ADAMS

Advanced **D**ata mining **A**nd **M**achine learning **S**ystem

Module: adams-weka-webservice

Peter Reutemann
Michael Fowke

June 10, 2013

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

Contents

List of Figures

Chapter 1

Set up

The default set up for the webservice is to run on the local machine, or *localhost*. If you want to publish the webservice, either within a LAN or over the internet, then you need to update the URL and/or port that the webservice binds to and is available for clients.

1.1 Client

If you use ADAMS clients, you need to change the WSDL for the WEKA webservice to point the clients to the right address. You can find the WSDL for the webservice at the following location:

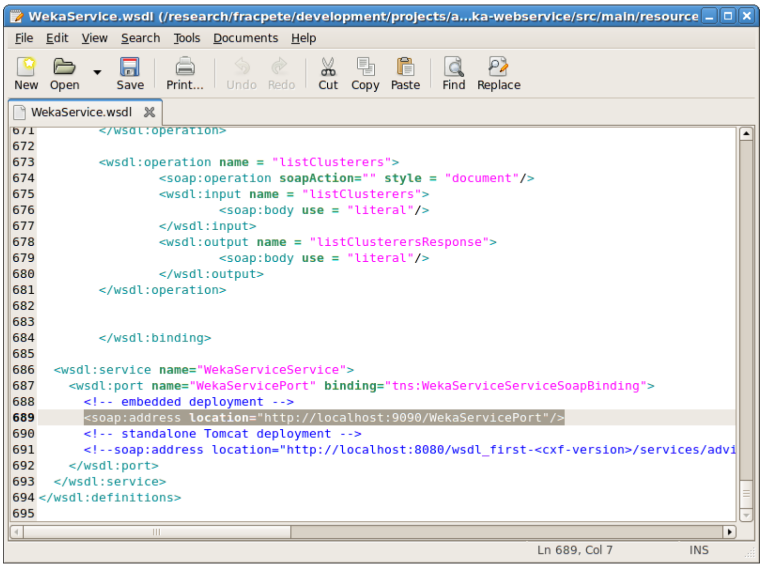
```
resources/wsdl/weka/WekaService.wsdl
```

In this file, change the *location* attribute of the *soap:address* tag appropriately. For instance, from this:

```
<soap:address location="http://localhost:9090/WekaServicePort"/>
```

