TFX Frontend: A Graphical User Interface for a Production-Scale Machine Learning Platform

Extended Abstract

Peter Brandt, Josh Cai, Tommie Gannert, Pushkar Joshi, Rohan Khot, Chiu Yuen Koo, Chenkai Kuang, Sammy Leong, Clemens Mewald, Neoklis Polyzotis, Herve Quiroz, Sudip Roy, Po-Feng Yang, James Wexler, Steven Euijong Whang
Google Inc.[*]

1 INTRODUCTION

The widespread use of machine learning has highlighted the need for platforms that provide integrated tools for all phases of the machine learning development process. An end-to-end machine learning platform contains components that span from ingesting raw data to serving a model and logging its predictions. TFX [1] is one implementation of such a platform that is used for production-scale machine learning at Google.

In this extended abstract we discuss the TFX frontend, which is a unified and guided interface that supports workflows that span across all TFX components. It is the primary interface via which our users interact with TFX and inspect the result of their interactions.

2 PROPERTIES

We will illustrate[1] high-level properties of the TFX frontend that distinguish it from others.

Integrated Workflows: A TFX pipeline consists of several components. From a DevOps perspective, quickly visualizing the status of each of these components is crucial. From a modeler or data scientist's perspective, a detailed, visual analysis of the output of the components is valuable. DevOps and modeler workflows are often related (e.g. a drop in model quality may be the result of corrupted input data due to the failure of a data processing job). The TFX frontend provides an integrated and guided view of all aspects of the pipeline, according to the gestalt principle [3].

At the same time, TFX integrates with many of Google's internal services (e.g. Vizier [2]) each of which has its own UI. The TFX frontend *deep-links* to these services so that experienced users can quickly access them, and novice users are guided to relevant information without having to learn unfamiliar navigation flows.

Support for Interactive Notebooks: Data scientists and modelers often prefer interactive workflows to iterate on model structure and analyze data and models. To support these workflows, several aspects of the TFX frontend are launched as customizable interactive (iPython) notebooks. Embedding the same visualizations in interactive notebook workflows as are shown on the TFX frontend reduces cognitive load and ensures consistent use of nomenclature. To support current and future notebook experiences, we developed visualization UIs as stand-alone web components that are embedded in an the integrated TFX frontend. We optimize the notebook experience by deep-linking (pre-filling inputs) to specific notebooks.

Importance of Pipeline Health Visualization: TFX pipelines operate at "Google-scale", which means they consist of jobs that consume large amounts of resources and may be evicted by higher priority jobs intermittently. Quickly understanding job health and the existence and cause of potential failures saves precious time during debugging. For this reason, the TFX frontend offers a wide range of tools for debugging pipeline health issues.

Granularity of Data- and Model-Analysis: Data scientists and modelers require tools for data- or model-analysis at different granularity: a quick inspection of aggregate metrics v/s a detailed deep-dive at the individual example level. The TFX frontend offers tools that provide aggregate views as well as the ability to drill down into detailed and rich data- and model-analysis views.

3 WORKFLOWS

We describe features for some workflows regularly performed by TFX users via the frontend. The workflows can be categorized as: (1) pipeline management, (2) input data understanding, and (3) model analysis.

In the following subsections, each workflow is prefixed by the question users usually are trying to answer during the process of the workflow.

3.1 Pipeline Management

Pipeline Structure

"What components are part of the pipeline and what are their dependencies?"

The TFX frontend provides a graph that visualizes all configured components, their dependencies, and a set of additional information to assess pipeline status and health (discussed below). In addition to providing actionable insights into the pipeline structure that resulted from the user configuration, this visualization is also commonly used in users' design documents to communicate the anatomy of the TFX pipeline to important stakeholders.

Status and Health

"What is the pipeline status and health?"

[*]Corresponding author: Pushkar Joshi: pushkarj@google.com
[1]Screenshots will be provided at the conference & are not included here for space reasons.

SysML 2018, February 2018, Stanford, CA, USA.
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

TFX components are executed as jobs on Google's cluster management system Borg [4]. For every component, the above-mentioned dependency graph also visualizes its job health and a color-coded success/failure report of its most recent iterations. In addition, a more detailed tabular view provides status messages for each component that indicate the duration and result of its most recent iteration, and, in cases of failure, the most pertinent log messages for debugging.

Debugging and Alerts
"What part of the pipeline needs attention and what is the root cause of a potential failure?"
To help with root cause analysis, every TFX component can send debug signals to the frontend. These signals are visualized in groups based on their criticality and can be configured to alert users via email or visually in the frontend through superimposed alert boxes. Each signal is accompanied by metadata provided by the component and instructions that help with root cause analysis.

3.2 Input Data

Feature Statistics
"Do the structure and distribution of my data match my expectations?"
Aggregate statistics for each feature in the data are visualized using Facets Overview [5]. This visualization includes histogram and quantile charts, as well as statistics specific to numerical (e.g. mean, standard deviation) and categorical (e.g. string domain, mode value) features.

