Enterprise Modelling and Information Systems Architectures
Vol. 12, No. 2 (2017). DOI:10.18417/emisa.12.2

Corrected version

LS3: Latent Semantic Analysis-based Similarity Search for Process Models

Andreas Schoknecht[*,a], Andreas Oberweis[a]

[a] Karlsruhe Institute of Technology, Institute AIFB - Building 05.20, KIT-Campus South, 76128 Karlsruhe, Germany

Abstract. *Large process model collections in use today contain hundreds or even thousands of conceptual process models. Search functionalities can help in handling such large collections for purposes such as duplicate detection or reuse of models. One popular stream of search functionalities is similarity-based search which utilizes similarity measures for finding similar models in a large collection. Most of these approaches base on an underlying alignment between the activities of the compared process models. Yet, such an alignment seems to be quite difficult to achieve according to the results of the Process Model Matching contests conducted in recent years. Therefore, the Latent Semantic Analysis-based Similarity Search (LS3) technique presented in this article does not rely on such an alignment, but uses a Latent Semantic Analysis-based similarity measure for retrieving similar models. An evaluation with 138 real-life process models shows a strong performance in terms of Precision, Recall, F-Measure, R-Precision and Precision-at-k, thereby outperforming five other techniques for similarity-based search. Additionally, the run time of the LS3 query calculation is significantly faster than any of the other approaches.*

Keywords. Process Model Querying • Similarity-based Search • Process Model Search • Latent Semantic Analysis

Communicated by S. Strecker. Received 2016-11-24. Accepted after 2 revisions on 2017-08-02.

1 Introduction

Calculating the similarity between business process models has been the focus of many research publications in the past years due to its importance regarding the management of large collections of process models. These collections can contain hundreds or even thousands of models nowadays, see, e. g., the collections mentioned by Lau et al. (2011) and Song et al. (2011), which makes sophisticated operations like conformance checking, duplicate detection, or the reuse of (parts of) models hard to conduct without automated support. For example, searching for duplicates or partially duplicated models in such collections would require an enormous manual effort. Automatic detection of similar models greatly helps in realizing such operations as, for instance, manual duplicate detection effort could only be applied to model pairs whose similarity value is above a certain threshold. Besides these application areas, process model similarity measures are also utilized for searching process model collections, which is also called *similarity-based search* (Dumas et al. 2009). In this context, a process model is used as query model, the input, with the aim to find similar models as output. These output models are then usually ranked by a decreasing similarity value to the input model.

Regarding the automatic similarity measurement between process models used in similarity-

* Corresponding author.
E-mail. andreas.schoknecht@kit.edu
We thank Tom Thaler from the Institute for Information Systems (IWi) in the German Research Center for Artificial Intelligence (DFKI) and Saarland University for his support during the similarity calculations conducted with the RefMod-Miner.
Note: This work is based on Schoknecht et al. (2017a).

based search techniques, many approaches have been published during the last years, see, e. g., the surveys by Schoknecht et al. 2017b and Becker and Laue (2012) for an overview. But while there are quite a lot of approaches, most of them base on an underlying alignment between the activities of the compared process models, which is also called *Process Model Matching* (Antunes et al. 2015). Thus, before calculating a final similarity value, these approaches require an alignment and, hence, the quality of the similarity calculation depends on it. Yet, such an alignment—and even more so a high quality alignment—seems to be quite difficult to achieve according to the results of the process model matching contests conducted in recent years (Antunes et al. 2015; Cayoglu et al. 2014). With respect to the results of the Process Model Matching Contest 2015 (Antunes et al. 2015), the participating matching techniques performed quite poorly. Even the best Recall values were below 0.7, which means that each matching technique did not detect at least 30 % of the correct matches. For one dataset the authors of the matching contest publication even stated that 36 % of the correct matches were not detected at all. Regarding the results of the first contest (Cayoglu et al. 2014), the overall performance had been even weaker compared to Antunes et al. (2015) with F-Measure values below 0.5.

The *Semantic Analysis-based Similarity Search* (LS3) approach described in the following circumvents the matching challenge by not requiring such a matching. Instead, it uses a *Latent Semantic Analysis-based Similarity Measure* (LSSM) which treats a whole process model as a text corpus and calculates a similarity value based on the contained words. It leaves aside the structure and the behavior of process models as these aspects are regarded less relevant with respect to the application in a search function. As an example, the order of activities might arguably be less important in a search function than in the context of conformance checking. While certain execution sequences of activities might be prohibited by legal regulations—which should be detected in the conformance checking case—from a search point of view the order of execution seems to be less relevant. In this case, it is more important to return models that describe the same process.

This article extends the work described in Schoknecht et al. (2017a) by additionally providing an algorithm for process model search using one specific process model as query. Besides, algorithms for handling model collection changes, i. e., insertion and deletion of models, are described. Finally, further comparative evaluations have been conducted, which include a comparison with five other state-of-the-art process model similarity measures. In Schoknecht et al. (2017a) the LS3 approach had been compared to a syntax-based information retrieval technique.

The evaluation results of the LS3 for a model collection containing 138 models are very promising as they yield an average F-Measure of 0.93 with a very high average Precision value of 0.97. Also average Precision-at-5 and R-Precision[1] are very high with values of 0.99 and 0.96. Besides, the run time is very competitive as a query is answered in 4.5 milliseconds on average. These results clearly outperform the performance of five other similarity-based search approaches.

The rest of the article is organized as follows: In Sect. 2 the underlying problem is described while also introducing basic terminology and definitions. Besides, related work with regard to similarity-based search and querying of process model collections is described. Afterwards, the newly proposed *Latent Semantic Analysis-based Similarity Search* is presented in Sect. 3. Thereby, we present algorithms for finding all similar models for each model in a collection, for finding similar models for a specific query model, and for handling insertion and deletion of models with respect to the underlying search structure. The evaluation and discussion of the results in Sect. 4 focuses then on the performance in terms of Precision, Recall, F-Measure, Precision-at-k, and R-Precision as well as the run time. Additionally,

[1] For further information on the used evaluation measures Precision, Recall, F-Measure, Precision-at-k, and R-Precision see Sect. 4 or Manning et al. (2008).

we compared the LS3 approach to five other techniques for similarity-based search. Finally, Sect. 5 summarizes the results of the article and provides an outlook on future research.

2 Similarity-based Search for Process Models

2.1 Business Process Models

The term 'Business Process' refers to sequences of manual or (partly) automated activities executed in a company or organization according to specific rules with a certain aim. Such processes are typically represented as process models with the help of business process modeling languages like EPC (Keller et al. 1992), BPMN (Business Process Model and Notation (BPMN) 2011), or Petri Nets (Reisig 1985) for various reasons like documentation or process analysis. While the LS3 approach could generally be applied for all the mentioned languages, the process models in the paper at hand are represented as Petri Nets as we used this representation for our prototypical implementation. The following definition provides a formal description based on Reisig (1985).

Definition 1 *Petri Net: A Petri Net is a 4-Tuple $PN = (P, T, F, \ell)$ with*

- *$P = \{p_1, p_2, \cdots, p_m\}$ being a finite set of places,*
- *$T = \{t_1, t_2, \cdots, t_n\}$ being a finite set of transitions, i. e., the activities of a process,*
- *$F \subseteq (P \times T) \cup (T \times P)$ being a set of arcs representing the control flow of a process and*
- *$\ell : P \cup T \longrightarrow L$ being a labeling function which assigns a label $l \in L$ to each place $p \in P$ and to each transition $t \in T$ with L being a set of labels.*
- *Additionally, it holds that $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$.*

Note that we specifically included a labeling function for the transitions and places as the *Latent Semantic Analysis-based Similarity Measure* used in the LS3 utilizes these labels to determine a similarity value between process models. In principle, these labels can contain information for various process perspectives. For instance, a transition label might describe the action, which has to be carried out by a specific role on a specific data object in the context of a process. An overview on different approaches and their specifics for transforming various business process modeling languages to Petri Nets can be in found in (Lohmann et al. 2009).

2.2 Problem Description

The similarity-based search approach described in the following section can be used to find process models from a process model collection M which have a certain degree of similarity to a *query model* q. Thereby, the similarity of models is based on a threshold value θ, i. e., a model is considered similar to another model if their degree of similarity is equal to or higher than θ. The similarity measure used in the following is based on *Latent Semantic Analysis* (LSA) (Dumais 1991), which has originally been developed for information retrieval. As the LSA analyzes the word choice of text passages based on so called *document vectors* representing word occurrences, a corresponding representation has to be defined for process models. Therefore, each process model is considered a document for which a document vector is created (Schoknecht et al. 2017a):

Definition 2 *Document vector of a process model: Let W_{all} be a set of terms, which contains all distinct terms of a process model collection. M is a set of process models (the process model collection) and $w(m)$ is a function, which returns the set of all terms (Bag-of-Words) W_m of a process model $m \in M$ with $W_m \subseteq W_{all}$, i. e., $W_{all} = \bigcup w(m)$ for all $m \in M$. The vector $d_m = (w_{1m}, w_{2m}, \ldots, w_{tm})$ then represents the document vector of process model m, whereby each index i represents a term of the set of all terms W_{all}, which are contained in the process model collection. An entry w_{im} reflects the term* *frequency weight which describes how often a certain term exists in a process model.*

Important to note is that we treat a query model q in principle as any other process model of a collection, i. e., we represent q as a document vector, too. The *Latent-Semantic Similarity Measure* (LSSM) introduced in Sect. 3.1 calculates a similarity value between the vector representations of q and each model $m \in M$ of a collection in the interval $[0, 1]$. As a result of such a query, all models in M exhibiting a similarity value equal to or higher than a threshold value θ are returned. Hence, the following result set QR is calculated by the LS3:

$$QR = \{m \mid m \in M \wedge LSSM(q, m) \geq \theta\} \quad (1)$$

Note also that, although we use only Petri Net-based process models, the LS3 approach for similarity search could also be applied to models in other notations. We restrict the presentation of the LS3 approach to Petri Net-based models for two reasons. Firstly, the current implementation of the LS3 only allows the processing of Petri Nets stored as PNML files (Weber and Kindler 2003). Secondly, the document vector representation of a process model depends on the modeling language used. One example would be the consideration of pools and lanes or data elements in BPMN diagrams, which could be treated differently than the labels of Petri Nets in definition 2. One decision to make would be if the words from lane labels should be included only once or if the word frequency should be weighted based on the number of activities in a lane.

