

README

Author: Niall Guerin

Class: CSD1

This document is a README file providing instructions for executing the hybrid workflow documented in thesis project from staging through to ML prediction step.

Resources used in the code files are explicitly commented in a 1,2,n manner or links to specific resources are listed preceding relevant code block per file.

Installation Instructions

- Install MySQL (if possible); otherwise Postgres or equivalent RDBMS should be fine
- Install PyCharm or run from Google Colab or Jupyter Notebook
- Tested against Python 3.7+
- Packages Required: gensim, scikit-learn, TensorFlow (ensure you have the 64-bit Python executable installed if running on Windows 64-bit as TensorFlow expects it), numpy, pandas, nltk, re, textstat, keras, matplotlib

MySQL

IF you want to run the entire workflow from scratch with MySQL, use the mysql directory for all relevant scripts for importing the data from the public Internet archive download, exporting the data, and custom utility scripts I added. The import/export commands are based on the sql scripts provided by Fabio Calefato at this [GitHub location](#). As the datasets are huge, be careful that the execution timeouts are disabled; the import can take up to between 26 and 28 hours on a Windows server with 2.5TB storage and 16GB of RAM. Just let it run and it will load the records fine. You can use MySQL console or use MySQL Admin (if using the MySQL Admin GUI again set your preferences to avoid SQL script timeouts hitting default values).

If you have any issue loading the dataset files into MySQL, refer to this thread and check the required parameter:

```
SHOW VARIABLES LIKE "secure_file_priv";
```

This is the directory you specify when uploading the XML files from the Stack Overflow dataset download.

For loading the XML dataset files, create the staging table schema, indices, and import per instructions on this page¹. In case of any issues, as I made minor modifications on my own system, I have a copy of the files that worked for me stored in the msqf folder and shortcut to this GitHub resource in case of issues and I named them Calefato_* to acknowledge the source for those staging scripts. I have added some additional custom MySQL tables for metrics tests for later in the end-to-end query experiment tests that are unrelated to the Calefato_* staging scripts and they have standard filenames.

Watch for disc space exhaustion errors on your MySQL *.tmp log locations when loading these large files. I moved everything off C to D which had over 1TB of space and that got rid of the runtime loading errors and disc space low errors. An example of the type of file to watch for is shown below:

¹ https://github.com/collab-uniba/emse_best-answer-prediction/tree/master/dumps/stackoverflow

| OS (C:) > Windows > ServiceProfiles > NetworkService > AppData > Local > Temp | | | | |
|---|--------------------|---------------|------------|--|
| Name | Date modified | Type | Size | |
| ML9E64.tmp | 6/24/2019 11:22 AM | TMP File | 0 KB | |
| ib7543.tmp | 6/24/2019 11:18 AM | TMP File | 0 KB | |
| ib73E7.tmp | 6/24/2019 11:17 AM | TMP File | 0 KB | |
| ib73E8.tmp | 6/24/2019 11:17 AM | TMP File | 0 KB | |
| ib73E9.tmp | 6/24/2019 11:17 AM | TMP File | 0 KB | |
| ib740A.tmp | 6/24/2019 11:17 AM | TMP File | 0 KB | |
| ML5D9B.tmp | 6/17/2019 5:29 PM | TMP File | 701,256 KB | |
| ib83E4.tmp | 6/17/2019 4:26 PM | TMP File | 0 KB | |
| ib83E5.tmp | 6/17/2019 4:26 PM | TMP File | 0 KB | |
| ib83E6.tmp | 6/17/2019 4:26 PM | TMP File | 0 KB | |
| ib83E7.tmp | 6/17/2019 4:26 PM | TMP File | 0 KB | |
| ib8475.tmp | 6/17/2019 4:26 PM | TMP File | 0 KB | |
| MpCmdRun.log | 6/10/2019 6:40 PM | Text Document | 446 KB | |
| MpSigStub.log | 11/9/2018 12:18 PM | Text Document | 17 KB | |
| MPTelemetrySubmit | 10/28/2018 3:19 PM | File folder | | |

Web Reference

<https://stackoverflow.com/questions/32737478/how-should-i-tackle-secure-file-priv-in-mysql>

CSV Mass Export CSV file

The SQL template provided by Calefato will generate a huge file; based on the Stack Overflow dataset from April 2019, about 67GB in my case. Be very careful when monitoring the export script job runs. Initially, MySQL Admin gave an error pop-up message but the log console was Green. However, the export only produced a partial file of 1.32GB with about 730K rows. During later IR system component test cases, records I expected to be in the corpus were absent and this is when the issue was detected. A re-run of the export was performed from staging with all timeouts removed and the full 67GB file was generated.

