

We will first setup all the services that are required in our data processing steps. The services need Python and Java installed in the machine where we host the services.

1. addDFT Service
 - a. Install the web.py framework. Follow the steps in the official link <http://webpy.org/install>
 - b. Open the console / terminal
 - c. Locate the addDFTREST.py script files in the python script directory under KarmaProcessing and execute the script to start the web service
python addDFTREST.py 8080
The command starts the addDFT service on port 8080
2. getLabel Service
 - a. Open a new terminal session
 - b. Locate the getLabelPOSTService.py script files in the python script directory under KarmaProcessing and execute the script to start the web service
 - c. Execute the getLabelPOSTService.py script
python getLabelPOSTService.py 8083
3. svmService
Follow the steps in setting up the Karma SVM service.
<https://github.com/InformationIntegrationGroup/karma-svm-service>

Steps using Karma

The steps using karma are divided into two categories:

1. Karma setup
These are tasks that are performed only the first time. They include modeling the different services – addDFT, getLabels, and svmTraining, as well as modeling the individual dataset – AccelerometerSensor and LocationProbe. All transformations and processing done here is recorded by karma and can be played automatically for the other data sets.
2. Karma execution
The Karma execution tasks are ones that are repeated for each datasets. They mainly include tasks like service invocation, joining data sets, and publishing RDFs.

Now we will discuss each part separately, starting with the setup process.

Karma Setup

1. Create a sample CSV file with columns – timestamp, speed, accuracy, magnitude, DFT_E1, DFT_E2, DFT_E3, mode. Add one additional row that contains values. The values can have any random numbers

2. Modeling the getLabel service
 - a. Load the sample CSV file in karma
 - b. Model each column for its semantic type.

☒ **acce:Timestamp of acce:AccelerometerReading1** Hide

Property	Class
<input type="text" value="acce:Timestamp"/>	<input type="text" value="acce:AccelerometerReading1"/>
Properties of Selected Class	Classes with Selected Property
acce:Raw_Timestamp	acce:AccelerometerReading1
acce:Timestamp	acce:AccelerometerReading2 (add)
geo:lat	acce:Input1 (add)
geo:long	acce:Output1 (add)
All Properties	All Classes
acce:acce_x	acce:AccelerometerReading1
acce:acce_y	acce:AccelerometerReading2 (add)
acce:acce_z	acce:DFT_Coefficients1 (add)
acce:Accuracy	acce:Input1 (add)

- c. Set the label for the model by changing the 'Name' of the worksheet. Click the top left corner of the current worksheet to change the Name.
- d. Set the service properties from the drop down menu after clicking the worksheet title bar and publish the model

