

# Arquitectura de un Framework para la Generación Automatizada de Datasets Temporales en Data Lakes.\*

Brial Sal<sup>1</sup>[0000-0002-0275-5082], Alfonso de la Vega<sup>1</sup>[0000-0002-7109-4249],  
Patricia López<sup>1</sup>[0000-0002-9562-6342], Diego García-Saiz<sup>1</sup>[0000-0002-7182-7879],  
Alicia Grande<sup>2</sup>, David López<sup>2</sup>, and Pablo Sánchez<sup>1</sup>[0000-0002-4617-0457]

- <sup>1</sup> Grupo de Ingeniería del Software y Tiempo Real, Universidad de Cantabria, España  
{salb, delavegaa, lopezpa, garciasd, sanchezbp}@unican.es  
<sup>2</sup> LIS Data Solutions, Santander, España  
{alicia.grande, david.lopez}@lisdatasolutions.com

**Resumen** En los últimos años, los *data lakes* se han popularizado como solución para el almacenamiento centralizado de grandes volúmenes de datos heterogéneos procedentes de fuentes dispares. Estos datos suelen tener un marcado carácter temporal, ya que los datos suelen extraerse periódicamente de diversas fuentes a diferentes frecuencias y se almacenan directamente en crudo. Por tanto, estos datos deben ser adecuadamente preprocesados antes de ser consumidos por las aplicaciones que los explotan. Esta tarea de preprocesamiento se realiza actualmente de manera manual, mediante la escritura de *scripts* en lenguajes de transformación de datos. Este proceso es laborioso, costoso y, por lo general, propenso a errores. Para tratar de aliviar este problema, este artículo presenta la arquitectura inicial de *Hannah*, un *framework* que busca automatizar la generación de *datasets* para la minería de series temporales a partir de datos en bruto provenientes de *data lakes*. El objetivo es que, utilizando la menor cantidad de información posible como entrada, el *framework* sea capaz de recuperar los datos requeridos del *data lake* y procesarlos para que encajen adecuadamente dentro de un *dataset*.

**Keywords:** Data Lake · Ingeniería de Datos · Ciencia de Datos · Series Temporales

## 1. Introducción

En la actualidad, los *data lakes* son una solución popular para el almacenamiento y gestión centralizada de grandes volúmenes de datos procedentes de diferentes fuentes heterogéneas [1]. Estos datos son luego consumidos por aplicaciones

\* Trabajo parcialmente financiado por MCIN/AEI/10.13039/501100011033/FEDER “Una manera de hacer Europa” en el proyecto PID2021-124502OB-C42 (PRESE-CREL) y por el proyecto Industrial Smart Data Pool (2021/INN/80), financiado por el Gobierno de Cantabria a través del programa INNOVA

de diversa índole, no conocidas a priori y que pueden variar desde sencillos cuadros de mando hasta complejos sistemas de entrenamiento de redes neuronales. Por ello, los datos se almacenan en crudo, siendo responsabilidad de cada aplicación limpiarlos y preprocesarlos de manera adecuada conforme a sus necesidades.

Este proceso de extracción y preprocesado de datos del *data lake* se realiza actualmente de manera manual, mediante la creación de largos y complejos *scripts* en lenguajes o librerías dedicadas a la manipulación de datos, como *Pandas*. La escritura de *scripts* de preprocesado requiere de sólidos conocimientos de estas herramientas, y, además, suele ser un proceso laborioso, tedioso y, por lo general, tendente a la generación de errores.

Para aliviar este problema, e inspirándonos en trabajos propios previos [3,2], hemos creado en colaboración con la empresa LIS Data Solutions la arquitectura inicial de un *framework* denominado *Hannah* que actualmente está en desarrollo. El objetivo de *Hannah* es automatizar lo máximo posible el proceso de generación de *datasets* para análisis y minería de series temporales de datos. Nos hemos centrado en series temporales ya que los datos que se almacenan dentro de un *data lake* suelen tener un marcado carácter temporal, al ser datos que se extraen de manera periódica de diversas fuentes heterogéneas.

