Specific to Orange’s implementation on internal server

Specific notes: <https://github.com/justwjr/Recommendation-Engine/blob/master/orange_project/daily_log.cpp>

Look in the #5/4 and #5/5 section

Screenshots: https://github.com/justwjr/Recommendation-Engine/tree/master/orange\_project

<https://github.com/actionml/analysis-tools>

if everything has been configured before{

after you have a deployed engine, export all the event possibilities to a HDFS

pio export --appid 10 --output /Tara/universalRecommenderNew/store\_events

then run split:

split:

SPARK\_HOME=/PredictionIO-0.10.0-incubating/vendors/spark-1.5.1-bin-hadoop2.6 PYTHONPATH=/PredictionIO-0.10.0-incubating/vendors/spark-1.5.1-bin-hadoop2.6/python:/PredictionIO-0.10.0-incubating/vendors/spark-1.5.1-bin-hadoop2.6/python/lib/py4j-0.9-src.zip ./map\_test.py split

next, Train and Deploy: Import the "train" dataset into the EventServer, train, and deploy the model. This can also be scripted but is not part of these scripts.

pio import --appid 10 --input /Tara/universalRecommenderNew/store\_events/train.1

pio build --verbose

pio train

finally, run test, which will output the report.xlsx file!:

test:

SPARK\_HOME=/PredictionIO-0.10.0-incubating/vendors/spark-1.5.1-bin-hadoop2.6 PYTHONPATH=/PredictionIO-0.10.0-incubating/vendors/spark-1.5.1-bin-hadoop2.6/python:/PredictionIO-0.10.0-incubating/vendors/spark-1.5.1-bin-hadoop2.6/python/lib/py4j-0.9-src.zip ./map\_test.py test –all

}

#Analysis Tools

These scripts do analysis of input data and run cross-validation tests using the Universal Recommender. The analysis show information about usage data, and the CV tests use a technique called MAP@k to rest the predictiveness of the many types of *indicators* used in a deployment of the UR. The tools run in 2 phases:

1. **Split**: Split the data into train and test splits, calculate usage stats.
2. **Train and Deploy**: Import the "train" dataset into the EventServer, train, and deploy the model. This can also be scripted but is **not** part of these scripts.
3. **Analyze**: Run several forms of MAP@k predictiveness tests on the splits, collect stats, and create a report for Excel.

**Note:** This toolset does #1 and #3 but does not do #2.

If any change is made to the engine.json, repeat #1-3

#Setup

Install Spark, PredictionIO v0.9.6 or greater, and the Universal Recommender v0.3.0 or greater. Make sure pio status completes with no errors and the integration-test for the UR runs correctly.

Make sure there’s a deployed engine running that outputs valid recommendations, check an example request call – there should be scored courses like this:

|  |
| --- |
| curl -H "Content-Type: application/json" \ |
|  |

-d '{ "user": "de22719ee50124fdb8c09882bc19fde8", "num": 4 }' http://0.0.0.0:8000/queries.json



1. Install Python and check the version

python --v

if the version is less than 2.7.9 upgrade to the most recent stable version of python using systems package management tools like apt-get for Ubuntu linux or brew for the Mac.

1. Install Python libraries using the Python package manager found [here](https://pip.pypa.io/en/stable/installing/)
2. sudo pip install numpy scipy pandas ml\_metrics predictionio tqdm click openpyxl
3. Setup Spark and Pyspark paths in .bashrc (linux) or .bash\_profile Mac.

<http://blog.prabeeshk.com/blog/2014/10/31/install-apache-spark-on-ubuntu-14-dot-04/>

1. export SPARK\_HOME=/path/to/spark
2. export PYTHONPATH=$SPARK\_HOME/python/:$SPARK\_HOME/python/build/:$PYTONPAHTH

**Run Analysis Script**

Analysis script should be run from UR (Universal recommender) folder. It uses two configuration files:

* engine.json (configuration of UR, this file is used to take event list and primary event)
* config.json (all other configuration including engine.json path if necessary)

##Configuration options

the config.json file in the docker container, Tara/universalRecommenderNew, has the correct fields filled in.

config.json has the following structure:

{

"engine\_config": "./engine.json",

"splitting": {

"version": "1",

"source\_file": "hdfs:...<PUT SOME PATH>...",

"train\_file": "hdfs:...<PUT SOME PATH>...train",

"test\_file": "hdfs:...<PUT SOME PATH>...test",

"type": "date",

"train\_ratio": 0.8,

"random\_seed": 29750,

"split\_event": "<SOME NAME>"