Set Properties ×

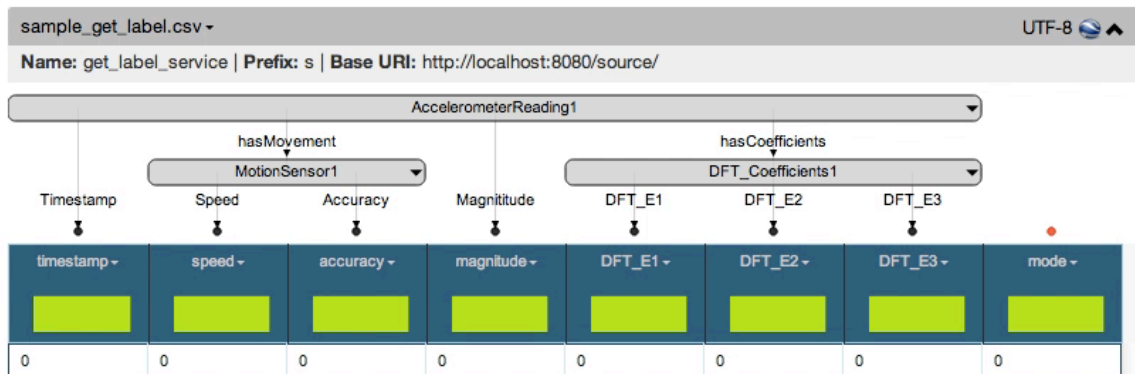
Service URL

Request Method POST ⌵

☐ Invoke service one row at a time

☒ Post whole worksheet

Cancel Submit

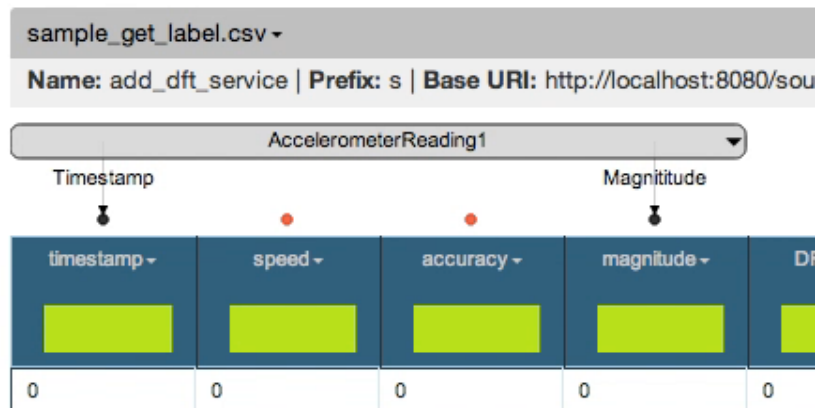


3. Modeling the addDFT service
 - a. Load the sample CSV file
 - b. Model the required columns for its semantic types
 - c. Set the Name of the model
 - d. Set the service properties

The screenshot shows a 'Set Properties' dialog box with a close button (X) in the top right corner. It contains the following fields and options:

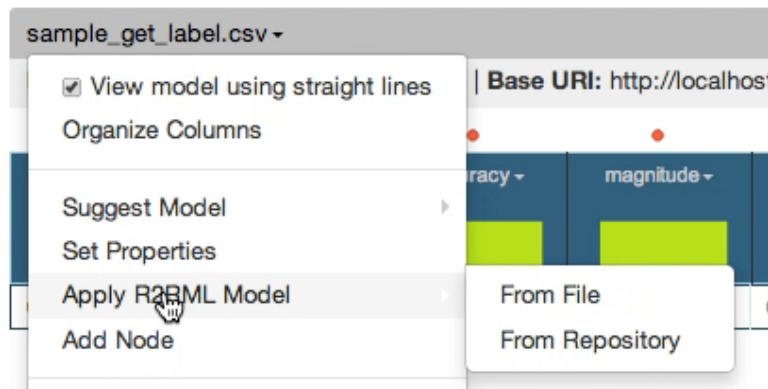
- Service URL:** A text input field containing 'http://localhost:8080/addDFT'.
- Request Method:** A dropdown menu with 'POST' selected.
- Options:** Two radio buttons are present:
 - ☐ Invoke service one row at a time
 - ☒ Post whole worksheet
- Buttons:** 'Cancel' and 'Submit' buttons at the bottom right.

- e. Publish the model



4. Modeling the SVM service

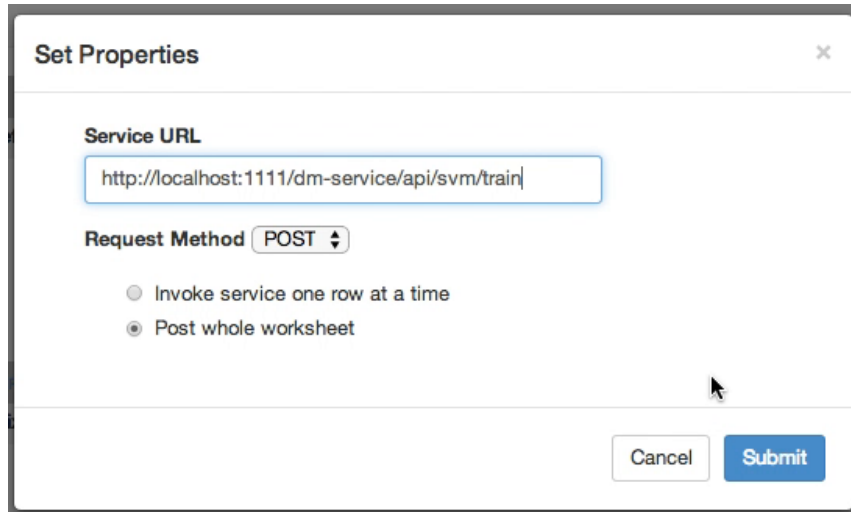
- Load the sample CSV file
- Apply the model for the getLabel service from the repository



- Add the additional semantic type mapping for the “mode” column
- Change the Name of the worksheet by clicking the top left of the worksheet

Apply Model From Repository				
Select Model:				
Name	Publish Time	URL	# Matched Columns	
<input type="radio"/> get_label_service	Mon Aug 04 2014 18:01:02	http://localhost:1234/R2RMLMapping/local/get_label_service-model.ttl	7	
<input type="radio"/> add_dft_service	Mon Aug 04 2014 18:02:01	http://localhost:1234/R2RMLMapping/local/add_dft_service-model.ttl	2	
				<input type="button" value="Cancel"/> <input type="button" value="Apply"/>

- e. Set the service properties
 - Use a fixed model name when setting the training service The model name can be specified as `http://localhost:8081/dm-service/api/svm/train/{the svm model name}`



