Supporting Event Log Extraction Based on Matching

Vinicius Stein Dani[1](✉), Henrik Leopold[2], Jan Martijn E. M. van der Werf[1], and Hajo A. Reijers[1]

[1] Utrecht University, Princetonplein 5, 3584 CC Utrecht, The Netherlands
{v.steindani,j.m.e.m.vanderwerf,h.a.reijers}@uu.nl

[2] Kühne Logistics University, Großer Grasbrook 17, 20457 Hamburg, Germany
henrik.leopold@the-klu.org

Abstract. Process mining allows organizations to obtain relevant insights into the execution of their processes. However, the starting point of any process mining analysis is an event log, which is typically not readily available in practice. The extraction of event logs from the relevant databases is a manual and highly time-consuming task, and often a hurdle for the application of process mining altogether. Available support for event log extraction comes with different assumptions and requirements and only provides limited automated support. In this paper, we therefore take a novel angle at supporting event log extraction. The core idea of our paper is to use an existing process model as a starting point and automatically identify to which database tables the activities of the considered process model relate to. Based on the resulting mapping, an event log can then be extracted in an automated fashion. We use this paper to define a first approach that is able to identify such a mapping between a process model and a database. We evaluate our approach using three real-world databases and five process models from the purchase-to-pay domain. The results of our evaluation show that our approach has the potential to successfully support event log extraction based on matching.

Keywords: Event log extraction · Natural language processing · Automated matching

1 Introduction

Process mining is used in many different organizations for tasks such as analyzing, improving, and auditing business processes [5,9,18]. However, the application of process mining requires an event log [1], which is often not readily available in practice [4]. One of the main reasons is that the information systems supporting the execution of many business processes do not produce event logs that can be used for process mining. As a result, event logs need to be extracted manually by exploring the underlying databases of these information systems. In essence, every activity executed in the context of the business process must be manually related to specific tables in the database. This mapping is then used to extract the event log. This effort for event log extraction is very time-consuming and requires considerable manual work [20]. It, thus, creates a substantial hurdle for the application of process mining in practice [22].

Recognizing this, many researchers have developed techniques to support the extraction of event logs. However, they usually require creating an intermediate data model [16] or using instance data [13]. Furthermore, they do not automatically identify the mapping between the tables of a database and the activities of a considered process because they do not focus on extracting event logs that relate to an already known process flow.

In this paper, we propose a novel approach for supporting event log extraction that takes an existing process model as a starting point. The core idea is to automatically identify to which database tables the activities of a given process model relate to and, based on the resulting mapping, provide an effective alternative for event log extraction. In prior work, the problem of mapping entities from two different representations has been addressed in various contexts. Among others, researchers have proposed techniques for finding mappings between database schemas [14,17], between ontologies [10,11], or between process models [21,23]. Such techniques for automatically deriving mappings between two different representations are commonly referred to as *matchers* [23]. However, to the best of our knowledge, there is no approach available that focuses on identifying a mapping between a database and a process model [20]. To accomplish this, we build on a two layer matching architecture and different notions of similarity.

The remainder of the paper is structured as follows. In Sect. 2, we illustrate the problem of and the challenges related to creating a mapping between database tables and process model activities. In Sect. 3, we describe our proposed approach to support event log extraction based on matching. Section 4 evaluates an implemented proof-of-concept. Finally, in Sect. 5, we discuss related work and in Sect. 6, we conclude this paper.

2 Problem Illustration and Challenges

In this paper, we approach the problem of event log extraction from a matching perspective. More specifically, we aim to develop an approach that automatically identifies a mapping between the tables of a database and the activities of a given process model. To illustrate the problem and the associated challenges, consider the example shown in Fig. 1. It shows a simplified purchase-to-pay process model (extracted from [5]) and a corresponding exemplary database. The goal of our approach is to identify for each activity from a given model to which database table it relates (if any). Formally, such a mapping is a relation over the activities and tables, such that (a, t) maps activity a to table t. In other words, table t contains data of an event for activity a. A *potential mapping* is a candidate mapping that needs to be verified for correctness. Figure 1 depicts several potential mappings. The relations with a checkmark are correct mappings, whereas the mappings marked with a cross are incorrect. Automatically identifying the correct mappings comes with four main challenges:

