Towards the Specification and Generation of Time Series Datasets from Data Lakes

Brian Sal*, Alfonso de la Vega*, Patricia López-Martínez*, Diego García-Saiz*, Alicia Grande†, David López† and Pablo Sánchez*
* Software Engineering and Real-Time Group, Universidad de Cantabria,
Santander (Cantabria), Spain
Email: {brian.sal, alfonso.delavega, patricia.lopez, diego.garcia, p.sanchez}@unican.es
† LIS Data Solutions,
Santander (Cantabria), Spain
Email: {alicia.grande, david.lopez}@lisdatasolutions.com

***Abstract*—**
These days, more and more organizations are building data lakes as a mechanism to store the information they generate. This information is considered as a valuable asset that, if properly analyzed, can help to make more informed decisions. However, since the analyses to be performed are often not known in advance, these data are stored in a raw format. This means that any application built on top of a data lake must carefully elicit what data will be used for a particular analysis and how those data will be transformed to make them all fit together into a dataset. This data selection and preparation task is typically performed by data scientists that write large and complicated scripts in data management languages to extract and transform the required data. This reduces the productivity of data scientists, who must write large pieces of highly similar code. It also makes it difficult for domain experts to participate in this process because they have little understanding of these scripts. To alleviate this problem, this work introduces a work-in-progress version of a high-level declarative language for specifying the requirements that a dataset coming from a data lake must satisfy. This language is then processed to automatically generate the specified dataset, allowing data scientists and domain experts to be agnostic about the details of how data are exactly retrieved and transformed.

I. Introduction

Data has become a very valuable asset, being even considered the *new gold* by some consulting companies [1]. Consequently, organizations have started storing data about their daily activities with the expectation of conducting various analyses in the future to derive valuable information. Since these analyses are not known in advance, these companies face uncertainty regarding how they must clean, process and store their data so that they can be efficiently retrieved by the analysis applications.

Therefore, many companies are adopting a data storage strategy known as *data lake* [2] [3], where data is stored without any prior processing in its raw format. Roughly speaking, a data lake can be seen as a large storage repository where different data bundles are stored in heterogeneous formats. The main advantage of a data lake is that it preserves the entirety of the information, allowing each application to process and transform the stored data as needed. On the other hand, data lakes present significant challenges, such as ensuring the data is easily searchable and structuring its internal organization for efficient data addition and retrieval [4] [5].

Since the data in a data lake is stored in a raw format, it requires extraction and processing before it can be consolidated into a dataset in order to be analyzed. This task is currently performed by data scientists by writing long and complex scripts in languages like Python or R. However, this approach presents two main problems: (1) data scientists must write large pieces of similar code, which hinders their productivity; and, (2) it complicates the inclusion of domain experts in this process, who can hardly understand these scripts. Domain experts are key for the process of selecting the data to be included in an analysis because they have the expertise to know the exact meaning of each piece of data and how to interpret it.

To alleviate these problems, this work presents a language for the specification and automated generation of datasets from data lakes. This language provides a high-level declarative syntax that allows data scientists and domain experts to specify the requirements that a dataset must satisfy, focusing on what data must be retrieved from the data lake, as well as on what high-level operations must be used to process these data so that they can fit all together into a dataset, and forget about how these operations need to be exactly performed. This specification is then processed by a set of language processors, which execute multiple data transformation operations until returning the specified dataset. This contributes first to increase data scientists productivity, who can work now at a higher abstraction level and write less code. Moreover, we expect that domain experts can understand the syntax of these dataset specifications more easily and, consequently, facilitate their involvement in the data selection processes.

This work is still in progress and it is being developed in the context of a project with LIS Data Solutions, a software company based in Northern Spain focused on the development of data collection and data analysis systems for clients across different sectors.

After this introduction, this work is structured as follows: Section II provides some background and describes the motivation for this work. Section III presents the language for dataset specification and the framework that supports it. Section IV comments on related work and Section V summarizes this paper.

