AN APPROACH FOR SEMANTIC BUSINESS PROCESS MODEL MATCHING USING SUPERVISED MACHINE LEARNING

Research in Progress

Sonntag, Andreas, Institute for Information Systems, German Research Center for Artificial Intelligence, Germany, Andreas.Sonntag@iwi.dfki.de

Hake, Philip, Institute for Information Systems, German Research Center for Artificial Intelligence, Germany, Philip.hake@iwi.dfki.de

Fettke, Peter, Institute for Information Systems, German Research Center for Artificial Intelligence, Germany, Peter.Fettke@iwi.dfki.de

Loos, Peter, Institute for Information Systems, German Research Center for Artificial Intelligence, Germany, Peter.Loos@iwi.dfki.de

Abstract

Matching business process models and their node labels plays an important role for business process management. Many matching algorithms using natural language processing (NLP) techniques exist but do not exploit the opportunities of machine learning though it is generally agreed that a learning approach has great potential in the field of NLP. Therefore, we develop a matching approach based on supervised learning using a language-driven similarity function in order to reproduce a human judgement. Additionally, we implement and evaluate our approach using established quality measures, consisting of precision, recall and F-measure. We conduct an evaluation based on real world process models that demonstrates the potential and the limitations of our machine learning approach. The results show a significant learning effect for matching unknown models without predefined rules. The matching quality is comparable to existing matchers. However, the matching quality seems to depend on the one hand on the available training data and on the other hand on the complexity of the chosen similarity function. Further research efforts have to be undertaken in order to improve our approach. This will include developing a more elaborate similarity function containing more linguistic characteristics of a label as well as integrating contextual information.

Keywords: BPM, Machine learning, Process Model Matching

1 Introduction

Business process models (process models) represent valuable knowledge about the superstruction of business processes and their execution as a fundamental aspect of business process management. Nowadays organizations have to maintain huge repositories containing a plethora of these models. Thus, the analysis of the models is necessary for optimizing and managing the repositories. This particularly includes enabling reusability, modularity and avoiding redundancy. Hence, identifying correspondences between process models is of major importance. In literature, the technique of *business process matching* has been proven to be an adequate mean to identify such correspondences (Weidlich et al. 2010). This is also reflected by the manifold matching approaches which have been recently submitted to a contest where twelve different approaches competed in a matching challenge (Antunes et al. 2015). As in the preceding matching contest (Cayoglu et al. 2013) and further matching evaluations (Weidlich et al. 2010), the approaches have been measured via a *gold standard* which represents a set of reference correspondences derived by human judgement. The more correct correspondences, according to the gold standard, a matching approach identifies the better it is measured.

Machine learning approaches applied in *natural language processing (NLP)*, especially the fields of text recognition and word classification, have been proven effective (Cambria and White 2014; Sebastiani and Fabrizio 2002) in order to reproduce human judgments. Basically machine learning is the genus of the artificial inference of knowledge from experience. Supervised machine learning approaches let the machine learn from training data containing problem-solution pairs and then these approaches use the gathered knowledge to process unknown test data (Mohri et al. 2012). Since the gold standard, which is the preferred result of a good matching approach, is based on human judgement, we consider a machine learning based matching approach adequate in order to solve matching problems. We therefore propose a *supervised machine learning* approach using an *NLP-based similarity function*.

The research we conduct is based on a *design-oriented* approach, since we aim at creating an innovative artifact in form of a novel matching algorithm. We implement our algorithm and embed the matcher based on this algorithm in our toolset (http://refmod-miner.dfki.de/) to make the algorithm serve a variety of functionalities dedicated to solving BPM problem, e.g., integrating models, refactoring models and creating reference models inductively. Moreover, we evaluated the newly created artifact using established evaluation methods for matching algorithms. Additionally, by using the datasets of the most recent matching contest (Antunes et al. 2015) for evaluation, we apply our approach to real world scenarios. The remainder of this paper is structured as follows: In section 2 we present existing matching approaches using natural language processing techniques. Then we introduce our novel language-based machine learning approach for matching process models in section 3. We finally evaluate our concept in section 4 and give an outlook on future research.