1. *Large search space*: Given that databases often contain hundreds of tables, the search space for the mapping is typically very big. To illustrate this, consider the example from Fig. 1. The combination of 6 activities and 26 tables already results in over 300

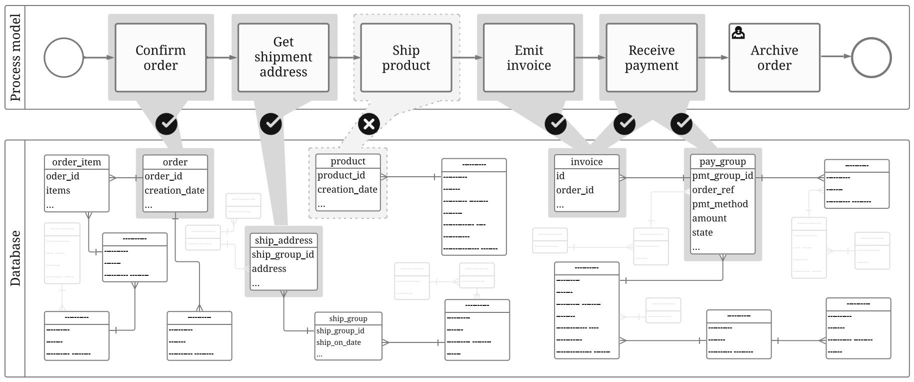

Fig. 1. A process model, a database, and the mappings between them.

million possible mappings. A useful matching technique, therefore, must be able to effectively reduce the search space and precisely recognize which activity-table pairs represent correct mappings.

2. *Granularity differences*: Processes and databases dramatically differ in their level of granularity. While a process model typically only has a handful of activities [6], a database often has hundreds of tables. This causes two related problems. First, this means that a single activity may have multiple corresponding tables. For example, in Fig. 1, the activity "*Receive payment*" produces a payment entry for the database table "*pay_group*" while also producing an update of an entry in the table "*invoice*". Second, this means that a single table may have multiple corresponding activities. For example, the table "*invoice*" stores data about a newly created invoice produced by the activity "*Emit invoice*". The same table also reflects a payment status updated via the execution of the activity "*Receive payment*".
3. *Scope differences*: The scope of the process model and the database rarely overlap to a full extent. As a result, the mapping between process model and database is partial. This means that some activities do not have a correspondence to any table and, the other way around, many tables do not have a correspondence to any activity. For example, in Fig. 1, the activity "*Archive order*" may be related to a manual status update executed on an external system managed by another department of the organization and, therefore, does not relate to any of the tables of the considered database.
4. *Ambiguous semantics*: Both process models and database tables typically have very short labels. As a result, it is often hard to identify which words from the considered labels carry the important semantics. To illustrate this, consider the activity "*Ship product*" from Fig. 1. We can see that this activity contains the action "*ship*" and the object "*product*". In Fig. 1 it is, however, incorrectly mapped to the table "*product*" instead of "*ship_group*". The problem is that it is hard to evaluate which term should be used in this context to decide about the mapping since both "*ship*" and "*product*" are used in the database tables.

In this work, we make a first attempt to address these challenges. We propose an approach that identifies a set of potential mappings between process model activities and database tables. While this does not provide the user with a final set of correct mappings, the user is provided with a small set of potential mappings. From those, the user can simply select the correct mappings and, hence, no longer needs to look at all possible mappings and identify each mapping manually. We realize that this only represents a first step. We are, nonetheless, convinced that this already dramatically reduces the burden of the process analyst and saves a considerable amount of manual work. In the next section, we introduce our approach on a conceptual level.

3 Mapping Database Tables to Process Activities

In this section, we describe our matching approach to automatically map database tables to process model activities. We first present an overview of the architecture of our matching approach in Sect. 3.1. Then, in Sect. 3.2 and Sect. 3.3, we discuss the main components of our matcher in detail.