II. Background and Motivation

A. Running Example

This section introduces the running example that will be used to illustrate the different concepts that appear in this work. This running example was provided by LIS Data Solutions, the software company that collaborates in this work. Along with their clients' data, LIS also retrieves and stores data from different open data sources, such as the State Meteorological Agency[1]. These data are used to complement customer data in different analyses.

One of the data sources that is integrated in the data lake provides data of logistics companies. These data are used for the forecasting of shipment volume and service delivery times for these companies. Specifically, we are interested in analyzing how the weather, the economy and the traffic conditions can influence these delivery activities. To achieve this goal, we can use time series analysis, as described in the next section.

B. Time Series Data Mining

Time series data mining [6] is a subfield of temporal data mining that focuses on analyzing datasets that consist of sequences of data points ordered by time. *Temporal data mining* [7] is a knowledge area that refers to the process of extracting valuable information from time-varying data, which includes time series and stream data mining. *Stream data mining* [8] is more oriented to the real-time processing of continuously generated data streams. For instance, stream data mining is widely used in Industry 4.0 [9] to analyze streams of data coming from different assembly line sensors.

Time series data mining is very suitable for analyzing the data contained within data lakes. In a data lake, data are periodically incorporated over time, forming a time sequence for which time series data mining becomes a natural analysis technique.

Figure 1 illustrates a typical time series data mining process. The two first steps (Figure 1, labels 1 and 2) aim to elicit two important requirements that will drive all the analysis process. The first step (Figure 1, label 1) identifies the business questions that need to be answered, with the help of the business and domain experts. This step involves understanding the domain of the problem, defining the objectives and specifying the scope of the analysis. In our running example, this business question is to predict the volume of shipments of the logistics companies.

The second step is the selection of the data sources that will be used to address the identified questions (Figure 1, label 2). When data is stored in a data lake, like in our running example, this step implies exploring the data lake to pick up the relevant data that may have influence in the business question and, therefore, may help to answer it. The participation of the domain experts in this task is key, since these experts are who can better reason about the influence of each piece of data in the business questions, due to their knowledge of the domain and their professional experience.

[1]https://www.aemet.es/en/datos_abiertos/AEMET_OpenData

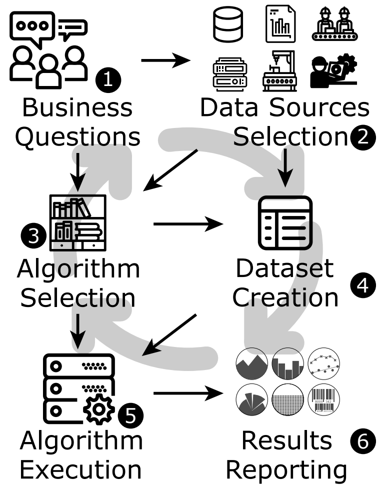

Fig. 1. Time series data mining process.

In the next step (Figure 1, label 3), the analysis technique and algorithms that will be used to try to answer the business question must be selected. This step involves more technical considerations, so it is typically undertaken by a data scientist.

Before the chosen algorithms can process the selected data sources, we need to put all the data together into a dataset (Figure 1, label 4). Roughly speaking, a dataset is a very concrete tabular format in which data must satisfy certain constraints that often depend on the algorithm to be used. When working with a data lake, this step implies retrieving concrete data fragments from different data bundles that may be stored in different formats. Moreover, some data may need to be transformed. For instance, several values might need to be aggregated in a single value. In these cases, domain experts' collaboration can be critical again, since their domain knowledge may be very helpful to ensure that data do not lose meaning or accuracy significantly during these transformations. This work focuses on this specific step of the whole analysis process.

After creating a dataset, the selected algorithms are finally executed (Figure 1, label 5) and the results reported (Figure 1, label 6). This involves visualizing the results appropriately and interpreting them, for which, again, the knowledge and experience of domain experts are key.

