order to provide a list of process model fragments that provide answers to the whole query. Our technique constitutes the

[8] http://code.google.com/p/jpmmt/

[9] http://opensource.org/licenses/mit-license.php

base for an extension of this approach. Instead of relying on 1:1 matches for the activities in the query this assumption is relaxed and more complex matching constellations are allowed. An evaluation which also relies on the university admission processes shows that the technique in combination with the approach from [26] yields promising results. However, the size of the collection and queries is relatively small, and further experiments need to be conducted.

3.6 PMLM - Process Matching Using Positional Language Model

Overview. This matching technique is tailored towards process models that feature textual descriptions of activities, introduced in detail in [28]. Using ideas from language modeling in Information Retrieval, the approach leverages those descriptions to identify correspondences between activities. More precisely, we combine two different streams of work on probabilistic language modeling. First, we adopt passage-based modeling such that activities are passages of a document representing a process model. Second, we consider structural features of process models by positional language modeling. While using those probabilistic language models, we create a similarity matrix between the activities and derive correspondences using second line matching.

Specific Techniques. *Activities as Passages.* Let $\mathcal{T}$ be a corpus of terms. For a process model P, we create a document $d = \langle T_1, \ldots, T_n \rangle$ as a sequence of length $n \in \mathbb{N}$ of passages, where each passage is a set of terms $d(i) = T \subseteq \mathcal{T}$, $1 \leq i \leq n$. The set $d(i) = T$ comprises all terms that occur in the label or description of the activity at position i. The length of d is denoted by $|d|$. We denote by $\mathcal{D}$ a set of processes, represented as documents.

Our model is built on a cardinality function $c : (\mathcal{T} \times \mathcal{D} \times \mathbb{N}) \rightarrow \{0, 1\}$, such that $c(t, d, i) = 1$ if $t \in T = d(i)$ (term t occurs in the i-th passage of d) and $c(t, d, i) = 0$ otherwise. To realize term propagation to close-by positions, a proximity-based density function $k : (\mathbb{N} \times \mathbb{N}) \rightarrow [0, 1]$ is used to assign a discounting factor to pairs of positions. Then, $k(i, j)$ represents how much of the occurrence of a term at position j is propagated to position i. We rely on the Gaussian Kernel $k^g(i, j) = e^{(-(i-j)^2)/(2\sigma^2)}$, defined with a spread parameter $\sigma \in \mathbb{R}^+$ [29]. Adapting function c with term propagation, we obtain a function $c' : (\mathcal{T} \times \mathcal{D} \times \mathbb{N}) \rightarrow [0, 1]$, such that $c'(t, d, i) = \sum_{j=1}^{n} c(t, d, j) \cdot k^g(i, j)$. Then, our positional, passage-based language model $p(t|d, i)$ captures the probability of term t occurring in the i-th passage of document d ($\mu \in \mathbb{R}$, $\mu > 0$, is a weighting factor):

$$p_\mu(t|d, i) = \frac{c'(t, d, i) + \mu \cdot p(t|d)}{\sum_{t' \in \mathcal{T}} c'(t', d, i) + \mu}. \tag{2}$$

Derivation of Passage Positions. To instantiate the positional language model for process models, we need to specify how to order the passages in the document to represent the order of activities in a process. In this matching contest, we chose to use a Breadth-First Traversal over the process model graph starting from an initial activity that creates the process instance (we insert a dummy node connect to all initial activities if needed).

Similarity of Language Models. Using the language models, we measure the similarity for document positions and, thus, activities of the process models, with the Jensen-Shannon divergence (JSD) [30]. Let $p_\mu(t|d,i)$ and $p_\mu(t|d',j)$ be the smoothed language models of two process model documents. Then, the probabilistic divergence of position i in d with position j in d' is:

$$jsd(d,d',i,j) = \frac{1}{2}\sum_{t\in\mathcal{T}} p_\mu(t|d,i)\lg\frac{p_\mu(t|d,i)}{p^+(t)} + \frac{1}{2}\sum_{t\in\mathcal{T}} p_\mu(t|d',j)\lg\frac{p_\mu(t|d',j)}{p^+(t)} \quad (3)$$
$$\text{with}\quad p^+(t) = \frac{1}{2}(p_\mu(t|d,i) + p_\mu(t|d',j))$$

When using the binary logarithm, the JSD is bound to the unit interval $[0,1]$, so that $sim(d,d',i,j) = 1 - jsd(d,d',i,j)$ can be used as a similarity measure.

