UHASSELT

KNOWLEDGE IN ACTION

Faculteit Bedrijfseconomische Wetenschappen

master handelsingenieur in de beleidsinformatica

Masterthesis

Generating realistic artificial event data

Marino Soro

Scriptie ingediend tot het behalen van de graad van master handelsingenieur in de beleidsinformatica

PROMOTOR :

Prof. dr. Benoit DEPAIRE

UHASSELT

KNOWLEDGE IN ACTION

www.uhasselt.be
Universiteit Hasselt
Campus Hasselt:
Martelarenlaan 42 | 3500 Hasselt
Campus Diepenbeek:
Agoralaan Gebouw D | 3590 Diepenbeek

2019
2020

UHASSELT

KNOWLEDGE IN ACTION

Faculteit Bedrijfseconomische Wetenschappen

master handelsingenieur in de beleidsinformatica

Masterthesis

Generating realistic artificial event data

Marino Soro

Scriptie ingediend tot het behalen van de graad van master handelsingenieur in de beleidsinformatica

PROMOTOR :

Prof. dr. Benoit DEPAIRE

Genereren van realistische artificiële event data *

Marino Soro[1]

Universiteit Hasselt, Agoralaan Gebouw D, 3590 Diepenbeek, België
marino.soro@student.uhasselt.be

Abstract. In het onderzoek naar het vergelijken van technieken die de control-flow (CF) van een proces kunnen ontdekken is een voldoende beschikbaarheid aan evaluatiedata een cruciale bouwsteen van de empirische analyse van CF discovery technieken. Wat betreft de evaluatiedata bestaan er twee verschillende soorten: reële data en artificiële data. Op dit moment bestaan er manieren om artificiële data te genereren om het gebrek aan reële data voor empirische analyse op te vangen. Echter blijft het een uitdaging om te verzekeren dat deze data realistisch is en de werkelijkheid weerspiegelt binnen het onderzoeksdomein. De paper onderzoekt het potentieel van representatiemodellen om karakteristieken van een event log te capteren en extraheren.

Keywords: Artificial event logs, process discovery

1 Introductie

Process mining is het onderzoeksdomein dat zich richt op het extraheren van kennis uit activiteitenlogboeken, ook wel event logs genoemd [14],[4]. Binnen process mining bestaat een veelheid aan algoritmes om de control-flow (CF) direct vanuit een event log te ontdekken (zie voor een overzicht [14],[3]). Terwijl onderzoekers zich in de beginperiode concentreerden op het ontwikkelen van algoritmen om specifieke procesconstructies te ontdekken, is het belang van het onderzoek op de dag van vandaag verschoven naar het vergelijken van dergelijke algoritmen ([13],[15],[16],[17]). Het in [13] geïntroduceerde raamwerk beschrijft een empirische evaluatiemethode om CF-vindingstechnieken te vergelijken.

Aangezien dergelijke evaluaties grote hoeveelheden geschikte gegevens (event logs) nodig hebben als input voor empirische analyse is de vraag waar deze data vandaan gehaald kan worden. In het ideale scenario zou dergelijk onderzoek uitsluitend gebruik maken van data uit de echte wereld omdat onderzoekers er op deze manier zeker van zouden kunnen zijn dat deze data, en bijgevolg de gevonden resultaten, als realistisch beschouwd kunnen worden. Echter is het aantal reële event logs beperkt in beschikbaarheid waardoor onderzoekers genoodzaakt zijn elders op zoek te gaan naar hun data.

* Deze masterproef werd geschreven tijdens de COVID-19 crisis in 2020. Deze wereldwijde gezondheidscrisis heeft mogelijk een impact gehad op het schrijf- en verwerkingsproces, de onderzoekshandelingen en de onderzoeksresultaten die aan de basis liggen van dit werkstuk.

Een mogelijke oplossing die wel eens bepleit wordt om het tekort aan reële event logs op te lossen is het genereren en gebruiken van artificiële event logs. Hiervoor zijn er reeds verschillende technieken ontwikkeld die de onderzoeker in staat stellen zelf te bepalen hoe deze artificiële data eruit moet zien [6],[5],[1]. Vanuit het onderzoek-standpunt is dit een geweldige oplossing: een oneindigheid aan beschikbare data die de onderzoeker volledig zelf in handen heeft. Jammer genoeg is er ook een kanttekening aan dit mooie verhaal: de onderzoeker heeft de data volledig in handen. Inderdaad, hetzelfde argument kan gebruikt worden om het gebruik van artificiële data zowel te bepleiten als af te keuren. Indien de onderzoeker het genereren van de artificiële data volledig in handen heeft is er natuurlijk geen zekerheid meer over de mate van realisme en het weerspiegelen van de werkelijkheid van de data en dus ook de gevonden resultaten van het onderzoek. Logischerwijs zouden we dus op zoek moeten gaan naar een manier om te verzekeren dat de data die de onderzoeker gebruikt niet enkel artificieel is, maar ook realistisch in het veld van het onderzoek. Echter is hier tot op de dag van vandaag nog niet veel aandacht aan besteed binnen het domein van process mining en zal deze paper daarom hier een bijdrage aan leveren. Op welke manier kunnen we ervoor zorgen dat artificiële event logs realistisch zijn binnen een bepaald onderzoeksveld?

Een mogelijke piste om realisme in artificiële event logs te verzekeren is het zorgen dat de artificiële event logs dezelfde karakteristieken hebben als een reële event log uit het onderzoeksveld. In andere domeinen is het gebruik van vector representaties om de relatie tussen objecten in kaart te brengen een techniek die succesvolle resultaten heeft kunnen opleveren [8],[9],[7]. Een vector representatie van een object is een vector die ontstaat door het trainen van een neuraal netwerk en vervolgens de essentie van het object weet te capteren in haar gewichten. In het domein van Natural Language Processing (NLP) zijn onderzoekers erin geslaagd woorden om te vormen tot vector representaties en de gelijkenissen van deze woorden hierin te capteren. Zo zou de vector die ontstaat uit de berekening $vector('Koning') - vector('Man')$ een vector zijn die opmerkelijk dicht ligt tegen de vector: $vector('Koningin')$. Omdat het gebruik van vector representaties in andere domeinen goed blijkt te werken zal deze paper onderzoeken of het gebruik van vector representaties in het domein van process mining al dan niet een goede techniek kan zijn om bepaalde karakteristieken van een event log te capteren, met als doel deze opnieuw te kunnen extraheren en te gebruiken om nieuwe artificiële event logs te genereren. Meer specifiek kan de bijdrage van deze paper worden beschreven door de volgende opsomming:

- De paper analyseert het potentieel van vector representaties in het capteren van essentiële data (karakteristieken) van event logs. (Secties 3, 4).
- De paper analyseert de mogelijkheid om karakteristieken van een event log te extraheren uit een vector representatie (Sectie 5).