2 Related Work

Process model matching originates from the field of schema matching and ontology matching (Euzenat and Shvaiko 2013; Bellahsene et al. 2011). The term Process Model Matching refers to an identification of corresponding nodes, which are contained in process models. In the context of process model matching, these correspondences are called matches. Matching process models is used to support many use cases, e.g, determining model similarities (Dijkman et al. 2011), merging process models (La Rosa et al. 2010) and inductively generating reference process models (Rehse et al. 2015). Algorithms which are dedicated to the generation of matchings between process models are called *matcher*. A matcher usually consists of two parts. The first part consists in determining similarities between the elements of the respective process models while the second part represents the selection of matches according to these similarities. The similarity measures used in recent approaches vary and focus on manifold criteria of a process model, e.g., the language contained in the labels (Klinkmüller et al. 2013) or the process model structure (Gao et al. 2014). An overview of recent matchers is provided by the process model matching contest (Antunes et al. 2015). In the contest and for further evaluations the performance measurement of a matcher is based on precision, recall and F-measure that are established measures in the context of information retrieval. This evaluation method requires a predefined matching, which is called gold standard. However, since matching process models is considered an unstructured decision problem (Thaler et al. 2014), derived gold standards might be ambiguous.

To our best knowledge, by now three approaches *NSCM/NHCM* (Cayoglu et al. 2013; Antunes et al. 2015), *VM2* (Antunes et al. 2015) and the approach of Klinkmüller et al. (2014) consider a matching problem as a set of process models that are matched. These matchers provide techniques, which exploit the information provided by a set of process models while the others only focus on pairs of process models. Additionally, only (Klinkmüller et al. 2014) provides a learning technique that builds up on a user's feedback. Hence, no matching approach exists that autonomously learns how to match. However, the approach introduced in this paper represents a learning approach, which requires gold standards as learning data. After a learning stage, it can be applied automatically to various data sets without requiring an interacting user.

3 Matching Approach Using Machine Learning

3.1 Related Learning Concepts

Searching for a suitable machine learning approach for matching process models that accounts for our situation that we have only little training data, we choose similarity learning instead of the widespread concepts of deep learning with *artificial neural networks* (ANNs) (Karayiannis and Venetsanopoulos 2013), *reinforcement learning* (Wiering and van Otterlo 2012) and evolutionary algorithms (Spears 2013). ANNs learn from given examples by establishing relations between input and output data. Deep learning using ANNs imitates human learning but needs huge amounts of given observations to establish synapses. Deep learning methods became quite popular in the last years for solving pattern recognition problems. But matching model labels required learning a whole language or at least a closed terminology. Our training data offers only a few example matches that are only a small segment of their semantic context. Reinforcement learning seeks to find a decision strategy by a reinforcement rule punishing bad decisions and rewarding good ones. This is not appropriate for the matching problem because there are too many possibilities to match words in the context of a model label. Learning a decision strategy is difficult in our case because the gold standard contains only matched model labels and not the single words. Evolutionary algorithms optimize with a fitness function that evaluates a solution. Their strength is that solutions can be evolved over an unknown search space. Their weakness is that they cannot profit from given examples. So, a fitness function could not usefully evaluate a new matching solution.

3.2 Similarity Learning Approach

In this section we introduce *SMML* (Semantic Matching based on Machine Learning), a n:m process model node label matching approach based on supervised similarity machine learning in order to learn the human judgement with help of a gold standard. We follow a similarity learning method: Learning a similarity function that measures how similar or related two objects are from given examples (Bellet et al. 2015). Finding a good similarity function is an approximate optimization problem, similar to a linear regression. Conceiving our learning approach we took an approved measure for word similarity (see 3.3) and reduced the learning problem to estimate only the difference or relatedness between two model labels. The difference between the gold standard matchings and the *SMML* matchings is to be minimized. The advantage of similarity is that it comes out with comparatively little training data.

