Vector space models for trace clustering: a comparative study

Mateus Alex dos Santos Luna[1], **André Paulino Lima**[2], **Thaís Rodrigues Neubauer**[1], **Marcelo Fantinato**[1], **Sarajane Marques Peres**[1]

[1]School of Arts, Sciences and Humanities – University of Sao Paulo
03828-000 – Sao Paulo – SP – Brazil

{theew,thais.neubauer,m.fantinato,sarajane}@usp.br

[2]Institute of Mathematical and Computer Sciences – University of Sao Paulo
13566-590 – Sao Paulo – SP – Brazil

andre.p.lima@usp.br

Abstract. *Process mining explores event logs to offer valuable insights to business process managers. Some types of business processes are hard to mine, including unstructured and knowledge-intensive processes. Then, trace clustering is usually applied to event logs aiming to break it into sublogs, making it more amenable to the typical process mining task. However, applying clustering algorithms involves decisions, such as how traces are represented, that can lead to better results. In this paper, we compare four vector space models for trace clustering, using them with an agglomerative clustering algorithm in synthetic and real-world event logs. Our analyses suggest the embeddings-based vector space model can properly handle trace clustering in unstructured processes.*

1. Introduction

Process Mining currently plays a key role for business process managers in many types of organization and in different management contexts. Specifically, when descriptive analysis of processes is needed, clustering analysis strategies have been explored in the form of trace clustering [Song et al. 2008, de Leoni et al. 2015, Appice and Malerba 2016, Maita et al. 2017, Lu 2018, De Koninck et al. 2021]. The profiles discovered by trace clustering provide knowledge about the particularities of the business process and provide an organization with information that facilitates the subsequent tasks in the typical process mining workflow. In this usage, trace clustering is seen as a preprocessing step.

The implementation of trace clustering involves making choices about algorithms, similarity functions and computational representation for the objects (traces) to be clustered. Such choices can lead to better or worse results and the more grounded in knowledge, the greater the chance of leading to successful results. However, the specialized literature does not deliver solutions that satisfactorily supports these choices. In this paper, we present a comparative study on four vector space models used to represent data for trace clustering. We use such models with an agglomerative clustering algorithm in synthetic and real-world event logs, aiming to bring to light knowledge about how trace clustering behaves in face of the information represented by each model. The contribution of our study is mainly characterized by providing a descriptive analysis of the effect obtained in trace clustering when a representation obtained through a learning model is used.

The rest of this paper is structured as follows. Section 2 presents basic concepts of process mining, trace clustering and vector space models, and brief descriptions of related work. Section 3 details the method applied to compare vector space models in the trace clustering context. Section 4 reports the results of our experiment and empirical comparative analysis of them. Finally, Section 5 concludes the paper.

2. Theoretical background and related work

In this section, we present the theoretical background and related work.

2.1. Process mining

A *process* can be described as a set of activities, how they should be executed and in which order. A *business process* consists of a set of activities coordinately executed in an organizational and technical environment to achieve a business goal [Weske 2007]. A *trace* is a sequence of activities related to a process. Executions of a trace originate a *case*. Each activity execution in a case is an *event*, which may be described by the executed activity, timestamp of the execution and any other relevant information related to the process. A collection of cases composes an *event log*, in which the execution of a process observed during a certain period is recorded. A *process model* is a representation of a process, presented in a formal, graphical, or structured text notation.

Even though process models are essential tools to achieve success in business management, organizations commonly do not have these models. When the model is available, it is rarely in conformity with the real-life process. According to Aalst [van der Aalst 2011], process mining aims to extract knowledge from event logs that are generated during the execution of the business process, to provide a better understanding of this process. The data regarding the events and the process models are confronted to (a) discover process models, (b) replay and analyze data from the process models or related to them (*conformance*), and (c) find information that enables process improvements (*enhancement*) [van der Aalst 2016]. Such process mining tasks are modeled with respect to one or more perspectives: control-flow, resources, performance, or data [van der Aalst 2016]. In this paper, we are interested in the control flow perspective, which focus on the ordering of activities and characterizes the traces (and cases) of the process under analysis.

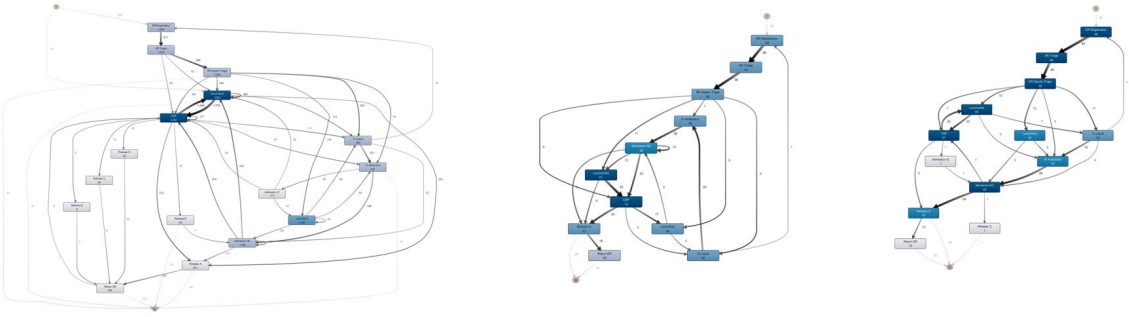
2.2. Trace clustering

The clustering task is known in process mining as trace clustering. Trace clustering is especially useful for decomposing the event logs into sublogs where process mining tasks can be more easily solved [de Leoni et al. 2015]. Figure 1[1] shows examples of process models[2] associated with sublogs built using trace clustering.