3.1 Overview

Figure 2 shows the architecture of our proposed approach. The first module is responsible for *preprocessing* and feeding input data into the matcher. Among others, the preprocessing component parses the input, removes irrelevant tokens (such as punctuation), and turns all strings into lower case. The input data includes a database and a process model. At this point, we expect that both have already been transformed into a textual format and are provided as CSV files. These files contain the table attributes from the database (e.g., tables names, descriptions, and columns with their names and descriptions), and the activity labels from the process model.

Inspired by [7], the *matcher* module consists of two main components: a first- and a second-line matcher (1LM and 2LM), where the 2LM builds on the output of the 1LM. The matcher automatically generates a set of potential mappings. To generate these potential mappings, we leverage natural language processing (NLP) techniques and the available input information. The main intuition behind relying on NLP techniques is that tables and activities with similar names are more likely to be conceptually similar and, therefore, related. In the following sections, we explain the details of the components from the matcher module.

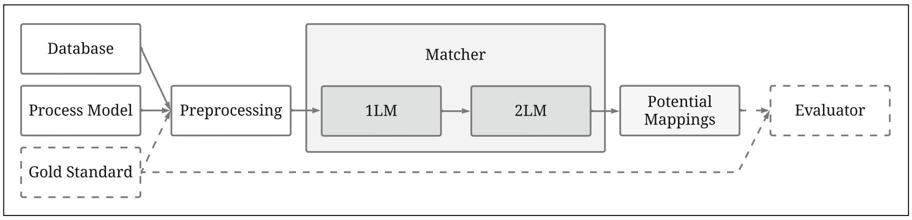

Fig. 2. Architectural overview of our approach.

3.2 First-Line Matcher (1LM)

Our approach starts with analyzing the set of activity labels A of the process model and tables T of the database using different similarity metrics. For each table, we consider all database table attributes, denoted by R. Then, for each activity a and database table attribute t_r, several similarity measures are calculated. This results in a set of similarity matrices M_s, for each similarity measure s.

Table 1 shows a cohort of the similarity matrix $M_{s(A \times R)}$ for the normalized Levenshtein-based similarity measure on a process model with two activities, "*Create order*" and "*Create invoice*", and a database consisting of two tables, "*Order*" and "*Invoice*". In this example, the table "*Order*" has two columns: "*id*", and "*creation_date*" and, the table "*Invoice*" has three columns: '*id*", "*id_order*", and "*date*".

3.3 Second-Line Matcher (2LM)

The 2LM derives the set of potential mappings between tables and activities by using as input the similarity matrix M_s generated by the 1LM. Our approach maps exactly one database table t to one activity label a, and the inner workings of the 2LM adheres to the following rationale: First, considering all available similarity scores in M_s (cf., Table 1), the 2LM determines a similarity score to represent a table with respect to each activity. This is performed for each tuple (a, t). Second, for each activity, it selects one table as a potential mapping considering the similarity score assigned to the table. Many different mechanisms can be implemented to derive the table's similarity score from its attributes' similarity scores.

We developed a baseline 2LM inspired by [21], which selects the *Highest raw 1LM-based Scoring Table* as a potential mapping for an activity label. Based on the output of this 2LM for each 1LM similarity matrix, we performed an inductive content analysis with open coding [19]. Recurrent observations from the coding served as a basis for the definition of two new 2LM implementations: one based on *Word Frequency (2LM$_2$)*, and another based on the *Surface Measure of Overall Table Scores (2LM$_3$)*. Next, we further explain each implemented 2LM.

Table 1. Similarity matrix generated by the 1LM for the normalized Levenshtein-based similarity algorithm. The closer the similarity score is to 1, the higher the similarity between the two compared objects.

Database tables attributes t_k	Process model activities a	
	Create order $f_s(a, t_k)$	Create invoice $f_s(a, t_k)$
Order	0.590	0.210
id	0.140	0.120
creation_date	0.480	0.440
Invoice	0.210	0.670
id	0.140	0.120
id_order	0.500	0.180
date	0.380	0.330

Highest Raw 1LM-Based Scoring Table ($\mathbf{2LM_1}$). Each row in a similarity matrix M_s produced by the 1LM represents the similarity scores of an activity and all attributes t_r of all tables $t \in T$. 2LM_1 selects for each activity and table combination the attribute with the highest similarity as *table score*. Then, for each activity, the table with the highest table score is selected as potential mapping.