Derivation of Correspondences. Finally, we derive correspondences from a similarity matrix over activities, which is known as second line matching. Here, we rely on two strategies, i.e., *dominants* and *top-k*, see [31]. The former selects pairs of activities that share the maximum similarity value in their row and column in the similarity matrix. The latter selects for each activity in one model, the k activities of the other process that have the highest similarity values.

Implementation. The application was built in C#, and uses the Lemur ToolKit for stemming terms, and calculating the probability of each term to be relevant given a certain passage and position in a document. In our implementation, we first read the XML files representing the process models, transform each element into an object according to its type (transition, place or arc) and order the transitions. In the first phase, we create an ordered document containing only the activity labels (with no propagation), create a similarity matrix using the Lemur ToolKit and find correspondences using *dominants* approach. In the second phase, we create another ordered document with activity labels, descriptions and term propagation, create a similarity matrix using the Lemur ToolKit and find correspondences using *top-3* approach. Finally, we choose matches according to the *dominants* result and add the selected *top-3* if their similarity score is no less then 80 % of the highest similarity value in their row.

The implementation is still in development stage, so for the time being it is not available for a public use.

Evaluations. We conducted experiments with several real-world model collections. First, we used models from the Bank of Northeast of Brazil (BNB) that capture business processes on three levels: business perspective, technical perspective, or executable process specification, also used in [2]. Second, we used models from an electronics company and from municipalities in the Netherlands, described and used for evaluation in [32]. All sets include textual annotations for at least some of the activities. Our results indicate that this matching technique is geared towards high recall, increasing it up to a factor of 5 over existing work [28]. While average precision is rather low, we observe k-precision values (k-precision extends precision to top-k lists, where a match is a top-k list where a correct pair is found) above 60 %. Hence, correct correspondences can be extracted by an expert with reasonable effort, thereby supporting semi-automated matching.

3.7 The ICoP Framework: Identification of Correspondences between Process Models

Overview. The ICoP framework [32] aims at solving the problem of matching process models with a particular focus on complex correspondences that are defined between sets of activities instead of single activities. Towards this end, the framework proposes an architecture and a set of re-usable components for assembling concrete matchers.

The ICoP architecture defines process model matching as a multi-step approach involving four different types of components.

Searchers try to cope with the combinatorial challenges induced by potentially complex correspondences by applying heuristics to search the space of possible matches. Here, different strategies are first applied for group activities and, second, for assessing the similarity of these groups of activities. Searchers return a set of candidate correspondences with assigned confidence scores.

Boosters aggregate candidate correspondences and adapt their scores. On the one hand, the multiset of matches returned by the searchers is aggregated to obtain a set of candidate correspondences. Also, scores are adapted, e.g., based on subsumption of candidate correspondences.

Selectors build up the actual final set of correspondences from the set of candidate correspondences, by selecting the best candidates that are non-overlapping in their sets of activities. Here, selection is guided by the scores of the candidates as well as an evaluation score computed by an evaluator (see below). Then, selection of correspondences is done iteratively. Yet, exhaustive search for the best selection is typically not possible, so that a greedy strategy or an approach with a certain lookahead is followed.

Evaluators assign a score to a set of correspondences. Computation of this score is typically based on the original process models, such that the consistency of certain structural or behavioural properties of the process models under the given correspondences is assessed.

In addition to this architecture, the ICoP framework provides different implementations of these four components that may be used to assemble matchers. Examples include searchers that rely on vector space scoring, different aggregation boosters, evaluators based on the graph edit distance, and selectors that implement different strategies for combining scores of individual candidates and the evaluation scores for sets of correspondences.

Specific Techniques. We want to highlight two specific techniques that are used in components of the ICoP framework:

Virtual Document Searchers. Searchers implement heuristics to first group activities in either process model and then assess the similarity of these groups to derive candidate correspondences. Given a set of activities groups in either model (e.g., derived based on proximity in terms of graph distance or by structural decomposition), searchers in the ICoP framework exploit virtual documents for similarity assessment. Here, the notion of a virtual document is inspired by work on ontology alignment [33] where a virtual document of a node consists of all textual information in an ontology that is related to that node. Then, two virtual documents are scored based their Cosine similarity in a vector space that is spanned by the terms that appear in the documents. In the ICoP searchers, a virtual document for a group of activities consists of the terms of the activity label and any additional textual information related to the activity, such as an activity description, data input and output artefacts, and names and descriptions of related roles and information systems. Combined with common techniques from information retrieval, e.g., stop-word filtering and term-frequency based weighting, this technique provides a means to consider not only activity labels, but a broad spectrum of textual information related to an activity for the matching.

