
The Universal Recommender

The Universal Recommender (UR) is a new type of collaborative filtering recommender based on an algorithm that can use data from a wide variety of user preference indicators—it is called the Correlated Cross-Occurrence algorithm. Unlike matrix factorization embodied in things like MLlib's ALS, CCO is able to ingest any number of user actions, events, profile data, and contextual information. It then serves results in a fast and scalable way. It also supports item properties for building flexible business rules for filtering and boosting recommendations and can therefore be considered a hybrid collaborative filtering and content-based recommender.

Most recommenders can only use conversion events, like buy or rate. Using all we know about a user and their context allows us to much better predict their preferences.

Requirements—The Universal Recommender Has Moved!

The UR 0.8.0+ requires the Harness Machine Learning Server. 0.7.3 and before run in PredictionIO 0.12.1. This repo is now build into Harness as a pre-packaged Engine. See Upgrading from PIO to Harness

Documentation

- The Universal Recommender
- The Correlated Cross-Occurrence Algorithm
- The Universal Recommender Slide Deck
- Multi-domain predictive AI or how to make one thing predict another

All docs for the Universal Recommender are here and are hosted at <https://github.com/actionml/docs.actionml.com>. If you wish to change or edit the docs make a PR to that repo.

Contributions

Contributions are encouraged and appreciated. Create a push request (PR) against the [develop](#) branch of the git repo. We like to keep new features general so users will not be required to change the code of the UR to make use of the new feature. We will be happy to provide guidance or help via the GitHub PR review mechanism.

Version Log

UR v0.8.0+ The UR Has Moved!

The Universal Recommender has moved. Future versions will be included as a built-in Engine for the new Harness Machine Learning Server. the UR v0.8.0+ is data compatible with previous versions that are integrated with PredictionIO. This means you can export from UR+PIO and import into UR+Harness. See Upgrading from PIO to Harness

Adds:

- Realtime changes to item properties with \$set.
- Ease of having more than one predictive model (Harness feature)
- TTL based trim of old events. After a user defined period, events may be dropped from the collection without running a separate process (Harness feature)
- Containerized deployment
- many other features are inherited from

Git Tag: v0.7.3

Adds:

- Switched to using `python3` wherever `python` is invoked. Before this branch it was assumed that the environment mapped `python` to `python3` which is required for PIO 0.12+ and the UR 0.7+. Since many distros have `python` invoke python 2.7 and `python3` is needed to invoke python 3.6 we now do also.
- Support for cross recommendations like “people who have viewed similar to you have bought these items”. Used to help find things in a browsing/searching scenario.

Git Tag: 0.7.2

Adds:

- Pagination support in query using `"from": 0, "num": 2` will return 2 recs from the first available, `"from": 2, "num": 2` will return 2 starting at the 3rd since `"from"` is 0 based.

Git Tag: 0.7.1

This tag take precedence over 0.7.0, which should not be used. Changes:

-
- Removes the need to build Mahout from the ActionML's fork and so is much easier to install.
 - Fixes a bug in the integration test which made it fail for macOS High Sierra in East Asian time zones.

Git Tag: 0.7.0

This README Has Special Build Instructions!

This tag is for the UR integrated with PredictionIO 0.12.0 using Scala 2.11, Spark 2.1.x, and most importantly Elasticsearch 5.x. Primary differences from 0.6.0:

- Faster indexing, and queries due to the use of Elasticsearch 5.x
- Faster model building due to speedups in the ActionML fork of Mahout, which requires the user to build Mahout locally. This step will be removed in a later version of the UR.
- Several upgrades such as Scala 2.10 → Scala 2.11, Python 2.7 → Python 3
- Spark 2.1.x support, PIO has a minor incompatibility with Spark 2.2.x
- Prediction 0.12.0 support
- Requires Elasticsearch 5.x. using the ES REST APIs exclusively now, enabling ES authentication use optionally. ES 5.x also improves indexing and query performance over previous versions.
- Fixed a bug in exclusion rules based on item properties

WARNING: Upgrading Elasticsearch or HBase will wipe existing data if any, so follow the special instructions below before installing any service upgrades.

Upgrade from UR v0.6.0 Instructions

You must build PredictionIO with the default parameters so just run `./make-distribution` this will require you to install Scala 2.11 and Python 3 (as the default Scala and Python). You can also run up to Spark 2.1.x (but not 2.2.x), ES 5.5.2 or greater (but 6.x has not been tested), Hadoop 2.6 or greater, you can get away with using older versions of services except ES must be 5.x. If you have issues getting pio to build and run send questions to the PIO mailing list.

Backup your data, moving from ES 1 to ES 5 will delete all data!!!! Actually even worse it is still in HBase but you can't get at it so to upgrade do the following:

- `pio export` with `pio < 0.12.0` =====**Before upgrade!**=====
- `pio data-delete` all your old apps =====**Before upgrade!**=====
- build and install pio 0.12.0 including all the services =====**The point of no return!**=====
- `pio app new ...` and `pio import ...` any needed datasets

Once PIO is running test with `pio status` and `pio app list`. To test your setup and UR integration, run `./examples/integration-test` from the URs home.

