REVISTA CUBANA
DE TRANSFORMACIÓN DIGITAL

RNPS 2487 • ISSN 2708-3411
Vol. 2 • Nro. 1 • enero-marzo 2021 • págs. 130-143

ARTÍCULO ORIGINAL

Hacia la democratizacióndel aprendizaje de máquinas usando AutoGOAL

Towards the Democratization of Machine Learning using AutoGOAL

Ernesto Luis Estevanell-Valladares
eestevanell@matcom.uh.cu • https://orcid.org/0000-0002-1168-1767

Suilan Estevez-Velarde
sestevez@matcom.uh.cu • https://orcid.org/0000-0001-6707-1442

Alejandro Piad-Morffis
apiad@matcom.uh.cu • https://orcid.org/0000-0001-9522-3239

FACULTAD DE MATEMÁTICA Y COMPUTACIÓN, UNIVERSIDAD DE LA HABANA, CUBA

Yoan Gutiérrez
y.gutierrez@dlsi.ua.es • https://orcid.org/0000-0002-4052-7427

Andrés Montoyo
montoyo@dlsi.ua.es • https://orcid.org/0000-0002-3076-0890

DEPARTAMENTO DE LENGUAJES Y SISTEMAS COMPUTACIONALES, UNIVERSIDAD DE ALICANTE, ESPAÑA

Yudivian Almeida-Cruz
yudy@matcom.uh.cu • https://orcid.org/0000-0002-2345-1387

FACULTAD DE MATEMÁTICA Y COMPUTACIÓN, UNIVERSIDAD DE LA HABANA, CUBA

Recibido: 2020-11-16 • Aceptado: 2021-01-19

RESUMEN

El aprendizaje automático es un campo de la inteligencia artificial que ha ganado un reciente interés en todas las áreas de la industria, motivado fundamentalmente por el acelerado crecimiento de las capacidades de cómputo y la disponibilidad de datos. Sin embargo, una de las principales dificultades para su aplicación es la necesidad de expertos que conozcan los detalles internos de los múltiples modelos que pueden ser utilizados. En este contexto ha surgido un nuevo campo de estudio, denominado *AutoML* (*Automated Machine Learning),* que facilita la utilización de estas técnicas por expertos de otros dominios. Este artículo presenta una propuesta concreta de un sistema —*AutoGOAL*— que ha sido diseñada para resolver problemas de aprendizaje automático de variada naturaleza. Además, se realiza una breve comparación entre sistemas existentes de relevancia en el campo. La propuesta es competitiva con herramientas del estado del

arte en problemas clásicos de aprendizaje, a la vez que puede desplegarse, sin esfuerzo adicional, en dominios más complejos, como el procesamiento de lenguaje natural. *AutoGOAL* constituye un paso más hacia la democratización del aprendizaje automático para usuarios no expertos en el tema.

Palabras clave: Aprendizaje de máquinas; aprendizaje automatizado; *AutoML*; inteligencia artificial.

ABSTRACT

Machine Learning is a field of Artificial Intelligence that has gained recent interest in all areas of the industry, motivated primarily by the accelerated growth of computer capabilities and data availability. However, one of the main difficulties for its application is the need for experts who know the internal details of the multiple models that can be used. In this context, a new field of study has emerged, AutoML (Automated Machine Learning), which facilitates the use of these techniques by experts from other domains. This paper presents a concrete proposal of a system —AutoGOAL— which has been designed to solve machine learning problems of various kinds. In addition, a brief comparison is made between relevant existing systems in the field. The proposal is competitive with state-of-the-art tools in classic machine learning problems, and it can be seamlessly deployed in more complex domains, such as natural language processing. AutoGOAL is another step towards the democratization of machine learning for non-expert users.

Keywords: *Artificial intelligence; AutoML; automated learning; machine learning.*

INTRODUCCIÓN

En la última década, el campo del aprendizaje automático (*machine learning,* ML) ha crecido en popularidad, lo cual ha permitido avances en la industria (Brunton, *et al.,* 2020; Wuest, *et al.,* 2016), comercio (Oliver, 1996; Ballestar, *et al.,* 2019), medicina (Bhardwaj, *et al.,* 2017), y otras esferas. Aplicar estas tecnologías a problemas particulares requiere especialistas que tomen decisiones de diseño sobre los algoritmos a utilizar y sepan cómo ajustar sus hiperparámetros[1]. El proceso de desarrollo de una aplicación de aprendizaje automático conlleva un

[1] Valores que guían el proceso de aprendizaje. Generalmente deben ser ajustados por el experto.

elevado costo de experimentación para encontrar la solución más efectiva. Teniendo en cuenta esto, el desarrollo de aplicaciones de minería de datos que utilizan algoritmos de ML para los pasos de modelación, contemplados en metodologías como SEMMA o CRISP-DM, (Azevedo y Santos, 2008) requiere de expertos tanto del dominio de la aplicación como de las técnicas de aprendizaje automático. Cada vez se hace más difícil suplir la demanda de personal cualificado con niveles de experiencia necesarios para la creación de dichas aplicaciones.

