MOOCdb Documentation

Release 0

ALFA

January 15, 2015

Contents

This documentation is for the MOOCdb project. The documentation provides detailed instructions for users to be able to use different software and analytic engines developed as part of the MOOCdb project. Our goal is to provide this as a open source repository to enable prolific development of MOOC data analytics to improve teaching and learning outcomes. We also refer to this open source software repository as MOOC Data Science Commons.

Contents:

CHAPTER 1

Translation software documentation

The purpose of this user guide is to make the users understand the following operations:

How the translation is done from the raw data files provided by Coursera and edX. How to run the software and Different assumptions and heuristics used during the translation Currently translation software is available for the following three platforms:

1.1 edX

The following step by step instructions will guide you to process the tracking log files and get into MOOCdb format, we provide the following detailed instructions. First step is to install a machine with required packages and disk space.

1.1.1 Step 1: Installing the required packages or downloading a VM

- Option 1: To run the software you may want to download a VM from Amazon cloud.

  This virtual machine image comes with all packages installed which are required to run the MOOCdb pipeline. To get a link to the image and request the '.pem' file, please email kalyan@csail.mit.edu.

  Important:

  - When instantiating this virtual machine on Amazon or locally, please provision the disk space (hard disk) such that there is atlease three times the size of the decrypted- uncompressed file size of the tracking logs.

  - Option 2: Install all the packages on your local machine

    The following packages are required on a MOOCdb machine

    1. Install **Unidecode** package available here
    2. Install **ijson** package which can be found here
    3. Install **python-setuptools**
    4. Install **pip** using **sudo easy_install pip**
    5. Install **pandas**
    6. Download the code from MOOCdb github:
       - Openedx diagnosis
       - Openedx apipe

  - Openedx qpipe

Note:

- Make sure your Pandas version is higher than **0.14.0**. If it is below that you would have to update Pandas by running:

  pip install pandas –upgrade

- You may have to upgrade **numpy** and **numexpr** before upgrading **pandas**, if upgrading **pandas** gives you an error. The command to upgrade numpy and numexpr is the same:

  pip install numpy –upgrade

  pip install numexpr –upgrade

1.1.2 Step 2: Processing the tracking logs

If your course is through edX you would get the files shown below. The most important and perhaps most tedious is processing the tracking log files. Some of the files listed below in the table could be representative of what MIT delivers to us. But tracking_log.json is the largest file and contains the detailed clickstream events. These are the events which are recorded along with event type.

1. Unzip tracking log file:

   All raw data files in **'data/raw/<course_name>'** have the same prefix in the format of '**<course_name>__<creation date>**', we will call the prefix '**COURSE_PREFIX**'.

   From within the tracking log file folder, run the command:

   ```
   gzip -d COURSE_PREFIX__tracking_log.json.gz
   ```

   This will extract the tracking log file into .json format, ready to be piped.

2. If there are multiple log files, merge all the log files for a single course into one log file.

3. Run JSON to relation code (a.k.a apipe):

   This tutorial covers the transfer of JSON tracking log file to CSV files. The code is written by Andreas Paepcke from Stanford. JSON tracking log file is stored with other raw data files. We will call the raw data files as '**raw data**' and the output CSV as '**intermediary CSV**'.

   Let us suppose that we want to pipe the course named '**<course_name>**'. We assume that the raw data is stored in the folder:

   ```
   /.../<course_name>/log_data/
   ```

   Create a folder called intermeidary_csv under the folder named '**<course_name>**'

   ```
   /.../<course_name>/intermediary_csv/
   ```

   Create another folder called moocdb_csv under the folder named '**<course_name>**'

   ```
   /.../<course_name>/moocdb_csv/
   ```

4. Launch the piping:

   From within the import.openedx.json_to_relation folder, run command:

   ```
   bash scripts/transformGivenLogfiles.sh /.../<course_name>/intermediary_csv/
   ```

```
/../<course_name>/log_data/COURSE_PREFIX__tracking_log.json
```