On OSX, install gsplit if you want extra parameters to control the filename output when splitting the file versus split command. On Windows, use the GNU Core Utils package. This version worked fine on Windows 10:

Web Reference for GNU Core Utils Package

<http://gnuwin32.sourceforge.net/packages/coreutils.htm>

sample syntax for file split command on Windows or OSX

```
gsplit -l 100000 -d --additional-suffix=.csv
all_answers_mass_export.csv all_answers_
```

The above command will output files in format like below to your directory, with 100K records per file, from which you run it.

| (D:) > ProgramData > MySQL > MySQL Server 8.0 > Uploads > corpus | | | | |
|--|-------------------|---|------------|--|
| Name | Date modified | Type | Size | |
| all_answers00.csv | 8/16/2019 4:41 PM | Microsoft Excel Comma Separated Values File | 166,473 KB | |
| all_answers01.csv | 8/16/2019 4:41 PM | Microsoft Excel Comma Separated Values File | 166,774 KB | |
| all_answers02.csv | 8/16/2019 4:41 PM | Microsoft Excel Comma Separated Values File | 174,810 KB | |
| all_answers03.csv | 8/16/2019 4:41 PM | Microsoft Excel Comma Separated Values File | 180,768 KB | |
| all_answers04.csv | 8/16/2019 4:41 PM | Microsoft Excel Comma Separated Values File | 180,873 KB | |

Ensure you have sufficient disc space as this will mean you still have the original 67GB file plus another 67GB of files split according to the number of rows highlighted above per chunked file split.

The **metrics** SQL scripts are not critical; I set them up as helper files to allow me run some level of comparison analysis on the overall corpus collection depending on the active .csv file. The tables are created per .csv being tested in the main query list of experiments, so all_answers00.csv = Queries4 to Query4 in terms of database content and all_answers_9056.csv covers Query 5.

The script just creates helper tables to load the corresponding .csv file and run simple matching queries to the experiment query list strings and check the results using NLP mode and Boolean mode in MySQL native functions.

Feature Engineering Scripts

Precondition

A .csv file exists in the **/data** directory below these files wherever you run them. The file should be one of the split .csv files from the original staging export so the format is as expected for the Python script.

There is no order to the main feature engineering scripts though I had a habit purely due to order of development or running vocabulary or linguistic feature conversions first, and then running the **feat_post_converter_merge.py** LAST. That one MUST ALWAYS RUN LAST as it merges the dataframe .csv files from the prior 3 scripts into a single .csv file that maps to the input format needed by the ML model scripts.

The same rules apply for dynamic feature conversion inside the IR system component directory files. These are really just a clone of these files, so they can be ported to a single library utility file and just imported as the functions are similar. I have not had time to clean up the packages to ensure better code re-use at time of writing.

The feat_post_converter_merge.py will create files with the following file-naming format:

- Modify the last line if you want custom or alternative filenames
- ```
df_output.to_csv("data/so_qs_all_answers01.csv", encoding='utf-8', index=False)
```

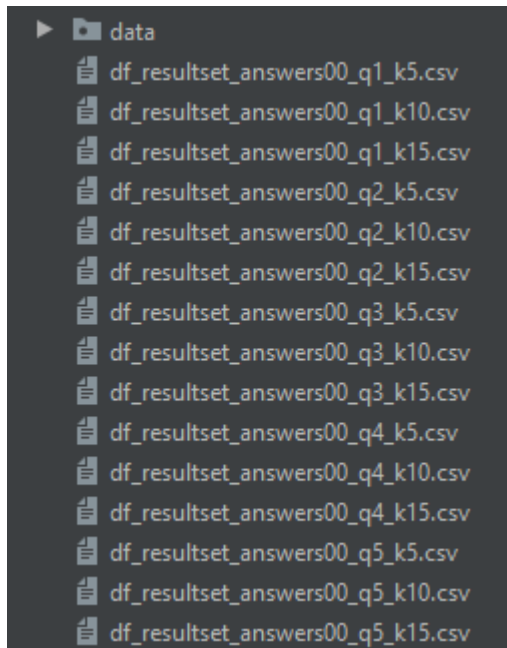
The ML model scripts can be run independently of the IR system component using the chunked all\_answers\*.csv files as input for training and as hold-out sets.