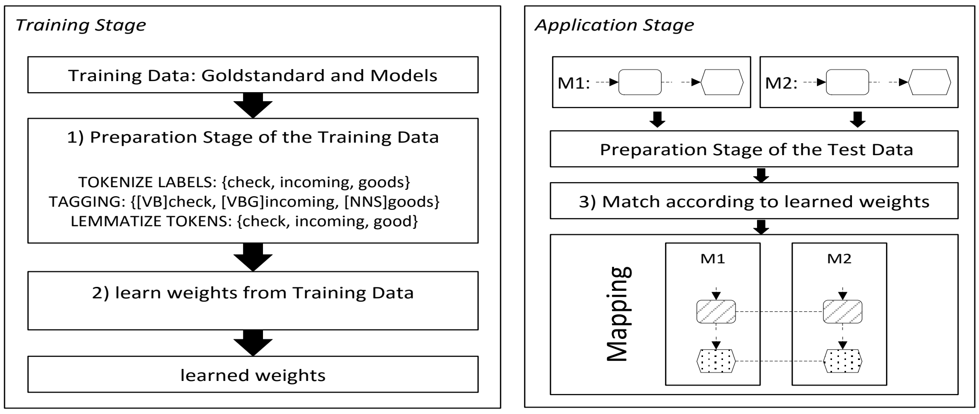
Training Stage
Training Data: Goldstandard and Models
1) Preparation Stage of the Training Data
TOKENIZE LABELS: {check, incoming, goods}
TAGGING: {[VB]check, [VBG]incoming, [NNS]goods}
LEMMATIZE TOKENS: {check, incoming, good}
2) learn weights from Training Data
learned weights
Application Stage
M1:
M2:
Preparation Stage of the Test Data
3) Match according to learned weights
Mapping
M1
M2

Figure 1. SMML stages: SMML matches two process models M1,M2

Our approach consists of three stages (see figure 1): First all node labels are lexically and syntactically prepared for the learning stage in which a training dataset with a given gold standard is used to learn the similarity function that assigns a similarity value to two node labels of a process model, each node label consisting of a vector of tokens. Therefore, a token similarity measure is defined. The learning procedure optimizes the similarity function until a satisfying matching quality is reached. Then the learned similarity function is applied on the unknown test dataset, matching all tokens in each process model node pair.

3.3 Preliminaries and Preparation Stage

Speaking of matching process models, we refer to the technique of automatically identifying correspondences between the activities of process models. For evaluating process matching techniques, in recent work three metrics have been established: precision, recall and F-measure (Weidlich et al. 2010). These metrics quantify the difference between the achieved and the expected matching result. Precision is the fraction of matched node labels that coincides a matching from the gold standard. Recall is the fraction of the correctly matched node labels. The F-measure is the harmonic mean between precision and recall.

In preparation of *SMML*, the Stanford tokenizer (Manning et al. 2014) is applied on the node labels of all process models to be matched and divides them in words respectively tokens (we refer to the tokenizer in the following as $getTag(token)$). For each token its tag is determined by the Stanford tagger (Manning et al. 2014). Tags are "verb", "noun", "adjective" and so on. Then the tokens are lemmatized so that grammatical forms are neutralized. The tokens in a sentence form a style-dependent structure formed by the respective token tags (Leopold 2013; Leopold et al. 2011). Hence the semantic of a sentence is based on that structure which makes the importance of tags in the context variable. Therefore, it is necessary to quantify/weight the influence of a certain tag for the sentence semantic. So we weight every token/tag combination differently which allows a verb to have another weight than, for example, a noun or an adjective.

In respect to the complexity of semantic word relationships, the complexity of finding for example synonyms hypernyms, homonyms, polysyms, etc. in the context of a sentence, we focus on the synonym respectively hypernym relationship between the lemma of two tokens. Finding this relationship is difficult because there can be several nearest common hypernyms candidates for two tokens. So, simultaneously to the tags, the influence of the token similarity measure has to be weighted.

The advantage of learning weights for tokens and their similarity measure is that many possible context dependent matching candidates for a node label can be considered. When sufficiently enough candidates were evaluated, the learned weights represent the knowledge of matching as the gold standard suggests.