2.3 Related Work

Two major streams of research can be identified which are related to the LS3 approach. The first one focuses on querying languages for process model collections, while the second one uses existing process models for similarity-based search. Works falling into the first category include, e. g., the specification of query languages for process models in BPMN (Awad 2007) or BPEL (Beeri et al. 2008). Going one step further, APQL has been published recently, which can be used independently of a particular process modeling language (ter Hofstede et al. 2013). Besides, the GMQL query language (Delfmann et al. 2015) has also been developed for querying process model collections with models in arbitrary notations. Furthermore, an approach for querying model-based process descriptions as well as corresponding textual process descriptions is described in (Leopold et al. 2016). Process model labels and natural language texts are thereby stored in a common data format, which can be queried using SPARQL. Another query language heading in this direction is described in (Fill 2016). It uses semantic model annotations as the foundation for model querying.

Yet, common to all these approaches is the need to formulate a query with the respective query language, whereas existing process models cannot be used as a query. In this respect the LS3 approach differs from the aforementioned works as it uses existing models as queries. Hence, it is closer to the second research stream.

An early work on similarity-based search is described in Dijkman et al. (2009). The presented algorithms focus on the graph structure of process models to determine a similarity value, while requiring similarity values of labels, i. e., an alignment of process model elements. The LS3 approach, however, explicitly focuses on the textual labels and does not require such an alignment. The approach described in Kunze et al. (2015) does also require an alignment while focusing on the behavior of process models. Furthermore, Kunze et al. (2015) does not calculate a similarity value, but requires models to include all of the behavior of a query model. The approach by Gater et al. (2012) uses an index based on behavioral characteristics of a model to speed up the query answering. Speeding up the querying process is also the aim of Yan et al. (2012) by using simple but representative abstractions of process models, so called features. Kastner et al. (2009) focuses on the structural aspects and clustering of workflows for similarity-based search, which the LS3 does not use. The technique described by Awad et al. (2008), however, is closer to our approach. They use an Enhanced Topic-based Vector Space Model, which is based on an ontology, for the comparison of a BPMN-Q query against process models. The LS3 in contrast does not rely on an ontology but on LSA to calculate a similarity value, hence there is no effort required for constructing such an ontology. Another approach by Qiao et al. (2011) uses topic language modeling, specifically the Latent Dirichlet Allocation (LDA) (Blei et al. 2003), combined with a matching-based structural similarity calculation and clustering to retrieve similar models. The LDA aspect is closely related to our Latent Semantic Analysis approach. But while the LSA compares vectors in a k dimensional space to determine a similarity value, the LDA-based approach requires the estimation of the probability that a process model can generate a certain query. However, this approach performed quite poorly in the reported evaluation with average precision values ranging from 0.4 to 0.79 for different model collections so that we decided to go one step back and proceed with the conceptually simpler LSA approach instead of the LDA. But although the evaluation results are very good for the LS3 search (see Sect. 4), incorporating LDA or similarity calculations based on word embeddings using, e. g., WORD2VEC (Mikolov et al. 2013) might be a promising future research direction as such methods are generally considered to exhibit a better performance in computational linguistics than LSA (Baroni et al. 2014).

Another recent approach described in Li and Cao (2015) performs a similarity search based on the structure of process models and by calculating an alignment of model elements using WORDNET (Miller 1995) to calculate the semantic relatedness of labels. In contrast, the LS3 approach does not require such an alignment nor WORDNET.

We acknowledge that there exist further publications describing a similarity measure for process models which could be used in a search technique. To our knowledge, the aforementioned publications relate most closely to our approach of similarity-based search. For an overview of further related publications see, e. g., the surveys in Becker and Laue (2012) and Schoknecht et al. (2017b). Furthermore, in the context of Process Model Matching, Klinkmüller et al. (2013) use a general bag-of-words approach for calculating an alignment of activities.

To conclude, most existing similarity-based search techniques require an alignment of process model elements (usually between activities) to determine a similarity value. Thereby, the alignment calculation itself is typically based on a linguistic comparison of the labels of the model elements combined with further structural or behavioral properties, see, e. g., Antunes et al. (2015) and Cayoglu et al. (2014) for an overview on different matching techniques. Additionally, the existing similarity measures use structural or behavioral properties of process models like Graph-Edit Distance to determine the final similarity score. The LS3 on the contrary focuses only on the textual content of process models, i. e. the labels, without considering structure or behavior. Therefore, it is conceptually simpler than most existing approaches, yet the evaluation shows that this is a promising direction.

2.4 Latent Semantic Analysis

Latent Semantic Analysis (LSA) is both a theory and a mathematical/statistical method for capturing the meaning of words and documents. It was developed to improve document retrieval based on user queries (Deerwester et al. 1990). LSA goes one step further than ordinary indexing techniques—which are based on a syntactical comparison of terms of the query and the documents (so called word matching approach)—by adding a semantic aspect to these comparisons. As a consequence, meanings of terms and documents can be used to improve retrieval results (Dumais et al. 1988).

LSA as a theory is based on the assumption that the semantics of terms are determined by the meanings of documents in which they appear and vice versa that the semantics of a document is determined by the meanings of the terms contained in it. Therefore the LSA is associated with the bag of words approach (Landauer 2007). The LSA aims at analyzing the choice of words, which reflects a hidden respectively latent semantic structure of a document (Landauer et al. 1998). This can be summed up by the following equation, which describes that the meaning of a text passage is the sum of the meanings of the terms it contains (Landauer 2007):

$$meaning_{passage} = \sum m_{term1}, m_{term2}, \ldots, m_{termn} \tag{2}$$

Equation 2 shows the LSA's main difference to common bag of words approaches: common bag of words techniques are based on the assumption that a bag of words (as a unit) reflects the semantics of the corresponding document, whereas the LSA uses (in addition) the semantics of each term (contained in a bag of words) to determine the semantics of a document (Landauer 2007; Turney and Pantel 2010).

LSA as a mathematical/statistical method is based on the vector space model, which can be represented by a *Term-Document Matrix* (TDM). In this context a document is a text passage with a predefined length and a term is a word or a meaningful unit (e. g. statue of liberty). Figure 1 shows the schematic structure of a Term-Document Matrix.

$$\begin{array}{c} \\ t_1 \\ t_2 \\ \vdots \\ t_t \end{array} \begin{array}{c} \begin{array}{cccc} D_1 & D_2 & \cdots & D_n \end{array} \\ \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{t1} & w_{t2} & \cdots & w_{tn} \end{pmatrix} \end{array}$$

Figure 1: Schematic structure of a Term-Document Matrix

The columns of the matrix represent the documents $D_1, D_2, \ldots, D_n$, whereas the rows represent the terms $t_1, t_2, \ldots, t_t$. An entry in the Term-Document Matrix represents the weight $(w_{11}, \ldots, w_{tn})$ of a specific term in a specific document such as the term frequency. By applying singular value decomposition, a $t \times n$ Term-Document Matrix A with rank r is decomposed into three matrices T, Σ and D^T. The rows of the orthogonal matrix T describe the term vectors whereas the columns of the orthogonal matrix D^T describe the document vectors. Matrix Σ contains the corresponding singular values sorted in decreasing order.

For analyzing semantics, the LSA uses a so called semantic space. It is created by keeping only the k largest singular values and the corresponding values of the term and document vectors thereby effectively reducing the dimensionality of the vector space spanned by the term or document vectors. This step is supposed to reduce noise within the document collection, i. e., handling problems related to synonymy and polysemy of words (Deerwester et al. 1990). Figure 2 shows how the dimension reduction (combined with the singular value decomposition) effects the vector representations and thus revealing semantics and similarities (Dumais 2007).

In a common vector space model the orthogonal dimensions are formed by unique terms and documents are represented as vectors by their contained terms. By using less dimensions than unique terms the relation of terms to specific latent semantic dimensions is revealed. Consequently the semantics of a term is determined by the direction of its vector.

The product A_k of the generated matrices T_k, Σ_k and $D_k{}^T$ is the best least squares approximation of the original Term-Document Matrix A (Martin and Berry 2007). The explained singular value decomposition and the reduction on k dimensions is shown in Figure 3. The four main steps of the LSA-based retrieval procedure according to Dumais (2007) are then as follows:

1. *Extraction of terms for generating a Term-Document Matrix:* In this process step, terms are extracted from each document of the examined collection. It can be useful to apply Natural Language Processing concepts before constructing the Term-Document Matrix (e. g., removal of stop words). The result of this step is a Term-Document Matrix containing term frequencies.

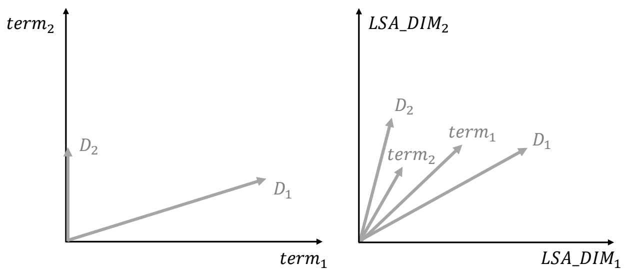


Figure 2: Schematic effect of dimension reduction on vector representations

2. *Transformation of the Term-Document Matrix:* In this step, the entries of the Term-Document Matrix (term frequencies) are transformed by applying a weighting scheme like log-entropy weighting (Landauer et al. 1998), which enables better differentiation of documents.

3. *Singular value decomposition and dimension reduction:* This step decomposes the Term-Document Matrix and truncates the resulting matrices as shown in Figure 3. The result of this step is the so called *semantic space*.