Trace clustering strategies that have been used in process mining can be divided into three non-excluding categories [Lu 2018]: *trace sequence similarity*, which is based on the trace syntax and order of activities, as a sequence; *model similarity*, which is based not directly on trace, but on the quality of the process model discovered from the traces;

[1]The figures presented in this paper can be analyzed in detail in `https://github.com/pm-usp/ENIAC-2021-results`.

[2]All process models presented in this paper were discovered using Disco: `https://fluxicon.com/disco/`.



Process model extracted from the complete event log

Process models extracted from two sublogs related to two clusters

Figura 1. Process models discovered in a complete event log and in sublogs

feature vector similarity, in which traces are mapped to a vector space by extracting features from specific profile (such as activity, transition, performance or resource profile [Appice and Malerba 2016]). In this paper, we are interested in the latter strategy, considering activity profile, and taking into account some aspects inherent to the first strategy.

2.3. Vector space models

Representation is an essential component of any problem-solving strategy. Given a problem X and representational schemes R_1 and R_2, it may be the case that X can be solved more efficiently and effectively if R_1 is employed instead of R_2. In this usage, the term representational scheme denotes: (a) how an entity from the problem domain D is mapped to an object in the abstract space Δ; (b) which characteristics of entities from D are relevant to solving the problem at hand, and how these characteristics are mapped to features of objects in Δ; (c) which operations can be applied to objects in Δ and how faithfully each operation reflects relevant processes involving the corresponding entities from D and (d) how an object in Δ is mapped back to an entity from D [Krantz et al. 1971]. When the abstract space Δ is some topological space (e.g., $\mathbb{R}^n$), the objects are usually referred to as *embeddings*. For example, physical documents (entities) can be represented as vectors (objects) embedded in a semantic vector space [Salton et al. 1975].

In process mining, traces related to a process can be individually represented as embeddings according to some representational scheme. Schemes commonly used in trace clustering are based on vectors in which each dimension denotes: (a) the presence or absence of an activity in the trace (BIN), named here as **binary vector space model**; (b) frequency with which a given activity is observed in the trace (AF); (c) a relation between the frequency of an activity in a trace and the count of traces in the event log in which that activity appears (AF-ILF)[3]. The schemes (b) and (c) are known as **count-based vector space models** [Appice and Malerba 2016]. In recent years, representations built through learning models have received the attention of process mining researchers [Koninck et al. 2018, Peeperkorn et al. 2020, Tavares and Barbon 2020]. Learning models are trained on the event log and encode entities (e.g., an activity or a trace) in a vector space such that entities involved in similar contexts are expected to be positioned closer in the vector space. In this paper, we are interested in learning vector representations for the activities and then building a representation vector for the trace by concatenating

[3]Adapting the representational scheme known as TF-IDF [Baeza-Yates and Ribeiro-Neto 1999].

the vectors corresponding to its activities following the order in which they appear in the trace. To differentiate this representational scheme from that traditionally used in trace clustering, we will refer to it as **embedding-based vector space model** (EMB).

2.4. Related work

The work of Koninck *et al.* [Koninck et al. 2018] was the precursor of the use of learning models to generate embeddings for use in process mining. Authors adapted the Word2Vec and Doc2Vec implementations to what they called Act2Vec and Trace2Vec, respectively. Act2Vec treats activities as if they were words in the Word2Vec implementation, and Trace2Vec maps traces to vectors as if they were documents in the Doc2Vec implementation. They introduce use cases and experiments for such representation schemes as trace clustering and process monitoring. The results showed Trac2Vec generates clusters with quality comparable to that obtained with traditional representation methods. A recent approach adds context attributes to embeddings, creating a representation called Case2Vec [Luettgen et al. 2021].

Solutions for other problems in process mining using embedding-based representations for traces were studied by Peeperkorn *et al.* [Peeperkorn et al. 2020] and Tavares and Barbon [Tavares and Barbon 2020]. The former applied such a representation scheme for conformance checking. The authors created synthetic event logs using ground truth models, applied diverse types of noise on them, and then, trained the learning models to build representations for traces. The conformance checking was implemented through the comparison of traces in an event log with a dissimilarity matrix built with encoded traces. The results were compared with traditional conformance checking methods based on alignment techniques. The comparison showed the use of embedding-based representation for the conformance checking is very promising.