Estudios recientes han encontrado que hasta un 86 % de las empresas no están listos para hacer frente a la creciente cantidad de datos (Mendoza, 2020). Un lento desarrollo o procesamiento de algoritmos de ML, falta de recursos computacionales, o la necesidad de personal cualificado en ciencia de datos o aprendizaje automático, son causas potenciales a dicho problema. El *AutoML* (Hutter, *et al.,* 2018) (*Automated Machine Learning)* es un campo de investigación que persigue la automatización incremental de todas las fases del desarrollo de aplicaciones de aprendizaje automático (Olson, *et al.,* 2016). A diferencia del proceso de diseño manual, el *AutoML* permite explorar inteligentemente las mejores combinaciones de algoritmos e hiperparámetros para la construcción de flujos de aprendizaje automático. Esto permite evaluar una mayor cantidad de soluciones de las que usualmente son analizadas por los investigadores, basándose en experiencia previa y conocimiento sobre el problema.

El *AutoML* promete acelerar los procesos de desarrollo e investigación en el aprendizaje automático ofreciendo diversas ventajas. Gracias a una exploración automática, los investigadores podrían centrarse en otras decisiones de diseño más importantes. Además, se pudieran descubrir nuevas combinaciones no analizadas con anterioridad potencialmente más efectivas. Igualmente, la exploración podría aportar datos estadísticos, proporcionando conocimientos útiles sobre el problema en sí mismo.

Este artículo presenta una breve comparación entre herramientas existentes relevantes en el dominio. Además, se propone la herramienta *AutoGOAL*, diseñada para optimizar flujos de algoritmos de aprendizaje automático en problemas de variada naturaleza. El sistema permite a usuarios, tanto expertos como no expertos[2] en el dominio, utilizar herramientas de ML para resolver problemas de múltiples dominios (p.e. PLN[3], clasificación, *clustering).* Esto constituye un paso más hacia la democratización del aprendizaje automático, brindando técnicas de ML a un nuevo sector de usuarios. *AutoGOAL* se encuentra en continuo desarrollo, y se encuentra disponible para la comunidad científica mediante una licencia de código abierto[4].

El contenido del artículo se encuentra organizado en varias secciones. En la sección *Metodología* se analiza una serie de técnicas de *AutoML* que forman parte del estado del arte, y se presenta un marco experimental para la evaluación del sistema propuesto. La sección *AutoGOAL* explica la estructura y diversos detalles de implementación y características de la herramienta. Los resultados obtenidos son estudiados en la sección *Resultados y Discusión.* Al final del documento se ofrecen las *Conclusiones* de la investigación.

[2] En este documento, refiere a un usuario no experto en aprendizaje automático.
[3] Procesamiento de lenguaje natural.
[4] http://bit.ly/3qaEyJf

METODOLOGÍA

Debido al creciente potencial del *AutoML* se hace necesario un análisis adecuado de distintas herramientas disponibles en este campo. En esta investigación se realiza una comparativa entre múltiples propuestas de *AutoML* basada en las características que influyen en la utilidad de cada una para la solución de diversos problemas. Los sistemas de *AutoML* presentados pueden verse como una capa de abstracción encima de una biblioteca de aprendizaje automático. El objetivo de estos sistemas consiste en automatizar las tareas, usualmente realizadas por un experto humano, de seleccionar el mejor algoritmo y ajustar sus parámetros, según el problema que se desea resolver.

Los sistemas de *AutoML* se diferencian en aspectos fundamentales, tales como el tipo y la variedad de los algoritmos sobre los cuáles se realiza la búsqueda, generalmente determinado por la biblioteca de aprendizaje automático usada como base, por ejemplo, *scikit-learn* (Pedregosa, *et al.,* 2011) o keras (Chollet, *et al.,* 2015). Además, algunos sistemas restringen el tipo de problemas a que se pueden aplicados (por ejemplo, solo a datos tabulares), mientras que otros pueden adaptarse a dominios variados, como reconocimiento de imágenes y procesamiento de lenguaje natural. Para ilustrar estas diferencias, la tabla 1 muestra una comparación entre herramientas de *AutoML* relevantes en el campo, se tienen en cuenta diversas características de interés en los sistemas. Aunque existe una gran cantidad de herramientas de *AutoML*[5], en este documento se analiza una selección entre todas las existentes.

Tabla 1. Comparación de características relevantes entre varios sistemas de AutoML

	Lenguaje	Bibliotecas	Modelos de ML	Extensibilidad	Espacio de búsqueda
AutoSklearn	Python	Scikit-learn	Clásico[6]	Automática	Fijo[7]
KNIME[8]	Java	XGBoost, Keras, ~	Clásico, NN[9]	~	Fijo
AutoKeras	Python	Keras	NN	Manual	Fijo
AutoWeka	Java	Weka	Clásico	Automática	Fijo
Hyperopt-Sklearn	Python	Scikit-learn	Clásico	Automática	Fijo
TPOT	Python	Scikit-learn, XGBoost	Clásico	Manual	Fijo
RECIPE	Python	Scikit-learn	Clásico	Manual	Fijo
AutoGOAL	Python	Scikit-learn, Nltk, Keras, Gensim, Pytorch	Clásico, NN, PNL	Automática	Personalizable[10]

Nota: El símbolo "~" indica que no se ofrece más información sobre la característica.