Set Properties

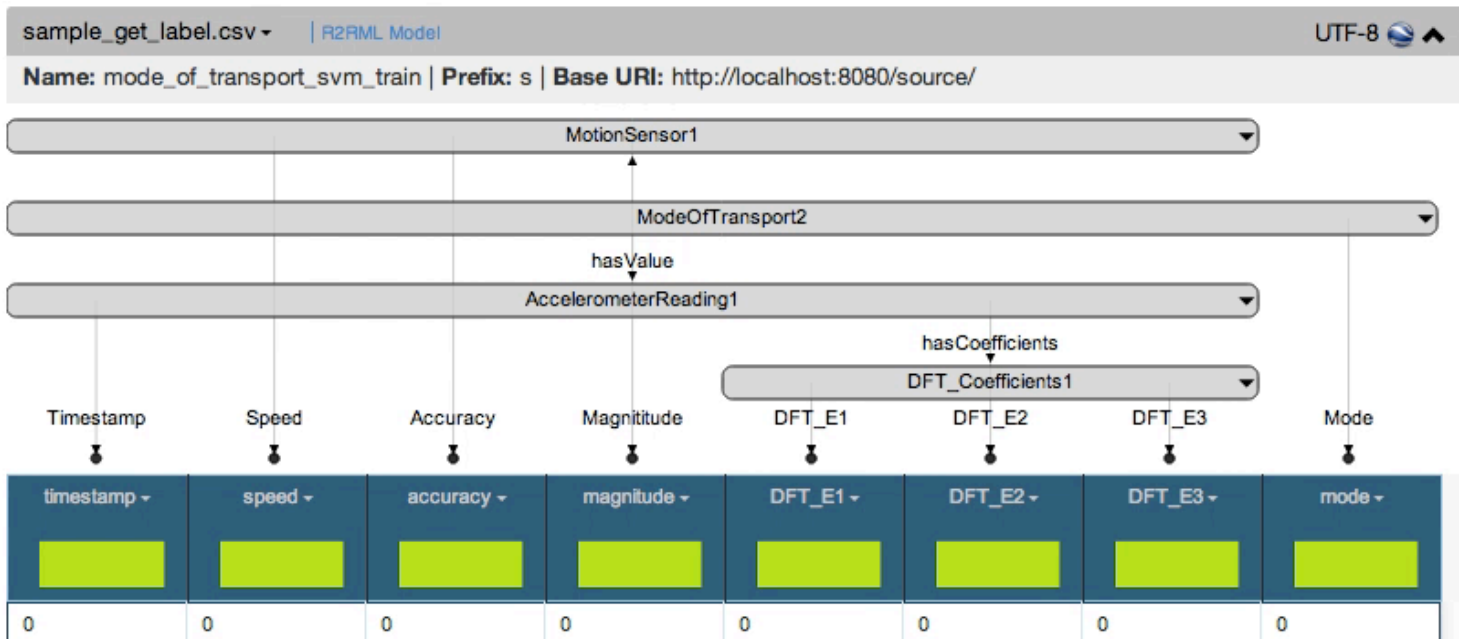
Service URL

Request Method POST

☐ Invoke service one row at a time
☒ Post whole worksheet

Cancel Submit

- f. Publish the model



sample_get_label.csv | R2RML Model UTF-8

Name: mode_of_transport_svm_train | **Prefix:** s | **Base URI:** http://localhost:8080/source/

MotionSensor1

ModeOfTransport2

hasValue

AccelerometerReading1

hasCoefficients

DFT_Coefficients1

Timestamp Speed Accuracy Magnitude DFT_E1 DFT_E2 DFT_E3 Mode

timestamp	speed	accuracy	magnitude	DFT_E1	DFT_E2	DFT_E3	mode
0	0	0	0	0	0	0	0

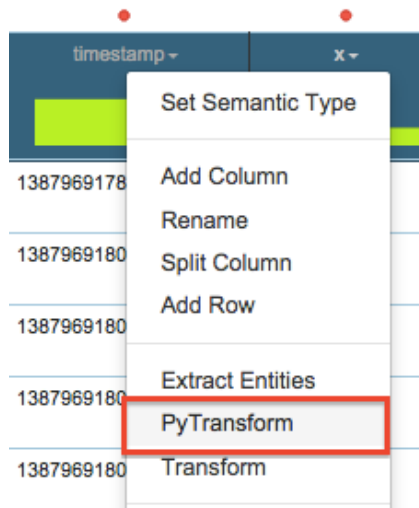
- g. Change the name of the model to 'svm test'
- h. Set the service properties to contain the url for the svm testing
 - Use a previously user model name when setting the testing service url.
 - The model name can be specified as

<http://localhost:8081/dm-service/api/svm/test/{the svm model name}>

- i. Publish the model

5. Modeling the AccelerometerSensor data set.

- a. Load the AccelerometerSensor1.csv file and import only 5 rows into the worksheet. We import a small set of data to speed up the process of modeling.
- b. Add the magnitude column by using a Python transform



