CoClean: Collaborative Data Cleaning

Mashaal Musleh
University of Minnesota
musle005@umn.edu

Mourad Ouzzani, Nan Tang
QCRI, HBKU
{mouzzani,ntang}@hbku.edu.qa

AnHai Doan
University of Wisconsin
anhai@cs.wisc.edu

ABSTRACT

High quality data is crucial for many applications but real-life data is often dirty. Unfortunately, *automated* solutions are often not trustable and are thus seldom employed in practice. In real-world scenarios, it is often necessary to resort to manual cleaning for obtaining pristine data. Existing human-in-the-loop solutions, such as Trifacta and OpenRefine, typically involve a single user. This is often error-prone, limited to a single-person expertise, and cannot catch up with the ever growing volume, variety and veracity of data.

We propose a crowd-in-the-loop cleaning system CoClean, built on top of Python Pandas dataframe, a widely used library for data scientists. The core of CoClean is a new Python library called Collaborative dataframe (CDF) that allows one to share data represented as a dataframe with other users. CDF is responsible for synchronizing and aggregating annotations obtained from different users. The attendees will have the opportunity to experience the following features: (1) *Data Assignment:* Given a dataframe, the *owner* can assign it (or a subset of it) to different users. (2) *Supporting both lay and power users:* lay users can use a GUI to do only manual data cleaning, while power users can work on the assigned data through a Jupyter Notebook where they can write scripts to do batch cleaning. (3) *Combining machines and humans:* Possible errors and repairs generated by machine algorithms can be highlighted as annotations, which can make the life of users easier for manual cleaning. (4) *Collaboration Modes:* CoClean supports two modes: blind-on (no user can see the annotations from others) and blind-off.

CCS CONCEPTS

• **Information systems → Data cleaning**.

SIGMOD'20, June 14–19, 2020, Portland, OR, USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-6735-6/20/06…$15.00
https://doi.org/10.1145/3318464.3384698

1 INTRODUCTION

Data is a critical asset for decision-making across an organization, but data is useful only if it is of high quality. Unfortunately, real-life data is often dirty [1] and data cleaning is undeniably a laborious and time-consuming task. Despite decade-long efforts to develop *automated* data cleaning solutions [2, 4], these may not be accurate enough, especially if the output of the cleaning process is to be used in mission-critical applications. A chief reason for this inaccuracy stems from the fact that detecting and repairing data errors often require human cognitive and decision-making abilities, which may be hard to fully automate. Hence, manual data cleaning is unavoidable in many practical scenarios. However, most human-in-the-loop data cleaning tools, such as Trifacta[1] and OpenRefine[2], only interact with a single user. For data cleaning, however, no user should work in a silo, nor should she be – a collaboration from a group of users is often expected and would deliver better and faster results. This is especially true when some parts of the data can only be cleaned by specific users.

To enable collaborative data cleaning, the following features should be supported: **(F1)** Share data or a subset of data across multiple users. **(F2)** Support both lay and power users: lay users do manual data cleaning in a Google Sheets style while power users write code, for example, in a Jupyter Notebook. **(F3)** Integrate output of machine algorithms: this output, in the form of possible errors or repairs, should be fed to users as hints. **(F4)** Allow different modes of interactions among users.

Our Proposal. We present CoClean, a collaborative data cleaning system to support the above required features. CoClean is built on top of Pandas dataframe, the most popular library for data scientists to explore and analyze data. The core of CoClean is a new Python library called Collaborative dataframe (CDF), which allows one dataframe to be shared by multiple users **(F1)**. Using CDF, power users can write code as if they are working on regular dataframes **(F2)**. In addition, we offer a GUI for lay users for direct manual cleaning of the data **(F2)**. Lay users inspect data cells on the GUI to see possible errors or repairs generated by automated algorithms, and power users can see them by invoking predefined CDF APIs **(F3)**. CoClean also supports two different

[1]https://www.trifacta.com
[2]https://openrefine.org

collaboration modes **(F4)**: *blind-on* where a user cannot see the annotations from others (such as in Amazon Mechanic Turk), thus avoiding possible biases; and *blind-off* where users can see others' annotations (similar to Google Sheets).