The visualization guides analysis by providing actionable insights such as the number of examples that are missing a specific feature (e.g. due to a failure in an upstream data processing job) or the uniformity of the feature distribution (e.g. to identify a skewed feature).

Data Samples
"Are examples structured as expected and how do they correlate with the label?"
The TFX frontend enables visualization of data samples using Facets Dive [5]. In addition to being able to see all feature values for any specific example in the data sample, Facets Dive allows users to organize the set of examples by multiple features in the dataset. Individual examples can be partitioned, positioned, and colored by different features, leading to easy creation of scatter plots, confusion matrices, and other powerful visualizations. Users can investigate relationships between features, such as seeing how features are correlated with a label feature, which can be useful in feature engineering.

Data Anomalies
"Do the structure and distribution of the data change over time?"
The TFX frontend displays data anomalies (e.g., examples with missing values, feature values that are out of range, previously unseen features) that are detected in the input data based on a schema [1], which captures the expected shape of the input data. The anomalies are visualized in a grid where each cell contains a summary of the anomalies that were identified for a feature over a specific range of input data. For each summary, the component suggests a fix to the schema in case the underlying change in the data is expected (e.g. a new feature was added).

The TFX frontend can also overlay distributions of multiple datasets in a single visualization, which allows users to investigate any skew between datasets (e.g. the value distributions between the training and serving data are different). When comparing multiple datasets, the features can be sorted by the distribution distance between the datasets being compared. This is helpful for investigating training/test skew, training/serving skew, or skew across training data from different time periods.

3.3 Model Analysis

Evaluation History
"How has the model improved over time?"
To evaluate and monitor the performance of a model over time (e.g. newly trained models or the same model over new versions of data), the TFX frontend provides time-series graphs of evaluation metrics (e.g. logistic loss, AUROC) relevant to the machine learning task (e.g. binary classification). This visualization can also be used to compare quality metrics of different models or specific data slices (see below) over time and is commonly used in making launch decisions for new models.

Evaluation Details
"How consistent is the model quality over all examples?"
In addition to computing quality metrics over the entire dataset, the TFX evaluation component can be configured to produce quality metrics sliced by features or specific feature values. The TFX frontend provides an interactive component that allows data scientists and modelers to visualize a histogram of metrics for different values of a feature (e.g. for all values of the feature 'country') and to drill down into the distribution of metrics for a specific feature value (e.g. for features with the value 'country:US').

Validation
"Does the model meet product and serving infrastructure requirements?"
TFX validates models both for quality (to meet product requirements) and compatibility with the serving system (to meet compatibility requirements). The results of these validations are visualized over time as new models are trained. In a time-series graph, users can visually inspect the validation result and whether it is above or below the configured threshold.

4 IMPLEMENTATION

The TFX frontend consists of a Polymer web application backed by a Go web server. The web server handles REST requests by connecting to a pipeline-specific Go server ("Controller"). The Controller serves three main purposes. It (1) authenticates the users and ensures they have access to the pipeline storage, jobs, and services, (2) serves requests to route the outputs of TFX components from the pipeline storage to the web server, and (3) serves requests to route information like pipeline state, job health, and job status from Google infrastructure services to the web server.

Upon loading the page the TFX frontend asynchronously sends XmlHttpRequests to the Controller to curate all of the information and data needed for a particular view. We assume that the user is on an unmetered, high bandwidth network and viewing the UI on a (large) desktop screen.

REFERENCES

[1] Denis Baylor, Eric Breck, Heng-Tze Cheng, Noah Fiedel, Chuan Yu Foo, Zakaria Haque, Salem Haykal, Mustafa Ispir, Vihan Jain, Levent Koc, Chiu Yuen Koo, Lukasz Lew, Clemens Mewald, Akshay Naresh Modi, Neoklis Polyzotis, Sukriti Ramesh, Sudip Roy, Steven Euijong Whang, Martin Wicke, Jarek Wilkiewicz, Xin Zhang, and Martin Zinkevich. 2017. TFX: A TensorFlow-Based Production-Scale Machine Learning Platform. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17)*. ACM, New York, NY, USA, 1387–1395. https://doi.org/10.1145/3097983.3098021

[2] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Elliot Karro, and D. Sculley (Eds.). 2017. *Google Vizier: A Service for Black-Box Optimization.* http://www.kdd.org/kdd2017/papers/view/google-vizier-a-service-for-black-box-optimization

[3] Kayur Patel, Naomi Bancroft, Steven M. Drucker, James Fogarty, Andrew J. Ko, and James Landay. 2010. Gestalt: Integrated Support for Implementation and Analysis in Machine Learning. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology (UIST '10)*. ACM, New York, NY, USA, 37–46. https://doi.org/10.1145/1866029.1866038

[4] Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. 2015. Large-scale Cluster Management at Google with Borg. In *Proceedings of the Tenth European Conference on Computer Systems (EuroSys '15)*. ACM, New York, NY, USA, Article 18, 17 pages. https://doi.org/10.1145/2741948.2741964

[5] James Wexler and Jimbo Wilson. 2017. Facets: Visualizations for machine learning datasets. (2017). https://github.com/pair-code/facets