TSB Data Engineering assessment

### Author: Nik Holden

### Date: 16/06/2020

Assessment provided by Justin Taylor, Head of Integration & Data, TSB



# Assessment outline:

Using data from the following three sources:

1 - <https://geohub.npdc.govt.nz/datasets/2f2aadf60dc54b7dacc2c62e0dc4d2e1_0/data>

Property data

2 - [https://geohub.npdc.govt.nz/datasets/209a23b0615f404d9b9e2db2ca2f8fdc\_4/data](https://aus01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgeohub.npdc.govt.nz%2Fdatasets%2F209a23b0615f404d9b9e2db2ca2f8fdc_4%2Fdata&data=02%7C01%7CJustin.Taylor%40tsb.co.nz%7C6a507cc30c95448e2a4d08d80e7ff6da%7Cb7f8d04400864ff192a370357b4e8f35%7C0%7C0%7C637275290051166090&sdata=zzBlhzGMtmNFdnViB4CqCBUqjdPX9W776ebn0QeOlT8%3D&reserved=0)

Parks and natural area data

3 - <https://twitter.com/NPDCouncil>

Tweets from New Plymouth District Council

Produce a file or DB table containing the output for the following fields:

|  |  |  |
| --- | --- | --- |
| label | field | source |
| Assess number | AS\_ASSESS\_NO | 1 |
| Property address | full\_address | 1 |
| Capital value | capital\_value | 1 |
| Annual rates | annual\_rates | 1 |
| Park Name | LocationSite | 2 |
| Latest Information | Derived value and full\_text | 3 |

In the case of the ‘Latest Information’ field, display the date of the latest update or if a tweet from the New Plymouth District Council (NPDC) contains the street name, display that tweet.

# Designed solution:

## See Appendix 1 for an overview of the processing steps.

## **Sourcing And Landing The Data:**

Data for sources 1 and 2 are managed by NPDC and made available through their GeoHUB system. The GeoHUB service is powered by Esri and provides data about NPDC natural and physical assets. These are made available through the use of ArcGIS REST API, or by downloading a data set from GeoHUB.

For my solution, I decided to use the available REST API and incrementally download source data at a maximum of 2000 records per-download (the maximum limit imposed on public API access). I chose to download the source data in GeoJSON format. Downloading the data was performed using Python3 and the ‘requests’ library.

Each download of source data was written to a raw source file as it was downloaded. The date and time (in format ‘yyyymmddHHMSS’) is included in the file name and the extension is ‘geojson’. The date and time has been included in the filename to enable identification of a specific file as well as allowing multiple files in a single day to be sourced without compromising

the authenticity of each file.

As well as saving the raw data, two individual CSV files containing the GeoJSON ‘[features][0][properties]’ data and ‘[features][geometry]’ data are created’ (npdc\_XXX.csv and npdc\_XXX\_coord.csv respectively). The field OBJECTID is included in both data sets as the primary key, thereby allowing the coordinate data to be paired with the object data for future analysis.

For the purposes of this assessment I chose to record the raw and processed data in either JSON or CSV format. My experience has shown me the importance of keeping raw data that is independent of a hosting solution. Separating the JSON data into separate CSV files allows both the easy ingestion of data into other data solutions as well as the use of the data by other tools, such as R.

The date and time is appended to the data in each CSV file, based on the date and time the data was retrieved from the source to add clarity of when the data was obtained. The data and time is not appended to the raw JSON data, but is included in the file name and can be extracted to reproduce the CSV files for a specified date and time.

With minor tweaks the Python script can directly insert data into a database table if preferred.

Twitter data is obtained in a latter process.

## **Cleaning The Data:**

Data from all the provided sources was obtained by API calls and provided in JSON format. Data from the NPDC GeoHUB service is split into CSV files which contain ‘[features][0][properties]’ (object) or ‘[features][geometry]’ (coord) data.

The ‘coord’ data is transformed to contain the ‘OBJECTID' field to be used as a primary key for joining to the ‘object’ data.

All CSV files have the date and time added as individual data fields in the format of ‘yyyymmdd’ for date named ‘rec\_date’ and ‘HHMMSS’ for time named as ‘rec\_time.

‘Object’ data from both the Rates Assessment and Parks and Natural Area’s sources include all available data fields.

The ‘full\_address’ field from the Rates and Assessment ‘object’ data source is split into the following fields based on the data contained with the ‘full\_address’ field:

|  |  |
| --- | --- |
| Column | Description |
| Street number | Street number of associated with the property assessment |
| Street | Street the street number is located on |
| State Highway | If the property is on a State Highway, the number of the State Highway |
| Area | The area or Taranaki managed by NPDC the property is located in |
| Alt Area | An alternative area included in the full\_address value |

All values mentioned above are appended to the data structure and are standardised in uppercase format to ensure consistency of values when querying, filtering or joining on columns.

Park and Natural Area ‘object’ CSV data is summarised to obtain the total number of parks and natural spaces, the total area, and a list of parks and natural spaces in a given area. The parks and natural spaces in a given area are retained in list/array format to indicate they are separate values that can be used for other purposes.

Both the Rates and Assessment property and the Parks and Natural Areas ‘object’ CSV files can be joined back to the ‘coord’ CSV files using the ‘OBJECTID’ primary key allowing for nearest neighbour analysis, route analysis or any other geospatial analysis.

## **Twitter Data:**

For the purposes of this assessment, I choose to download all publicly available tweets from the NPDCouncil twitter feed. I compare the a compiled list of street names from the Rates and Assessment property data to check if any streets appeared in any of the available tweets. If a street name does appear, the tweet is displayed. If a street name does not appear the record date is shown.