Config for PIO 0.12.0 and the UR 0.7.0

a sample of `pio-env.sh` that works with one type of setup is below, but you'll have to change paths to match yours. This example show the new way to configure for Elasticsearch 5.x, which uses a new port number:

```
1 #!/usr/bin/env bash
2
3 # SPARK_HOME: Apache Spark is a hard dependency and must be configured.
4 # using Spark 2.2.1 here
5 SPARK_HOME=/usr/local/spark
6
7 # ES_CONF_DIR: You must configure this if you have advanced
8 # configuration for
9 # using ES 5.6.3
10 ES_CONF_DIR=/usr/local/elasticsearch/config
11
12 # HADOOP_CONF_DIR: You must configure this if you intend to run
13 # PredictionIO
14 # using hadoop 2.8 here
15 HADOOP_CONF_DIR=/usr/local/hadoop/etc/hadoop
16
17 # HBASE_CONF_DIR: You must configure this if you intend to run
18 # PredictionIO
19 # using HBase 1.2.x here or whatever the highest numbered stable
20 # release is
21 HBASE_CONF_DIR=/usr/local/hbase/conf
22
23 # Filesystem paths where PredictionIO uses as block storage.
24 PIO_FS_BASEDIR=$HOME/.pio_store
25 PIO_FS_ENGINESDIR=$PIO_FS_BASEDIR/engines
26 PIO_FS_TMPDIR=$PIO_FS_BASEDIR/tmp
27
28 # Storage Repositories
29 PIO_STORAGE_REPOSITORIES_METADATA_NAME=pio_meta
30 PIO_STORAGE_REPOSITORIES_METADATA_SOURCE=ELASTICSEARCH
31
32 PIO_STORAGE_REPOSITORIES_MODELDATA_NAME=pio_
33 PIO_STORAGE_REPOSITORIES_MODELDATA_SOURCE=LOCALFS
34
35 PIO_STORAGE_REPOSITORIES_APPDATA_NAME=pio_appdata
36 PIO_STORAGE_REPOSITORIES_APPDATA_SOURCE=ELASTICSEARCH
37
38 PIO_STORAGE_REPOSITORIES_EVENTDATA_NAME=pio_eventdata
39 PIO_STORAGE_REPOSITORIES_EVENTDATA_SOURCE=HBASE
```

```
36
37 # ES config
38 PIO_STORAGE_SOURCES_ELASTICSEARCH_TYPE=elasticsearch
39 PIO_STORAGE_SOURCES_ELASTICSEARCH_HOSTS=localhost
40 PIO_STORAGE_SOURCES_ELASTICSEARCH_PORTS=9200 # <===== notice 9200 now
41 PIO_STORAGE_SOURCES_ELASTICSEARCH_CLUSTERNAME=elasticsearch_xyz #
    <===== should match what you have in you ES config file
42 PIO_STORAGE_SOURCES_ELASTICSEARCH_HOME=/usr/local/elasticsearch
43
44 PIO_STORAGE_SOURCES_LOCALFS_TYPE=localfs
45 PIO_STORAGE_SOURCES_LOCALFS_HOSTS=$PIO_FS_BASEDIR/models
46
47 PIO_STORAGE_SOURCES_HBASE_TYPE=hbase
48 PIO_STORAGE_SOURCES_HBASE_HOME=/usr/local/hbase
```

Build Mahout After PredictionIO!

Mahout has speedups for the Universal Recommender's use that have not been released yet so you will have to build from source. To make this easy we have a fork hosted here, with special build instructions. Make sure you are on the "sparse-speedup" branch and follow instructions in the README.md

Build the Universal Recommender

- download the UR from here be sure move to the 0.7.0 tag.
- replace the line: `resolvers += "Local Repository"at "file:///Users/pat/.custom-scala-m2/repo"` with your path to the local mahout build. **the UR will not build unless this line is changed, this is expected**
- build the UR with `pio build` or run the integration test to get sample data put into PIO `./examples/integration-test`

v0.6.0

This is a major upgrade release with several new features. Backward compatibility with 0.5.0 is maintained. **Note:** We no longer have a default `engine.json` file so you will need to copy `engine.json.template` to `engine.json` and edit it to fit your data. See the Universal Recommender Configuration docs.

- **Performance:** Nearly a 40% speedup for most model calculation, and a new tuning parameter that can yield further speed improvements by filtering out unused or less useful data from model building. See `minEventsPerUser` in the UR configuration docs.