},

"reporting": {

"file": "./report.xlsx"

},

"testing": {

"map\_k": 10,

"non\_zero\_users\_file": "./non\_zero\_users.dat",

"consider\_non\_zero\_scores\_only": true,

"custom\_combos": {

"event\_groups": [["ev2", "ev3"], ["ev6", "ev8", "ev9"]]

}

},

"spark": {

"master": "spark://<some-url>:7077"

}

}

* **engine\_config** - file to be used as engine.json (see configuration of UR)
* **splitting** - this section is about splitting data into train and test sets
  + **version** - version to append to train and test file names (may be helpful is different test with different split configurations are run)
  + **source\_file** - file with data to be split
  + **train\_file** - file with training data to be produced (note that version will be append to file name)
  + **test\_file** - file with test data to be produced (note that version will be append to file name)
  + **type** - split type (can be time in this case eventTime will be used to make split or random)
  + **train\_ratio** - float in (0..1), share of training samples
  + **random\_seed** - seed for random split
  + **split\_event** - in case of **type** = "date", this is event to use to look for split date, all events with this name are ordered by eventTime and time which devides all such events into first **train\_ratio** and last (1 - **train\_ratio**) sets is used to split all the rest data (events with all names) into training set and test set
* **reporting** - reporting settings
  + **file** - excel file to write report to
  + **csv\_dir** - directory name for csv reporting
  + **use\_uuid** - append to csv file names unique uuid associated with script run (can be useful to manage different results and reports)
* **testing** - this section is about different tests and how to perform them
  + **map\_k** - maximum map @ k to be reported
  + **non\_zero\_users\_file** - file to save users with scores != 0 after first run of test set with primary event, this set may be much smaller then initial set of users so saving it and reusing can save much time
  + **consider\_non\_zero\_scores\_only** (default: true) whether take into account only users with non-zero scores (i.e. users for which recommendations exist)
  + **custom\_combos**
    - **event\_groups** - groups of events to be additionally tested if necessary
* **spark** - Apache Spark configuration
  + **master** - for now only master URL is configurable

##Split the Data

Get data from the EventServer with:

pio export --appid 9 --output /Tara/universalRecommenderNew/store\_events

if there’s an error that the folder already exists, delete it or rename.



Use this command to run split of data into "train" and "test" sets

split:

SPARK\_HOME=/PredictionIO-0.10.0-incubating/vendors/spark-1.5.1-bin-hadoop2.6 PYTHONPATH=/PredictionIO-0.10.0-incubating/vendors/spark-1.5.1-bin-hadoop2.6/python:/PredictionIO-0.10.0-incubating/vendors/spark-1.5.1-bin-hadoop2.6/python/lib/py4j-0.9-src.zip ./map\_test.py split

SPARK\_HOME=/usr/local/spark PYTHONPATH=/usr/local/spark/python:/usr/local/spark/python/lib/py4j-0.9-src.zip ./map\_test.py split

Additional options are available:

* --csv\_report - put report to csv file not excel
* --intersections - calculate train / test event intersection data (**Advanced**)

##Train a Model

The above command will create a test and training split in the location specified in config.json. Now you must import, setup engine.json, train and deploy the "train" model so the rest of the MAP@k tests will be able to query the model.

##Test and Analysis

To run tests

test:

SPARK\_HOME=/PredictionIO-0.10.0-incubating/vendors/spark-1.5.1-bin-hadoop2.6 PYTHONPATH=/PredictionIO-0.10.0-incubating/vendors/spark-1.5.1-bin-hadoop2.6/python:/PredictionIO-0.10.0-incubating/vendors/spark-1.5.1-bin-hadoop2.6/python/lib/py4j-0.9-src.zip ./map\_test.py test --all

SPARK\_HOME=/usr/local/spark PYTHONPATH=/usr/local/spark/python:/usr/local/spark/python/lib/py4j-0.9-src.zip ./map\_test.py test --all

Additional options are available and may be used to run not all test:

* --csv\_report -
* --dummy\_test - run dummy test
* --separate\_test - run test for each separate event
* --all\_but\_test - run test with all events and tests with all but each every event
* --primary\_pairs\_test - run tests with all pairs of events with primary event
* --custom\_combos\_test - run custom combo tests as configured in config.json
* --non\_zero\_users\_from\_file - use list of users from file prepared on previous script run to save time

##Generated Report

###Old ipython analysis script This is not recommended old approach to run ipython notebook.

IPYTHON\_OPTS="notebook" /usr/local/spark/bin/pyspark --master spark://spark-url:7077