Word Frequency ($\mathbf{2LM_2}$). This technique multiplies the table attributes similarity score by the number of activity label word repetition within the table attribute. This is done before the *table score* definition and, if there is no word repetition, the similarity score is kept as is. Hence, this matcher derives each of its potential mappings similarly to 2LM_1.

Surface Measure of Overall Table Scores ($\mathbf{2LM_3}$). This technique is inspired by [8], and leverages all similarity scores of a table to build a radar chart, where each similarity score is an axis of the chart. The *table score* $S(a, t)$ is then determined by calculating its surface area, as shown in Eq. 1, where R denotes the set of table attributes.

$$S(a, t) = \sin\left(\frac{\pi}{|R|}\right) \sum_{x \in R} \sum_{y \in R} (M_s(a, t_x) \cdot M_s(a, t_y)) \tag{1}$$

4 Evaluation

In this section, we present a quantitative evaluation of our approach. In Sect. 4.1 and Sect. 4.2, we describe the data and our setup. In Sect. 4.3, we report on the results and provide a discussion in Sect. 4.4.

4.1 Data

The evaluation builds on three inputs: 1) a set of databases, 2) a set of process models, and 3) a gold standard.

Databases. For the evaluation, we used three databases: 1) Odoo (former Open ERP), 2) Magento Commerce, and 3) Oracle ATG Webcommerce. The selected databases cover two scenarios we want to evaluate: databases with and without textual descriptions of the tables and columns. Oracle is accompanied by a textual description, whereas Odoo and Magento are not. Additionally, these databases were selected considering two other factors: 1) they store purchase-to-pay data; and, 2) they are widely used. Table 2 summarizes the overall characteristics of the selected databases.

Process Models. We used five process models of a purchase-to-pay process of different sizes. The set of process models contains one small process model extracted from [5], and four medium-sized process models extracted from the BPM Academic Initiative (BPMAI) repository [24]. The BPMAI models were selected based on the following criteria: 1) it is modelled in English, 2) it contains at least 10 activities, and 3) it relates to a purchase-to-pay process. To make sure the latter is the case, we selected process models containing the business objects "*order*", "*invoice*", and "*shipment*". Table 3 summarizes the overall characteristics of the selected process models.

Table 2. Characteristics of the databases used in the evaluation of our approach.

Characteristic	Odoo	Magento	Oracle
Database			
Nº of tables	571	358	239
Nº of columns	6294	3561	1199
Nº of words	57297	42189	37051
Table			
Avg Nº of words per table name	2.794	3.502	2.838
Avg Nº of words per table description	2.356	3.815	14.197
Avg Nº of words per column name	1.978	2.216	1.952
Avg Nº of words per column description	1.974	2.311	12.009

Table 3. Characteristics of the process models used in the evaluation of our approach.

Characteristic	PM_1	PM_2	PM_3	PM_4	PM_5
Process model					
Nº of activities	6	10	11	12	14
Nº of words	13	34	33	34	50
Activity label					
Min Nº of words	2	1	2	1	2
Max Nº of words	3	6	5	5	6
Avg Nº of words	2.166	3.400	3.000	2.833	3.571

Gold Standard. The gold standard G contains the true mappings between the database tables t and the process model activities a. It is a set of relations (a, t). To evaluate the quality of the output of our approach (i.e., the potential mappings), we compare it to G as we further explain in the next section. We manually compiled G based on prior experience and insights into which tables hold the information related to the considered activities. For activities we did not know the corresponding table, we consulted the documentation of the database. We fine-tuned G based on discussions until consensus.

4.2 Setup

For each combination of database and process model, we generated ten similarity matrices $M_{s(A \times R)}$ via 1LM, one for each similarity algorithm $s \in S$, comprising different string-similarity scoring techniques, such as: edit-based (via Levenshtein, and a normalized Levenshtein-based algorithm), Jaccard, n-gram, and Cosine similarity. Then, we implemented the 2LMs as discussed in Sect. 3.3, and to assess the performance of our approach we use precision, recall, and F1-score. This is in line with evaluations from other matching papers from the BPM domain (see e.g. [21]). To calculate these metrics, we compare the output from our approach with the mappings from the gold standard G.

