Maximizing Efficiency in Existing Data Preparation Pipelines

Angelo Mozzillo[1]

[1]First Year PhD Sudent, ICT Doctorate at DBGroup, University of Modena and Reggio Emilia, Modena, Italy

Abstract

Data preparation involves transforming, cleaning, and converting raw data into a usable format for further analysis. This process can be time-consuming and resource-intensive. Typically, data to be analyzed is placed in two-dimensional tabular structures called DataFrames. These are the de facto standards for data science and machine learning tasks to store and process large amounts of structured data. Pandas is the most commonly used API for manipulating DataFrames in Python due to its popularity and comprehensive functionality. However, it is important to note that Pandas has some limitations, such as being single-core and non-distributed, which can impact its efficiency and performance for large datasets. Several libraries have been developed to expand the functionality of Pandas by utilizing multi-core and distributed computing capabilities. Therefore, the choice of library can significantly impact the efficiency and performance of the data preparation pipeline, and it is important to consider the specific requirements of the project when selecting a library.

In this paper, I will discuss my primary contributions during the first months of my PhD program, which mainly focused on creating an open-source framework to evaluate the performance of seven Python libraries on five datasets of different sizes. The primary objective is to identify the best combination of these libraries to create efficient data preparation pipelines that deal with the needs of users who frequently encounter several libraries claiming to perform similar tasks. Finally, I will describe some ideas concerning the future directions of this research.

Keywords

Data Preparation, Big Data, Data Science Pipeline

1. Introduction

Data preparation is an important process that involves exploring, combining, cleaning, and transforming raw data into curated datasets that can be used for various purposes [1, 2]. This process is essential for ensuring that the data is accurate, reliable, and suitable for the intended use. In simpler terms, data preparation can be defined as the set of pre-processing operations that are the pre-requisite for an effective data science task. These operations are aimed at transforming the raw data into a structured format that is more suitable for analysis and decision-making.

One of the most common data structures used for storing and manipulating data in the data preparation process is the DataFrame. A DataFrame is a two-dimensional data structure that consists of rows and columns [3]. Pandas is the most widely used library for operating on

SEBD 2023: 31st Symposium on Advanced Database System, July 02–05, 2023, Galzignano Terme, Padua, Italy

angelo.mozzillo@unimore.it (A. Mozzillo)

0000-0002-3465-5027 (A. Mozzillo)

DataFrames [4]; it is considered the standard for all data preparation operations. It provides an API with a wide range of functions for carrying out the various stages of data preparation. These functions allow users to perform tasks such as data cleaning, merging, aggregation, and transformation to ensure that the data is of high quality and suitable for further analysis.

Despite its many advantages, Pandas has significant limitations when dealing with Big Data. The main drawbacks of Pandas are:

- *Memory*: Since Pandas loads the full dataset into memory, the size of the dataset that can be analyzed is constrained by the memory capacity of the system.
- *Speed*: Pandas is a single-threaded library by design, so working with large datasets might make it slow. Certain tasks, including grouping and sorting, can take a very long time to complete.
- *Scalability*: Pandas cannot readily grow over different nodes or a cluster since they are not optimized to fully exploit distributed or parallel systems.

Over the years, in response to the need to process large amounts of data efficiently, several libraries have been developed to overcome this limitation. There has been limited research conducted in the field of optimizing data preparation pipelines in DataFrame systems. Previous studies have mainly focused on evaluating different libraries on data processing tasks. First, Petersohn et al. [5] proposed a technique that improves the efficiency and scalability of data processing in parallel DataFrame systems. Second, Watson et al. [6] introduced a data science benchmark specifically designed to evaluate the performance of systems handling data processing and analytics tasks. Furthermore, it is important to note that the existing literature often refers to individual articles presenting specific libraries, rather than providing a comprehensive comparison of these libraries in the context of data preparation pipelines [3, 7, 8]. Although these articles help to provide valuable insights into the capabilities of individual libraries, a comprehensive evaluation and comparison of libraries in the context of a complete data preparation pipeline remains limited. The DFBen framework, introduced in the 2 section, aims to fill this gap by conducting a systematic evaluation and comparison of various libraries in terms of their performance and effectiveness in executing data preparation pipelines.

DFBen focuses on investigating various crucial aspects of these pipelines, enabling researchers and practitioners to make informed decisions. Firstly, the investigation of lazy evaluation and its impact on overall performance is essential, as it offers advantages such as reduced memory consumption and improved processing efficiency. Secondly, scalability is a critical factor, considering the exponential growth of data volumes. Assessing the libraries' ability to scale efficiently and utilize computational resources is crucial for real-world implementation. Moreover, the choice of I/O formats in data preparation plays a significant role, as different formats have distinct characteristics related to storage efficiency, data compression, and compatibility with downstream processing. Lastly, the setup and configuration requirements of each library significantly influence the ease of adoption and integration into existing pipelines.