to this:

```
<soap:address location="http://weka.blah.com:8080/WekaServicePort"/>
```



```
        </wsdl:operation>

        <wsdl:operation name = "listClusterers">
                <soap:operation soapAction="" style = "document"/>
                <wsdl:input name = "listClusterers">
                        <soap:body use = "literal"/>
                </wsdl:input>
                <wsdl:output name = "listClusterersResponse">
                        <soap:body use = "literal"/>
                </wsdl:output>
        </wsdl:operation>


        </wsdl:binding>

  <wsdl:service name="WekaServiceService">
    <wsdl:port name="WekaServicePort" binding="tns:WekaServiceServiceSoapBinding">
      <!-- embedded deployment -->
      <soap:address location="http://localhost:9090/WekaServicePort"/>
      <!-- standalone Tomcat deployment -->
      <!--soap:address location="http://localhost:8080/wsdl_first-<cxf-version>/services/advi
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Figure 1.1: Tag in the WSDL to change for the clients.

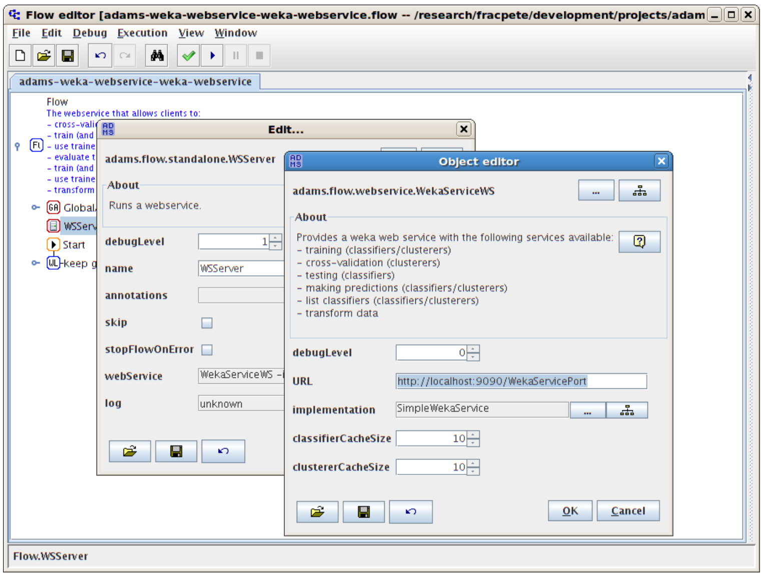
1.2 Server

In addition to the changes to the WSDL as described in the section on the *Client*, the server requires another modification when launched from the flow. In case of the example server workflow[1], you can do this by changing the *URL* property located here:

- *WSServer* actor
- *webService* property
- *URL* property

For instance, if the server's running the webservice is *weka.blah.com* and is supposed to use port 8080, then use the URL *http://weka.blah.com:8080/WekaServicePort*.

NB: Ensure that the port is not blocked by a firewall running on the server or already used otherwise.



adams.flow.standalone.WSServer

About

Runs a webservice.

debugLevel 1

name WSServer

annotations

skip ☐

stopFlowOnError ☐

webService WekaServiceWS -

log unknown

adams.flow.webservice.WekaServiceWS

About

Provides a weka web service with the following services available:
- training (classifiers/clusterers)
- cross-validation (clusterers)
- testing (classifiers)
- making predictions (classifiers/clusterers)
- list classifiers (classifiers/clusterers)
- transform data

debugLevel 0

URL http://localhost:9090/WekaServicePort

implementation SimpleWekaService

classifierCacheSize 10

clustererCacheSize 10

Figure 1.2: Displaying dialog with URL the webservice binds to.

---

[1] adams-weka-webservice-weka-webservice.flow

Chapter 2

Usage

Before you can use the webservice, you need to start the server side. You can do this by simply starting the example server flow[1].

2.1 Classifiers

The webservice offers the following functionality for classifiers:

- *cross-validation* – cross-validates a specified classifier setup on a provided dataset and returns the statistics.[2]
- *train* – trains a classifier setup on a provided dataset and caches the model for future use.[3]
- *test* – evaluates a cached model with a new dataset and returns the statistics.[4]
- *predict* – uses a cached model to generate predictions for a provided dataset.[5]
- *list* – lists all the names of the currently cached classifier models.[6]
- *display* – returns the string representation of a cached classifier model.[7]
- *optimize* – performs parameter-optimization for a classifier using a dataset, for an arbitrary number of search parameter settings using the `weka.classifiers.meta.MultiSearch` meta-classifier. The best setup is then returned.[8]

NB: If the number of cached classifier models is to small, then simply change that setting on the server (*WSServer* → *webService* → *classifierCacheSize*).

2.2 Clusterers

The webservice offers the following functionaliuty for clusterers:

---

[1] adams-weka-webservice-weka-webservice.flow

[2] adams-weka-webservice-crossvalidate-classifier.flow

[3] adams-weka-webservice-train-classifier.flow

[4] adams-weka-webservice-test-classifier.flow

[5] adams-weka-webservice-predict-classifier.flow

[6] adams-weka-webservice-list-classifiers.flow

[7] adams-weka-webservice-display-classifier.flow

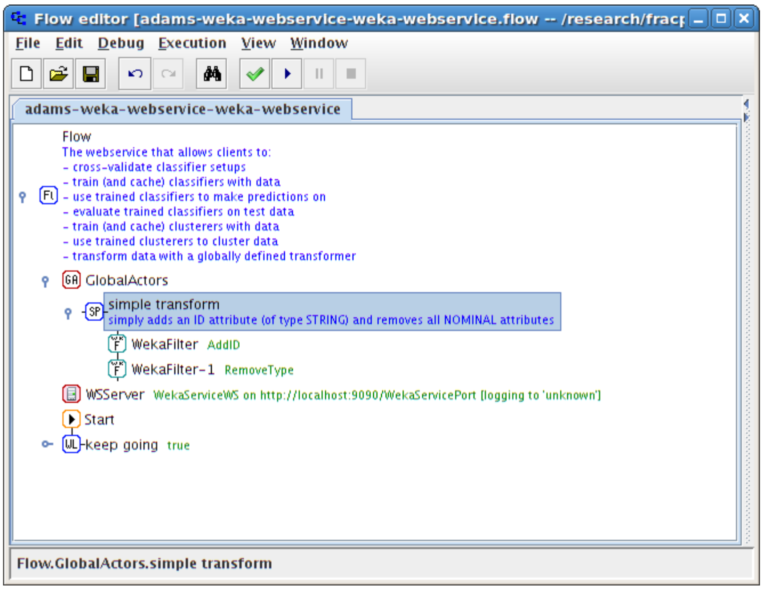
[8] adams-weka-webservice-optimize-classifier-multi-search.flow

- *train* – trains a clusterer setup on a provided dataset and caches the model for future use.[9]
- *predict* – uses a cached model to predict cluster membership for a provided dataset.[10]
- *list* – lists all the names of the currently cached clusterer models.[11]
- *display* – returns the string representation of a cached clusterer model.[12]

NB: If the number of cached classifier models is to small, then simply change that setting on the server (*WSServer* → *webService* → *clustererCacheSize*).

2.3 Filters

The webservice also allows you to transform datasets using WEKA filters. This works by providing a dataset and the name of the global actor on the server. You can define an arbitrary number of global actors that transform datasets. The only requirement is that they accept a *weka.core.Instances* object and generate such one as well.[13]


Flow editor [adams-weka-webservice-weka-webservice.flow -- /research/fracp
File
Edit
Debug
Execution
View
Window
adams-weka-webservice-weka-webservice