4. *Retrieval in semantic space:* Term-Term, Document-Document and Term-Document comparisons are computed in the semantic space by applying an appropriate similarity measure for vectors such as the cosine similarity (Rijsbergen 1979, p. 25).

For further information on the LSA and its specifics (e. g., dimension reduction and different weighting schemes), we refer to (Landauer et al. 1998). This article also provides an example for the calculations conducted for querying document collections.

3 LS3: Latent Semantic Analysis-based Similarity Search

We describe the *Latent Semantic Analysis-based Similarity Search* (LS3) in this section, whereby we first introduce an approach for finding similar models for each model in a collection (LS3-QueryAll approach). Subsequently, we describe how a specific query model can be used for querying a model collection (LS3-Query approach). Thirdly, we explain how insertion, deletion and updating of models can be handled with respect to the underlying search structure of the LS3 search approaches.

3.1 Finding Similar Models in a Model Collection

The similarity-based search between all process models of a collection is conducted according to the following five steps, which are also illustrated in Figure 4. Thereby, the first four steps represent the similarity value calculation of the *Latent Semantic Analysis-based Similarity Measure* (LSSM), while the fifth step represents the actual query result calculation. The underlying algorithm is shown in Algorithm 1.

1. *Extraction of terms for generating a Term-Document Matrix:* For generating a Term-Document Matrix every process model has to be represented by its document vector as specified in Definition 2. These vectors then form the columns of the Term-Document Matrix after step 1 in Figure 4.

In our case of Petri Net-based process models, we first extract the distinct terms of all transition and place labels L from each process model $m \in M$, whereby the set M contains all models of a collection (L according to Definition 1). Then, the following three preprocessing steps are executed: (1) All words are transformed into lower-case letters, (2) stop words are removed and (3) the remaining terms are stemmed[2] according to Porter (1980) (Porter-Stemmer).

[2] In the evaluation all labels are in English. For other natural languages different stemmers would be need.

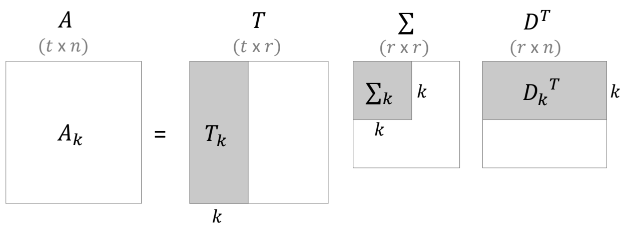
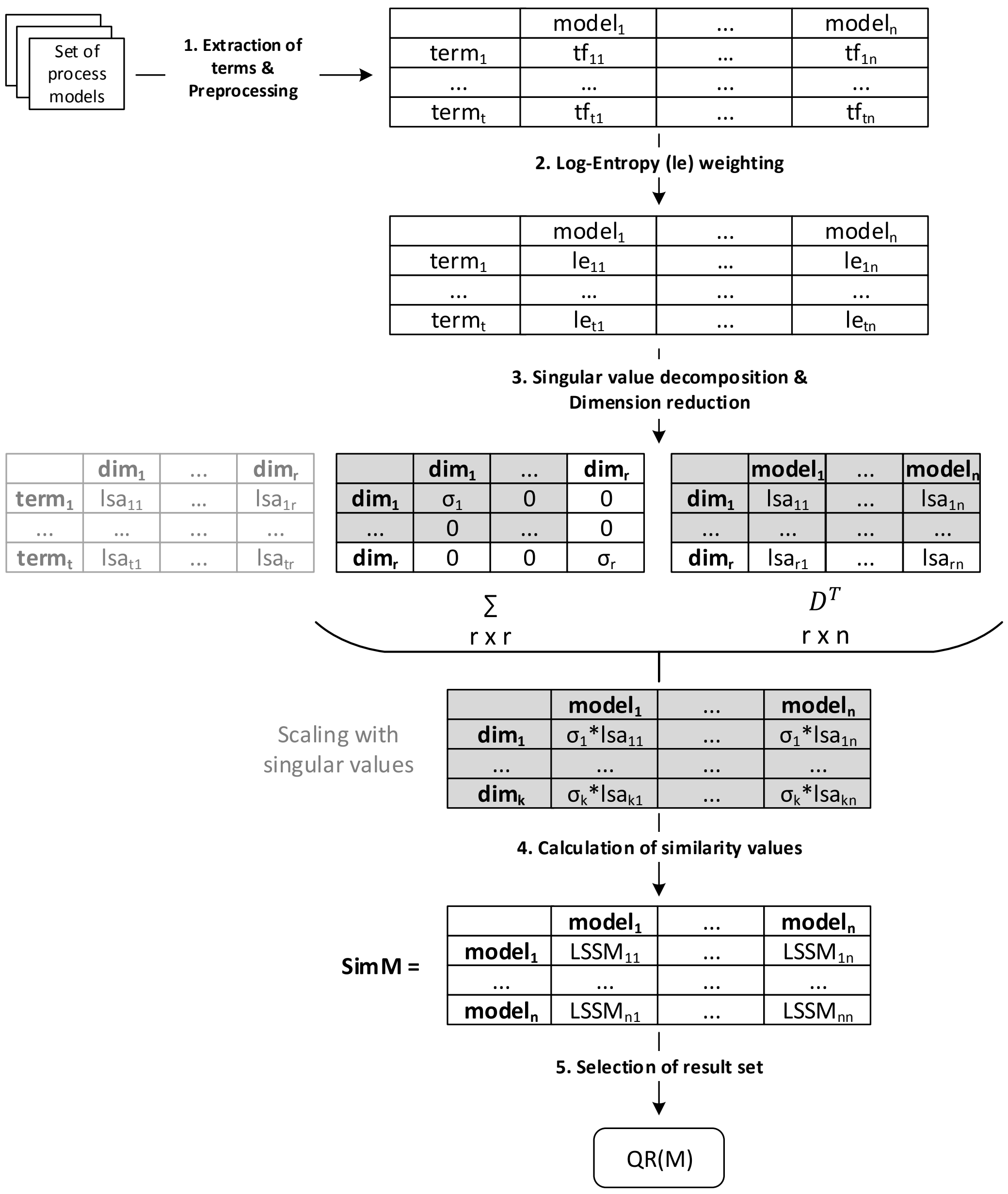


Figure 3: Singular value decomposition and dimension reduction


Set of process models
1. Extraction of terms & Preprocessing
2. Log-Entropy (le) weighting
3. Singular value decomposition & Dimension reduction
$\sum$
r x r
D^T
r x n
Scaling with singular values
4. Calculation of similarity values
SimM =
5. Selection of result set
QR(M)

	model$_1$	…	model$_n$
term$_1$	tf$_{11}$	…	tf$_{1n}$
…	…	…	…
term$_t$	tf$_{t1}$	…	tf$_{tn}$

	model$_1$	…	model$_n$
term$_1$	le$_{11}$	…	le$_{1n}$
…	…	…	…
term$_t$	le$_{t1}$	…	le$_{tn}$

	dim$_1$	…	dim$_r$
term$_1$	lsa$_{11}$	…	lsa$_{1r}$
…	…	…	…
term$_t$	lsa$_{t1}$	…	lsa$_{tr}$

	dim$_1$	…	dim$_r$
dim$_1$	σ$_1$	0	0
…	0	…	0
dim$_r$	0	0	σ$_r$

	model$_1$	…	model$_n$
dim$_1$	lsa$_{11}$	…	lsa$_{1n}$
…	…	…	…
dim$_r$	lsa$_{r1}$	…	lsa$_{rn}$

	model$_1$	…	model$_n$
dim$_1$	σ$_1$*lsa$_{11}$	…	σ$_1$*lsa$_{1n}$
…	…	…	…
dim$_k$	σ$_k$*lsa$_{k1}$	…	σ$_k$*lsa$_{kn}$

	model$_1$	…	model$_n$
model$_1$	LSSM$_{11}$	…	LSSM$_{1n}$
…	…	…	…
model$_n$	LSSM$_{n1}$	…	LSSM$_{nn}$

Figure 4: Conceptual LS3 search approach for calculating all similar models in a process model collection

Algorithm 1: QueryAll(M,t,k)

```
input  : Model collection $M$, threshold $t$,
         dimensionality parameter $k$.
output : A set of tuples $QR = \{(m, M_{sim})\}$ each
         containing a model $m \in M$ and the set of
         similar models $M_{sim}$.
/* Calculate similarity values matrix
   SimM. */
1  TDM = calculateTDMatrix($M$);
2  WTDM = weightTDMatrix($TDM$);
3  SVD = calculateSVD($TDM$);
4  SVDK = calculateReducedSVD (SVD,k);
5  SimM = calculateLSSM($SVDK$);
/* Calculate QueryAll results. */
6  QR = ∅;
7  for each (row i in SimM) do
8      results = ∅;
9      for each (column j in SimM) do
10         if ($SimM[i][j] \geq \theta$) then
11             Add model $m_j$ to $results$;
12         end
13     end
14     Add tuple $(m_i, results)$ to $QR$;
15 end
16 return QR;
```

This preprocessing is conducted to reduce the vector space by eliminating terms (stop words) that do not contribute to the semantics of a process model. The stemming of terms is executed to ensure that words which have the same stem are assigned to the same term. Through that originally syntactically different terms increase the count of the same stemmed term in the Term-Document Matrix instead of being listed as separate terms.

Finally, this step results in a Term-Document Matrix containing absolute term frequencies t_{ij} as matrix entries. These term frequencies specify how often a term i occurs in model j. It holds that $1 \leq i \leq t$ with t being the amount of unique terms and $1 \leq j \leq n$ with n being the amount of models within the process model collection M. This step is conducted in line 1 of Algorithm 1.