As show in the command above, transfromGivenLogFiles.sh takes two arguments. First argument is the path to the destination folder, and second argument is the tracking log json file to pipe. '**/.../**' represents the path to the directory where the <course_name> folder is located on your machine. The command may run for a few hours and depends on the size of the raw json tracking log file.The output csv files will be in '**/.../<course_name>/intermediary_csv**'. The following gives an example of the output csv files produced for link5_10x course:

```
link5_10x_trace_merged.2014-11-02T23_46_45.622627_28028.sql
link5_10x_trace_merged.2014-11-02T23_46_45.622627_28028.sql_ABExperimentTable.csv
link5_10x_trace_merged.2014-11-02T23_46_45.622627_28028.sql_AccountTable.csv
link5_10x_trace_merged.2014-11-02T23_46_45.622627_28028.sql_AnswerTable.csv
link5_10x_trace_merged.2014-11-02T23_46_45.622627_28028.sql_CorrectMapTable.csv
link5_10x_trace_merged.2014-11-02T23_46_45.622627_28028.sql_EdxTrackEventTable.csv
link5_10x_trace_merged.2014-11-02T23_46_45.622627_28028.sql_EventIpTable.csv
link5_10x_trace_merged.2014-11-02T23_46_45.622627_28028.sql_InputStateTable.csv
link5_10x_trace_merged.2014-11-02T23_46_45.622627_28028.sql_LoadInfoTable.csv
link5_10x_trace_merged.2014-11-02T23_46_45.622627_28028.sql_StateTable.csv
```

5. Run relation to MOOCdb (a.k.a qpipe):

   This tutorial covers the transfer of CSV files as output by Andreas Paepcke's json_to_relation to MOOCdb CSV files. We will call the source CSV as '**intermediary CSV**' and the output CSV as '**MOOCdb CSV**'.

   Let us suppose that we want to pipe to MOOCdb the course named **'<course_name>'**. We assume that the course's log file has been processed by json_to_relation, and that the output files are stored in the folder :

   /.../<course_name>/intermediary_csv/

   We want the MOOCdb CSV to be written to folder

   /.../<course_name>/moocdb_csv/

   (a) Edit **import.openedx.qpipe/config.py**

   (b) **QUOTECHAR**: The quote character used in the intermediary CSV files. Most commonly a single quote (')

   (c) **TIMESTAMP_FORMAT**: describes the timestamp pattern used in '***_EdxTrackEventTable.csv**' intermediary CSV file. See python doc to understand syntax.

   (d) **COURSE_NAME**: The name of the folder containing the intermediary CSV files. Here it is **'<course_name>'**.

   (e) **CSV_PREFIX**: All the intermediary CSV file names in '**/.../<course_name>/intermediary_csv/**' folder share a common prefix that was generated when running JSON to relation. This prefix is also the name of the only '.sql' file in the folder. For example, in the above case this prefix would be :

   ```
   link5_10x_trace_merged.2014-11-02T23_46_45.622627_28028.sql
   ```

   (f) **DOMAIN: the domain name of the course platform URL, most commonly they are https://www.edx.org or h**
   (No slash at the end of the domain name). To be sure, you can look at the URL's appearing in '***_EdxTrackEventTable.csv**' intermediary CSV file.

Note: The variables not mentioned in the tutorial must simply be left untouched.

6. Launch the piping:

   When the variables mentioned above have been correctly edited in `config.py`, the script is ready to launch. From within the `import.openedx.qpipe` folder, run the command:

   ```
   time python main.py
   ```

7. Delete log file:

   When the piping is done, if everything went well, go to the output directory **'/.../<course_name>/moocdb_csv/'** and delete the **'log.org'** file that takes a lot of space.

8. Load course into MySQL:

   Copy the file **'/.../<course_name>/moocdb_csv/6002x_2013_spring/moocdb.sql'** to **'/.../<course_name>/moocdb_csv/'** folder. Change directory to **'/.../<course_name>/moocdb_csv/'**. Replace '6002x_spring_2013' by '<course_name>' in `moocdb.sql` file.

   Run the command:

   ```
   mysql -u root -p --local-infile=1 < moocdb.sql
   ```

   This creates a database named '**<course_name>**' in MySQL, and loads the CSV data into it.