Tavares and Barbon [Tavares and Barbon 2020] applied embedding-based representation for traces to be submitted to anomaly detection procedures. Activities were encoded by applying the Word2Vec implementation, and traces were encoded considering a mean of all vectors of activities that composed them. The experiment consisted of classifying traces as anomalous or not anomalous, using a Random Forest classifier.

3. Method

The comparative analysis method is depicted in Figure 2. First, we mapped the synthetic and real-world event logs to vector space models using representational schemes cf. Section 2.3: BIN, AF, AF-ILF and EMB. Second, we carried out clustering on data in each vector space model and extracted quantitative information about clusters and the traces allocated in each cluster[4]. Then, we split the original event log according to clusters, creating sublogs that represent profiles of the associated business process. Finally, we generated statistics about the profiles and discovered a process model for each.

To assess the quality of the partitioning obtained from applying any clustering algorithm to a dataset, we usually resort to intrinsic measures, such as the silhouette index

[4]The analysis in process mining tasks must be concerned with the frequency with which a trace appears in the event log, i.e., the number of cases that follow that trace contains important information for the management of a business process. This characteristic leads to the repetition of vectors in the dataset that is presented to an algorithm, which can influence the obtained answers. In the experiments performed in this paper, we chose to keep these repetitions in the dataset.

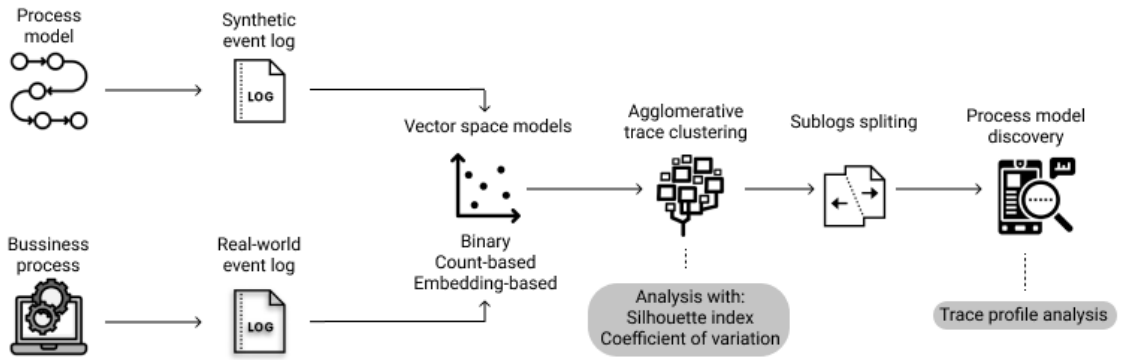


Figura 2. Method applied for comparative analysis of vector space models used to represent data in clustered clustering

and the coefficient of variation, when there is no ground-truth to inform supervised approaches. We applied these measures to evaluate the resulting clusters from a data mining standpoint. The Silhouette index (SI) is computed over the partitioning produced by the clustering algorithm according to the following rule [Han et al. 2012]:

$$SI(D,C) = \frac{1}{|D|} \sum_{o \in D} \frac{b(o,C) - a(o,C)}{max\{b(o,C),\, a(o,C)\}},$$

where D is a non-empty dataset, $C = \{C_1, \ldots, C_k\}$ is the partitioning[5] of D, $a(o,C)$ is the average (Euclidean) distance between o and all other objects in the cluster to which o belongs, and $b(o,C)$ is the minimum average distance between o to all objects in the clusters to which o does not belong. As for the coefficient of variation (CV), this is computed as follows:

$$CV(C) = \frac{\sigma(C)}{\mu(C)} = \frac{1}{\bar{C}} \sqrt{\frac{1}{k} \sum_{i=1}^{k} \left(|C_i| - \bar{C}\right)^2},$$

where $|C_i|$ is the number of objects in the cluster C_i, and $\bar{C}$ is the average number of objects per cluster. In this use, the coefficient of variation is a measure of the imbalance of the partitioning C. For example, let $C = \{C_1, C_2\}$, with $|C_1| = 90$ and $|C_2| = 110$. Then, CV = $\frac{1}{100}\left(\frac{1}{2}\left(10^2 + 10^2\right)\right)^{\frac{1}{2}} = 0.1$ (or 10%). This is to say that the observed variation $\sigma(C) = 10$ corresponds to 10% of the average number of objects per cluster, $\bar{C} = 100$.

The embeddings for activities were built using the Word2Vec implementation of the Gensim Python library[6]. The trace clustering was implemented with an agglomerative hierarchical clustering algorithm available on Scikit-learn Python library[7]. The statistics and process models were obtained with Disco tool. The event logs and the experiments setup are described in sections 3.1 and 3.2, respectively.

3.1. Event logs

Two types of event logs were used in the experiments described in this paper: a synthetic event log, generated using a ground truth model and about which we are aware of

[5]It is assumed that for every partition $C_i \in C$, $|C_i| > 1,\ \forall i \in 1 \ldots k$.

