


CS486C – Senior Capstone Design in Computer Science

Project Description

Project Title: Event-driven Machine Learning, Intelligent Assessor (EMELIA)	
Sponsor Information:  Mission Systems	Rick Duarte , Systems Engineer Space and Intelligence Systems General Dynamics Mission Systems Richard.Duarte@GD-MS.com 480-441-4893

Project Overview:

In the modern world, software has become the life blood of almost every aspect of modern life, driving systems ranging from your microwave to your smartphone to the guidance systems on aircraft. As the complexity of these systems has grown, it has become increasingly critical to develop effective software engineering techniques and tools, both to avoid buggy systems in the first place through careful structured architecting and, when bugs inevitably do arise in complex software systems, to be able to detect and resolve them as efficiently as possible. To address this latter challenge, the software engineering community has developed a broad range of tools, e.g., testing frameworks, powerful debuggers, systems profilers. Such tools are often effective, but gaps in coverage remain; there are some situations for which no good tool exists...which drives the development of a new tool. This project is motivated by one such situation, and is aimed at producing an effective tool for helping our engineers quickly identify and localize the underlying software bug.

When issues in complex systems consisting of interacting software and hardware occur, it is the job of system engineers to analyze those failures and determine what happened and how it impacted the system. To track down exactly when and where the error occurred, hundreds of data fields with hundreds of possible value must be painstakingly analyzed by a human in each failure case to make a determination as to what happened, i.e., at what point the bug occurred (as indicated by erroneous data appearing), leading to identifying the function(s) responsible in the codebase. This can be a lengthy and mind-numbing process for an engineer to do manually, especially since the problem is often not localized in a single data field, but is more of an issue of multiple related fields becoming erroneous, i.e., more of a problematic *pattern* than just a single.

In recent years, AI technologies have made tremendous progress in precisely the area of pattern recognition or, more formally, training AI “classifiers” to be able to correctly identify members of some class (in this case “error states”) based on a large set of training examples. This project aims to develop and explore the efficacy of a machine-learning tool that utilizes AI technologies to correctly analyze these failures, thereby reducing the time needed to manually look at every failure case.

Specifically, we envision a command-line tool running on the Linux platform (though platform choice may change based on the team’s initial analysis and recommendations) that implements a machine learning solution that can be trained using a large database of human-classified data related to failures (provided by the client), and can then effectively analyze new failure cases. Some key functions of the solution would include:

Level 0: the Basics (minimum viable product)

- Able to pull from a (growing) database of previously classified failure cases to train an appropriate machine-learning classifier.
- Once trained, able to effectively classify new failure cases with a high degree of accuracy.
- Some sort of basic test harness to make it easy to explore system performance, i.e., can change various parameters (training set, feature set, target set to classify, etc.) and then run the classifier on the target set, recording its accuracy, speed, etc. The idea is to get an idea of what feature sets are most relevant for

accurate classification, and how many training cases are needed to reach a certain degree of classification accuracy.

Level 1: a well-appointed tool

- A GUI front end to embody the testing harness, allowing selection of parameters, doing runs, and comparing results.
- A graphical visualization of the key characteristics of both input (training) data sets as well as processing results. In “big data” scenarios like this, it is often an aggregate visualization of large dataset features that reveals interesting patterns.

As a performance goal for the prototype, such a system could be seen as “useful” in this context if it were able to correctly classify novel cases with an accuracy of 90%.

Knowledge, skills, and expertise required for this project:

- Expertise in machine learning: how it works, what are best applications/limitations
- Familiarity with various freely available machine learning frameworks and tools, e.g., fastai, TensorFlow, Keras, Conda, etc.
- Strong programming skills in a variety of programming languages likely to be applied in this project, including C#, C++, Python. Final determination of appropriate programming language(s) will, of course, be made by team after initial problem analysis.
- Familiarity with Linux operating systems and shell scripting.
- Familiarity with systems failure analysis; client will provide guidance in helping team acquire knowledge of the particular systems and failure profiles relevant to this application.

Equipment Requirements:

- There should be no equipment or software required other than a development platform and software/tools freely available online.
- Access to high-performance computing (e.g. for realistic testing of a beta prototype) will be facilitated by SICCS faculty on the Monsoon cluster as needed.
- Timely access to the large training datasets will be provided by the client.

Deliverables:

- The software applications as described above, deployed and tested successfully with real data. Must include a complete and clear User Manual for configuring and operating the software.
- A strong as-built report detailing the design and implementation of the product in a complete, clear and professional manner. This document should provide a strong basis for future development of the product.
- Complete professionally-documented codebase, delivered both as a repository in GitHub, BitBucket, or some other version control repository; and as a physical archive on a USB drive.
- Professional presentation of the project and key outcomes and conclusions to the client’s team is required.