2. *Transformation of the Term-Document Matrix:* As the usage of absolute term frequencies is discouraged in studies (Zaman and Brown 2010), we apply the *log-entropy* weighting scheme (Landauer et al. 1998) on the values of the Term-Document Matrix generated in step 1. The log-entropy weighting scheme is defined as follows:

$$le_{ij} = log_2(tf_{ij} + 1)\cdot \\ (1 + \sum_{j=1}^{n} \frac{p_{ij} \cdot log_2 p_{ij}}{\log_2 n}), \ \ \forall\, tf_{ij} > 0 \qquad (3)$$

Thereby, $p_{ij} = \frac{tf_{ij}}{gf_i}$ is the quotient of the term frequency tf_{ij} and the global frequency gf_i of term i. The global frequency indicates how often a term i appears in the whole process model collection.

The log-entropy weighting scheme consists of two parts. On the one hand there exists a local weighting function reducing big differences between term frequencies by a logarithmic transformation. On the other hand, the Shannon entropy (Shannon 1948) is selected as a global weighting function. By this, high weights are assigned to terms having high information content as they enable better differentiation between process models, whereas terms with low information content receive low weights. Terms have a high information content when they appear in few models and a low information content when they appear in many models (Cover and Thomas 2006). See also the second step of Figure 4 and line 2 of Algorithm 1.

The intuition behind this weighting scheme is that terms appearing infrequently in the models should receive a higher weight as they can be used to differentiate models. E.g., in case of finding similar customer process models the term “customer” might appear in all models, whether it might be a complaints handling or a customer contact process model. Hence, the term “customer” is less suited for differentiating these models and should therefore receive a lower weight compared to “complaint”, which can be used to differentiate the two process models.

3. *Singular value decomposition and dimension reduction:* In this step, the transformed Term-Document Matrix is decomposed into three matrices (see step 3 in Figure 4 and line 3 of Algorithm 1). Only the matrices Σ and D^T are relevant for the purpose of determining the similarity between process models. The matrix D^T contains the calculated process model vectors, while matrix Σ contains the corresponding singular values sorted in descending order. The amount of singular values greater than zero corresponds to the rank r of the original Term-Document Matrix. Therefore the upper boundary for choosing a certain amount of dimensions for the following dimension reduction is determined by the rank r. By choosing r dimensions the calculated similarity values in step 4 are the same as if they were calculated with the original (transformed) Term-Document Matrix.

For values $k < r$ the k highest singular values are kept, whereas the remaining values are set to 0. The grey colouring of the matrices in Figure 4 visualises this dimension reduction step (see also line 4 of Algorithm 1). The determination of a dimension k is application-dependent—thus, there exists no general recommendation for an optimal retrieval quality (Dumais 1991).

4. *Similarity value calculation:* Before calculating the similarity between two column vectors, it is necessary to scale the entries of these vectors with their corresponding singular values (Deerwester et al. 1990; Martin and Berry 2007). The semantics of a process model is represented by the direction of its document vector, whereas the length of a vector might be misleading. To be independent of the vector length, we prefer the cosine similarity over distance measures (e. g., the euclidean distance) (Turney and Pantel 2010). As the degree of similarity between two models respectively their vector representations is based on the cosine similarity (Rijsbergen 1979, p. 25), the angle between the vector representations is decisive. The cosine similarity is defined as follows:

$$cos_{sim}(d_1, d_2) = \frac{d_1 \cdot d_2}{|d_1| \cdot |d_2|} = \frac{\sum_{i=1}^{n} d_{i1} \cdot d_{i2}}{\sqrt{\sum_{i=1}^{n} (d_{i1})^2} \cdot \sqrt{\sum_{i=1}^{n} (d_{i2})^2}} \quad (4)$$

While the numerator calculates the scalar product of the two vectors d_1 and d_2, the denominator calculates the product of the euclidean lengths of the two vectors. This results in a value range in the interval $[-1, 1]$ for the cosine similarity with 1 indicating the highest possible similarity and -1 indicating the largest possible difference. In a vector space model, the value 1 corresponds to a degree of 0° and the value -1 corresponds to a degree of 180°.

For the actual determination of the similarity between two models m_x and m_y, we then use the cosine similarity of their vector representations transformed on the interval $[0, 1]$ as specified in Equation 5. This interval is used frequently for expressing the degree of similarity with 0 indicating dissimilarity and 1 indicating equality.

$$LSSM(m_x, m_y) = \frac{cos_{sim}(m_x, m_y) + 1}{2} \quad (5)$$

For the calculation of the $n \times n$ similarity matrix $SimM$ Equation 5 is applied on every combination of models respectively vectors. The entries of the matrix $SimM_{xy}, 1 \leq x, y \leq n$ then correspond to the similarity values $LSSM(m_x, m_y)$. For clarification see step 4 in Figure 4 and line 5 in Algorithm 1.

5. *Retrieval of query results:* After generating the similarity matrix $SimM$, all similar models in a process model repository are calculated by comparing the similarity values to a threshold value θ ($0 \leq \theta \leq 1$). Eventually, the retrieval result of a similarity based search $QR(M)$ contains a set of tupels $(m_x, \{m_y\})$, which specify the set of similar models $\{m_y\}$ for a model m_x by applying Equation 6 (see also lines 6–16 in Algorithm 1):

$$QR(M) = \{(m_x, \{m_y\}) \mid SimM_{xy} \geq \theta\} \quad \forall\, x, y, 1 \leq x, y \leq n \tag{6}$$

3.2 Model Querying

The computation of query results based on a query model q is shown in Algorithm 2. Thereby, the first four lines represent the same steps as in the LS3-QueryAll approach described previously. I. e., a weighted Term-Document Matrix is generated, which is then decomposed by singular value decomposition and projected onto k dimensions. The remaining lines represent three new steps:

1. *Generation of a pseudo document:* To be able to find similar models in a model collection, the query model q has to be projected into the k-dimensional vector space computed according to the LSA concept. Therefore, a so called *pseudo document* has to be created (line 5 in Algorithm 2), which is essentially a document vector of a process model (see Definition 2). Referring to Deerwester et al. (1990), a pseudo document of a query model q can be calculated according to Equation 7.

$$pseudoDoc = q^T U_k \Sigma_k^{-1} \tag{7}$$

Thereby, q^T is a log-entropy weighted vector of the term frequencies of query model q while term frequencies of further terms appearing in the model collection M but not in q are set to 0. U_k and Σ_k are the k-dimensional term and singular value matrices of the LSA.

Two further aspects should be noted with respect to vector q^T. (1) Terms appearing in q but not in M are not considered, i. e., the original Term-Document Matrix from M is not changed. This makes sense as we want to project q into the existing vector space for finding similar models but we do not want to include q into the model collection M. (2) Regarding the log-entropy weighting of terms in q, we decided to not change the absolute frequency of terms. Hence, we weigh the terms with the same term frequency and document number values as when originally generating the weighted Term-Document Matrix of M. The reason is, again, that we want to project q into the existing vector space without changing M.

2. *Calculation of similarity values:* After generating a pseudo document from q, the LSSM similarity values between *pseudoDoc* and all models $m \in M$ are calculated (line 6 in Algorithm 2). This calculation is proceeded as previously described in Equation 5.

3. *Finding similar models:* Finally, the result set of similar models to q is determined according to lines 7–13 of Algorithm 2. Each model in M having an equal or higher LSSM similarity value to q than θ is added to the result set QR:

$$QR(q, M) = \{m_i \mid LSSM(q, m_i) \geq \theta\} \quad \forall\, m_i \in M \tag{8}$$

Also note that the reduced singular value decomposition has to be created only once for a model collection to answer any number of queries as long as the collection is not changed, i. e., no models are inserted, deleted, or changed.

3.3 Insertion, Deletion and Updating of Models

When a model collection is changed through the insertion of a new model, the deletion of an existing model or through updating a model, i. e., changing an existing model, the LS3 search cannot directly incorporate the changes of the collection in the querying results. As the semantic querying space is constructed once from a Term-Document Matrix, a mechanism is needed which enables the inclusion of changed model data in terms of the contained terms and term frequencies into the semantic vector space. To do this, the algorithms described in the following base on a Term-Document Matrix which has been created from a model collection and change the corresponding matrix. Afterwards, the newly created Term-Document Matrix has to be processed again to be able to calculate LSSM similarity values and query the semantic space.

Algorithm 2: Query(M,q,t,k)

```
input  : Model collection $M$, query model $q$, threshold
         $t$, dimensionality parameter $k$.
output : A set of models $QR$ containing the models
         similar to $q$.
   /* Calculate reduced singular value
      decomposition. */
1  TDM = calculateTDMatrix($M$);
2  WTDM = weightTDMatrix($TDM$);
3  SVD = calculateSVD($TDM$);
4  SVDK = calculateReducedSVD (SVD,k);
   /* Calculate pseudo document. */
5  pseudoDoc = generatePseudoDoc (q,SVDK);
   /* Calculate similarity values. */
6  SimValues[] = calculateLSSM($SVDK,pseudoDoc$);
   /* Calculate query results. */
7  QR = $\emptyset$;
8  for ($i = 0$ to $SimValues.length - 1$) do
9      if ($SimValues[i] \geq \theta$) then
10         Add model $m_i$ to $QR$;
11     end
12 end
13 return QR;
```

Algorithm 3: InsertModel(TDM,m)

```
input  : Term-Document Matrix $TDM[i][j]$, model $m$.
output : Output Term-Document Matrix with $m$
         included.
1  int n = amountColumns (TDM);
2  int t = amountTerms (TDM);
3  MultiSet<String> termsM = extractTerms($m$);
4  String[] termsTDM = getTerms($TDM$);
5  int[t] mVector;
   /* Calculate column vector of $m$. */
6  for ($i = 0$ to $t - 1$) do
7      term = termsTDM[i];
8      if ($termsM.contains(term)$) then
9          $mVector[i] = termsM.count(term)$;
10         termsM.remove(term);
11     end
12     else
13         $mVector[i] = 0$
14     end
15 end
16 TDM = addColumn (TDM, mVector);
   /* Add additional terms of $m$. */
17 for ($k = 0$ to $termsM.size() - 1$) do
18     TDM = addRow (TDM, termsM.get(k));
19     for ($j = 0$ to $n - 1$) do
20         TDM[t+k][j] = 0;
21     end
22     TDM[t+k][n] = termsM.count(termsM.get(k));
23 end
24 return TDM;
```

The algorithms for updating the semantic query space of a model collection operate on a Term-Document Matrix thereby limiting the performance penalty when changing a model collection. Only newly inserted or changed models have to be parsed for the terms they contain and their term frequencies, while the Term-Document Matrix is adapted based only on these data. In case of model deletions it is even only necessary to know which models should be deleted.