Execution Semantics Evaluator. An evaluator scores a set of correspondences, typically based on the original process models. The ICoP framework defines an evaluator that exploits the execution semantics of the process models for scoring a set of correspondences. To this end, it relies on the relations of the behavioural profile of a process model, cf., [34]. Such a profile abstracts trace semantics of a process by a set of binary behavioural relations defined over its activities: two activities are ordered (if one can occur before the other but not vice versa), exclusive (if they cannot occur jointly in an execution sequence), or interleaved (if they can occur in either order). This information is used for assigning a score to a set of correspondences by checking for each pairs of activities of distinct correspondences in one model, whether their behavioural relation is mirrored by all the corresponding activities in the other model. Then, the ratio of consistent pairs and all investigated pairs provides us with a score that captures the extent to which the behavioural characteristics of one model are preserved in the other model under the given correspondences.

Implementation. The ICoP framework has been implemented in Java and is available upon request from the authors of [32]. Currently, process models are expected to be given as Petri nets in the PNML format.

A major revision of the framework is under way. By building upon the jBPT library [35], this new implementation will support a broader class of process model descriptions and serialization formats.

Evaluations. The ICoP framework has been designed with a particular focus on the identification of complex correspondences. An evaluation of the framework can be found in [32]. It illustrates that the ICoP architecture allows for the creation of matchers that find a significant share of complex correspondences. Yet, it also shows that a certain homogeneity of the process model vocabulary is required for the identification of complex correspondences.

4 Results

For assessing the submitted process model matching techniques, we compare the computed matches against a manually created gold standard. Using the gold standard, we classify each computed activity match as either true-positive (TP), true-negative (TN), false-positive (FP) or false-negative (FN). Based on this classification, we calculate the precision (TP/(TP+FP), the recall (TP/(TP+FN)), and the f-measure, which is the harmonic mean of precision and recall (2*precision*recall/(precision+recall)). Table 3 gives an overview of the results for the university admission data set and Table 4 presents the results for the birth certificate data set. For getting a better understanding of the result details, we report the average (AVG) and the standard deviation (STD) for each metric. The highest value for each metric is marked using bold font.

Table 3. Results of university admission matching

		Precision		Recall		F-Measure	
No.	Approach	AVG	STD	AVG	STD	AVG	STD
1	Triple-S	0.31	0.19	0.36	0.26	0.33	0.12
2	BP Graph Matching	**0.60**	0.45	0.19	0.30	0.29	0.29
3	RefMod-Mine/NSCM	0.37	0.22	0.39	0.27	0.38	0.19
4	RefMod-Mine/ESGM	0.16	0.26	0.12	0.21	0.14	0.17
5	Bag-of-Words Similarity	0.56	0.23	0.32	0.28	**0.41**	0.20
6	PMLM	0.12	0.05	**0.58**	0.20	0.20	0.08
7	ICoP	0.36	0.24	0.37	0.26	0.36	0.23

Table 4. Results of birth certificate matching

		Precision		Recall		F-Measure	
No.	Approach	AVG	STD	AVG	STD	AVG	STD
1	Triple-S	0.19	0.21	0.25	0.33	0.22	0.23
2	BP Graph Matching	0.55	0.48	0.19	0.28	0.28	0.30
3	RefMod-Mine/NSCM	**0.68**	0.19	0.33	0.22	**0.45**	0.18
4	RefMod-Mine/ESGM	0.25	0.28	0.18	0.26	0.21	0.23
5	Bag-of-Words Similarity	0.29	0.35	0.22	0.30	0.25	0.31
6	PMLM	0.19	0.09	**0.60**	0.20	0.29	0.12
7	ICoP	0.42	0.27	0.28	0.23	0.33	0.24

