Jens Kolb et al. (Eds.): Enterprise Modelling and Information Systems Architectures,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2015 127

The Process Model Matching Contest 2015

Goncalo Antunes[1] Marzieh Bakhshandeh[1] Jose Borbinha[1] Joao Cardoso[1] Sharam Dadashnia[2] Chiara Di Francescomarino[3] Mauro Dragoni[3] Peter Fettke[2] Avigdor Gal[4] Chiara Ghidini[3] Philip Hake[2] Abderrahmane Khiat[5] Christopher Klinkmüller[6] Elena Kuss[7] Henrik Leopold[8] Peter Loos[2] Christian Meilicke[7] Tim Niesen[2] Catia Pesquita[9] Timo Péus[10] Andreas Schoknecht[11] Eitam Sheetrit[4] Andreas Sonntag[2] Heiner Stuckenschmidt[7] Tom Thaler[2] Ingo Weber[12] Matthias Weidlich[13]

Abstract: Process model matching refers to the automatic identification of correspondences between the activities of process models. Application scenarios of process model matching reach from model validation over harmonization of process variants to effective management of process model collections. Recognizing this, several process model matching techniques have been developed in recent years. However, to learn about specific strengths and weaknesses of these techniques, a common evaluation basis is indispensable. The second edition of the Process Model Matching Contest in 2015 hence addresses 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. Next to a description of the contest matching problems, the paper provides short descriptions of all matching techniques that have been submitted for participation. In addition, we present and discuss the evaluation results and outline directions for future work in the field of process model matching.

Keywords: Process matching, model alignment, contest, matching evaluation

1 Introduction

To achieve control over their business operations, organizations increasingly invest time and effort in the creation of process models. In these process models, organizations capture the essential activities of their business processes together with the activity's execution

[1] Instituto Superior Tecnico, Universidade de Lisboa and INESC-ID, Lisbon, Portugal, marzieh.bakhshandeh|joao.m.f.cardoso|goncalo.antunes|jose.borbinha@tecnico.ulisboa.pt
[2] Institute for Information Systems (IWi) at the German Research Center for Artificial Intelligence (DFKI) and Saarland University, Saarbrücken, Germany, Sharam.Dadashnia|Peter.Fettke|Philip.Hake|Peter.Loos|Tim.Niesen|Andreas.Sonntag|Tom.Thaler@dfki.de
[3] Fondazione Bruno Kessler, Trento, dragoni|dfmchiara|ghidini@fbk.eu
[4] Technion - Israel institute of Technology, Technion City, Haifa, Israel, avigal |eitams@ie.technion.ac.il
[5] LITIO Lab, University of Oran, Oran, Algeria, abderrahmane_khiat@yahoo.com
[6] University of Leipzig, Leipzig, Germany, klinkmueller@wifa.uni-leipzig.de
[7] Universität Mannheim, Mannheim, Germany, elena|christian|heiner@informatik.uni-mannheim.de
[8] VU University Amsterdam, Amsterdam, The Netherlands, h.leopold@vu.nl
[9] LaSIGE, Faculdade de Ciencias, Universidade de Lisboa, Portugal, cpesquita@di.fc.ul.pt
[10] Technische Hochschule Mittelhessen, KITE - Kompetenzzentrum für Informationstechnologie, Friedberg, Germany, timo.peus@mnd.thm.de
[11] Karlsruhe Institute of Technology, Institute AIFB, Karlsruhe, Germany, andreas.schoknecht@kit.edu
[12] Software Systems Research Group, NICTA, Sydney, Australia, Ingo.Weber@nicta.com.au
[13] Humboldt Universität zu Berlin, Berlin, Germany, matthias.weidlich@informatik.hu-berlin.de

dependencies. The increasing size of process model repositories in industry and the resulting need for automated processing techniques has led to the development of a variety of process model analysis techniques. One type of such analysis techniques are process model matching approaches, which are concerned with supporting the creation of an alignment between process models, i.e., the identification of correspondences between their activities. The actual importance of process model matching techniques is demonstrated by the wide range of techniques that build on an existing alignment between process models. Examples for such techniques include the validation of a technical implementation of a business process against a business-centered specification model [Br12], delta-analysis of process implementations and a reference model [KKR06], harmonization of process variants [WMW11, La13], process model search [DGBD09, KWW11, Ji13], and clone detection [Ek12].

In this paper, we report on the setup and results of the Process Model Matching Contest (PMMC) 2015. It was the second edition of this event after the first PMMC in 2013 [Ca13a] and took place on September 4, 2015, at the 6th International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA) in Innsbruck, Austria. The Contest Co-Chairs were Elena Kuss, Henrik Leopold, Christian Meilicke, Heiner Stuckenschmidt, and Matthias Weidlich.

The Process Model Matching Contest (PMMC) 2015 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. Inspired by the Ontology Alignment Evaluation Initiative (OAEI)[3], the PMMC was organized as a controlled, experimental evaluation. In total, three 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 12 submissions to the contest covering diverse techniques for addressing the problem of process model matching. All submissions provided 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 provides details on the process model matching problems of the PMMC 2015. Section 3 features the short descriptions of the submitted matching approaches. Section 4 presents the evaluation results. Section 6 concludes and discusses future directions.

[3] `http://oaei.ontologymatching.org`

2 Data Sets

The contest included three sets of process model matching problems:

- **University Admission Processes (UA & UA$_S$)**: This set consists of 36 model pairs that were derived from 9 models representing the application procedure for Master students of nine German universities. The process models are available in BPMN format. Compared to the 2013 version of the dataset, we have fixed several issues with the models that have to be matched, changed the format of the models, and have strongly improved the quality of the gold standard. With respect to the gold standard, we have distinguished between equivalence matches and subsumption matches (a general activity is matched on a more specific activity). We use in our evaluation both a strict version of the gold standard which contains only equivalence correspondences (UA) and a relaxed version which contains additionally a high number of subsumption correspondences (UA$_S$).
- **Birth Registration Processes (BR)**: This set consists of 36 model pairs that were derived from 9 models representing the birth registration processes of Germany, Russia, South Africa, and the Netherlands. The models are available as Petri-Nets (PNML format). This version of the dataset has also been used in the 2013 contest.
- **Asset Management (AM)**: This set consist of 36 model pairs that were derived from 72 models from the SAP Reference Model Collection. The selected process models cover different aspects from the area of finance and accounting. The models are available as EPCs (in EPML-format). The dataset is new to the evaluation contest. The evaluation of this dataset is done blind, i.e., the participants do not know the gold standard of the dataset in advance.[4]

Characteristic	**UA**	**UA$_S$**	**BR**	**AM**
No. of Activities (min)	12	12	9	1
No. of Activities (max)	45	45	25	43
No. of Activities (avg)	24.2	24.2	17.9	18.6
No. of 1:1 Correspondences (total)	202	268	156	140
No. of 1:1 Correspondences (avg)	5.6	7.4	4.3	3.8
No. of 1:n Correspondences (total)	30	360	427	82
No. of 1:n Correspondences (avg)	0.8	10	11.9	2.3

Tab. 1: Characteristics of Test Data Sets

Table 1 summarizes the main characteristics of the three data sets. It shows the minimum, maximum, and average number of activities per model as well as the total and average number of 1:1 and 1:n correspondences. A 1:1 correspondence matches two activities A and A' such that no other correspondence in the gold standard matches A or A' to some

[4] This dataset was developed by Christopher Klinkmüller based on the SAP Reference Model. We thank Christopher for making that dataset available to the contest.

other activity. Contrary to this, 1:n correspondences match an activity A to several other activities $A_1, \ldots, A_n$. This can, for example, happen when an activity has to be matched to a sequence of activities. A high number of 1:n correspondences indicates that the matching task is complex and that the models describe processes on a different level of granularity.

The numbers show that the model sets differ with regard to the number of 1:n correspondences. Obviously, adding subsumption correspondences results in a high number of 1:n correspondences, while the restriction to equivalence correspondences suppresses 1:n correspondences (compare the data sets UA and UA_S). The highest fraction of 1:n correspondences can be found in the BR data set. Even though the number of activities of the models is quite close ranging from 9 to 25, the modeling style seems to differ, because only $\approx$27% of all correspondences are 1:1 correspondences.

3 Matching Approaches

In this section, we give an overview of the participating process model matching approaches. In total, 12 matching techniques participated in the process model matching contest. Table 2 provides an overview of the participating approaches and the respective authors. In the following subsections, we provide a brief technical overview of each matching approach.