The pseudo code in Algorithm 3 shows the procedure for updating a Term-Document Matrix (TDM) in case new models are inserted in a collection. First, the terms and frequencies contained in the new model *m* are extracted and stored in a multi set (line 3). After that, lines 4–16 calculate the frequency entries for a vector representation of *m* according to Definition 2. For each term contained in the original Term-Document Matrix in case the term is also contained in *m*, the corresponding vector entry is set to the number of occurrences in model *m* (line 9) and the term is removed from the multi set created in line 3 (line 10). Otherwise the entry is set to 0 (line 13). Afterwards, a new column vector for *m* is added to the TDM.

As a new model might contain terms not already included in the TDM, these new terms have to be added to the TDM, too. This is done in lines 17–23. For each term remaining in the multi set of *m* a new row vector is added to the TDM (line 18), the entries are set to 0 in case the column represents a model other than *m* (lines 19–21), and for the column representing *m* the entry is set to the term frequency in *m* (line 22). Finally, an updated TDM is returned in line 24.

Algorithm 4 shows the procedure for deleting an existing model *m* from a collection. As in the case of model insertion, the TDM for the model collection has to be updated to reflect the model

Algorithm 4: DeleteModel(TDM,m)

```
input  : Term-Document Matrix $TDM[i][j]$, model $m$.
output : Output Term-Document Matrix with $m$
         removed.
TDM = removeColumn (TDM, m);
/* Remove empty term rows of $TDM$. */
int n = amountColumns (TDM);
int t = amountTerms (TDM);
boolean isNull = true;
for ($i = 0$ to $t - 1$) do
    for ($j = 0$ to $n - 1$) do
        if ($TDM[i][j] \neq 0$) then
            isNull = false;
        end
    end
    if ($isNull == true$) then
        TDM = removeRow (TDM, i);
        $t = t - 1$;
        $i = i - 1$;
    end
    isNull = true;
end
return TDM;
```

deletion. Therefore, the column representing model *m* is deleted first (line 1). Afterwards we need to check whether any term vectors are not necessary anymore, i. e., we remove row vectors which only contain frequency values of 0 as this means that the corresponding term is not contained in any model of the model collection anymore. The search and removal for such term vectors are conducted in lines 2–17. Finally, the new TDM is returned in line 18.

Lastly, an update of a model—e. g., changes in a model like adding activities or changing a label—is treated as removal of the original model according to Algorithm 4 and then adding the updated model to the collection according to Algorithm 3 afterwards. Hence, an update is just a combination of the deletion and insertion algorithms presented previously.

In principle, we could have also updated the semantic querying space according to existing proposals from the LSA literature. These folding-in and folding-out approaches (Berry et al. 1995) might be faster than our solution as they do not have to recalculate the semantic space. Yet, these techniques might have also reduced query accuracy so we decided against them. Besides, our evaluation in Sect. 4 shows that at least for medium sized model collections the semantic space calculation does not prevent the LS3 search from being used in practical usage scenarios. If the performance would decrease for very large model collections, one could also resort to other distributed algorithms for SVD calculation (Řehůřek 2011).

4 Evaluation

4.1 Evaluation setup

For the empirical evaluation of the LS3 we used three real-life process model collections. These three can be divided into the categories mined models, field models, and models from controlled modeling environments to provide for different model sets with varying matching difficulty as has been described in Thaler et al. (2017). This is especially interesting as the results show that matching-based similarity search approaches perform indeed worse in case of higher matching difficulty.

The first model set, which has been used, was introduced by Vogelaar et al. (2011). This collection contains models of eight different business processes performed by 10 different dutch municipalities (DM). Hence, there exist 80 process models in the collection. This model collection is linguistically harmonized, i. e., node labels are unambiguous and consistent. Thus, same activities are labeled in the same manner, which should be helpful to search approaches calculating similarity values based on an alignment of activities. To provide an example of the term linguistically harmonized, consider an activity “Fill in payment data and approve” from one process model of one of the municipalities. If in any of the other nine municipalities payment data has to be filled in and be approved, then the corresponding activity would also be labelled “Fill in payment data and approve”. As also described in Thaler et al. (2017), these models belong to the mined models category, for which correct matches can be calculated comparatively easy.

The second model collection contains two model sets of the Process Model Matching Contest 2015 (Antunes et al. 2015), namely the *University Admission* (UA) and *Birth Registration* (BR) data sets. Each model set contains nine models describing the admission processes of nine German universities and, respectively, the birth registration processes of different countries. These models are not linguistically harmonized, i. e., they contain semantically equal activities described with labels using a differing word choice. This collection belongs to the field category according to Thaler et al. (2017) for which correct matches are difficult to calculate.[3]

Finally, the third model collection belongs to the controlled environment category. We used models provided by Camunda.[4] This model collection (CM models) resulted from various BPMN training sessions, in which students had to model processes according to a textual description. We selected ten models for each of the four processes of the complete data set for the evaluation.

Further characteristics of the different model sets can be found in Tab. 1. We combined all models in one collection so that the total amount of models used for the evaluation is 138, which refer to 14 underlying processes.[5]

For our purpose, we transformed these models from the original representations into Petri Nets and stored them in the XML-based format PNML (Weber and Kindler 2003), which enables an automatic processing of the models. A prototype[6] of the LS3 has been implemented in JAVA using the STANFORD PARSER for tokenizing labels (Klein and Manning 2003). The evaluation has been conducted on a laptop with the following specifications: Intel(R) i7-4750HQ CPU, 4 GB RAM, Windows 8.1 and JAVA 1.8.

[3] See also the matching results in Antunes et al. (2015).
[4] For further details on this model set see https://github.com/camunda/bpmn-for-research.
[5] The models can be obtained from http://butler.aifb.kit.edu/asc/models.zip
[6] The JAVA code can be obtained from https://github.com/ASchoknecht/LS3.

To evaluate the LS3 we, firstly, determined Precision, Recall and F-Measure on the basis of the described process model sets. Thereby, Precision is defined as the fraction of relevant and received results (true positives TP) to all received results (B); Recall is defined as the fraction of relevant and received results to all relevant results (A); and F-Measure is defined as the harmonic mean of Precision and Recall. Formally, these values are calculated as follows:

$$P = \frac{|TP|}{|B|}, \quad R = \frac{|TP|}{|A|}, \quad F = 2 \cdot \frac{P \cdot R}{P + R}$$

To calculate these measures we used each model as a query model with all 138 models in one collection and expected the search approaches to return the corresponding relevant models from the collection including the query model itself, i. e., we did not delete the query model from the collection. As the four model sets only contain different processes, the relevance of a model with respect to a query is determined by the underlying modeled process. E. g., in case a model from the UA collection was used as query only the nine UA models were relevant results, while returned models from the DM, BR, or CM collections were considered false positives.

Hence, in case of the *Dutch municipalities* model set, ten corresponding models should have been returned for each model used as query. In case a query model was from the *University Admission* or *Birth Registration* sets nine corresponding models should have been returned. And in case of a query from the CM model collection, again, ten corresponding models should have been returned. This means that the amount of relevant results $|A|$ for a specific query model is determined by the number of models per process, which is shown in the third row in Tab. 1.

Besides these quality measures for the query results, we secondly measured the run time by calculating the average millisecond values of ten

Table 1: Characteristics of the evaluation model sets

	DM	UA	BR	CM
Number of processes	8	1	1	4
Number of models per process	10	9	9	10
Total number of models	80	9	9	40
Total number of distinct terms[1]	391	149	141	191
∅ number of terms per model	78.08	71.22	61.56	36.25
STD number of terms	49.08	32.46	17.78	10.94
Min. number of terms	19	38	37	21
Max. number of terms	236	134	87	68

[1] Terms sometimes occur in more than one model. Such terms are only counted once for the total number of distinct terms.

runs. This means in case of the LS3-QueryAll approach that we measured the time for calculating all results ten times including the generation of the LSSM matrix each time. In case of the single query LS3 approach (LS3-Query) we measured the time for generating a pseudo document, calculating similarity values, and determining query results.

Thirdly, we calculated R-Precision and Precision-at-k values to evaluate the ranked retrieval results. R-Precision measures Precision with respect to the first $|A|$ models, i. e., with respect to the amount of relevant results $|A|$ for a query. Considering again a query from the UA model set, for instance, Precision would be determined for the nine models with the highest similarity values to the query. Precision-at-k proceeds in the same direction, but considers k models instead of $|A|$. As the maximum value of $|A|$ is ten, i. e. the maximum number of models per process, we decided to use $k = 5$ to provide a value which is not too close to ten but also not too small. The provided ranked result lists for determining R-Precision and Precision-at-5 values were calculated by setting $\theta = 0.0$ and ordering the results by decreasing similarity values. For further details on all the used measures have a look at Manning et al. (2008).