This process is not always sequential and it can be iterative, as indicated by the cycle of arrows in the background of the image. For example, the reported results may raise new questions that may require the selection of new data sources and the refinement of the datasets. Similarly, after executing an algorithm, we may detect that some features generates noise and it is better to remove them, so we would need to update the datasets.

Date	Shipments	Temp. (°C)	Rain (mm)	GDP	Traffic (trucks/h)
2019-02-07	31	13.4	1.7	965.6	1230
2019-02-08	28	10.2	0.5	965.6	1320
2019-02-09	34	11.8	0.8	965.6	1280
...	...	...	...	...	...

Fig. 2. Excerpt of a dataset for our running example.

C. Problem Statement

This section details the specific problem this work aims to solve. As it was commented in the previous section, this work focuses on the data selection and preparation step of a time series data mining process.

The goal of this step is to produce a dataset that can be digested by time series analysis algorithms. These datasets are tabular data structures where each row represents a specific point of time. In many cases, the period between rows is uniform, so we can say that each row is placed in the dataset according to a *sampling rate*. Figure 2 shows an example of a toy dataset for our case study, where data are sampled daily.

The first problem we face to generate a dataset such as shown in Figure 2 is that each column comes from separate data bundles that might be in different formats. In our case, the information of shipments of the logistic company are stored in a single file using the Apache Parquet format. *Parquet* is a very popular format in data lake and big data systems as it allows the storage of tabular data in very compact sizes. Each row of this Parquet file contains data about a single shipment of the logistic company. These data includes its sender, receiver, origin, destination, weight, number of items, and delivery time, among other elements. On the other hand, weather data are added daily to the data lake in a JSON file that contains information about temperature, rainfall or wind speed, among other elements, for different Spanish cities. Traffic conditions are incorporated once per month in an XML file that specifies the total number of cars and trucks that have circulated in that month by all Spanish highways. For the economy indicators, data about the Spanish *Gross Domestic Product (GDP)*, contained in a JSON file, are added to the data lake once each three months.

Therefore, if we wish to analyze the period corresponding to the first semester of 2023, to collect the required data, we must perform the following actions:

1) For the shipment data, extract the concrete set of rows from the Parquet file corresponding to the desired period.
2) For the weather data, retrieve the JSON files that correspond to the first semester of 2023, and extract temperature and rainfall data for a specific city from each file.
3) For the GDP, retrieve the JSON files corresponding to the two first quarters of 2023 and select the required data.
4) For the traffic, retrieve the XML files of the first six months of 2023 and extract the numbers of trucks per hour for each month.

As it can be noticed, depending on each source, we must retrieve a different number of files and navigate its internal structure. This is a very technical aspect that is beyond the typical skills of domain experts.

A second problem we need to solve is the harmonization or matching of the sampling frequencies of each variable to be included in the dataset. As it can be observed, in our running example, each variable is recorded at a different rate. For instance, shipment information is recorded daily but it is stored at the shipment level, not in a temporal basis. Therefore, to produce a single data per day, we need to group all shipments per day and count how many of them are handled each day. Weather data, fortunately, is in a daily basis, so we do not need to transform it. However, economic and traffic data are quarterly and monthly, respectively. Therefore, we need to design a strategy to generate a data for each day from these values.

For this purpose, there are several strategies. For example, we may simply copy a monthly value to each day of the month. On the other hand, we may use different interpolation techniques for the missing values at these points. The collaboration of domain experts becomes important again in this point, since they have the knowledge to better decide how to transform each data. For example, in the case of the GDP, since the economy does not change suddenly from one day to the other, it would make sense to use an interpolation technique that shows how this indicator increases or decreases smoothly during each quarter.

Finally, a third problem to be solved is how to deal with noise, outliers or missing data. For example, the logistic company may close one day for whatever reason, such as an important internal reorganization, so there were not records for that day. In that case, again, we need to decide how to deal with this issue. We may simply copy the value of the last day or use the average value for that week or month, among other options.