Translation details

Some examples contextualized presented via the two urls below show for an actual course show how the translation from raw JSON logs to MOOCdb takes place

- Interaction Scenario
- Problem Check Example

More details can be found in Quentin Agrens thesis here

1.2 Coursera

1.2.1 Setting Up the Coursera-MOOCdb Transformation Scripts

Pre-requisites

In order to set up MOOCDB and transform courses, the following are required:

- Elementary Python programming
- Installing Python packages
- Elementary usage of any MySQL client
- Creating MySQL databases and importing data from SQL dumps into MySQL databases
- Familiarity with the types of databases Coursera exports and what kind of data is stored in each

Installing Dependencies

Before setting up MOOCdb, you will need to install Python, and a number of python modules listed below. Python MOOCdb was tested and found to work on:

- Python 2.7

Python Modules

You will need to install the following Python modules:

- **MySQLdb** (mandatory): pip install MySQL-python
- **phpserialize** (req. for Coursera data piping): pip install phpserialize

Setting Up MOOCdb

Before converting a course to MOOCdb, there are two databases you need to create and import into.

1.2.2 MOOOCdb Core

This database is required by all transformation tasks, but no task will ever alter its data. Hence, you can import it once and use it many times. Call this database **moocdb_core**, and avoid using this name for any other databases. After creating this empty database, import the file `.../core/moocdb_core.sql` located inside the piping scripts directory.

Note: The **moocdb_core** does not have to reside on the same machine on which the output database will be built.

1.2.3 The Schema

This database is also required for all transformations, and can be created once and used many times. Call this database **moocdb_clean**, and avoid using this name for any other databases. Unlike moocdb_core, this database has to be created on the same machine on which the output databases will be created. After you create **moocdb_clean** as an empty database, import the file `.../core/schema.sql` located inside the piping scripts directory. Using MOOCdb

You are now ready to transform a Coursera course to MOOCdb.

To do so, you have to supply a set of parameters telling the program where to find the input data, where to put the output data, and specifying other required information and options. To transform a course, you simply have a call a **main** function with a dictionary of these parameters. The scripts expect this dictionary to follow a very specific structure. To familiarize yourself with the parameter dictionary structure and how to call the **main** transformation function, please look at the **run_coursera.py** file located in the piping scripts directory. In this file, the main function is first imported from main.py. Second, the parameter dictionary is constructed. The variable holding it in the example is called **vars**. Finally, main(vars) is called. This should start the transformation. Below is an explanation of the parameters you need to specify. Before, we start, however, you will need to watch out for the following.

Coursera changed its export format at some point in the past. Previously, table columns referring to anonymized user IDs (such as in the lecture_submission_metadata) in the anonymized_general database were called **anon_user_id**. User IDs were anonymized in the anonymized_forum database as well and placed in columns called **forum_user_id**. If your course following this format, we will refer to its format as **coursera_1**.

More recently, Coursera changed the **anon_user_id** to **session_user_id**, and stopped anonymizing user IDs in the anonymized_forum database. If your course follows this format, we will refer to its format as **coursera_2**.

These naming changes will also be visible in the hash_mapping table directly. If your course does not strictly adhere to one of these 2 formats, this program might not be able to handle its data. Please request a new export of your course from Coursera. You should receive it in the second format regardless of when the course ran.

source (Coursera)

Contains the parameters related to the input (Coursera side)

- **platform_format**: **'coursera_1'** or **'coursera_2'**
- **course_id**: For **coursera_2** courses, set this to None. For **coursera_1** courses, lookup the numeric part of any of the kvs tables in the anonymized_general database, and write it here as a number. For example, if you see a table called kvs.136.quizzes, then set this parameter to 136
- **course_url_id**: This must be the session ID of the course, as a string (ex: **algorithms-001**)
- **host**, **user**, **password**, **port**: The MySQL connection parameters to the server hosting the coursera databases. If you do not know which port the MySQL server uses, try the default value (3306).
- **hash_mapping_db**: The name of the course hash-mapping database
- **general_db**: The name of the course anonymized-general database
- **forum_db**: The name of the course anonymized-forum database

core

Contains the following parameters required to connect to the MOOCdb-core database.

- **host**, **user**, **password**, **port**: The MySQL connection parameters to the server hosting the moocdb_core database. If you do not know which port the MySQL server uses, try the default value (3306)

target (Independent of source platform)

Contains the MOOCdb output and clean database connection parameters.