3.4 Learning Application

Given two models M_1, M_2, the resemblance of two tokens $t_1 \in M_1, t_2 \in M_2$ of different node labels is calculated by the similarity function $sm(t_1, t_2)$. sm is composed of the weighted path length $length(t_1, t_2)$ between t_1, t_2 in WordNet by Miller and George (1995), the weighted LIN measure by Lin (1998) and the weights of the tokens tags. Principally, any lexical database/network of synonyms can be used here because the path length in a synonym network should be language-independent. We apply the LIN measure as it has been shown to reflect human judgment (Pittke et al. 2015) and weight it together with the path length. Each token receives a weight in combination with its tag. The LIN measure between two tokens is weighted by the weight of the tokens supertoken and the path length between both tokens $length(t_1, t_2)$, weighted by both tokens supertoken. The supertoken *super* (see definition 1, point 4) is the closest hypernym to both tokens in WordNet which is defined by the central word on the shortest path ($Path(t_1, t_2).getCentralElement()$) between the hypernyms of the original tokens (see definition 1, point 3).

<u>Definition 1:</u>

The form of *sm* for two tokens t_1, t_2 is:

$sm(t_1, t_2) = w_{length} * w(super(t_1, t_2)) * length(t_1, t_2) + w_{LIN} * w(super(t_1, t_2)) * LIN(t_1, t_2)$
with, let *w* be a weight and t_i a token:

1. $w(t_i) = w(t_i, getTag(t_i))$
2. $Path(t_1, t_2) =$ The shortest path between the tokens t_1, t_2 considering only edges in WordNet that represent hypernym relations. $length(t_1, t_2) = |Path(t_1, t_2)|$
3. $Path(t_1, t_2).getCentralElement() = Path(t_1, t_2).getElementAt\left(round\left(\frac{|Path(t_1,t_2)|}{2}\right)\right)$

   $Path(t_1, t_2).getElementAt(p)$ returns the p th element of a path.
4. $super(t_1, t_2) = Path(t_1, t_2).getCentralElement()$

<u>Definition 2:</u>

$pcs = \{(t_1, w(t_1)), (t_2, w(t_2)), \ldots, (t_n, w(t_n))\} \cup \{w_{LIN}, w_{length}, th\} \cup$
$\{(t_1, t_2, super(t_1, t_2)), (t_1, t_3, super(t_1, t_3)), \ldots, (t_{n-1}, t_n, super(t_{n-1}, t_n))\}$ with $w_{LIN}, w_{length}, th \in [0; 1]$, tokens $t \in R$ and $n = |R|$. R is the set of all tokens in all models.

th, w_{length} and w_{LIN} and all $w(t_i)$ and all $super(t_i, t_j)$ are the parameters to be learned/optimized. We want to find the highest F-measure for each possible parameter combination *pc* out of *pcs*, whereby *pc.sm* is the similarity function with the parameters from *pc*. Matching all node labels in the training dataset, a local search is performed on a high performance computing cluster (HPC) trying as many *pcs* as possible storing the combination reached the highest F-measure. Principally any complete search algorithm can be used to find the best *pc*. Now *sm**, *sm* with the best found parameter combination can be applied to test data.

After the best similarity function *sm** is learned, *SMML* determines for each mutual node label pair in process model *M1* and *M2* whether two process model node labels are matched or not. Two node labels are matched if $sm^*(t_1, t_2)$ for all mutual token pairs $(\mathrm{t}_1, \mathrm{t}_2)$ reaches the learned threshold *th**. The matching between *M1* and *M2* is then the set containing all tuples of matched node labels. A more detailed description of our algorithm can be viewed in figure 2.

For each label pair, in the worst case, *sm** has to be computed for each token pair in both labels. Similar WordNet queries can be executed in advance so that most weights in *sm** are constants which reduces the overall computation effort.

