Data Dependcy Graph in Machine Learning system

Introduction

I was told that the team was always open to project proposals.

This is my first time writting such a proposal, and first time writtong an academic proposal, it is thus probably riddled with issues, please forgive me for that.

Problem statement

"Hidden Technical Debt in Machine Learning Systems D Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, Dan Dennison", NIPS 2015 mentions current technical debt in data analysis systems.

The hidden data dependency one seemed to me as an interesting and solvable problem.

The problem is as follows:

- Data analysis jobs (whether batch or streaming) are consumers and producers of data.
- The data are an integral parts of these pipelines, a change in data can have drastic consequences on the correctness of the results of pipelines.
- Data Dependencies do not have easy solutions to be expressed (to the best of my knowledge), this can lead to hidden (or even cyclic) dependencies between data sources.
- Not knowing the dependencies can lead to broken pipelines when schemas are updated (changing a field in a protobuf for example) which may prove costly to fix. This is also an engineering problem as it may make the importance of each data in the whole system troubling to evaluate (which can be used for promoting the data from spindles to SSDs for example).
- The problem of code dependencies is currently solved using static analysis systems.

Objective

An objective of this project would be to have a tool that can create a directed graph of all data dependencies which could not be bypassed.

Proposed approach

A first aproach using the methods from software would be to explicitely write the dependencis in code, that could then be analysed with static code analysis tools.

This method seems promising but lacks an enforcement policy, and is thus brittle (in the same way that CI which accepts code with failing tests do not work from my industry experience).

A better solution in my opinion would be to enforce this at the executor level using file access policies.

A pipeline binary (or set of binaries) would be registered at a central authority and read and write accesses to the (possibly distributed) file system would be granted to this binary, and the binary would be launched by this central authority.

This approach has multiple advantages:

- Read access cannot be bypassed, thus dependency on data cannot be hidden.
- Write accesses cannot be bypassed, thus data without tracable source cannot exist.
- Data Dependency graph can be created.

There are also some added benefits that come from this approach:

- Registering at the binary level allows for versioning of binaries, which helps reproducibilty.
- Having a central authority with versionned binaries allows to recreate historical data with the same exact pipeline, which can be useful for debugging or recreating erased data.
- We can create a exact origin of data, containing both the data sources and the processing steps, which can be usefull for tracing the origin of entries in knowledge bases (and an easy way to introduce "confidence" in the truthness of entries depending on the source)

The current drawbacks that I see are the following

- The original data (data without father in the dependency graph) could be hard to introduce in the system.
- Development of pipelines would be slowed down because it will need to be registered in the system. Having an exception for develpment phase purposes could be introduced, but could be used as a bypass of the whole system.

Proof of concept

A proof of concept that could be used for the experiments section of a paper could be a Hadoop module, that would thus handle file accesses and execution of jobs.

It would also require a database to store the binaries (and their previous versions), that would need to be decently duplicated (failures at that level would be dramatic).

Expected amount of work

Coding such a Hadoop module with a decent user interface could be done in 4 month.person, if my (extremely incomplete) undersanding of the Hadoop codebase is correct.

Impact

This project seems to fit with the problems dealt with in the CEDAR team (from my once again incomplete understanding) and could lead to a publication in either NIPS (where the problem was first introduced) or ICDE like conference.