DeepAnomaly

Gsoc Proposal-Cern-HSF/ATLAS - 2017

"Optimizing computing operations with machine learning algorithms" By Vyom Sharma

Introduction

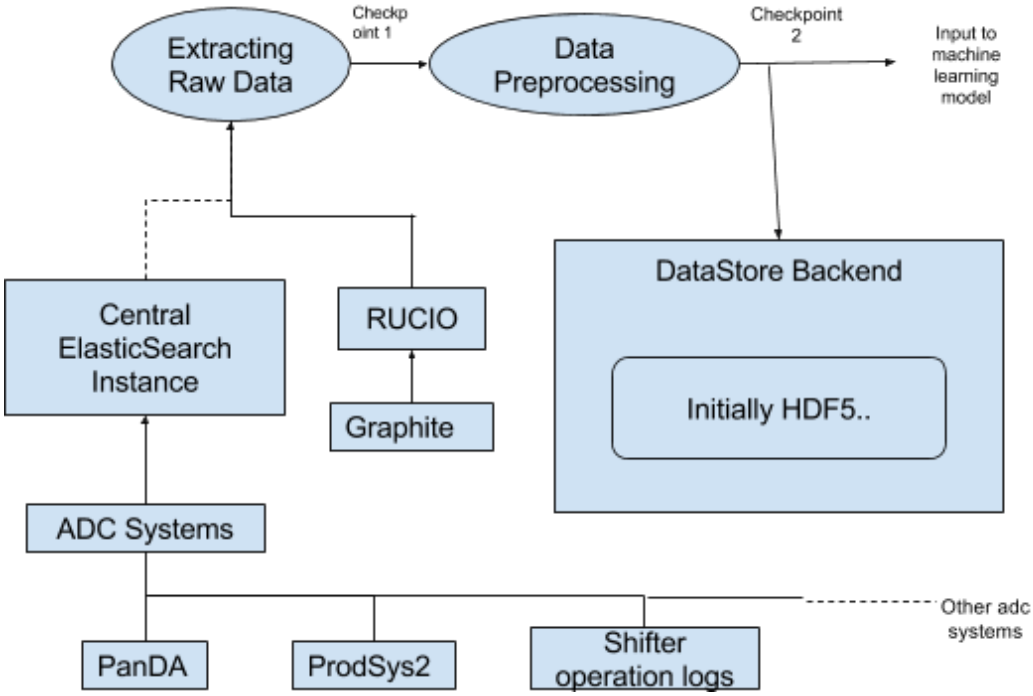
ATLAS Open Analytics is a platform which collects data generated from the various computing operations between the multiple participating heterogeneous computing centers.

The idea is to design, build and deploy a framework that is able to monitor, classify and cluster anomalous behaviour in this real-time ATLAS Computing Operations data and then act on this information autonomously.

The tasks involve building a complex pipeline, at the heart of which will a machine-learning algorithm that detects these anomalies and provides intelligent, actionable predictions. The output from this anomaly detector will then be used to handle these triggered anomalies by implementing another machine-learning model, which has been specifically trained to predict the best possible solution using the historical operator decision data. This second model will analyze the triggered anomalies and notify the on-shift operator about the event as well as suggest the best potential resolutions.

Implementation

Data Aggregation and Preprocessing-



This task involves three major parts-

1. Writing a generic function which extracts data from the backend, in a way that's scalable and fast(and can be adapted to other ADC systems apart from RUCIO)
2. Preprocess the data so that it can be fed into our Machine learning model as input.
3. Build a datastore backend for storing and backing-up up of historical data. I'm already familiar using hdf5 for this purpose(I would also like to explore some alternative options like ROOT , Apache arrow for this task).