Het vervolg van de paper is als volgt opgebouwd. Sectie 2 beschrijft de stappen die we zullen ondernemen tot het bekomen van de resultaten van het onder-

zoek. Hierbij inbegrepen zijn onder meer de methoden van data generatie en preparatie. Sectie 3 gaat meer in detail over het bekomen van vector representaties van event logs en de architectuur van het neuraal netwerk dat we hiervoor zullen gebruiken. Vervolgens geeft Sectie 4 meer uitleg over hoe we zullen onderzoeken of we de onderliggende karakteristieken van een event log hebben weten te capteren door middel van clustering om hierna in Sectie 5 te onderzoeken of we deze karakteristieken opnieuw kunnen extraheren uit de vector representaties van de event logs. Gerelateerd werk word besproken in Sectie 6 en Sectie 7 rond de paper af met enkele conclusies.

2 Methodologie [1]

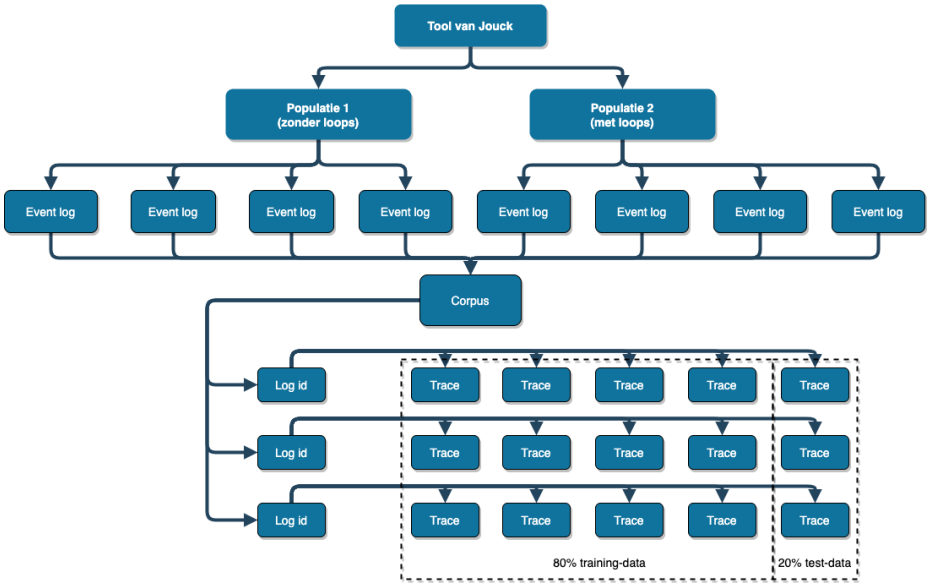

Fig. 1. Werkwijze voor het genereren en voorbereiden van de data voor één dataset

Om het potentieel van vector representaties in het capteren van zinvolle informatie (karakteristieken) van een event log te onderzoeken, alsook de mogelijkheid om deze karakteristieken opnieuw te extraheren uit deze representaties gaan we in drie stappen te werk.

In een eerste fase zullen we verschillende event logs, waarvan we de karakteristieken kennen, omvormen tot hun vector representaties. Zoals eerder vermeld kunnen vector representaties van objecten, in dit geval event logs, geleerd worden

[1] The software is available on GitHub. Codebase: https://github.com/marinosoro/generating-realistic-artificial-event-data.git.

Tabel 1. Configuratie van de parameters

Afkorting	Naam	Vast/Variabel	Waarde/Interval
Sampling parameters			
P1	Aantal procesbomen per populatie	Vast	1
P2	Aantal event logs per procesboom	Variabel	[50, 100]
Log parameters			
P3	Aantal traces per event log	Vast	1000
Neuraal netwerk parameters			
P4	Lengte van de vector representaties	Variabel	{32, 64}
Procesboom parameters			
P5	Minimum no. of activities	Variabel	[5, 10]
P6	Maximum no. of activities	variabel	[10, 15]
P7	No. of activities mode	variabel	[Minimum, Maximum]
P8	Sequence	Vast	0.2
P9	Choice	Vast	0.2
P10	Parallel	Variabel	[0, 0.5]
P11	Or	Vast	0.2
P12	Silent	Variabel	[0, 0.3]
P13	Duplicate	Variabel	{0, 1}
P14	Long term dependency	Variabel	[0, 0.1]
P15	Infrequent	Variabel	[0, 0.5]
P16	Unfold	Vast	1
P17	Loop	Variabel	t.b.c.
P18	Max repeat	Variabel	[2, 10]

door het trainen van een neuraal netwerk. In deze paper zullen we gebruik maken van een variant op de log2vec-architectuur, voorgesteld in [2]. De kwaliteit van deze fase meten we door de accuraatheid van het getrainde neurale netwerk te meten aan de hand van een test-dataset.

Vervolgens zullen we de verkregen vector representaties uit de eerste fase trachten te clusteren. Indien we erin slagen verschillende clusters te herkennen en de event logs binnen deze clusters onderling over dezelfde karakteristieken beschikken kunnen we er van uit gaan dat de vector representaties essentiële informatie omtrent deze karakteristieken hebben weten te capteren.

Ten slotte zullen we de mogelijkheid testen om uit deze vector representaties opnieuw de karakteristieken van de event logs te extraheren. Hiervoor zullen we opnieuw gebruik maken van enkele neurale netwerken. De architectuur van de netwerken die we hiervoor zullen gebruiken wordt uitgebreid beschreven in Sectie 5. Om een concreet doel voor ogen te hebben zullen we in deze paper trachten te voorspellen of een event log al dan niet kan beschikken over loops en zo ja, hoe vaak deze loops zich kunnen herhalen.