PyTransform Column

☐ Change existing column:

accuracy

☒ Name of new column:

```
1 import math
2 x = float(getValue("x"))
3 y = float(getValue("y"))
4 z = float(getValue("z"))
5 return str(math.sqrt(x*x + y*y + z*z))
```

On Error:

View Errors

Preview results for top 5 rows

Cancel

Save

- c. Set semantic types for the required columns
- d. Publish the model

Name: AccelerometerSensorProbe | **Prefix:** s | **Base URI:** http://loc

AccelerometerReading1

Timestamp Magnitude

timestamp ▾	speed ▾	accuracy ▾	magnitude ▾

6. Modeling the AccelerometerData with DFT coefficient
 - a. Load the sample CSV file
 - b. Create Uris using python transform for the AccelerometerReading class

PyTransform Column

☐ Change existing column: timestamp
☒ Name of new column: acce_url

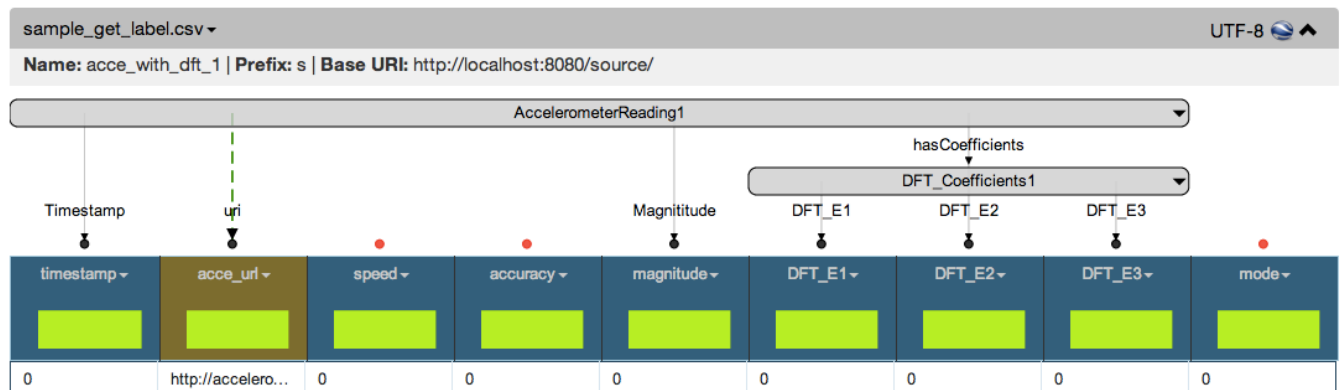
```

1 return 'http://accelerometer.co/data/' + getValue("timestamp")
  
```

On Error:

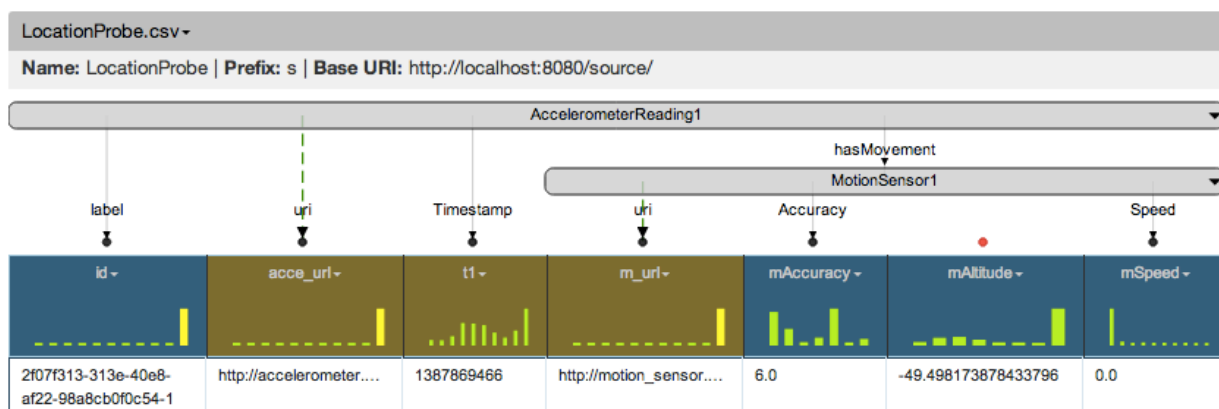
http://accelerometer.co/data/0

- c. Set semantic types for the required columns
- d. Publish the model



7. Modeling the LocationProbe.csv file

- Load the LocationProbe.csv file and import only 1 row.
- Generate uris using python transform for MotionSensor and AccelerometerReading classes.
- Set the sematic types for the required columns
- Publish the model



