## HDX Notifications: Discovery to detect changes for tabular data files

<https://humanitarian.atlassian.net/browse/HDX-10021>

Ian Hopkinson 2024-07-25

### Requirements

This is part of the HDX Notifications work. The requirements as specified in the JIRA ticket (https://humanitarian.atlassian.net/browse/HDX-10021) are as follows:

Having 2 versions of the same tabular file, we need a JSON output describing the changes:

* header changes
* column changes
* cell content changes

The first 2 are already integrated via hxl proxy (fs-check-info column). Therefore cell content changes are the primary focus of this work.

### Background

Provided in the JIRA ticket is this example dataset.

<https://data.humdata.org/api/action/package_show?id=wfp-food-prices-for-afghanistan>

There are two resources in this dataset, one for QuickCharts which is 881 rows of three columns and a full file of 14 columns and 32565 rows. The data appears to be updated weekly and as of 24th July versions back to 12th May were available in S3.

The fs\_check\_info key holds some change measures, we can access it directly by navigating to this link:

<https://data.humdata.org/api/action/fs_check_info_show?id=wfp-food-prices-for-afghanistan>

A typical entry in the fs\_check\_info looks like this:

A screenshot of a computer

Description automatically generated

New entries are created each time a resource is updated and by comparing the information in entries some types of change can be identified (numbers and names of columns and HXL tags and numbers of rows). Sheet\_changes entries in fs\_check\_info look like this:

A computer screen shot of a program code

Description automatically generated

HXL-proxy does not access multiple versions of a resource.

The fs\_check\_info is computed by <https://github.com/OCHA-DAP/gisrestlayer> which uses the hxl-proxy (https://github.com/HXLStandard/hxl-proxy) to do the checks and then puts them into CKAN https://github.com/OCHA-DAP/gisrestlayer/blob/dev/filestructurecheckapi/tasks/file\_structure\_check.py#L71.

Hdx-python-utilities (https://github.com/OCHA-DAP/hdx-python-utilities) has a compare files function which is based on difflib:

<https://github.com/OCHA-DAP/hdx-python-utilities/blob/main/src/hdx/utilities/compare.py>

Difflib, a Python built-in is here:

[https://docs.python.org/3/library/difflib.html#](https://docs.python.org/3/library/difflib.html)

This provides output of all the rows in two input lists with the following markup:

A screenshot of a computer

Description automatically generated

Thus it should be straightforward to identify lines added, lines removed and lines modifies. The lines modified along with the indicators on “?” lines can be used to identify cell level changes in tabular data.

### Taxonomy of changes

Based on the Insecurity Insights datasets which are time series data we see these types of changes to resources:

1. Rows added at the beginning of the dataset

2. Rows added at the end of the dataset

3. In Excel, change in cell types

4. latitude, longitude and Geo Precision censored for PSE

5. Date column format changed

6. Revisions to rows – we don’t know if this has happened.

The Afghanistan – Food Prices dataset shows the addition of new rows of data, as well as spot changes to prices on a limited number of rows.

In production the number of files, the size of the files and the amount of change will have an impact on the time taken to process changes.

### Implementation

A test implementation of a cell level comparison tool was written using the Python built-in library difflib. This reflects a preference to use existing tools, and favour those from Python rather than third parties. The implementation can be found here: <https://github.com/OCHA-DAP/hdx-file-comparison> The tests demonstrate its use, currently there is no other interface.

The hdx-file-comparison library takes the output from difflib.ndiff (shown below) and processes it to count lines added and removed as well as determining which cell in a row has changed. Difflib does not provide this information explicitly but rather inserts a line below the changed line which starts with a “?” and contains ^ characters where characters have changed. A change to a row is therefore indicated with four lines in the difflib output: a deleted line followed by a “change” indicator line followed by an added line followed by a second “change” line. The work required in the library is in associating the change markers to a particular column.

A screen shot of a computer

Description automatically generated

The output from the hdx-file-comparison library looks like this, although this can be changed easily.

A computer screen with text and numbers

Description automatically generated

Initially tests were carried out on the resources in the Afghanistan – Food Prices dataset. <https://data.humdata.org/dataset/wfp-food-prices-for-afghanistan> This dataset contains two resources both are CSV format with HXL tags. One has approximately 32,000 rows with 14 columns and a size of 3.8MB. The other is used for QuickCharts and has only 881 rows and 3 columns with a size of 49.5KB. The resources were from 21st July, they are updated weekly.

In the first instance changes were made manually to these files to provide files for comparison with well-known changes (rows were deleted from the start and end of the file, and a single value was modified on one row). Subsequently versions of the files from 14th July and 12th May were extracted from the S3 filestore with a view to getting examples representing typical amounts of change (14th July) and the maximum amount of change available (12th May).

The performance of difflib depends on the file changes, as well as the file size. For example, for the manually altered large file the single cell change detection takes 0.4 seconds whilst for the real-life file with 473 changes it takes 16 seconds.

Experiments were done on processing files column by column rather than as whole rows. These highlighted a couple of issues. One is that edits are more difficult to spot in a single column since they will often be for a large fraction of a cell whilst in a whole row most of the row will remain the same.

Secondly, we observed for one modest sized file that processing a single column could take fractions of a second or many minutes, depending on the file content. This was addressed by applying a time limiter function to the column processor – borrowed from here:

<https://towardsdatascience.com/limiting-a-python-functions-execution-time-using-a-decorator-and-multiprocessing-6fcfe01da6f8>

Modified slightly to use the multiprocess library rather than multiprocessing because it works, multiprocessing produces a pickle error. This could be used across the library to control the time for the checks.

Why there is this large difference in processing time is not clear. The column in question was comprised of many long text strings with relatively little change (probably).

### Further Work

1. The row number for a changed cell will currently be wrong because the difflib output contains more rows than either input file.

2. The code cannot currently handle Excel files, even in simple tabular form although this will be easy to fix

3. Testing for multiple changes on a single row.

4. Survey file sizes and change fractions for files on HDX to establish what fraction of files can be covered by a tool like this – the limiting factor will be the size and number of files to be processed.

5. Establish required output, in particular do we want to detail cell level changes if 100s have occurred in a resource?