Om het hierboven beschreven onderzoek uit te kunnen voeren hebben we uiteraard ook nood aan een bepaalde hoeveelheid data, oftewel event logs. Omdat we in deze paper het potentieel van vector representaties om karakteristieken van een event log te capteren willen analyseren, alsook de mogelijkheid om deze karakteristieken later terug te extraheren uit deze vector representaties is het belangrijk dat de karakteristieken van de event logs die we in dit onderzoek zullen gebruiken controleerbaar zijn. We zullen in dit onderzoek dan ook gebruik maken van artificiële event logs die we zullen genereren met behulp van PTandLogGenerator [5]. Deze maakt gebruik van een set van parameters om de karakteristieken van event logs uit een bepaalde populatie te beschrijven. Omdat we later in dit onderzoek willen onderzoeken of we enerzijds event logs met verschillende karakteristieken (dus uit verschillende populaties) kunnen clusteren door middel van hun vector representaties en anderzijds deze karakteristieken terug kunnen extraheren uit hun vector representaties zullen we als volgt te werk gaan om de artificiële data te genereren en voor te bereiden op het verder onderzoek:

1. We beginnen met het definiëren van twee populaties door gebruik te maken van de parameters uit PTandLogGenerator ([5]). Een overzicht van deze parameters kan worden geraadpleegd onder 'procesboom parameters' in tabel 1. Initieel zullen we beide van deze populaties op dezelfde manier, met dezelfde parameterwaarden, definiëren. Om de potentiële invloed van deze procesboom parameters op de accuraatheid van het model uit te middelen zullen we deze parameters op een willekeurige manier definiëren zoals beschreven in tabel 1. Vervolgens zullen we om een onderscheid te introduceren tussen deze twee populaties de twee parameterwaarden aanpassen die overeenstemmen met de karakteristieken die we later in dit onderzoek zullen trachten te voorspellen. De parameter die overeenstemt met de karakteristiek die bepaalt of

een event log al dan niet kan beschikken over loops is parameter 'loop' (P17). De parameter die bepaalt hoe vaak een loop zich maximaal kan herhalen is parameter 'max repeat' (P18). In één van de populaties zullen we de waarde voor beide parameters gelijkstellen aan 0. In de andere populatie zal de loop parameter variëren in het interval [0,1 en 0,5] en de max repeat parameter in het interval [2, 10].

2. Gebruik makende van PTandLogGenerator ([5]) zullen we vervolgens voor iedere populatie één procesboom genereren. Deze procesboom zal op zijn beurt gebruikt worden om enkele artificiële event logs te genereren. Het aantal event logs dat we genereren wordt willekeurig bepaald en ligt in het interval [50, 100], zoals beschreven in tabel 1. We zullen voor ieder van deze event logs ook informatie opslaan omtrent de populatie waaruit ze tot stand zijn gekomen en de parameterwaarden die gebruikt werden om deze populaties te definiëren.

3. Al deze event logs zullen vervolgens samengevoegd worden in één grote dataset die we in deze paper de corpus zullen noemen. De corpus bevat informatie omtrent de verschillende traces uit iedere event log en bewaart daarbij de log id van de event log waartoe deze individuele traces behoren. Een voorbeeld van hoe een corpus er uit zou kunnen zien wordt beschreven in tabel 2. De corpus die wordt beschreven door deze tabel is ontstaan door het samenvoegen van twee event logs, waarbij de eerste event log twee verschillende traces bevat en de tweede event log drie verschillende traces bevat.

4. De corpus vormt de basis voor het neurale netwerk en zal worden opgesplitst in training- en test-data. Omdat we streven naar een vector representatie voor iedere event log in de corpus moeten we ervoor zorgen dat iedere event log vertegenwoordigd word in zowel de training- als test-data. Om deze reden zullen we de opsplitsing in training- en test-data maken op log niveau, wat betekent dat 80% van alle data voor iedere event log in de corpus in de training-data zal worden geplaatst en de resterende 20% van alle data voor iedere event log in de corpus in de test-data zal worden geplaatst.

De stappen 1 tot en met 4 zullen in dit onderzoek honderd maal herhaald worden. De data die resulteert uit ieder van deze herhalingen zullen later in dit onderzoek als één dataset benoemd worden. Figuur 1 visualiseert de totale werkwijze voor het genereren en voorbereiden van één dataset.

Tabel 2. Voorbeeld data van een corpus bestaande uit twee event logs

Log id	Trace
1	[f, c, d]
1	[f, c, e, k, i]
2	[a, c, d, g, b, e, g, b, e, h, j, k, i]
2	[g, h]
2	[f]

3 Het genereren en omvormen tot vector representaties van artificiële event logs

De techniek van het bekomen van vector representaties van objecten berust op het gebruik van neurale netwerken om deze vector representaties te construeren. Het neuraal netwerk dat we zullen gebruiken om event logs om te vormen tot hun vector representaties is een variant op de log2vec-architectuur, voorgesteld in [2].

De log2vec-architectuur is een architectuur die informatie omtrent enkele voorafgaande activiteiten als input neemt, in combinatie met een trace- en log id waartoe deze activiteiten behoren. Onze variant van de log2vec-architectuur zal geen gebruik maken van de trace id en zullen we in de rest van deze paper het representatie-model noemen. Het doel van deze architectuur is het voorspellen van de volgende activiteit, gegeven haar context (de voorafgaande activiteiten) en de id van de event log waartoe deze activiteiten behoren. Het idee is dat de gewichten die deze log id verbinden aan de rest van het netwerk tijdens het trainingsproces zodanig aangepast zullen worden dat deze informatie omtrent de karakteristieken van de event log zullen capteren. Na afloop van het trainingsproces zal de vector representatie van een event log dus gelijk zijn aan de vector die de activatie-waarden bevat van deze gewichten. Welke activatie-functie we zullen gebruiken, alsook enkele andere design beslissingen die we hebben genomen omtrent het representatie-model zullen worden beschreven in sectie 3.1.

3.1 Architectuur en design beslissingen van het representatie-model

De input laag De input laag van het representatie model bestaat uit twee onderdelen:

1. Een aantal activiteiten die de te voorspellen activiteit voorafgaan.
2. De log id van de event log waartoe deze activiteiten behoren.

Zowel de activiteiten als de log id zullen in deze input laag worden voorgesteld door hun respectievelijke one-hot-encoding vector. Een one-hot-encoding vector is een vector met een lengte die gelijk is aan het aantal categorieën in een dataset, waarbij alle elementen uit deze vector een waarde van 0 krijgen toegewezen, behalve het i'de element, die een waarde van 1 toegewezen krijgt indien het data punt dat we encoderen behoort tot de i'de categorie. Voor de activiteiten is het aantal categorieën en de lengte van de one-hot-encoding vector dus gelijk aan het aantal unieke activiteiten dat voorkomt in de corpus. Voor de log id is het aantal categorieën en de lengte van de one-hot-encoding vector gelijk aan het aantal event logs uit de corpus.