-
- **Complimentary Purchase aka Item-set Recommendations:** “Shopping-cart” type recommendations. Can be used for wishlists, favorites, watchlists, any list based recommendations. Used with list or user data.
 - **Exclusion Rules:** now we have business rules for inclusion, exclusion, and boosts based on item properties.
 - **PredictionIO 0.11.0:** Full compatibility, but no support for Elasticsearch 5, an option with PIO-0.11.0.
 - **New Advanced Tuning:** Allows several new per indicator / event type tuning parameters for tuning model quality in a more targeted way.
 - **Norms Support:** For large dense datasets norms are now the default for model indexing and queries. This should result in slight precision gains, so better results.
 - **Mahout 0.13.0 Support:** the UR no longer requires a local build of Mahout.
 - **GPU Support:** via Mahout 0.13.0 the core math of the UR now supports the use of GPUs for acceleration.
 - **Timeout Protection:** Queries for users with very large histories could cause a timeout. We now correctly limit the amount of user history that is used as per documentation, which will all but eliminate timeouts.
 - **Bug Fixes:** The use of `blackListEvents` as defined in `engine.json` was not working for an empty list, which should and now does disable any blacklisting except explicit item blacklists contained in the query.

v0.5.0

- **Apache PIO Compatible:** The first UR version compatible with Apache PredictionIO-0.10.0-incubating. All past versions do not work and should be upgraded to this. The ActionML build of PIO is permanently deprecated since it is merged with Apache PIO.

v0.4.2 Replaces 0.4.1

- **Fixes bug** when a `pio build` failure triggered by the release of Apache PIO. If you have problems building v0.4.0 use this version. It is meant to be used with PredictionIO-0.9.7-aml.
- **Requires a custom build of Apache Mahout:** instructions on the doc site This is temporary until the next Mahout release, when we will update to 0.4.3 (uses predicitionio-0.9.7-aml) and 0.5.0 (which uses predictionio-0.10.0 from Apache)

v0.4.0

- This version requires PredictionIO-0.9.7-aml found [here](#).
- **New tuning params** are now available for each “indicator” type, making indicators with a small number of possible values much more useful—things like gender or category-preference. See docs for configuring the UR and look for the [indicators](#) parameter.
- **New forms of recommendations backfill** allow all items to be recommended even if they have no user events yet. Backfill types include random and user defined. See docs for configuring the UR and look for the [rankings](#) parameter.

v0.3.0

- This version requires PredictionIO-0.9.7-aml from the ActionML repo [here](#).
- **Implements a moving time window if events:** Now supports the [SelfCleanedDataSource](#) trait. Adding params to the [DataSource](#) part of [engine.json](#) allows control of de-duplication, property event compaction, and a time window of event. The time window is used to age out the oldest events. Note: this only works with the ActionML fork of PredictionIO found in the repo mentioned above.
- **Parameter changed:** [backfillField](#): [duration](#) to accept Scala Duration strings. This will require changes to all engine.json files that were using the older # of seconds duration.
- **Event-types used in queries:** added support for indicator predictiveness testing with the MAP@k tool. This is so only certain mixes of user events are used at query time.
- **Bug fix:** which requires that the [typeName](#) in engine.json is required be ["items"](#), with this release the type can be any string.

v0.2.3

- removed isEmpty calls that were taking an extremely long time to execute, results in considerable speedup. Now the vast majority of [pio train](#) time is taken up by writing to Elasticsearch. This can be optimized by creating and ES cluster or giving ES lots of memory.

v0.2.2

- a query with no item or user will get recommendations based on popularity
- a new integration test has been added
- a regression bug where some ids were being tokenized by Elasticsearch, leading to incorrect results, was fixed. **NOTE: for users with complex ids containing dashes or spaces this is an important fix.**

-
- a dateRange in the query now takes precedence to the item attached expiration and available dates.

v0.2.1

- date ranges attached to items will be compared to the prediction servers current data if no date is provided in the query.

v0.2.0

- date range filters implemented
- hot/trending/popular used for backfill and when no other recommendations are returned by the query
- filters/bias < 0 caused scores to be altered in v0.1.1 fixed in this version so filters have no effect on scoring.
- the model is now hot-swapped in Elasticsearch so no downtime should be seen, in fact there is no need to run `pio deploy` to make the new model active.
- it is now possible to have an engine.json (call it something else) dedicated to recalculating the popularity model. This allows fast updates to popularity without recalculating the collaborative filtering model.
- Elasticsearch can now be in cluster mode

v0.1.1

- ids are now exact matches, for v0.1.0 the ids had to be lower case and were subject to tokenizing analysis so using that version is not recommended.

v0.1.0

- user and item based queries supported
- multiple usage events supported
- filters and boosts supported on item properties and on user or item based results.
- fast writing to Elasticsearch using Spark
- convention over configuration for queries, defaults make simple/typical queries simple and overrides add greater expressiveness.

Known issues

- see the github issues list

License

This Software is licensed under the Apache Software Foundation version 2 license found here:
<http://www.apache.org/licenses/LICENSE-2.0>