Sistemas como Auto-Sklearn (Feurer, *et al.*, 2015), Auto-Weka (Kotthoff, *et al.*, 2017), Hyperopt-Sklearn (Bergstra, *et al.*, 2015), TPOT (Olson y Moore, 2019), RECIPE (de Sá, *et al.*, 2017)

[5] http://bit.ly/2M78AyQ, http://bit.ly/2XVMYbg
[6] Refiere a modelos para problemas de clasificación, clústering o regresión.
[7] Refiere a un espacio de búsqueda predeterminado e invariante para problemas de un mismo tipo.
[8] https://kni.me/c/33fQGaQzuZByy6hE.
[9] Refiere a redes neuronales (Neural Networks, NN).
[10] Refiere a un espacio de búsqueda modificable y variable para un tipo de problema determinado.

y *AutoGOAL* (Estévez-Velarde, *et al.*, 2020a), tienen como objetivo la resolución del problema del *AutoML* como selección de modelos y optimización de hiperparámetros. Muchas técnicas utilizadas por estos pueden ser recombinadas para alcanzar nuevas propuestas o ayudar a otras existentes a la hora de afrontar problemas más complicados. Si se sigue esta idea, el rendimiento de estos sistemas, en un problema determinado, depende en gran medida de las técnicas del dominio que están presentes entre sus componentes (i.e. en un problema de detección de emociones en *tweets* las técnicas de preprocesamiento de texto juegan un papel fundamental). Por esta razón, el soporte nativo de múltiples bibliotecas de ML se vuelve crucial. *Auto-Sklearn, Hyperopt-Sklearn* y *RECIPE* utilizan como base la biblioteca *Scikit-Learn* mientras que *TPOT* y *AutoGOAL* recogen algoritmos de distintas bibliotecas de ML. Sistemas como *AutoGOAL* y *Knime* permiten, además, optimizar arquitecturas de redes neuronales.

Construir una herramienta capaz de ser desplegada con facilidad en distintas situaciones constituye un paso más hacia el objetivo del *AutoML*. Sistemas como *Auto-Sklearn, TPOT, Knime* y *Auto-Weka* se centran en generar flujos de clasificación mientras que *RECIPE* permite modelar problemas de clasificación, clustering o regresión sin modificaciones importantes al sistema. En cambio, *Auto-Keras* y *AutoGOAL* pueden, además, resolver problemas de PLN, este último permite entrada de datos en forma de texto.

Auto-Sklearn, Auto-Weka e *Hyperopt-Sklearn* integran completamente las APIs de sus bibliotecas base, permiten que cualquier adición de un algoritmo en la biblioteca pueda ser automáticamente contemplable por el sistema. *Auto-Sklearn* y *Hyperopt* se benefician particularmente de encontrarse implementados en *Python* y utilizar *Scikit-Learn.* Esta biblioteca presenta un crecimiento rápido en popularidad y, por lo tanto, es probable la adición y mejora de algoritmos en un futuro, que serán contemplados por los sistemas. Por otra parte, *TPOT* y *RECIPE* utilizan implementaciones directas de métodos de *Scikit-Learn,* y la adición de una nueva componente requiere hacer modificaciones al sistema. *AutoGOAL*, a diferencia de los anteriores, posee un proceso de introspección de código[11] que se encarga de explorar las bibliotecas base en búsqueda de algoritmos. Además, ofrece una API que permite a cualquier usuario la adición de nuevas componentes personalizadas.

Los sistemas de *AutoML* más utilizados presentan restricciones para ser aplicados a los dominios más interesantes. Algunos de ellos están limitados por las bibliotecas de aprendizaje automático que utilizan y otros por los tipos de problemas a los que se orientan. *AutoGOAL* es una propuesta de sistema de *AutoML* más general, que ha sido diseñado para eliminar estas restricciones. En cambio, debido a su generalidad, el sistema puede potencialmente presentar menos rendimiento en tareas de propósito específico, comparándose con técnicas del dominio.

Para evaluar la flexibilidad, adaptabilidad y rendimiento de la propuesta se presenta un marco experimental compuesto de tareas con distinta complejidad. La metodologí a de com-

[11] Proceso de análisis de código encargado de explorar bibliotecas de ML para la identificación de clases de algoritmos, además de los rangos de hiperparámetros a partir de la documentación ofrecida.

paración más usual en el campo de *AutoML* consiste en ejecutar el proceso de optimización durante un tiempo determinado y reportar la evaluación del mejor flujo obtenido. En esta investigación se compara *AutoGOAL* con seis herramientas de *AutoML* estado-del-arte distintas en siete conjuntos de datos diferentes; se informa la media de precisión obtenida en 20 ejecuciones de una hora cada una para cada herramienta. Además, *AutoGOAL* se evalúa en dos problemas adicionales de procesamiento de lenguaje natural en los cuáles las herramientas de *AutoML* existentes no pueden ser aplicadas.

AUTOGOAL

AutoGOAL ha sido diseñado para permitir a los usuarios, tanto expertos como no expertos, reducir la complejidad de diseñar flujos de algoritmos de aprendizaje automático a través de una serie de herramientas, algoritmos pre-definidos, y protocolos extensibles. Para usuarios no expertos en ML, el sistema ofrece una API de alto nivel intuitiva, mientras que usuarios con mayor conocimiento en el área pueden sacar más provecho de la API de bajo nivel. La figura 1 ilustra los componentes más relevantes de *AutoGOAL,* desde la API de alto nivel hasta la implementación de adaptadores de algoritmos existentes y las interfaces a recursos externos y bibliotecas de *back-end.* La herramienta se encuentra disponible bajo una licencia de código abierto.