No.	Approach	Authors
1	AML-PM	Marzieh Bakhshandeh, Joao Cardoso, Goncalo Antunes, Catia Pesquita, Jose Borbinha
2	BPLangMatch	Eitam Sheetrit, Matthias Weidlich, Avigdor Gal
3	KnoMa-Proc	Mauro Dragoni, Chiara Di Francescomarino, Chiara Ghidini
4	Know-Match-SSS (KMSSS)	Abderrahmane Khiat
5	Match-SSS (MSSS)	Abderrahmane Khiat
6	RefMod-Mine/VM^2 (RMM/VM2)	Sharam Dadashnia, Tim Niesen, Philip Hake, Andreas Sonntag, Tom Thaler, Peter Fettke, Peter Loos
7	RefMod-Mine/NHCM (RMM/NHCM)	Tom Thaler, Philip Hake, Sharam Dadashnia, Tim Niesen, Andreas Sonntag, Peter Fettke, Peter Loos
8	RefMod-Mine/NLM (RMM/NLM)	Philip Hake, Tom Thaler, Sharam Dadashnia, Tim Niesen, Andreas Sonntag, Peter Fettke, Peter Loos
9	RefMod-Mine/SMSL (RMM/SMSL)	Andreas Sonntag, Philip Hake, Sharam Dadashnia, Tim Niesen, Tom Thaler, Peter Fettke, Peter Loos
10	OPBOT	Christopher Klinkmüller, Ingo Weber
11	pPalm-DS	Timo Péus
12	TripleS	Andreas Schoknecht

Tab. 2: Overview of Participating Approaches

3.1 AML-PM

3.1.1 Overview

The AgreementMakerLight (AML) [Fa13] is an ontology matching system which has been optimized to handle the matching of larger ontologies. It was designed with flexibility and extensibility in mind, and thus allows for the inclusion of virtually any matching algorithm. AML contains several matching algorithms based both on lexical and structural properties, and also supports the use of external resources and alignment repair. These features have allowed AML to achieve top results in several OAEI 2013 and 2014 tracks [Dr14]. The modularity and extensibility of the AML framework made it an appropriate choice to handle the matching of the datasets of this contest. However, AML works over OWL ontologies, so there was a need to pre-process the input data and translate it into OWL. Then a matching pipeline was applied that included several lexical-based matchers and a global similarity optimization step to arrive at a final alignment.

3.1.2 Specific techniques

The workflow we used is composed of four steps (see Figure 1):

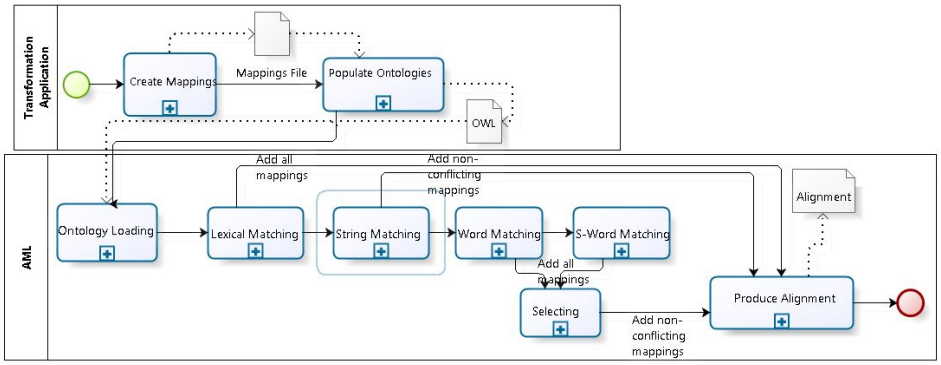

Fig. 1: Transformation Application-AML model matching process

- Transformation: Since the contest involved three datasets represented using three different modelling languages, an application for the transformation of the datasets into an ontological representation was used. This transformation application uses data to create and populate ontologies, independently from the schema used for organizing source data. Independence is achieved by resorting to the use of a mappings specification schema. This schema defines mappings to establish relations between data elements and the various ontology classes. Those relations are then used to create and populate an ontology with individuals (instances), thus representing the original data in the form of an OWL ontology.

- Ontology Loading: We updated AML to load individuals, which up until now were not handled by this system. When loading an ontology, AML creates efficient data structures that store lexical, structural and semantic information. These include a lexicon, that includes all the labels used in the ontology, and also derived synonyms, by removing leading and trailing stop words.

- Ontology Matching: We employed three distinct matchers: The Lexical Matcher, which is one of the simplest and most efficient matching algorithms, looks for literal name matches in the Lexicons of the input ontologies; the String Matcher, which implements a variety of string similarity metrics; and the Word Matcher, which measures the similarity between two individuals through a weighted Jaccard index between the words present in their names. These three matchers are employed in a four step sequential pipeline: first we apply the lexical matcher, and since this is a high-confidence matcher and we include all mappings above a given threshold in our final alignment; then, we apply the string matcher, and all mappings above a threshold that are not in conflict with the mappings already in the alignment are added; finally we apply the word matcher with and without stemming of words. These mappings, given their lower confidence are then run through a selection step before being added to the final alignment.

- Selection: Selectors are algorithms used to trim an alignment by excluding mappings below a given similarity threshold and excluding competing mappings to obtain the desired cardinality, typically one-to-one. The selector algorithm sorts the mappings in the Alignment in descending order of their similarity values, then adds mappings to the final alignment, as long as they do not include individuals already selected, until it hits the desired cut-off threshold.

3.2 BPLangMatch

3.2.1 Overview

This matching technique is tailored towards process models that feature textual descriptions of activities, introduced in detail in [We13]. 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. Combining these aspects, we rely on a novel positional passage-based language model to create a similarity matrix. The similarity scores are then adapted based on semantic information derived by Part-Of-Speech tagging, before correspondences are derived using second line matching. Figure 2 illustrates the various steps of our approach.

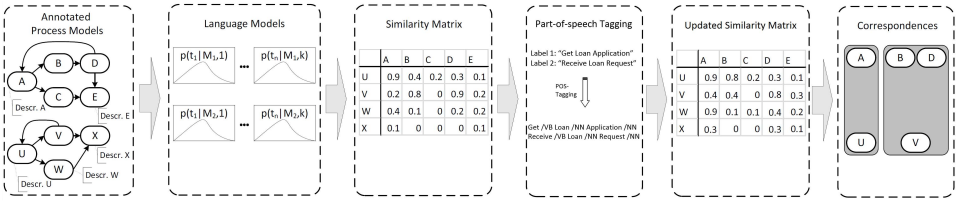

Fig. 2: Overview of the process model matching steps

3.2.2 Specific Techniques

Activities as Passages. Let $\mathscr{T}$ 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 $d(i) = T_i \subseteq \mathscr{T}$, $1 \leq i \leq n$, is a set of terms. The set $d(i)$ 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 $\mathscr{D}$ a set of processes, represented as documents.

Our model is built on a cardinality function $c : (\mathscr{T} \times \mathscr{D} \times \mathbb{N}) \to \{0,1\}$, such that $c(t,d,i) = 1$ if $t \in d(i)$ (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 $k : (\mathbb{N} \times \mathbb{N}) \to [0,1]$ 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 $\sigma \in \mathbb{R}^+$ [LZ09]. In this contest we used $\sigma = 1$. Adapting function c with term propagation, we obtain a function $c' : (\mathscr{T} \times \mathscr{D} \times \mathbb{N}) \to [0,1]$, 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):

$$p_\mu(t|d,i) = \frac{c'(t,d,i) + \mu \cdot p(t|d)}{\sum_{t' \in \mathscr{T}} c'(t',d,i) + \mu}. \tag{1}$$

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) [Li91]. 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:

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

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.

Increasing Similarity Scores. In many cases, when we encounter textual heterogeneity in the label and description of two similar activities, the nouns remain the same, and the heterogeneity is limited to verbs, adjectives, and other words. Thus, once a similarity matrix has been derived for two process models, we increase score of activities who share the same nouns. For identifying the nouns of each activity, we rely on the Stanford Log-linear Part-Of-Speech Tagger [To03].

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 [GS10]. 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.

3.3 KnoMa-Proc

3.3.1 Overview

The proposed KnoMa-Proc system addresses the process model matching problem in an original way. It implements an approach based on the use of information retrieval (IR) techniques for discovering candidate matches between process model *entities* [5]. The use of IR-based solutions for matching knowledge-based entities is a recent trend that has already shown promising results in the ontology matching [ES07] field [Dr15] and in the process matching one [We13].

