

**ENotebook Website**

Firewall

Firewall

**U Wisconsin Servers**

ENotebook

Server #1

Server #2

ENotebook side of things is illustrated above, and described below. It has three parts: gunicorn (see the gunicorn command in the github README.md for the Pyreval-mongo branch; also below), and the scripts, pyreval\_service.py and pyreval\_wsgi.py.

The gunicorn command runs pyreval as a service; # workers etc can be changed.

gunicorn --workers 4 --timeout 1200 --log-level=DEBUG --bind 0.0.0.0:5000 pyreval\_wsgi:app

pyreval\_service.py defines a function pyreval\_request to invoke pyreval as a service; instead of using a function call, a service will be an exterior shell for pyreval that waits for pyreval requests to process. A function would return a value and not keep waiting for new requests.

pyreval\_wsgi.py invokes the pyreval\_request function in a few lines of code; it is recommended to separate the inner workings of a given service from the flask app trigger in this way.

The pyreval\_wsgi:app needs an “address” (url) to bind to, which is an “internal” url address where the service will run, given the hardware setup. This can be changed if needed.

The timeout parameter is for moving on to the next request coming from the notebook in case PyrEval gets stuck on a particular essay and the worker remains tied up. Can be reduced.

The log-level=DEBUG provides very detailed tracing, but should not be necessary when it is determined that things are working well

### INSTRUCTIONS TO INSTALL [PYREVAL-MONGO](https://github.com/psunlpgroup/PyrEvalv2/tree/PyrEval-mongo)

**0** These instructions do not explain how to install and start a mongodb. This is assumed. It is also assumed that Compass is installed, and that you are familiar with its use.

**1** Download [PyrEvalv2-mongo](https://github.com/psunlpgroup/PyrEvalv2/tree/PyrEval-mongo) from the github. The github includes the current pyramid to be used with the first essay, and a script to add this pyramid to the db, along with an older pyramid we are not using. MAKE SURE YOU USE essay1\_pyramid\_readable\_20221207.pyr; this will be automatic if you update the parameters.ini file as indicated next in step 2.

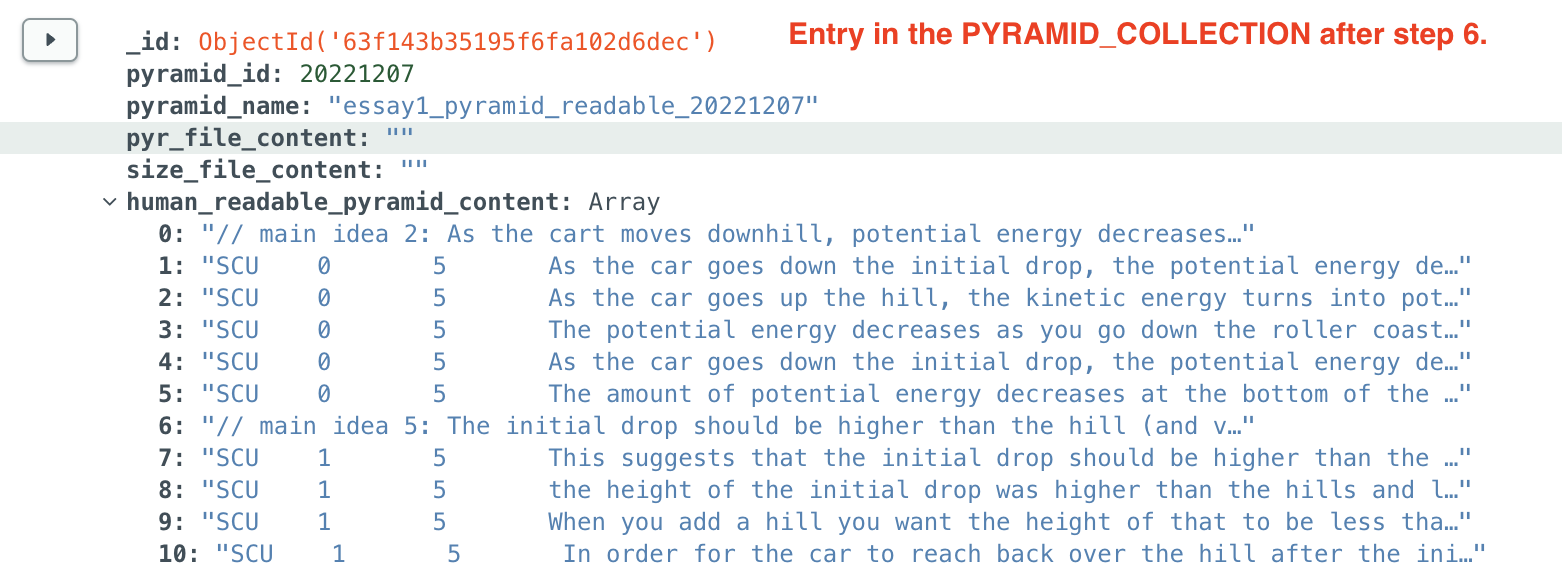
**2** Update the value of the basedir parameter in the parameters.ini file with the local path to your installation directory, the databasename parameter and the dbconnection parameter. DO NOT EDIT AN OLDER parameters.ini file; parameter values for scoring and for segmentation have changed