3.5 Implementation

Our *SMML* approach is implemented in Java within a prototypical software platform (http://refmod-miner.dfki.de/) to demonstrate the concept applicability. The software platform offers tools for business process management such as process model matching, model analysis and inductive reference modelling. Many input modelling languages like *event driven process chains* (EPCs), *Business Process Modeling and Notation* and *Petri nets* are supported. After the models are imported, they are abstractly represented in Java classes. For the tokenizer we use the implementation from (Manning et al. 2014), a lemmatizer relying on word relations maintained in the Wiktionary Database (Zesch et al. 2008) and the free available WordNet by Miller (1995). The pseudocode for the learning stage and for the application stage is presented in figure 2. The application stage is given a model corpus and puts out label mappings for each model combination. <M1, M2> is a pairwise model combination and <L1, L2> is a pairwise label combination.

```
LEARNING_STAGE Output: optimal parameter combination sm*
1: Let Fmeasures be a vector
2: PCS = set of all possible parameter combinations
3: Local_search(Test_parameter_combination) over PCS
4: sm* = get pc associated with highest F-measure from Fmeasures
5: Function Test_parameter_combination(pc)
6:    Let matches be a vector
7:    For each <M1, M2> Do
8:        For each <L1, L2>, L1 in M1 and L2 in M2 Do
9:            Lemmatize(Tokenize(L1), Tokenize(L2))
10:           For each token T1 in L1 and token T2 in L2 Do
11:               If pc.sm(T1, T2) < pc.th then
12:                   break loop
13:           matches.add(<M1, M2>, <L1, L2>)
14:    Fmeasures.add(pc, mean F-measure of matches[<M1, M2>])
```

```
APPLICATION_STAGE Output: matchings
1: Let matches be a vector
2: For each pairwise model combination <M1, M2> Do
3:    For each <L1, L2> with L1 in M1 and L2 in M2 Do
4:        For each token T1 in L1 and T2 in L2 Do
5:            If sm*(T1, T2) < th* then
6:                break loop
7:        matches.add(<M1, M2>, <L1, L2>)
```

Figure 2. Pseudo code for the learning stage (left) and the application stage (right)

4 Evaluation

4.1 Evaluation Set-up and Use Case

In order to evaluate *SMML*, we measure the reached matching quality with the in 3.3 introduced measures precision, recall and F-measure. For the evaluation set-up, *SMML* learns its weights on a training dataset consisting of all pairs of process models. The similarity function is learned for each model pair by searching those weights that reduces the F-measure difference between the gold standard and the *SMML* matching. When the similarity function for each model pair has been learned, the arithmetic mean of all weights forms the final similarity function.

As an evaluation use case, we compare our matching approach to the results of the Process Model Matching Contest 2015 (Antunes et al. 2015). The contest consists of three equally sized, contentual different datasets with manually defined gold standards. Each dataset served as a different evaluation scenario in the matching contest. Dataset 1 contains 9 models concerning university admission processes. Dataset 2 contains 9 models about birth registration. Dataset 3 has 9 models about asset management from the SAP Reference Model Collection. In order to show a learning effect, *SMML* passed four scenarios: S1) *SMML* was independently applied on each dataset with unlearned weights (all weights set to 1.0).S2) *SMML* learned from random 50% of the model combinations in dataset 1 and 50% of dataset 2's model combinations (training data) and was then applied to the third dataset (test data). S3) *SMML* learned from the whole dataset 1 and 2 and was applied and evaluated on the whole dataset 3. S4) *SMML* learned independently inside each dataset, ten percent-wise portions of the model combinations and was then applied on the remaining model combinations of the respective dataset. For avoiding overfitting, mean weights are learned over the portion. As search for the best parameter combination, we used a simple hill climber that starts from multiple random parameter combinations and from each starting point, one randomly chosen parameter is changed per iteration.

4.2 Results

The results of our three scenarios are presented in table 1-3. As described in the set-up, for each dataset the reached average (*AVG*) precision, recall, F-measure and their respective standard deviation (*SD*) are outlined. Therefore we are able to compare our results to the results of the matching contest scenarios. The highest precision is reached for dataset 3 in all scenarios. Overall scenarios, the precision in dataset 2 and 3 is significantly higher than the respective recall. Scenario S1 resulted in a mean F-measure of 0.22, 0.24 and 0.25. In the second scenario we observe an improvement respectively learning effect of 0.43/0.24=1.79 in dataset 1, 0.36/0.22=1.64 in dataset 2 and 0.29/0.25=1.16 in dataset 3. The learning effect in scenario S3 towards scenario S1 amounts to 0.54/0.24=2.25 for dataset 1, 0.38/0.22=1.73 for dataset 2 and 0.61/0.25=2.44 for dataset 3. The result of scenario S4 is shown in figure 3. For each portion of learned model combinations, the reached F-measure is plotted. For every dataset, the F-measure increases steadily without decrease.

