TraceHub - A Platform to bridge gap between State-Of-The-Art Time-Series Analytics and Datasets

Shubham Agarwal[1], Christian Muise[1], Mayank Agarwal[1], Sohini Upadhyay[2]
Zilu Tang[3], Zhongshen Zeng[2], Yasaman Khazaeni[4]
[1,2,3]{firstname.lastname}@ibm.com, [4]yasaman.khazaeni@us.ibm.com
[1]MIT-IBM Watson AI Lab
[2,4]IBM Research
[3]IBM Cloud and Cognitive Software

Abstract

In this paper, we present `TraceHub` - a platform that connects new non-trivial state-of-the-art time-series analytics with datasets from different domains. Analytics owners can run their insights on new datasets in an automated setting to find insight's potential and improve it. Dataset owners can find all possible types of non-trivial insights based on latest research. We provide a plug-n-play system as a set of *Dataset, Transformer pipeline*, and *Analytics APIs* for both kinds of users. We show a usefulness measure of generated insights across various types of analytics in the system. We believe that this platform can be used to bridge the gap between time-series analytics and datasets by significantly reducing the time to find the true potential of budding time-series research and improving on it faster.

Introduction

The ubiquity of temporal data across disciplines has drawn the attention of researchers over the last few decades. Formally, (Brockwell and Davis 1986) describes time series as a series of observations x_i, where each observation x corresponds to a specific time t. In this paper, we focus on *discrete, multivariate* time series.

`TraceHub` is a playground where separate analytics and dataset owners can plug in time-series analytics and datasets as APIs. The system then runs automated transformer pipelines to generate insights from plugged in analytics against the datasets, allowing dataset owners to access new analyses and enabling analytics owners to find new use cases and inspiration for model improvement. `TraceHub` also reports the usefulness of the generated insights.

Time-series data has been studied extensively across the fields of predictive analysis, classification, anomaly detection, clustering, and temporal data mining. The abundance of unlabeled data has precipitated a need for unsupervised approaches like clustering time series which aims to learn the distribution generating given data point, avoiding assumptions between time series (Khaleghi et al. 2016).

While the above approach tackles unsupervised time series data, the insights generated are not easily interpretable, reducing their worth to dataset owners. To best demonstrate `TraceHub`'s value to both analytics and dataset owners, we present in this paper a use-case focusing on a recent analytics that generates easily interpretable contrastive explanations between two groups of events in data using its temporal properties (Kim et al. 2019).

Architecture

`TraceHub`'s architecture consists of three main components : *Data loaders*, *Transformer Pipelines* and *Analytics*. The system offers APIs to plug in a custom dataset, transformer or analytics. Figure 1 shows the architecture overview.

Data Loaders `TraceHub` loads data by automatically detecting the type of values. It is then represented as *traces* where each *trace* is a sequence of events in time. The splitting of data into traces happens based on a certain attribute of the data specified by the user.

Transformer Pipelines `TraceHub` comes with some standard out-of-the-box data transformers like *one-hot encoding*, *missing value handling*, *quantiling*, *binarizing column*, etc. The Transformer API of `TraceHub` also allows the user to define a custom transformer performing transformation to traces. The system automates the transformer pipeline by suggesting the applicable ones on the traces. The user can then choose which of the applicable transformers to apply.

Analytics After the transformer pipeline is completed, the resulting output of traces is fed to analytics which then generates insights along with a usefulness score of it.

Evaluation

In this section, we validate and evaluate the capabilities of `TraceHub` on a simulated business process dataset of a loan processing workflow. [1]

Simulated Loan Application Dataset

We simulate a simplified loan application workflow to serve as a demonstration of `TraceHub`'s capabilities. Each loan

[1]We thank *Vatche Isahagian, IBM Research* for the dataset.

