Google FRA Proposal: Towards a JavaScript-Empowered Smart Interactive Computing Architecture (JESICA)

Skeleton for clickable DOIs

Abstract

As data analysis pipelines become more complex as well as ubiquitous, the need for standardization and community data platforms emerge. At the same time JavaScript has become the most widely used language on the planet, powering any kind of computing device. We therefore propose to develop JESICA, an open access Web-based data platform allowing experts and users alike to visually compose state-of-the-art processing pipelines. Furthermore meta data about each experiment will be stored; this will enable heuristics to drive a recommendation engine which is able to propose optimal processing paths if presented with input data plus problem specification. As the logical centerpiece of JESICA, we will develop a Smart Computational Pipeline (SCP) capable of self-configuration and user interaction, as well as a Webworker-based mechanism for background execution of individual stages. Our goal is to demonstrate the SCP's feasibility within a year using predefined test data sets.

1. The problem with today's data analysis pipelines

In order to realize the vision of a bright future of data accessibility (Halevy, 2012), (Gupta et al., 2013), we need standardized mechanisms to implement data analysis pipelines transforming raw data into structured information. Today, much effort is potentially squandered by implementing complex pipelines within isolated teams in a non-standardized fashion: Proprietary approaches - both in technology as in methodology - hinder the exchange of information within the data community. Let's take a look at some problems in detail:

- Isolation. Disparate approaches to data analysis makes dealing with errors at any stage of the pipeline unnecessarily hard due to a lack of reference values, whereas solving problems and achieving superb results has no beneficial effect on the potential of others.
- Proprietary Software. Professionals often develop algorithms in highly proprietary environments. This prevents an influx of solid, community-tested algorithms while restraining proliferation of gained knowledge to those unwilling to pay for a particular software package.
- Irreproducibility. Results from experiments conducted in isolation cannot be easily corroborated. This might be advantageous with respect to product development and patent procedures, but is usually detrimental to the efforts of researchers trying to gain acceptance.
- Lack of scalability. Highly heterogeneous and customized data processing pipelines may not lend themselves well to parallelization, preventing their use on quickly expanding datasets.

2. JESICA - overview

The central idea behind our project is to utilize the power of modern JavaScript Engines to perform data processing tasks recently only conducted on servers. Our first experiments in graph extraction from images (the project which spawned the idea of JESICA) showed that even with unoptimized JS code image processing can be done at about the speed of Matlab (Holzinger et al., 2014). While technologies like asm.js or WebGL/CL will increase the performance of single computations manyfold, there are presently no JS-based libraries able to configure whole processing pipelines based on a standard format with dependency resolution and the ability to self-configure based on heuristics. JESICA is designed to be an open, web-based platform delivering data analysis algorithms in JS to browsers, providing a Smart Computational Pipeline (SCP) equipped with a problem case analyzer as well as a (Machine Learning) model recommender. An example use case would see a user drop an image folder into their browser and specify a desired outcome (object recognition / classification etc.). JESICA would then recommend an analysis workflow, configure itself accordingly, and upon execution send a package of metadata describing the pipeline including results to a server heuristics database. This in turn would train the recommender, so that new ML models produced by individuals could immediately benefit others using the platform.

3. JESICA - characteristics

- Accessibility. Obviously, delivering JS to browsers is the most convenient form of 'installation'.
- Effortless scalability. As users of our platform will provide their own computing power, the server role initially can be reduced to that of a static document server plus database.
- Meta Machine Learning → Heuristics. As researchers start using our platform, their pipeline configurations, input descriptions, task specifiers as well as results will be stored on the server. Provided enough data, meta machine learning could provide heuristics as to what succession of algorithms might be best suited to tackle a specific input / problem combination.
- Pipeline recommender. Based on the previous step, an optimal pipeline configuration could be recommended. The SCP would then self-assemble and be ready to commence the experiment.
- User interaction will be possible via a modern browser UI allowing to assemble a pipeline by dragging visual representations of building blocks (tasks / algorithms) into place. At every step, the SCP must be able to check and assert the feasibility of a user-provided pipeline.
- Goodies. Input and output filters (Latex!), the formation of research groups, social collaboration, bookmarks to fully-configured experiments (reproducibility) etc. would be typical WebApp addons which could turn the basic service into a promising data analysis startup.

4. The business case for JESICA

The first and obvious case is a data analysis / machine learning platform for experts who do not wish to spend weeks or even months designing, writing, optimizing, deploying and monitoring proprietary pipelines. While this problem is also tackled by others, JESICA's unique advantages will open up possibilities towards end users as well:

- Because JESICA will be Web/JS based, the barrier of entry to getting involved is simply going online. This will benefit editors of journals upon receiving submissions, journalists writing about new data insights, students desiring to learn from real world examples, and probably many others.
- JESICA will be running on practically every computing device in the world. It will not be long before a simple smart phone will reliably classify dermatological images for cancer detection routinely - without the need for transfering sensitive data over the network.
- Combining modern speech recognition capabilities with JESICA's self-configuring SCP, users will be able to entrust their devices with personalized analysis tasks far beyond the reach of contemporary search engines. In short: NLP + JESICA = actually "intelligent" personal assistant!