Currently, all these tasks are carried out mainly by data scientists, who write large and complicated scripts in languages like Python or R to extract data from the data lake, and process and transform them. This has two main problems: (1) it decreases their productivity; and, (2) it complicates the inclusion of domain experts in this process, as they perceive these scripts as difficult to understand and work with.

To alleviate this problem, this work proposes a language that allows data scientists to work at a higher abstraction level, becoming more productive and allowing domain experts to get more easily involved in the dataset creation process. Next section describes this language as well as the framework that supports its.

III. The Hannah Language and Framework

Figure 3 provides a general overview of the framework we are working on as a solution to the problem described in previous sections. This framework assumes that the data available in the data lake are described in a data catalog. A *data catalog* [10] is a software system specifically designed

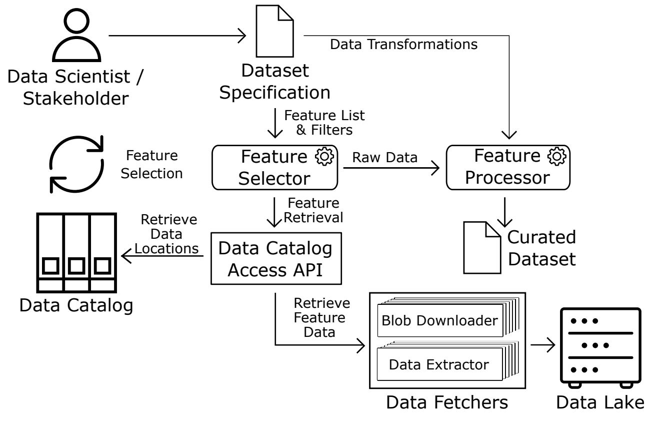

Fig. 3. General overview of our solution

```
1   dataset ShipmentForecasting
2     sampling daily
3     from 2023/01/01 to 2023/06/30
4           with currentDay as index
5     with features
6       from ShipmentInformation {
7         Shipments is
8            count(id[orderDate=currentDay])}
9       from WeatherData[city='Santander'] {
10           temperature;
11           rain;}
12      from EconomicData {
13        gdp
14        expanded by linear_interpolation; }
15      from TrafficData {
16        trucks expanded by repetition; }
```

Listing 1. An example of Hannah specification.

to keep a well-organized and searchable inventory of all data assets in an organization. Data catalogs are strongly connected to data lakes, as they become a natural solution to describe what data are contained in the data lake and how to locate them.

Thus, to generate a dataset for time series data mining, the data scientists and the domain experts would start navigating the data catalog to find those data, or features, that may be helpful for their objectives. As a result, they get a set of data bundles and feature identifiers, along with a set of potential filters, which might be applied to these elements. For instance, in the case of the weather information, the data catalog would specify that we can filter the information by city.

Using this information and using our language, the data scientists and the domain experts would write a dataset specification like shown in Listing 1. This specification contains two different parts: the first one defines the general shape of the dataset (lines 1 to 4); and the second one specifies the selection of data to be included in the dataset (lines 5 to 16).

In our case, the first part indicates that the dataset will be named *ShipmentForecasting* (line 1), it will have a row per day or daily sampling (line 2), and it will cover the time frame corresponding to the first semester of 2023 (line 3). Moreover, the variable *currentDay* will hold the value of the day corresponding to each row during the generation process (line 4).

Once the base format for the dataset is defined, the data scientists and the domain experts specify what data must be included in it. In order to do it, they specify the name of the data bundle, or blob, to be processed, and the features to be extracted from that data bundle. For example, lines 9 to 11 specify that the *temperature* and *rain* values must be extracted from the *WeatherData* files.

Moreover, they can add filters to these data bundles to specify that they are interested in just a portion of the whole set of data. For example, line 9 specifies that only the weather data corresponding to the city of Santander will be considered.