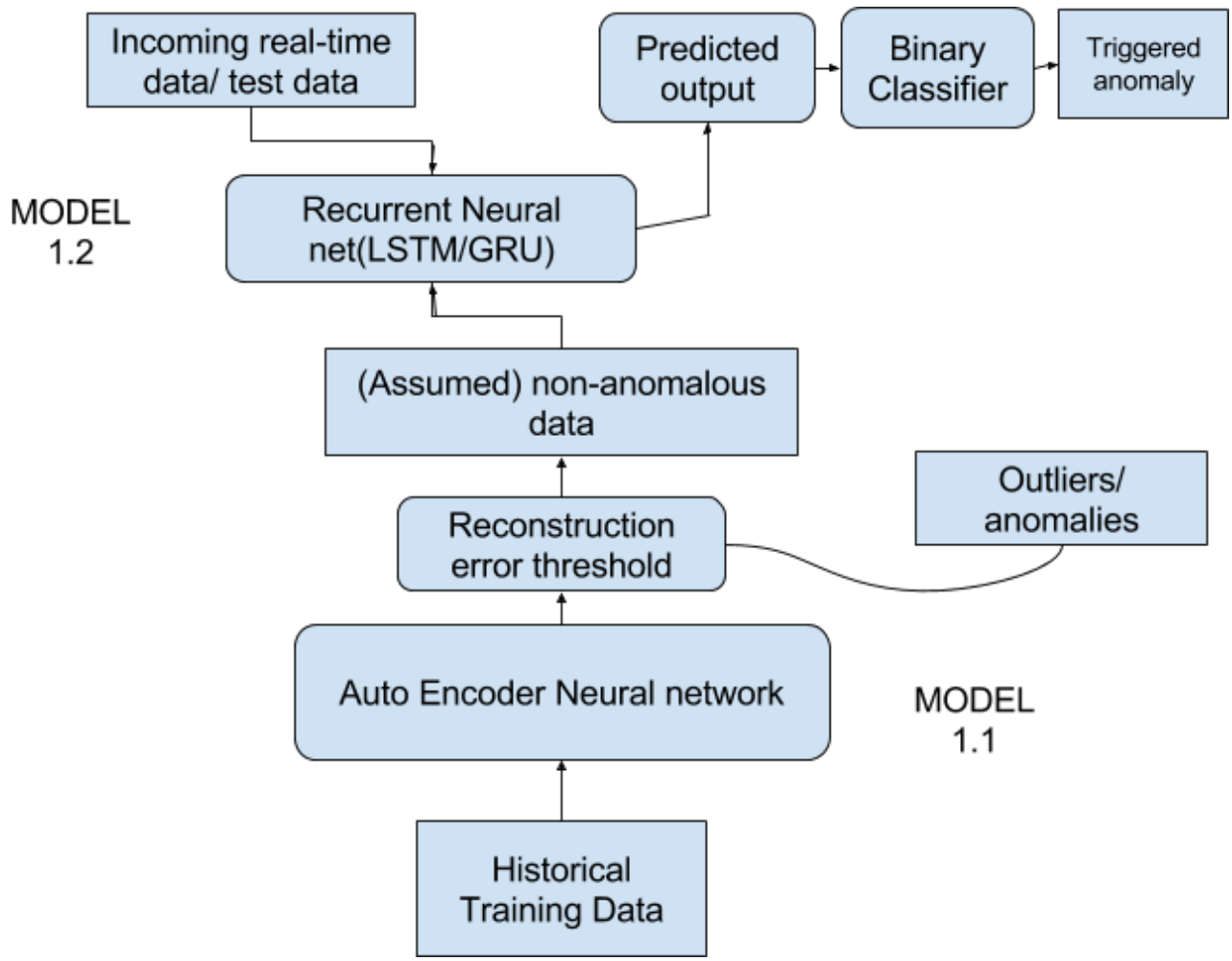
Machine Learning for Anomaly Detection-

This is the core part of the project and will require the maximum experimentation and exploration.

The available historical data, currently, has *no form of segregation* between anomalous and normal/non-anomalous data. Most of the Machine learning approaches for detecting anomalies involve training the model initially on completely normal data. Only after this can the model be used for making any kind of predictions on anomalous data.

My plan to accomplish the task is as follows:

- Split the available historical data into training and test sets.
- Use a recurrent autoencoder to roughly remove outliers from our training data.
- Again split this cleaned data into training and validation sets.
- Train to recurrent net on this data to make predictions on time-series data.()
- Build a binary classifier on top of this recurrent network(based on thresholding or mixture of gaussians) to predict anomalies based on the network output.
- Evaluate this architecture using the unseen testing data.
- Tune the binary classifier to improve throughput metrics.



