Extracción de conocimiento basado en técnicas de aprendizaje automático a partir de registros de eventos*

Tomás Benavides Álvarez, Manuel Mucientes, and Manuel Lama

Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS)
Universidade de Santiago de Compostela, España
tomas.benavides.alvarez@rai.usc.es, {manuel.mucientes,manuel.lama}@usc.es

Resumen Uno de los principales objetivos en minería de procesos consiste en entender lo que ha sucedido durante la ejecución de un proceso. Típicamente, este objetivo se alcanza explorando manualmente el modelo real que describe el comportamiento del proceso y las analíticas temporales y de frecuencia sobre las variantes del proceso y los indicadores de negocio. En este artículo se sigue una aproximación diferente: se ha desarrollado una técnica basada en árboles de decisión para la extracción automática de conocimiento sobre la ejecución del proceso. Esta técnica se ha validado con un conjunto de datos sintético y se ha comparado con una aproximación de referencia del estado del arte, obteniendo las mismas reglas de conocimiento en ambos casos, con la diferencia de que nuestra aproximación guía la extracción de las reglas a partir de las necesidades del usuario.

Keywords: Minería de procesos · Árboles de decisión · Extracción de conocimiento.

1. Introducción

La minería de procesos es una disciplina que hace de puente entre la minería de datos y la gestión de procesos de negocio tradicional al ofrecer una serie de técnicas que permiten extraer conocimiento de los registros de eventos en los que la mayoría de los sistemas de información almacenan los datos sobre las diferentes ejecuciones de los procesos a los que dan soporte. Actualmente, esta área está ganando cada vez más importancia gracias al aumento de la información que se recoge durante la ejecución de los diferentes procesos, y al gran interés que está

* Este trabajo ha sido financiado por la Consellería de Educación, Universidade e Formación Profesional (accreditaciónn 2019-2022 ED431G-2019/04), el Fondo Europeo de Desarrollo Regional (FEDER), que reconoce al CiTIUS —Centro Singular de Investigacion en Tecnoloxías Intelixentes da Universidade de Santiago de Compostela— como un Centro de Investigación del Sistema Universitario Gallego, y el Ministerio de Ciencia e Innovación (a través de los proyectos PDC2021-121072-C21 y PID2020-112623GB-I00).

suscitando en las empresas a la hora de aumentar tanto su competitividad como su rendimiento [1].

Dentro de la minería de procesos, existen numerosas técnicas y herramientas que permiten *descubrir*, *monitorizar* y *mejorar* estos procesos. Dentro de las técnicas de descubrimiento, las que han recibido una mayor atención se han centrado en la generación de un modelo que contiene la relación explícita entre las actividades del proceso [2]. Sin embargo, otras técnicas se han centrado en el descubrimiento de procesos declarativos, definidos mediante un lenguaje declarativo como DECLARE [3], donde se describen una serie de restricciones que se deben satisfacer durante la ejecución del proceso [4], ofreciendo una descripción más clara de las relaciones entre las actividades de un proceso cuando se trata de trabajar con procesos poco estructurados. Además, estos procesos declarativos también pueden contener restricciones (entendidas como reglas) relacionadas con indicadores de negocio y con información temporal entre actividades, para lo cual se ha extendido el lenguaje DECLARE a una versión multi-perspectiva que contiene este tipo de elementos, *MP-DECLARE* [5].

Ahora bien, en el descubrimiento de procesos declarativos o bien se intentan obtener todas las restricciones especificadas en el lenguaje MP-DECLARE, lo cual puede conllevar a que los algoritmos de descubrimiento no converjan (sobre todo cuando el número de actividades es elevado), o bien se obtiene únicamente un conjunto de esas restricciones, lo cual puede ocultar información que podría ser relevante para los usuarios. En este artículo se propone una técnica que pretende abordar este problema: el descubrimiento de las restricciones (o reglas) se basa en las consultas que desean resolver los usuarios y que está relacionada con la forma en la que se han ejecutado los casos del registro de eventos. Por ejemplo, se extraen el conjunto de reglas que explican los casos en los cuales el procedimiento aplicado a un paciente ha sido la cirugía.

