Disdat: Bundle Data Management for Machine Learning Pipelines

Ken Yocum, Sean Rowan, and Jonathan Lunt, ***Intuit, Inc.;***
Theodore M. Wong, ***23andMe, Inc.***

https://www.usenix.org/conference/opml19/presentation/yocum

This paper is included in the Proceedings of the 2019 USENIX Conference on Operational Machine Learning (OpML '19).

May 20, 2019 • Santa Clara, CA, USA

ISBN 978-1-939133-00-7

Open access to the Proceedings of the 2019 USENIX Conference on Operational Machine Learning is sponsored by USENIX.

Disdat: Bundle Data Management for Machine Learning Pipelines

Ken Yocum
Intuit, Inc.

Sean Rowan
Intuit, Inc.

Jonathan Lunt
Intuit, Inc.

Theodore M. Wong
23andMe, Inc.

Abstract

Modern machine learning pipelines can produce hundreds of data artifacts (such as features, models, and predictions) throughout their lifecycle. During that time, data scientists need to reproduce errors, update features, re-train on specific data, validate / inspect outputs, and share models and predictions. Doing so requires the ability to publish, discover, and version those artifacts.

This work introduces *Disdat*, a system to simplify ML pipelines by addressing these data management challenges. Disdat is built on two core data abstractions: *bundles* and *contexts*. A bundle is a versioned, typed, immutable collection of data. A context is a sharable set of bundles that can exist on local and cloud storage environments. Disdat provides a bundle management API that we use to extend an existing workflow system to produce and consume bundles. This bundle-based approach to data management has simplified both authoring and deployment of our ML pipelines.

1 Introduction

Managing data artifacts associated with ML pipelines remains challenging for data scientists, even with existing tools for code versioning, continuous deployment, and application container execution. The development and deployment lifecycle of a pipeline may create thousands of artifacts, including features, trained models, and predictions. At any point in time, the data science team may need to share inputs to reproduce errors, re-train on specific data, or validate model behavior.

Naming and storing data artifacts is frequently an ad-hoc and error-prone process in which data is managed per project, found via tribal knowledge, and shared by e-mail or instant messaging. This leads to significant data scatter across local computers (such as laptops) and cloud storage (such as AWS S3 [4]). Worse, data science team members often convolve naming and versioning. For example, where one expects a logical name like `financials` for a data set, one instead finds a taxonomy of names like `financials_v_1-20190520`.

We introduce *Disdat*, a system that leverages two practical abstractions–the *bundle* and *context*–to strike a balance between prescription and the need for data scientists to use the latest tools when authoring and deploying ML pipelines. The bundle is a named collection of files and literals, and is the unit at which data is produced, versioned, and consumed. The context is a view abstraction that gathers together one or more bundles, and assists with managing bundles across multiple locations. Bundles and contexts are minimally prescriptive in the same sense as high-level pipelining systems such as Luigi [9], Airflow [1], and Pinball [8] that encode dependencies between user-defined tasks.

Bundles and contexts in Disdat together support common data science activities. Conceptually, Disdat accomplishes for data what Docker does for application images. Bundles allow users to find the latest version of related pipeline data with a single "human" name instead of parsing ad-hoc names. Contexts facilitate simple sharing and synchronization of bundles between different users and across local and cloud storage locations through intuitive "push"/"pull" operations.

Disdat stands in contrast to existing systems for managing pipeline data artifacts. Many are closed, monolithic ecosystems, providing pipeline authoring, model versioning, deployment, feature storage, monitoring, and visualization. Examples include Palantir's Foundry [6], Facebook's FBLearner [3], and Uber's Michelangelo and PyML [10]. Perhaps closer in spirit to Disdat are MLFlow [2], Pachyderm [5], and DVC [7], which aim to version pipeline experiments to enable reproducibility.

Unlike prior approaches, Disdat treats bundles as first-class citizens. Where Pachyderm and DVC support `git`-like operations, Disdat eschews some version control concepts, such as branching and merging, whose semantics for ML artifacts remain an open question (e.g., merging ML model weights between branches). In addition, their units of data versioning are implementation specific; each Pachyderm container produces a single commit to a "repository", while DVC relies on an extant `git` repository to version DVC metadata. Like Disdat, the MLFlow API captures parameters and data outputs, but

users must still organize and manage their data.

The core of Disdat consists of an API to create and publish bundles in contexts. We use that API to instrument the Luigi pipelining system from Spotify [9], allowing data scientists to author pipelines that automatically produce bundles. By virtue of this design, Disdat pipelines automatically re-use prior results where possible and can publish new versions of data products on a cloud storage service.

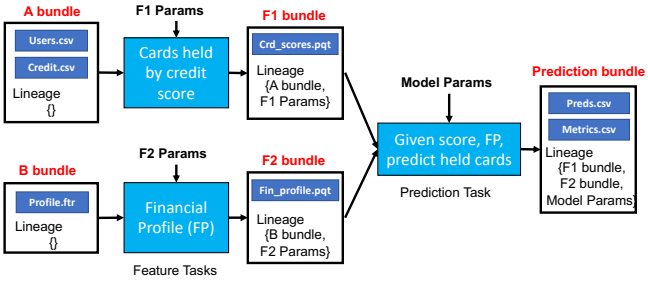

Figure 1: An ML pipeline with two featurization tasks feeding a predictive model and producing three output bundles.