1. **Historical Data segregation-**

a. Build an AutoEncoder neural network to detect outliers in the historical data.
    i. Autoencoders are an unsupervised learning approach to anomaly detection.They basically consist of a pair of non-linear transformations- encoders and decoders.
    ii. There are many types of auto encoders but for this project, a **Recurrent Autoencoder** would probably be best.(VRAE- variational recurrent autoencoder). This architecture implements recurrent nets for encoding data instead of simple feedforward implementations.
    iii. The idea is that autoencoder will not learn to map outliers/anomalies in our data correctly, leaving us with non-anomalous data which can then be used in a supervised setting.
    iv. The threshold for the reconstruction error value(as given by the auto encoder) will be a hyperparameter that will need to be selected carefully.

2. **Recurrent neural network -**
    a. Use core TensorFlow/ Keras to build an RNN to classify input data as anomalous or normal

        i. Try a host of different loss functions(MAE, MSE ,RMSE)and performance metrics.
        ii. Optimize the performance by tweaking the model hyperparameters such as:
            1. LSTM or GRU or Simple RNN
            2. No. of hidden layers
            3. Width of the hidden layers
        iii. Apply regularization and dropout layers to make sure the model does not overfit and generalizes well on the testing data
        iv. Initially, during training this model will be trained on the output of the autoencoder network.
        v. This is the actual classifier that will be used(in conjunction with the binary classifier) for predicting anomalies. (not the autoencoder)

    b. Visualize the built network using tensorboard to compare training cycles and see how it learns.

3. **Anomaly detection using a binary classifier-**
    a. **Thresholding**- if the predicted output deviates too far away from the actual value, then it's labelled as an anomaly.
        i. This method uses a threshold value; **d(t)=||actual - predicted||**; for predicting anomalies.
        ii. It's fast and scalable but runs the risk of detecting too many false negatives
    b. Gaussian Mixture model or Mixture of gaussians
        i. Mixture of gaussians on norm of the difference
        ii. Mixture of gaussians on norm of the oscillations

        1. This method is more efficient than simple thresholding but is computationally more expensive
        2. Can detect anomalies even where d(t) is very low.
    c. Other clustering algorithms like K-means clustering.

4. **Save the learnt model in the same datastore backend**
5. **(optional)** Although I am pretty confident that RNNs will prove quite effective for the task at hand, there might be some other options we could explore.
    a. Sometimes CNNs also work well for time-series data. We can make a 1-D kernel and have it scan across the series.
    b. Adversarial networks invented by Ian Goodfellow have been all the buzz ever since I started tinkering with neural nets. There might be a way to train a classifier to detect anomalies by using these(where both the generator and discriminator networks are RNNs). This might not be a good idea but i want to keep this in as a last resort.

Monitoring incoming data and Triggering anomalies, Operator decisions-

1. **Deploy the Anomaly Detection algorithm to the Kafka platform** where it can be used directly on the incoming data.
2. Design and build a **new software(say 'axb' for reference) which stores the output from our anomaly-detector model and monitors it for the detected anomalies**.
3. Implement a **Reinforcement learning algorithm** which will analyze the attributes within the triggered anomaly to **predict a sorted list of n-best solutions**.
    a. This algorithm will better itself over time by collecting rewards based on the feedback obtained by the manual operators.
4. When an anomaly is detected, it will lead to a ranked/sorted list of n-solutions being created and sent to the human-operator on shift for feedback.
5. Build a simple feedback system for the human-operators. Ideally the solution on the top of the list should be always the correct one.Feedback will be in the form of categorical variables - 'good', 'wrong', 'not-applicable'.
6. The feedback received from the operators will be used to reinforce the 'solution-ranking' algorithm.
7. Store the operator decision, underlying cause of the anomaly, proposed feedback as well as the executed solution in **'axb'**.

Automation-

- Review the code up to this point. Measure the performance of individual pieces and models.
- Identify some of the most common anomaly occurrences and come up with end-to-end implementations for automating the job of fixing/repairing their causes.

Project Goals

- A generic way of extracting data from backends/ElasticSearch instances.
- A preprocessor which converts raw data into input for the ML model and stores it as HDF5 files on a backend datastore.
- A machine learning classifier which classifies anomalies in the time-series data with high accuracy deployed on ‘kafka’.
- A way of notifying the on-shift operators of triggered anomalies and proposing a list of best potential resolutions.
- A feedback system to collect feedback from the manual operators to be used in conjunction with a reinforcement learning algorithm to improve the efficiency of the proposed solutions over time.
- A new software(under ADC) which stores operator decisions, triggered anomaly characteristics/causes and feedback from the manual-operators for each entry.