En este artículo se presenta una técnica para la extracción automática de conocimiento a partir de registros de eventos que está basada en árboles de decisión, facilitando con ello la explicación de los resultados. Esta técnica es una aproximación inicial al problema y se ha validado con un conjunto de datos sintético, comparándola con otra propuesta del estado del arte [6]. Cabe destacar que en la validación se han obtenido los mismos resultados que en la propuesta del estado del arte, aunque en nuestra aproximación no es necesario indicar qué tipo de restricciones entre actividades se deberán identificar.

El resto del artículo se estructura de la siguiente manera: la Sección 2 describe la aproximación basada en árboles de decisión para la extracción de conocimiento a partir de registros de eventos; la Sección 3 muestra los resultados obtenidos en la validación llevada a cabo; y finalmente, la Sección 4 presenta las principales conclusiones del artículo y futuras lineas de trabajo a afrontar.

2. Descripción de la aproximación

La solución propuesta está basada en *árboles de decisión*, un tipo de modelo de predicción que permite clasificar un conjunto de datos en base a sus carac-

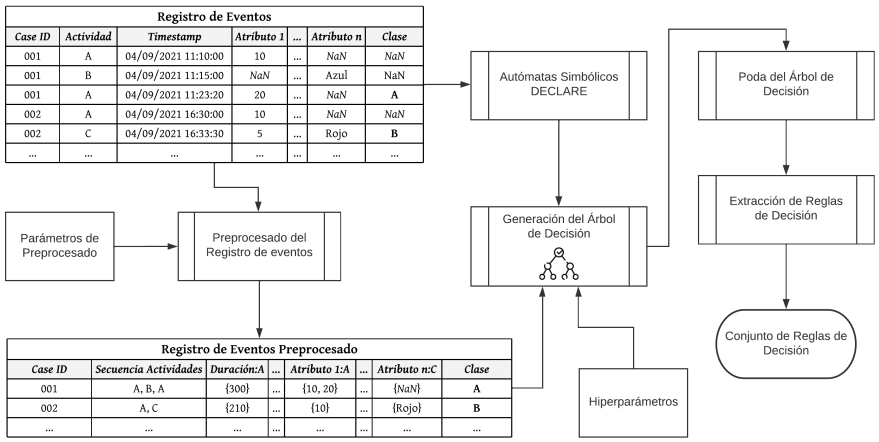
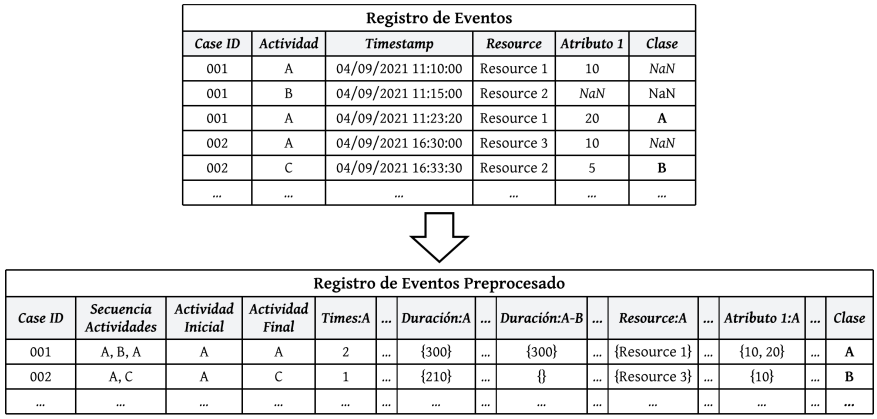


Figura 1. Arquitectura del *pipeline* diseñado para la solución propuesta.

terísticas diferenciales. El uso de árboles de decisión se debe, principalmente, a que ofrecen la posibilidad de extraer de manera intuitiva el conjunto de reglas *if-then* que se suceden en sus nodos de decisión, haciendo que el modelo resultante sea *explicable* e *interpretable*. La arquitectura de la solución se muestra en la Figura 1, donde se toma como entrada un *registro de eventos*, que almacena toda la información generada en las diferentes ejecuciones del proceso, y a cuyos casos se les deberá asignar previamente una clase que define la consulta que se desee realizar. Por ejemplo, cada caso se etiquetará que ha finalizado con éxito, de acuerdo con algún criterio, o que ha sido cancelado.