```
Flow
The webservice that allows clients to:
- cross-validate classifier setups
- train (and cache) classifiers with data
- use trained classifiers to make predictions on
- evaluate trained classifiers on test data
- train (and cache) clusterers with data
- use trained clusterers to cluster data
- transform data with a globally defined transformer
GlobalActors
simple transform
simply adds an ID attribute (of type STRING) and removes all NOMINAL attributes
WekaFilter AddID
WekaFilter-1 RemoveType
WSServer WekaServiceWS on http://localhost:9090/WekaServicePort [logging to 'unknown']
Start
keep going true
```

Flow.GlobalActors.simple transform

Figure 2.1: Global actor for transforming datasets.

---

[9] adams-weka-webservice-train-clusterer.flow

[10] adams-weka-webservice-predict-clusterer.flow

[11] adams-weka-webservice-list-clusterers.flow

[12] adams-weka-webservice-display-clusterer.flow

[13] adams-weka-webservice-transform.flow

Chapter 3

Development

The default webservice implementation, `adams.flow.webservice.SimpleWekaService`, is a very simply version providing the functionality. All processing happens in a single thread. If you require a version that scales better and can handle more and concurrent requests, then you might want to implement your own backend. This is quite simple, as you only need to create a class that implements the following interfaces:

- `nz.ac.waikato.adams.webservice.weka.WekaService`
- `adams.flow.webservice.OwnedByWekaServiceWS`

Or, you can simply subclass `adams.flow.webservice.SimpleWekaService` and override the method that needs your attention.

The class needs to be placed in the following package:

```
adams.flow.webservice
```

Bibliography

[1] *ADAMS* – Advanced Data mining and Machine learning System
`https://adams.cms.waikato.ac.nz/`

[2] *WSDL* – Web Services Description Language
`http://en.wikipedia.org/wiki/Web_Services_Description_Language`

[3] *SOAP* – Simple Object Access Protocol
`http://en.wikipedia.org/wiki/SOAP`