5. Our work - what the Google grant will finance

JESICA as a platform will comprise several components, but as its centerpiece the Smart Computational Pipeline (SCP) will form the logical starting point. Our 1-year plan thus encompasses

- developing a suitable data format for representing the pipeline, including dependencies between tasks (algorithms) on individual stages plus a result format for later server side analysis,
- implementing a DAG dependency resolution algorithm with visual feedback and user interaction through an intuitive UI based on graphical representation of algorithmic building blocks,
- devising a Webworker-based system for executing individual stages of the pipeline in the background without the need for excessive copying of large datastructures,
- and outlining as well as conducting a series of test cases to demonstrate the SCP's feasibility with the use of modern browsers. As use cases we will build on the experiments described in (Holzinger et al., 2014) as well as text analysis tasks with publicly available datasets.

Challenges to this approach (and any algorithmic platform based on JavaScript) include the lack of a JS Machine Learning community and therefore a lack of available code. As it is unrealistic to try and implement hundreds of interesting algorithms ourselves, we will focus on transfering existing implementations from other languages to JavaScript with as much automation as possible. Our idea so far is compiling C/C++ as well as Python libraries to asm.js using LLVM / Emscripten. Although completion of our set goal does not require the availability of a broad spectrum of algorithms, this transfer tool would represent the next logial step towards introducing JESICA to the community. We will thus tackle it first in case we are able to advance speedier than expected.

6. **Prior work & levels of data analysis pipelines**

   As existing stacks indicate, there are several possible levels of pipelines which can be sorted by increasing homogeneity amongst their stages as well as decreasing technical demands on their users:

   - Level 0: Writing all of the pipeline manually. As every combination of technologies are usable, this approach gives the most flexibility but is hard to maintain and almost impossible to reproduce. Furthermore, as (Sculley et al., 2014) perfectly states: "Using self-contained solutions often results in a glue code system design pattern, in which a massive amount of supporting code is written to get data into and out of general-purpose packages."
   - Level 1: Automated, but self designed and coded. This entails the usage of tools like Unix Make which is language agnostic and therefore supports any number of technologies as long as they are executable from a Shell. Apart from slightly better decoupling, same problems as Level 0.
   - Level 2: Establishing a common understanding of the components and structure of a pipeline while still using individual technologies. Such a common standard exists in the form of PMML - the Predictive Model Markup Language. PMML defines stages of pipelines as well as their inputs, parameter types and ranges, the output format etc. Thus, developers can use their favorite technologies in the development phase while PMML consuming tools then produce community-standard compliant code (Hadoop, Spark, ..).
   - Level 3: Language specific libraries exposing an API to conveniently assemble a pipeline. Such libraries have been released by projects like scikit-learn or Apache Spark. While currently becoming popular, APIs still restrict the creation of data applications to experts capable of coding.
   - Level 4: The use of a custom Domain Specific Language (DSL) would widen the ability to create complex data pipelines to any kind of domain expert. Similar in nature to SQL, it is able to either compile itself into code or an intermediate representation like PMML.
   - Level 5: A fully integrated data analysis platform that offers intuitive, visual pipeline assembly. Ideally, tools for reporting, reproduction and collaboration would also be included. In addition, the platform could offer experts the means to write stages themselves via an online code editor. JESICA is designed to be such a platform.

7. **Data Policy**

   The outcome of this project will be delivered as Open Source Software as well as Scientific / Engineering Publications. We will host our Code in a public github repository at https://github.com/cassinius/smart-computational-pipeline. Papers will be submitted to journals committed to an open access policy. More information about the JESICA Idea can be found at http://berndmalle.com/jesica.

8. **Budget**

Matter of expense	**Intended use**	**Expenditure in USD**
Bernd MALLE	PhD Salary for one year	45,000
Travel costs	conferences (e.g. iUI2016)	3,000
Total expenditures		**48,000**

References

Gupta, N., Halevy, A. Y., Harb, B., Lam, H., Hongrae, L., Madhavan, J., Fei, W., & Cong, Y. (2013). Recent progress towards an ecosystem of structured data on the web. In *29th International IEEE Conference on Data Engineering (ICDE)* (pp. 5–8).

Halevy, A. Y. (2012). Towards an ecosystem of structured data on the web. In *Proceedings of the 15th International Conference on Extending Database Technology*, EDBT '12 (pp. 1–2). New York, NY, USA: ACM.

Holzinger, A., Malle, B., & Giuliani, N. (2014). On graph extraction from image data. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8609 LNAI, 552–563.

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., & Young, M. (2014). Machine learning: The high interest credit card of technical debt. In *SE4ML: Software Engineering for Machine Learning (NIPS 2014 Workshop)*.