Como entrada también se proporcionará una serie de *parámetros de preprocesado* que indicarán los atributos que se desean tener en cuenta para la generación del modelo, como puede ser la duración de las actividades, el recurso que las ejecuta o el valor de algún indicador de negocio concreto para alguna actividad. De esta forma, se realizará un **preprocesado** del registro de eventos inicial cuyo resultado será la entrada del algoritmo de generación del árbol de decisión. En la Figura 2 puede verse un ejemplo del proceso de preprocesado sobre un registro de eventos de un proceso cualquiera. Podemos ver cómo lo que se hace es reducir cada caso del proceso a una fila del *dataframe* para adaptarlo al algoritmo de generación del árbol de decisión y cómo, además, se realiza tanto la secuenciación de las actividades para emplearlas como características de entrada, como la generación de aquellos atributos que se habían indicado en los parámetros de preprocesado, entre los que se pueden encontrar el número de veces que se repite cada actividad, su duración o el recurso que las ejecuta.

Este algoritmo requerirá de dos entradas adicionales. La primera consiste en una serie de *hiperparámetros* proporcionados por el usuario, que serán los que dirijan el crecimiento del árbol indicando, por ejemplo, el número mínimo de



Registro de Eventos

Case ID	*Actividad*	*Timestamp*	*Resource*	*Atributo 1*	*Clase*
001	A	04/09/2021 11:10:00	Resource 1	10	*NaN*
001	B	04/09/2021 11:15:00	Resource 2	*NaN*	NaN
001	A	04/09/2021 11:23:20	Resource 1	20	**A**
002	A	04/09/2021 16:30:00	Resource 3	10	*NaN*
002	C	04/09/2021 16:33:30	Resource 2	5	**B**
...	...	...	...	...	...

Registro de Eventos Preprocesado

Case ID	*Secuencia Actividades*	*Actividad Inicial*	*Actividad Final*	*Times:A*	...	*Duración:A*	...	*Duración:A-B*	...	*Resource:A*	...	*Atributo 1:A*	...	*Clase*
001	A, B, A	A	A	2	...	{300}	...	{300}	...	{Resource 1}	...	{10, 20}	...	**A**
002	A, C	A	C	1	...	{210}	...	{}	...	{Resource 3}	...	{10}	...	**B**
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Figura 2. Preprocesado del registro de eventos.

muestras presentes en un nodo para que una división se considere válida o para que se pueda realizar una nueva división, o el valor límite de impureza permitida en un nodo para que este sea considerado un nodo hoja. La segunda de las entradas consiste en una serie de *autómatas simbólicos* que se generarán a partir del registro de eventos inicial y que permitirán verificar las diferentes restricciones impuestas por el lenguaje *DECLARE* [3] sobre las secuencias de actividades que conforman cada caso del proceso, generadas en la etapa de preprocesado. En la Figura 3 se muestra un ejemplo de autómata para la plantilla *Response(A, B)* de DECLARE, donde se indica que B siempre ocurre después de A. Cabe destacar que para la generación de estos autómatas se ha hecho uso de *FLLOAT*[1].

Por último, para la generación del árbol de decisión se ha decidido seguir un enfoque avaro (*greedy*) similar al que se utiliza en el algoritmo *CART* [7], de forma que para la división de cada nodo se seleccione siempre la característica de entrada que ofrezca la mayor ganancia de información, siguiendo como criterio de medición del error el criterio de *entropía*. La parte diferencial de este algoritmo radica en las características de entrada que se tienen en cuenta: están formadas por la combinación de las diferentes restricciones definidas por el lenguaje *DECLARE*, mostradas en la Tabla 1, con los atributos definidos en los parámetros de preprocesado proporcionados por el usuario, de forma que se discrimine los diferentes casos del proceso conforme cumplan o no alguna de las restricciones que se definen entre las actividades (por ejemplo, *Existence(A)*, *Response(A, B)*, etc.) y satisfagan (o no) un umbral dado para alguno de los atributos de estas actividades.