Het aantal activiteiten dat we zullen meegeven als input van het representatie-model hangt af van een bepaalde venstergrootte die we zelf kunnen bepalen. De

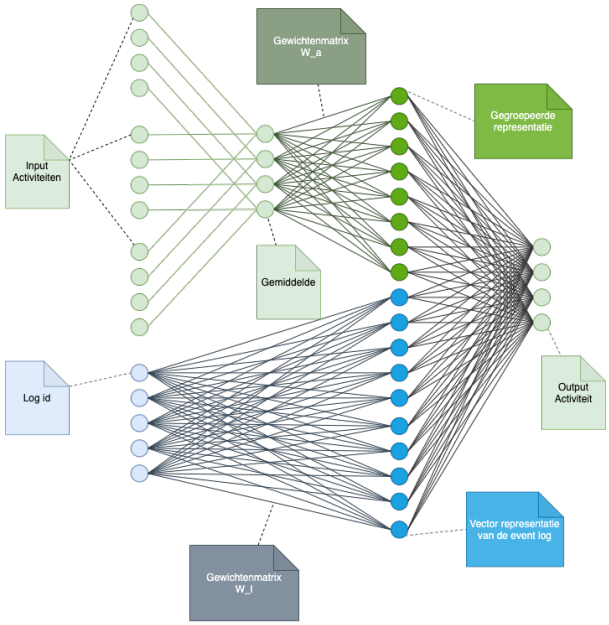

Fig. 2. Architectuur van het representatie-model

verschillende traces uit de corpus zullen worden opgesplitst in kleinere groepen, ofwel training-samples, die even groot zijn als deze venstergrootte. De eerste j activiteiten uit deze groepen, waarbij j gelijk is aan de $venstergrootte - 1$, zullen aan het representatie-model worden meegegeven als inputs. De laatste activiteit uit ieder van deze groepen is de te voorspellen activiteit en zal dus aan bod komen in de output laag van het representatie-model. In deze paper zullen we experimenteren met een venstergrootte van vier activiteiten. Dit wil zeggen dat we in deze paper bijgevolg drie input activiteiten zullen meegeven aan het representatie-model. Alle input activiteiten zullen vervolgens samengevoegd worden door het gemiddelde te berekenen van hun one-hot-ecoding vectoren in een volgende laag van het netwerk. Deze nieuwe laag zal vervolgens via de gewichtex-matrix W_a verbonden worden met de rest van het netwerk. De log id input zal gebruik maken van een aparte gewichten-matrix W_l die uiteindelijk de basis zal vormen voor de vector representaties.

Om het aantal groepen waarin een trace kan worden verdeeld en dus het aantal training-samples voor het representatie-model te maximaliseren zullen we afhankelijk van de venstergrootte een aantal dummy-variabelen toevoegen aan het begin van ieder trace. Het aantal dummy-variabelen dat we zullen toevoegen is gelijk aan $venstergrootte - 2$. Door $venstergrootte - 2$ dummy-variabelen toe te voegen aan het begin van iedere trace introduceren we een vorm van padding en verzekeren we dat er in iedere training-sample minstens één input-activiteit zal zijn en kunnen alle traces gebruikt worden om het representatie-model te

trainen vanaf dat de traces een minimale lengte hebben van twee activiteiten (één input- en één output-activiteit). Tabel 3 toont de transformatie van de corpus beschreven in tabel 2 tot een reeks training-samples gebruik makende van een venstergrootte van vier activiteiten.

Tabel 3. From corpus to training samples

Training sample	Trace	Activiteit-inputs	Output
1	[f, c, d]	[DUMMY, DUMMY, f]	c
2	[f, c, d]	[DUMMY, f, c]	d
3	[f, c, e, k, i]	[DUMMY, DUMMY, f]	c
4	[f, c, e, k, i]	[DUMMY, f, c]	e
5	[f, c, e, k, i]	[f, c, e]	k
6	[f, c, e, k, i]	[c, e, k]	i
7	[a, c, d, g, b, e, g, b, e, h, j, k, i]	[DUMMY, DUMMY, a]	c
8	[a, c, d, g, b, e, g, b, e, h, j, k, i]	[DUMMY, a, c]	d
9	[a, c, d, g, b, e, g, b, e, h, j, k, i]	[a, c, d]	g
10	[a, c, d, g, b, e, g, b, e, h, j, k, i]	[c, d, g]	b
11	[a, c, d, g, b, e, g, b, e, h, j, k, i]	[d, g, b]	e
12	[a, c, d, g, b, e, g, b, e, h, j, k, i]	[g, b, e]	g
13	[a, c, d, g, b, e, g, b, e, h, j, k, i]	[b, e, g]	b
14	[a, c, d, g, b, e, g, b, e, h, j, k, i]	[e, g, b]	e
15	[a, c, d, g, b, e, g, b, e, h, j, k, i]	[g, b, e]	h
16	[a, c, d, g, b, e, g, b, e, h, j, k, i]	[b, e, h]	j
17	[a, c, d, g, b, e, g, b, e, h, j, k, i]	[e, h, j]	k
18	[a, c, d, g, b, e, g, b, e, h, j, k, i]	[h, j, k]	i
19	[g, h]	[DUMMY, DUMMY, g]	h
20	[f]	/	/

De verborgen laag Net zoals de input laag bestaat ook de verborgen laag van het representatie-model uit twee onderdelen:

1. Een vector met een lengte van 32 knooppunten, volledig verbonden met de gemiddelde-laag van de input activiteiten.
2. Een vector met een lengte gelijk aan de gewenste lengte van de te verkrijgen vector representaties van de event logs, volledig verbonden met de one-hot-encoding vector van de log id input.

De gewichten-matrix W_l kan als volgt worden beschreven met een set van variabelen beschreven in tabel 4:

$$W_l = \begin{bmatrix} W_{1,1} & W_{1,2} & ... & W_{1,L-1} & W_{1,L} \\ W_{2,1} & W_{2,2} & ... & W_{2,L-1} & W_{2,L} \\ ... & ... & ... & ... & ... \\ W_{r-1,1} & W_{r-1,2} & ... & W_{r-1,L-1} & W_{r-1,L} \\ W_{r,1} & W_{r,2} & ... & W_{r,L-1} & W_{r,L} \end{bmatrix}$$

Variabele	Beschrijving
L	Het aantal event logs in de corpus
r	De gewenste lengte van de vector representaties

Tabel 4. Beschrijving variabelen

De output laag De output laag bestaat uit één one-hot-encoding vector voor de te voorspellen activiteit. Deze vector is volledig verbonden met de beide delen uit de verborgen laag. De uiteindelijke architectuur van het representatie-model is te zien in Figuur 2. De gebruikte activatie- en loss-functies worden beschreven in tabel 5.