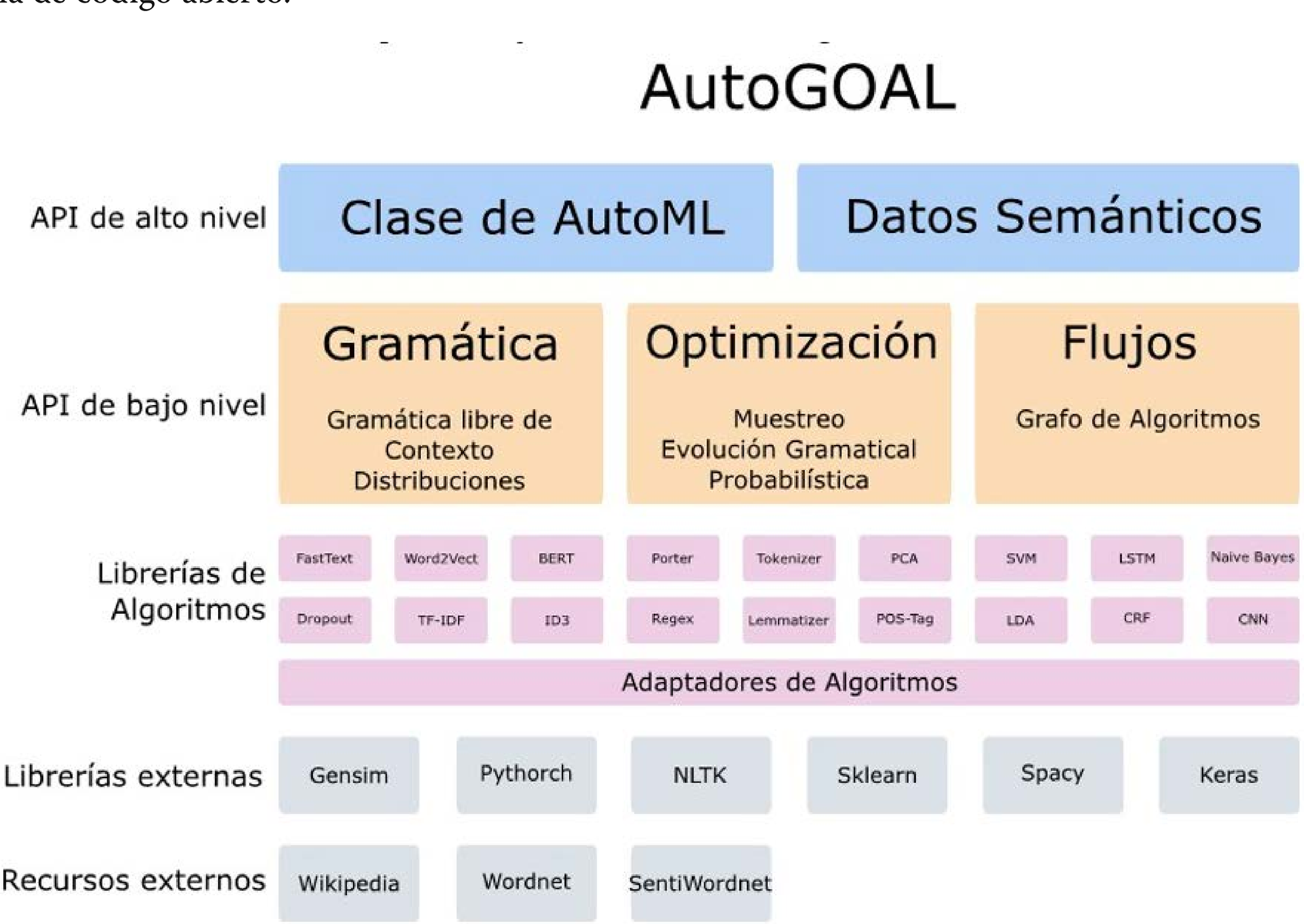

Figura 1. Esquema de arquitectura de la biblioteca AutoGOAL.

El núcleo de la biblioteca *AutoGOAL* es la API de bajo nivel, compuesta de los siguientes elementos: un módulo de gramáticas probabilísticas libres del contexto (*Grammar);* un módulo de muestreo y optimización; un módulo de descubrimiento de flujos (*Pipelines).* Esta API de bajo nivel permite a los usuarios: proporcionar sus propias implementaciones de algoritmos de aprendizaje automático, declarar los elementos que se deben optimizar (por ejemplo, hiperparámetros), y definir cómo se pueden conectar en flujos complejos de varios pasos.

El módulo *Grammar* proporciona un conjunto de anotaciones de tipo que se utilizan para definir el espacio de hiperparámetros de una técnica o algoritmo arbitrario. Cada técnica se representa como una clase de *Python,* y los hiperparámetros correspondientes se representan como argumentos anotados del método *__init__*, ya sea valores primitivos (por ejemplo, numéricos, texto, etc.) o instancias de otras clases, anotadas recursivamente. Dada una colección de clases anotadas, este módulo infiere automáticamente una gramática libre de contexto (Hopcroft y Ullman, 1979) que describe el espacio de todas las instancias posibles de esas clases.