Given a combination of a process model containing the activities A and a database containing the tables T, we compare the set of mappings between A and T from the gold standard G with the set of potential mappings P automatically produced by our approach. Based on this comparison, we can identify: 1) the correct mappings (i.e., the true positives *TP*) via $G \cap P$, 2) the incorrect mappings (i.e., the false positives *FP*) via $P \setminus TP$, and, 3) the missing mappings (i.e., the false negatives *FN*) via $G \setminus TP$. Thus, we can calculate precision via $\frac{TP}{TP+FP}$ and recall via $\frac{TP}{TP+FN}$. The F1-score is the harmonic mean between precision and recall.

4.3 Results

Table 4 summarizes the performance results of our approach in terms of precision, recall, and F1-score. For each database, the fourth column of this table presents the number of mappings in the gold standard G. This allows us to compare the number of mappings from G to the amount of correct mappings (TP) generated by each of the 2LMs.

On average the implemented 2LM$_3$ finds 39$\%$ of the correct mappings for the databases with table and column descriptions. The implementations 2LM$_2$ and 2LM$_3$, perform similarly for a scenario where the database does not have useful textual descriptions, as shown in Figs. 3d and 3g, for example. The 2LM$_3$ implementation performs better than 2LM$_2$ for a scenario where the database has textual descriptions, as shown in Figs. 3f and 3i. In both scenarios, the 2LM$_3$ performs well when the process model does not have too many similar activity labels, which is the case for the results related to PM$_1$. All the results presented in this work are based on a 1LM using cosine similarity, which is the similarity algorithm that performed best. Figures 3a to 3i depict,

Table 4. Evaluation summary with Precision, Recall, F1-scores, total true positives (*TP*), and false positives (*FP*) for the three different 2LM implementations. The baseline 2LM$_1$ results are zero for Odoo and Magento because more than one table had the same highest similarity score for each activity and the baseline selects the first table with the highest similarity score as a potential mapping.

DB	PM	$\|A\|$	$\|G\|$	2LM$_1$					2LM$_2$					2LM$_3$				
				P	R	F1	TP	FP	P	R	F1	TP	FP	P	R	F1	TP	FP
Odoo	1	6	7	0.000	0.000	0.000	0	6	0.000	0.000	0.000	0	6	0.167	0.143	0.154	1	5
	2	10	9	0.000	0.000	0.000	0	10	0.100	0.111	0.105	1	9	0.100	0.111	0.105	1	9
	3	11	12	0.000	0.000	0.000	0	11	0.091	0.083	0.087	1	10	0.091	0.083	0.087	1	10
	4	12	6	0.000	0.000	0.000	0	12	0.000	0.000	0.000	0	12	0.000	0.000	0.000	0	12
	5	14	8	0.000	0.000	0.000	0	14	0.000	0.000	0.000	0	14	0.000	0.000	0.000	0	14
Magento	1	6	6	0.000	0.000	0.000	0	6	0.167	0.167	0.167	1	5	0.167	0.167	0.167	1	5
	2	10	5	0.000	0.000	0.000	0	10	0.100	0.167	0.125	1	9	0.000	0.000	0.000	0	10
	3	11	6	0.000	0.000	0.000	0	11	0.182	0.333	0.235	2	9	0.000	0.000	0.000	0	11
	4	12	4	0.000	0.000	0.000	0	12	0.000	0.000	0.000	0	12	0.000	0.000	0.000	0	12
	5	14	7	0.000	0.000	0.000	0	14	0.071	0.143	0.095	1	13	0.071	0.143	0.095	1	13
Oracle	1	6	8	0.000	0.000	0.000	0	6	0.167	0.125	0.143	1	5	0.834	0.625	0.714	5	1
	2	10	8	0.100	0.125	0.112	1	9	0.100	0.125	0.112	1	9	0.300	0.375	0.334	3	7
	3	11	11	0.091	0.091	0.091	1	10	0.273	0.273	0.273	3	8	0.454	0.454	0.454	5	6
	4	12	6	0.167	0.334	0.223	2	10	0.167	0.334	0.223	2	10	0.250	0.500	0.334	3	9
	5	14	8	0.071	0.125	0.091	1	13	0.143	0.250	0.182	2	12	0.357	0.625	0.454	5	9