The source code for CoClean along with example notebooks are available at https://github.com/qcri/coclean.

Related Work. (1) *Online Spreadsheet Sharing.* Sharing spreadsheets is a common way to work collaboratively on the same table. This includes tools such as Google Sheets or online Excel. Collaborators on one shared task can chat with each other, and can overwrite each other. (2) *Online Code Sharing.* Another line of work is to share code online such as Google Colab[3] or Github. They are designed for scenarios where different collaborators can work on one task in an *incremental* fashion. (3) *Crowdsourcing Platforms.* There are also Crowdsourcing platforms such as Amazon Mechanical Turk, Figure Eight[4], and Amazon SageMaker Ground Truth for machine learning labeling[5].

With regards to the required features stated earlier, (1) is designed for collaborators to work in a centralized fashion allowing them to see and overwrite each other annotations without the ability to aggregate the results. In addition, power users can write code for more advanced operations only outside these systems; (2) is to collaboratively write a program – code sharing is different from data sharing; and (3) offers limited collaboration since one cannot see annotations from others and there are no simple ways where power users could write code to annotate (flagging cells as erroneous or changing them to new values).

CoClean fills the gap left by the above systems for a true and easy to use collaborative data cleaning system where both lay users and power users can collaborate. Moreover, CoClean opens opportunities for future studies, such as offering more interaction modalities amongst users, and federated cleaning and inference that can generalize the feedback collected from multiple users.

2 AN OVERVIEW OF COCLEAN

Figure 1 gives a high level overview of how CoClean works.

Server. The server takes a table D as input, and optionally runs automated data cleaning algorithms on D, which will output either candidate data errors (*e.g.,* $D[i,j]$ is wrong), candidate data repairs (*e.g.,* the correct value of $D[i,j] = v$ should be v' where $v' \neq v$), or both. The server will then send the data to users, receive their annotations, aggregate the results, and synchronize users' actions to reflect the global annotations and aggregated results so far. Note that there are two modes: *blind-on* where users work individually and send their annotations to the server, and *blind-off* where each user can see the annotations of other users as soon as those users commit their (partial) annotations to the server.

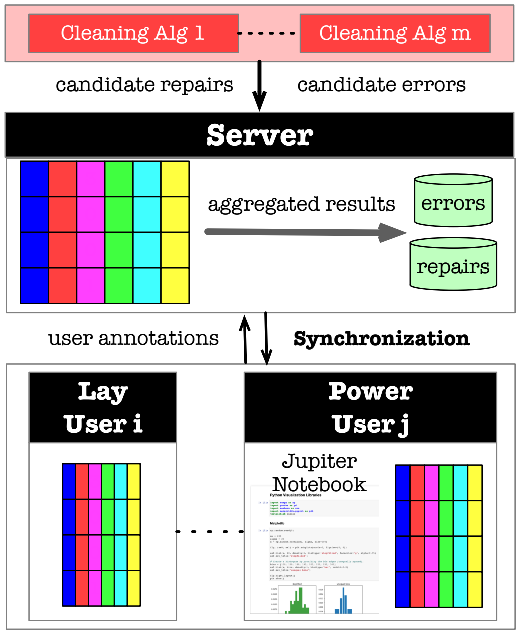


Figure 1: An Overview of CoClean

Users. Each user takes a copy of D, with any automatically generated candidate errors and candidate repairs being highlighted, indicating that these annotations are generated by machines. Each user will then produce local annotations (error flags or new values) and commit these annotations in a *small-batch* fashion. The user which first initializes the data and shares it with others is the *owner* of this data.

We consider two types of users: *lay users* who put their annotations through a GUI, similar to Google Sheets; and *power users* who annotate the data either directly using a GUI, or through writing code in a Jupyter Notebook.

GUI Interface. Users can perform two basic operations on any given cell, Flag and Update. CoClean provides lay users with a web-based GUI interface which mainly consists of a data view table and multiple facets that facilitate filtering each column and navigating across the data. Flags and updates are done via simple UI interactions such as clicks, right-clicks and typing-in the new suggested values. Depending on the blind mode (on or off), we utilize the right click context menu to show suggested flags and updates from other users, and to allow users to quickly reuse them. Color codes are used to indicate cells with suggested flags or updates and thus provide hints to guide the user to such cells for more feedback. Each dataset has a unique URL that is created by the server when shared either by the lay user through the GUI interface (by simply uploading the dataset) or by the power user when she invokes the *share* command as will be explained next.