El módulo *Optimization* proporciona estrategias de muestreo sobre una gramática libre del contexto que construye recursivamente una instancia específica basada en las anotaciones. Se implementan dos estrategias de optimización: búsqueda aleatoria y evolución gramatical probabilística (Kim y Ahn, 2015). Esta última realiza un ciclo de muestreo/actualización que selecciona las instancias de mejor rendimiento de acuerdo con alguna métrica predefinida (p.e., precisión) y actualiza iterativamente el modelo probabilístico interno del algoritmo de muestreo. Al utilizar las probabilidades, esta estrategia permite realizar una búsqueda en un espacio de tamaño exponencial (no todas las combinaciones de algoritmos e hiperparámetros son examinadas), a costo de ofrecer una solución no necesariamente óptima.

El módulo *Pipelines* proporciona una abstracción para que los algoritmos se comuniquen entre sí a través de un patrón *Facade,* es decir, la implementación de un método *run* con anotaciones para los tipos de entrada y salida. Las clases que implementan este patrón se conectan automáticamente en un grafo de algoritmos, en el que cada ruta representa un posible flujo para resolver un problema, especificado por los tipos de datos de entrada y salida. La API de alto nivel se basa en estos módulos, y proporciona la clase *AutoML* y los tipos de datos semánticos. Los usuarios de la biblioteca pueden interactuar con la API de alto nivel de forma transparente, o interactuar directamente con los componentes internos de bajo nivel para tener mayor control sobre las soluciones. *AutoGOAL* también proporciona una biblioteca de algoritmos con adaptadores pre-definidos para tecnologías de aprendizaje automático disponibles en bibliotecas y recursos externos. Al momento de la escritura de este artículo, en su versión 0.3.2, el sistema cuenta con adaptadores para un total de 133 algoritmos de siete bibliotecas externas, varios de los cuales se crean semi-automáticamente mediante introspección de código, y el resto se agregan manualmente por los desarrolladores de la biblioteca. Esta biblioteca está en continuo desarrollo.

AutoGOAL se puede instalar como un paquete de *Python* independientemente de cualquier biblioteca de aprendizaje automático. En este caso funciona como una herramienta li-

gera que proporciona todos los bloques de construcción, pero ninguno de los adaptadores predefinidos. Los usuarios pueden instalar opcionalmente cualquiera de las bibliotecas externas compatibles. *AutoGOAL* la descubrirá automáticamente y registrará los adaptadores correspondientes, que estarán disponibles para su uso mediante la API de alto nivel. Además, se proporciona una imagen de Docker con todas las dependencias opcionales y bibliotecas externas ya instaladas[12].

Interfaz de alto nivel

Esta API permite utilizar *AutoGOAL* como un algoritmo de clasificación o regresión de caja negra con una interfaz similar a la biblioteca *scikit-learn.* Detrás de esta interfaz, se realiza un proceso completo que incluye preprocesamiento, selección de características, reducción de dimensionalidad y aprendizaje. El usuario debe definir un conjunto de datos de entrenamiento y evaluación, una métrica[13] para optimizar y el tipo de datos de entrada y salida. En muchos casos, *AutoGOAL* puede inferir automáticamente el tipo de entrada y salida del conjunto de datos. Los tipos de datos pueden variar desde tabulares a tipos más complejos, como imágenes, texto en lenguaje natural con diferentes estructuras semánticas y combinaciones de estos. La figura 2 muestra un ejemplo de código fuente, específicamente en el contexto de un problema de clasificación de texto.

```
from autogoal.ml import AutoML
from autogoal.datasets import haha
from autogoal.kb import List, Sentence, CategoricalVector

automl = AutoML(
    input=List(Sentences()),    # tipos de entrada
    output=CategoricalVector() # y salida
)

X, y = haha.load() # cargar datos del dominio especifico
automl.fit(X, y)   # ejecutar optimizacion
```

Figura 2. Ejemplo de código fuente para ejecutar AutoGOAL en un conjunto de datos específico.

Interfaz de bajo nivel

Esta API está diseñada para usuarios con más experiencia que necesitan control sobre el proceso de *AutoML.* Para este tipo de usuario, *AutoGOAL* proporciona un lenguaje simple para definir una gramática que describe el espacio de la solución. Esto se realiza utilizando un enfoque orientado a objetos donde el usuario define una clase de *Python* para cada componente de la solución (por ejemplo, cada algoritmo). Los parámetros del constructor de estas clases se

[12] http://dockr.ly/3qa3UXJ
[13] Pueden utilizarse tanto métricas de enfoque supervisado (p.e. precisión, F1) como métricas de aprendizaje no supervisado (p.e. coeficiente de silueta, índice de Dunn).

anotan con atributos que describen el espacio de valores posibles, que pueden ser tipos básicos (es decir, numéricos, texto, etc.) e instancias de otras clases, recursivamente. En base a las anotaciones, *AutoGOAL* puede construir automáticamente todas las formas posibles en las que se pueden instanciar las clases del usuario.