De este modo, se generará un árbol de decisión que posteriormente será *podado* para evitar el sobreajuste (*overfitting*); es decir, se procesará el árbol para eliminar nodos de decisión que dan lugar a divisiones demasiado específicas

[1] https://doi.org/10.5281/zenodo.2577006

Nombre	***Expresión LTL***	***Descripción***
Existence	$\Diamond a$	a debe ejecutarse al menos una vez.
Init	a	a debe ser la primera en ser ejecutada.
Last	$\Diamond(a \wedge \bigcirc\neg T)$	a debe ser la última en ser ejecutada.
Choice	$\Diamond a \vee \Diamond b$	a o b deben ser ejecutadas.
Exclusive Choice	$(\Diamond a \vee \Diamond b) \wedge \neg(\Diamond a \wedge \Diamond b)$	a y b deben ser ejecutadas, pero nunca ambas.
Responded Existence	$\Diamond a \rightarrow \Diamond b$	Si a es ejecutada, b también debe ejecutarse.
CoExistence	$(\Diamond a \rightarrow \Diamond b) \wedge (\Diamond b \rightarrow \Diamond a)$	a y b son ambas ejecutadas, o no lo hace ninguna de las dos.
Response	$\Box(a \rightarrow \Diamond b)$	Cada vez que a se ejecuta, b debe ejecutarse posteriormente.
Precedence	$\neg b \mathcal{W} a$	b puede ser ejecutada sólo si a ha sido ejecutada antes.
Sucession	$\Box(a \rightarrow \Diamond b) \wedge (\neg b \mathcal{W} a)$	Combinación de *Response* y *Precedence*.
Alternate Response	$\Box(a \rightarrow \bigcirc(\neg a \,\mathcal{U}\, b))$	Cada ejecución de a debe ir seguida de una ejecución de b, sin que a se vuelva a ejecutar entre ellas.
Alternate Precedence	$(\neg b \mathcal{W} a) \wedge$ $\Box(b \rightarrow \bigcirc(\neg b \mathcal{W} a))$	Cada b debe ir precedida de a, sin ninguna b entre ambas.
Alternate Sucession	$\Box(a \rightarrow \bigcirc(\neg a \mathcal{U} b)) \wedge$ $(\neg b \mathcal{W} a) \wedge$ $\Box(b \rightarrow \bigcirc(\neg b \mathcal{W} a))$	Combinación de *Alternate Response* y *Alternate Precedence*.
Chain Response	$\Box(a \rightarrow \bigcirc b)$	Cada vez que a se ejecuta, b debe ejecutarse inmediatamente a continuación.
Chain Precedence	$\Box(\bigcirc b \rightarrow a)$	b sólo puede ejecutarse inmediatamente a continuación de a.
Chain Sucession	$\Box(a \equiv \bigcirc b)$	Combinación de *Chain Response* y *Chain Precedence*.
Not CoExistence	$\neg(\Diamond a \wedge \Diamond b)$	a o b pueden ser ejecutadas, pero nunca ambas.
Not Sucession	$\Box(a \rightarrow \neg\Diamond b$	a no puede ir seguida de b, y b no puede ir precedida de a.
Not Chain Sucession	$\Box(a \equiv \bigcirc\neg b)$	a y b no pueden ser ejecutadas sucesivamente.

Tabla 1. Plantillas de DECLARE para las restricciones tenidas en cuenta en la aproximación junto con su expresión en lógica LTL. Extraídas de [8].

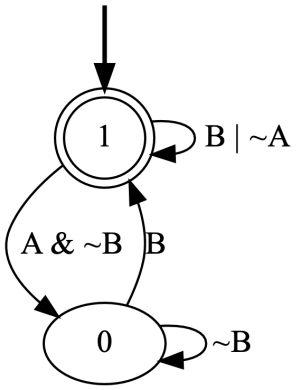


Figura 3. Autómata simbólico generado por *FLLOAT* para la plantilla *Response(A, B)*.

debido al sobreajuste que se produce sobre los datos con los que se ha generado el árbol, lo cual mejora su generalización a otros casos y la capacidad de predicción de las reglas que se extraerán a partir de este. Esta extracción de reglas, será la última fase de nuestra aproximación, y en ella cada conjunto de reglas está formado por las reglas que se suceden hasta cada nodo hoja del árbol, que tendrá asociada una de las posibles clases del proceso (la cual se explicará con este conjunto de reglas).