2 Motivating Example

We motivate Disdat's design with a simple data processing scenario. Consider a financial services company wishing to predict credit card ownership among users. To do so, it creates a three-task ML pipeline shown in Figure 1. This pipeline featurizes the input data and applies a trained model to assign a likelihood of ownership to each user.

In general, ML pipelines consist of tasks that read and write one or more files. For example, the first featurization step ("F1") in Figure 1 reads two `.csv` files describing the user population and produces a Parquet `.pqt` feature file. The model task consumes the features to produce predictions and performance metrics. Data scientists may re-run individual tasks or the whole pipeline many times to explore features, fix bugs, and tune hyper-parameters.

Many challenges face the data scientist in managing the flow of data through this example pipeline. They must create a naming and file system directory scheme that disambiguates input, intermediate, and output files relating to different populations, which usually results in names like `users-popA.csv` and `users-popB.csv`. They often incorporate ad-hoc versioning to track updates to populations or pipeline code changes, which leads to clumsily embedded metadata such as `users-popA-20190520.csv` or `Crd_scores-with-low-score-cutoff.pqt`. Lastly, sharing and re-using data requires mechanisms to find artifacts across local and cloud locations as well as polices to define "latest" among multiple versions of the same artifact.

Disdat builds on *bundles* and *contexts* to address this challenge. Bundles organize collections of data items flowing through pipelines; thus, each task in Figure 1 produces a single bundle. A bundle is an immutable set of tags, lineage information, and named arrays. Each named array may store *scalar-typed* data, *file links*, or *pointers* to bundles. File links are references to files, such as POSIX filenames or S3 URLs.

In Figure 1, the "Prediction" bundle has one named array with two file links. When Disdat creates the bundle, it places the files and bundle metadata in the current context (on the local file system). A context serves as an organizational unit for bundles–the user decides whether the context represents a project, pipeline, or data in a test or deploy environment. Contexts hold any number of bundles and can exist at different *locations*–the local file system and a cloud storage service.

Disdat bundles provide three distinct names by which to distinguish data versions. These are a *human_name*, *processing_name*, and a UUID. The *human_name* indicates the logical data use; it supports data sharing among colleagues. In our example, the final output bundle may have *human_name* `card_predictions`. The *processing_name* is a unique string computed from the parameterized task; it allows a pipeline to re-use the most recent upstream task's output.

Note that each pipeline execution can produce bundles with the same *human_name*, but that differ by UUID and creation date. Thus synchronization between local and cloud locations is as simple as downloading bundles whose UUIDs are not present. This allows the data scientist to easily get the latest version either from a local context or one hosted on AWS S3.

3 Discussion

Disdat is a Python-based system consisting of an API for creating and managing bundles in contexts, a command-line interface, and an instrumented pipelining system. Disdat uses the API to extend Spotify's Luigi so that tasks transparently ingest bundles and produce bundles as output. In addition, Disdat can *dockerize* pipelines to run on container execution services like AWS Batch or AWS SageMaker.

At Intuit, we use Disdat for batch prediction pipelines and have found this approach valuable. Sometimes data scientists may not access raw data on their laptops or their laptop may have insufficient resources. During development, Disdat makes it easy to run that portion of a pipeline on the cloud and retrieve the output to test locally. Similarly, errors often occur during large-scale tests, and it is easy to find the set of input bundles that caused failures. Bundles have also simplified performance monitoring, as any data scientist may pull all versions of a pipeline's outputs for analysis (for example, in a notebook via the Disdat API).

4 Availability

Disdat is open-source (ASL 2.0) software available on github at `http://github.com/kyocum/disdat`.

References

[1] M. Beauchemin. Airflow. https://airflow.apache.org/, 2014.

[2] Databricks. Mlflow. https://mlflow.org/, 2019.

[3] Facebook. Introducing FBLearner Flow: Facebook's AI backbone. https://code.facebook.com/posts/1072626246134461/introducing-fblearner-flow-facebook-s-ai-backbone/, May 2016.

[4] A. Halevy, F. Korn, N. F. Noy, C. Olston, N. Polyzotis, S. Roy, and S. E. Whang. Goods: Organizing Google's datasets. *SIGMOD*, 2016.

[5] Pachyderm. Pachyderm. https://pachyderm.readthedocs.io/, 2019.

[6] Palantir. Foundry. https://www.palantir.com/palantir-foundry/, 2018.

[7] D. Petrov. Data version control. https://blog.dataversioncontrol.com/data-version-control-beta-release-iterative-machine-learning-a7faf7c8be67, May 2017.

[8] Pinterest. Pinball. https://github.com/pinterest/pinball, 2019.

[9] Spotify. Luigi. https://github.com/spotify/luigi.org, 2016.

[10] Uber. Michelangelo. https://eng.uber.com/michelangelo/, September 2017.