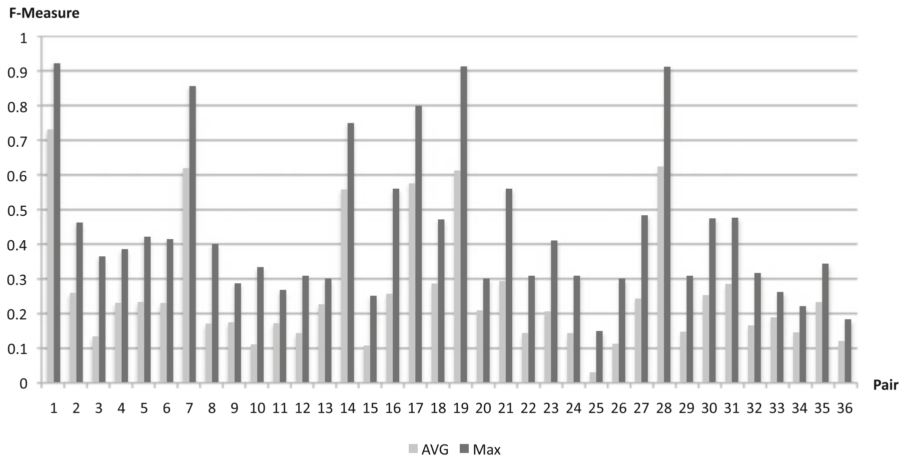

Fig. 1. Detailed results of admission data set

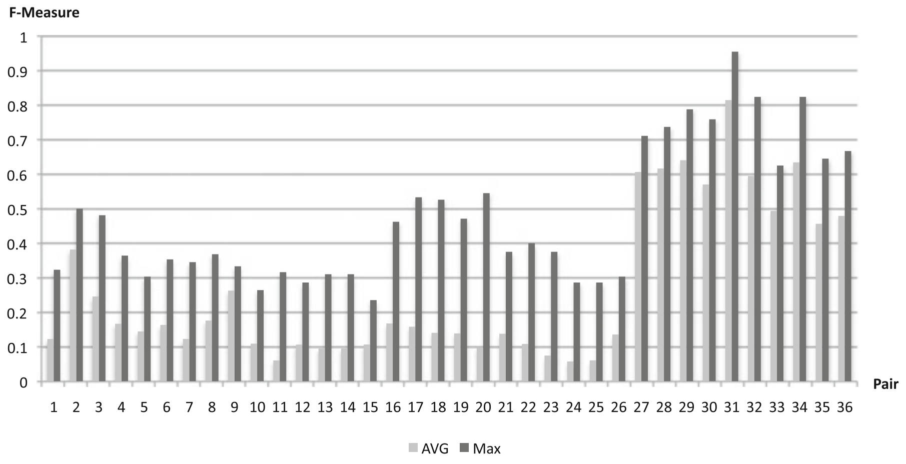

Fig. 2. Detailed results of birth certificate data set

From the results presented in Table 3 and Table 4, we can draw the following conclusions. Most importantly, it has to be noted there is no clear winner. As the employed data sets are different with respect to characteristics such as the number of complex correspondences and the linguistic consistency, different capabilities are required to come up with a good matching result. Apparently, no technique can perfectly deal with both data sets. However, there are a couple of interesting observations.

Focussing on the f-measure, the bag-of-words similarity approach yields the best result for the university admission set (0.41) and the RefMod-Mine/NSCM approach yields the best result for the birth certificate set (0.45). However, it should be noted that the RefMod-Mine/NSCM approach is quite close to the f-measure of the bag-of-words similarity approach for the university admission set (0.38) while the bag-of-words approach has a rather average result quality for the birth certificate models (0.25). Interestingly, the best f-measure is not necessarily associated with the best recall and precision. The PMLM approach yields the best recall (0.60 and 0.58) for both sets. Nevertheless, due to its rather low precision, it only yields average f-measures. The opposite situation can be observed for the BP Graph Matching approach. While it has rather low recall values, it yields top precision values (0.48 and 0.60). Apparently, the trade-off between precision and recall is still a major issue in the context of process model matching.

Looking at the standard deviation, we can see that many approaches suffer from quite unstable results. A detailed consideration of the results for individual model pairs reveals that there are some model pairs that are matched well, while others represent a considerable challenge for all participating techniques. Figures 1 and 2 illustrate this fact by showing the average and maximum f-measure among all techniques for each matching pair. In the admission set, we observe particular high results for the pairs 1,7, 14, 17, 19, and 28. The pairs 25 and 36 apparently represent complex matching problems. For the birth certificate data set, we observe a quite similar constellation. While the techniques yield good results for the pairs 31, 32, and 34, they fail to adequately match the pairs 10 and 15. Having a closer look into these extreme cases, we can identify two main characteristics that influence the result quality for a matching pair: the similarity of labels and the number of complex matches.