Laag	Activatie functie	Loss functie
Verborgen laag, onderdeel 1	Rectified linear (ReLu)	/
Verborgen laag, onderdeel 2	Rectified linear (ReLu)	/
Output laag	Softmax	Sparse categorical crossentropy

Tabel 5. Beschrijving design keuzes representatie-model

3.2 Experimenting with the representation network

Tabel 6. Example iteration configurations

Dataset	Populatie	model	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18
1	1	1	1	65	1000	32	7	11	8	0.2	0.2	0.4	0.2	0	0	0.1	0.2	0	0	3
1	2	1	1	65	1000	32	7	11	8	0.2	0.2	0.4	0.2	0	0	0.1	0.2	0	0.25	3
1	1	2	1	65	1000	64	7	11	8	0.2	0.2	0.4	0.2	0	0	0.1	0.2	0	0	3
1	2	2	1	65	1000	64	7	11	8	0.2	0.2	0.4	0.2	0	0	0.1	0.2	0	0.25	3
2	1	1	1	82	1000	32	6	14	12	0.2	0.2	0.1	0.2	0.25	1	0.1	0.3	0	0	7
2	2	1	1	82	1000	32	6	14	12	0.2	0.2	0.1	0.2	0.25	1	0.1	0.3	0	0.4	7
2	1	2	1	82	1000	64	6	14	12	0.2	0.2	0.1	0.2	0.25	1	0.1	0.3	0	0	7
2	2	2	1	82	1000	64	6	14	12	0.2	0.2	0.1	0.2	0.25	1	0.1	0.3	0	0.4	7

Het hoofddoel van het representatie-model is het genereren van een vector representatie van een event log die essentiële informatie omtrent de karakteristieken van de event log capteert. Om het hoofddoel van het representatie-model zo goed mogelijk te bereiken trachten we de nauwkeurigheid van het model te maximaliseren omdat dit zou betekenen dat de essentiële informatie over de karakteristieken van de event log succesvol werd gecapteerd in de gewichten van het model en bijgevolg in de vector representaties van de event logs. Er zijn verschillende factoren die de nauwkeurigheid van het representatie-model kunnen beïnvloeden. De factoren waarvan we hun invloed op de nauwkeurigheid van het representatie-model in dit onderzoek zullen analyseren, alsook reden waarom we voor deze factoren hebben gekozen zullen hieronder worden beschreven.

1. **Het aantal artificieel gegenereerde event logs per procesboom (Parameter P2):** We zullen de impact van deze parameter onderzoeken om

Parameter	Breekpunt
Het aantal artificieel gegenereerde event logs per procesboom (P2)	75
Het totaal aantal verschillende activiteiten in de corpus (P7)	10
De lengte van de vector representatie (P4)	32

Tabel 7. Een overzicht van de breekpunten voor de t-statistiek

mogelijk een beter zicht te krijgen op de hoeveelheid data die nodig om een accurate vector representatie te kunnen genereren.

2. **Het totaal aantal verschillende activiteiten in de corpus (Parameter P7):** Intuïtief gaan we ervan uit dat indien er meer verschillende activiteiten aanwezig zijn in de corpus, de accuraatheid van het representatie-model achteruit zal gaan. Door de impact van deze parameter te onderzoeken kunnen we deze hypothese bevestigen of verwerpen.
3. **De lengte van de vector representatie (Parameter P4):** Hoe langer de vector representatie, hoe meer gewichten het representatie-model zal moeten trainen en dus hoe langer het zal duren om een vector representatie van een event log te bekomen. Met dit in ons achterhoofd zullen we de impact van deze parameter onderzoeken met het oog op het optimaliseren van het representatie-model.

Na het genereren en voorbereiden van de data kunnen we eindelijk beginnen met het trainen en evalueren van het representatie-model. Om over voldoende gevarieerde data te beschikken zullen we de data generatie- en preparatieprocedure uitgebeeld in figuur 1 honderd maal herhalen, resulterend in honderd datasets. Vervolgens zullen we ieder van deze honderd datasets gebruiken om het representatie-model te trainen en te evalueren.

Omdat de lengte van de vector representatie eigenlijk een hyperparameter is kunnen we echter niet zomaar de lengte van de representatie op dezelfde manier willekeurig bepalen zoals we dit bij andere parameters wel kunnen. Een verandering in de lengte van een vector representatie betekent namelijk een verandering in de architectuur van het representatie-model en bijgevolg een apart neuraal netwerk dat getraind zal moeten worden. Om die reden zullen we twee afzonderlijke neurale netwerken trainen. Elk netwerk wordt getraind en geëvalueerd op exact dezelfde gegevens, met als enige verschil de lengte van de vector representatie van respectievelijk 32 en 64 knooppunten voor beide netwerken. Enkele voorbeeld configuraties zijn te vinden in tabel 6.

Uit dit training- en evaluatieprocedure ontstaan honderd observaties met telkens twee metingen omtrent de accuraatheid van het representatie-model. De metingen weerspiegelen de accuraatheid van het representatie-model indien we gebruik maken van vector representatie lengtes van respectievelijk 32 en 64 knooppunten. Aan de hand van deze observaties zullen we door middel van een t-test het effect van enkele parameters op de accuraatheid van het representatie-model testen.

Omdat de eerste twee parameters (P2, P7) vrij en willekeurig kunnen variëren zoals beschreven in Tabel 1, zullen we de observaties in twee groepen splitsen om de t-statistiek te testen. Hierbij hebben we ervoor gekozen om het midden van de parameter-intervallen, zoals beschreven in tabel 1, te kiezen als breekpunten om de observaties op te splitsen. Aangezien de derde parameter (De lengte van de vector representaties) slechts één van twee keuzes kan zijn, namelijk 32 en 64, spreekt het voor zich om de honderd waarnemingen in twee groepen te splitsen op basis van deze parameterwaarden. Een overzicht van de breekpunten die werden gebruikt om de waarnemingen op te splitsen in twee groepen is te vinden in de tabel 7.

