PyPandas: A scalable data cleaning library

Pei-Lun Liao
New York University
pll273@nyu.edu

Chia-Hsien Lin
New York University
chl566@nyu.edu

Shang-Hung Tsai
New York University
st3127@nyu.edu

ABSTRACT

This is abstract. This is abstract. This is abstract. This is abstract. This is abstract. This is abstract.

1 INTRODUCTION

1.1 Problem description and goal

In recent days, as storage devices become cheaper, storing big data in thousands of computers become easier. At that time people start to think how to mine big value from the huge dataset. The term, Big Data, was created to describe the phenomenon. As the result, data mining[1, 2] and machine learning[3] get popular in recent years. In these fields the quality of data is the key point to mine good value in the huge dataset. Hence, data cleaning become a challenge in the first step of data management and analysis[4, 5, 6].

To process huge dataset efficiently, distributed systems and frameworks [7, 8] are introduced. Spark[9] is one of the popular open source project used in industry and academia. It is built on Hadoop[7] and provide a way to manage distributed memory. PySpark[10] is an extended library for Python users to use Spark.

In the Python machine learning and data mining environment, Pandas[11] is the most popular data management library. Pandas dataframe provides a way to collect data and the data can be transform into Python primitive data structures or Numpy[12] array easily. Moreover, Pandas provides several data management methods such as missing value filling, data indexing and data profiling.

However, Pandas can only be used in a single machine and could not be scaled to manage huge dataset. In this paper, we propose PyPandas, a library built on PySpark, which support basic data management features, missing value handling or replication removing. Moreover, we add more data processing features into our library to provide user friendly functions such as outlier detection and numerical data scaling. These features are important for the machine learning tasks and are supported in Scikit-Learn, a mainstream machine learning library [13].

Furthermore, text data is another common data type in our storage. Text data are hard to clean because of multiple languages, new word, typo, abbreviation, url and emoji etc. Although we have NLTK[14], a popular text cleaning library, there is no scalable text cleaning library. In this paper, we also provide tools for users to clean text easily.

The features of our library is summarized below.

- Data management library runs on Spark
- Support missing value handling, replicated data removing, outlier detection and numerical data scaling
- Text processing such as url detection, punctuation removing, pattern searching and replacement.

2 RELATED WORK

2.1 Previous work

There exists a number of open-source library for data cleaning and parallel data frames. Optimus is framework that can perform distributed data cleaning and preprocessing. This framework works well with Spark and its DataFrame can scale in big cluster. It comes with helpful tools such as removing special characters and replacing null values. Dask is another open-source library that supports parallel analytic computing. It provides scalable parallel data structures that extend interfaces like pandas. At the same time, it offers low latency and high responsiveness. SparklingPandas library attempts to combine the power of spark and pandas to scale data analysis. It provides a Pandas-like API that is built using Spark?s DataFrame class. Unfortunately, it only support spark v1.4 and Python 2.7, and its development has ended.

3 PROPOSED METHODS

4 EXPERIMENT

4.1 Dataset

This is the dataset we will use. This is the dataset we will use. This is the dataset we will use. This is the dataset we will use. This is the dataset we will use. This is the dataset we will use.

4.2 Evaluation

We will evaluate our project by comparing it against other existing open-source libraries, including Optimus, Dask, and SparkingPandas. We will compare our data cleaning features with Optimus, particularly those frequently used functionality in data science such as null value handling, duplicate detection, etc. Furthermore, we will evaluate the performance of our implementation of parallel data management by measuring the time and space consumed because it is important to have an efficient implementation to make this library useful in production. We will also examine and compare the overall software architecture of the project. We might perform a survey to assess the usability of the library.

5 CONCLUSIONS

This is our conclusion. This is our conclusion. This is our conclusion. This is our conclusion. This is our conclusion. This is our conclusion.

6 REFERENCE

This is the reference. This is the reference. This is the reference. This is the reference. This is the reference. This is the reference.