The idea of the work is based on the construction and exploitation of a structured representation of the entity to map and of its "context", starting from the associated textual information. In case of ontologies, the notion of "context" refers to the set of concepts that are directly connected (via a "is-a" property) to the concept to map, or that have a distance from it (in terms of "is-a" relations to traverse) lower than a certain degree. When considering processes, the semantics of "context" has to be revised. In the proposed implementation, the "context" of a process entity is the set of entities that are *directly connected* to it, i.e., for which there exists a path in the process model that does not pass through any other entity.

[5] Here on we use the term *entity* in a wider sense to denote process model flow elements that do not control the flow, e.g., activities and events in BPMN, transitions in Petri-Nets, functions and events in EPC.

In the current prototype, only flow elements that do not control the flow of the process model diagram (e.g., activities and events) have been considered, abstracting from other flow elements (e.g., BPMN gateways and Petri-Net conditions).

3.3.2 Specific Techniques

The matching operation is performed in two different steps: (i) creation of an index containing a structured description of each entity, and (ii) retrieval procedure for finding candidate matches.

Index Creation The index creation phase consists in exploiting information about entities and their "contexts" for building an inverted index for each process model to be matched (i.e., for each process in the challenge dataset). To this aim, for each process and for each entity of the process, the system extracts: (i) the entity label; (ii) the set of labels of the entities that *directly precede* the current one (`inputlabel`) if any; and (iii) the set of labels of the entities that *directly follow* the current one (`outputlabel`), if any. Intuitively, an entity e_1 directly precedes an entity e if there exists a path from e_1 to e (and no other entity occurs in the path). Similarly, an entity e_2 directly follows an entity e if there exists a path from e to e_2 (and no other entity occurs in the path). In the current implementation the system explores only the set of entities that directly precede and follow the current entity. In the future more sophisticated techniques will be investigated for improving the effectiveness of the system.

Once the information has been extracted, the textual information contained in each label is processed in order to obtain the lemmatized version of each textual token and the structured representation of each entity is built (Fig. 3) and indexed.

```
label: entity_label
inputlabel: input_label_1, ..., input_label_n
outputlabel: output_label_1, ..., output_label_n
```

Fig. 3: Entity structured representation

Match Search The matching operation inherits part of the procedure adopted for creating the index. Given two processes that have to be mapped (for example "Process 1" and "Process 2"), the structured representation of each entity of "Process 1" is transformed in a query performed on the indexes of the entities of the other process. The matching operation between two processes consists in performing queries by using entities of "Process 1" on the index of entities of "Process 2" and vice versa. Once all queries in both directions have been performed, the two sets of identified matches (M_{12} and M_{21}) are analyzed to compute the set M of the best matches, i.e., the set of matches that will be returned by the system.

To this purpose, the following three rules are applied by the system in the given order:

1. if a match m is identified for a given entity in both sets ($m \in M_{12}$ and $m \in M_{21}$), it is automatically stored in M;

2. if a match m is identified for a given entity only in one set (either $m \in M_{12}$ or $m \in M_{21}$), if the confidence score (computed during the retrieval) is higher than a threshold $th = 0.75$, the match is automatically stored in M;

3. if an entity is matched with several entities but none of the two conditions above apply (i.e., none of the matches is present in both sets), the two matches with the highest confidence score from $M_{12} \cup M_{21}$ are stored in M.

Purpose of the second and the third rules is avoiding to have a too restrictive system. The set of the best matches M is finally stored in the output file.

3.4 Match-SSS and Know-Match-SSS

3.4.1 Overview

The Match-SSS (MSSS) system uses NLP techniques to normalize the activity descriptions of the two models to be matched. It first uses string-based and WordNet-based algorithms. Finally, the approach selects the similarities calculated by these two matchers based on a maximum strategy with a threshold to identify equivalent activities. The Know-Match-SSS (KMSSS) system is similar, but uses another technique based on the category of words.

3.4.2 Specific Techniques

Extraction and Normalization The systems take as input the two process models to be matched and extract their labels. Then, NLP [Ma14] techniques are applied to normalize these labels. In particular, three preprocessing steps are performed: (1) case conversion (conversion of all words in same upper or lower case) (2) lemmatization stemming and (3) stop word elimination. Since String and WordNet based algorithms are used to calculate the similarities between labels, these steps are necessary.

Similarity Calculation In this step, both approaches calculate the similarities between the normalized labels using various base matchers. More precisely, the edit distance as string-based algorithm and the Lin algorithm [Li98] for WordNet-based similarity are applied. The Know-Match-SSS additionally uses another matcher based on the category of words. This matcher calculates the similarities between words based on their categories using a dictionary.

Aggregation and Identification In this step, our two systems select the similarity values calculated by different matchers using the maximum strategy. Finally, we apply a filter on similarity values retained in order to select the correspondences (equivalent activities between the two models) using a threshold.

Implementation To parse the process models, we used the jDOM API. For the normalization step, we made use of the Stanford CoreNLP API. To implement our matcher, we used the edit distance and the Lin WordNet-based Similarity. The retained similarity between words of a sentence is based on a maximum strategy.

3.5 RefMod-Mine/VM^2

3.5.1 Overview

The RefMod-Mine/VM^2 approach to business process model matching presented in the following is a refinement of our concept outlined in [NH15]. It focuses on the labels of a process model to determine mappings between activities based on their textual similarity. Therefore, established techniques from the field of Information Retrieval are combined with Natural Language Processing (NLP) to leverage information from text statistics.
As a preparatory step, every model to be compared is imported and transformed into a generic model format, where importers for BPMN, EPC and Petri-Nets are provided. As the notion of distinct 1:1 matches – i. e. a node label from a model A cannot be mapped to more than one node label from a model B – is underlying, possible multiple matches are removed from the final mapping as a last step.

3.5.2 Specific Techniques

The general procedure is defined by a three-step process, which is referred to as *multi-stage matching approach*. This process is carried out on each pairwise combination of all node labels that constitute the process models that are to be compared. A subsequent stage is only reached if the proceeding stage does not determine an appropriate match.

Trivial Matching First, a *trivial matching* is performed to identify identical labels as well as labels that are substrings of each other. Since this kind of similarity is valued most important, it constitutes the first step in our approach. Two labels A and B are considered "similar" if either $A == B$ or $A \; is \; substring \; of \; B \;||\; B \; is \; substring \; of \; A$.

Lemma-based Matching As an extension to the trivial matching approach, labels are further processed by NLP methods to harmonize the set of label terms and, thus, reach a

higher level of abstraction. First we split labels into constituting words – so-called *tokens* – and subsequently perform *lemmatization* on those tokens to unify different inflected word forms. Labels are then compared based on their set of lemmas, i. e. the intersection of terms in the label lemma sets is computed while abstracting from a specific word order (*bag of words* [Wa06]). In order to ensure high precision during matching, lemma sets may only differ by a small amount of terms (parameter i) and must have a certain length (parameter j) to be considered a match. The ratio between identical lemmas and absolute lemma set size depicts another threshold (parameter t_1). Values of i, j and t_1 have been determined iteratively using the provided gold standards with respect to high precision. As this stage only aims to identify "mostly identical" labels with a different order or inflection of words, thresholds are set very tight.

Vector-based detail matching At the centerpiece of this contribution is a *vector space model* (VSM) approach that enables both the retrieval of similar models to a given query model as well as the calculation of similarities between labels within these models. This procedure is three-part: *First*, for each combination of two models that have to be matched, the *k-nearest neighbors* (k-NN) are determined per model [CD07]. This is done by computing the *cosine similarity* between the vectors spanning across all lemmas within the set of all process models with respect to the particular query model. *Second*, label vectors are built per label pair combination within the two models to be matched, i. e. the number of dimensions of these vectors equals the sum of distinct lemmas in the two labels. To weight vector dimensions, the k-NN set is considered a new sub-corpus, which is in turn used to calculate *tf-idf* values for every label term lemma t in document d in corpus D according to formula (1) [Ra03].

$$tfidf(t,d,D) = tf(t,d) \times idf(t,D) = count(t \in d) \times \log \frac{|D|}{df(t,d)} \qquad (3)$$

Third, cosine similarity sim_{cos} is then calculated between label vectors and checked against a predefined threshold (parameter t_2) as depicted in formula (2).

$$t_2 \quad \leq \quad sim_{cos}(\theta) = \frac{\sum_{i=1}^{n} v_i \times w_i}{\sqrt{\sum_{i=1}^{n} (v_i)^2} \times \sqrt{\sum_{i=1}^{n} (w_i)^2}} \qquad (4)$$

By using this approach as a third stage in the overall matching procedure, the approach seeks to exploit statistical information from word occurrences and, thus, to reflect the importance of specific terms within the corpus. By including the k-NN of a model it further seeks to broaden the scope of consideration in order to obtain significant information about term distributions.