3.3 Resultaten

De gemiddelde accuraatheid van het representatie-model bedraagt 49,63%. Om dit resultaat op een correcte manier te kunnen interpreteren hebben we nood aan een baseline waarmee we de behaalde accuraatheid kunnen vergelijken. De baseline die we hier zullen gebruiken staat gelijk aan de gemiddelde accuraatheid van het representatie-model indien het model tijdens het trainen van iedere dataset simpelweg de meest voorkomende activiteit in de event log zou voorspellen. De gemiddelde baseline over alle datasets heen bedraagt 23,87%. Hoewel de gevonden accuraatheid op het eerste zicht eerder negatief lijkt te zijn hebben we met dit representatie-model toch een verbetering van 25,76% kunnen bereiken ten opzichte van de baseline. Om tot dit resultaat te komen werd het gemiddelde genomen van de totale 200 metingen die ontstaan zijn uit de 100 observaties met telkens twee metingen, komende uit de 100 verschillende datasets. Hieronder worden de resultaten besproken omtrent de invloed van de parameters opgelijst in tabel 7.

1. Het aantal artificieel gegenereerde event logs per procesboom (P2):

   H_0: Het aantal artificieel gegenereerde event logs per procesboom heeft geen effect op de accuraatheid van het representatie-model.

   H_1: Het aantal artificieel gegenereerde event logs per procesboom heeft wel een effect op de accuraatheid van het representatie-model.

   De p-waarde van de t-statistiek uit deze test bedraagt: 0.578548328792989. Het lukt ons op basis van deze t-test dus niet om de nulhypothese niet verwerpen op een 95% of hoger betrouwbaarheidsinterval.

   De gemiddelde accuraatheid voor de groep ¡= breekpunt: 49,33% De gemiddelde accuraatheid voor de groep ¿ breekpunt: 50,05%

   Dit resultaat suggereert dat het benodigde aantal event logs per procesboom niet direct een impact heeft op het potentieel om een accurate vector representatie van een event log te bekomen. Omdat er in dit onderzoek slechts één procesboom werd gegenereerd per populatie staat het aantal event logs

per procesboom hierbij gelijk aan het aantal event logs per populatie. Dit zou betekenen dat een relatief beperkte hoeveelheid event logs per populatie voldoende zou kunnen zijn om en accurate vector representatie van een event log te bekomen.

2. Het totaal aantal verschillende activiteiten in de corpus (P7):

   H_0: Het totaal aantal verschillende activiteiten in de corpus heeft geen effect op de accuraatheid van het representatie-model.

   H_1: Het totaal aantal verschillende activiteiten in de corpus heeft wel een effect op de accuraatheid van het representatie-model.

   De p-waarde van de t-statistiek uit deze test bedraagt: 8.295638327866961e-07. Op basis van deze t-test kunnen we de nulhypothese dus verwerpen op een 99% betrouwbaarheidsinterval.

   De gemiddelde accuraatheid voor de groep ¡= breekpunt: 51,19% De gemiddelde accuraatheid voor de groep ¿ breekpunt: 43,76%

   Hoe meer verschillende activiteiten in de corpus, hoe moeilijker het is om de correcte activiteit te voorspellen.

3. De lengte van de vector representatie (P4):

   H_0: De lengte van de vector representatie heeft geen effect op de accuraatheid van het representatie-model.
   H_1: De lengte van de vector representatie heeft wel een effect op de accuraatheid van het representatie-model.

   De p-waarde van de t-statistiek uit deze test bedraagt: 0.9845604091236053. Het lukt ons op basis van deze t-test dus niet om de nulhypothese niet verwerpen op een 95% of hoger betrouwbaarheidsinterval.

   De gemiddelde accuraatheid voor de groep ¡= breekpunt: 49,64% De gemiddelde accuraatheid voor de groep ¿ breekpunt: 49,62%

   Uit deze resultaten blijkt dat de lengte van de vector representatie geen effect heeft op de accuraatheid van het representatie-model en bijgevolg de kwaliteit van de vector representatie. Gezien de populaties waartoe de verschillende event logs behoren telkens beschikken over grotendeels dezelfde karakteristieken, zou dit een eventuele verklaring kunnen zijn voor dit resultaat. Het is namelijk mogelijk dat door het kleine verschil in de karakteristieken van de populaties slechts een beperkte lengte voor de vector representaties voldoende is om dit verschil te capteren.

4 Clusteren van vector representaties van event logs

Nu de training- en validatieprocedure van het representatie-model achter de rug is zullen we het antwoord op de volgende vragen analyseren: Kunnen we, gebruik makende van de vector representaties van de event logs uit de corpus, clusters herkennen, en zo ja, weerspiegelen deze clusters de verschillende populaties uit de verschillende datasets?

Om een antwoord op deze vraag te vinden zullen we gebruik maken van het k-means clustering algoritme om op zoek te gaan naar twee clusters die de verschillende populaties zouden weerspiegelen. Het k-means clustering algoritme probeert een anonieme dataset op te splitsen in een vast aantal clusters (k). Hierbij wordt voor iedere cluster een initieel en willekeurige zwaartepunt aangemaakt. Een zwaartepunt is een datapunt in het centrum van een cluster. Vervolgens bestaat het k-means clustering algoritme uit twee stappen die herhaald kunnen worden.

1. De k zwaartepunten worden gebruikt om k clusters te maken door ieder datapunt uit de dataset toe te voegen aan de cluster van het zwaartepunt dat er het dichtst bij ligt.
2. Nadat alle datapunten uit de dataset verdeeld werden onder de k clusters wordt de locatie van ieder zwaartepunt geoptimaliseerd en verplaatst naar het centrum van de cluster die ervoor gemaakt werd.

Dit proces wordt herhaald totdat de waarden van de zwaartepunten stabiliseren. Tot slot worden de uiteindelijke zwaartepunten gebruikt om de uiteindelijke clustering van de invoergegevens te produceren.

Omdat de resultaten van het k-means clustering algoritme afhankelijk kunnen zijn van de initiële positie van de zwaartepunten gaan we als volgt te werk:

- We voeren het k-means clustering algoritme individueel uit voor de honderd verschillende datasets.
- We bepalen de accuraatheid van de clustering op basis van een confusion matrix. Een confusion matrix is een matrix waarin het aantal correcte en foutieve voorspellingen wordt samengevat met telwaarden en uitgesplitst per klasse. In het kader van dit onderzoekt toont de confusion matrix het aantal correct geclusterde event logs (Correct) en het aantal foutief geclusterde event logs (Foutief). De accuraatheid van het clustering algoritme wordt dan bepaald door:
$$Correct/(Correct + Foutief)$$
- We herhalen dit proces tien maal per dataset en berekenen het gemiddelde van de waargenomen accuraatheden.
- Aangezien we beschikken over honderd datasets hebben we nu dus ook honderd metingen omtrent de gemiddelde accuraatheid van het clustering algoritme. Om een totaal beeld te krijgen over de accuraatheid van het clustering

algoritme over alle datasets heen berekenen we vervolgens ook de gemiddelde accuraatheid van de gemiddelden uit de verschillende datasets.