Besides calculating these metrics for the LS3 approaches, we compared the results with five other approaches for similarity-based search for process models.[7] These comprise of van Dongen et al. (2008), Dijkman et al. (2009), Akkiraju and Ivan (2010), La Rosa et al. (2010), and Yan et al. (2012). van Dongen et al. (2008) is based on causal footprints describing the precedence relations between process model activities. The underlying matching is based on equally labeled nodes in our comparison and parameters were set as described in the publication. Dijkman et al. (2009) use a graph edit distance similarity measure for searching model collections. In our comparison the concept of edit distances is applied to both, node labels (string edit distance) and the graph structure (graph edit distance). We used the Greedy algorithm for the optimization of the similarity matrix and the three quotients mentioned in Dijkman et al. (2009) are equally weighted. In Akkiraju and Ivan (2010), the similarity is calculated based on the number of identically labeled activities. La Rosa et al. (2010) extend Dijkman et al. (2009) by also considering control flow connectors and by calculating a node matching not only based on the Levenshtein similarity (Levenshtein 1966) but also using a linguistic similarity measure. The original implementation has been used with standard parameters. Yan et al. (2012) calculate the Levenshtein similarity for the labels of each node

[7] Regarding a comparison of the LS3 with a naive syntax-based Information Retrieval approach as has, e. g., been used in (Dijkman et al. 2011) see (Schoknecht et al. 2017a).

pair and also consider graph-structural aspects by defining five roles characterizing a node. The thresholds are set as proposed in the original publication, while the resulting similarity matrix is optimized using the Greedy algorithm. Hence, all compared approaches use an underlying node matching for calculating similarity values and query results.

The calculations for these search approaches were conducted using the RefMod-Miner Service[8] to determine similarity values. Afterwards, we determined query results for each model by applying a threshold value on the similarity values. For each approach we used the threshold value which maximized the corresponding F-Measure value. Regarding the LS3 approaches we determined the best values for dimensionality $k = 25$ and a threshold value of 0.79. The thresholds of the other techniques are listed in Tab. 2.

4.2 Results

The results of the evaluation are summarized in Tab. 2 and Tab. 3. Tab. 2 shows the average values for Precision, Recall and F-Measure for all 138 queries. The fifth column shows the number of perfect queries, i. e., queries for which Precision as well as Recall values are 1. The next column shows the average run time of the ten runs conducted, while the last column shows the threshold value θ for which the highest F-Measure value could be achieved. The results for the R-Precision and Precision-at-5 evaluations are presented in Tab. 3.

Regarding Precision, Recall and F-Measure values, the LS3 query approaches generally have high to very high values. Precision values are very high with 0.97 and 0.98 respectively, while Recall values are lower (0.88 and 0.81) but still high with respect to the compared approaches. Yet, the resulting F-Measure is still very high for both the LS3-QueryAll and the LS3-Query approaches with values of 0.93 and 0.89.

Three of the compared techniques nearly achieve the same F-Measure values: SSBOCAN scored 0.78, GEDS 0.76 and LAROSA 0.75. FBSE follows with an F-Measure value of 0.32 due to a very low Precision of 0.23. Finally, no values could be calculated for CF due to a memory overflow for the evaluated model collection.

[8] http://rmm.dfki.de

With respect to the amount of queries for which the search approaches achieved an F-Measure value of 1, again, the LS3 approaches performed best. The LS3-QueryAll approach scored an F-Measure of 1 for 77 of the 138 queries followed by the LS3-Query approach with 69. SSBOCAN was ranked third with 51, GEDS fourth with 44 and LAROSA fifth with 6 queries. The FBSE approach did never achieve an F-Measure of 1 for a query.

When looking at the R-Precision and Precision-at-5 values in Tab. 3 the difference between the LS3-QueryAll approach and the SSBOCAN, LAROSA and GEDS techniques is not that high compared to the differences in F-Measure values. The LS3-QueryAll approach ist still best with very high values of 0.96 for R-Precision and 0.99 for Precision-at-5, but SSBOCAN as second best approach is quite close with values of 0.93 and 0.98 respectively. LAROSA and GEDS are very close once more with values of 0.81 and 0.78 for R-Precision and 0.93 and 0.91 for Precision-at-5. The FBSE approach however performed quite poorly with values of 0.38 and 0.53.

Regarding the final evaluation aspect—the run time to calculate query results—the biggest difference could be observed. While the LS3-Query all approach needed only 1551 milliseconds on average of the ten runs to calculate all query results, LAROSA as the next fastest approach already required about 105 minutes. For the remaining compared approaches the average execution time was even higher. As to the execution time of a single query, in the LS3-Query case it took only 4.5 milliseconds on average to calculate the result.

4.3 Discussion

The evaluation showed four aspects which will be discussed in this section: (1) The LS3 as well as the matching-based approaches achieve good or very good results with respect to the ranking-based measures R-Precision and Precision-at-5; (2) LS3

Table 2: Precision, Recall, F-Measure and run time values for the compared search approaches

	P	**R**	**F**	**F = 1**	Run time	θ
LS3-QueryAll	0.97	**0.88**	**0.93**	**77**	1551 Millisec.	0.79
LS3-Query	**0.98**[2]	0.81	0.89	69	4.5 Millisec.	0.79
CF (van Dongen et al. 2008)[1]	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
GEDS (Dijkman et al. 2009)	0.94	0.63	0.76	44	298:33 Min.	0.48
SSBOCAN (Akkiraju and Ivan 2010)	0.94	0.67	0.78	51	295:37 Min.	0.58
LAROSA (La Rosa et al. 2010)	0.82	0.69	0.75	6	104:50 Min.	0.18
FBSE (Yan et al. 2012)	0.23	0.53	0.32	0	681:52 Min.	0.92

[1] No values could be determined for van Dongen et al. (2008) due to memory overflow.
[2] Bold entries represent best values in each column.

Table 3: R-Precision and Precision-at-5 values. Bold entries represent best values in each column.

	R-Precision	Prec. at 5
LS3-QueryAll	**0.96**	**0.99**
GEDS	0.78	0.91
FBSE	0.38	0.53
SSBOCAN	0.93	0.98
LAROSA	0.81	0.93

search is better compared to the matching-based approaches in terms of Precision, Recall and F-Measure; (3) difficulty of matching is reflected in query results; (4) run time of the LS3 is much faster than for the compared approaches.

Regarding the first aspect, it can be stated that all search approaches except of FBSE return mostly correct results for the first five models with the highest similarity values to a query model (Precision-at-5). And also the R-Precision values are still reasonably high, although the difference between LS3-QueryAll and GEDS is already quite considerable with 0.18. Hence, the resulting ranked list would be suitable for a person searching for similar models to reuse or refactor.

What is also interesting is that the two approaches which do not consider the structure or behavior[9] of a process model—LS3-QueryAll and SSBOCAN—perform considerably better regarding R-Precision than the other approaches. This is somewhat counter intuitive at first sight as LS3 and SSBOCAN actually use less information. Yet, using structural and behavioral information for similarity calculation turns out to be problematic, as correct matches are required, which can not be calculated reliably (see also the third discussion aspect). GEDS, e. g., uses a Graph-Edit distance to calculate the similarity of models thereby using structural information like inserted or substituted nodes and edges. But, for instance, to correctly decide if a node is inserted or substituted requires a correct matching of nodes. As this is not the case the query results are impacted. However, it might be interesting to see how techniques using structural and behavioral information perform against the LS3 when provided with a correct matching, which was out of scope for this evaluation.

With respect to the second aspect the aim of the evaluation is slightly different than with the R-Precision and Precision-at-5 evaluation. By using Precision, Recall and F-Measure we wanted to assess how good the result quality of the search approaches is in terms of returning exactly the relevant models to a query. This could be useful

[9] For an overview on possible dimensions used for similarity calculation and a classification of the compared approaches see Thaler et al. (2017).

for clone detection or in cases multiple models should be merged. In these cases the similarity value of models should typically not be below a certain threshold to reduce the effort of selecting suitable models. As the evaluation shows the LS3 approaches clearly outperform the matching-based search techniques with differences between 0.11 and 0.18 points for the F-Measure values. Thereby, false positives seem to be no real problem for most of the compared approaches as can be seen from the high Precision values. GEDS, SSBOCAN and the LS3 approaches all have values above 0.93 and LAROSA still has 0.82. False negatives however impact the F-Measure values negatively. Except of the LS3 all other techniques miss more than 30 % of the correct results.

The poorer performance of the matching-based approaches is due to poor query results for the UA and BR model sets, which pose the most difficult sets for all approaches. For the 18 queries related to the UA and BR collections 162 relevant models should have been returned by the search techniques. But while the LS3-QueryAll approach performs reasonably well with 76 (47 %) relevant and returned models, the LS3-Query approach only returns 60 models (37 %). For LAROSA, GEDS and SSBOCAN the performance is even worse with 40 (25 %), 38 (24 %) and 32 (20 %) relevant returned models. Considering the DM model collection, which has the easiest matching difficulty, the performance between the LS3 approaches and the compared techniques is even partly reversed. While the LS3-QueryAll approach returns 758 (95 %) of the 800 relevant models, LAROSA, SSBOCAN and GEDS return 638 (80 %), 782 (98 %) and 768 (96 %) relevant models.

The same pattern can be detected for the R-Precision results. While for the UA and BR model sets the LS3-QueryAll approach returns 130 (80 %)[10] of the 162 relevant models, SSBOCAN returns 111 (68 %), GEDS 83 (51 %), and LAROSA 63 (39 %). For the DM model set however, LS3-QueryAll delivers 782 (98 %), SSBOCAN 781 (98 %), GEDS 779 (97 %), and LAROSA 679 (85 %). Hence, the matching quality indeed has an impact on the quality of the search results.

[10] Note that these values differ from the ones above as R-Precision measures Precision of the ranked result list without applying a threshold value.

The performance differences between LS3-QueryAll and LS3-Query are obviously not due to differences in matching quality but base on the calculation of a pseudo document. Including a vector for a query model in the constructed vector space does decrease the query result quality slightly. The difference in terms of F-Measure values between LS3-QueryAll and LS3-Query is 0.04 points in the evaluation.

Finally, another big difference between the LS3 and the compared techniques is the run time for calculating query results. While the LS3 calculates the result of a query in a few milliseconds, the other approaches need multiple seconds or even minutes. LAROSA as the next fastest approach required about 45 seconds on average to calculate the result of one query. The long run time of the matching-based approaches is also a consequence of the calculation of matches. For the determination of matches between two process models each pair of activities has to be compared, which is computationally expensive.