De esta forma, se generará un árbol de decisión que clasifica los casos de ejecución del proceso en base a una serie de clases previamente definidas, lo cual nos permitirá identificar qué conjunto de reglas nos lleva a cada resultado del proceso. Un ejemplo del tipo de reglas (en lenguaje natural) que se pretende alcanzar sería el siguiente: *Cuando actividad$_A$ va seguida de actividad$_B$ y el tiempo transcurrido entre actividad$_A$ y actividad$_B$ es inferior a 2 minutos, el proceso finaliza con éxito.*

3. Validación de la aproximación

Para validar la aproximación se ha decidido utilizar uno de los conjuntos de datos sintéticos generados en [6], concretamente el correspondiente a *Running Example*, cuyos casos del proceso se intentarán clasificar en base a varios criterios. Cabe destacar que en [6] se extraen las correlaciones entre los atributos de las actividades que satisfacen o no una determinada restricción de DECLARE *previamente dada*; mientras que en nuestra aproximación lo que interesa es extraer aquel conjunto de reglas que mejor clasifique las trazas (casos) del proceso en base a una clase previamente asignada a cada una de ella (que vendrá motivada por el resultado final que se desee analizar).

Los resultados de la validación muestran que se han obtenido las mismas reglas que las generadas por [6], haciendo una asignación de clases adecuada a

los datos sobre los que ha aplicado nuestra aproximación. En las Tablas 2 a 5 se presentan varios ejemplos de reglas extraídas del conjunto de datos, donde se muestran tanto su *antecedente* como su *consecuente* (*clase*), así como su *soporte* y su *confianza*.

Ahora bien, Un análisis más detallado de los resultados indica que, si bien las reglas extraídas por ambas aproximaciones describen correctamente las restricciones que contiene el registro de eventos, en nuestra aproximación aparece la restricción *Existence* en todos los antecedentes de las reglas extraídas, mientras que en [6] también encontramos *Response*, *Responded Existence* o *Chain Response*.

Esto se debe a que en [6] se proporcionan previamente las plantillas de *DECLARE* que se desean tener en cuenta para extraer las correlaciones entre los atributos, mientras que en nuestra aproximación se prueban todas las combinaciones posibles que ofrece el lenguaje para seleccionar la que mejores resultados proporcione en cada iteración del algoritmo. Esta es la principal diferencia entre ambas propuestas: en nuestra aproximación, la búsqueda está guiada por la consulta que el usuario desea resolver, es decir, qué reglas han llevado a que el caso se ejecute de una determina forma.

Teniendo esto en cuenta, dada la simplicidad de las restricciones definidas en el conjunto de datos utilizado, siempre se cumple la restricción de presencia de la actividad que activa cada restricción (*Existence*) y, además, la ganancia de información que se alcanza con ella no es mejorada por otras restricciones de DECLARE (*Response*, *ChainResponse*, etc.). En esta situación, como la comparación que se hace entre la ganancia de información a la hora de escoger una característica de entrada frente a otra es estricta ($>$), la variable de *Existence(A)* termina siendo la que devuelve nuestra aproximación.

4. Conclusiones

En este artículo se ha presentado una aproximación inicial para la extracción de conocimiento a partir de registros de eventos, siguiendo las restricciones del lenguaje MP-DECLARE. Esta aproximación está basada en la aplicación de árboles de decisión, que tiene como entrada una serie de autómatas generados a partir de las restricciones del lenguaje MP-DECLARE. Los resultados del trabajo indican que la técnica propuesta es capaz de extraer todas las restricciones presentes en un conjunto de datos sintético que ha sido utilizado por uno de los trabajos de referencia del campo.

Como trabajo futuro, se aplicarán optimizaciones para mejorar el rendimiento del algoritmo y se incluirán de un mayor número de combinaciones entre atributos que permitan el descubrimiento de relaciones que a priori no son fáciles de detectar en los datos. Además, también se aplicará el algoritmo sobre conjuntos de datos reales con una mayor complejidad que la mostrada en la fase de validación que se ha tratado previamente.