The more *similar* the labels of the matching pair, the better the matching result. By contrast, if many business objects are different or even missing, the identification of the matches may represent a serious challenge. As example, consider the match between *Checking if complete* and *Check documents*. Here, the rather unspecific verb *check* is the only connection between the labels. The second characteristic indicating the hardness of the matching challenge is the *the number of complex matches*. As such matches often require a semantic grouping of activities, their identification is a complicated and error-prone task. The identification of complex matches is often further aggravated by the fact that the connection between actions and business objects is hard to detect. As example, consider the complex match between the activity *Clarify name* and the activities *Consult mother* and *Consult father*. Taking a standard semantic similarity measure such as the Lin metric, the similarity between these labels is close to zero. In order to adequately address such problems, more sophisticated approaches are required.

Besides this comparative discussion, the obtained precision and recall values indicate that matching techniques cannot yet be expected to provide an out-of-the-box solution for fully automated matching. However, the detailed analysis of individual model pairs reveals that very good results can be obtained for a certain setting. Also, the variability of the techniques in terms of their preference for either precision or recall outlines potential for further improvements.

5 Future Directions

Based on the results and the observations from the Process Model Matching Contest 2013, we use this section to outline major directions for future work in the field of process model matching. In particular, we discuss strategies to address the overall matching result quality, the need for addressing semantics, the applicability of push-button approaches, and the question of how process model matching can be evaluated.

The results from this contest highlight that the *overall result quality* still needs to improved. Still, the differences between the employed data sets also indicate that many techniques can properly match a particular set of models. This raises the question whether appropriate matchers can be automatically selected based on the matching problem at hand. This, however, requires a precise understanding of the capabilities of the different matchers and an accurate selection algorithm. A promising strategy to address this problem might be the incorporation of prediction techniques as they have been recently proposed in the area of schema matching [36]. If the quality of the result of a matching technique can be predicted based on certain characteristics of the model or the match constellation, the best matching technique can be selected in an automated fashion. In this context, it could be also a promising strategy to determine a set of matchers that jointly address the given matching problem.

The detailed consideration of the matching results revealed that particular *semantic* relationships are hard to detect. Hence, we are convinced that semantic technologies need to be explored in more detail. While it turned out to be helpful to differentiate between label components such as action and business object, the simple comparison with semantic similarity measures is not sufficient. In order to detect more complex semantic relationships, it might be necessary to include ontologies or additional information such as textual descriptions of the models.

Most of the currently existing process model matching techniques represent *push-button approaches* that compute results without any user interaction. Thus, matching shall be considered as an iterative process that includes feedback cycles with human experts, a process known as reconciliation in data integration [37,38]. Given the general complexity of the matching task, such a semi-automated technique could still provide significant support to the user. By collecting feedback from the user, important decisions during the construction of a matching can be validated, leading to a better overall result.

So far, many matching techniques *evaluate* the result quality using precision, recall, and f-measure. However, considering the complexity of the matching setting, it can be doubted that these simplistic metrics are appropriate. In many cases, a match constellation is not necessarily true or false and the decision is even hard for humans. Against this background, it might be worth to pursue different evaluation strategies, such as non-binary evaluation [39]. Also, one shall consider the actual benefit achieved by (semi-) automated matching. However, measuring the post-match effort turned out to be challenging and is also not well understood for related matching problems [40]. Further work is needed to understand how tool-supported matching compares to manual matching in terms of time and quality.

Altogether, it must be stated that there are many directions for future research. Many of them are concerned with improving existing techniques. However, acknowledging that process model matching is not a simple task with a single correct result, it is also important to focus on alternative evaluation strategies.

6 Conclusion

In this paper, we reported on the setup and the results of the Process Model Matching Contest 2013. This contest addressed the need for effective evaluation of process model matching techniques. We provided two different process model matching problems and received automatically generated results of 7 different techniques. The evaluation of the results showed that their is no clear winner of the contest since no approach yielded the best performance for both data sets. We learned that there is still a huge trade-off between precision and recall, and that semantic and complex correspondences represent considerable challenges.