4.4 Limitations

One limitation of the LS3 approach is the difficulty of finding an appropriate number of dimensions k. As can be seen in Figure 5, the results regarding Precision, Recall and F-Measure are quite different depending on the chosen number of dimensions k. For very low values of k Recall is very good, yet Precision values are quite bad, which leads to bad or medium F-Measure values. For higher values of k nearly the opposite can be observed. The Recall values are trending down while the optimal Precision value of 1 is achieved. Hence, some training on test data or experimentation is necessary to find the best k-value for a given model collection in terms of F-Measure.

Another limitation is the omission of structural or behavioral information contained in process

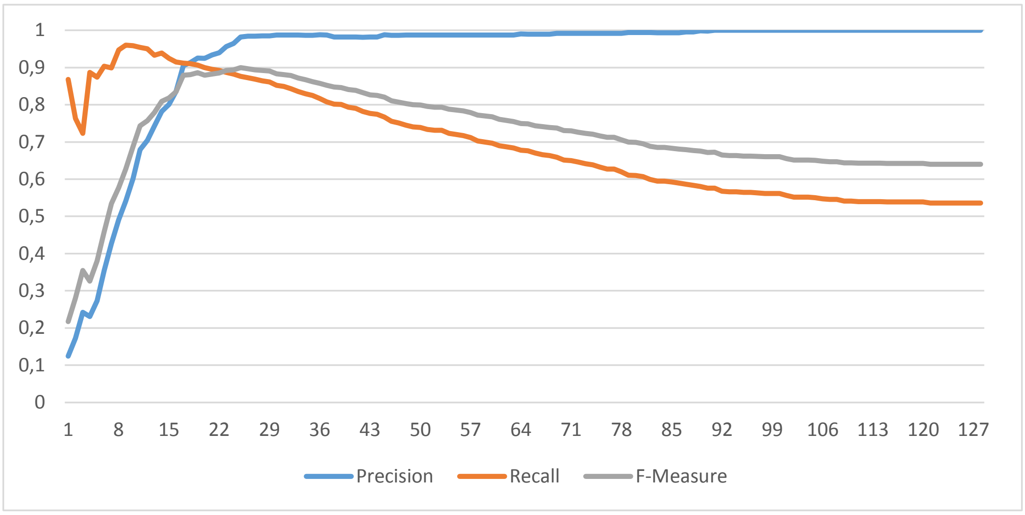


Figure 5: Coomparison of Precision, Recall and F-Measure for different values of k

models in the sense that models with the same activities and labels cannot be distinguished depending on their graph structure or behavior. Hence, if a model collection would contain multiple models only differing in structure or behavior and not in their labels (or more precisely if each of those models would be mapped to the same document vector), then each model pair would receive a similarity value of 1. Yet, in case a distinction between these models is desired, another similarity approach could be executed on these models to distinguish them on the structural or behavioral level. In our opinion, the usefulness of such distinctions depends on the use case and should be evaluated empirically in the future.

Finally, the models used for the evaluation could in general be favorable for the LS3 approaches compared to the matching-based techniques. However, such a distortion is unlikely as model sets with different characteristics have been used (see Section 4.1). And as discussed in the previous subsection, the matching-based techniques performed good for the DM set while they performed poorly for the UA and BR sets. Hence, our results should also apply for other model sets, but this has to be confirmed.

5 Conclusion

Two approaches for similarity-based search, called *Latent Semantic Analysis-based Similarity Search* (LS3) have been described in this article. The first one, the LS3-QueryAll approach, can be used for finding similar models for each process model in a process model collection. The second one, the LS3-Query approach, can be used to find similar models with respect to a specific query model. These search techniques base on the Latent Semantic Analysis concept which is used in the *Latent Semantic Analysis-based Similarity Measure* to determine the similarity of process models. All models having a similarity value equal to or higher than a certain threshold value are judged as results of a query. Besides, algorithms for updating the underlying search structure in cases of model insertions, deletions and changes of the model collection have been presented.

The evaluation of the LS3 approaches against five other similarity-based search techniques showed a superior performance, especially with respect to run time, for a model collection consisting of 138 real-life models. The LS3 approaches achieved the highest Precision, Recall, F-Measure, R-Precision and Precision-at-k values. In terms of run time performance the LS3 approaches clearly outperformed the other techniques. All queries could be processed in about 1.5 seconds with the LS3 approaches which is significantly faster even compared to the next fastest technique which took about 105 minutes to calculate all query results.

Future work could tackle the problem of how to determine, in terms of query performance, the optimal dimensionality reduction for the underlying singular value decomposition. As possible dimensionality numbers depend on the rank of the singular value decomposition the optimal dimensionality might be difficult to detect. In the evaluation with 138 models, for instance, the rank of the singular value decomposition was 128, which means that there exist 128 possible and reasonable values for the dimensionality reduction. Furthermore, the inclusion of additional process documentation into the search procedure as contemplated in Leopold et al. (2016) could be useful. As the underlying Latent Semantic Analysis works especially well for large text corpora such additional textual process information might improve the LS3 querying performance even further. From a practical point of view, we plan to extend the document vector definitions so that process models in other notations can also be included.

Additionally, further comparative experiments with other model sets, which exhibit differing characteristics to the used collections would be useful to, e. g., investigate the run time for larger model collections. One especially interesting comparison between matching-based techniques and the LS3 would be when the matching-based techniques would be provided with a correct matching. Then, one could more precisely assess whether structural or behavioral information contained in process models is indeed helpful for similarity calculation. Until now this is unclear as apart from Thaler et al. (2017) and Dijkman et al. (2011) comparative evaluations have not been published.

Another research direction would be the usage of topic modeling techniques like the LDA (Blei et al. 2003) or word embedding approaches like WORD2VEC (Mikolov et al. 2013) instead of the LSA. Such approaches might be even better suited as they are associated with better results in the computational linguistics field (Baroni et al. 2014).

References

Akkiraju R., Ivan A. (2010) Discovering Business Process Similarities: An Empirical Study with SAP Best Practice Business Processes. In: Maglio P. P., Weske M., Yang J., Fantinato M. (eds.) 8th International Conference on Service Oriented Computing (ICSOC). Lecture Notes in Computer Science Vol. 6470. Springer Berlin Heidelberg, pp. 515–526

Antunes G., Bakhshandeh M., Borbinha J., Cardoso J., Dadashnia S., Francescomarino C. D., 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) The Process Model Matching Contest 2015. In: Kolb J., Leopold H., Mendling J. (eds.) 6th International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA). Lecture Notes in Informatics Vol. P-248. Gesellschaft für Informatik, pp. 127–155

Awad A. (2007) BPMN-Q: A Language to Query Business Processes. In: Reichert M., Strecker S., Turowski K. (eds.) 2nd International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA). Lecture Notes in Informatics Vol. P-119. Gesellschaft für Informatik, pp. 115–128

Awad A., Polyvyanyy A., Weske M. (2008) Semantic Querying of Business Process Models. In: Chi C., Gasevic D., van den Heuvel W. (eds.) 12th International IEEE Enterprise Distributed Object Computing Conference (EDOC). IEEE Computer Society, pp. 85–94

Baroni M., Dinu G., Kruszewski G. (2014) Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In: Fraser A., Liu Y. (eds.) 52nd Annual Meeting of the Association for Computational

Linguistics (ACL). The Association for Computer Linguistics, pp. 238–247

Becker M., Laue R. (2012) A comparative survey of business process similarity measures. In: Computers in Industry 63(2), pp. 148–167

Beeri C., Eyal A., Kamenkovich S., Milo T. (2008) Querying business processes with BP-QL. In: Information Systems 33(6), pp. 477–507

Berry M. W., Dumais S. T., O'Brien G. W. (1995) Using Linear Algebra for Intelligent Information Retrieval. In: SIAM Review 37(4), pp. 573–595

Blei D. M., Ng A. Y., Jordan M. I. (2003) Latent Dirichlet Allocation. In: Journal of Machine Learning Research 3, pp. 993–1022

Business Process Model and Notation (BPMN) (2011). Version 2.0. Object Management Group (OMG)

Cayoglu U., Dijkman R., Dumas M., Fettke P., García-Bañuelos 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. (2014) Report: The Process Model Matching Contest 2013. In: Lohmann N., Song M., Wohed P. (eds.) Business Process Management Workshops. Lecture Notes in Business Information Processing Vol. 171. Springer International Publishing, pp. 442–463

Cover T. M., Thomas J. A. (2006) Elements of Information Theory Vol. 2. Wiley, Hoboken, New Jersey, USA

Deerwester S. C., Dumais S. T., Landauer T. K., Furnas G. W., Harshman R. A. (1990) Indexing by Latent Semantic Analysis. In: Journal of the American Society for Information Science 41(6), pp. 391–407

Delfmann P., Steinhorst M., Dietrich H.-A., Becker J. (2015) The generic model query language {GMQL} – Conceptual specification, implementation, and runtime evaluation. In: Information Systems 47, pp. 129–177

Dijkman R. M., Dumas M., García-Bañuelos L. (2009) Graph Matching Algorithms for Business Process Model Similarity Search. In: Dayal U., Eder J., Koehler J., Reijers H. A. (eds.) 7th International Conference on Business Process Management (BPM). Lecture Notes in Computer Science Vol. 5701. Springer Berlin Heidelberg, pp. 48–63

Dijkman R. M., Dumas M., van Dongen B. F., Käärik R., Mendling J. (2011) Similarity of Business Process Models: Metrics and Evaluation. In: Information Systems 36(2), pp. 498–516

Dumais S. T., Furnas G. W., Landauer T. K., S. Deerwester, R. Harshman (1988) Using Latent Semantic Analysis to Improve Access to Textual Information. In: O'Hare J. J. (ed.) SIGCHI Conference on Human factors in Computing Systems. ACM, pp. 281–285