**3** Ensure your environment meets the six items under “Installation Requirement” in README.md.

**4** Start your mongod process for a mongo database, or ensure that you have it running already.

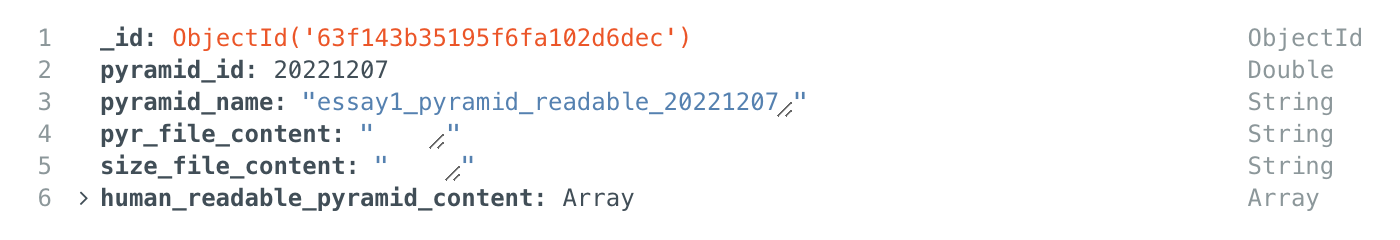
**5** Start mongo shell, and start Compass and connect to your local db. You will use them to confirm the success of the database initialization, and correct installation of PyrEval-mongo.

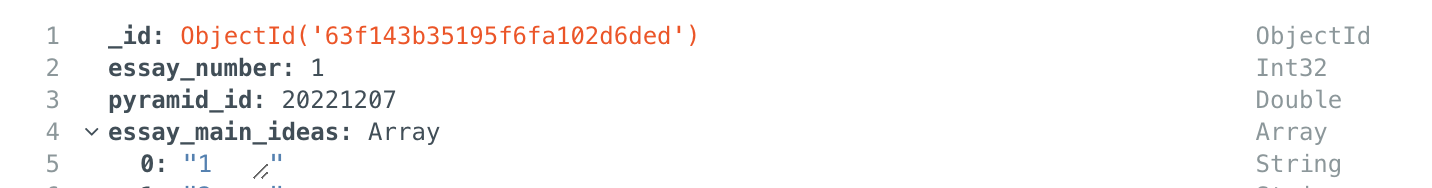
**6** Execute MongoDB/populate\_pyramid\_table\_mongo\_new.py –pyramidNumber <N> twice, for each pyramid, using ‘1’ for the essay 1 pyramid and ‘2’ for the essay 2 pyramid. It will place the human readable pyramid files that are in MongoDB/HRP\_Files into the db in the appropriate collection (per the parameters.ini file), it will produce the xml version used by PyrEval and the size file, and it will also create the ESSAY\_PYRAMID collection. After running this script, you should check Compass for two new collections: PYRAMID\_COLLECTION and ESSAY\_PYRAMID\_COLLECTION each of which should be populated. You will see that two of the attributes in the two PYRAMID\_COLLECTION documents, “pyr\_file\_content” and “size\_file\_content”, are empty. These will be populated later.



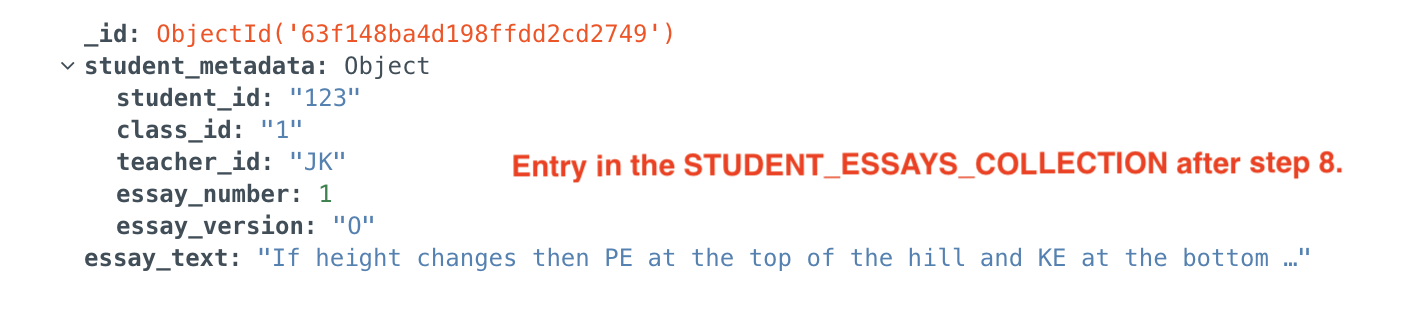


**7** In Compass, **the datatype of the pyramid\_id attribute should be double in PYRAMID\_COLLECTION and in ESSAY\_PYRAMID\_COLLECTION.** If it isn’t already, mouse over the object and select the edit icon that appears. It will look like this when it is correct in both collections. THIS IS CRUCIAL.