3.6 RefMod-Mine/NHCM

3.6.1 Overview

This matcher enhances the RefMod-Mine/NSCM approach presented at the PMC 2013 and consists of 3 general phases. In the pre-processing phase (1), the input models are transformed into a generic format, which allows an application of the matching approach to models of different modeling languages. In the processing phase (2), all available models of a dataset are used as an input for the *n-ary cluster matcher*, which uses a *natural language based similarity measure* for a pairwise node comparison. As a result, several sets of clusters containing nodes of all considered models are produced, which are then being extracted to *binary complex mappings* between two models. Finally, that binary complex mappings are being post-processed (3) in order to eliminate non corresponding maps resulting from the clusters.

The technique has been implemented in the form of a php command line tool and can publicly checked out at https://github.com/tomson2001/refmodmine. It is also available as an online tool in the context of the *RefMod-Miner as a Service* at http://rmm.dfki.de.

3.6.2 Specific Techniques

In the pre-processing phase of the approach, the input models are *transformed* into a generic format which constructs are similar to the extended EPC. At the moment, the transformation of BPMN, Petri-Net and EPCs is supported, whereby it is generally tried to lose as few information as possible. Especially in the case of BPMN it is important to keep the information caused by the variety of constructs, since they might be very useful in the context of process matching. Additionally, the pre-processing phase contains a *semantic error detection*, where defects of modeling are being identified and automatically corrected. This also includes a mechanism modifying the models concerning a consistent modeling style within a dataset and the solution of abbreviations, which are learned from the dataset.

The processing phase consists of the following components.

N-Ary cluster matching In contrast to existing matching techniques, the authors use a 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 [JMF99], it starts with clusters of size 1 (=node) and consolidates two nodes to a cluster if their similarity is approved by the matching algorithm. 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 [JMF99].

Semantic similarity measure The used similarity measure consists of three phases. The first phase splits node labels L into single words (stop word are being removed) w_{i_L}, so that $split(L) = \{w_{1_L}, ..., w_{n_L}\}$. The second phase computes the Porter Stem [Po97] $stem(w_{i_L})$ 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 $sim(L_1, L_2)$ passes a user-defined threshold, the labels are being checked for antonyms using the lexical database WordNet [Mi95] and checking the occurrence of negation words like "not".

Homogeneity-based detail matching Since the *semantic similarity measure* is not able to match synonyms, it is necessary to apply an additional technique. Based on the homogeneity degree of the model set, it is decided, whether and which further node pairs are being considered as potentially equivalent. The homogeneity degree is defined as:

$$HD = \frac{|multi.occuring.label| - |min.multi.occuring.label|}{|max.multi.occuring.label| - |min.multi.occuring.label|}$$

with $|multi.occuring.label|$ is the number of different node labels occuring in at least two models, $|max.multi.occuring.label|$ is the number of all nodes minus the number of different node labels and $|min.multi.occuring.label| = \frac{2*|min.multi.occuring.label|}{|num.epcs.in.dataset|}$.

The potential node pairs are now analyzed in detail. It is checked whether verb, object and further elements of the labels are equivalent by using WordNet [Mi95] and Wiktionary [6].

Binary matching extraction 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. This returns a binary complex (N:M) mapping for each model pair.

Since the matching approach might produce transitive correspondences over several models which are not meaningful in all cases, the binary mappings are additionally checked for antonyms and verb-object-correspondences using WordNet and Wiktionary as well as for organizational mismatches. Therefore, the bag-of-words [Kl13] of the nodes related to the organizational units are being calculated in order to match the organization units. Finally and depending on the *homogeneity degree*, the arity of the complex matches is being justified. This bases on the assumption, that a growing homogeneity degree leads to a reduction of the mapping complexity (the arity). Thus, the models describe the processes on a similar granularity.

[6] http://www.wiktionary.org

3.7 RefMod-Mine/NLM

3.7.1 Overview

The Natural Language Matcher (NLM) identifies corresponding process model labels and consequently corresponding nodes. It is predominantly based on natural language processing techniques using a bag of words concept. In contrast to the existing bag of words matching approach [Kl13], the NLM makes use of word classification. The matcher is capable of identifying simple matches as well as complex matches between two process models. Since the approach mainly relies on the labels used in process models, it can be applied to any kind of process modeling language. The matcher is implemented in Java 1.8 and embedded in the *RefMod-Mine*[7] toolset.

3.7.2 Specific Techniques

The approach is divided into two major steps. In the first step the natural language that is contained in the labels is processed. This includes a tokenization of the labels to identify the words contained in a label, a part-of-speech analysis to determine the syntactic category, and a lemmatization of the identified words. The second step represents the actual matching. Given two models M_1 and M_2 with their respective node sets N_1 and N_2. Based on the node types and the extracted linguistic information of step one, the matcher decides in the second step which pairs $(n_1, n_2) \in N_1 \times N_2$ are considered a match.

At first, the matcher checks the feasibility of a node pair. A node pair is considered feasible if the node types are marked as corresponding. These type correspondences can be parametrized and unless otherwise specified only identical node types are considered corresponding. Let NN be the list of nouns, VB the list of verbs, and JJ the list of adjectives that a label l can contain. A feasible node pair (n_1, n_2) is considered a match if their labels l_1, l_2 containing the word lists NN_1, VB_1, JJ_1 and NN_2, VB_2, JJ_2 meet at least one of the conditions listed:

- *identical condition*
  - each noun of NN_1 corresponds to at least one noun of NN_2 and vice versa
  - each verb of VB_1 corresponds to at least one verb of VB_2 and vice versa
  - each adjective of JJ_1 corresponds to at least one adjective of JJ_2 and vice versa
- *cross-category condition*
  - l_1 only contains one adjective or one verb and l_2 contains at most two words of which at least one word is a noun that corresponds to the single word contained in l_1, or

[7] http://refmod-miner.dfki.de

– l_2 only contains one adjective or one verb and l_1 contains at most two words of which at least one word is a noun that corresponds to the single word contained in l_2

The conditions are based on the assumption that identical nodes share the same nouns, verbs and adjectives. However, similar nodes might only share a subset of words in their respective word categories. Therefore, the cross-category condition is applied. Independent of the word category the lexical relation between two words determines their correspondence. The words w_1, w_2 correspond if their lemmata meet at least one of the following conditions:

- w_1 is identical to w_2
- w_1 is a synonym of w_2 or w_2 is a synonym of w_1
- w_1 is a hyponym of w_2 or w_1 is a hyponym of w_2
- w_1 is an etymologically related term of w_2 or w_2 is an etymological related term of w_1

Beside the identity relation, a synonym and a hyponym relation are considered appropriate lexical relations to determine similar words. The etymological relation is primarily used to determine similar words of different word categories.

3.7.3 Implementation

The presented approach uses the *Stanford CoreNLP API*[8] [Ma14] for Java to perform the language processing. The matcher determines the lexical relations based on *Wiktionary*[9] and the *Java-based Wiktionary Library*[10].

3.8 RefMod-Mine/SMSL

3.8.1 Overview

RefMod-Mine/SMSL is a semantic matching algorithm based on a supervised machine learning approach. The approach consists of two stages: (1) First the algorithm is given a repository of process models and its gold standard. The algorithm identifies the tokens of the process labels and determines their tags (verb, noun, ...). Then it performs a search for semantically related words in the Wordnet [Mi95]. As a measure of the quantification of the semantic relation of two words, a composed function is used that depends on the

[8] http://nlp.stanford.edu/software/corenlp.shtml
[9] https://en.wiktionary.org
[10] https://www.ukp.tu-darmstadt.de/software/jwktl

semantic distance between both words and the intermediate words in Wordnet. All tags and the semantic distance are weighted. When the algorithm calculated all semantic relations as matchings, it stores all weights of the function and the reached precision, recall and F-value. These weights are then optimized by the resulting F-value in a local search. (2) When the weights have been stored/learned, the algorithm applies the best found weights on new given matchings.

3.8.2 Specific Techniques

(1) At the beginning, the node labels of the process models are divided in tokens by the Stanford tokenizer [Ma14]. The tokens are lemmatised so that grammatical forms are neutralized. Then for each token its tag is determined by the Stanford tagger [To03]. After all tokens with their tags are determined, a similarity function is defined. This function calculates the similarity between two tokens $t1, t2$ and is composed of the LIN score by [Li98], the path length between $t1, t2$ in Wordnet and the weights of the tokens tags. More exactly, the similarity between tokens $t1, t2$ is equal to $weight\_LIN * LIN(t1,t2)$ + $weight\_pathLen$ * $pathLength(t1,t2)$ + $weight\_tag(t1)$ + $weighted\_tag(t2)$ with $weighted\_tag(token)$ = $weight\_tag(getTagFromToken(token))$. Each tag has its own weight. So a verb can have another weight than a noun or a gerund.

