Sparked

Meeting notes and a first draft for the Sparked-Coda Interface

Robin Ruth

13.06.2018

Contents

1 Naming **2**

2 Kafka Interface **2**
2.1 Notes 2

3 Spark REST Interface **3**
3.1 Configuration API 3
3.1.1 List classifiers 3
3.1.2 Return format 3
3.1.3 Notes 4
3.1.4 List validation methods 4
3.1.5 List Evaluation Metrics 4
3.2 Evaluation API 5
3.2.1 Run Task 5
3.2.2 GetStatus 5
3.2.3 GetOutput 5
3.2.4 ClearOutput 6
3.2.5 Notes 6

4 Configuration and Defaults **6**
4.1 Kafka 6
4.2 Spark REST interface 6

1 Naming

The naming was done in just a couple of seconds. Changes are expected.

- Working title for the program that is developed as part of this thesis is **Sparked**.
- What was formerly named **Evaluations** will now be called **Order**. An Order is a list of one or more Tasks.
- What was formerly named **Test Run** will now be named **Task**.
- **Coda Team** is everyone working on the backend application, represented towards Sparked by Christian Geißler.
- The GUID that points to an uploaded file is named **DatafileId**.
- The GUID that points to a running Task evaluation is named **TaskId**.

2 Kafka Interface

To create an Order it will be necessary to upload a file (csv formatted) to a specific Kafka topic. For this a java class has been developed that allows a filestream (or file) -upload to a specified Kafka server and topic. The expected format of the file is topic specific. There will be no validation of the uploaded file on Sparked side. On upload a GUID (**DatafileId**) is specified to reference the data in later steps.

2.1 Notes

Does one upload the data on creation of an Order is that a seperate step? If it is seperate, a database table is required to save uploaded GUIDs and a name has to be given and...
That is unwieldly. Datafile should be uploaded on Order creation unless otherwise spezified by Coda team.

Files are send to the hadoop filesystem on the backend via kafka topics. An upload to kafka does not necessarilly mean, that it is already available in hadoop. This might pose problems if the DatafileId is referenced in a Task evaluation to early. There either needs to be a status function with which Sparked can check the availability of a Datafile on Hadoop or this needs to be dealt with on Coda side.

Is there default data, that does not need an upload?

Does the user select the topic it wants to send the file to, or is there another selection proedure?

3 Spark REST Interface

There will be a single REST endpoint per spark server. There might be several spark servers (See chapter Configuration and Defaults). The endpoints will not be secured. At a later stages there might be an ip whitelisting (and more?).

3.1 Configuration API

These are all endpoints that deliver the values from which the UI is build.

3.1.1 List classifiers

There will be a REST endpoint listClassifiers that takes no parameters and returns a JSON with the available classifiers.

3.1.2 Return format

Return format taken from an example json. This is still subject to change and will be defined bei the CODA team.

```
{
  "classifiers":[
    {
      "id":"org.apache.spark.ml.classification.
         LogisticRegression",
      "name":"LogisticRegression",
      "parameters":[
        {
          "name":"labelCol",
```

```
          "doc":"Some␣documentation␣to␣show␣the␣user"
        },{
          "name":"regParam",
          "doc":"regularization␣parameter␣(>=␣0)"
        }
      ]
    },...
  ]
}
```

The list of classifiers is dynamically expandable. The processing code can deal with a list of variable length. Any classifier must contain an id and a name to be displayed. Additional fields may be added in the design- and development process of Sparked.

3.1.3 Notes

Does the parameter object need a range or accepted values field?

3.1.4 List validation methods

There will be a REST endpoint listValidationMethods that takes no parameters and returns a JSON with the available validation methods.

```
{
  "validators":[
    {
      "id":"com.gt_arc.coda.ml.validation.SimpleSplitValidator",
      "name":"SimpleSplitValidator",
      "parameters":[
        {
          "name":"dummy␣parameter",
          "doc":"for␣future␣extention"
        },...
      ]
    },...
  ]
}
```

3.1.5 List Evaluation Metrics

There will be a REST endpoint listEvaluationMetrics that takes no parameters and returns a JSON with the available evaluation metrics.

```
{
  "metrics":[
    {
      "id":"truepositives_perClass",
      "highValueBetter":true
    },{
      "id":"truenegatives_perClass",
      "highValueBetter":true
    },...
  ]
}
```

3.2 Evaluation API

These are the endpoints that control the evaluation of Tasks and Orders.

3.2.1 Run Task

This function gets a task (DatafileId, Classifier with parameters, evaluation method, metric, taskId) and starts the evaluation. It returns a TaskId. If the task is finished, the old result (file) will be overwritten? (See ClearOutput) A task may only be started, if no task is currently running.

3.2.2 GetStatus

This function gets a taskId and returns the status of the last started task (maybe later: of the task with the corresponding taskId) as a string. Valid status are

- Running
- Finished
- Error(Errorcode, Errormessage)
- more? Does this need to be a complex datatype (JSON)?

3.2.3 GetOutput

This function gets a taskId and returns the result of the last finished task (maybe later: of the result with the corresponding taskId). The return value

is a JSON? Stream? The return format is not defined as of yet. I believe I already have a sample file, have to check that later.

3.2.4 ClearOutput

This tells Coda that the results file is no longer needed.

3.2.5 Notes

What happens if this is not called. Will RunTask pause befor writing a new results file if one is still there or automatically delete the old one?

4 Configuration and Defaults

4.1 Kafka

There will be a config page, that allows the user to add or remove kafka servers. The user can set which kafka server to use. There will be one (or more) kafka servers automatically added at start time. These will be defined in a properties file.

4.2 Spark REST interface

There will be a config page, that allows the user to add or remove the base url to a spark interface. The user can set which spark server to use. There will be one (or more) spark servers automatically added at start time. These will be defined in a properties file.