**8.** In Compass or in the mongo shell, insert a new collection into your database called STUDENT\_ESSAYS\_COLLECTION.  Read in the gold standard json file from the testsuite folder into this collection, for the relevant essay. For essay assignment 1, the test suite will have 40 entries: 39 student essay documents that are original submissions, and 1 revision. For essay assignment 2, the test suite will have 39 entries for the second original submission to essay 2 from the same 39 students. Each document in this new collection will look like the screenshot below from Compass, but with different values.



**9.** Modify line 287 of pyreval\_mongo\_launcher.py to process the essays added in step 8, e.g., for example, for the essay T1\_JN1\_ROW17 for essay 1 from the database you would use:

student\_metadata("T1\_JN1\_ROW17", "1", "JN", 1, "O")

and for the essay 2 version from the same student you would use:

student\_metadata("T1\_JN1\_ROW17", "1", "JN", 2, "O")

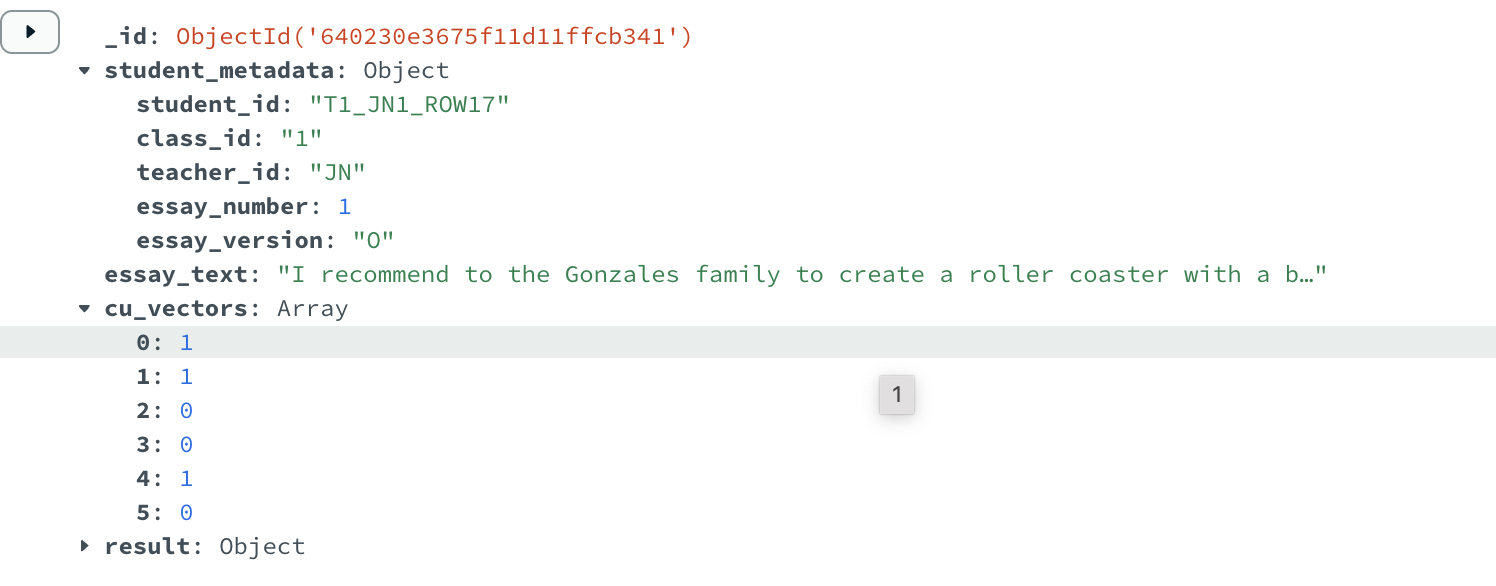
The five arguments here are the student\_id, class\_id, teacher\_id, essay\_number, and essay\_version. Execute the interactive script pyreval\_mongo\_launcher.py, entering 1 the first time you see >>>, then 2, then 3, then 4. Although you will see some warning messages, you should ignore them. After step 4, enter q to exit the launcher.

**10** The PYRAMID\_COLLECTION entry will now have values for the two attributes that remained empty after step 6: pyr\_file\_content, size\_file\_content. Also, the entry for the student in STUDENT\_ESSAYS\_COLLECTION will now have a cu\_vectors attribute, and a result attribute.

Go to Compass to verify that the pyramid entry now looks like this, with values for the pyr\_file\_content, and size\_file\_content attributes:



Verify that the student document for T1\_JN1\_ROW17 now looks like this:



The csv file from the testsuite folder shows the correct values of the cu\_vectors field for each of the 39 gold standard essays (“O”), and a revision for one of them (“R”).

**11** The db you created should also now have a DEBUG\_DATA\_COLLECTION. This will store timestamped debugging information, with a new entry every time you process or reprocess an essay. The stage logs are the most interpretable: there should be a value for every attribute in stage logs. If an essay does not get fully processed, the stage logs will reflect which processing steps did or did not complete.

### DB NOTES

**1** If a student submits a revision through the Notebook, it needs to go into the database with essay\_version: "R".