Timeline

Generally, the timeline is expected to follow the order of tasks outlined in the implementation section above. I’ve included

Before May 4:

- University Practical Exams - April 24 - April 28
- Waiting for the announcement!

May 5 – May 30 (community bonding):

- Getting familiar with all the major relevant ADC systems.
- Work on understanding the relationships amongst data on the various interconnected ADC systems.
- Explore and gather potential insights from the actual data.
- If possible, observe how the current human operators detect anomalies within the incoming event data.
- Scour the web for potentially useful advancements in machine learning and time-series data analysis.
- Remain in constant contact with my mentors as well as the ATLAS community to discuss and finalize on any modifications in the projects as well as to get crystal clear about the project goals and milestones.
- University Theory Exams - May 12 - May 31.

May 30 - June 26: Coding phase-1

- Milestones- data aggregation and anomaly detection
- Optimize models to improve throughput metrics
- June 3 - summer vacations begin

June 26- June30 - Phase 1 evaluations

June 26 - July 24: Coding phase-2

- Summer vacations end-(July 16)
- Milestones - steaming trigger, operator decision
- Test the individual components on real-time data to evaluate the system.

June 24 - June 28 - Phase 2 evaluations

June 24 - Aug 10: Coding phase-3

- Milestone - automation
- Fuse together all the different pieces of code to build a deployable framework.
- Beta testing with real-time data.
- FIXING Issues and bugs.
- Implement some features for the long-term deployment of the framework such as a way for the Machine learning classifier to keep on improving its performance by training itself on new data(periodically)

Aug 10 - Aug 21 :

- Buffer Time for unexpected delays

Final Evaluation and submission - AUG 21 -AUG 29

About Me

I'm Vyom, a student studying Information Technology from New Delhi. I absolutely fell in love with deep learning and artificial intelligence in the first year of college. Ever since then I've been trying to learn everything I can about this topic. Currently I spend my days working on all things 'deep and neural'. I love listening to classical piano music, watching 'Game of Thrones' and swimming. Also I recently started a deep learning blog - Deep Diving.

Motivation-

My primary field of interest is deep learning and artificial neural networks. I specifically searched for Gsoc projects for these topics and I believe this project is ideal for my skills and interests.This project poses a unique challenge from a machine learning perspective. This being said I'm confident that I posses all the skills required to successfully complete this project. Apart from this, working for CERN is like a dream come true for any engineer. If I do get accepted for this project I'll be deeply honored.

Relevant experience-

- Intimate familiarity with Google's machine learning library TensorFlow.
- Student of Udacity's Deep Learning Foundation Nanodegree(completion date- June 5)
- Taken Andrew Ng's Machine learning course on coursera.
- Intimate knowledge of recurrent neural networks and other deep learning algorithms.

- Implemented projects based specifically on recurrent networks-
    - Anna KaRNNa - a lstm network trained on ‘Anna Karenina’ which generates new characters.
    - Sentiment Analysis - a recurrent net trained on word-embeddings to predict sentiment
    - Script Generation - generated a script for the ‘The Simpsons’ TV show by training a model on scripts of previous 27 seasons.
    - A recurrent net to generate music.

Other obligations -

I have my semester exams during the community bonding period(from may 13 to may 31) but I’m sure they won’t be the cause of any hurdle to my Gsoc work as I’ll be preparing in advance for them during the month of April.
Apart from this my only other commitment is to the Udacity’s Nanodegree program which requires somewhere from 3-5 hours per week. This too will end at most by June 5 i.e. 5 days after the phase -1 of coding officially begins. This program is not really a hinderance but actually provides me with another resource for guidance which can be in hugely helpful with the machine learning part of this project.

Overall I pledge on devoting about 40-45 hours per week working on this project.

Email - vyomshm@gmail.com
Github - https://github.com/vyomshm/
College - University School of Communication and Information Technology
University - G.G.S. Indraprastha University
Major - Information Technology
Year of study - 2nd