

11-792 Project Report

Nicholas Gekakis, Boyue Li

February 1, 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 able to save intermediate results so that it can be interrupted and 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 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 also needs a description file to specify following fields:

- Unique ID of the module.
- Name of the module.
- Number of inputs.
- Number of outputs.
- Data type of inputs.
- Data type of outputs.
- Number of parameters.
- Default values of parameters.
- Tuning interval of parameters.
- Tuning steps of parameters.

Figure 3.1 is the UML diagram for the abstract

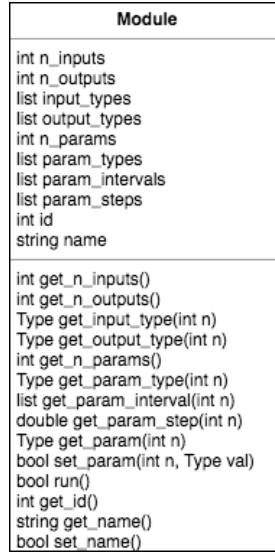


Figure 1: UML diagram for abstract module class

3.2 Information object

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

- 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.

3.3 File format

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

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

Figure 3.3 is the UML diagram for the information object.

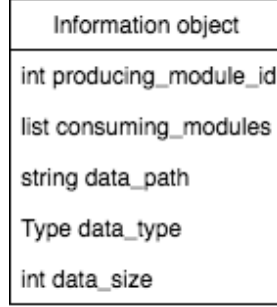


Figure 2: UML diagram for information object

3.4 Configuration file

We use YAML files to configure the framework.

3.5 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.

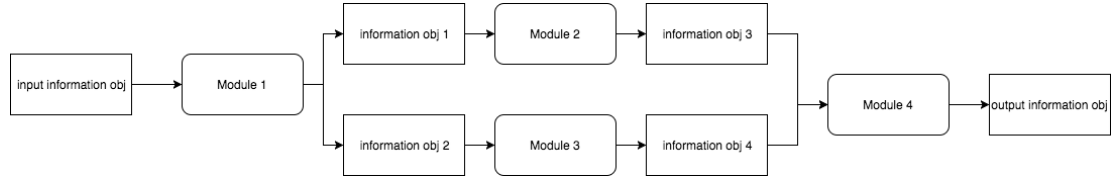


Figure 3: A sample pipeline

Figure 3.5 shows a sample pipeline which passes the input information object through some modules and produces the output information object.

4 Implementation

5 Experiments

5.1 Dataset1

5.2 Dataset2

5.3 Dataset3

6 Conclusion

Acknowledgements