- **host**, **user**, **password**, **port**: The MySQL connection parameters to the server hosting the MOOCdb output and clean databases. If you do not know which port the MySQL server uses, try the default value (3306)
- **db**: The name of the MOOCdb output database

options

Sets transformation options.

- **log_path**: The path in which the log file for the transformation should be placed. This should be a path to a directory not a file. The log file for a single transformation task will be placed inside that directory and will be named based on the course_url_id and the task start date/time.
- **log_to_console**: True | False, whether or not log messages should also be written to the console.
- **debug**: True | False. If True, the script will only transform data for a limited number of users, speeding up the run for development purposes.
- **num_users_debug_mode**: The number of users to transform in debug mode

Checking the Outputs:

After a transformation process has exited, please check the console and log file to verify that it completed successfully. Also, please visit the MySQL server hosting the output database and make sure its tables are populated.

CHAPTER 2

Collaborative platforms

This documentation is for folks to install different open source platforms that MOOCdb project has. They include Collaborative visualization platform - MOOCviz, platform to crowd source feature and variable formation - Feature Factory, platform for getting labels from the crowd - LabelMe-Text. These instructions allow folks to create their own instantiation of the platform for internal use. The instructions will enable to download the software and deploy a server for each one of these platforms.

2.1 MOOCviz

For info on MOOCviz, including purpose, features, and architecture, read the following papers:

MoocViz: A Large Scale, Open Access, Collaborative, Data Analytics Platform for MOOCs

MOOCdb: An collaborative environment for MOOC data

MOOCviz 2.0: A Collaborative MOOC Analytics Visualization Platform

Consult Kalyan for access to the last two.

If you have questions about anything and can't find the answer here on this wiki, feel free to email me at pwrtsoccer@gmail.com and I will do my best to help you.

2.1.1 Setup

Follow the tutorial to set up for developing and upgrading MOOCviz.

Server

I tried my best to setup the server according to this guide (http://robmclarty.com/blog/how-to-setup-a-production-server-for-rails-4), however it is not all the same. The server uses Apache 2 to deploy the Rails 4 MOOCviz web application.

Capistrano

Capistrano (https://github.com/capistrano/capistrano) is a remote server automation tool. At one point I tried integrating Capistrano into MOOCviz using this guide (http://robmclarty.com/blog/how-to-deploy-a-rails-4-app-with-git-and-capistrano), but there were more important things to be done and I had at least a somewhat consistent method for upgrading the server with new features. However, it would be much better to have Capistrano integrated as it offers lots of cool features and is less error prone than updating the server yourself. I pushed the work I had done to another branch on GitHub here.

Tutorial for MOOCviz:

Clone from GitHub

Clone the project from GitHub onto your local machine. Once you've done this, start a new branch to make any changes, push the branch to GitHub when you are finished, and use GitHub to compare and merge the branch into master. This way you can see merged branches and easily compare any changes you've made in case you need to backtrack.

Add your keypair on Nimbus

OpenStack is a cloud computing platform that hosts virtual machines. Nimbus is CSAIL's OpenStack account. The virtual machine instance that hosts MOOCviz is named moocviz-prod. The IP address is 128.52.128.146, which is registered under MIT's WebDNS for the domain name moocviz.csail.mit.edu. For enough RAM to run a Rails-Apache server, the instance needs an s1.4 core size.

To be able to SSH into the instance, you'll need to have the moocviz keypair private file. Due to a misunderstanding when creating the snapshot, the associated keypair is prestonthompson, so you'll need the prestonthompson.pem file. Talk to Kalyan about getting this.

You won't need my password, just the keypair file, and if at some point a new instance is created, you can simply use another keypair and forget about mine.

