Apache Liminal

Abstract

Apache Liminal is an end-to-end platform for data engineers & scientists, allowing them to build, train and deploy machine learning models in a robust and agile way. The platform provides the abstractions and declarative capabilities for data extraction & feature engineering followed by model training and serving. Apache Liminal's goal is to operationalise the machine learning process, allowing data scientists to quickly transition from a successful experiment to an automated pipeline of model training, validation, deployment and inference in production, freeing them from engineering and non-functional tasks, and allowing them to focus on machine learning code and artifacts.

Rationale

The challenges involved in operationalizing machine learning models are one of the main reasons why many machine learning projects never make it to production.

The process involves automating and orchestrating multiple steps which run on heterogeneous infrastructure - different compute environments, data processing platforms, ML frameworks, notebooks, containers and monitoring tools.

There are no mature standards for this workflow, and most organizations do not have the experience to build it in-house. In the best case, dev-ds-devops teams form in order to accomplish this task together; in many cases, it's the data scientists who try to deal with this themselves without the knowledge or the inclination to become infrastructure experts.

As a result, many projects never make it through the cycle. Those who do suffer from a very long lead time from a successful experiment to an operational, refreshable, deployed and monitored model in production.

The goal of Apache Liminal is to simplify the creation and management of machine learning pipelines by data engineers & scientists. The platform provides declarative building blocks which define the workflow, orchestrate the underlying infrastructure, take care of non functional concerns, enabling focus in business logic / algorithm code.

Some Commercial E2E solutions have started to emerge in the last few years, however, they are limited to specific parts of the workflow, such as Databricks MLFlow. Other solutions are tied to specific environments (e.g. SageMaker on AWS).

Proposal

Liminal platform is aimed to provide data engineers & scientists with a solution for end to end flows from model training to real time inference in production. It’s architecture enables and promotes adoption of specific components in existing (non-Liminal) frameworks, as well as seamless integration with other open source projects. Liminal was created to enable scalability in ML efforts and after a thorough review of available solutions and frameworks, which did not meet our main KPIs:

- Provide an opinionated but customizable end-to-end workflow
- Abstract away the complexity of underlying infrastructure
- Support major open source tools and cloud-native infrastructure to carry out many of the steps
- Allow teams to leverage their existing investments or bring in their tools of choice into the workflow

We have found that other tech companies in the Israeli Hi-Tech ecosystem also have an interest in such a platform, hence decided to share our work with the community.

A classical data scientist workflow includes some base phases: Train, Deploy and Consume.

- The Train phase includes the following tasks:
  - Fetch - get the data needed to build a model - usually using SQL
  - Clean - make sure the data is useful for building the model
  - Prepare - split data and encode features from the data according to model needs
  - Train - Build the model and tune it
  - Evaluate - make sure the model is correct - run it on a test set, etc…
  - Validate - make sure the model is up to the standards you need

- The Deploy phase includes these tasks:
  - Deploy - make it available for usage in production
  - Inference - Batch or Real-time - use the model to evaluate data by your offline or online by your applications

- Consume - The actual use of the models created by applications and ETLs, usually through APIs to the batch or real-time inference that usually rely on Model and Feature stores.

Liminal provides its users a declarative composition capabilities to materialize these steps in a robust way, while exploiting existing frameworks and tools. e.g. Data science frameworks such as scikit-learn, Tensor flow, Keras and such, for running core data science algorithms; as numerous core mechanisms as data stores, processing engines, parallelism, schedulers, code deployment as well as batch and real-time inference.

Liminal allows the creation and wiring of these kinds of functional and non functional tasks while making the underlying infrastructure used by these tasks very easy to use and even abstracted away entirely. While handling the non-functional aspects as monitoring (in a standard fashion) deployment, scheduling, resource management and execution.
Data pipeline task types

Data science task types - programs based on data scientist code.
Example of tasks that users can create in a data science pipeline:

1. Data preparation:
   a. SQL query
   b. Source configuration
   c. Sink configuration
2. Data transformation
3. Model training
4. Model deploy
5. Batch inference/execution
6. Update model for real-time inference/execution service
7. ETL applications - YARN/EMR applications based on user’s code.

Service types

1. HTTP server - HTTP server with endpoints mapped to user code. Useful for real time inference/execution of user code by their clients.

Non-functional

The platform will take care of non-functional aspects of users’ applications, namely:

1. Build
2. Deploy
3. Parallelism (via data partitioning)
4. Job status metrics
5. Alerting

API

Liminal will introduce a declarative API to define data pipelines and services which references user code by module/path.