[3]https://colab.research.google.com/
[4]https://www.figure-eight.com
[5]https://aws.amazon.com/sagemaker/

CDF Library. CoClean implements a Python library called Collaborative dataframe (CDF) to allow power users to easily share their data represented as a Pandas dataframe with other collaborators. CDF extends Python's dataframe[6] and is therefore compatible with any other data science framework that processes regular Pandas dataframes. In addition, CDF provides additional APIs that can be invoked by users to share and work collaboratively on a dataframe. Table 1 shows some of the extended CDF APIs. Intuitively, data in CDF are accessed and modified using the traditional accessibility methods of dataframes such as *at, loc, iloc*, and so on, or indexing using "*[]*" or filtering using the methods *filter, mask*, and *where*. Batch operations such as updating a whole column are done by selecting a subset of the data.

Accessing CoClean **through a Jupyter Notebook.** Power users can use CoClean through a Jupyter Notebook for more flexibility in sharing and working on the data. They can create a new CDF dataframe from CSV files, existing dataframes, or by providing the unique URL of a previously shared dataset. The created CDF will automatically connect to CoClean backend server to fetch the data and establish the collaboration session. During the annotation process, users can write code or use a GUI inside the Jupyter Notebook. For example, a user can execute the following Python snippet to clean and standardize city names `cdf[cdf.city == 'nyc'] = 'New York City'`. In all cases, CDF keeps a copy of the original dataframe and automatically captures all the updates applied to the dataframe and regularly pushes them to stay in sync with the server and other users.

Jupyter users also have a GUI interface shipped with CDF and can work simultaneously with the code side-by-side. This GUI is similar to the lay users' interface and provides similar interactions such as color indicators, filtration facets and the right-click menu to show annotations from other collaborators. CDF can be accessed directly in this view, which gives users the flexibility to perform single operations using the GUI and/or batch operation using the code and to quickly switch between them without leaving the notebook.

Aggregation Policy. Once all users commit their work, the data owner can consolidate the data through a resolution policy. CoClean currently implements basic policies such as *at least* which defines the minimum number of annotations provided by users to consider a cell as erroneous, and *majority voting* to choose between multiple updates. If a certain cell cannot be aggregated automatically, it will be returned to the owner for arbitration.

Backend Server for Synchronization. The server uses MongoDB to store flags and updates in two separate collections in the form $A[i, j, a']$ and $U[i, j, v']$ where i, j are row and column indices, respectively, and a' is the flag and v' is the new value. This simple design allows for any arbitrary flag or update. It is also easy to modify CDF and GUI interfaces to allow users to customize the annotations by providing an extra configuration of possible predefined labels to be used for annotation without the need to touch the underlying schema.

[6]https://pandas.pydata.org

API	Usage
share	Shares the dataframe and returns a unique URL for other collaborators to work on it
commit	Commits the changes to the server
resolve	Aggregates the annotations and updates based on a given policy
original_df	Shows the original shared dataframe without any input from current user
list_my_updates	Lists all updates added by the current user

Table 1: Extended CDF APIs

3 DEMONSTRATION OVERVIEW

Datasets. In this demonstration, we will use two publicly available datasets: (1) Diabetes dataset[7] which contains 9 attributes describing health readings for 769 records, and (2) Casualties dataset with more than 185k traffic accidents[8].