RefMod-Mine/SMSL seeks to find the weights that reach the highest F-value by local search. Therefore the algorithm calculates the token similarity function with different weight combinations and records the associated F-value. First the weights are defined with a wide range and then the weight combination with the highest F-value is the basis for refining the weights until no better F-value appears. (2) Then the algorithm has completed and can now apply its learned weights on new matchings.

3.8.3 Implementation

The matching algorithm itself has been implemented in Java 1.8 and the local search has been implemented in Python 2.7.9.

3.9 OPBOT

3.9.1 Overview

The ***O**rder **P**reserving **B**ag-**O**f-Words **T**echnique* (OPBOT) is based on two cornerstones: improved label matching and order preservation. To *improve the effectiveness of label matching*, we first identify equally labeled activities and then reduce the level of detail in the remaining labels. The former is motivated on the observation that equally labeled activities most often constitute 1:1-correspondences. The latter builds upon our previous work [Kl13] where label pruning was used to increase the recall. Here, we employ our

maximum-pruning bag-of-words similarity (MPB) that performed well in the previous iteration of the matching contest [Ca13a]. *Order preservation* builds on the idea that multiple correspondences between two models occur in the same order in both models. To this end, we employ the *relative start node distance* (RSD) [Kl14]. OPBOT processes all model pairs in a single run. It is based on the general matching workflow for schema matching [Ra11]. Below we discuss the processing steps in more detail.

3.9.2 Specific Techniques

In the pre-processing step, the process models from the collection are loaded as business process graphs [Di09]. Then, label normalization, tokenization, and stemming[11] are applied to transform each label into a bag-of-words. Finally, we count how many times two words co-occur in the same bag-of-words.

Next, a filter assigns a similarity score of 1 to all equally labeled activity pairs. In case an activity is part of such an activity pair, a similarity value of 0 is assigned to any other activity pair that includes this activity. A second filter assigns a value of 0 to all remaining activity pairs whose RSD difference yields an absolute value of at least 0.5. The RSD difference of two activities a and b is defined as $\Delta_{RSD}(a,b) := RSD(a) - RSD(b)$.

For the activity pairs with a similarity of neither 0 nor 1, three matchers then calculate similarity scores independently – resulting in three alignments per model pair. All matchers rely on the MPB, but employ different word similarity measures and threshold values t. Given an activity pair, each matcher computes a similarity score. If it is greater or equal to the respective t, it will be assigned to the activity pair; 0 otherwise. The *syntactic matcher* uses the longest common subsequence similarity [ES07], with $t = 0.76$ in the experiments. The *paradigmatic sense relation matcher* is based on Lin's similarity metric [Li98] and WordNet [Mi95], with $t = 0.76$. The *syntagmatic sense relation matcher* utilizes the co-occurrence counts from the pre-processing. To determine the similarity for a pair of words it identifies – for each word individually – the two most frequently co-occurring words and then calculates the cosine co-occurence similarity [Na09], with $t = 0.84$.

Next, the three matchers are ranked based on their *order preservation score* (OPS). In the calculation of OPS, only correspondences (activity pairs with a similarity score not equal to 0) are considered. A pair of correspondences $((a_1,b_1),(a_2,b_2))$ yields an OPS of 1 if it is order preserving, i.e., $\Delta_{RSD}(a_1,a_2)$ and $\Delta_{RSD}(b_1,b_2)$ are either both positive or negative; and 0 otherwise. The OPS is determined for all possible correspondence pairs and averaged per alignment. Then, the overall OPS for a matcher is the average of the OPSs of it's proposed alignments. The correspondences proposed by the matcher with the highest overall OPS are chosen, along with the according similarity scores. Each correspondence that was not selected, but is in the intersection of the results of the other two matchers. Its similarity is the maximum score yielded by any matcher.

[11] We use the stemmer from the JWI library (`http://projects.csail.mit.edu/jwi/`).

Subsequently, the resulting alignments are revised. For data sets that includes roles, pools, or lanes – like the University Admission data set – a first filtering step removes correspondences where the respective roles mismatch. That is, all correspondences where the role names do not contain at least one overlapping word are removed. Afterwards, the alignments are optimized by a greedy algorithm that maximizes the average similarity of the correspondences and the average OPS for each alignment. It iterates over each correspondence and computes both scores in case the correspondence is removed. The correspondence that improves the scores from the previous iteration is removed. The algorithm will stop once a fixpoint is reached, i.e., there is no further improvement. All remaining correspondences are then returned as the final alignment.

Acknowledgements. This work was partly funded by the German Federal Ministry of Education and Research under the project LSEM (BMBF 03IPT504X). NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

3.10 pPalm-DS

3.10.1 Overview

With this approach, we provide a rudimentary basis for process matching, concentrating on finding an alignment between activities (nodes) with semantic similar labels. We do not consider information like structure, behaviour or different node types within the process models. In a nutshell, from each process p, we retrieve the set of relevant nodes, hereafter called *activities* (node types used for the alignment in the according gold standard). From the set of activities we obtain the according set of labels $l \in L_p$. To compute the matches of a process-pair (p_1, p_2), we compare each label $l \in L_{p_1}$ to each label $l' \in L_{p_2}$ by a similarity function. If the similarity of both labels is equal or greater than a certain threshold, $(sim(l,l') \geq threshold)$, we include the corresponding activity-pair to the set of matches.

3.10.2 Specific Techniques

For deriving $sim(l,l')$ we use, differing from most of the existing approaches, a vector based approach from the field of distributional semantics. The idea behind distributional semantics is the *distributional hypothesis*: "[...] words that occur in similar contexts tend to have similar meaning."[Pa05]. To be prepared for process models of different domains, we need a large as possible cross-domain set of words with corresponding contexts. A broad coverage of label terms, independent from the basing domain, we ensure by using a corpus of 7.8B words which we derived from Gigaword [Pa11] and the contents of the English Wikipedia (Version 20140102). We utilised `word2vec`[12] to extract semantic relationships between words and their contexts. Therefore `word2vec` uses a local context window to capture co-occurrences of words [Mi13]. For each word having a sufficient number of

[12] `https://code.google.com/p/word2vec/`

occurrences within the corpus, this context information is concentrated to a semantic vector having 300 contextual-dimensions. After we trained `word2vec` on the corpus mentioned before, the resulting database consists of more than 1.6m 300-dimensional semantic vectors.

Finally, we compute $sim(l,l')$ as follows: Given a process label l consisting of words $w_1,\ldots,w_n$, for each word w_i we collect its vector $\mathbf{x}_{w_i}$ from the database and we perform the element-wise sum to obtain one final vector $\mathbf{x}_l$ for the label. Words missing in the database we treat as null-vectors in the calculation. Given two labels l and l', we derive similarity by taking the respective final vectors for computing cosine similarity (see [MRS08]):

$$sim(l,l') = \cos(\theta) = \frac{\mathbf{x}_l \cdot \mathbf{x}_{l'}}{\|\mathbf{x}_l\|\|\mathbf{x}_{l'}\|} = \frac{\sum_{i=1}^{n} \mathbf{x}_{l,i} \times \mathbf{x}_{l',i}}{\sqrt{\sum_{i=1}^{n} \mathbf{x}_{l,i}^2} \times \sqrt{\sum_{i=1}^{n} \mathbf{x}_{l',i}^2}} \tag{5}$$

We include all label pairs having $sim(l,l') \geq threshold$ to the final alignment. For this matching contest, we used a threshold of 0.77 which performed best according to the combination of dataset1 and dataset2.

Finally it should be remarked that this approach is not intended as standalone matcher. Rather it aims at being used as basis for further alignments respecting structure, behaviour and different node types within process models.

3.11 TripleS

3.11.1 Overview

The matching approach used in the second Process Model Matching Contest in 2015 is essentially the same as the one used in 2013. The Triple-S matching approach [Ca13b] still adheres to the KISS principle by avoiding complex matching techniques and *keeping it simple and stupid*. This years version has been extended to match not only transitions in Petri-Nets but also functions of EPC models and tasks of models in BPMN notation, i.e. the "active" components of process models are matched.