UI

Liminal’s UI will allow users of Liminal to create pipelines via a web ui (as well as REST API).

For example:

1. Allow a user to define a pipeline via filling a web form.
2. Allow a user to define tasks via filling a web form. For example: define an SQL ETL task with dropdowns for available sources/sinks/"tables" in the organization.

Initial Goals

Our initial goals are to bring Liminal into the ASF, transition internal engineering processes into the open, and foster a collaborative development model according to the "Apache Way."

Current Status

Liminal is in development, leveraging existing Apache projects:

Currently mainly using Apache Airflow in order to run pipelines defined by users in Liminal APIs (YAML, with plans of UI/REST), this reduces the engineering requirements for transitioning data science code into production. We also leverage Apache Spark and Apache Hive for data preparation features and there are plans to integrate with Apache Karaf as well.

The current license is already Apache 2.0.

Meritocracy

We intend to radically expand the initial developer and user community by running the project in accordance with the "Apache Way". Users and new contributors will be treated with respect and welcomed. By participating in the community and providing quality patches/support that move the project forward, they will earn merit. They also will be encouraged to provide non-code contributions (documentation, events, community management, etc.) and will gain merit for doing so. Those with a proven support and quality track record will be encouraged to become committers.
Community

We hope to extend the user and developer base in the future and build a solid open source community around Liminal. We identify a huge need in the industry for an end-to-end machine learning open source platform, which enables composition and reuse of existing projects. We believe that Liminal will become a key project for the big data ecosystem, easily on-boarding more developers and users.

Several companies in the local industry in Israel have shown interest in use and contribute to such a project.

Known Risks

Development has been sponsored mostly by a single company (Natural Intelligence). For the project to fully transition to the Apache Way governance model, development must shift towards the meritocracy-centric model of growing a community of contributors balanced with the needs for extreme stability and core implementation coherency.

Orphaned products

We, the first code committers of the project, created it because we believe that companies have a need for platforms that solve these common problems. We intend to keep developing this solution because as employees in our current companies or in future companies we believe we will need to solve the same problem set again, and having a robust, open source platform that we can improve upon is in our best interest for our personal futures.

Inexperience with Open Source

Initial committers are already Apache contributors. Additionally, several ASF veterans agreed to mentor the project and are listed in this proposal. The project will rely on their guidance and collective wisdom to quickly transition the entire team of initial committers towards practicing the Apache Way.

Reliance on Salaried Developers

Most of the contributors are paid to work in big data space. While they might wander from their current employers, they are unlikely to venture far from their core expertise and thus will continue to be engaged with the project regardless of their current employers.

An Excessive Fascination with the Apache Brand

While we intend to leverage the Apache ‘branding’ when talking to other projects as testament of our project’s ‘neutrality’, we have no plans for making use of Apache brand in press releases nor posting billboards advertising acceptance of Liminal into Apache Incubator.

Initial Source

Internal private implementation exists in the company Natural Intelligence’s code repositories.

With this project the intention is to create a completely open source version of this implementation, augmenting it to adhere to Apache needs/principles.

External Dependencies

All external dependencies are licensed under an Apache 2.0 license or Apache-compatible license. As we grow the Liminal community we will configure our build process to require and validate all contributions and dependencies are licensed under the Apache 2.0 license or are under an Apache-compatible license.

- Apache Spark
- Apache Karaf
- Apache Camel
- Apache CXF
- Apache Airflow
- Docker
- Kubernetes
- Presto
- ...

Required Resources

Mailing lists
- private@liminal.incubator.apache.org (moderated subscriptions)
- commits@liminal.incubator.apache.org
- dev@liminal.incubator.apache.org
- issues@liminal.incubator.apache.org

**Git Repository**

- https://gitbox.apache.org/repos/asf/incubator-rainbow.git

**Issue Tracking**

- JIRA Project Liminal (Liminal)

**Initial Committers**

- Aviem Zur
- Jean-Baptiste Onofré
- Lior Schachter
- Amihay Zer-Kavod
- Assaf Pinhasi

**Affiliations**

- Talend : Jean-Baptiste Onofré
- Natural Intelligence : Aviem Zur
- Natural Intelligence : Lior Schachter
- Natural Intelligence : Amihay Zer-Kavod
- Huawei: Liang Chenliang
- Not affiliated: Assaf Pinhasi (Machine learning consultant)

**Sponsors**

**Champion**

- Jean-Baptiste Onofré - Apache Member

**Mentors**

- Henry Saputra
- Jean-Baptiste Onofré
- Uma Maheswara Rao G
- Davor Bonaci
- Liang Chenliang

**Sponsoring Entity**

The Apache Incubator.