Data scientists can also add calculated values to the output dataset. For instance, lines 7 and 8 specify that the number of shipments is calculated by counting the number of identifiers corresponding to shipments that belongs to a same day.

Finally, for each feature that does not conform to the sampling rate of the dataset, data scientists and domain experts must specify how to transform it. For example, line 15 indicates that the GDP for each day is calculated using linear interpolation.

To process this specification, two different modules are executed: the *feature selector* and the *feature processor*. The feature selector communicates with the *Data Catalog Access API* to retrieve the selected data. The Data Catalog Access API is basically a mechanism to uniform the communication with different concrete data catalog and data lake technologies. This way, we isolate the feature selector from particularities of concrete technologies and we facilitate the incorporation of new data catalog and data lake technologies into our framework.

To retrieve feature data, the Data Catalog Access API firstly accesses to the Data Catalog to get references to the locations where these data are stored in the data lake. Then, it invokes different *data fetchers* to extract the desired data from these specific locations. The *data fetchers* are software components able to extract data from concrete data lake technologies and data formats. The data fetchers are comprised of two subcomponents: the *blob downloader* and the *data extractor*. The first ones get blobs, i.e. data blocks, from specific data lakes technologies e.g., *Azure Data Lake*, whereas the second ones retrieve specific pieces of data from these data blocks, e.g, an attribute value of a specific object in a JSON file.

All the retrieved data are passed to the *Feature Processor*, which is in charge of automatically transforming the raw data to create a curated dataset. This processing uniforms the sampling frequency of each feature, using the information provided by the dataset specification. After that, the final dataset is generated.

IV. Related Work

In this section, we will discuss related work in the areas of data lake management, time series data integration, and automation of data preprocessing tasks and dataset generation.

Data lakes have gained popularity as scalable and cost-effective solutions for storing and managing large volumes of heterogeneous data. They enable organizations to store raw, unprocessed data and transform it into valuable insights. However, data lake management poses challenges, including the extraction and integration of diverse datasets within data lakes. Several studies have proposed frameworks and approaches for efficient data lake management [11], [12], [13].

Among the proposed frameworks, CLAMS stands out as a system designed to discover and enforce expressive integrity constraints from large amounts of data in a data lake. CLAMS addresses the assessment of data quality and cleaning of heterogeneous data sources, tasks prior to data integration, by efficiently detecting errors and interacting with human experts for validation and data repairs. It has been successfully deployed in a real large-scale enterprise data lake, demonstrating its ability to uncover data inconsistencies and errors early in the data processing stack.

Furthermore, ALITE is a proposal for the scalable integration of tables that may have been discovered using join, union, or correlated table search. It addresses the underexplored area of proper integration of discovered tables by relaxing the assumptions about shared attribute names, completeness, and acyclic join patterns in tables. In order to do it, ALITE exploits a new Full Disjunction algorithm that aims to be more efficient integrating real data lake tables than existing baselines.

Another notable system for data lake management is *DeepDive*, which specializes in extracting relational databases from *dark data*, such as text, tables, and images that cannot be exploited by standard relational tools. It offers high precision and recall at a reasonable engineering cost, allowing analysts to create databases with accuracy comparable to human annotators. DeepDive has been successfully deployed in various domains, including insurance, materials science, genomics, paleontology, and law enforcement.

These works highlight the importance of addressing the challenges associated with data extraction and integration within data lakes. However, there is still a need for comprehensive solutions that specifically address the extraction and integration of time series datasets from data lakes [5], like the language and the framework that we have introduced in this work do.

V. Conclusions

This work has presented a work-in-progress version of a high-level language for the specification and automated generation of temporal datasets from data lakes. The language allows data scientists to work at a higher abstraction level, which we expect that contributes to increase their productivity. As an indicator of the advantages of this language, to build manually the dataset corresponding to specification of Listing 1, we had to write approximately 700 lines of Python code, whereas in our language the same work can be done with just around 20.