3.11.2 Specific Techniques

The following three levels and scores are considered:

- **Syntactic level -** $SIM_{syn}(a,b)$**:** For the syntactic analysis of active components labels we perform two preprocessing steps: (1) tokenization and (2) stop word elimination. The actual analysis is based on the calculation of Levenshtein [Le66] distances between each combination of tokens (i.e. words) from the labels of active components 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 -** $SIM_{sem}(a,b)$**:** First, we perform the same preprocessing steps as mentioned above. Subsequently, we apply the approach of Wu & Palmer [WP94] to calculate the semantic similarity between each token of labels of active components a and b based on path length between the corresponding concepts. The final semantic score is the maximum average similarity analogous to the final syntactic score.
- **Structural level -** $SIM_{struc}(a,b)$**:** At this level, we investigate the similarity of active components a and b through a comparison of (i) the ratio of their in- and outgoing arcs and (ii) their relative position in the complete model. The two values are combined through the calculation of a weighted average.

These three scores are combined to the final score $SIM_{total}(a,b)$ which represents the matching degree between two active components 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 active components actually match, i.e. iff $SIM_{total} \geq \theta$, two transitions positively match.

3.11.3 Implementation

The Triple-S approach has been implemented using Java. For the calculation of the semantic score with the apporach of Wu & Palmer, the *WS4J Java API*[13] has been used to query Princeton's English *WordNet* 3.0 lexical database [Mi95]. Relative positions of transitions are calculated using the implementation of Dijkstras algorithm by Vogella[14].

During our experiments we tried to approximate optimal results based on the gold standard examples. For the contest, we have used the following values: $\omega_1 = 0.5$, $\omega_2 = 0.35$, $\omega_3 = 0.15$ and $\theta = 0.7$. Thereby, the weights for $SIM_{struc}(a,b)$ have been set to 0.25 for value (i) and 0.75 for value (ii).

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

4 Results

For assessing the submitted process model matching techniques, we compare the computed correspondences 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

[13] `https://code.google.com/p/ws4j/`
[14] `http://www.vogella.com/articles/JavaAlgorithmsDijkstra/article.html`

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)).

Tables 3 to 6 give an overview of the results for the datasets. For getting a better understanding of the result details, we report the average (∅) and the standard deviation (SD) for each metric. The highest value for each metric is marked using bold font. In our evaluation we distinguish between micro and macro average. Macro average is defined as the average of precision, recall and f-measure scores over all testcases. On the contrary, micro average is computed by summing up TP, TN, FP, and FN scores applying the precision, recall and f-measure formula once on the resulting values. Micro average scores take different sizes of test cases into account, e.g., bad recall on a small testcase has only limited impact on the micro average recall scores.

Some agreements are required to compute macro average scores for two special cases. It might happen that a matcher generates an empty set of correspondences. If this is the case, we set the precision score for computing the macro average to 1.0, due to the consideration that an empty set of correspondences contains no incorrect correspondences. Moreover, some of the testcases of the AM data set have empty gold standards. In this case we set the recall score for computing the macro average to 1.0, because all correct matches have been detected.

	Precision			**Recall**			**F-Measure**		
Approach	∅-mic	∅-mac	SD	∅-mic	∅-mac	SD	∅-mic	∅-mac	SD
RMM/NHCM	.686	.597	.248	.651	.61	.277	**.668**	.566	.224
RMM/NLM	.768	.673	.261	.543	.466	.279	.636	.509	.236
MSSS	**.807**	**.855**	.232	.487	.343	.353	.608	.378	.343
OPBOT	.598	.636	.335	.603	.623	.312	.601	**.603**	.3
KMSSS	.513	.386	.32	.578	.402	.357	.544	.374	.305
RMM/SMSL	.511	.445	.239	.578	.578	.336	.543	.477	.253
TripleS	.487	.685	.329	.483	.297	.361	.485	.249	.278
BPLangMatch	.365	.291	.229	.435	.314	.265	.397	.295	.236
KnoMa-Proc	.337	.223	.282	.474	.292	.329	.394	.243	.285
AML-PM	.269	.25	.205	**.672**	**.626**	.319	.385	.341	.236
RMM/VM2	.214	.186	.227	.466	.332	.283	.293	.227	.246
pPalm-DS	.162	.125	.157	.578	.381	.38	.253	.18	.209

Tab. 3: Results of University Admission Matching

The results for the UA data set (Table 3) illustrate large differences in the quality of the generated correspondences. Note that we ordered the matchers in Table 3 and in the other results tables by micro average f-measure. The best results in terms of f-measure (micro-average) are obtained by the RMM/NHCM approach (0.668) followed by RMM/NLM (0.636) and MSSS (0.608). At the same time three matching systems generate results with an f-measure of less than 0.4. When we compare these results against the results achieved in the 2013 edition of the contest, we have to focus on macro-average scores, which have been computed also in the 2013 edition. This year, there are several matchers with a macro average of >0.5, while the best approach achieved 0.41 in 2013. This improvement indicates that the techniques for process matching have progressed over the last two years. Anyhow, we also have to take into account that the gold standard has been improved and the format of the models has been changed to BPMN. Thus, results are only partially comparable.

Comparing micro and macro f-measure averages in 2015, there are, at times, significant differences. In most cases, macro scores are significantly lower. This is caused by the existence of several small testcases (small in numbers of correspondences) in the collection that seem to be hard to deal with for some matchers. These testcases have a strong negative impact on macro averages and a moderated impact on micro average. This is also one of the reasons why we prefer to discuss the results in terms of micro average.

It interesting to see that the good results are not only based on a strict setting that aims for high precision scores, but that matchers like RMM/NHCM and OPBOT manage to achieve good f-measure scores based on well-balanced precision/recall scores. Above, we have described the gold standard of this data set as rather strict in terms of 1:n correspondences. This might indicate that the matching task should not be too complex. However, some of the approaches failed to generate good results. Note that this is caused by a low precision, while at the same time recall values have not or only slightly been affected positively. A detailed matcher specific analysis, that goes beyond the scope of this paper, has to reveal the underlying reason.

	Precision			**Recall**			**F-Measure**		
Approach	∅-mic	∅-mac	SD	∅-mic	∅-mac	SD	∅-mic	∅-mac	SD
RMM/NHCM	**.855**	**.82**	.194	.308	.326	.282	**.452**	**.424**	.253
OPBOT	.744	.776	.249	.285	.3	.254	.412	.389	.239
RMM/SMSL	.645	.713	.263	.277	.283	.217	.387	.36	.205
KMSSS	.64	.667	.252	.273	.289	.299	.383	.336	.235
AML-PM	.385	.403	.2	**.365**	**.378**	.273	.375	.363	.22
KnoMa-Proc	.528	.517	.296	.282	.281	.278	.367	.319	.25
BPLangMatch	.545	.495	.21	.247	.256	.228	.34	.316	.209
RMM/NLM	.787	.68	.267	.211	.229	.308	.333	.286	.299
MSSS	.829	.862	.233	.19	.212	.312	.309	.255	.318
TripleS	.543	.716	.307	.205	.224	.336	.297	.217	.284
RMM/VM2	.327	.317	.209	.27	.278	.248	.296	.284	.226
pPalm-DS	.233	.273	.163	.316	.328	.302	.268	.25	.184

Tab. 4: Results of University Admission Matching with Subsumption

The results for the UA data set where we used the extended gold standard including subsumption correspondences are shown in Table 4. Due to the experimental status of this gold standard the results shown are thus less conclusive. However, we decided finally to include these results because subsumption correspondences will often occur when two process models differ in terms of granularity. A comparison against the strict version of the gold standard (Table 3) reveals that there are some slight changes in the f-measure based ordering of the matchers. OPBOT climbs up from rank #4 to rank #2, AML-PM climbs from up from rank #10 to rank #5, while other matchers are only slightly affected. This shows that some of the implemented methods can be used to detect subsumption correspondences, while other techniques are in particular designed to focus on direct 1:1 correspondences only.