		PRECISION		**RECALL**		**F-MEASURE**	
		AVG	SD	AVG	SD	AVG	SD
S1	*dataset 1*	0.46	0.32	0.16	0.37	0.24	0.17
	dataset 2	0.36	0.29	0.15	0.20	0.22	0.25
	dataset 3	1.00	0.00	0.14	0.35	0.25	0.35
S2	*dataset 1*	0.37	0.22	0.50	0.31	0.43	0.23
	dataset 2	0.46	0.14	0.29	0.22	0.36	0.16
	dataset 3	0.70	0.29	0.19	0.34	0.29	0.30
S3	*dataset 1*	0.51	0.24	0.58	0.34	0.54	0.25
	dataset 2	0.51	0.15	0.31	0.23	0.38	0.18
	dataset 3	0.80	0.30	0.49	0.42	0.61	0.37

Table 1. Results for scenario S1-S3

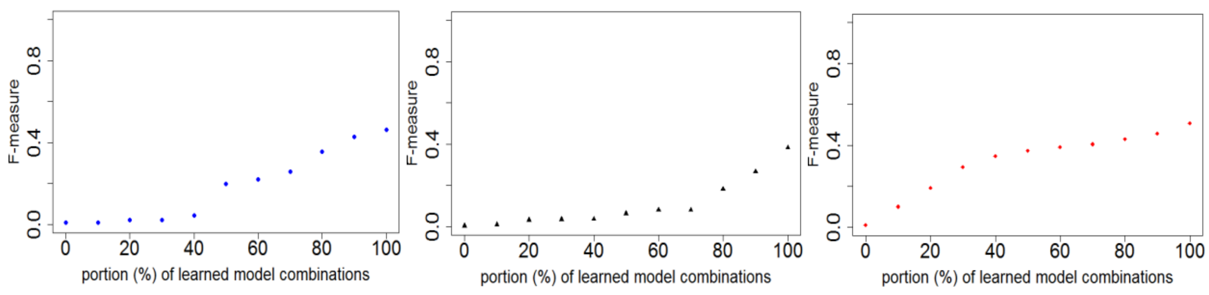

Figure 3. Results for scenario S4 (dataset1, datset2, dataset3)

4.3 Discussion

Comparing to the other contest approaches in (Antunes et al. 2015), *SMML* achieves an over-average precision and an average recall and F-measure for dataset 3. For dataset 1, *SMML* reaches a slightly over-average F-measure (0.54) and for dataset 2 *SMML* reaches an under-average F-measure (0.38).

For the second dataset, the other contest approaches resulted in worse F-measures than for the other datasets which might be caused by a particularly difficult language.

Reaching the highest precision and recall, the third scenario outperforms the other scenarios. The learning effect in terms of the F-measure increase is maximal in the third scenario. That implies that the more training data has learned from, the higher the F-measure becomes because in scenario S2 was only learned from two half datasets. We see that also as an indicator for the found weights in the first two datasets to be representative for a variety of process models because the three datasets are from different domains. Especially the third dataset comprises varied models. This speaks for the applicability of *SMML* in diverse domains. At this point, more training data is needed for improving our approach in respect to applicability and quality. Scenario S4 demonstrates that the matching quality steadily increases when more training data is provided. For datset1 and 2, the matching quality jumps up from 50% / 80% learned data so that we expect an over-linear trend for the learning effect over more training data. Scenario S3 and S4 show that the more training data from different domains was learned, the better *SMML* matches new/unknown data such as the SAP models in dataset 3.