Managing the server

Upgrading MOOCviz

When you've finished developing a new feature, pushed it to GitHub master branch, and need to upgrade MOOCviz production, fire up a terminal and follow these steps:

Backup the production database on your computer scp -i path/to/prestonthompson.pem ubuntu@128.52.128.146:/var/www/MOOCdb/moocenimages/db/production.sqlite3 path/to/store/db/backup

SSH into the server ssh -i path/to/prestonthompson.pem ubuntu@128.52.128.146

Change into the MOOCviz Rails directory ubuntu$ cd /var/www/MOOCdb/moocenimages

Backup the current branch git branch backupXX

Pull the changes from master on GitHub git pull

Migrate the database if there are any migrations bundle exec rake db:migrate RAILS_ENV=production

Precompile the changed assets bundle exec rake assets:precompile RAILS_ENV=production

Restart Apache sudo service apache2 restart

Restarting when OpenStack Fails

If for some reason Nimbus fails and needs to be restarted (e.g. power outage), follow these steps to get the server back up and running:

Log into NIMBUS take a snapshot of the current instance (named moocviz-prod)

Wait until the snapshot is completely finished, then delete that instance

Launch a new instance from that snapshot. The instance should be set to IP address 128.52.128.146 using an s1.4 core size.

2.2 Feature Factory

2.2.1 Feature Factory Instructions

The following two github links will be useful reference:

- Feature Factory
- Feature Discovery

2.2.2 Fresh setup:

The ultimate guide for a fresh setup can be found here.

```
sudo apt-get update --fix-missing
sudo apt-get install python-pip python-dev build-essential
sudo pip install --upgrade pip
sudo apt-get install apache2 apache2.2-common apache2-mpm-prefork apache2-utils libexpat1
sudo apt-get install libapache2-mod-wsgi
sudo service apache2 restart
sudo pip install django celery django-celery django-kombu hoover django_extensions
sudo apt-get install python-mysqldb

sudo nano /etc/apache2/apache2.conf #add this
ServerAdmin kiarashplusplus@gmail.com
ServerName featurefactory.csail.mit.edu
WSGIScriptAlias / var/www/featurefactory.csail.mit.edu/index.wsgi

sudo mkdir /var/www/featurefactory.csail.mit.edu
sudo nano /var/www/featurefactory.csail.mit.edu/index.wsgi
import os
import sys
```

path = '/home/ubuntu/alfa'

If path not in sys.path then use the following command:

```
sys.path.append(path)
os.environ['DJANGO_SETTINGS_MODULE'] = 'alfa.settings'
import django.core.handlers.wsgi
application = django.core.handlers.wsgi.WSGIHandler()
copy files to ~/alfa
```

To send email:

```
sudo apt-get install postfix

sudo service apache2 restart
```

2.3 LabelMe-Text

2.3.1 Install Process - Installing Ruby on Rails

First step is to install ruby on your linux environment. The commands used here come from https://gorails.com/setup/ubuntu/14.10

In terminal, run these two commands:

```
sudo apt-get update
sudo apt-get install git-core curl zlib1g-dev build-essential libssl-dev libreadline-dev libyaml-dev
```

For this project I ran Ruby using RVM, so then run

```
sudo apt-get install libgdbm-dev libncurses5-dev automake libtool bison libffi-dev
curl -L https://get.rvm.io | bash -s stable
source ~/.rvm/scripts/rvm
echo "source ~/.rvm/scripts/rvm" >> ~/.bashrc
rvm install 2.1.5
rvm use 2.1.5 --default
ruby -v
echo "gem: --no-ri --no-rdoc" > ~/.gemrc
```

Install node.js

```
sudo add-apt-repository ppa:chris-lea/node.js
sudo apt-get update
sudo apt-get install nodejs
```

Install Rails

```
gem install rails
```

Install depedencies for pg gem **not in online instructions to install rails

```
sudo apt-get install ruby-dev libpq-dev
```

Server Setup

Time to set up the server. Move to the LabelMe-Text folder and run the following command:

```
bundle install
bundle exec rake db:migrate
bundle exec rake db:reset
bundle exec rails s
```

This starts a copy of the server running on the machine's localhost. You can access it in your browser at:

`localhost:3000/`.

2.3.2 Making an Admin

- Now you want to make an admin user. Go to the website and sign up with the account info for your admin user.
- From the home page, click on "Create profile"
- Fill in the account details, check the box agreeing not to scrape the data from the website, and click "Create my account"
- Then go to terminal and type `Ctrl+C` to stop the server from running. Run the following commands:

```
bundle exec rails console.
a = User.find_by_email("your@email.here")
a.update_column(:admin, true)
exit
```

2.3.3 Leaving server running

If you are using a virtual machine and wish the process to keep running after you close the 'ssh' connection, we will use a program called **screen** to achieve this.