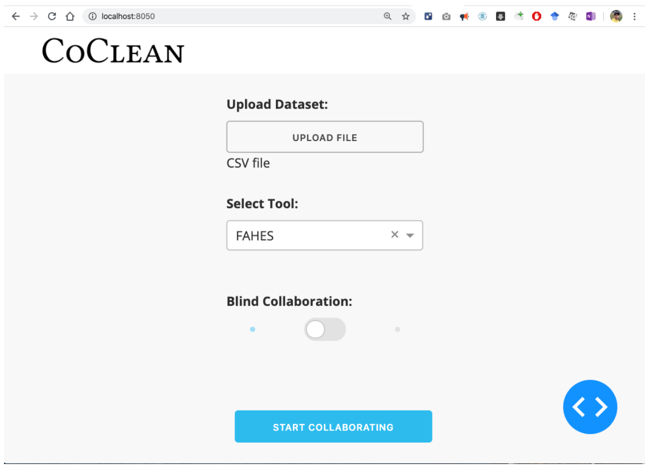
Both datasets need cleaning before they can be used further and we will use CoClean to demonstrate how multiple users can clean them collaboratively.

Demo attendees will be able to use CoClean and learn how a data owner can invite other collaborators, both lay users and power users, to work simultaneously and clean the dataset. We will use the aforementioned datasets and CoClean interfaces to demonstrate the complete process starting from data uploading until data is consolidated.

Initialization. An *owner* will upload a dataset that needs to be cleaned, where the owner could either be a lay user or a power user. Hence, we provide a GUI (Figure 2(a)) if the owner is a lay user, where he/she can upload a dataset by simply selecting a CSV file from their local storage. CoClean will return a URL that can be shared with other users. If the owner is a power user, he/she can interact with CoClean through the Collaborative DataFrame (CDF). That is, a CDF is constructed in a similar way to a regular Pandas datarame (i.e., from CSV files, web pages, databases, etc.) or from an existing dataframe. The owner can then invoke the *share* command which will connect to the CoClean server, create a copy of the data, and return a unique URL for the dataset, as shown in the first four lines in Figure 3.

Lay User Cleaning. Figure 2(b) shows the interface of a lay user for manual data cleaning. It shows a table that can be

[7]https://www.kaggle.com/uciml/pima-indians-diabetes-database
[8]https://data.gov.uk/dataset/cb7ae6f0-4be6-4935-9277-47e5ce24a11f/road-safety-data



Upload Dataset:

UPLOAD FILE

CSV file

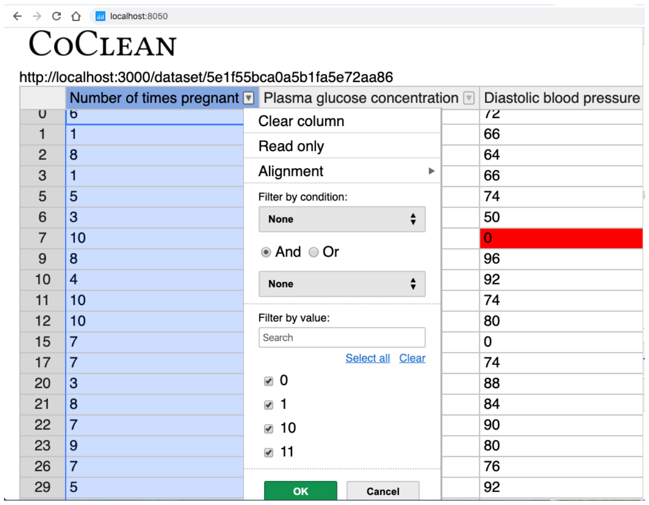
Select Tool:

FAHES

Blind Collaboration:

START COLLABORATING

(a) Data Upload



http://localhost:3000/dataset/5e1f55bca0a5b1fa5e72aa86

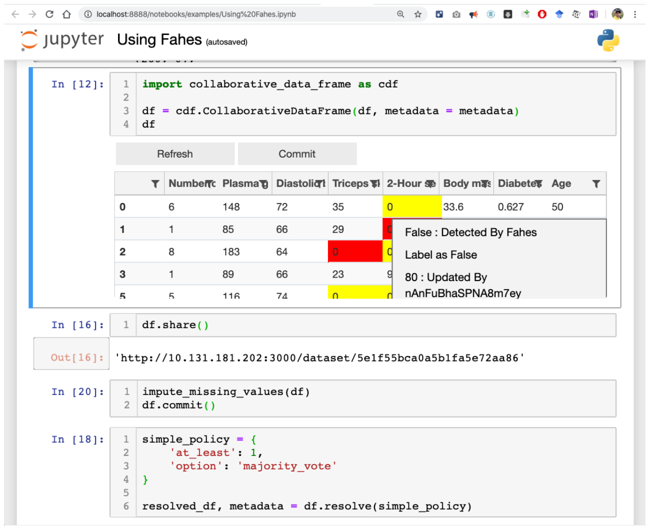
	Number of times pregnant	Plasma glucose concentration	Diastolic blood pressure
0	6		72
1	1		66
2	8		64
3	1		66
5	5		74
6	3		50
7	10		0
9	8		96
10	4		92
11	10		74
12	10		80
15	7		0
17	7		74
20	3		88
21	8		84
22	7		90
23	9		80
26	7		76
29	5		92

