11-792 Project Report

Nicholas Gekakis, Boyue Li

February 14, 2018

1 Overview

In this project, we are building a distributed question answering pipeline framework, which allows users to easily configure multiple modules, create complex pipelines and tune parameters automatically.

2 Requirements

2.1 Easy to configure and deploy

The framework should be easy to configure and deploy.

2.2 Save and resume

The framework should be save intermediate results so that it can be interrupted resume running at a later time.

2.3 Pipeline topology

The framework should be able to create pipelines with complex topologies.

2.4 Automatical parameter tuning

The framework should be able to automatically tune some parameters that have been exposed to the system by the pipeline developers.

2.5 Distributed parallel processing

The framework should support distributed parallel processing to handle large datasets and complicated pipelines (e.g. forked pipelines).

3 Design

A pipeline is constructed from several independent modules. Users only need to specify

1. The correspondence between inputs and outputs
2. The connections between modules

The framework will automatically handle all execution.

3.1 Pipeline

The pipeline class manages modules and parameters. It reads configuration file, creates the pipeline and controls parameters tuning when running.

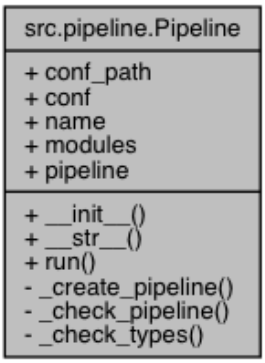

Figure 1: UML diagram for the pipeline class.

3.2 Module

A module is the basic calculation unit which takes an arbitrary number of inputs and produces an arbitrary number of outputs. Every input and output is an information object defined below. A module needs to save following fileds:

- Unique ID of the module.
- Name of the module.
- Inputs.
- Outputs.
- Parameters.

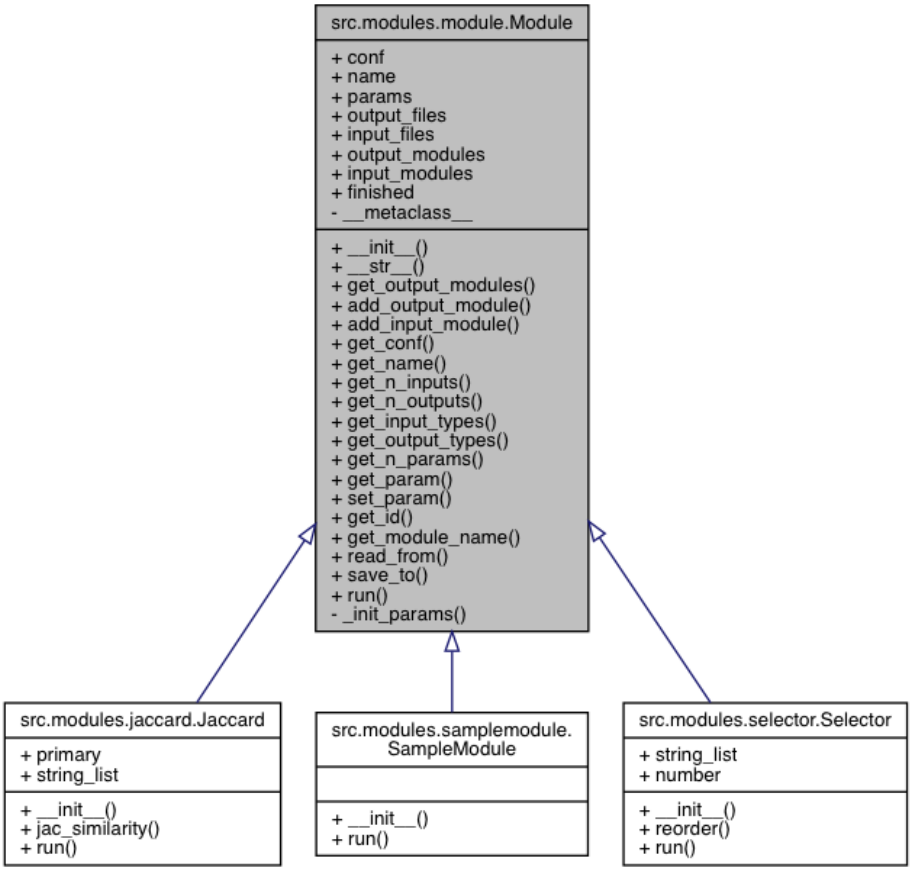

Figure 2: UML diagram for the abstract module class and derived classes.

3.3 Parameter

The parameter class manages one parameter. It should handle all operations related the parameter, including updating the parameter value according to its step size, set and reset the value. It also needs to save the following fileds:

- Name of parameters.
- Default values of parameters.
- Tuning interval of the parameter.
- Step size of the parameter.

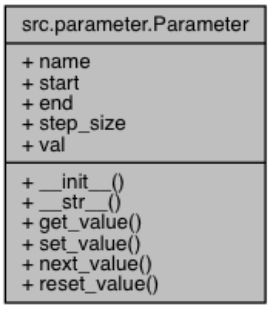

Figure 3: UML diagram for the parameter class.

3.4 Information object

The information object used to pass data between modules contains the following fileds:

- Producing module: the module that produced this information object.
- Consuming module: the module that this information object to be passed to.
- Data path: the path to actual data file.
- Data type: the type of data (one of number, binary and string or user defined data type).
- Data size: the number of data instances.

3.5 File format

A data file has a description file which contains the following fileds:

- The configuration applied to it.
- The timestamp when it is created.
- Data type.
- Data size.

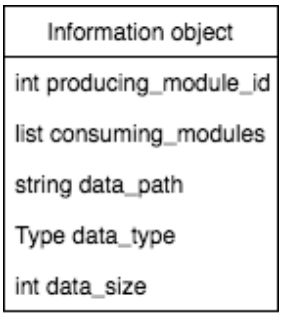

Figure 4: UML diagram for the dataset class.

3.6 Configuration file

We use YAML files to configure the framework.

3.7 Pipeline construction

We assign an unique ID to each module. Then we use these IDs to specify the connection of the pipeline. Therefore, we can easily design very complex pipelines. We use information object to pass data between modules.

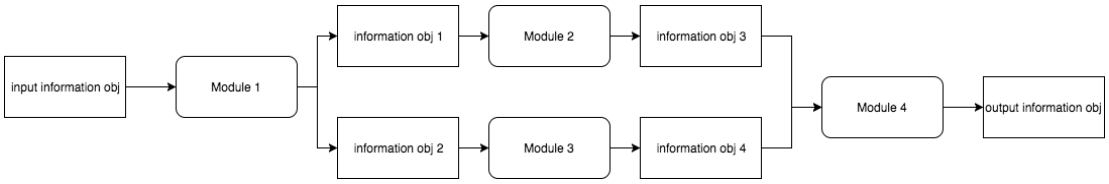

Figure 5: A sample pipeline

Figure 3.7 shows a sample pipeline which passes the input information object through some modules and produces the output information object.