Error handling

Contents

Overview

This is rudimentary and would need to be improved for any production service.

Currently any detected error condition causes the script to end with an error message, using PHP's **trigger_error()** function.

e.g. in **enhanceCitationsFromScopus.php::scopusApiQuery()** we have:

```
if (!$scopusResponse) {
   trigger_error("Error: No response from Scopus API for [".$URL."]", E_USER_ERROR);
}
```

There is no error log generated by the scripts.

Potential causes of error

- Temporary unavailability of or fault with API
    - *e.g. the Scopus API has occasionally been unavailable, with the error "GENERAL_SYSTEM_ERROR (Error calling Solr Search Service)"*
- Too many requests for API quota
    - *The code logs the number of remaining allowed Scopus and WoS requests in the JSON citations file, so you can go back and find the latest good request, and see whether the remaining request count is indeed approaching zero*
- Missing config.ini file or incorrect API key in config.ini
    - *A config.ini file must be created from config.template.ini and populated with API keys before running the scripts*
    - *The code will generate a relevant error message where an API key is missing or invalid*
- Unanticipated issue with data from API
    - *e.g. the code may assume a particular property will always be present in the data returned by the API, or will always be of a certain data type or structure: when doing a live run across a large number of citations, it is possible a counter-example will be hit*

- Code errors
    - *The data and the code are complex - it is possible some code has not been tested adequately*
- Out-of-memory
    - *Despite breaking up the process by module and step, it is possible that a very large reading list might consume available memory: if this problem is encountered, a line like the following could be added to the top of the affected scripts:*

      ```
      ini_set('memory_limit', '512M');
      ```

Running individual scripts

e.g. if running:

```
php enhanceCitationsFromScopus.php <Data/temp/MODULE_A.json >Data/temp/MODULE_S.json
```

If there is an error then as well as being reported to the console, it will be written to the file MODULE_S.json, meaning that file will not contain valid JSON.

If you miss the error on the console and straight away run the next step:

```
php enhanceCitationsFromWos.php <Data/temp/MODULE_S.json >Data/temp/MODULE_W.json
```

That script in turn will end with an error, because it will not be able to collect valid JSON data from STDIN.

This is a defect of the design of the process - taking I/O from STDIN/STDOUT means errors from one step can pollute the input to the next.

Running the batch script

The batch script offers some protection against this problem by checking for errors after each step, and only proceeding to the next step if there were none. Additionally, the batch script has the individual steps write output to temporary files in Data/tmp/. It only copies them up to the main data file in Data/ if the step was successful.

If any individual script called by the batch script ends with an error, the whole batch script ends, so it is not possible to produce reports if the individual steps have not succeeded.

Suggestions for debugging

When debugging, it is probably easiest to run individual scripts e.g. in case of a problem with Scopus integration it is probably easier to run:

```
php enhanceCitationsFromScopus.php <Data/MODULE.json
```

(assuming you have some good citation data in the file Data/MODULE.json) rather than:

```
php batch.php -m MODULE_A -s scopus
```

If you are running individual scripts rather than the batch script, then possible approaches to debugging include:

- Preparing a custom input JSON file with a single citation which demonstrates the problem, rather than waiting while a lot of irrelevant citations are checked
- Adding code to a script that checks for a particular case (e.g. a missing array value), prints the value of a variable, and exits e.g.:

  ```
  if (!isset($scopusSearchData["search-results"]["entry"])) {
      print "Unexpected: no search-results.entry in Scopus search data\n";
      print_r($scopusSearchData);
      exit;
  }
  ```

- Adding a line to the code that interacts with the APIs to print the raw headers and response from the API e.g. add the lines in bold to the file **enhanceCitationsFromScopus.php** circa line 627:

  ```
  $scopusResponse = curl_get_file_contents($apiURL);
  print "Scopus response:\n";
  print "Headers:\n";
  print_r($http_response_header);
  print "Content:\n";
  print "$scopusResponse\n";
  ```

  $http_response_header is written as a by-product by curl_get_file_contents()
  Depending on the situation, you may want to exit after these lines or carry on and print them for each call to the API

- Directing output from the script to a text file that you can follow and inspect e.g.

  ```
  php enhanceCitationsFromScopus.php <Data/MODULE.json >Data/tmp/output.txt
  ```

Possible improvements

This crude approach means that a (possibly minor) issue with a single citation prevents the processing of a whole batch of thousands of citations: it might be preferable to accept occasional mild errors (and have an idea of how many, and where they are) when doing a large-scale process.

It would be better to modify the scripts so that they:

- Take their I/O from specific files and not from STDIN/STDOUT
- Log errors, warnings and information to a file as well as the console, and continue running unless the error is so severe that it is impossible to proceed

The reporting scripts could summarise the errors so that IT and Library staff can decide whether to accept these results or modify the process and retry.