La API de bajo nivel proporciona utilidades para anotar los tipos de parámetros en constructores de clase y métodos, lo que indica el rango válido para sus valores. El usuario define un adaptador para un componente de una biblioteca externa, que sea compatible con la API de *AutoGOAL,* por ejemplo, un algoritmo de *scikit-learn.* Como ejemplo, la figura 3 muestra la definición de clases que envuelven algoritmos *scikit-learn* y definen el espacio de búsqueda de hiperparámetros al anotar los parámetros de interés en el constructor de la clase. Además, cada clase debe definir un método run cuyos parámetros de entrada y salida se anotan con tipos semánticos. Esto permite que *AutoGOAL* detecte qué componentes se pueden conectar. Los algoritmos de aprendizaje supervisados de *scikit-learn* reciben como entrada tanto la matriz de características como las clases (durante el entrenamiento). Los métodos run en estas clases actúan como un adaptador entre la API de *AutoGOAL* y la API *scikit-learn.* Una implementación simplificada se muestra en la figura 3.

```
class LR(sklearn.linear_model.LogisticRegression):
    def __init__(
        self,
        penalty: Categorical("l1", "l2"),
        C: Continuous(0.1, 10)
    ):
        super().__init__(penalty=penalty, C=C)

    def run(self, input: Tuple(MatrixContinuous,
                               CategoricalVector))
        -> CategoricalVector:
        if self.training:
            X, y = input
            self.fit(X, y)
            return y
        else:
            return self.predict(X)
```

Figura 2. Ejemplo de código fuente para ejecutar AutoGOAL en un conjunto de datos específico.

Esta API se puede usar en cualquier nivel de detalle. Por ejemplo, de la misma manera puede implementarse un algoritmo para obtener representaciones de *word2vec* de palabras individuales utilizando *gensim.* En este caso el algoritmo se ejecuta a nivel de *token,* a diferencia de los adaptadores de *scikit-learn,* que se ejecutan a nivel del conjunto de datos completo. *AutoGOAL* es capaz automáticamente de combinar algoritmos que funcionan a diferentes niveles de la estructura de los datos. Como ejemplo final, esta API también puede representar técnicas de extracción de características que utilizan recursos externos, como Wikipedia y *WordNet.*

RESULTADOS Y DISCUSIÓN

En esta sección se compara *AutoGOAL* con otros sistemas de *AutoML* estado-del-arte siguiendo la metodología planteada. La tabla 2 muestra los resultados de la comparación.

Tabla 2. Comparación de AutoGOAL y otros sistemas AutoML para nueve conjuntos de datos clásicos de aprendizaje automático en términos de precisión, excepto MEDDOCAN, en el que se utiliza la métrica F_1.

DATASETS	Cars	Credit G.	Abalone	Shuttle	Yeast	Dorothea	Gisette	HAHA	MEDDOCAN
ML-Plan (Weka)	1.27	25.54	73.72	0.01	39.37	6.49	2.92	-	-
Auto-Weka	0.66	26.50	73.46	0.12	39.72	-	3.90	-	-
ML-Plan (Sklearn)	0.34	24.56	73.77	0.02	39.52	8.69	2.76	-	-
Auto-Sklearn-v	1.38	25.95	82.92	0.02	40.51	6.32	2.56	-	-
Auto-Sklearn-we	1.26	25.39	80.59	0.02	38.99	6.06	2.24	-	-
TPOT	0.37	23.91	73.14	0.02	38.47	-	-	-	-
AutoGOAL	0.60	27.01	74.33	0.11	39.94	5.97	2.25	21.1	3.99

Fuente: Tomado de Estévez-Velarde, S., Piad-Morffis, A., Gutiérrez, Y., Montoyo, A., Muñoz-Guillena, R. y Almeida-Cruz, Y. (2020b). Demo Application for the AutoGOAL Framework.

Los resultados experimentales demuestran que las herramientas de *AutoML* existentes se comportan de forma similar en los problemas clásicos. A partir de pruebas estadísticas de comparación de medias, se llegó a la conclusión de que no existen diferencias significativas entre ninguna de las herramientas evaluadas. Esto significa que *AutoGOAL* es competitivo con el estado del arte en los problemas de *AutoML* que dichas herramientas son capaces de solucionar, a la vez que puede aplicarse a otros dominios.

Es importante tener en cuenta que *AutoGOAL* se implementa mediante el uso del mismo código en todos los experimentos (figura 2), varía solo la definición de los tipos de entrada y salida. Por ejemplo, configurar los conjuntos de datos de HAHA y MEDDOCAN tomó aproximadamente una y cuatro horas respectivamente, dedicadas fundamentalmente a preparar los corpus en un formato adecuado. En los conjuntos de datos UCI, *AutoGOAL* converge rápidamente a una combinación de clasificadores simples y métodos no supervisados (reducción de dimensionalidad, selección de características, entre otros). Sin embargo, en HAHA y MEDDOCAN, los clasificadores simples se descartan en favor de las técnicas de PLN y redes neuronales. En el caso de MEDDOCAN, *AutoGOAL* pudo detectar que las tecnologías de PLN son cruciales en este dominio dado el uso de la terminología médica, mientras que en el caso de HAHA, *AutoGOAL* identificó los *embeddings* de propósito general como los más efectivos considerando el contenido coloquial. Esto proporciona evidencia de que, durante la optimización, *AutoGOAL* aprende automáticamente qué familias de algoritmos son las más adecuadas para cada problema.