[6]`https://radimrehurek.com/gensim/`

[7]`https://scikit-learn.org/stable/`

the control flow followed by the associated synthetic business process; and a real-world event log, generated from a real business process execution for which we have semantic information but do not know, a priori, the associated control flow.

Synthetic event log The synthetic event log was generated based on a benchmark event log proposed by [Ostovar et al. 2020], whose ground truth process model contains 43 activities, three simple XOR-gateways, one simple AND-gateway, two XOR-AND compound gateways, two AND-XOR compound gateways and two loops (see Figure 3 for examples of control flow elements in Business Process Model Notation). The complete process model is presented in [Ostovar et al. 2020]. We generated the event log considering 40 percent chance of loops occurrence for each loop and 50 percent chance in each XOR-split gateway. The resultant event log contains $107,627$ events, $3,000$ cases and $1,886$ traces. The shortest trace contains 31 events. The longest trace contains 57 events, having one occurrence of the first loop and 11 occurrences of the second loop.

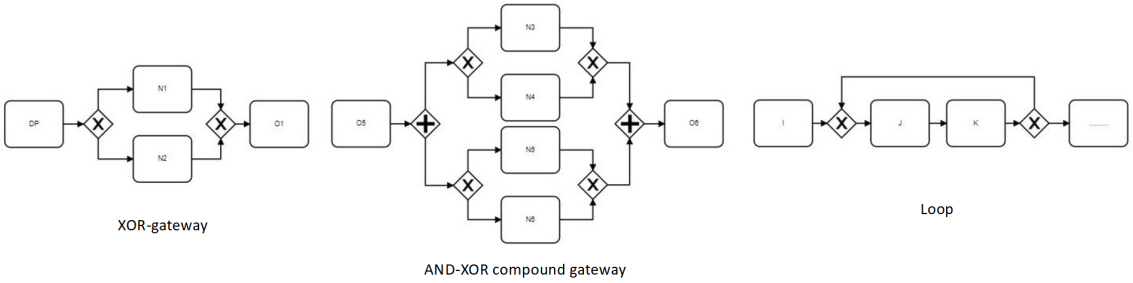


Figura 3. Examples of control flow elements present in the synthetic process model in Business Process Model Notation: $\times$ is the notation for exclusive disjunction (XOR) and $+$ is the notation for conjunction (AND)

Real-world event log The "Sepsis cases" event log described by Mannhardt and Blinde [Mannhardt and Blinde 2017] is the real-world event log. It was generated with event data from a process execution related to treatments of sepsis cases and registered by an enterprise resource planning system (ERP) in a Dutch hospital. Such an event log contains 16 activities, $15,214$ events, $1,050$ cases and 846 traces. A process model discovered for this log, by using Disco, is shown in Figure 4. The process model in the upper left corner of the figure considers all existing traces in the event log. This view shows the process underlying the log is unstructured, possibly noisy, and difficult to analyze – in process mining, a process model like this is known as "spaghetti model". In the lower left corner, an intermediate model is presented, with the 50% more frequent transitions in the event log. On the right side of the figure, the simplest possible process model is presented (the discovery algorithm maintains the most frequent transitions in the event log, with the restriction that all activities in the model are connected) – in process mining, a process model like this is known as "lasagne model". Although the simplest view is easy to analyze, it hides details that could represent strategic information. Analyzing the statistics provided by Disco, we found that the most frequent trace occurs 35 times and is composed by the sequence of three process activities: $<$ *START, ER Registration, ER Triage, ER Sepsis Triage, END* $>$[8]. Such a trace cannot be identified in the simplest process model because the transition of *ER Sepsis Triage* to the *END point* was not represented.

[8] *START* and *END* are artificial activities added to traces to establish a common connection between all traces and the start and end points in the process model.

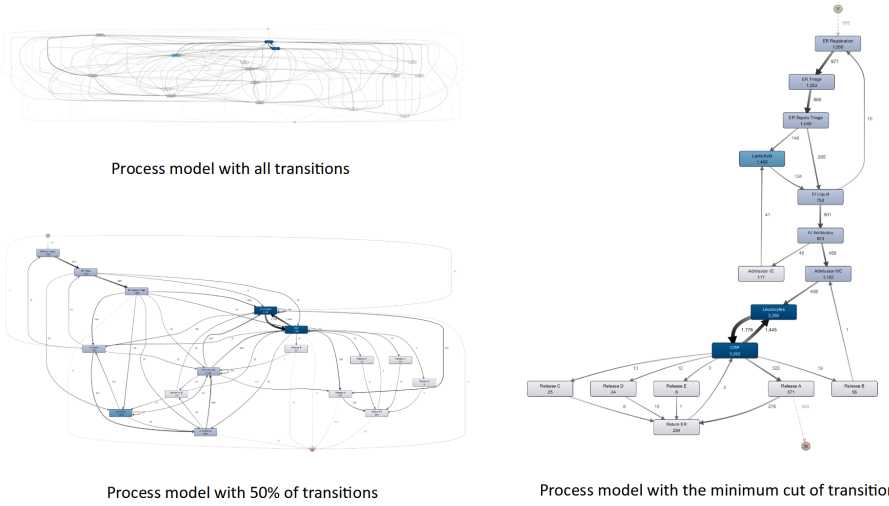


Figura 4. Process models discovered for the "Sepsis cases" event log using Disco tool. In the graphs, activities are represent by boxes and transitions are represent by arrows. The darker the color of the box (arrow), the more frequent is that activity (transition) in the event log.

