Declarative Metadata Management: A Missing Piece in End-To-End Machine Learning

Sebastian Schelter, Joos-Hendrik Böse, Johannes Kirschnick, Thoralf Klein, Stephan Seufert
Amazon
{sseb,jooshenb,kirschnj,thoralfk,seufert}@amazon.com

ABSTRACT

We argue for the necessity of managing the metadata and lineage of common artifacts in machine learning (ML). We discuss a recently presented lightweight system built for this task, which accelerates users in their ML workflows, and provides a basis for comparability and repeatability of ML experiments. This system tracks the lineage of produced artifacts in ML workloads and automatically extracts metadata such as hyperparameters of models, schemas of datasets and layouts of deep neural networks. It provides a general declarative representation of common ML artifacts, is integrated with popular frameworks such as MXNet, SparkML and scikit-learn, and meets the demands of various production use cases at Amazon.

ACM Reference Format:
Sebastian Schelter, Joos-Hendrik Böse, Johannes Kirschnick, Thoralf Klein, Stephan Seufert. 2018. Declarative Metadata Management: A Missing Piece in End-To-End Machine Learning. In *Proceedings of SysML Conference, Stanford, USA, Feb 2018 (SYSML'18),* 3 pages.
https://doi.org/10.475/123_4

1 INTRODUCTION

When developing and productionizing ML models, a major proportion of the time is spent on conducting model selection experiments which consist of training and tuning models and their corresponding features [2, 3, 13, 19, 22, 27]. Typically, data scientists conduct this experimentation in an ad-hoc style without a standardized way of storing and managing the resulting experimentation data and artifacts. As a consequence, the results of these experiments are often not comparable, as there is no standard way to determine whether two models had been trained on the same input data, for example. Even more, it is tedious and time-consuming to repeat successful experiments later in time, and it is hard to get an overall picture of the progress made in ML tasks towards a specific goal, especially in larger teams. Simply storing the artifacts (datasets, models, feature sets, predictions) produced during experimentation in a central place is unfortunately insufficient to mitigate this situation. Achieving repeatability and comparability of ML experiments forces one to understand the metadata and, most importantly, the lineage of artifacts produced in ML workloads [13]. For example, in order to re-use a persisted model, it is not sufficient to restore its contents byte by byte; new input data must also be transformed into a feature representation that the model can understand, so information on these transforms must also be persisted. As another example, in order to reliably compare two experiments, we must ensure that they have been trained and evaluated using the same training and test data respectively, and that their performance was measured by the same metrics.

To address the aforementioned issues and assist data scientists in their daily tasks, we proposed a lightweight system for handling the metadata of ML experiments [20]. This system allows for managing the metadata (e.g., *Who created the model at what time? Which hyperparameters were used? What feature transformations have been applied?*) and lineage (e.g., *Which dataset was the model derived from? Which dataset was used for computing the evaluation data?*) of produced artifacts, and provides an entry point for querying the persisted metadata. Data scientists can leverage this service to enable a variety of previously hard-to-achieve functionality, such as regular automated comparisons of models in development to older models (similar to regression tests for software). Additionally, the proposed service helps data scientists to easily ad-hoc test their models in development and provides a starting point for quantifying the accuracy improvements that teams achieve over time towards a specific ML goal, e.g., by storing and analyzing the evaluation results of their models and showing them via a leaderboard. In order to ease the adoption of our metadata tracking system, we explore techniques to automatically extract experimentation metadata from common abstractions used in ML pipelines, such as 'data frames' which hold denormalized relational data, and ML pipelines which comprise a way to define complex feature transformation chains composed of individual operators. For applications built on top of these abstractions, metadata tracking should not require more effort than exposing a few data structures to our tracking code.

In the following, we summarize the design decisions for our system (Section 2), list related work (Section 3) and discuss future research directions for ML metadata management (Section 4).

2 SYSTEM DESIGN

Data model. The major challenge in designing a data model for experimentation metadata is the *trade-off between generality and interpretability of the schema.* The most general solution would be to simply store all data as bytes associated with arbitrary key-value tags. Such metadata however would be very hard to automatically interpret and analyze later, as no semantics are enforced. A too narrow schema on the other hand might hinder adoption of our service, as it does not allow scientists to incorporate experimentation data from a variety of use cases. We propose a middle ground with a schema[1] that strictly enforces the storage of lineage information (e.g., which dataset was used to train a model) and ML-specific

[1] available at https://github.com/awslabs/ml-experiments-schema

SYSML'18, Feb 2018, Stanford, USA

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

attributes (e.g., hyperparameters of a model), but still provides flexibility to its users by supporting arbitrary application-specific annotations. The most important principle we embrace is *declarativity*: we store metadata of the artifacts but not code that produces it, and only store pointers to the actual input data or serialized parameters. This enforces a strict decoupling, and enables querying and analysis of the metadata and lineage. The second important principle for our system is *immutability*: metadata entries are only written once, ruling out a variety of potential consistency problems.