## ML Model Scripts

For both the Neural Network Keras model and the scikit-learn XGBoost model, there is a preprocessor script that does some minor tweaks to the above file output from the feature engineering merge script. Once you run that preprocessor script in either model, you don't need to run it again in the second model you are training or testing. Just drop the preprocessor output df\*

file into the target second model and it's good to go. That preprocessor can possibly be bypassed in future as it does trivial enough prep and it could all just be bundled into the feature engineering dataframe merge script anyway as an optimisation for the code overall.

I have made a list of filename constants (and this could be repeated for the feature engineering scripts so right now that is only part where there is no log of the previous file in case you need it, so keep in mind if running experiments at that step), so that the workflow can be run from that point in time too for query1, query2, ...queryN for precision@K=5, K=10, K=15.



This lets you further tune or reconfigure the ML model without restarting the pipeline from scratch for a given input query test case.

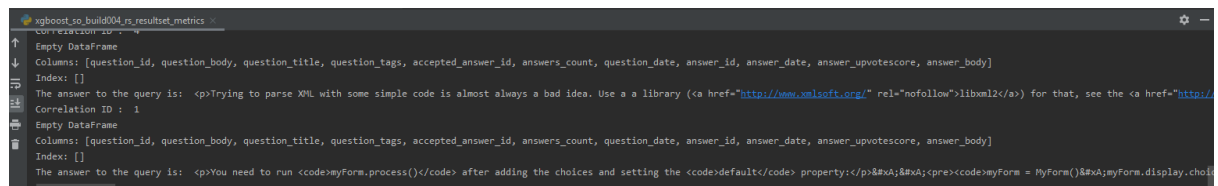
Neural Network using Keras Files:

|                                                                                                                                                                                                                              |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. ml_model_generic_ir_system_component_resultset_preprocessor.py<br>This is the preprocessing script to take the IR result set .csv and ready it for the ML model training, testing, or prediction on held-out corpus file. |
| 2. neural_network_keras_train_fit_save_model.py (main file for training, saving model)<br>nn_model_normalized_1.h5 (NN model with Batch Normalized layer added)                                                              |
| 3. neural_network_keras_resultset_processor_predictor.py<br>This loads the previously trained model and makes predictions on the query input resultsets.                                                                     |

XGBoost Files:

|                                                                                                                                                                                                                              |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. ml_model_generic_ir_system_component_resultset_preprocessor.py<br>This is the preprocessing script to take the IR result set .csv and ready it for the ML model training, testing, or prediction on held-out corpus file. |
| 2. xgboost_train_fit_save_model<br>xgboost_so_model.sav                                                                                                                                                                      |
| 3. xgboost_so_resultset_processor_predictor<br>This loads the previously trained model and makes predictions on the query input resultsets.                                                                                  |

## Snippet Extractor sample in Console from XGB model



```
xgboost_so_build004_rs_resultset_metrics
Correlation ID : 1
Empty DataFrame
Columns: [question_id, question_body, question_title, question_tags, accepted_answer_id, answers_count, question_date, answer_id, answer_date, answer_upvotescore, answer_body]
Index: []
The answer to the query is: <p>Trying to parse XML with some simple code is almost always a bad idea. Use a library (libxml2) for that, see the <a href="http://
Correlation ID : 1
Empty DataFrame
Columns: [question_id, question_body, question_title, question_tags, accepted_answer_id, answers_count, question_date, answer_id, answer_date, answer_upvotescore, answer_body]
Index: []
The answer to the query is: <p>You need to run <code>myForm.process()</code> after adding the choices and setting the <code>default</code> property:</p>

<pre><code>myForm = myForm()
myForm.display.chof
```