Karma Execution

1. Load the LocationProbe.csv file and import all rows by setting 0 in the rows text field
2. Apply the model “LocationProbe” form the repository

Apply Model From Repository ✕

Select Model:

Name	Publish Time	URL	# Matched Columns
<input checked="" type="radio"/> LocationProbe	Tue Aug 05 2014 18:23:04	http://localhost:1234/R2RMLMapping/local/LocationProbe-model.ttl	3
<input type="radio"/> AccelerometerSensorProbe	Mon Aug 04 2014 18:09:15	http://localhost:1234/R2RMLMapping/local/AccelerometerSensorProbe-model.ttl	1
<input type="radio"/> acce_with_dft_1	Tue Aug 05 2014 19:42:58	http://localhost:1234/R2RMLMapping/local/acce_with_dft_1-model.ttl	1
<input type="radio"/> acce_with_dft_2	Tue Aug 05 2014 18:27:50	http://localhost:1234/R2RMLMapping/local/acce_with_dft_2-model.ttl	1
<input type="radio"/> acce_with_dft_n_motion	Mon Aug 04 2014 19:28:45	http://localhost:1234/R2RMLMapping/local/acce_with_dft_n_motion-model.ttl	1
<input type="radio"/> acce_with_dft_temp_4	Tue Aug 05 2014 19:43:53	http://localhost:1234/R2RMLMapping/local/acce_with_dft_temp_4-model.ttl	1
<input type="radio"/> acce_with_dft_temp_5	Tue Aug 05 2014 19:47:50	http://localhost:1234/R2RMLMapping/local/acce_with_dft_temp_5-model.ttl	1

Cancel Apply

3. Publish the RDF

Publish RDF ✕

RDF Graphs Create New Context

Create New Graph

http://localhost.com/worksheets/LocationProbe

☐ Replace Existing Data

☒ Append

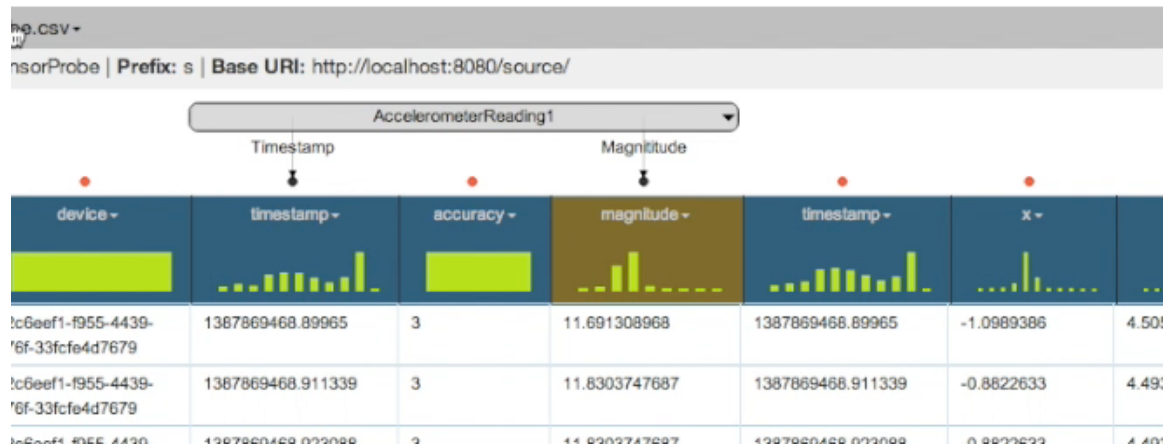
☒ Add Inverse Properties

☒ Index Data to Support Joins

Cancel Publish

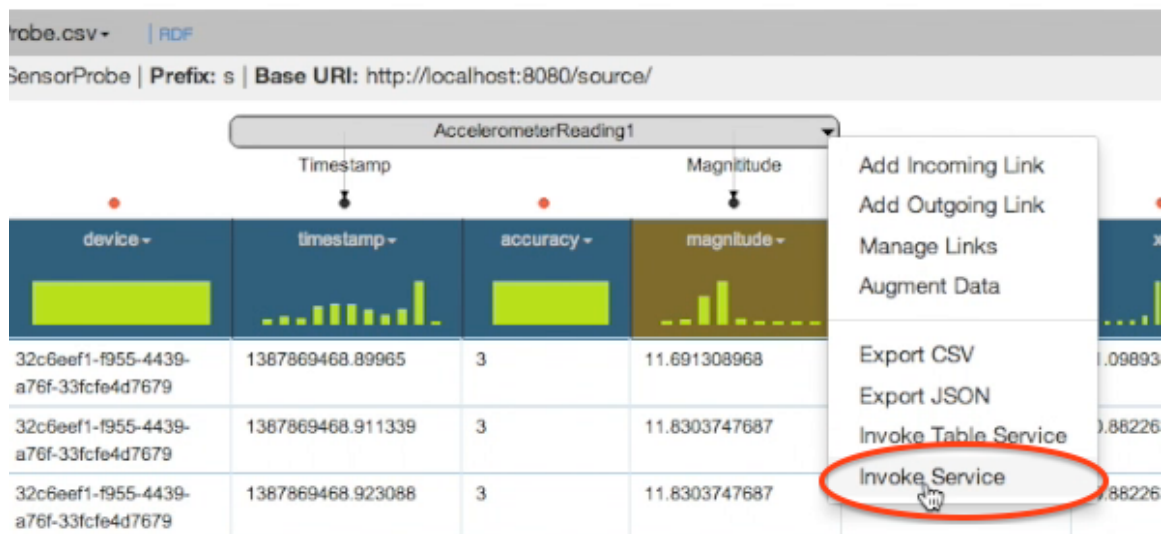
4. Load the AccelerometerSensor1.csv file