En el campo de *AutoML* existe una variedad considerable de herramientas y sistemas para diversos propósitos. Algunos sistemas, como *auto-sklearn* y *auto-keras,* han sido diseñados específicamente para automatizar el desarrollo de soluciones basadas en una biblioteca con-

creta, y por este motivo, son más efectivos en el dominio específico en que se enfocan. Otros sistemas, como TPOT, aportan una mayor generalidad en los tipos de algoritmos disponibles, por lo que pueden explotar menos el diseño específico de cada biblioteca. En *AutoGOAL* se prioriza la generalización y la capacidad de adaptar nuevos algoritmos por encima de la especialización en una u otra biblioteca concreta, y por tanto es menos efectivo si se desea optimizar a fondo una clase específica de algoritmos (p.e., arquitecturas de redes neuronales).

Al abordar el problema de *AutoML* desde la perspectiva de su modelo de optimización subyacente, se hacen evidentes dos consideraciones interesantes: la naturaleza jerárquica de las soluciones y la naturaleza multiobjetivo de la métrica de rendimiento. La propuesta de esta investigación representa explícitamente soluciones como un proceso de decisión jerárquico, de la misma manera que otros enfoques alternativos de *AutoML* (p.e., *RECIPE).* Esto permite modelar y tener en cuenta el hecho de que algunas decisiones tienen una mayor influencia en la optimización de los flujos. Por ejemplo, la selección entre clasificadores lineales, no lineales o basados en árboles es más importante que la selección de valores específicos de sus hiperparámetros. A medida que los flujos de aprendizaje automático crecen en complejidad e involucran algoritmos de diferentes bibliotecas y tecnologías, una conceptualización jerárquica, como la propuesta en esta investigación, será aún más relevante.

Con respecto a la métrica de rendimiento, la mayoría de los enfoques actuales de *AutoML* se centran en una única función objetivo, por ejemplo, precisión, recobrado o *F*1, en correspondencia con el problema de aprendizaje automático en cuestión. Sin embargo, en escenarios prácticos, puede ser necesario equilibrar diferentes métricas de rendimiento, incluido también el uso del tiempo y la memoria, y cualidades más subjetivas como la interpretabilidad de los modelos o su capacidad para lidiar con datos sesgados. El enfoque de optimizar una métrica principal sujeta a restricciones de tiempo y memoria es insuficiente en un escenario en el que el usuario final tiene que decidir sobre cuestiones prácticas como el despliegue de estos flujos en un sistema de producción. A modo de ejemplo, TPOT considera este problema desde el enfoque multi-objetivo mediante la optimización conjunta de la precisión y la complejidad del modelo (en términos de longitud de los flujos). Sin embargo, los enfoques futuros deberán considerar este problema en profundidad, lo que podría incluir al usuario final en el ciclo de evaluación de los flujos de algoritmos.

Durante el proceso de optimización en *AutoGOAL*, se genera una cantidad significativa de datos en cada iteración sobre los mejores flujos y sus características. En principio, esta información se utiliza para actualizar el modelo probabilístico de forma que los nuevos flujos muestreados sean similares a los mejores flujos ya evaluados. Además, esta información puede proporcionar conocimiento adicional cuando se agrega para todo el proceso de optimización, analizando las características (es decir, las producciones de la gramática) que aparecen constantemente en los mejores flujos. Una tendencia reciente en el área de *AutoML* es incluir técnicas de meta-aprendizaje (*meta-learning)* que permitan reaprovechar la experiencia obtenida en evaluaciones anteriores en problemas similares para condicionar los algoritmos de optimización a mejores regiones del espacio de búsqueda desde el inicio. Los datos obtenidos

por *AutoGOAL* durante la optimización son una fuente valuable de información para el futuro desarrollo de este tipo de técnicas.

CONCLUSIONES

La democratización de la inteligencia artificial es una de las preocupaciones fundamentales, tanto de la comunidad científica como de los expertos de la industria. El campo del *AutoML* se presenta como una alternativa prometedora para disminuir substancialmente el esfuerzo que conlleva la aplicación de técnicas de inteligencia artificial, y específicamente de aprendizaje automático, a problemas concretos. Aunque existen varias herramientas de *AutoML* que han sido exitosas en la resolución de problemas específicos de inteligencia artificial, estas herramientas son aún demasiado rígidas para ser utilizadas en problemas prácticos que requieren la combinación de algoritmos y tecnologías de diferente naturaleza.

Este trabajo propone una nueva formulación de *AutoML*, más completa y flexible, que puede ser adaptada a un amplio rango de problemas de aprendizaje automático. Para lograrlo, se define el problema de *AutoML* heterogéneo, un marco conceptual que abarca los problemas usualmente considerados en el *AutoML* clásico y escenarios novedosos como el procesamiento de lenguaje natural y el descubrimiento de conocimientos. Se propone la herramienta *AutoGOAL*, un sistema de *AutoML* heterogéneo que genera flujos complejos, y puede ser aplicado a un conjunto amplio de escenarios de machine learning. El despliegue del sistema, para un problema determinado, es relativamente sencillo, solo requiere de la definición de los tipos de entrada y salida correspondientes. De esta manera, la propuesta se hace accesible incluso para profesionales sin conocimiento específico de ML. Una versión gratuita de *AutoGOAL* se encuentra disponible para la comunidad científica.