4.1 Resultaten

Het clusteren van de vector representaties en het berekenen van de gemiddelde accuraatheid over alle datasets heen behaalt consistent de volgende resultaten.

- Accuraatheid voor clustering van vector representaties met lengte 32: 75% met een standaard deviatie van 12%.
- Accuraatheid voor clustering van vector representaties met lengte 64: 75% met een standaard deviatie van 12%.

Naast de consistente resultaten op basis van de hierboven beschreven gemiddelden behalen ook de individuele datasets consistent zo goed als dezelfde resultaten bij het clusteren. Sommige datasets behalen dus een accuraatheid van 80% of 90% terwijl andere datasets een accuraatheid hebben van 50% of 60%. Waarin deze datasets van elkaar verschillen is op dit moment niet duidelijk en daarom een kandidaat voor verder onderzoek.

5 Extraheren van karakteristieken uit de vector representatie van een event log

Omdat we de karakteristieken van de populaties in de corpus zelf hebben bepaald, hebben we nu de mogelijkheid te testen of we deze karakteristieken opnieuw kunnen extraheren uit de verkregen vector representaties van de event logs uit deze populaties. In deze paper zullen we meer specifiek een antwoord zoeken op de volgende vraag: Kunnen we aan de hand van een vector representatie van een event log voorspellen of deze event log al dan niet beschikt over loops, en zo ja, hoe vaak kunnen deze loops zich dan herhalen binnen één enkele trace?

We zullen deze vraag beantwoorden door twee verschillende neurale netwerken te trainen die ieder een van de twee parameters zullen voorspellen. In de rest van deze paper zullen we deze neurale netwerken benoemen door het predictie-model voor respectievelijk de loops en max repeat parameters.

5.1 Architectuur en design beslissingen van de predictie-modellen

In deze paper zullen we voor zowel het predictie-model voor de loops parameter als het predictie-model van de max repeat parameter op de output laag na dezelfde architectuur gebruiken.

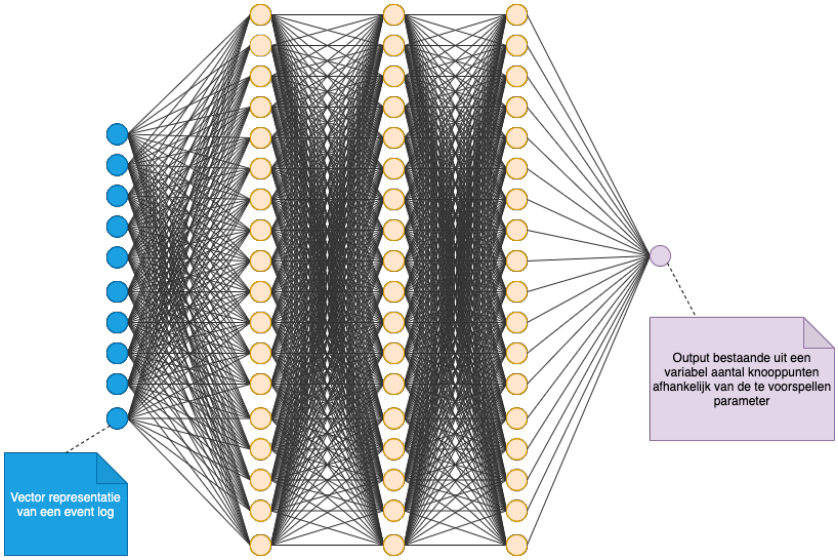

Fig. 3. Prediction network architecture

De input laag De input laag voor beide predictie-modellen bestaat uit één enkele invoer. Deze invoer is een vector representatie van een event log die wordt verkregen uit het representatie-model. Omdat we twee afzonderlijke representatie-modellen hebben die ieder vector representaties met een respectievelijke lengte van 32 en 64 knooppunten genereren, hebben we per parameter (loops en max repeat) ook twee afzonderlijke predictie-modellen nodig die elk een vector representatie van een van de lengtes als invoer kunnen nemen.

De training- en test-data voor de predictie-modellen voor de loops parameter zullen bestaan uit respectievelijk 80% en 20% van alle vector representaties uit iedere dataset. De verdeling van het aantal event logs dat al dan niet beschikt over loops is hierbij gelijk verdeeld.

Voor de training- en test-data voor de predictie-modellen voor de max repeat parameter zullen we daarentegen twee verschillende varianten testen. De eerste variant zal beschikken over identiek dezelfde data die gebruikt werd voor de predictie-modellen voor de loops parameter. Omdat in deze data echter een duidelijke meerderheidsklasse te vinden is wat betreft de max repeat parameter waarbij 50% van alle datapunten een max repeat waarde hebben van 0 zullen we deze data niet meenemen in de training- en test-data voor de tweede variant. De data voor de tweede variant zal dus enkel bestaan uit respectievelijk 80% en 20% van alle vector representaties waartoe event logs behoren waarvan de karakteristieken bepalen dat loops mogelijk zijn. Op deze manier zal er ook hier geen meerderheidsklasse bestaan die de accuraatheid van de predictie-modellen zou kunnen beïnvloeden.

De verborgen lagen De predictie-modellen bevatten ieder drie verborgen lagen. Deze verborgen lagen zijn telkens volledig verbonden met de vorige laag in het netwerk en bestaan telkens uit 128 knooppunten. Ieder van deze verborgen lagen maken bovendien gebruik van de rectified linear (relu) activatie functie.

De output laag Zowel de predictie-modellen voor de loops parameter als de predictie-modellen voor de max repeat parameter bevatten ieder één enkele output in de output laag. Deze outputs maken gebruik van de softmax activatie functie en bestaan uit een vector met een aantal knooppunten die gelijk is aan het aantal waarden dat de output kan aannemen. Voor de loops parameter zijn dit twee waarden (0 of 1), voor de max repeat parameter zijn dit alle gehele getallen in het interval dat wordt beschreven in tabel 1. Aangezien deze predictie-modellen trachten een classificatie probleem op te lossen maken we om de accuraatheid van de modellen te bepalen gebruik van de sparse categorical crossentropy loss functie.