5. Apply the model “AccelerometerSensorModel” from the repository



6. Publish the RDF

7. Invoke the addDFT service



8. Karma lists the services that could be invoked from the given model

Invoke Services

Service Uri	Method	Arguments
⊙ http://localhost:8080/addDFT	POST (invokeWithWholeWorksheet)	2

SPARQL End Point

http://localhost:1234/openrdf-sesame/repositories/karma_data

Graph

- ✓ Current Worksheet
- All graphs
- http://localhost.com/worksheets/acce_raw
- http://localhost.com/worksheets/LocationProbe

Cancel Invoke

9. The output of the DFT service is loaded into a new worksheet

table_service_results_04-Aug-2014_192727707 - UTF-8

Name: table_service_results_04-Aug-2014_192727707 | Prefix: s | Base URI: http://localhost:8080/source/

timestamp	magnitude	DFT_E1	DFT_E2	DFT_E3
1387869469	11.691308968	136.686705385	139.957767168	139.957767168
1387869485	12.0369934127	144.889210417	154.717868621	157.276679743
1387869493	11.4100099525	130.188327116	130.188327116	126.902099251
1387869499	13.8061809213	190.610631632	188.240769534	218.256882131
1387869522	12.5691860936	157.984439056	146.690473229	127.597477346

10. Apply the “Acce_with_dft” model on the new worksheet

table_service_results_04-Aug-2014_192727707 - UTF-8

Name: acce_with_dft_1 | Prefix: s | Base URI: http://localhost:8080/source/

AccelerometerReading1

Timestamp

uri

Magnitude

hasCoefficients

DFT_Coefficients1

DFT_E1

DFT_E2

DFT_E3

timestamp	acce_uri	magnitude	DFT_E1	DFT_E2	DFT_E3
1387869469	http://accelerometer....	11.691308968	136.686705385	139.957767168	139.957767168
1387869485	http://accelerometer....	12.0369934127	144.889210417	154.717868621	157.276679743
1387869493	http://accelerometer....	11.4100099525	130.188327116	130.188327116	126.902099251

11. Publish the RDF for this worksheet in a new graph named “acce_with_dft_1”
12. Select “Augment Data” option to perform the join between MotionSensor data and the AccelerometerData

The screenshot shows the Apache Spark Table Service interface. At the top, the table is named 'acce_with_dft_1' with a Base URI of 'http://localhost:8080/source/'. Below this, a diagram illustrates the data structure: an 'AccelerometerReading1' class has properties 'Timestamp', 'url', 'Magnitude', and 'hasCoefficients'. The 'hasCoefficients' property points to a 'DFT_Coefficients1' class, which has properties 'DFT_E1', 'DFT_E2', and 'DFT_E3'. A context menu is open over the 'DFT_Coefficients1' class, with the 'Augment Data' option highlighted. Other options in the menu include 'Add Incoming Link', 'Add Outgoing Link', 'Manage Links', 'Export CSV', 'Export JSON', 'Invoke Table Service', and 'Invoke Service'.

timestamp	acce_url	magnitude	DFT_E1	DFT_E2	DFT_E3
1387869469	http://accelerometer....	11.691308968	136.686705385	139.957767168	139.957767168
1387869485	http://accelerometer....	12.0369934127	144.889210417	154.717868621	157.276679743
1387869493	http://accelerometer....	11.4100099525	130.188327116	130.188327116	126.902099251
1387869499	http://accelerometer....	13.8061809213	190.610631632	188.240769534	218.256882131
1387869522	http://accelerometer....	12.5691860936	157.984439056	146.690473229	127.597477346

13. Karma will list the properties that were matched

The screenshot shows the Karma interface with a window titled 'Augment data for acce:AccelerometerReading'. The window displays a table of properties and their matches. The 'acce:hasMovement' property is selected, showing 207 matches with the 'acce:MotionSensor' class. The 'acce:hasValue' property is also listed, showing 2 matches with the 'acce:ModeOfTransport' class. The 'Same As' property is set to 'owl:sameAs'. The 'Submit' button is highlighted.