In Section 3, I present future research activities, including transitioning to a FaaS (Function-as-a-Service) framework and utilizing machine learning models to determine the ideal combination of libraries based on input dataset features. These innovations promise to enhance the efficiency and accuracy of data preparation pipelines.

2. DFBen: DataFrame Evaluation Framework

DFBen is an open-source project that aims to compare various commercial libraries, such as Pandas[1], Dask[2], Spark[3], Modin[4], Polars[5], Vaex[6], and Rapids[7], that utilize DataFrames for data preparation tasks. In order to determine how much the libraries have improved on the standard set by Pandas, this project conducts a comparison of the most commonly used functions grouped into four stages of data preparation. To thoroughly assess the libraries' potential, tests were conducted on a number of distinct datasets with different sizes, schemas, and types.

To identify the most effective libraries for constructing data preparation pipelines, I evaluate their performance in four key areas. First, I assess the performance of individual functions used for data preparation. Next, I test libraries in the four key stages of the data preparation pipeline: Input/Output (I/O), Exploratory Data Analysis (EDA), Data Transformation, and Data Cleaning. By looking at the performance of the libraries in each of these areas, I can identify the best ones for each stage of the pipeline. Lastly, I analyze how well libraries perform across the full data preparation pipeline. In addition, I investigate the impact of scalability on the performance of these libraries. This enables me to assess their effectiveness in handling large datasets and complex pipelines.

The project's open-source nature enables seamless incorporation of additional libraries and preparators through a *Library Class* that encapsulates the library with the implemented preparators, as well as a *Dataset Class* that allows for the inclusion of additional datasets in the configuration file. Moreover, the pipelines are managed using a JSON file, enabling the easy addition and modification of existing pipelines. In its current state, DFBen is a helpful tool for anyone looking to compare the many DataFrame librariesavailable and select the one that best fits their needs and hardware capabilities in terms of performance and scalability.

2.1. Framework Design

Data preparation involves multiple stages and is typically performed using preparators, which are responsible for small-scale preparation steps. In line with Naumann et al.'s (2020) Data Preparation: A Survey of Commercial Tools [1], similar to data preparation tools, libraries often offer a range of preparator implementations for sequential application or the creation of data preparation pipelines. The mentioned paper involved the selection of 40 commonly preparators, which I subsequently implemented within the DFBen framework.

To ensure optimal pipeline design, I select one of the top three *Kaggle*[8] most voted notebooks, specifically tailored to the pre-selected dataset. This approach leverages the collective expertise of the data science community and ensures that the pipeline design is based on best practices

[1] https://pandas.pydata.org/.
[2] https://www.dask.org/.
[3] https://spark.apache.org/.
[4] https://modin.readthedocs.io/en/stable/.
[5] https://pandas.pydata.org/.
[6] https://vaex.io/docs/.
[7] https://rapids.ai/.
[8] https://www.kaggle.com/.

and state-of-the-art techniques. To provide an evaluation with different levels of granularity, as described in Section 2.3, I organized the preparators into four key stages:

- *I/O*: this stage includes the functions that handle the input and output of data in various formats, such as databases, *CSV* or *JSON* files and *web APIs*;
- *EDA*: here, the functions are grouped that deal with the exploratory analysis of data characteristics for better understanding and detecting any error or anomaly;
- *Data Transformation*: at this stage, functions deal with transformation, such as data normalization, deduplication, categorical data encoding, data aggregation, and other techniques that make the data more suitable for analysis;
- *Data Cleaning*: the final stage involves data cleaning, which may include handling missing or erroneous data, correcting outliers, eliminating outliers, and other techniques that ensure data quality and reliability of analysis results.

The datasets selection for this study, as described in [5], was driven by their suitability for evaluating the performance and capabilities of the libraries in various contexts. The first dataset, *120 years of Olympic history-athletes and results* presents a comprehensive historical record of all Olympic Games held between 1896 and 2016. This dataset is ideal for testing a lot of preparatrs since its popularity on kaggle. The second dataset, *NYC Yellow Taxi* encompasses detailed information about every taxi trip taken in New York City. To evaluate the scalability of the libraries, two versions of this dataset are utilized: one containing data from 2015, and the other encompassing data from 2009 to 2015. This allows for a comprehensive examination of the libraries' ability to handle large volumes of real-world data. To evaluate type inference and handling of skewed and sparse string datasets, two additional datasets are included alongside the mentioned datasets. The *Loan Data* dataset contains valuable information about loans issued by a financial institution. The *California State Patrol* dataset encompasses information about traffic stops conducted by the California Highway Patrol.