Architecture. Our system employs a three-layered architecture: On the lowest layer, a document database stores the actual experimentation data. In the next layer, this centralized data store is exposed to the outside world via a REST API, for which we provide so-called low-level clients for the JVM and Python, which allow users to explicitly store metadata for particular artifacts and query the existing data. The uppermost layer is formed by what we call high-level clients that are geared towards popular ML libraries such as *SparkML* [14], *scikit-learn* [17], *MXNet* [4].

Automated Metadata Extraction. Our high-level clients enable automated metadata extraction from internal data structures of popular ML frameworks. ML workloads in SparkML for example are comprised of pipeline stages which operate on `DataFrames`, a relational abstraction for a partitioned table with a well-defined schema. The architecture of SparkML pipelines allows us to automatically track all the schema transformations (e.g, reading, adding and removing columns) each pipeline operator conducts, as well as the parameterization of the operators. We create and store directed acyclic graph representations of Spark pipelines (where edges denote pipeline operators and vertices correspond to dataframe columns) by extracting the schema of the input data frame and replaying the schema changes the pipeline conducts. Frameworks for deep neural networks offer their users a very fine-grained abstraction to declaratively define their models by combining mathematical operators (e.g., linear layers, convolutions, activation functions) into the layout of the network to learn. Our high-level client for MXNet [4] for example extracts and stores the resulting computational graph, together with the parameterization and dimensionality of the contained operators, and corresponding hyperparameters such as optimizer settings.

3 RELATED WORK

In order to foster collaboration between scientists, platforms such as *OpenML* [25] and the *W3C ML Schema* initiative [10] allow researchers to share descriptions and evaluation results of their ML experiments. Managing and efficiently executing model selection workloads has been identified as an upcoming challenge [11, 13, 21] in the data management community. The *ModelDB* [26] project puts a specific focus on organizing models, and comes very close to our system design-wise with the difference that we support more general classes of models and apply more detailed tracking. Other classes of systems specialize on deep learning [16], aim at efficiently serving the resulting models for prediction [5, 6] or concentrate on tracking and indexing provenance information [7, 15, 18]. Modeling ML workloads via pipelines (which are typically inspired by the 'estimator/transformer' abstraction in *scikit-learn* [17]) and efficiently executing such pipelines at scale has become an active area of research. Established work describes production systems and platforms [2, 3], investigates software engineering aspects [22, 24] and pipeline abstractions for machine learning workloads [1], often on top of the massively parallel dataflow system *Apache Spark* [14, 23].

4 OUTLOOK

We see a huge potential in enabling declarative management of ML metadata: data scientists are provided with infrastructure that allows them to accelerate their experimentation via dashboards and leaderboards that list experiments, email notifications which summarize experimentation progress, and automated regression tests for the prediction quality of ML models during development, which compare the results on hold out data with historical prediction results, e.g. upon every commit to the codebase. In the remainder, we elaborate on research directions and upcoming challenges for ML metadata management systems:

Replicable Model Training and Deployment. Systems like ours allow companies to accelerate their experimentation and innovation cycle, and have the potential to form a corner stone of replicable ML model training, which will become more important in the light of upcoming legal requirements for the real-world usage of machine learning. In order to enable such a replicable model training, it is not sufficient to be able to access the metadata of the ML workload, additionally the training source code (e.g., via the corresponding git commit) as well as the computational environment (e.g., via a docker image) need be tracked and stored. Ideally such a system would not only automate replicable model training but also integrate the resulting models with model serving systems for easy deployment.

Further Automation of Metadata Tracking. Although we provide elaborate extraction functionality, we currently still rely on users understanding our complex schema and correctly integrating their code with our API. We aim to increase the automation of our extraction code and to decrease the amount of additional code and effort required to enable the metadata tracking in a workload. One direction to investigate is the instrumentation of notebooks like *Jupyter* [12], where we would ideally assist the user interactively in tracking ML metadata during explorative model tuning efforts [18]. Another orthogonal direction would be to extract ML metadata posthoc from logfiles produced by model training systems.

Meta Learning. Our long term research goals include enabling meta learning [8] on top of our experiment repository, e.g. to recommend features, algorithms or hyperparameter settings for new datasets. This would require us to implement the automated computation of *metafeatures* [9] for contained datasets, as well as similarity queries allowing users to find the most similar datasets for new data, based on these metafeatures. It would furthermore be beneficial to additionally leverage data from open repositories such as *OpenML* [25] for this task.

REFERENCES

[1] Pierre Andrews, Aditya Kalro, Hussein Mehanna, and Alexander Sidorov. 2016. Productionizing Machine Learning Pipelines at Scale. *Machine Learning Systems workshop at ICML* (2016).

[2] Denis Baylor, Eric Breck, Heng-Tze Cheng, Noah Fiedel, Chuan Yu Foo, Zakaria Haque, Salem Haykal, Mustafa Ispir, Vihan Jain, Levent Koc, et al. 2017. TFX: A TensorFlow-Based Production-Scale Machine Learning Platform. In *KDD*. 1387–1395.

[3] Joos-Hendrik Böse, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Dustin Lange, David Salinas, Sebastian Schelter, Matthias Seeger, and Yuyang Wang.