Property	Class	# Matches (approx)	Direction
<input type="checkbox"/> acce:Timestamp		629	Outgoing
<input type="checkbox"/> acce:hasCoefficients	acce:DFT_Coefficients	629	Outgoing
<input type="checkbox"/> acce:Magnitude		629	Outgoing
<input checked="" type="checkbox"/> acce:hasMovement	acce:MotionSensor	207	Outgoing
<input type="checkbox"/> acce:hasValue	acce:ModeOfTransport	2	Incoming

"Same As" property
owl:sameAs

Cancel Submit

14. Select the “hasMoment” property. The AccelerometerReading class has this object property that maps it to the MotionSensor class, which has data properties like “accuracy” and “speed”

15. Once the data is augmented, a new column for MotionSensor url with the semantic types are added to the worksheet

table_service_results_04-Aug-2014_192727707- | RDF UTF-8

Name: acce_with_dft_1 | Prefix: s | Base URI: http://localhost:8080/source/

AccelerometerReading1

hasMovement
MotionSensor1
uri

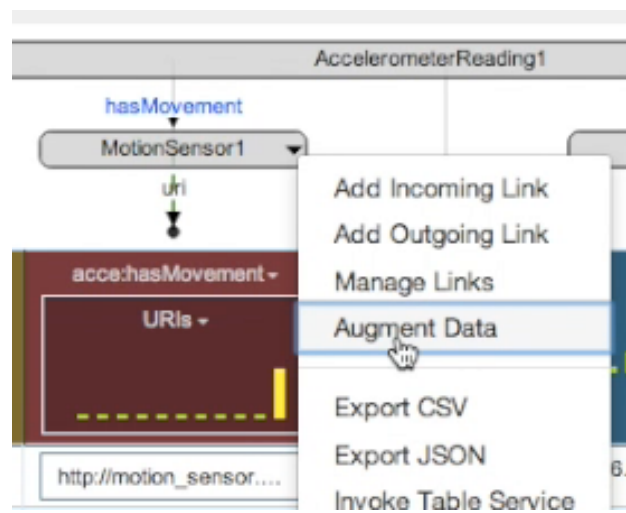
hasCoefficients
T_Coefficients1

Timestamp
uri

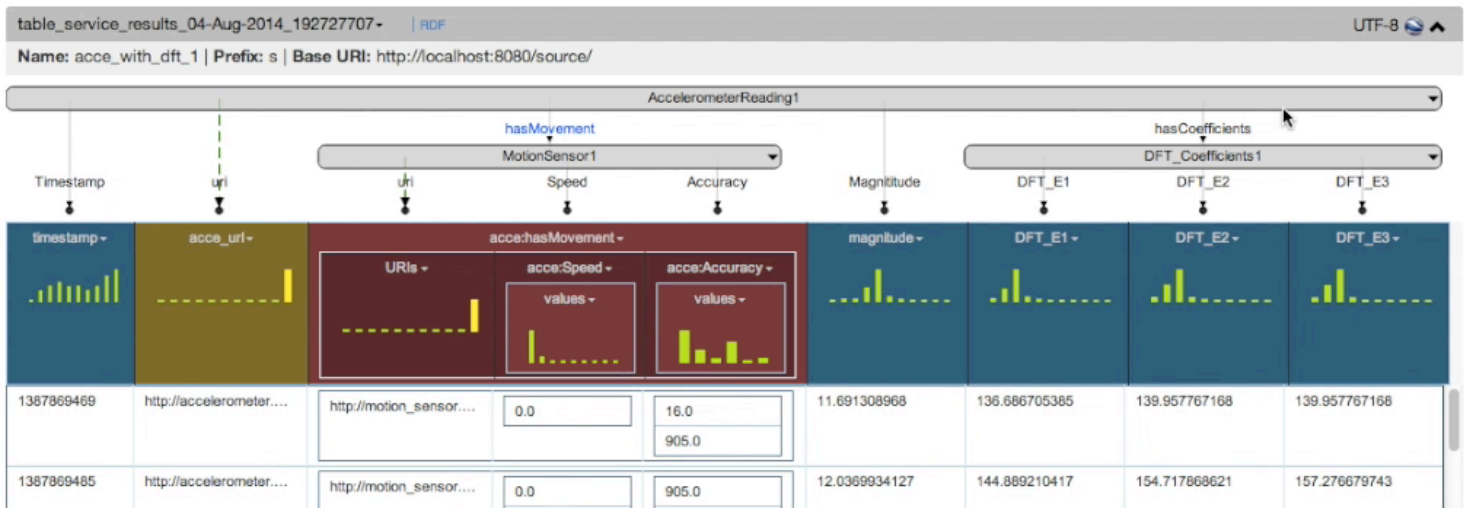
Magnitude
DFT_E1
DFT_E2
DFT_E3

timestamp	acce_url	acce:hasMovement URIs	magnitude	DFT_E1	DFT_E2	DFT_E3
1387869469	http://accelerometer...	http://motion_sensor....	11.691308968	136.686705385	139.957767168	139.957767168
1387869485	http://accelerometer...	http://motion_sensor....	12.0369934127	144.889210417	154.717668621	157.276679743
1387869493	http://accelerometer...		11.4100099525	130.188327116	130.188327116	126.902099251

16. Now we again augment the MontionSensor url to get its data properties for “speed” and “accuracy”



17. After joining we have all our columns that are required for the getLabel service



18. Since this model was created by augmenting data sets, we will publish the model

19. Publish the RDF

20. Now we again invoke a service from the AccelerometerReading class

21. Karma shows two services that could be invoked with the model. We select getLabel

Invoke Services

Service Uri	Method	Arguments
• http://localhost:8083/getLabel	POST (invokeWithWholeWorksheet)	7
• http://localhost:8080/addDFT	POST (invokeWithWholeWorksheet)	2

SPARQL End Point

http://localhost:1234/openrdf-sesame/repositories/karma_data

Graph

- ✓ Current Worksheet
- All graphs
- http://localhost.com/worksheets/acce_raw
- http://localhost.com/worksheets/acce_with_dft_1
- http://localhost.com/worksheets/acce_dft_motion
- http://localhost.com/worksheets/LocationProbe

Cancel Invoke