A la hora de generar *datasets* para el análisis de series temporales, debemos solventar una serie de problemas, como son la alineación de las frecuencias de cada variable del *dataset* o la presencia de valores faltantes. Por ejemplo, un proyecto podría querer predecir el volumen del tráfico de una ciudad usando el valor de las precipitaciones en dicha ciudad, el número de turistas recibidos y el tiempo empleado en completar una determinada ruta de reparto. En un *data lake* concreto, los valores de las precipitaciones podrían registrarse por cada hora, el número de turistas por el total recibido en cada semana y el tiempo de la ruta de transporte de manera diaria, exceptuando domingos y festivos.

Por tanto, para construir un *dataset* con estas tres variables necesitaríamos representarlas en una misma frecuencia. En nuestro caso podríamos optar por utilizar, por ejemplo, una frecuencia base diaria, lo que requeriría generar valores diarios para cada una de las variables. Para ello podemos utilizar diversas estrategias. Por ejemplo, se puede obtener un valor de precipitación total diaria sumando las precipitaciones de cada hora de un día. Para los turistas, podríamos simplemente dividir el número de turistas recibidos a la semana por 7, o utilizar técnicas de interpolación de valores más avanzadas.

El objetivo de *Hannah*, el *framework* que estamos desarrollando, es automatizar la mayor parte posible de todo este proceso de transformación, que se ejecutaría de manera transparente a las personas encargadas de producir el *dataset*, y utilizando como entrada la menor cantidad de información posible. Para ello, hemos propuesto la arquitectura que se describe en la siguiente sección.

## 2. Solución Propuesta

La Figura 1 muestra el esquema general de *Hannah*. El *framework* asume la existencia de un *data lake* y de un catálogo de datos que describe los datos

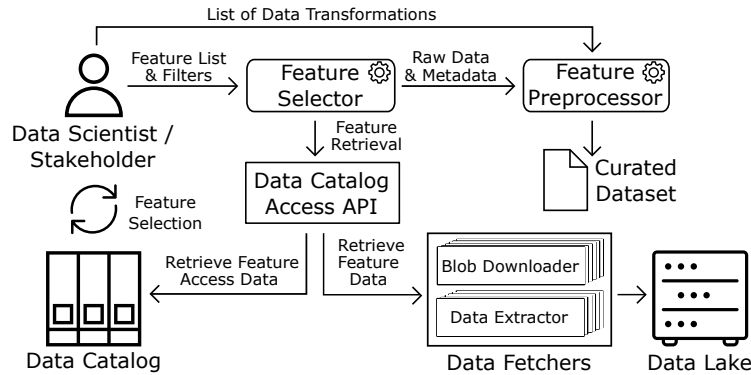


Figura 1. Visión global de *Hannah*.

que contiene dicho *data lake* y especifica dónde se localizan. De esta forma, para generar un *dataset* para análisis de series temporales, los científicos de datos comenzarían navegando por el catálogo de datos para encontrar las variables (*features*) que consideren útiles para dicho análisis.

A continuación, los científicos de datos proporcionarían los identificadores de estas variables al *feature selector*, junto con un conjunto de filtros que indicarían los valores a recuperar. Por ejemplo, se podrían seleccionar las precipitaciones en la ciudad de Santander durante todo el año 2022. Con esta información, el *feature selector* se comunica con la *Data Catalog Access API* para recuperar los datos y metadatos de las variables seleccionadas.

El objetivo de la *Data Catalog Access API* es proporcionar un punto de acceso uniforme para la comunicación con diferentes tecnologías de catálogo de datos y *data lake*, de manera que esta comunicación se realice siempre de la misma forma con independencia de las tecnologías subyacentes empleadas. Además, gracias a esta API, se podrían incorporar nuevos catálogos de datos y *data lakes* a nuestro *framework* siempre y cuando se implemente dicha API para las nuevas tecnologías.