3.2. Experiments setup

Three main steps were performed in the experiments: (a) mapping event logs to vector space models, generating a vector representation for each trace in the event log and organizing the process cases into a data set; (b) execution of an agglomerative hierarchical clustering algorithm on these data sets, implementing trace clustering and generating clusters that will determine the sublogs; (c) extraction of statistics and discovery of process models on each sublog.

To map event logs to vector space models, each trace in the event log was treated as a bag of activities for BIN, AF and AF-ILF based representational schemes, and a sequence of activities for embedding based representational scheme. Table 1 lists the parameters used to train the embedding models, considering the Gensim Python library requirements. Such set of parameters were chosen based on the observation of the loss function decay behavior during exploratory experiments. With the trained embedding models, we still need to build a representation for each trace, as the embedding models have generated only a representation for activities. To represent the traces, we have decided to concatenate the activities representation following the sequence of activities established in the respective trace. We ensured that the representation for all traces had the same size using a zero-padding strategy. Finally, the cases are mapped to the representation of their respective traces to compose the full dataset.

Tabela 1. Embedding models training parameters

Parameters	Values	Parameters	Values
Algorithm	Skip-gram	Activation function	Negative sampling
Number of iterations	50,000	Embedding vector size	[2, M/4, M/2, M]*
Window size	[1, 2, 3]	* M is the average size of the traces in the event log	

The agglomerative hierarchical clustering was carried out with the same parameters for all representation schemes: Euclidean distance as similarity function, Ward

linkage as the approach to guide the clusters-merging process, number of clusters (k) varying from 2 to 10. Once the clusters were obtained, we split the original event log into k sublogs, each containing the cases corresponding to the vectors allocated in each cluster. Thus, for each set of parameters, we built from 2 to 10 sublogs.

The last step refers to the statistics extraction and discovery of process models to characterize the trace profile present in a sublog. For this purpose, each event sublog was provided as input to analysis by the Disco tool. The parameters chosen in this step refers to percentage of activities (100%, to analyze the complete context of the business process) and percentage of transitions (0% - simplest and minimal process model, to enable a visual analysis of the process profile associated with each cluster) considered by Disco. The statistical information given by Disco always considers the analysis of the entire event log provided as input, regardless of the percentages chosen for viewing the model. This entire step is not mandatory in the context of process mining preprocessing, but it is valuable for the comparison purposes aimed at in this paper.

4. Results and analysis

In this section, the comparative analysis is presented based on the measures SI and CV, and the statistics and process models provided by the Disco tool.

Synthetic event log This is a structured process whose control flow logic is known and can be used to analyze the results obtained from clustering. Table 2 presents the SI and CV for each clustering and each vector space model (VSM). The highest SI obtained for each VSM and the most balanced clustering for each value of k are highlighted in bold[9]. A comparative analysis allows us to state that:

- The **BIN VSM** was more suitable for clustering with small values for k and allowed finding clusters with almost fully balanced distribution of traces among clusters. In the case with strongly balanced clusters and high SI, traces were clustered according to two XOR-gateways, pointing trace profiles oriented to concurrent paths within the process.
- the **AF** and **AF-ILF VSM** achieved the better results for SI with $k = [2, 5]$, but generating unbalanced clusters. Considering as an example the clusters obtained with $k = 3$, the difference among the clusters is the frequency (higher or lower) of the two loops existing in the process, in the traces allocated to each cluster.
- The **embeddings-based VSM** produced balanced clusters for almost all clusterings with $k > 2$. Considering *embedding vector size* $= 2$, the higher the value of k, the higher the values for SI and the more balanced the obtained clusters. Considering *embedding vector sizes* > 2, the best SI results were obtained with $k = \{4, 5\}$. The information present in this representation caused the clustering to be influenced by the size of the traces, distributing traces through clusters as possible according to the number of events present in it. Process structure information (gateways) did not influence clustering. The prediction window size did not influence the results for the most clusterings generated in this experiment.

[9]Balancing is not necessarily required to indicate good solutions. Nevertheless, mainly in real-world processes, the common existence of anomalous executions (anomalous traces) leads to the generation of exclusive clusters for them and prevent the adequate discovery of profiles mainly for small values of k.