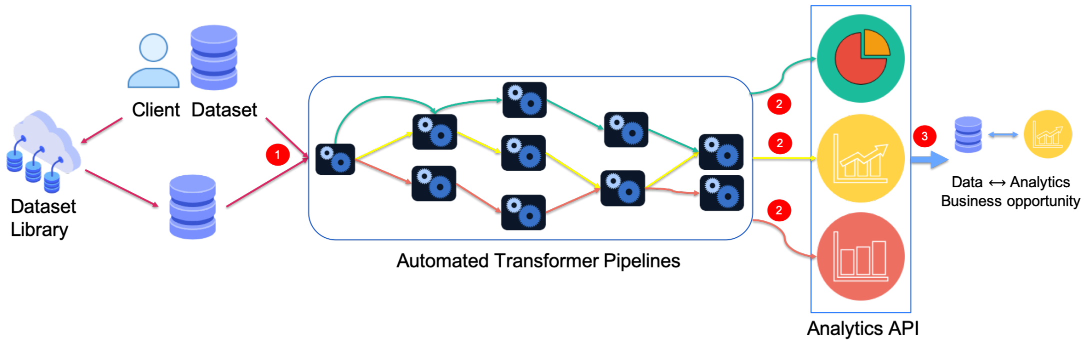

Figure 1: TraceHub architecture of dataset, transformers and analytics APIs

application is characterized by the loan type: {`Medical`, `Vacation`, `Vehicle`}, the expertise level of the agent: {`Skilled`, `Novice`}, and consists of the following events: 1) `Receive`: A new application is submitted by the user, 2) `Evaluate`: An assessor (agent) evaluates the application, 3) `Gather`: The agent decides to gather additional documents for the application, and 4) `Decision`: The agent either approves or denies the application.

An ordered occurrence of these events gives the dataset its time-series characteristics, thus making it amenable for analysis through `TraceHub` .

To serve as ground truth, we inject two distinct agent behaviors in the data: 1) A novice agent follows a simple workflow of receive, evaluate, and decide, and 2) A skilled agent tends to analyze the applications more carefully, and asks for more information encoded as the `Gather` step.

Insight generation through `TraceHub`

In this section, we walk through the entire process of generating insights through `TraceHub` .

Data loading: Since the loan application dataset is in the form of tabular data, we use the `TabularData` loader within `TraceHub` package to load it. Post data loading, we generate the set of traces by splitting the data on its `application id` column.

```
loader = TabularData()
data = loader.load(data_path="dataset.csv")
setoftraces = TraceSplit().split(data, "appid")
```

Transformer pipelines: `TraceHub` allows identification of applicable transformers on the data by simply iterating through the list of transformers and checking for their applicability. The user can then decide to apply a certain transformer using its `transform` function.

```
applicable_tfs = [tf for tf in transformers_list
                  if tf.applicable(setoftraces)]
applicable_tfs[i].transform(setoftraces, *args)
```

Insight generation: With the data ready for analysis, all applicable analytics can be queried through the `applicable_ts_analytics` function of `tracehub.analytics` subpackage. Any desired analytic method can then be applied using its `apply` method.

```
analytics = applicable_ts_analytics(setoftraces)
analytics[i].apply(setoftraces)
```

The results of contrastive LTL explanation of (Kim et al. 2019) between two groups of loan application decision *accept* and *reject* are:

```
1. Formula:  response: (activity_gather,agent_skilled),
    (agent_skilled,duration_6.0)
Meaning:  If ("activity_gather") occurs, ("agent_skilled
    ") eventually follows AND If ("agent_skilled")
    occurs, ("duration_6.0") eventually follows
Accuracy:  0.874
-------------------------------------
2. Formula:  until: agent_novice , duration_4.0
Meaning:  ("agent_novice") has to be true until ("
    duration_4.0") eventually becomes true
Accuracy:  0.77
```

The `duration_4.0` proposition refers to a simple workflow where the event `Gather` does not occur, whereas `duration_6.0` proposition refers to the workflow where the `Gather` event occurs.

The first response captures the `Skilled` agent's behavior, where the agent asks for more information through the `Gather` event before *accepting* an application, thus making the process six steps long. The second response captures the `Novice` agent's behavior, where the agent follows a simple four step workflow before *accepting* an application. The above two insights extracted through `TraceHub` match perfectly to the injected ground truth agent behaviors in the data, and provide an evidence of `TraceHub` 's capabilities to extract non-trivial insights from data. The accuracy measures of 0.874 and 0.77 are indication of usefulness of the generated insights from `TraceHub`

References

Brockwell, P. J., and Davis, R. A. 1986. *Time Series: Theory and Methods*. Berlin, Heidelberg: Springer-Verlag.

Khaleghi, A.; Ryabko, D.; Mary, J.; and Preux, P. 2016. Consistent algorithms for clustering time series. *Journal of Machine Learning Research* 17(3):1–32.

Kim, J.; Muise, C.; Shah, A.; Agarwal, S.; and Shah, J. 2019. Bayesian inference of linear temporal logic specifications for contrastive explanations. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 5591–5598. International Joint Conferences on Artificial Intelligence Organization.