The BR data set has not been modified compared to its 2013 version. Thus, we can directly compare the 2015 results against the 2013 results. Again, we have to focus on the macro average scores. In 2013, the top results were achieved by RefMod-Mine/NSCM with an macro average f-measure of 0.45. In 2015 the best performing matcher on this data set is the

Approach	Precision			Recall			F-Measure		
	∅-mic	∅-mac	SD	∅-mic	∅-mac	SD	∅-mic	∅-mac	SD
OPBOT	.713	.679	.184	**.468**	**.474**	.239	**.565**	**.54**	.216
pPalm-DS	.502	.499	.172	.422	.429	.245	.459	.426	.187
RMM/NHCM	.727	.715	.197	.333	.325	.189	.456	.416	.175
RMM/VM2	.474	.44	.2	.4	.397	.241	.433	.404	.21
BPLangMatch	.645	.558	.205	.309	.297	.22	.418	.369	.221
AML-PM	.423	.402	.168	.365	.366	.186	.392	.367	.164
KMSSS	.8	.768	.238	.254	.237	.238	.385	.313	.254
RMM/SMSL	.508	.499	.151	.309	.305	.233	.384	.342	.178
TripleS	.613	.553	.26	.28	.265	.264	.384	.306	.237
MSSS	**.922**	**.972**	.057	.202	.177	.223	.332	.244	.261
RMM/NLM	.859	.948	.096	.189	.164	.211	.309	.225	.244
KnoMa-Proc	.234	.217	.188	.297	.278	.234	.262	.237	.205

Tab. 5: Results of Birth Certificate Matching

OPBOT approach with macro average f-measure of 0.54, which is a significant improvement compared to 2013. The systems on the follow-up positions, which are pPalm-DS (0.426), RMM/NHCM (0.416), and RMM/VM2 (0.402), could not outperform the 2013 results. However, the average approach (≈0.35) in 2015 is clearly better than the average approach in 2013 (≈0.29), which can be understood as an indicator for an overall improvement.

While it is possible for the UA data set to generate high f-measures with a balanced approach in terms of precision and recall, the BR data set does not share this characteristics. All matchers, with the exception of KnoMa-Proc, favor precision over recall. Moreover, a high number of non-trivial correspondences cannot be found by the participants of the contest. We conducted an additional analysis where we computed the union of all matcher generated alignments. For this alignment we measured a recall of 0.631. This means that there is a large fraction of non-trivial correspondences in the BR data set that cannot be found by any of the matchers. Note that we measured the analogous score also for the other data sets, with the outcome of 0.871 for the UA dataset (0.494 for the extended UA data set) and 0.68 for the AM data set. These numbers illustrate that the BR data set is a challenging data set, which requires specific methods to overcome low recall scores. This can also be the reason why some of the systems that perform not so well on the UA data set are among the top-5 systems for the BR data set. These systems are OPBOT, pPalm-DS, and RMM/VM2.

The results for the AM data set are presented in Table 6. The top performing matchers in terms of macro f-measure are AML-PM (0.677), RMM/NHCM (0.661), and RMM/NLM (0.653). While these systems are close in terms of f-measure, they have a different characteristics in terms of precision and recall. The two RMM-based systems have a high precision in common. Especially RMM/NLM has a precision of 0.991, which means that less than 1 out of 100 correspondences are incorrect. AML-PM, the top performing system, has only a precision of .786 and a (relatively high) recall of .595. It is notable that these results have been achieved by the use a standard ontology matching systems instead of using a specific approach for process model matching. For the details we refer the reader to the respective system description in the previous section. The best results in terms of recall have been

Approach	Precision			Recall			F-Measure		
	∅-mic	∅-mac	SD	∅-mic	∅-mac	SD	∅-mic	∅-mac	SD
AML-PM	.786	.664	.408	.595	.635	.407	**.677**	.48	.422
RMM/NHCM	.957	.887	.314	.505	.521	.422	.661	**.485**	.426
RMM/NLM	**.991**	**.998**	.012	.486	.492	.436	.653	.531	.438
BPLangMatch	.758	.567	.436	.563	.612	.389	.646	.475	.402
OPBOT	.662	.695	.379	**.617**	**.634**	.409	.639	.514	.403
MSSS	.897	.979	.079	.473	.486	.432	.619	.519	.429
RMM/VM2	.676	.621	.376	.545	.6	.386	.603	.454	.384
KMSSS	.643	.834	.282	.527	.532	.417	.579	.482	.382
TripleS	.614	.814	.261	.545	.546	.434	.578	.481	.389
pPalm-DS	.394	.724	.348	.595	.615	.431	.474	.451	.376
KnoMa-Proc	.271	.421	.383	.514	.556	.42	.355	.268	.279
RMM/SMSL	.722	.84	.307	.234	.37	.366	.354	.333	.327

Tab. 6: Results of Asset Management Matching

achieved by the OPBOT matcher (0.617). Looking at the recall scores in general, it can be concluded that it is hard to top a recall of 0.6 without a significant loss in precision.

The results of our evaluation show that there is a high variance in terms of the identified correspondences across the different data sets. However, there are also some systems that perform well over all three data sets (we exclude the UA_S data set in this consideration due to its experimental character). These systems are RMM/NHCM and OPBOT. RMM/NHCM is ranked #1, #3 and #2, OPBOT is ranked #4, #1, and #5 in terms of macro-average. None of the other approaches is among the top-five with respect to all three data sets. This illustrates again how hard it is to propose a mechanism that works well for the different modeling styles and labeling conventions that can be found in our test data collection.

5 Future Directions

6 Conclusion

In this paper, we reported on the setup and the results of the Process Model Matching Contest 2015. We provided three different process model matching problems and received automatically generated results of twelve different techniques. The high number of participants showed that there is a vivid process matching community that is interested in an experimental evaluation of the developed techniques to better understand its pros and cons. We are also happy that we were able to attract participants from the ontology community (e.g., AML-PM). We believe that both research fields (Process Model Matching and Ontology Matching) are closely related and can mutually benefit from each other in the future.

The results of our experimental evaluation show that there is an overall improvement compared to the results that have been achieved in 2013. This becomes obvious from the results of the UA and BR data set. The underlying reasons for these improvements cannot be detailed in the aggregated view that we presented in the results section. It requires a

detailed analysis of the techniques implemented by the top performing approaches which are presented in Section 3. However, the presented results can give some hints on the different characteristics of the proposed approaches, which helps to better understand the concrete impact on a given matching task.

The results show also that many proposed approaches do not generate good results for all data sets. An counterexample for our claim are the two matching systems RMM/NHCM and OPBOT. These two systems are among the top-5 systems for all data sets used in our evaluation. However, the results also show that the test data collection is heterogeneous in terms of specifics that need to be considered and problems that need to be solved for generating high quality correspondences. Especially the BR data set seems to be challenging. Here we discovered that more than 36% of all correspondences in the gold standard have not been generated by any of the participating matchers. This number shows that there is still large room for improvement related to methods that aim at a high recall without suffering too much in terms of precision.

For 2015 we did not define a strict set of rules for participation. However, this creates a certain bias in the comparison of the results. In particular, we have noticed that a wide range of different techniques have been proposed, including approaches that rely on joint matching of all process models from a data set as well as supervised machine learning techniques. All these techniques have been described precisely and in a transparent way. Yet, they postulate slightly different settings for process model matching, which are all reasonable from an application point of view, but shall be evaluated separately. That is, results obtained when relying solely on the two models to be matched may differ significantly from results obtained when considering a whole corpus of process models that should be aligned. Hence, in future editions, we plan to evaluate these scenarios separately.

For the future we consider establishing a fully automated evaluation procedure similar to the one that is applied since 2011/2012 in the context of the Ontology Alignment Evaluation Initiative [Ag12]. In such a setting, the matching tools are submitted to the organizers instead of submitting the generated correspondences. The submitted tools are then executed by the organizers in a controlled environment. Such an evaluation approach has several advantages. First, the generated results are a 100% reproducible and it can be guaranteed that no data set specific parameter settings have been chosen. Second, the matching tools themselves become available as executable tools. So far, they often represent academic prototypes that are not available to the public. We believe that this is an important step for the adoption of process matching tools to solve real world matching problems.

References

[Ag12] Aguirre, José Luis; Grau, Bernardo Cuenca; Eckert, Kai; Euzenat, Jérôme; Ferrara, Alfio; Van Hague, Robert Willem; Hollink, Laura; Jiménez-Ruiz, Ernesto; Meilicke, Christian; Nikolov, Andriy et al.: Results of the ontology alignment evaluation initiative 2012. In: Proc. 7th ISWC workshop on ontology matching (OM). No commercial editor., pp. 73–115, 2012.

[Br12] Branco, Moisés Castelo; Troya, Javier; Czarnecki, Krzysztof; Küster, Jochen Malte; Völzer, Hagen: Matching Business Process Workflows across Abstraction Levels. In

(France, Robert B.; Kazmeier, Jürgen; Breu, Ruth; Atkinson, Colin, eds): MoDELS. volume 7590 of Lecture Notes in Computer Science. Springer, pp. 626–641, 2012.

[Ca13a] Cayoglu, Ugur; Dijkman, Remco; Dumas, Marlon; Fettke, Peter; Garcıa-Banuelos, Luciano; Hake, Philip; Klinkmüller, Christopher; Leopold, Henrik; Ludwig, André; Loos, Peter et al.: The process model matching contest 2013. In: 4th International Workshop on Process Model Collections: Management and Reuse (PMC-MR'13). 2013.