Tabela 2. Results from the synthetic event log. Column *k* shows the number of clusters (or sublogs); the other columns refers to each vector space model and associated parameters. The SI and CV calculated for each clustering are listed in blank and gray rows, respectively.

k	BIN	AF	AF-ILF	EMB 2			EMB M/4			EMB M/2			EMB M		
				1	2	3	1	2	3	1	2	3	1	2	3
2	**0.31**	0.12	0.16	0.33	0.35	0.35	0.18	0.19	0.16	0.18	0.18	0.18	0.18	0.18	0.18
	1%	26%	8%	10%	10%	10%	28%	28%	28%	28%	28%	28%	28%	28%	28%
3	0.25	0.14	**0.18**	0.35	0.36	0.36	0.24	0.25	0.21	0.24	0.24	0.24	0.24	0.24	0.24
	35%	28%	59%	28%	28%	28%	**11%**	**11%**	**11%**	**11%**	**11%**	**11%**	**11%**	**11%**	**11%**
4	0.24	**0.15**	0.17	0.38	0.39	0.38	0.27	0.29	0.23	0.28	0.28	**0.28**	0.28	0.28	0.28
	1%	49%	68%	20%	20%	20%	31%	31%	31%	31%	31%	31%	31%	31%	31%
5	0.22	0.14	0.17	0.39	0.40	0.39	**0.28**	**0.30**	**0.25**	**0.29**	**0.30**	0.21	**0.29**	**0.29**	**0.30**
	32%	52%	73%	45%	46%	48%	54%	54%	54%	54%	54%	**21%**	54%	54%	54%
6	0.22	0.14	0.12	0.42	0.42	0.40	0.22	0.24	0.18	0.22	0.23	0.23	0.23	0.23	0.23
	40%	64%	66%	**35%**	36%	40%	40%	39%	45%	39%	39%	39%	39%	39%	39%
7	0.19	0.12	0.09	0.47	0.42	0.41	0.22	0.25	0.14	0.19	0.20	0.20	0.19	0.20	0.20
	39%	56%	47%	34%	48%	49%	57%	56%	41%	**26%**	**26%**	**26%**	**26%**	**26%**	**26%**
8	0.18	0.10	0.09	0.55	0.46	0.45	0.22	0.19	0.16	0.20	0.21	0.21	0.20	0.21	0.21
	31%	41%	38%	**31%**	42%	43%	58%	47%	32%	43%	43%	43%	43%	43%	43%
9	0.18	0.11	0.09	0.55	0.54	0.53	0.17	0.17	0.16	0.17	0.19	0.19	0.18	0.19	0.19
	32%	50%	**30%**	36%	36%	37%	44%	49%	44%	44%	44%	44%	44%	44%	44%
10	0.18	0.12	0.10	**0.59**	**0.58**	**0.56**	0.18	0.17	0.17	0.18	0.19	0.20	0.18	0.19	0.19
	32%	61%	38%	**24%**	**24%**	26%	38%	44%	38%	56%	56%	56%	56%	56%	56%

Real-world event log This is an unstructured process about which we do not have *a priori* knowledge about the process model that represents it. Thus, there is no association between the analysis performed and any *a priori* knowledge about the process. Table 3 presents the SI and CV for each clustering. The highest SI obtained for each VSM and the most balanced clustering for each value of k are highlighted in bold. A comparative analysis allows us state that:

- The **BIN VSM** was more suitable for clustering with high values for k. A specific analysis of two clusterings revealed that the distribution of traces in clusters is slightly influenced by the traces size (number of events in a trace) and the number of cases associated with each trace. In the clustering with $k = 2$, there is a clear distinction between the clusters regarding the size of the shortest and most frequent traces (traces with at least 4 associated cases): one cluster contains the traces with 3 to 7 events; the other cluster contains the traces with 8 to 12 events. For the clustering with $k = 10$, such influences become more subtle.
- The **AF** and **AF-ILF VSM** was more suitable for clustering with small values for k, however, all clusterings produced unbalanced clusters. High SIs were obtained for clustering with strongly unbalanced clusters. An analysis of the clusterings with $k = 2$ revealed that one cluster generated in the AF VSM has the 5 longest traces; similarly, in the AF-ILF VSM, one cluster was generated with 2 longest traces. In both cases, such traces were associated with only 1 case and may indicate one cluster with outliers characterized by long loops involving three activities.

- The **embeddings-based VSM** was more suitable for clustering with high values for k and small sizes for embedding vectors. The distribution of traces in clusters was strongly guided by the traces size and the number of cases associated with each trace. The analysis of a clustering with $k = 2$ revealed that one of the clusters was composed with long and low frequent traces (with only 1 case associated with each trace); the analysis of a clustering with $k = 10$ reinforced the influence of traces size and number of cases associated with them in the distribution of objects in clusters. In process mining theory, infrequent traces are associated with noise [van der Aalst 2011]. Based on such an assumption, we claim that the information present in this representation allows organizing noise in clusters, separating them from the mainstream behavior in the process under analysis.