For future work, we highlighted that it is important to further improve the result quality achieved by the matching techniques. This may be accomplished by automatically selecting the best matcher based on the matching problem at hand, by exploiting semantics in a more elaborated way, or by incorporating user feedback. Further, we emphasized the importance of proper evaluation. As precision, recall, and f-measure are overly simplistic and only allow matches do be true or false, it might be worth to consider alternative evaluation strategies. This may, for instance, include the comparison of a matching technique with a human matching in terms of time and quality.

Acknowledgement. This work has been developed with the support of DFG (German Research Foundation) under the project SemReuse OB 97/9-1.

References

1. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.: Fundamentals of Business Process Management. Springer, Berlin (2012)
2. Castelo Branco, M., Troya, J., Czarnecki, K., Küster, J., Völzer, H.: Matching business process workflows across abstraction levels. In: France, R.B., Kazmeier, J., Breu, R., Atkinson, C. (eds.) MODELS 2012. LNCS, vol. 7590, pp. 626–641. Springer, Heidelberg (2012)
3. Küster, J.M., Koehler, J., Ryndina, K.: Improving business process models with reference models in business-driven development. In: Eder, J., Dustdar, S. (eds.) BPM Workshops 2006. LNCS, vol. 4103, pp. 35–44. Springer, Heidelberg (2006)
4. Weidlich, M., Mendling, J., Weske, M.: A foundational approach for managing process variability. In: Mouratidis, H., Rolland, C. (eds.) CAiSE 2011. LNCS, vol. 6741, pp. 267–282. Springer, Heidelberg (2011)

5. La Rosa, M., Dumas, M., Uba, R., Dijkman, R.: Business process model merging: An approach to business process consolidation. ACM Trans. Softw. Eng. Methodol. **22**(2), 11:1–11:42 (2013)
6. Dumas, M., García-Bañuelos, L., Dijkman, R.M.: Similarity search of business process models. IEEE Data Eng. Bull. **32**(3), 23–28 (2009)
7. Kunze, M., Weidlich, M., Weske, M.: Behavioral similarity – a proper metric. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) BPM 2011. LNCS, vol. 6896, pp. 166–181. Springer, Heidelberg (2011)
8. Jin, T., Wang, J., Rosa, M.L., ter Hofstede, A.H., Wen, L.: Efficient querying of large process model repositories. Comput. Ind. **64**(1), 41–49 (2013)
9. Ekanayake, C.C., Dumas, M., García-Bañuelos, L., La Rosa, M., ter Hofstede, A.H.M.: Approximate clone detection in repositories of business process models. In: Barros, A., Gal, A., Kindler, E. (eds.) BPM 2012. LNCS, vol. 7481, pp. 302–318. Springer, Heidelberg (2012)
10. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer, Heidelberg (DE) (2007)
11. Bellahsene, Z., Bonifati, A., Rahm, E. (eds.): Schema Matching and Mapping. Springer, Heidelberg (2011)
12. Wu, Z., Palmer, M.: Verbs semantics and lexical selection. In: Proceedings of the 32nd annual meeting on Association for Computational Linguistics, ACL '94, pp. 133–138, Stroudsburg, PA, USA. Association for Computational Linguistics (1994)
13. Ehrig, M., Koschmider, A., Oberweis, A.: Measuring similarity between semantic business process models. In: Roddick, J.F., Hinze, A. (eds.) Proceedings of the 4th Asia-Pacific Conference on Conceptual Modelling. Australian Computer Science Communications, vol. 67, pp. 71–80 (2007)
14. Miller, G., Fellbaum, C.: WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998)
15. Bunke, H.: On a relation between graph edit distance and maximum common subgraph. Pattern Recogn. Lett. **18**(8), 689–694 (1997)
16. Dijkman, R.M., Dumas, M., van Dongen, B.F., Käärik, R., Mendling, J.: Similarity of business process models: metrics and evaluation. Inf. Syst. **36**(2), 498–516 (2011)
17. Dijkman, R., Dumas, M., García-Bañuelos, L.: Graph matching algorithms for business process model similarity search. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 48–63. Springer, Heidelberg (2009)
18. 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)
19. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Comput. Surv. **31**(3), 264–323 (1999)
20. Porter, M.F.: Readings in Information Retrieval. Morgan Kaufmann Publishers Inc., San Francisco (1997)
21. Miller, G.A.: WordNet: a Lexical database for english. Commun. ACM **38**(11), 39–41 (1995)
22. Klinkmüller, C., Weber, I., Mendling, J., Leopold, H., Ludwig, A.: Increasing recall of process model matching by improved activity label matching. In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 211–218. Springer, Heidelberg (2013)
23. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Sov. Phys. Dokl. **10**(8), 707–710 (1966)