Dumais S. T. (1991) Improving the retrieval of information from external sources. In: Behavior Research Methods, Instruments, & Computers 23(2), pp. 229–236

Dumais S. T. (2007) LSA and Information Retrieval: Getting Back to Basics. In: Landauer T. K., McNamara D. S., Dennis S., Kintsch W. (eds.) Handbook of Latent Semantic Analysis. Lawrence Erlbaum Associates, pp. 293–321

Dumas M., García-Bañuelos L., Dijkman R. M. (2009) Similarity Search of Business Process Models. In: IEEE Data Engineering Bulletin 32(3), pp. 23–28

Fill H.-G. (2016) Semantic Evaluation of Business Processes Using SeMFIS In: Domain-Specific Conceptual Modeling: Concepts, Methods and Tools Karagiannis D., Mayr H. C., Mylopoulos J. (eds.) Springer International Publishing, pp. 149–170

Gater A., Grigori D., Bouzeghoub M. (2012) Indexing Process Model Flow Dependencies for Similarity Search. In: Meersman R., Panetto H., Dillon T. S., Rinderle-Ma S., Dadam P., Zhou X., Pearson S., Ferscha A., Bergamaschi S., Cruz I. F. (eds.) On the Move to Meaningful Internet Systems: OTM 2012, Part I. Lecture Notes in Computer Science Vol. 7565. Springer Berlin Heidelberg, pp. 128–145

Kastner M., Saleh M. W., Wagner S., Affenzeller M., Jacak W. (2009) Heuristic Methods for Searching and Clustering Hierarchical Workflows. In: Moreno-Díaz R., Pichler F., Quesada-Arencibia A. (eds.) 12th International Conference on Computer Aided Systems Theory (EUROCAST). Lecture Notes in Computer Science Vol. 5717. Springer Berlin Heidelberg, pp. 737–744

Keller G., Nüttgens M., Scheer A.-W. (1992) Semantische Prozeßmodellierung auf der Grundlage "Ereignisgesteuerter Prozeßketten (EPK)". In: Veröffentlichungen des Institut für Wirtschaftsinformatik, Universität Saarbrücken 89

Klein D., Manning C. D. (2003) Accurate Unlexicalized Parsing. In: Hinrichs E. W., Roth D. (eds.) 41st Annual Meeting of the Association for Computational Linguistics. ACL, pp. 423–430

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.) 11th International Conference on Business Process Management (BPM). Lecture Notes in Computer Science Vol. 8094. Springer, Berlin, Heidelberg, pp. 211–218

Kunze M., Weidlich M., Weske M. (2015) Querying process models by behavior inclusion. In: Software and System Modeling 14(3), pp. 1105–1125

La Rosa M., Dumas M., Uba R., Dijkman R. M. (2010) Merging Business Process Models. In: Meersman R., Dillon T., Herrero P. (eds.) On the Move to Meaningful Internet Systems: OTM 2010 - Confederated International Conferences: CoopIS, IS, DOA and ODBASE, Part I. Lecture Notes in Computer Science Vol. 6426. Springer Berlin Heidelberg, pp. 96–113

Landauer T. K., Foltz P. W., D. Laham (1998) An introduction to Latent Semantic Analysis. In: Discourse Processes 25(2-3), pp. 259–284

Landauer T. K. (2007) LSA as a Theory of Meaning. In: Landauer T. K., McNamara D. S., Dennis S., Kintsch W. (eds.) Handbook of Latent Semantic Analysis. Lawrence Erlbaum Associates, pp. 3–34

Lau C. K., Fournier A. J., Xia Y., Recker J., Bernhard E. (2011) Process Model Repository Governance at Suncorp.. Business Process Management Research Group, Queensland University of Technology

Leopold H., van der Aa H., Pittke F., Raffel M., Mendling J., Reijers H. A. (2016) Integrating Textual and Model-Based Process Descriptions for Comprehensive Process Search. In: Schmidt R., Guédria W., Bider I., Guerreiro S. (eds.) 17th International Conference on Enterprise, Business-Process and Information Systems Modeling (BPMDS 2016). Lecture Notes in Business Information Processing Vol. 248. Springer International Publishing, pp. 51–65

Levenshtein V. I. (1966) Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet Physics Doklady 10(9), pp. 707–710

Li S., Cao J. (2015) A New Similarity Search Approach on Process Models. In: Cao J., Wen L., Liu X. (eds.) 1st International Workshop on Process-Aware Systems (PAS). Communications in Computer and Information Science Vol. 495. Springer Berlin Heidelberg, pp. 11–20

Lohmann N., Verbeek E., Dijkman R. (2009) Petri Net Transformations for Business Processes – A Survey. In: Jensen K., van der Aalst W. M. P. (eds.) Transactions on Petri Nets and Other Models of Concurrency II: Special Issue on Concurrency in Process-Aware Information Systems. Lecture Notes in Computer Science Vol. 5460. Springer Berlin Heidelberg, pp. 46–63

Manning C. D., Raghavan P., Schütze H. (2008) Introduction to information retrieval. Cambridge University Press, Cambridge, England

Martin D. I., Berry M. W. (2007) Mathematical Foundations Behind Latent Semantic Analysis. In: Landauer T. K., McNamara D. S., Dennis S., Kintsch W. (eds.) Handbook of Latent Semantic Analysis. Lawrence Erlbaum Associates, pp. 35–56

Mikolov T., Sutskever I., Chen K., Corrado G. S., Dean J. (2013) Distributed Representations of Words and Phrases and their Compositionality. In: Burges C. J. C., Bottou L., Ghahramani Z., Weinberger K. Q. (eds.) Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013, pp. 3111–3119

Miller G. A. (1995) WordNet: a lexical database for English. In: Communications of the ACM 38(11), pp. 39–41

Porter M. F. (1980) An algorithm for suffix stripping. In: Program 14(3), pp. 130–137

Qiao M., Akkiraju R., Rembert A. J. (2011) Towards Efficient Business Process Clustering and Retrieval: Combining Language Modeling and Structure Matching. In: Rinderle-Ma S., Toumani F., Wolf K. (eds.) 9th International Conference on Business Process Management (BPM). Lecture Notes in Computer Science Vol. 6896. Springer Berlin Heidelberg, pp. 199–214

Řehůřek R. (2011) Subspace Tracking for Latent Semantic Analysis. In: Clough P., Foley C., Gurrin C., Jones G. J. F., Kraaij W., Lee H., Mudoch V. (eds.) Advances in Information Retrieval: 33rd European Conference on IR Research (ECIR). Lecture Notes in Computer Science Vol. 6611. Springer, Berlin, Heidelberg, pp. 289–300

Reisig W. (1985) Petri Nets – An Introduction Vol. 4. Springer, Berlin Heidelberg

van Rijsbergen C. J. (1979) Information Retrieval, 2nd ed. Butterworth-Heinemann, Newton, MA, USA

Schoknecht A., Fischer N., Oberweis A. (2017a) Process Model Search Using Latent Semantic Analysis. In: Dumas M., Fantinato M. (eds.) Business Process Management Workshops: BPM 2016 International Workshops, Revised Papers. Springer, Cham, pp. 283–295

Schoknecht A., Thaler T., Fettke P., Oberweis A., Laue R. (2017b) Similarity of Business Process Models — A State-of-the-Art Analysis. In: ACM Computing Surveys 50(4), 52:1–52:33

Shannon C. E. (1948) A mathematical theory of communication. In: The Bell System Technical Journal 27(3), pp. 379–423

Song L., Wang J., Wen L., Wang W., Tan S., Kong H. (2011) Querying Process Models Based on the Temporal Relations between Tasks. In: 15th IEEE International Enterprise Distributed Object Computing Conference Workshops. IEEE Computer Society, pp. 213–222

ter Hofstede A. H. M., Ouyang C., Rosa M. L., Song L., Wang J., Polyvyanyy A. (2013) APQL: A Process-Model Query Language. In: Song M., Wynn M. T., Liu J. (eds.) 1st Asia Pacific Business Process Management Conference (AP-BPM). Lecture Notes in Business Information Processing Vol. 159. Springer Berlin Heidelberg, pp. 23–38

Thaler T., Schoknecht A., Fettke P., Oberweis A., Laue R. (2017) A Comparative Analysis of Business Process Model Similarity Measures. In: Dumas M., Fantinato M. (eds.) Business Process Management Workshops: BPM 2016 International Workshops, Revised Papers. Springer, Cham, pp. 310–322

Turney P. D., Pantel P. (2010) From Frequency to Meaning: Vector Space Models of Semantics. In: Journal of Artificial Intelligence Research 37, pp. 141–188

van Dongen B. F., Dijkman R. M., Mendling J. (2008) Measuring Similarity between Business Process Models. In: Bellahsene Z., Léonard M. (eds.) 20th International Conference on Advanced

Information Systems Engineering (CAiSE). Lecture Notes in Computer Science Vol. 5074. Springer Berlin Heidelberg, pp. 450–464

Vogelaar J., Verbeek H., Luka B., van der Aalst W. M. (2011) Comparing Business Processes to Determine the Feasibility of Configurable Models: A Case Study. In: Daniel F., Barkaoui K., Dustdar S. (eds.) Business Process Management Workshops. Lecture Notes in Business Information Processing Vol. 100. Springer Berlin Heidelberg, pp. 50–61

Weber M., Kindler E. (2003) The Petri Net Markup Language. In: Ehrig H., Reisig W., Rozenberg G., Weber H. (eds.) Petri Net Technology for Communication-Based Systems - Advances in Petri Nets. Lecture Notes in Computer Science Vol. 2472. Springer Berlin Heidelberg, pp. 124–144

Yan Z., Dijkman R., Grefen P. (2012) Fast business process similarity search. In: Distributed and Parallel Databases 30(2), pp. 105–144

Zaman A. N. K., Brown C. G. (2010) Latent Semantic Indexing and Large Dataset: Study of Term-Weighting Schemes. In: 5th International Conference on Digital Information Management (ICDIM). IEEE Computer Society, pp. 1–4