ID	***Antecedente***
A1	$Existence(Submit\ Loan\ Application)\ \&\ Salary > 70\,000$
A2	$Existence(Submit\ Loan\ Application)\ \&\ Salary > 12\,000$

ID	***Regla***	***Clase***	***Soporte***	***Confianza***
R1	$A1$	$Result = Accepted$	*0.09*	*1.0*
R2	$\neg A1$	$Result \neq Accepted$	*0.91*	*0.67*
R3	$\neg A1\ \&\ \neg A2$	$Result = Rejected$	*0.12*	*1.0*

Tabla 2. Reglas extraídas en base al valor del atributo *Result* de la actividad *Notify Outcome*.

ID	***Antecedente***
A3	$Existence(Check\ Career)\ \&\ Coverage > 5$

ID	***Regla***	***Clase***	***Soporte***	***Confianza***
R4	$A3$	$Cost > 100$	*0.59*	*1.0*
R5	$\neg A3$	$Cost \leq 100$	*0.41*	*1.0*

Tabla 3. Reglas extraídas en base al valor del atributo *Cost* de la actividad *Check Medical History*.

ID	***Antecedente***
A4	$Existence(Assess\ Application)\ \&\ AssessmentCost > 100$

ID	***Regla***	***Clase***	***Soporte***	***Confianza***
R6	$A4$	$Coverage \geq 16$	*0.44*	*1.0*
R7	$\neg A4$	$Coverage < 16$	*0.56*	*1.0*

Tabla 4. Reglas extraídas en base al valor del atributo *Coverage* de la actividad *Check Career*.

ID	*Antecedente*
A5	$Existence(Submit\ Loan\ Application)$ & $Amount > 99\,000$
A6	$Existence(Submit\ Loan\ Application)$ & $Salary > 24\,000$
A7	$Existence(Submit\ Loan\ Application)$ & $Amount > 50\,000$

ID	*Regla*	*Clase*	*Soporte*	*Confianza*
R8	$A5$	$AssessmentCost \geq 110$	*0.28*	*1.0*
R9	$\neg A5$ & $A6$	$AssessmentCost < 110$	*0.55*	*1.0*
R10	$\neg A5$ & $\neg A6$ & $A7$	$AssessmentCost \geq 110$	*0.16*	*1.0*
R11	$\neg A5$ & $\neg A6$ & $\neg A7$	$AssessmentCost < 110$	*0.01*	*1.0*

Tabla 5. Reglas extraídas en base al valor del atributo *Assessment Cost* de la actividad *Assess Application*.

Referencias

1. Aalst, W. van der, *y col.*: Process Mining Manifesto. En: Daniel, F., Barkaoui, K., y Dustdar, S. (eds.) Business Process Management Workshops, págs. 169-194. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
2. Augusto, A., *y col.*: Automated Discovery of Process Models from Event Logs: Review and Benchmark. IEEE Trans. Knowl. Data Eng. 31(4), 686-705 (2019)
3. Maggi, F.M., Mooij, A.J., y Aalst, W.M. van der: User-guided discovery of declarative process models. En: 2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), págs. 192-199 (2011). DOI: 10.1109/CIDM.2011.5949297
4. Maggi, F.M.: Declarative Process Mining. En: Encyclopedia of Big Data Technologies. Ed. por S. Sakr y A.Y. Zomaya. Springer (2019)
5. Burattin, A., Maggi, F.M., y Sperduti, A.: Conformance checking based on multi-perspective declarative process models. Expert Systems with Applications 65, 194-211 (2016). DOI: https://doi.org/10.1016/j.eswa.2016.08.040
6. Leno, V., Dumas, M., Maggi, F.M., La Rosa, M., y Polyvyanyy, A.: Automated discovery of declarative process models with correlated data conditions. Information Systems 89, 101482 (2020). DOI: https://doi.org/10.1016/j.is.2019.101482
7. Breiman, L..: Classification and Regression Trees. Taylor & Francis Ltd (1984)
8. De Giacomo, G., De Masellis, R., y Montali, M.: Reasoning on LTL on finite traces: Insensitivity to infiniteness. En: Proceedings of the AAAI Conference on Artificial Intelligence (2014)