2017. Probabilistic Demand Forecasting at Scale. *PVLDB* 10, 12 (2017), 1694–1705.
[4] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. 2015. MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. *Machine Learning Systems workshop at NIPS* (2015).
[5] Daniel Crankshaw, Peter Bailis, Joseph E Gonzalez, Haoyuan Li, Zhao Zhang, Michael J Franklin, Ali Ghodsi, and Michael I Jordan. 2015. The missing piece in complex analytics: Low latency, scalable model management and serving with velox. *CIDR* (2015).
[6] Daniel Crankshaw, Xin Wang, Guilio Zhou, Michael J Franklin, Joseph E Gonzalez, and Ion Stoica. 2017. Clipper: A Low-Latency Online Prediction Serving System.. In *NSDI*. 613–627.
[7] Deepsense.ai. 2017. Neptune - Machine Learning Lab, https://neptune.ml/. (2017). https://neptune.ml/
[8] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and robust automated machine learning. In *NIPS*. 2962–2970.
[9] Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. 2015. Initializing Bayesian Hyperparameter Optimization via Meta-Learning.. In *AAAI*. 1128–1135.
[10] Machine Learning Schema Community Group. 2017. W3C Machine Learning Schema. (2017). https://www.w3.org/community/ml-schema/
[11] Joseph M Hellerstein, Vikram Sreekanti, Joseph E Gonzalez, James Dalton, Akon Dey, Sreyashi Nag, Krishna Ramachandran, Sudhanshu Arora, Arka Bhattacharyya, Shirshanka Das, et al. 2017. Ground: A Data Context Service.. In *CIDR*.
[12] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B Hamrick, Jason Grout, Sylvain Corlay, et al. 2016. Jupyter Notebooks-a publishing format for reproducible computational workflows.. In *ELPUB*. 87–90.
[13] Arun Kumar, Robert McCann, Jeffrey Naughton, and Jignesh M Patel. 2015. Model Selection Management Systems: The Next Frontier of Advanced Analytics. *SIGMOD Record* (2015).
[14] Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, et al. 2016. Mllib: Machine learning in apache spark. *JMLR* 17, 34 (2016), 1–7.
[15] Hui Miao, Ang Li, Larry S Davis, and Amol Deshpande. 2017. On Model Discovery For Hosted Data Science Projects. In *Workshop on Data Management for End-to-End Machine Learning at SIGMOD*. 6.
[16] Hui Miao, Ang Li, Larry S Davis, and Amol Deshpande. 2017. Towards unified data and lifecycle management for deep learning. In *ICDE*. 571–582.
[17] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *JMLR* 12 (2011), 2825–2830.
[18] Joao Felipe Pimentel, Leonardo Murta, Vanessa Braganholo, and Juliana Freire. 2017. noWorkflow: a tool for collecting, analyzing, and managing provenance from python scripts. *VLDB* 10, 12 (2017), 1841–1844.
[19] Neoklis Polyzotis, Sudip Roy, Steven Euijong Whang, and Martin Zinkevich. 2017. Data Management Challenges in Production Machine Learning. In *SIGMOD*. ACM, 1723–1726.
[20] Sebastian Schelter, Joos-Hendrik Boese, Johannes Kirschnick, Thoralf Klein, and Stephan Seufert. 2017. Automatically Tracking Metadata and Provenance of Machine Learning Experiments. *Machine Learning Systems workshop at NIPS* (2017).
[21] Sebastian Schelter, Juan Soto, Volker Markl, Douglas Burdick, Berthold Reinwald, and Alexandre Evfimievski. 2015. Efficient sample generation for scalable meta learning. In *ICDE*. 1191–1202.
[22] D Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, and Dan Dennison. 2015. Hidden technical debt in machine learning systems. In *NIPS*. 2503–2511.
[23] Evan R Sparks, Shivaram Venkataraman, Tomer Kaftan, Michael J Franklin, and Benjamin Recht. 2017. KeystoneML: Optimizing Pipelines for Large-Scale Advanced Analytics. *ICDE* (2017).
[24] Tom van der Weide, Dimitris Papadopoulos, Oleg Smirnov, Michal Zielinski, and Tim van Kasteren. 2017. Versioning for End-to-End Machine Learning Pipelines. In *Workshop on Data Management for End-to-End Machine Learning at SIGMOD*. 2.
[25] Joaquin Vanschoren, Jan N Van Rijn, Bernd Bischl, and Luis Torgo. 2014. OpenML: networked science in machine learning. *SIGKDD* 15, 2 (2014), 49–60.
[26] Manasi Vartak, Harihar Subramanyam, Wei-En Lee, Srinidhi Viswanathan, Saadiyah Husnoo, Samuel Madden, and Matei Zaharia. 2016. ModelDB: A System for Machine Learning Model Management. In *Workshop on Human-In-the-Loop Data Analytics at SIGMOD*. 14:1–14:3.
[27] Martin Zinkevich. 2017. Rules of Machine Learning: Best Practices for ML Engineering. (2017). http://martin.zinkevich.org/rules_of_ml/rules_of_ml.pdf