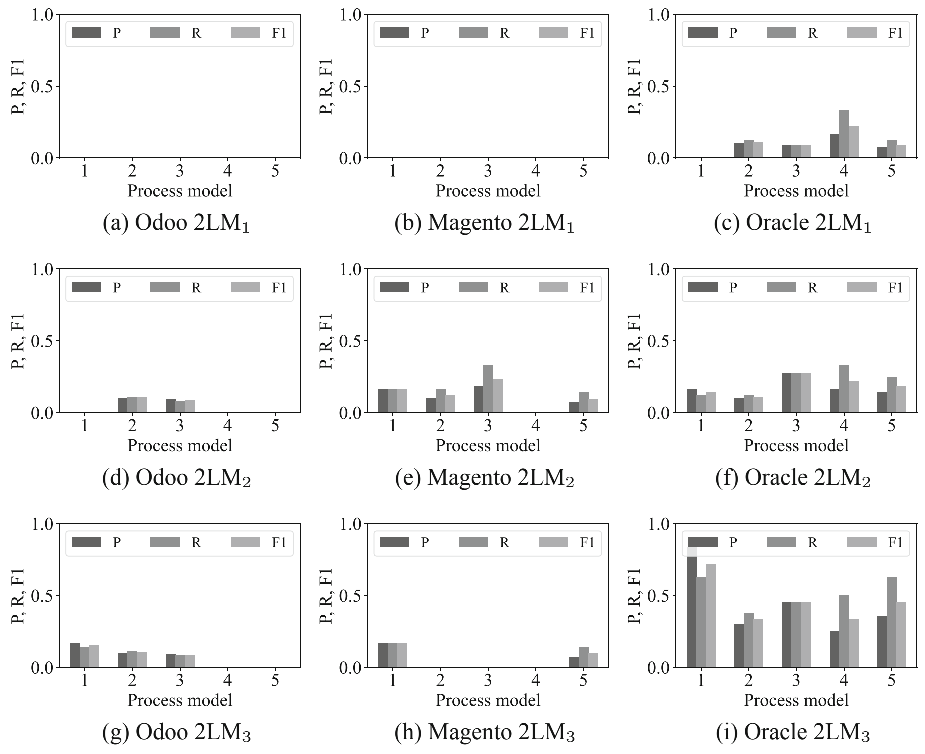

Fig. 3. Evaluation output for three databases and five process models used in this evaluation. The first row of figures shows the output for the baseline 2LM$_1$, while the second and the third rows show the output for the other two implemented 2LMs. Figures 3a, 3d, and 3g refer to Odoo; Figs. 3b, 3e, and 3h refer to Magento; and, Figs. 3c, 3f, and 3i refer Oracle. On each figure, the vertical axis represents the value of precision, recall, and F1-score, for each of the five process models, shown in the horizontal axis.

respectively, the output of our approach for the three different implemented 2LMs presented in Sect. 4.2. The first column of Fig. 3 presents the output for Odoo, the second for Magento, and the third for Oracle.

In summary, we can state that all 2LM implementations performed better on the scenario where textual descriptions were available for the tables and columns. Moreover, the 2LM$_2$ and the 2LM$_3$ improve consistently when compared to the baseline implementation on the Oracle database, which is the database with textual descriptions for both tables and columns. For the databases without textual descriptions, the results deteriorate in general, showing the importance of additional textual information about the objects being mapped. The reason for this results deterioration is that multiple tables end up receiving the same similarity score, driven by similarly named columns throughout different tables.

4.4 Discussion

To generate the $2LM_2$ and the $2LM_3$, we derived improvement opportunities based on recurrent observations acquired via an inductive content analysis with open coding [19] performed over all outputs from the $2LM_1$. By doing so, we avoided optimizing a new 2LM to any particular scenario.

