**How to run the NLP Engine**

Step 1. Log in the Azure VM

1. Please open your terminal and enter: **ssh ucl**[**team43@**](mailto:team43@20.77.17.178)51.142.241.173
2. Please enter the password: UCLmiemie2021

Step 2. Run the NLP Engine

Please enter: cd comp0016miemie\_vm

Please enter: cd src

Then run the lines below

**Scraping Modules:**

First, we can initialise departmental data, which is necessary to perform prior to scraping. It can be done by running the command below:

**python3 global\_controller.py MOD initialise**

Furthermore, it is vital to reset the current module data to ensure the end result only reflects the current teaching activity at UCL. This is done by running the following command:

**python3 global\_controller.py MOD resetDB**

Lastly, to reflect the current student population data, keep the file ”studentsPerModule.csv” up-to-date in directory

src/main/MODULE\_CATALOGUE/STUDENTS\_PER\_MOD. To synchronise that data with the database run the following command:

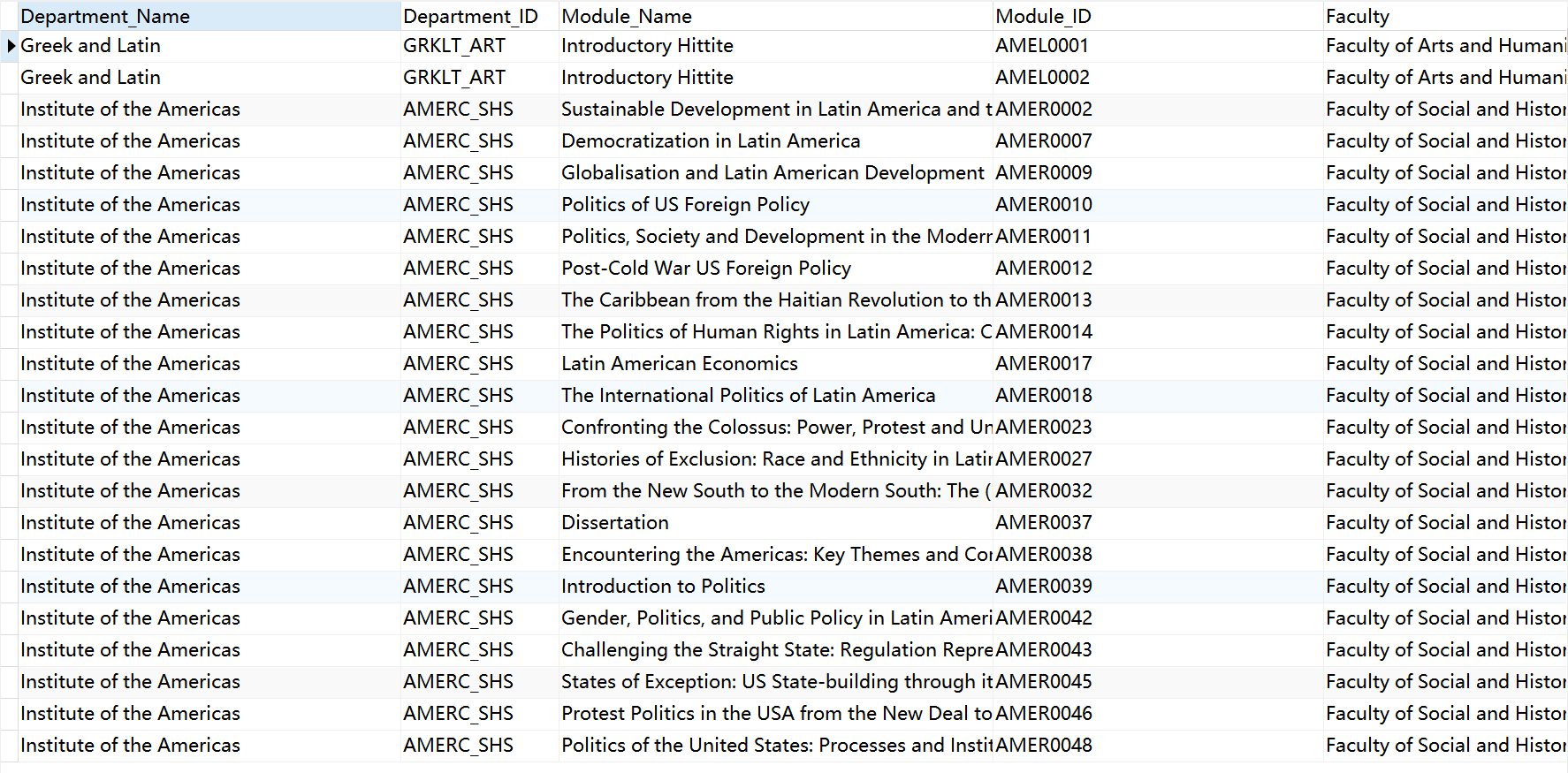
**python3 global\_controller.py MOD updateStudentCount**