The limitation of our concept is the quality of this machine learning concept to highly depend on the quality of the training dataset and its gold standard. The tiny learning effect in dataset 2 shows that when only little or uniform training data is at hand, the learning effect is low or not significant. This can be also the case if *SMML* is applied on a completely unknown terminology, language or syntax. It is difficult to evaluate how *SMML* or a similar approach would perform if a few models featured a disparate syntax. Here it becomes necessary to explore the strengths and weaknesses of a more complex similarity function with more weights. Here also the question rises if the arithmetic mean between the learned weights is appropriate for generalizing the similarity function. When more data is available, the similarity function can become more complex. With only 3 datasets, the weights are estimated in a simple linear similarity function in order to avoid overfitting in new datasets. Overfitted weights would reach a high matching quality inside a dataset but would fail in new/unknown data. *SMML* currently does not consider the context of node labels in the process such as who is the actor in an process event. The consideration of the context of model labels is an essential next step in the development of *SMML* to improve the matching quality.

5 Conclusion

A central aspect in business process management is matching business process models and their node labels. There are effective matching techniques but recent research overlooks the great potential of machine learning. In this paper we present *SMML*, a semantic matching approach based on supervised similarity machine learning. *SMML* first learns from a training dataset and then matches node labels of an unknown dataset. We evaluated our *SMML* approach with three different datasets from various domains offered by the process model matching contest 2015 (Antunes et al. 2015). Our first implementation of *SMML* demonstrates hereby its potential to understand a human matcher by a gold standard and apply its learned knowledge to new and diverse process models. An evaluation shows that the more training data is provided, the better the matching quality becomes.

Because our approach does not need predefined rules, we see great potential in similarity learning for matching business process models. With a more complex similarity function, we expect *SMML* to become a good matcher. Therefore, in future work we will develop a more elaborate similarity function, addressing additional linguistic characteristics of process models such as word frequency, choice of words and the underlying syntax (Leopold 2013). Additionally, the underlying graph structure and the topological context of a node label will be considered in the way that preceding and succeeding node labels will be used for determining similarity of a label. Hereby, the relation between referenced actors and resources of process events shall be understood over the whole process model. Therefore, the similarity learning concept will be extended by pattern recognition with neural networks respectively Boltzmann machines (Ackley et al. 1985). Finally, we will enhance the search for optimal learning parameters and evaluate their efficiency.

References

Ackley, DH., Geoffrey EH., and Terrence JS. (1985) A learning algorithm for boltzmann machines. Cognitive science 9.1 147-169.

Antunes, G., Bakhshandeh, M., Borbinha, J., Cardoso, J., Dadashnia, S., Di Francescomarino, C., Dragoni, M., Fettke, P., Gal, A., Ghidini, C., Hake, P., Khiat, A., Klinkmüller, C., Kuss, E., Leopold, H., Loos, P., Meilicke, C., Niesen, T., Pesquita, C., Péus, T., Schoknecht, A., Sheetrit, E., Sonntag, A., Stuckenschmidt, H., Thaler, T., Weber, I., Weidlich, M. (2015) Process Model Matching Contest. In: Koln, J., Leopold, H., Mendling, J. (eds). Proceedings of the 6th International Workshop on Enterprise Modelling and Information Systems Architectures. International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA-15), September 3-4, Innsbruck, Austria, Köllen Druck+Verlag GmbH, Bonn, 9/2015.

Bellahsene, Z., Bonifati, A., Duchateau, F., Velegrakis, Y. (2011) On Evaluating Schema Matching and Mapping. In: Bellahsene, Z., Bonifati, A., Rahm, E. (eds) Schema Matching and Mapping, Springer, Berlin, Heidelberg. pp. 253-291.

Bellet, A., Habrard, A., Sebban, M. (2015), Metric Learning, Morgan & Claypool Publishers.

Cambria, E., and White, B.: Jumping NLP curves: a review of natural language processing research. Computational Intelligence Magazine, IEEE 9.2 (2014): 48-57.

Cayoglu, U., Dijkman, R., Dumas, M., Fettke, P., Garcia-Banuelos, L., Hake, P., Klinkmüller, C., Leopold, H., Ludwig, A., Loos, P., Mendling, J., Oberweis, A., Schoknecht, A., Sheetrit, E., Thaler, T., Ullrich, M., Weber, I., Weidlich, M. (2013) The process model matching contest 4th International Workshop on Process Model Collections: Management and Reuse, PMC-MR.

Dijkman, R., Dumas, M., van Dongen, B., Käärik, R., Mendling, J. (2011) Similarity of business process models: Metrics and evaluation. Inf Syst 36:498–516.