5.2 Resultaten

De resultaten voor de verschillende te voorspellen parameters uit het predictiemodel zijn als volgt:

- Voor vector representaties van lengte 32:
  - Accuraatheid loop voorspelling: 63,79%
  - Accuraatheid max repeat voorspelling (variant 1): 64,62%
  - Accuraatheid max repeat voorspelling (variant 2): 85,36%
- Voor vector representaties van lengte 64:
  - Accuraatheid loop voorspelling: 64,51%
  - Accuraatheid max repeat voorspelling (variant 1): 63,69%
  - Accuraatheid max repeat voorspelling (variant 2): 90,26%

De baseline voor de loop voorspelling is hierbij gelijk aan 50%. De baseline voor de max repeat parameter is verschillend voor beide varianten en bedraagt 50% voor variant 1 en 17,61% voor variant 2. Tabel 8 toont de verdeling van de waarden voor de max repeat parameter.

Tabel 8. Overzicht van het aantal voorkomens van de waarden voor de max repeat parameter

Waarde	0	2	3	4	5	6	7	8	9
Voorkomens	7256	697	1127	943	940	468	813	990	1278

6 Gerelateerd werk

De opzet van dit onderzoek draait rond het creëren van realistische artificiële event data. Dit onderzoek is voornamelijk interessant in het kader van het onderzoek naar het vergelijken van algoritmen die procesmodellen kunnen herkennen. Onderzoek naar het vergelijken van dergelijke algoritmen kan worden gevonden in [13],[17],[16],[11],[10].

Omdat dergelijke evaluaties grote hoeveelheden geschikte gegevens nodig hebben als input voor empirische analyse wordt het gebruik van artificiële event logs wel eens bepleit ([6]). Naast de PTandLogGenerator ([5]) die in deze paper gebruikt werd voor het creëren van artificiële procesbomen en event logs bestaan er nog andere tools die hetzelfde trachten te bereiken ([1]).

Wat betreft het gebruik van vector representaties om event logs voor te stellen kan het interessant zijn [2] door te nemen voor meer informatie omtrent de log2vec architectuur. Naast event logs worden vector representaties eveneens gebruikt om onder andere woorden of zinnen voor te stellen ([8],[9],[12]),[7]. Hierbij kan het interessant zijn om gelijkenissen, verschillen en succesfactoren te onderzoeken.

7 Conclusie

In het kader van het onderzoek naar het vergelijken van algoritmen die procesmodellen kunnen herkennen uit een event log wordt het gebruik van artificiële event logs wel eens bepleit om een tekort aan reële data op te vangen. Echter is het hierbij wel belangrijk dat deze artificiële data als realistisch beschouwd kan worden en de inhoud van reële event logs weerspiegelt. Desondanks werd er tot op heden niet veel aandacht besteed aan het garanderen van realisme binnen artificieel gegenereerde event logs. Het doel van deze paper was om hier een bijdrage aan te leveren door te analyseren of we onderliggende proceskenmerken (karakteristieken) kunnen capteren uit event logs met behulp van representatiemodellen.

De resultaten gevonden in deze paper suggereren dat het gebruik van vector representaties om event logs voor te stellen een beloftevolle techniek kan zijn om essentiële informatie omtrent de karakteristieken van deze event logs te capteren. Hoewel dit het eerste onderzoek is tonen we het potentieel om karakteristieken van een event log te herkennen uit een vector representatie.

Als uitbreiding van dit onderzoek zou men op zoek kunnen gaan naar manieren om het predictie-model uit te breiden zodanig dat naast de karakteristieken waarmee in deze paper werden geëxperimenteerd, ook andere karakteristieken uit de vector representaties herkend kunnen worden.

Referenties

1. Burattin, A.: Plg2: Multiperspective processes randomization and simulation for online and offline settings (06 2015)

2. De Koninck, P., vanden Broucke, S., Weerdt, J.: act2vec, trace2vec, log2vec, and model2vec: Representation Learning for Business Processes: 16th International Conference, BPM 2018, Sydney, NSW, Australia, September 9–14, 2018, Proceedings, pp. 305–321 (01 2018). https://doi.org/10.1007/978-3-319-98648-7$_1$8
3. De Weerdt, J., De Backer, M., Vanthienen, J., Baesens, B.: A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs. Inf. Syst. **37**, 654–676 (11 2012). https://doi.org/10.1016/j.is.2012.02.004
4. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer, Heidelberg (2013), http://link.springer.com/content/pdf/10.1007/978-3-642-33143-5.pdf
5. Jouck, T., Depaire, B.: PTandLogGenerator: a generator for artificial event data (09 2016)
6. Jouck, T., Depaire, B.: Generating artificial data for empirical analysis of control-flow discovery algorithms: A process tree and log generator. Business Information Systems Engineering **61** (03 2018). https://doi.org/10.1007/s12599-018-0541-5
7. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. 31st International Conference on Machine Learning, ICML 2014 **4** (05 2014)
8. Mikolov, T., Corrado, G., Chen, K., Dean, J.: Efficient estimation of word representations in vector space. pp. 1–12 (01 2013)
9. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. Advances in Neural Information Processing Systems **26** (10 2013)
10. Ribeiro, J., Carmona, J.: A Method for Assessing Parameter Impact on Control-Flow Discovery Algorithms, vol. 9930, pp. 181–202 (09 2016). https://doi.org/10.1007/978-3-662-53401-4$_9$
11. Ribeiro, J., Carmona, J., Mısır, M., Sebag, M.: A recommender system for process discovery (09 2014). https://doi.org/10.1007/978-3-319-10172-9$_5$
12. Rong, X.: word2vec parameter learning explained (11 2014)
13. Rozinat, A., De Medeiros, A.K.A., Günther, C.W., Weijters, A.J.M.M., van der Aalst, W.: Towards an evaluation framework for process mining algorithms. Reactivity of Solids (01 2007)
14. van der Aalst, W.: Process Mining. Springer, Berlin, Heidelberg (2016)
15. vanden Broucke, S.K., Delvaux, C., Freitas, J., Rogova, T., Vanthienen, J., Baesens, B.: Uncovering the relationship between event log characteristics and process discovery techniques. pp. 41–53 (05 2014). https://doi.org/10.1007/978-3-319-06257-0$_4$
16. Wang, J., Wong, R.K., Ding, J., Guo, Q., Wen, L.: On recommendation of process mining algorithms. pp. 311–318 (06 2012). https://doi.org/10.1109/ICWS.2012.52
17. Weber, P., Bordbar, B., Tino, P.: A framework for the analysis of process mining algorithms. Systems, Man, and Cybernetics: Systems, IEEE Transactions on **43**, 303–317 (03 2013). https://doi.org/10.1109/TSMCA.2012.2195169