Esta propuesta presenta un potencial para proveer de los expertos de prototipos iniciales (modelos preconstruidos) en diversas áreas del aprendizaje automático y, por lo tanto, de ser una herramienta para ayudar al investigador y disminuir el tiempo de desarrollo. *AutoGOAL* representa un paso más hacia la democratización del proceso de desarrollo de técnicas ante problemas de aprendizaje, permitiendo a usuarios no expertos desarrollar sus soluciones. Se considera que el objetivo no es reemplazar al experto, sino, dotarlo de una herramienta complementaria y traer a la mesa lo mejor de ambos bandos. Al igual que los compiladores trajeron una mejora significativa en la eficiencia del desarrollo de *software,* el campo de *AutoML* promete revolucionar la forma en que se realiza la investigación y la práctica del aprendizaje automático. La facilidad de utilización y el nivel de flexibilidad presentado por *AutoGOAL* proponen un enfoque interesante a la industria: la obtención de usuarios con menos preparación y, por tanto, la generalización del uso del aprendizaje de máquinas.

REFERENCIAS

Azevedo, A. & Santos, M. F. (2008). KDD, SEMMA y CRISP-DM: a parallel overview. En A. Abraham (ed.), *IADIS European Conf. Data Mining* (pp. 182-185). IADIS. ISBN: 978-972-8924-63-8

Ballestar, M. T., Grau-Carles, P., & Sainz, J. (2019). Predicting customer quality in e-commerce social networks: a machine learning approach. *Review of Managerial Science, 13*(3), 589-603.

Bergstra, J., Komer, B., Eliasmith, C., Yamins, D., & Cox, D. D. (2015). Hyperopt: a Python library for model selection and hyperparameter optimization. *Computational Science & Discovery, 8*(1), 014008. doi: 10.1088/1749-4699/8/1/014008

Bhardwaj, R., Nambiar, A. R., & Dutta, D. (2017). A study of machine learning in healthcare. In *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC),* volume 2, pages 236-241. IEEE. doi: 10.1109/COMPSAC.2017.164

Brunton, S. L., Noack, B. R., y Koumoutsakos, P. (2020). Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics, 52*:477-508. doi: 10.1146/annurev-fluid-010719-060214

Chollet, F., et al. (2015). Keras. Github. https://¬keras.io.

de Sá, A. G., Pinto, W. J. G., Oliveira, L. O. V., & Pappa, G. L. (2017). Recipe: a grammar-based framework for automatically evolving classification pipelines. En *European Conference on Genetic Programming,* (pp. 246-261). Springer. doi: 10.1007/978-3-319-55696-3_16

Estévez-Velarde, S., Gutiérrez, Y., Almeida-Cruz, Y., & Montoyo, A. (2020a). General-purpose hierarchical optimisation of machine learning pipelines with grammatical evolution. *Information Sciences, 543,* 58-71. doi: 10.1016/j.ins.2020.07.035

Estévez-Velarde, S., Piad-Morffis, A., Gutiérrez, Y., Montoyo, A., Muñoz-Guillena, R. & Almeida-Cruz, Y. (2020b). Demo Application for the AutoGOAL Framework. En Ptaszynski, Michal y Ziolko, Bartosz (editors). *Proceedings of the 28th International Conference on Computational Linguistics: System Demonstrations* (pp. 18-22). Recuperado de http://bit.ly/3sDDeAt.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., & Hutter, F. (2015). Efficient and robust automated machine learning. In *Advances in neural information processing systems, 28,* 2962-2970.

Hopcroft, John E. & Ullman, Jeffrey D., editors (1979). *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 77-106.

Hutter, F., Kotthoff, L., & Vanschoren, J., editors (2018). *Automated Machine Learning: Methods, Systems, Challenges.* Springer. Recuperado de http://link.springer.com/978-3-030-05318-5. doi: 10.1007/978-3-030-05318-5

Kim, H. T. & Ahn, C. W. (2015). A new grammatical evolution based on probabilistic context-free grammar. En *Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems, 2,* pages 1-12. Springer.

Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F., & Leyton-Brown, K. (2017). Auto-weka 2.0:

Automatic model selection and hyperparameter optimization in weka. *The Journal of Machine Learning Research, 18*(1), 826-830.

Mendoza, N. F. (2020). 86% of businesses say they're not ready for the next stage of the Data Age. *TechRepublic.* Recuperado de https://tek.io/3nvzmxP.

Mohr, F., Wever, M., & Hüllermeier, E. (2018). Ml-plan: Automated machine learning via hierarchical planning. *Machine Learning, 107*(8-10), 1495-1515. doi: 10.1007/s10994-018-5735-z

Oliver, J. R. (1996). A machine-learning approach to automated negotiation y prospects for electronic commerce. *Journal of management information systems, 13*(3),83-112. doi: 10.1080/07421222.1996.11518135

Olson, R. S., Bartley, N., Urbanowicz, R. J., & Moore, J. H. (2016). Evaluation of a tree-based pipeline optimization tool for automating data science. En *Proceedings of the Genetic and Evolutionary Computation Conference 2016,* abs/1603.06212, 485-492.

Olson, R. S. & Moore, J. H. (2019). Tpot: A tree-based pipeline optimization tool for automating machine learning. En *Proceedings of the Workshop on Automatic Machine Learning,* en PMLR, 64, 66-74.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12,* 2825-2830.

Wuest, T., Weimer, D., Irgens, C., & Thoben, K.-D. (2016). Machine learning in manufacturing: advantages, challenges, and applications. *Production & Manufacturing Research, 4*(1), 23-45.