DFFML : Implementing Distributed Orchestrator And Adding DataFlow Tutorials

Organization: Python Software Foundation

About Me

Name : Aghin Shah Alin K
University :
  Name: Indian Institute Of Technology,Madras
  Program : Dual Degree(B.Tech + M.Tech)Computer Science And Engineering
  Year : 3rd
  Expected Graduation:2022
Contact:
  Email : aghinsa@gmail.com
  Linkedin,GitHub,Gitter : @aghinsa
Time zone : IST (UTC +5:30)

Contributions

[WIP][pull#492]:Auto start operations without inputs (issue#392).

[Merged][pull#454]:Input validation using operations(issue#382):
Add ability to validate inputs using defined operations in the dataflow.

[Merged][pull#387]:Input forwarding to subflows(issue#386,issue#368):

[Merged][pull#357]:Add validation to definitions(issue#349):
Inputs of certain function can be validated according to the predefined function in the definition.

[Merged][pull#340]:Change classification to predict(issue#316)

[Merged][pull#333]:Prediction as feature specific dictionary(issue#106)

[Merged][pull#300]:ConfigLoader(issue#255):Configloader which recursively loads directory and filenames in a dictionary.

[Merged][pull#299]:Few minor fixes(issue#265,issue#292,issues#291,issue#288):
Rename entry_point to entrypoint.Remove msg parameter from model.Remove def when defining features.Raise exception when model is not an instance when passing to configs.

[Merged][pull#268]:Operation:RunDataFlow(issue#254):
Operation to run a dataflow as a subflow in an existing dataflow.

[Merged][pull#253]:Config,args cleanup and int values for src_url(now key)(issue#207):
Change models to use config decorator.Fix src_url(key) being int instead of string in csv source.

[Merged][pull#238]:Move features from model context to model config(issue#231)

[Merged][pull#226]:Adding Tensorflow regression model(issue#75)

[Merged][pull#221]:Add hidden layer info to hash of model_dir(issue#220)

Project

Sub Org : DFFML

Abstract

Add networks/orchestrator to run distributed dataflows.Add new tutorials on using DataFlow/Operations and how models/database abstractions etc interoperate within a flow.

Description

This project consists of two parts:

1. Adding more **concise tutorials for dataflow/operation** in dffml docs.
2. Adding a **Distributed Dataflow Orchestrator**

Data Flow Facilitator for Machine Learning (DFFML) makes it easy to generate datasets, train and use machine learning models, and integrate machine learning into new or existing applications. It provides APIs for dataset generation, storage, model definition etc. Under the hood ,a major portion of the work is done by the concept

Dataflow to facilitate flow of data between ***Operations*** as defined by the user or based on ***Definitions***.

1.The current tutorial on dataflow/operation is very long and misses out on some of the new features.This project aims at adding tutorials on four topics

- **Basic:** Simple usage of operations and how data flows from one operation to others.This will be illustrated by a dataflow which does basic calculations of numerals.
- **Locking:** Usage of dataflow to manage locking of shared data.
- **Configs:** A gitter/irc bot which gets live messages from the channel and publishes it to the dataflow.
- **Subflows:** The message received by the bot is forwarded to additional subflow(s), which formats the message and runs nlp inference on the message.

2.Distributed dataflow: Dffml aims to make it easy for users to generate/forward data to models with the least effort possible,machine learning models are often trained on very large datasets and as such adding functionality for distributed computing moves dffml closer to this goal.

The plan is to have a new ***Distributed Orchestrator*** which uses a messaging queue to publish data between nodes,where a ***node*** runs arbitrary operation(s).The nodes are provided configuration on start as to how to communicate between each other. ***NATS stream******(STAN)*** will be used as a message queue.STAN is selected because it offers *at least once* delivery and as such which allows nodes to start their subscription at an earlier point in the message stream, allowing them to receive messages that were published before this client registered interest.The ***subjects*** used in STAN and *definitions* used in dffml are more or less equivalent and both are *asyncio* based.

The current approach is to have a *node* cli command instantiating a ***NatsNodeOperationImplementationNetwork***,which lets the ***master node*** know what operations are allowed to be run by the node,example cli command instance would be `` `dffml service node run run_bandit run_safety` ``,with any additional configs required by the node to connect to the requires *STAN* channel(s).The *master node* which runs a ***NatsPrimaryOperationImplementationNetwork*** has the *DataFlow* and instantiates every operation instance required by the flow.The network publishes the operations to a STAN channel which all the nodes are listening to.Any node which is allowed to run the operation instantiates it and sends back an acknowledgment.At this point the *DataFlow* is set to run and is waiting for inputs.

Inputs in the network are used to generate *ParameterSetsPairs. NatsNodeOperationImplementationNetwork* sends these to a *STAN* channel where all the nodes are listening.The nodes which have the operations instantiated will accept the corresponding parameter sets,and after running the operation they add their outputs back to the ***NatsNodeInputNetwork***,which publishes the *InputSet* to a channel where the *master node* is listening.***NatsMasterInputNetwork*** accepts the new *InputSet* and the cycle repeats.

Timeline(Tentative)

Pre-GSoC (April 1 - May 4)

- Go through documentation of nats.
- Implement basic nats communication tutorials
- Work on other issues related to dataflow.

Community Bonding (May 4 - June 1)

- Finish any pending issues which might affect addition of the examples in the tutorial.
- Complete the calculator example for the tutorial

Week 1 (June 2 - June 8)

- Complete the locking example

Week 2 (June 8 - June 14)

- Implement operations
  - to get data from live chat.
  - format it suitably to be forwarded as input to prediction model

Week 3 (June 15 - June 21)

- Implement operation(s) to run the received message through nlp models

Week 4 (June 22 - June 28)

- Complete bot example

Week 5 (June 29 - July 5)

- Add documentation for the added dataFlow operations and finish the tutorial

Week 6 (July 6 - July 12)

- Implement
  - NatsPrimaryOperationImplementationNetwork
  - NatsNodeOperationImplementationNetwork

Week 7 - Week 8 (July 13 - July 19)

- Add unittest to verify that the dataflow is being instantiated properly

Week 9 (July 20 - July 26)

- Implement NatsMasterInputNetwork

Week 10 (July 27 - August 2)

- Implement NatsNodeInputNetwork
- Write tests to verify InputNetworks are working properly

Week 11 (August 3 - August 9)

- Integrate all the networks in the distributed orchestrator
- Final tests

Week 12 (August 10 - August 16)

- Document all the added features

Week 13 (August 17 - August 24)

- Buffer week

Stretch Goals

In case everything gets done earlier,i'm planning to work on integrating dataflow with the web ui under development.

Post GSoC

Contributing to Dffml has helped me learn a lot.I'd be happy to stay part of the community and keep contributing.I will be working on issues in the codebase,and will work towards completing the web ui.

Other Commitments

I have my end semester exams scheduled in the first week of May,during the community bonding period.I can only work for 2-3 hours during exam days(4-days in the first week of GSoC timeline),other than that i don't have any commitments.I'm only applying to DFFML,and can easily give around 40 hours per week to the project.