2.2. Implementation

As described in Section 2.1, I used 50 preparators for my study. To implement these preparators, I relied on functions defined in the libraries as well as custom functions designed to cover all possible cases. To ensure optimal isolation, repeatability, and efficient management of machine configurations, the code is executed within a Docker [9] container for each library.

The tests are run on 3 machine configurations representing 3 different types of use: PC (8 cores-16GB RAM), Workstation (16 cores-64GB RAM) and Server (24 cores-180 GB RAM). In all configurations, the Tesla T4 GPU with 16GB RAM used by Rapids library is included. In this way, users can test the framework in different contexts and verify its performance on machines with different hardware configurations. Finally, the management of DFBen's configurations is entrusted to *JSON* files that can be easily configured by users, who can customize the framework's settings and the functions to be executed according to their own needs and preferences.

[9] https://www.docker.com/.

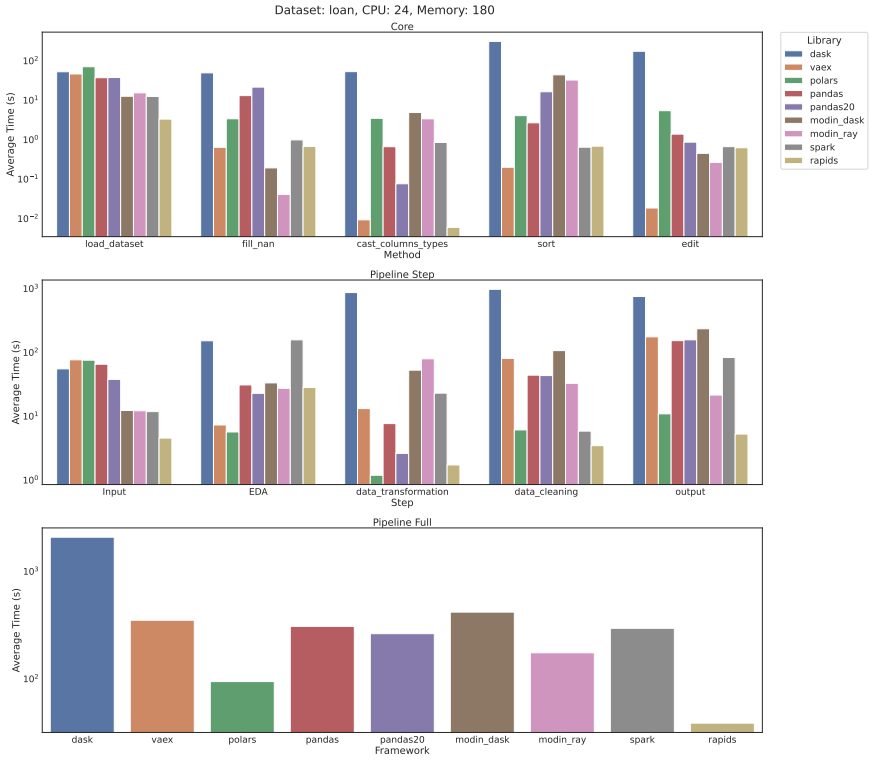

Figure 1: Results of DFBen Execution on Loan Dataset

2.3. Evaluation

For evaluating the performance, I track the execution time in different cases [6]. It is important to note that some libraries utilize *lazy evaluation* [9], which is a programming technique that delays the execution of a computation or evaluation until it is actually needed. This is in contrast to *eager evaluation* where computations are executed immediately regardless of whether the result is actually used or not.

Lazy evaluation can lead to more efficient use of resources as unnecessary computations are avoided. When comparing libraries like Pandas, which do not use lazy evaluation, with those that do, it is necessary to calculate the execution time accurately in order to have a fair and objective comparison. In order to do that, the evaluation framework consists of three types of tests:

- *Core*, is responsible for forcing the execution of lazy libraries and evaluating the performance of individual operations. This allows an accurate assessment of the efficiency of each library's function and the identification of any performance bottlenecks.
- *Pipeline-Step*, executes a data preparation pipeline and calculates the execution time at the conclusion of each stage. This method shows how each library's function performs at each stage of the pipeline and shows any inefficiencies or room for improvement.
- *Pipeline-Full*, component runs the entire data preparation pipeline. This approach provides

a comprehensive evaluation of each library's performance and allows comparison of the overall efficiency of completing the entire data preparation process.

Figure 1 displays the initial result of the three types of evaluation explained above, which have been obtained by running the tool on the Loan Dataset using a machine configuration of 24 cores and 180 GB of RAM. However, it is important to note that these results are only in the preliminary stage, and the analysis of the data is still ongoing. Further explorations will be needed to provide more detailed information, considering factors beyond efficiency. The effectiveness of the libraries involves various aspects that need to be evaluated. Therefore, it will be crucial to consider the specific requirements of the dataset and employ a refined selection process to identify the libraries that best meet the desired criteria.