Para recuperar los datos de las variables solicitadas, la *Data Catalog Access API* accede primero al catálogo de datos para obtener las referencias a las ubicaciones donde se almacenan estos datos e invoca a diferentes *data fetchers* para extraer estos datos del *data lake*. Cada *data fetchers* se compone de dos subcomponentes: el *blob downloader* y el *data extractor*. Los primeros obtienen bloques de datos enteros, tal como están almacenados en el *data lake*. Cada bloque de datos puede estar en un formato diferente, como puede ser XML o JSON, entre otros. La obtención de datos concretos dentro de estos bloques de datos se realiza mediante los *data extractors*, que serían capaces, por ejemplo, de recuperar el valor de un nodo específico de un archivo XML.

Tras recuperar los datos solicitados, éstos se proporcionan como entrada, junto con sus potenciales metadatos, al *feature preprocessor*. Este módulo se encarga de transformar los datos para que éstos puedan encajar adecuadamente

en un *dataset*. Para ello, entre otras cuestiones, necesita alinear las frecuencias con las que se ha recogido cada variable. Para ejecutar estas transformaciones, el *feature preprocessor* podría necesitar la ayuda de los científicos de datos y los expertos en el dominio. Por ejemplo, para convertir datos semanales en datos diarios, el *feature preprocessor* podría preguntar si se desea simplemente replicar un valor semanal por cada día de la semana o, por contra, se prefiere aplicar alguna técnica de interpolación más compleja, entre otras opciones. Tras seleccionar las transformaciones a aplicar, el *feature preprocessor* las ejecutaría de manera automática, generando el *dataset* requerido.

### 3. Conclusiones

Este artículo ha presentado la arquitectura inicial de *Hannah*, una solución para la extracción, limpieza e integración de datos en *datasets* para análisis y minería de series temporales de datos. Gracias a este *framework* esperamos que los científicos de datos no tengan que escribir a mano largas rutinas para extraer datos de *data lakes* concretos y luego alinearlos y limpiarlos para su adecuada inclusión en un *dataset*. Estas tareas se harían ahora automáticamente, guiadas en ciertos pasos por los científicos de datos. Gracias a ello, esperamos que se reduzca el tiempo necesario para generar estos *datasets*, así como el número de errores que se introducen durante este proceso.

Tal como se ha comentado, *Hannah* está en sus primeras fases de desarrollo. Actualmente, se ha completado una revisión de la literatura con dos objetivos: conocer potenciales trabajos relacionados y averiguar cuáles son las técnicas de preprocesamiento de datos utilizadas en la generación de *datasets* de datos temporales. Además, se ha alimentado un *data lake* con diversos conjuntos de datos de prueba, y se han registrado dichos datos en el correspondiente catálogo de datos. En los próximos meses, esperamos completar el diseño e implementación de cada uno de los elementos que conforman este sistema, así como hacer un estudio detallado de sus potenciales beneficios.

### Referencias

1. Nargesian, F., Zhu, E., Miller, R.J., Pu, K.Q., Arocena, P.C.: Data lake management: challenges and opportunities. Proceedings of the VLDB Endowment **12**(12), 1986–1989 (2019). <https://doi.org/0.14778/3352063.3352116>
2. Sal, B., García-Saiz, D., Sánchez, P.: Automated generation of datasets from fishbone diagrams. In: Attiogbé, C., Yahia, S.B. (eds.) Proceedings of 10th International Conference on Model and Data Engineering (MEDI). Lecture Notes in Computer Science, vol. 12732, pp. 249–263 (June 2021). [https://doi.org/10.1007/978-3-030-78428-7\\_20](https://doi.org/10.1007/978-3-030-78428-7_20)
3. Vega, A.D.L., García-Saiz, D., Zorrilla, M., Sánchez, P.: Lavoisier: A DSL for increasing the level of abstraction of data selection and formatting in data mining. Journal of Computer Languages **60**, 100987 (Oct 2020). <https://doi.org/10.1016/j.cola.2020.100987>