Moreover, we expect that the high-level declarative syntax of our language eases the participation of domain experts in this process. This language may also be used inside *self-service business intelligence* approaches [14]. In these approaches, domain experts build dashboards and reports by themselves that help them visualize data and make decisions. Inside this paradigm, our language could be used to build datasets that would be later visualized in an appropriate dashboard.

As commented, this work is in progress, more specifically, in its initial steps. We still need to refine the grammar of our language by applying it to a wider range of case studies that cover a comprehensive set of scenarios for dataset generation. Once we have agreed on a definitive grammar, we need to implement each component of our framework, for which we have sketched its initial design.

Acknowledgment

This work was partially supported by MCIN/ AEI /10.13039/501100011033/FEDER "Una manera de hacer Europa" under the grant PID2021-124502OB-C42 (PRESECREL) and the project "Industrial Smart Data Pool" (2021/INN/80), funded by the INNOVA program of the Gobierno de Cantabria.

References

[1] J. von Ditfurth and H. Aholt, "Data is the new gold," Deloitte, Tech. Rep., Feb. 2018.

[2] H. Fang, "Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem," in *Proc. of the International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, Shenyang (China), Jun. 2015, pp. 820–824.

[3] C. Quix and R. Hai, "Data Lake," in *Encyclopedia of Big Data Technologies*, S. Sakr and A. Zomaya, Eds. Springer, 2018, pp. 1–8.

[4] C. Giebler, C. Gröger, E. Hoos, H. Schwarz, and B. Mitschang, "Leveraging the data lake: Current state and challenges," in *Big Data Analytics and Knowledge Discovery*, C. Ordonez, I.-Y. Song, G. Anderst-Kotsis, A. M. Tjoa, and I. Khalil, Eds. Springer, 2019, vol. 11708, pp. 179–188, series Title: Lecture Notes in Computer Science.

[5] F. Nargesian, E. Zhu, R. J. Miller, K. Q. Pu, and P. C. Arocena, "Data lake management: challenges and opportunities," *Proceedings of the VLDB Endowment*, vol. 12, no. 12, pp. 1986–1989, 2019.

[6] T.-C. Fu, "A review on time series data mining," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, pp. 164–181, Feb. 2011.

[7] T. Mitsa, *Temporal Data Mining*. Chapman & Hall/CRC, Mar. 2010.

[8] L. Rutkowski, M. Jaworski, and P. Duda, *Stream Data Mining: Algorithms and Their Probabilistic Properties*, ser. Studies in Big Data. Springer, 2020, vol. 56.

[9] R. Sahal, J. G. Breslin, and M. I. Ali, "Big data and stream processing platforms for Industry 4.0 requirements mapping for a predictive maintenance use case," *Journal of Manufacturing Systems*, vol. 54, pp. 138–151, Jan. 2020.

[10] O. Olesen-Bagneux, *The Enterprise Data Catalog*. O'Reilly, 2023.

[11] M. Farid, A. Roatis, I. F. Ilyas, H.-F. Hoffmann, and X. Chu, "Clams: bringing quality to data lakes," in *Proceedings of the 2016 International Conference on Management of Data*, 2016, pp. 2089–2092.

[12] A. Khatiwada, R. Shraga, W. Gatterbauer, and R. J. Miller, "Integrating data lake tables," *Proceedings of the VLDB Endowment*, vol. 16, no. 4, pp. 932–945, 2022.

[13] C. Zhang, J. Shin, C. Ré, M. Cafarella, and F. Niu, "Extracting databases from dark data with deepdive," in *Proceedings of the 2016 International Conference on Management of Data*, 2016, pp. 847–859.

[14] P. Alpar and M. Schulz, "Self-Service Business Intelligence," *Business & Information Systems Engineering*, vol. 58, no. 2, pp. 151–155, Apr. 2016.