(b) Data View

Figure 2: GUI interface for lay users

directly updated for flagging erroneous cells or updating cell values. To facilitate navigation, each column has a drop down menu located at the header that can be used to filter column values. Different colors are used to indicate candidate cells for repairs; yellow cells in the table are the candidate erroneous cells identified by an automated tool such as FAHES [3], and red cells are the ones that has been updated by other users. User can right click on any given cell to update the annotations or see suggested values from other collaborators if the blind mode is off.

Power Users Cleaning. They interact with CoClean through the Collaborative DataFrame (CDF). They can simply create a Collaborative dataframe using the URL shared by the owner and work as if it is a normal dataframe. Power users can exploit automated data cleaning tools such as FAHES [3] (Figure 3) which detects disguised missing values and use it with CoClean to indicate candidate cells for repairs. We will show how batch operations such as updating



```
In [12]:
import collaborative_data_frame as cdf

df = cdf.CollaborativeDataFrame(df, metadata = metadata)
df
```

	Number of	Plasma g	Diastolic	Triceps sk	2-Hour se	Body mas	Diabetes	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0			
2	8	183	64	0	0			
3	1	89	66	23	9			
5	5	116	74	0	0			

```
In [16]: df.share()

Out[16]: 'http://10.131.181.202:3000/dataset/5e1f55bca0a5b1fa5e72aa86'

In [20]:
impute_missing_values(df)
df.commit()

In [18]:
simple_policy = {
    'at_least': 1,
    'option': 'majority_vote'
}

resolved_df, metadata = df.resolve(simple_policy)
```

Figure 3: Code + GUI interface for power users

multiple cells are done by executing arbitrary python code against the dataframe. CoClean will keep track of annotations and updates and log all the contributions made from each user. With the help of CDF extended APIs (Table 1), users can commit their changes or consolidate the dataset based on a given resolution policy as explained earlier.

Moreover, Figure 3 shows the Jupyter notebook-based interface which consists of code cells and a GUI interface. Code cells are for writing and executing code, and the GUI interface allows direct update to the table. Both views are side-by-side within the notebook and users have the flexibility to use them simultaneously in a customized way.

Resolving Results from Multiple Users. The owner can either invoke a default aggregation policy, as shown at the end of Figure 3, or provide their own aggregation policy through writing CDF APIs, so as to resolve the results returned from multiple users.

Conclusion. We demonstrate CoClean, a system to enable collaborative data cleaning such that both lay users and power users can contribute. A common use case is that a data scientist finds that her dataset is wrong. She can invite other users to clean. Meanwhile, she can keep exploring her dataset while the other users are collaboratively cleaning it.

REFERENCES

[1] Z. Abedjan, X. Chu, D. Deng, R. C. Fernandez, I. F. Ilyas, M. Ouzzani, P. Papotti, M. Stonebraker, and N. Tang. Detecting data errors: Where are we and what needs to be done? *PVLDB*, 9(12):993–1004, 2016.

[2] M. Mahdavi, Z. Abedjan, R. C. Fernandez, S. Madden, M. Ouzzani, M. Stonebraker, and N. Tang. Raha: A configuration-free error detection system. In *SIGMOD*, pages 865–882, 2019.

[3] A. A. Qahtan, A. Elmagarmid, R. Castro Fernandez, M. Ouzzani, and N. Tang. Fahes: A robust disguised missing values detector. In *ACM SIGKDD*, 2018.

[4] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré. Holoclean: Holistic data repairs with probabilistic inference. *PVLDB*, 10(11):1190–1201, 2017.