22. The results are loaded into a new worksheet.

table_service_results_04-Aug-2014_192918897

UTF-8

Name: table_service_results_04-Aug-2014_192918897 | Prefix: s | Base URI: http://localhost:8080/source/

accuracy	speed	timestamp	magnitude	DFT_E2	DFT_E3	DFT_E1	mode
16.0	0.0	1367869469	11.691308968	139.957767168	139.957767168	136.686705385	walking
905.0	0.0	1367869469	11.691308968	139.957767168	139.957767168	136.686705385	walking
905.0	0.0	1367869485	12.0369934127	154.717868621	157.276679743	144.889210417	walking

23. We now apply the mode of transport model on this worksheet.

Import Manage Models Reset

Apply Model From Repository

Select Model:

Name	Publish Time	URL	# Matched Columns
<input checked="" type="radio"/> mode_of_transport_svm_train	Mon Aug 04 2014 18:03:12	http://localhost:1234/R2RMLMapping/local/mode_of_transport_svm_train-model.ttl	8
<input type="radio"/> get_label_service	Mon Aug 04 2014 18:01:02	http://localhost:1234/R2RMLMapping/local/get_label_service-model.ttl	7
<input type="radio"/> acce_with_dft_1	Mon Aug 04 2014 19:03:46	http://localhost:1234/R2RMLMapping/local/acce_with_dft_1-model.ttl	5
<input type="radio"/> acce_with_dft_n_motion	Mon Aug 04 2014 19:28:45	http://localhost:1234/R2RMLMapping/local/acce_with_dft_n_motion-model.ttl	5
<input type="radio"/> add_dft_service	Mon Aug 04 2014 18:02:01	http://localhost:1234/R2RMLMapping/local/add_dft_service-model.ttl	2
<input type="radio"/> AccelerometerSensorProbe	Mon Aug 04 2014 18:09:15	http://localhost:1234/R2RMLMapping/local/AccelerometerSensorProbe-model.ttl	1
<input type="radio"/> LocationProbe	Mon Aug 04 2014 18:14:32	http://localhost:1234/R2RMLMapping/local/LocationProbe-model.ttl	1

Cancel Apply

24. Publish the RDF

25. If this is the first data set, then we invoke the SVM training service. The output of the training returns us

table_service_results_05-Aug-2014_195832584		UTF-8
Name: table_service_results_05-Aug-2014_195832584 Prefix: s Base URI: http://localhost:8080/source/		

summary

value

values

attribute

[1] "/Users/shri/Downloa... distribution- 9.1.0.v20131115/tem...	raw
<div>auto</div> <div>bus</div> <div>stationary</div> <div>walking</div>	Levels
svm.default(x = x, y = y, type = c_type, kernel = kerneltype)	summary
4	Number of Classes
545	Number of Support Vectors
linear	SVM-Kernel
0.1428571	gamma
Rscripts/svmTraining.R	CommandName
TrainData_linear_05- Aug- 2014_195831917.csv	InputFile
1	cost
http://localhost:1111/...	InputFilePath