Tabela 3. Results from the real-world event log. Column *k* shows the number of clusters (or sublogs); the other columns refers to each vector space model and associated parameters. The SI and CV calculated for each clustering are listed in blank and gray rows, respectively.

k	BIN	AF	AF-ILF	EMB 2			EMB M/4			EMB M/2			EMB M		
				1	2	3	1	2	3	1	2	3	1	2	3
2	0.39	**0.92**	**0.94**	**0.34**	**0.25**	**0.45**	0.16	0.15	0.16	0.08	0.12	0.14	0.12	0.12	0.11
	59%	99%	100%	50%	31%	19%	60%	27%	83%	60%	27%	83%	81%	**1%**	10%
3	0.40	0.70	0.64	0.22	0.20	0.18	0.20	0.16	0.15	0.01	**0.13**	**0.15**	**0.13**	0.12	**0.12**
	56%	123%	117%	**35%**	36%	39%	63%	69%	79%	101%	52%	56%	89%	44%	38%
4	0.44	0.43	0.59	0.23	0.15	0.16	0.23	0.21	0.20	0.04	0.12	0.14	0.11	0.10	0.10
	83%	103%	146%	36%	43%	**26%**	92%	61%	47%	62%	62%	64%	99%	28%	44%
5	0.49	0.43	0.23	0.24	0.18	0.16	0.27	0.25	0.24	0.07	**0.13**	0.14	0.11	0.12	0.11
	46%	125%	146%	41%	50%	**25%**	86%	46%	33%	67%	68%	73%	99%	36%	33%
6	0.54	0.43	0.23	0.22	0.18	0.16	0.32	0.30	0.28	0.08	0.12	0.13	0.12	0.12	**0.12**
	59%	144%	166%	56%	48%	44%	57%	67%	57%	66%	67%	71%	71%	42%	**41%**
7	0.60	0.31	0.20	0.23	0.19	0.19	0.35	0.32	0.30	0.10	**0.13**	0.11	0.07	**0.13**	0.11
	58%	109%	113%	62%	59%	49%	66%	55%	62%	67%	67%	59%	63%	55%	**44%**
8	0.63	0.31	0.20	0.24	0.19	0.19	0.37	0.34	0.32	0.12	0.09	0.11	0.08	0.09	0.11
	72%	122%	125%	**42%**	61%	**42%**	74%	45%	54%	67%	67%	57%	51%	59%	52%
9	0.65	0.31	0.23	0.21	0.19	0.17	0.39	0.36	**0.34**	**0.16**	0.11	0.13	0.09	0.09	0.08
	80%	134%	127%	44%	41%	**40%**	51%	52%	62%	66%	68%	57%	45%	52%	59%
10	**0.67**	0.31	0.23	0.21	0.20	0.18	**0.42**	**0.37**	**0.34**	0.14	0.12	0.13	0.09	0.10	0.09
	86%	145%	38%	55%	42%	49%	63%	55%	60%	63%	67%	51%	**34%**	45%	69%

By analyzing process models obtained from sublogs generated by trace clustering in the embeddings-based VSM, we observed a tendency to get more complex (spaghetti) process models than those obtained from trace clustering in the BIN VSM (see Figure 5 for a general shape analysis). We hypothesize the complexity is due the organization of infrequent traces in clusters. In the case of clusters generated in the BIN VSM, the infrequent traces are clustered together frequent traces. Consequently, the visualization provided by Disco hides the noise of each cluster (the infrequent traces) following its procedure to show the more frequent transitions in the simplest visualization. We conclude that EMB VSM is more useful as a preprocessing step for analysts aware of such characteristic. Clusters with infrequent traces can be used for anomalies analysis. If infrequent

traces were organized in exclusive clusters, the process mainstream behavior is allocated in a (at some level) "filtered" cluster. Thus, the next steps of process mining should be carried out in such a cluster to have better and more objective analysis conditions.

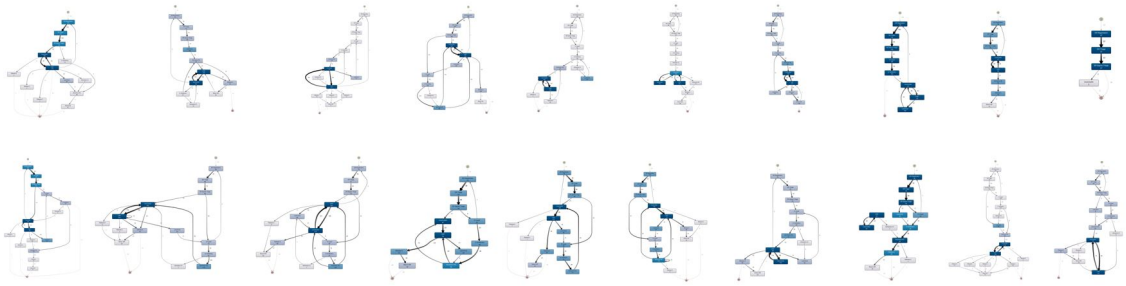


Figura 5. Comparison: process models discovered on clusters obtained in the BIN VSM (top row); process models discovered on clusters obtained in the EMB VSM (bottom row). Clustering with $k = 10$ and visualization of the simplest process models (using Disco). The first process model in the bottom row probably comes from the cluster with the mainstream behaviour.