Dijkman R, Dumas M, García-Bañuelos L. (2009) Graph Matching Algorithms for Business Process Model Similarity Search. In: Dayal U, Eder J, Koehler J, Reijers H (eds) Business Process Management SE - 5. Springer Berlin Heidelberg, pp 48–63

Euzenat, J. and Shvaiko, P. (2013): Ontology Matching. Springer-Verlag New York, Inc.

Gao, X., Chen, Y., Ding, Z. (2014) Process model fragmentization, clustering and merging: An empirical study. Springer International Publishing, Cham

Klinkmüller, C., Leopold, H., Weber I., Ludwig, A. (2014) Listen to Me: Improving Process Model Matching through User Feedback. In: Sadiq S, Soffer P, Völzer H (eds) Business Process Management SE - 6. Springer International Publishing, pp 84–100

Klinkmüller, C., Weber, I., Mendling, J., Leopold, H., Ludwig, A. (2013) Increasing Recall of Process Model Matching by Improved Activity Label Matching. In: Daniel F, Wang J, Weber B (eds) Business Process Management SE - 17. Springer Berlin Heidelberg, pp 211–218

La Rosa, M., Dumas, M., Uba, R., Dijkman, R. (2010) Merging Business Process Models. In: Meersman R, Dillon T, Herrero P (eds) On the Move to Meaningful Internet Systems: OTM 2010 SE - 10. Springer Berlin Heidelberg, pp 96–113

Karayiannis, N. and Venetsanopoulos (2013) AN. Artificial neural networks: learning algorithms, performance evaluation, and applications. Vol. 209. Springer Science & Business Media.

Leopold, H. (2013) Natural Language in Business Process Models: Theoretical Foundations, Techniques, and Applications. In: Lecture Notes in Business Information Processing, Vol. 168, Springer-Verlag, 2013.

Leopold, H., Smirnov, S., Mendling, J. (2011) Recognising Activity Labeling Styles in Business Process Models. Enterprise Modelling and Information Systems Architectures 6(1): 16-29.

Levenshtein, I. (1966) Binary code capable of correcting deletions, insertions and rever-sals. Cybernetics and Control Theory 10(8), 707–710.

Lin, D. (1998) An information-theoretic definition of similarity” in MLDM, ser. ICML ’98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 296–304.

Manning, C. D.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S. J. & McClosky, D. (2014), The Stanford CoreNLP natural language processing toolkit, in 'Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations', pp. 55-60.

Miller, G. A. (1995), WordNet: A Lexical Database for English, Commun. ACM 38(11), 39--41.

Mohri, M., Rostamizadeh, A. and Talwalkar, A. (2012) Foundations of machine learning. MIT press.

Pittke, F., Leopold, H., Mendling, J. (2015) Automatic Detection and Resolution of Lexical Ambiguity in Process Models IEEE Transactions on Software Engineering 41(6): 526-544.

Rehse J-R, Fettke P, Loos P. (2015) A graph-theoretic method for the inductive development of reference process models. Softw Syst Model 1–41.

Sebastiani, Fabrizio (2002) Machine learning in automated text categorization. ACM computing surveys (CSUR) 34.1, 1-47.

Spears, WM. (2013) Evolutionary algorithms: the role of mutation and recombination. Springer Science & Business Media.

Weidlich, M., Dijkman, R., Mendling, J. (2010) The ICoP Framework: Identification of Correspondences between Process Models. In: Pernici B (ed) Advanced Information Systems Engineering SE - 37. Springer Berlin Heidelberg, pp 483–498.

Wiering, M., and van Otterlo, M. (2012) Reinforcement Learning: State-of-the-art. Vol. 12. Springer Science & Business Media.

Zesch, T., Müller, C., Gurevych, I. (2008) Extracting Lexical Semantic Knowledge from Wikipedia and Wiktionary. In: (eds) Calzolari, N. (Conference Chair), Khalid, C., Maegaard, B., JMariani, J., Odijk, J., Piperidis, S., Tapias, D. Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08). European Language Resources Association (ELRA), Marrakech, Morocco