24. Lin, D.: An information-theoretic definition of similarity. In. In Proceedings of the 15th International Conference on Machine Learning, pp. 296–304. Morgan Kaufmann (1998)
25. Leopold, H., Smirnov, S., Mendling, J.: On the refactoring of activity labels in business process models. Inf. Syst. **37**(5), 443–459 (2012)
26. Sakr, S., Awad, A., Kunze, M.: Querying process models repositories by aggregated graph search. In: La Rosa, M., Soffer, P. (eds.) BPM 2012 Workshops. LNBIP, vol. 132, pp. 573–585. Springer, Heidelberg (2013)
27. Awad, A.: BPMN-Q: A language to query business processes. In: Reichert, M., Strecker, S., Turowski, K. (eds.) EMISA. LNI, vol. P-119, pp. 115–128 St. Goar, Germany, GI (2007)
28. Weidlich, M., Sheetrit, E., Branco, M., Gal, A.: Matching business process models using positional passage-based language models. In: Ng, W., Storey, V.C., Trujillo, J.C. (eds.) ER 2013. LNCS, vol. 8217, pp. 130–137. Springer, Heidelberg (2013)
29. Lv, Y., Zhai, C.: Positional language models for information retrieval. In: Allan, J., Aslam, J.A., Sanderson, M., Zhai, C., Zobel, J. (eds.) SIGIR, pp. 299–306. ACM, New York (2009)
30. Lin, J.: Divergence measures based on the shannon entropy. IEEE Trans. Inf. Theory **37**(1), 145–151 (1991)
31. Gal, A., Sagi, T.: Tuning the ensemble selection process of schema matchers. Inf. Syst. **35**(8), 845–859 (2010)
32. 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)
33. Qu, Y., Hu, W., Cheng, G.: Constructing virtual documents for ontology matching. In: Carr, L., Roure, D.D., Iyengar, A., Goble, C.A., Dahlin, M. (eds.) WWW, pp. 23–31. ACM, New York (2006)
34. Weidlich, M., Mendling, J., Weske, M.: Efficient consistency measurement based on behavioral profiles of process models. IEEE Trans. Softw. Eng. **37**(3), 410–429 (2011)
35. Polyvyanyy, A., Weidlich, M.: Towards a compendium of process technologies - the jbpt library for process model analysis. In: Deneckère, R., Proper, H.A. (eds.) CAiSE Forum. CEUR Workshop Proceedings, vol. 998, pp. 106–113. www.CEUR-WS.org (2013)
36. Sagi, T., Gal, A.: Schema matching prediction with applications to data source discovery and dynamic ensembling. VLDB J. **22**(5), 689–710 (2013)
37. Belhajjame, K., Paton, N.W., Fernandes, A.A.A., Hedeler, C., Embury, S.M.: User feedback as a first class citizen in information integration systems. In: CIDR, pp. 175–183. www.cidrdb.org (2011)
38. Quoc Viet Nguyen, H., Wijaya, T.K., Miklós, Z., Aberer, K., Levy, E., Shafran, V., Gal, A., Weidlich, M.: Minimizing human effort in reconciling match networks. In: Ng, W., Storey, V.C., Trujillo, J.C. (eds.) ER 2013. LNCS, vol. 8217, pp. 212–226. Springer, Heidelberg (2013)
39. Sagi, T., Gal, A.: Non-binary evaluation for schema matching. In: Atzeni, P., Cheung, D., Ram, S. (eds.) ER 2012. LNCS, vol. 7532, pp. 477–486. Springer, Heidelberg (2012)
40. Duchateau, F., Bellahsene, Z., Coletta, R.: Matching and alignment: what is the cost of user post-match effort? In: Meersman, R., et al. (eds.) OTM 2011, Part I. LNCS, vol. 7044, pp. 421–428. Springer, Heidelberg (2011)