Finally, module scraping can be performed. Ensure the MySQL credentials are valid and up-to-date in config.ini file (under SQL\_SERVER section) and run the following command to freshly scrape the UCL module catalogue data:

**python3 global\_controller.py MOD scrape**

Finally, module scraping can be performed. Ensure the MySQL credentials are valid and up-to-date in config.ini file (under SQL\_SERVER section) and run the fol

Once finished, the data is automatically synchronised with the database, and can be viewed in the ModuleData table through any database management system by inputting login credentials found in config.ini SQL\_SERVER section. It should have the following appeal:

****

**Scraping Publications**

Prior to scraping, firstly ensure the file titled “cleaned\_RPS\_export\_2015.csv” in directory src/main/SCOPUS/GIVEN\_DATA\_FILES is up-to-date. The file should contain a column titled “DOI”.

The scraper examines given DOIs, compares them to existing records and scrapes only those not already present in the database. It is vital for the file to retain its structural integrity to avoid any unexpected script errors. Secondly, follow instructions below to setup Scopus API key and initiate scraping procedure:

1.Go to: <https://dev.elsevier.com/documentation/AbstractRetrievalAPI.wadl>

2.Locate My API Key section on the top right

3. Sign in with your institution (needs to verify affiliation identifier)

4. Create API key (tick agreement boxes and submit)

5. \*Run the following command: **python3 global\_controller.py SCRAPE\_PUB**

6. When prompted to enter your API key, copy it from the website and enter it

7. Click enter once more to skip the Authtoken authentication prompt

8. This will create the scopus config.ini file so that this process doesn’t have to be

repeated for future scrapes

\*Once the Scopus API key has been set up, ensure that you are on a UCL network

(either using UCL WI-FI or connected to a UCL virtual machine). It can also be

achieved via UCL VPN (instructions).

**Automatic operation**

To run the NLP engine automatically, we used crontab in our virtual machine. The

engine will be run automatically every 1st of January.

View current crontab automatic command:

**crontab -l**

View and change the crontab command:

**crontab -e**

**Using the Engine**

To run the tool, examine the list of available commands, each dedicated to a specific action. Note running a command impacts files, as well as certain database contents (possibility of overwriting existing values). Chronologically coherent sequence of commands is outlined below. Loading the publication and module data:

**python3 global\_controller.py LOAD publications**

**python3 global\_controller.py LOAD modules**

Train the LDA for SDGs and/or IHEs and/or HAs

**python3 global\_controller.py NLP run\_LDA\_SDG**

**python3 global\_controller.py NLP run\_LDA\_IHE**

**python3 global\_controller.py NLP run\_LDA\_HA**

Perform string-matching for SDGs or HAs (skip if only focusing on IHEs)

**python3 global\_controller.py NLP module\_string\_match**

**python3 global\_controller.py NLP ha\_string\_match**

**python3 global\_controller.py NLP scopus\_string\_match\_SDG**

**python3 global\_controller.py NLP scopus\_string\_match\_IHE**

**python3 global\_controller.py NLP scopus\_string\_match\_HA**

Use SDG and HA LDA results to classify publications (skip if only focusing on IHEs)

**python3 global\_controller.py NLP predict\_scopus\_data**

Prepare pickled dataset for SVM training (for SDGs and/or IHEs and/or HAs)

**python3 global\_controller.py NLP create\_SDG\_SVM\_dataset**

**python3 global\_controller.py NLP create\_IHE\_SVM\_dataset**

**python3 global\_controller.py NLP create\_HA\_SVM\_dataset**

Train the SVM for SDGs and/or IHEs and/or HAs

**python3 global\_controller.py NLP run\_SVM\_SDG**

**python3 global\_controller.py NLP run\_SVM\_IHE**

**python3 global\_controller.py NLP run\_SVM\_HA**

Validate SVM SDG/HA results against string-match ((skip if only focusing on IHEs)

**python3 global\_controller.py NLP validate\_sdg\_svm**

**python3 global\_controller.py NLP validate\_ha\_svm**

Once publication and module data are scraped, synchronise MongoDB with PostgreSQL

**python3 global\_controller.py SYNC synchronize\_raw\_mongodb**

After LDA & SVM training, synchronise results with PostgreSQL

**python3 global\_controller.py SYNC synchronize\_mongodb**

After LDA & SVM training, use publication classification to update user data + bubble chart data

**python3 global\_controller.py SYNC synchronize\_bubble**