1. First we need to install screen.

   ```
   sudo apt-get install screen
   ```

2. Start the screen program

   ```
   screen
   ```

3. Navigate to the Label-Me Text folder and start the server again

   ```
   bundle exec rails s
   ```

4. While the server is running, detach from the current screen using the following command

   ```
   Ctrl + a, d
   ```

5. The server should now run even if you stop the ssh session. To return to the running server terminal, use the command:

   ```
   screen -r
   ```

MOOCviz

A MOOCviz platform offers:

- A central, shared gallery of visualizations with a demonstration list of courses for which the participant-generated visualizations have been rendered.
- The ability for the participants to download the software that generates visualizations and execute it over their own course data that is formatted in MOOCdb schema. They will also be able to automatically package the resulting rendered visualization and upload it to the gallery, adding it to the demonstration list.
- A means to contribute software for new visualizations to the gallery via the MOOCviz web-based interface.
- A means of commenting on any existing visualization by posting in the comments section underneath it. Discussions are free form. They likely will extend beyond the interpretation or thoughts provoked by the visualization to the ways that the data have been transformed in extraction and aggregate steps. We expect that discussions will stimulate ideas for new visualizations.

System Requirements

Installation steps

Testing

Feature Factory

We are developing a second web-based collaborative platform called Feature Factory. Our current version of this platform can be seen at .._Feature Factory : http://featurefactory.csail.mit.edu

Feature Factory offers two modes of engagement:

- The "solicit" mode is used by MOOC data science, education technology, or learning science research teams. A team describes the outcome it is currently studying or trying to predict. It explicitly explains what features or explanations are sought and it solicits help from the "MOOC crowd".
- In the second mode, "helping", the MOOC crowd proposes, explanations or variables, and suggests means to operationalize them. They provide comments on proposal or vote them up or down in popularity. The software savvy among them write and share software scripts written to operationalize the most popular or compelling proposals.

System Requirements

Installation steps

Testing

LabelMe-Text

We developed an online platform where users would post their tagging projects and a crowd of helpers can p articipate in MOOC data science by selecting a project and tagging the content based on some instructions. We call the online collaborative platform that serves this purpose Label Me-Text. Label Me's current incarnation is shown in Figure 3. It works in the following ways:

- Users requiring annotation of natural language can create an annotation project by providing a csv file for the content, instructions and examples for tagging.
- Taggers (Label Me's crowd) can participate by selecting a project, following the instructions and tagging the content
- A database consisting of tags for the content for the project is initialized and populated as taggers work. A number of analytic services are provided around this database such as – evaluation of inter rater reliability, summary of tags, summary of activity for a project (how many taggers helped, time series of number of tags).
- A service can be called upon to filter the tagged data based on the reliability measures just mentioned. It then uses methods based upon latent semantic analysis to learn a tagging model.
- Taggers (Label Me's crowd) are given credit for every tag they have provided and the number of their tags that pass the filters to be used in model learning.

System Requirements

Installation steps

Testing

Curate Me

Coming soon

CHAPTER 3

Analytic

Feature Engineering

Data from edX platform is primarily stored in JSON logs. A fundamental axis which is used to record precisely the activity performed by the learner is an “event type”. Multiple “event types” differentiate between different activities done by the learner. We base our software on this fundamental axis. Below we provide detailed description of how each event type is translated into an entry in MOOCdb. This detailed information gives researchers and plaform providers information about MOOCdb translation and how data is mapped syntactically and semantically.

3.1 Feature Engineering

3.1.1 Coming Soon

CHAPTER 4

Machine learning apps

Predict dropout

Data from edX platform is primarily stored in JSON logs. A fundamental axis which is used to record precisely the activity performed by the learner is an "event type". Multiple "event types" differentiate between different activities done by the learner. We base our software on this fundamental axis. Below we provide detailed description of how each event type is translated into an entry in MOOCdb. This detailed information gives researchers and plaform providers information about MOOCdb translation and how data is mapped syntactically and semantically.

Topic Models

Problem analytics

4.1 Dropout prediction

Coming Soon

4.2 Topic Models

Coming Soon

4.3 Community detection

Coming Soon

Contact

Founded at CSAIL, MIT. Please contact kalyan@csail.mit.edu for more information.