3. Future Directions

During the first few months of my PhD project, I mainly focused on creating an open-source framework that enables the comparison of seven libraries for the construction of data preparation pipelines. DFBen provides an opportunity for future research directions. In the next year, I plan to work on two main research activities:

1. The first one can be enabled by the tool with the automatic selection of the most effective combination of libraries for constructing pipelines. To address this, the tool will gather information about the characteristics of the dataset and the required pipeline and, by taking into account the execution times of individual functions, will be able to suggest the proper library. This would remove the difficulty of manually selecting the appropriate one for a given task.
2. The second research activity can be the transition of DFBen to a FaaS framework. One reason for this is that implementing a data preparation pipeline requires iterating the steps described in Section 2.1 several times. When resources are limited, a small sample of the reference dataset may suffice for testing, but this may not always be representative, and it may not show certain record types that would lead to errors. Frameworks such as Spark currently dominate this context [10], but in the implementation phase, it may not be necessary to use and configure a framework such as this due to the iterative nature of the process. FaaS has the potential to revolutionize the way data transformation pipelines are implemented by making it easier to manage and scale individual functions in response to events or triggers. However, this approach also requires careful consideration of issues such as security, data privacy, and performance optimization. [11, 12]

To achieve these objectives, I want to study and test various methods and approaches for creating data transformation pipelines based on FaaS. Additionally, I will explore novel approaches for incorporating machine learning to construct pipelines that combine libraries, thus enhancing their effectiveness and efficiency. By doing so, I believe that this research will make a valuable contribution to the field of data preparation pipelines and help advance the field of data science as a whole.

Acknowledgments

I wish to express my gratitude to my tutor, Prof. Sonia Bergamaschi and my co-tutor, Giovanni Simonini for their guidance and support throughout my PhD journey, as well as to my colleagues at the DBGroup who have been a constant source of encouragement. In addition, I would like to express my sincere appreciation to Leonardo for funding my research grant.

References

[1] M. Hameed, F. Naumann, Data preparation: A survey of commercial tools, ACM SIGMOD Record 49 (2020) 18–29.

[2] L. Gagliardelli, G. Papadakis, G. Simonini, S. Bergamaschi, T. Palpanas, Generalized supervised meta-blocking, Proc. VLDB Endow. 15 (2022) 1902–1910. URL: https://www.vldb.org/pvldb/vol15/p1902-gagliardelli.pdf.

[3] D. Petersohn, S. Macke, D. Xin, W. Ma, D. Lee, X. Mo, J. E. Gonzalez, J. M. Hellerstein, A. D. Joseph, A. Parameswaran, Towards scalable dataframe systems, arXiv preprint arXiv:2001.00888 (2020).

[4] W. McKinney, pandas: a foundational python library for data analysis and statistics, Python for high performance and scientific computing 14 (2011) 1–9.

[5] D. Petersohn, D. Tang, R. Durrani, A. Melik-Adamyan, J. E. Gonzalez, A. D. Joseph, A. G. Parameswaran, Flexible rule-based decomposition and metadata independence in modin: a parallel dataframe system, Proceedings of the VLDB Endowment 15 (2021).

[6] A. Watson, D. S. V. Babu, S. Ray, Sanzu: A data science benchmark, in: Procedings International Conference on Big Data (Big Data), IEEE, 2017, pp. 263–272.

[7] M. A. Breddels, J. Veljanoski, Vaex: big data exploration in the era of gaia, Astronomy & Astrophysics 618 (2018) A13.

[8] M. Rahmany, A. M. Zin, E. A. Sundararajan, Comparing tools provided by python and r for exploratory data analysis, IJISCS (International Journal of Information System and Computer Science) 4 (2020) 131–142.

[9] A. Bloss, P. Hudak, J. Young, Code optimizations for lazy evaluation, Lisp and Symbolic Computation 1 (1988) 147–164.

[10] S. Werner, J. Kuhlenkamp, M. Klems, J. Müller, S. Tai, Serverless big data processing using matrix multiplication as example, in: Procedings International Conference on Big Data (Big Data), IEEE, 2018, pp. 358–365.

[11] V. Sreekanti, C. Wu, X. C. Lin, J. Schleier-Smith, J. M. Faleiro, J. E. Gonzalez, J. M. Hellerstein, A. Tumanov, Cloudburst: Stateful functions-as-a-service, arXiv preprint arXiv:2001.04592 (2020).

[12] H. Shafiei, A. Khonsari, P. Mousavi, Serverless computing: a survey of opportunities, challenges, and applications, ACM Computing Surveys 54 (2022) 1–32.