5. Conclusions

In this paper, we report the results of a comparative study of vector space models for trace clustering. The idea was to explore the traditional vector representations used in trace clustering and the new opportunity brought by the construction of vector representations through learning models. Our experiments showed clusters generated in the binary vector space and in the vector space of embeddings have higher quality in terms of Silhouette index and tend to generate more uniform distributions of traces across the clusters. Analysis considering the process model standpoint allowed us to observe embeddings-based vector space can properly handle infrequent traces, helping to isolate the process mainstream behavior in a specific cluster. The comparative study discussed in this paper is empirical and does not present statistical generalization. However, it has analytic generalization, contributing to expand the correlated theory and showing reasonableness to motivate tests in other contexts (combining event logs, trace clustering algorithms and learning models for embeddings). We argue that more comparative studies like this are needed to clarify what kind of information is embedded in each vector space model and what effects this information has on the clustering results. Knowing about such effects helps process mining analysts make choices that are more appropriate and adherent to their analysis objectives.

6. Acknowledgments

This research was funded by: the National Council for Scientific and Technological (CNPq), Brazil – Program PIBIC; the Coordination of Superior Level Staff Improvement (CAPES), Brazil — Finance Code 001.

Referências

Appice, A. and Malerba, D. (2016). A co-training strategy for multiple view clustering in process mining. *IEEE Trans. Serv. Comput.*, 9(6):832–845.

Baeza-Yates, R. A. and Ribeiro-Neto, B. (1999). *Modern information retrieval*. Addison-Wesley Longman Publishing Co., Boston.

De Koninck, P., Nelissen, K., vanden Broucke, S., Baesens, B., Snoeck, M., and De Weerdt, J. (2021). Expert-driven trace clustering with instance-level constraints. *Knowl. Inf. Syst.*, 63:1197–1220.

de Leoni, M., van der Aalst, W. M. P., and Dees, M. (2015). A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. *Inf. Syst.*, 56:235–257.

Han, J., Pei, J., and Kamber, M. (2012). *Data Mining: Concepts and Techniques*. Morgan Kauffman, Waltham, 3rd edition.

Koninck, P., Broucke, S., and Weerdt, J. (2018). act2vec, trace2vec, log2vec, and model2vec: Representation learning for business processes. In *Bus. Process Manage.*, volume 11080 of *Lect. Notes Comput. Sci.*, pages 305–321, Berlin. Springer.

Krantz, D., Luce, D., Suppes, P., and Tversky, A. (1971). *Foundations of Measurement*, volume 1. Dover, US.

Lu, X. (2018). *Using behavioral context in process mining: exploration, preprocessing and analysis of event data*. PhD dissertation, Eindhoven University of Technology.

Luettgen, S., Seeliger, A., Nolle, T., and Mühlhäuser, M. (2021). Case2vec: Advances in representation learning for business processes. In *Process Mining Workshops*, pages 162–174. Springer.

Maita, A. R. C., Martins, L., Paz, C. R. L., Rafferty, L., Hung, P. C. K., Peres, S. M., and Fantinato, M. (2017). A systematic mapping study of process mining. *Enterprise Inf. Syst.*, 12:1–45.

Mannhardt, F. and Blinde, D. (2017). Analyzing the trajectories of patients with sepsis using process mining. In *Proc. Radar tracks at the 18th Int. Working Conf. on BPMDS*.

Ostovar, A., Leemans, S. J. J., and La Rosa, M. (2020). Robust drift characterization from event streams of business processes. *ACM Trans. Knowl. Discov. Data*, 14(3).

Peeperkorn, J., vanden Broucke, S., and Weerdt, J. (2020). Conformance checking using activity and trace embeddings. In *BPM*, pages 105–121, Berlin. Springer.

Salton, G., Wong, A., and Yang, C. S. (1975). A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620.

Song, M., Gunther, C. W., and van der Aalst, W. M. P. (2008). Trace clustering in process mining. In *Bus. Process Manage. Workshops*, pages 109–120, Berlin. Springer.

Tavares, G. and Barbon, S. (2020). Analysis of language inspired trace representation for anomaly detection. In *ADBIS, TPDL and EDA 2020 Common Workshops and Doctoral Consortium*, volume 1260, pages 296–308. Springer.

van der Aalst, W. M. P. (2011). *Process mining – discovery, conformance and enhancement of business processes*. Springer, Berlin, 1st edition.

van der Aalst, W. M. P. (2016). *Process Mining: Data Science in Action*. Springer, Berlin, 2nd edition.

Weske, M. (2007). *Business Process Management: Concepts, Languages, Architectures*. Springer, Berlin, 2nd edition.