[Ca13b] Cayoglu, Ugur; Oberweis, Andreas; Schoknecht, Andreas; Ullrich, Meike: Triple-S: A Matching Approach for Petri Nets on Syntactic, Semantic and Structural level. Technical report, 2013.

[CD07] Cunningham, P.; Delany, S. J.: k-Nearest neighbour classifiers. Multiple Classifier Systems, pp. 1–17, 2007.

[DGBD09] Dumas, Marlon; García-Bañuelos, Luciano; Dijkman, Remco M.: Similarity Search of Business Process Models. IEEE Data Eng. Bull., 32(3):23–28, 2009.

[Di09] Dijkman, R.; Dumas, M.; Garcia-Banuelos, L.; Kaarik, R.: Aligning Business Process Models. In: Enterprise Distributed Object Computing Conference. 2009.

[Dr14] Dragisic, Zlatan; Eckert, Kai; Euzenat, Jérôme; Faria, Daniel; Ferrara, Alfio; Granada, Roger; Ivanova, Valentina; Jiménez-Ruiz, Ernesto; Kempf, Andreas Oskar; Lambrix, Patrick et al.: Results of the ontology alignment evaluation initiative 2014. In: Proceedings of the 9th International Workshop on Ontology Matching Collocated with the 13th International Semantic Web Conference (ISWC 2014). 2014.

[Dr15] Dragoni, Mauro: Exploiting Multilinguality For Creating Mappings Between Thesauri. In: Proceedings of the 30th Annual ACM Symposium on Applied Computing. SAC 2015. ACM, pp. 382–387, 2015.

[Ek12] Ekanayake, Chathura C.; Dumas, Marlon; García-Bañuelos, Luciano; Rosa, Marcello La; ter Hofstede, Arthur H. M.: Approximate Clone Detection in Repositories of Business Process Models. In (Barros, Alistair P.; Gal, Avigdor; Kindler, Ekkart, eds): BPM. volume 7481 of Lecture Notes in Computer Science. Springer, pp. 302–318, 2012.

[ES07] Euzenat, Jérôme; Shvaiko, Pavel: Ontology Matching. Springer-Verlag New York, Inc, Secaucus, NJ, USA, 2007.

[Fa13] Faria, Daniel; Pesquita, Catia; Santos, Emanuel; Cruz, Isabel F; Couto, Francisco M: AgreementMakerLight results for OAEI 2013. In: OM. pp. 101–108, 2013.

[GS10] Gal, Avigdor; Sagi, Tomer: Tuning the ensemble selection process of schema matchers. Inf. Syst., 35(8):845–859, 2010.

[Ji13] Jin, Tao; Wang, Jianmin; Rosa, Marcello La; ter Hofstede, Arthur H.M.; Wen, Lijie: Efficient querying of large process model repositories. Computers in Industry, 64(1), 2013.

[JMF99] Jain, A.K.; Murty, M.N.; Flynn, P.J.: Data clustering: a review. ACM Computing Surveys (CSUR), 31:264–323, 1999.

[KKR06] Küster, Jochen Malte; Koehler, Jana; Ryndina, Ksenia: Improving Business Process Models with Reference Models in Business-Driven Development. In (Eder, Johann; Dustdar, Schahram, eds): Business Process Management Workshops. volume 4103 of Lecture Notes in Computer Science. Springer, pp. 35–44, 2006.

[Kl13] Klinkmüller, Christopher; Weber, Ingo; Mendling, Jan; Leopold, Henrik; Ludwig, André: Increasing Recall of Process Model Matching by Improved Activity Label Matching. In (Daniel, Florian; Wang, Jianmin; Weber, Barbara, eds): Business Process Management, volume 8094 of Lecture Notes in Computer Science, pp. 211–218. Springer Berlin Heidelberg, 2013.

[Kl14] Klinkmüller, Christopher; Leopold, Henrik; Weber, Ingo; Mendling, Jan; Ludwig, André: Listen to Me: Improving Process Model Matching through User Feedback. In: Business Process Management. pp. 84–100, 2014.

[KWW11] Kunze, Matthias; Weidlich, Matthias; Weske, Mathias: Behavioral Similarity - A Proper Metric. In (Rinderle-Ma, Stefanie; Toumani, Farouk; Wolf, Karsten, eds): BPM. volume 6896 of Lecture Notes in Computer Science. Springer, pp. 166–181, 2011.

[La13] La Rosa, Marcello; Dumas, Marlon; Uba, Reina; Dijkman, Remco: Business Process Model Merging: An Approach to Business Process Consolidation. ACM Trans. Softw. Eng. Methodol., 22(2):11:1–11:42, 2013.

[Le66] Levenshtein, Vladimir: Binary codes capable of correcting deletions, insertions, and reversals. Cybernetics and Control Theory, 10(8):707–710, 1966.

[Li91] Lin, Jianhua: Divergence measures based on the Shannon entropy. IEEE Transactions on Information Theory, 37(1):145–151, 1991.

[Li98] Lin, Dekang: An Information-Theoretic Definition of Similarity. In: International Conference on Machine Learning. pp. 296–304, 1998.

[LZ09] Lv, Yuanhua; Zhai, ChengXiang: Positional language models for information retrieval. In (Allan, James; Aslam, Javed A.; Sanderson, Mark; Zhai, ChengXiang; Zobel, Justin, eds): SIGIR. ACM, pp. 299–306, 2009.

[Ma14] Manning, Christopher D; Surdeanu, Mihai; Bauer, John; Finkel, Jenny; Bethard, Steven J; McClosky, David: The Stanford CoreNLP natural language processing toolkit. In: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations. pp. 55–60, 2014.

[Mi95] Miller, George A.: WordNet: A Lexical Database for English. Communications of the ACM, 38(11):39–41, 1995.

[Mi13] Mikolov, T.; Chen, K.; Corrado, G.; Dean, J.: Efficient Estimation of Word Representations in Vector Space. CoRR, abs/1301.3781, 2013.

[MRS08] Manning, C. D.; Raghavan, P.; Schütze, H.: Introduction to information retrieval. Cambridge University Press, New York, 2008.

[Na09] Navigli, Roberto: Word Sense Disambiguation: A Survey. ACM Computing Surveys, 41(2):10:1–10:69, 2009.

[NH15] Niesen, T.; Houy, C.: Zur Nutzung von Techniken der Natürlichen Sprachverarbeitung für die Bestimmung von Prozessmodellähnlichkeiten – Review und Konzeptentwicklung. In (Thomas, O; Teuteberg, F., eds): Proceedings der 12. Internationalen Tagung Wirtschaftsinformatik. Internationale Tagung Wirtschaftsinformatik (WI-15). Springer, Osnabrück, pp. 1829–1843, 2015.

[Pa05] Pantel, P.: Inducing Ontological Co-occurrence Vectors. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics. ACL '05, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 125–132, 2005.

[Pa11] Parker, R.: English gigaword fifth edition. Linguistic Data Consortium, [Philadelphia, PA], 2011.

[Po97] Porter, M.F.: An algorithm for suffix stripping. Readings in information retrieval, pp. 313–316, 1997.

[Ra03] Ramos, J.: Using tf-idf to determine word relevance in document queries. Proceedings of the first instructional conference on machine learning, 2003.

[Ra11] Rahm, Erhard: Towards Large-Scale Schema and Ontology Matching. In: Schema Matching and Mapping. pp. 3–27, 2011.

[To03] Toutanova, Kristina; Klein, Dan; Manning, Christopher D.; Singer, Yoram: Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In: IN PROCEEDINGS OF HLT-NAACL. pp. 252–259, 2003.

[Wa06] Wallach, H. M.: Topic modeling: beyond bag-of-words. Proceedings of the 23rd international conference on Machine learning, 2006.

[We13] Weidlich, Matthias; Sheetrit, Eitam; Branco, Moises; Gal, Avigdor: Matching Business Process Models Using Positional Language Models. In: 32nd International Conference on Conceptual Modeling, ER 2013. Hong Kong, 2013.

[WMW11] Weidlich, Matthias; Mendling, Jan; Weske, Mathias: A Foundational Approach for Managing Process Variability. In (Mouratidis, Haralambos; Rolland, Colette, eds): CAiSE. volume 6741 of Lecture Notes in Computer Science. Springer, pp. 267–282, 2011.

[WP94] Wu, Zhibiao; Palmer, Martha: Verbs Semantics and Lexical Selection. In: Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics. ACL '94, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 133–138, 1994.