The performed content analysis supported the identification of commonalities and differences among all potential mappings (correctly and incorrectly identified mappings) versus the ones that should have been identified, but were not. We made the following key observations: First, the missing mappings (i.e., FN) often had repetition of words that were similar to the ones within the activity label, while it was not the case for the incorrect mappings. Second, the incorrect mappings often had multiple table elements with mild similarity scores, while the wrongly selected potential mapping had usually only one slightly higher score, which then misled the baseline mapping derivation. Therefore, the matcher should consider the table attributes scores altogether.

With the current work, we provide a first step towards supporting event log extraction based on a given process model. Our approach is able to identify a set of potential mappings, which then can be processed by a process analyst. While our technique can be further improved, we also provide some insights into how this can be accomplished (cf. $2LM_3$).

5 Related Work

While this paper is the first work on database to process model matching, there are three major research areas that are concerned with matching: schema matching, ontology matching, and process model matching.

Approaches for *schema matching* aim to identify matches between the elements of two different database schemata. The purpose of schema matching techniques include data integration, schema evolution, and maintenance [14,17]. The matching strategies pursued by these techniques are similar to the ones presented in this paper. For example, in [14], the authors determine the similarity between two database schema elements using attributes, such as names and data types, and combine it with structural similarity. In [17], the authors leverage the results of a variety of basic matchers to determine whether two schema elements match.

Approaches for *ontology matching* are concerned with matching the elements of two ontologies [10,11]. One of the key use cases for ontology matching is ontology merging, i.e., the combination of two ontologies. The matching strategies are again similar to one presented here. For example, in [10], the authors leverage lexical and structural characteristics of the considered ontologies to determine matching elements.

Approaches for *process model matching* aim to identify correspondences between the activities of two process models [12,15,23]. The main use case of process model matching is to detect differences and commonalities between two processes. Available approaches for process model matching exploit textual, structural, and behavioral features of the models. Early work mainly built on simple textual similarity features, such as the Levenshtein distance, and mainly focused on structural features [23]. Later, also semantic similarity measures and behavior were used to identify corresponding activities [12].

This brief review illustrates that existing matching approaches are closely related to our work. There is, however, a key difference: The works above focus on matching entities of the same type. While this does not guarantee that the to-be-matched entities are similar, they are at least comparable. In the setting addressed in this paper, we need to deal with the fact that the entities are very different in nature. A process model, for example, does not come with instance data and a database does not have a clear notion of control-flow or activities. Hence, while we partially build on matching strategies explored in previous work, the conceptual setting of our work differs considerably.

6 Conclusion

In this paper, we presented a new approach to support event log extraction based on matching. The main idea of our approach is to automatically identify the mappings between a database and a process model. Against the background of the challenges associated with this task, we focused on the automated identification of potential mappings in this paper. While this requires process analysts to select the correct mappings, it still saves them from a considerable amount of manual work. To evaluate our approach, we tested it using three different databases and five different process models related to a purchase-to-pay process. We found that textual information is highly important to improve the performance of our approach. At the same time, we also found that more sophisticated mechanisms are required to further improve our approach.

As for future work, we see several directions. First, we plan extend the idea from the syntactic level to a level that incorporates semantic relations as well [2,3] between the activities and tables by, for example, leveraging bidirectional encoder representations from transformers. Second, we aim to take order relations between the database instance data and the process model activities into account. In this way, we can, for instance, exclude candidate matches if the order relations from the process model contradict the timestamps from the associated database tables. Third, we intend to incorporate feedback from humans. By letting the user select which potential mappings are correct, we can leverage a feedback loop to further improve the potential mappings generated by our approach.

Acknowledgements. Part of this research was funded by NWO (Netherlands Organisation for Scientific Research) project number 16672.

References

1. van der Aalst, W.M.P.: Process Mining: Data Science in Action. Springer, Heidelberg (2016)
2. Calvanese, D., Kalayci, T.E., Montali, M., Santoso, A.: OBDA for log extraction in process mining. In: Ianni, G., et al. (eds.) Reasoning Web 2017. LNCS, vol. 10370, pp. 292–345. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61033-7_9
3. Calvanese, D., Kalayci, T.E., Montali, M., Tinella, S.: Ontology-based data access for extracting event logs from legacy data: the onprom tool and methodology. In: Abramowicz, W. (ed.) BIS 2017. LNBIP, vol. 288, pp. 220–236. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59336-4_16
4. Diba, K., Batoulis, K., Weidlich, M., Weske, M.: Extraction, correlation, and abstraction of event data for process mining. WIREs Data Min. Knowl. Discov. **10**(3) (2020)

5. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer, Heidelberg (2018)
6. Figl, K., Mendling, J., Strembeck, M.: The influence of notational deficiencies on process model comprehension. J. Assoc. Inf. Syst. **14**, 312–338 (2013)
7. Gal, A.: Uncertain Schema Matching, vol. 3. Morgan & Claypool (2011)
8. Jagroep, E., Van der Werf, J.M., Broekman, J., Blom, L., van Vliet, R., Brinkkemper, S.: A resource utilization score for software energy consumption. In: Proceedings of ICT for Sustainability 2016 (2016)
9. Jans, M., Alles, M., Vasarhelyi, M.: The case for process mining in auditing: sources of value added and areas of application. Int. J. Account. Inf. Syst. **14**, 1–20 (2013)
10. Jean-Mary, Y.R., Shironoshita, E.P., Kabuka, M.R.: Ontology matching with semantic verification. Web Semant. **7**(3), 235–251 (2009)
11. Lambrix, P., Tan, H.: Sambo - a system for aligning and merging biomedical ontologies. J. Web Semant. **4**(3), 196–206 (2006)
12. Leopold, H., Niepert, M., Weidlich, M., Mendling, J., Dijkman, R., Stuckenschmidt, H.: Probabilistic optimization of semantic process model matching. In: Barros, A., Gal, A., Kindler, E. (eds.) BPM 2012. LNCS, vol. 7481, pp. 319–334. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32885-5_25
13. Li, G., de Murillas, E.G.L., de Carvalho, R.M., van der Aalst, W.M.P.: Extracting object-centric event logs to support process mining on databases. In: Mendling, J., Mouratidis, H. (eds.) CAiSE 2018. LNBIP, vol. 317, pp. 182–199. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-92901-9_16
14. Madhavan, J., Bernstein, P., Rahm, E.: Generic schema matching with cupid. In: Proceedings of the 27th VLDB Conference (2001)
15. Meilicke, C., Leopold, H., Kuss, E., Stuckenschmidt, H., Reijers, H.A.: Overcoming individual process model matcher weaknesses using ensemble matching. Decis. Support Syst. **100**, 15–26 (2017)
16. Murillas, E., Reijers, H., Aalst, W.: Connecting databases with process mining: a meta model and toolset. Softw. Syst. Model. 231–249 (2016)
17. Nikovski, D., Esenther, A., Ye, X., Shiba, M., Takayama, S.: Matcher composition methods for automatic schema matching. In: Cordeiro, J., Maciaszek, L.A., Filipe, J. (eds.) ICEIS 2012. LNBIP, vol. 141, pp. 108–123. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40654-6_7
18. Post, R., et al.: Active anomaly detection for key item selection in process auditing. In: Munoz-Gama, J., Lu, X. (eds.) ICPM 2021. LNBIP, vol. 433, pp. 167–179. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-98581-3_13
19. Saldaña, J.: The Coding Manual for Qualitative Researchers. Sage (2009)
20. Stein Dani, V., et al.: Towards understanding the role of the human in event log extraction. In: Marrella, A., Weber, B. (eds.) BPM 2021. LNBIP, vol. 436, pp. 86–98. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-94343-1_7
21. van der Aa, H., Leopold, H., Reijers, H.A.: Comparing textual descriptions to process models - the automatic detection of inconsistencies. Inf. Syst. **64**, 447–460 (2017)
22. Aalst, W.M.P.: Extracting event data from databases to unleash process mining. In: vom Brocke, J., Schmiedel, T. (eds.) BPM - Driving Innovation in a Digital World. MP, pp. 105–128. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-14430-6_8
23. Weidlich, M., Dijkman, R., Mendling, J.: The ICoP framework: identification of correspondences between process models. In: Pernici, B. (ed.) CAiSE 2010. LNCS, vol. 6051, pp. 483–498. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13094-6_37
24. Weske, M., Decker, G., Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Model collection of the bpm academic initiative (2020)