I used Python to download the tweets using the ‘python-twitter’ library and to perform the analysis of a street name appearing in the tweet. To achieve this I:

1. From the Rates and Assessment ‘objects’ CSV file data, I create a list of all street names
2. The ‘extended\_mode’ in the twitter API is used to obtain the full text from the tweet
3. From the downloaded tweets I iterate a loop using each street name over each tweet
   1. The tweet is capitalised to ensure any street name mentioned is recognised
4. If the street name appears in the tweet I add street name and tweet text to a CSV file
5. The created CSV file is read into a Pandas dataframe and the tweet added to the ‘latest\_info’ column.

For this assessment, I downloaded the tweets as a batch of tweets with the expectation a street name appeared. The assessment guidelines call for consideration to be given to real time streaming of tweets and the assignment of a tweet to the data set during the day as opposed to once a day. While the process I developed downloads all available tweets from a specified handle, altering the application to use the Twitter Streaming API and following the user ID for NPDCouncil will enable real time tweets to be received. With minor modifications, as a tweet is received the process of comparing the tweet against street names can be performed and the tweet updating the column for any address on the mentioned street. Alternative a Lambda function could be invoked on a cloud based service to process the incoming tweet (a sample program is provided).

# **Automating The Process:**

The script I have developed will source data from the three sources mentioned, save the raw data, process the raw files and produce the desired end result. The script can be executed on an on premise or hosted solution. For the assessment exercise all files are saved to the users local folder, although with minor adjustments, files can be saved to a designated location and records inserted into DB tables.

In this assessment, two of the sources are pull only so a scheduling system of sorts would be required. A solution I have used is to call a Python script via shell command action from an Oozie workflow on a Cloudera Hadoop instance, with the data written to an HDFS location. By using an Oozie coordinator, the workflow was scheduled to run at a given time at a given frequency. This is but one option of automating this script due to the need to pull data rather than initiate a process/code from an event that was pushed to a listening service

# **Considerations:**

## **Robustness Of Process:**

Considerations towards the redundancy provided by the platform and it’s availability to manage misbehaving processes or misbehaving hardware is critical to effective data processing pipelines. Enabling a mechanism whereby processing scripts and code is managed external to the hardware and software performing the processing, would allow for the availability of processing resources to be monitored and code appropriately sent to available resources.

Pipelines should be developed in such a way that allows for data to be discarded. If a pipeline fails only the pipeline itself should be impacted and each step would be independent of others except when waiting on a pre-processing activity to complete. For this reason I am a firm believer in sourcing data into a raw environment while simultaneously processing it.

Should a fault in the processing pipeline occur, the raw data can be used to restart the process, or in the event of a catastrophic failure or data store unavailability, the raw data can be used to reestablish a data situation to a point in time. By decoupling the sourcing of raw, and the processing of data, sourcing data can continue in the event the processing mechanism fails or is significantly slowed. In such an event, other clusters, nodes, or resources can be activated to address the processing needs.

## **Maintaining Data Quality And Authenticity:**

Authenticity of data and data sources should be constantly considered. Quality user data is only as good as the raw material. Notwithstanding quality issues of the raw data, ensuring strict standards in the processing of data to follow standardised formats will ensure consistency of data assets, reduction in confusion and adherence to confirmed processing standards.

Ensuring the authenticity of the source and maintaining authenticity of raw data allows the holder of the raw data to ensure and maintain quality of all processed and transformed data.

## 

## **Adjusting The Process To Handling An Increased Data Set:**

In this exercise the data is intended to be sourced in two different ways: Streamed (pushed data) and static (pulled) data. In the event of streamed/pushed data, utilising an on demand/cloud computing environment which allows the dynamic allocation of resources to cope with peaks in processing and utilising serverless, always ready, architecture with low latency data stores. This will ensure the flow of pushed data is unimpeded and resource/computational costs are aligned to the level of processing performed.

In the case of static/pull data, in this exercise, the source was limited to 2000 records per call. WIth the small number of records (less than 40,000) downloading the complete data set in one looped operation was feasible. However, as the final data set increased in size, parallel importation of the data could be considered. Multiple streams of data importation, assuming the source system allows it, with processing on the client side (either on premise or cloud based) could maintain the level of overall processing duration. How the processing is charged needs to be considered. Time processing may be less or more than the number of processing operations. In other words reducing the time may increase costs.

# **Considerations For Sensitive Data:**

A balance between maintaining protection of sensitive and identifiable data and a need to utilise data for strategic and planning purposes should be considered. A strategy to decouple sensitive information (such as personally identifiable information that can be linked to publicly sourced information), financial information that could be misused for personal gain or controlled by scheme requirements, or personal data managed by an organisation should be in place. The ability to separate sensitive data while maintaining a means to reinstate the relationship is a must to preserve confidence in the use of the sensitive data while maintaining it’s fit for purpose.

In this hypothetical use case, the source contains data that could benefit from being split into different data sets while still enabling the analysis of it. Splitting the address from the financial information relating to the capital value of the address and the rates assessment amount and replacing them with a classification rating to still allow comparative analysis but maintain privacy of who the data is about.

Appendix 1.

Pipeline flow

1. Download GeoHUB data (using ‘requests’ and ‘csv’ libraries - this step is performed twice for each GeoHUB endpoint)
   1. Query source for total number of records
   2. Set remaining records = to max number of records
   3. Begin loop while counter for remaining records is > 0
      1. Create call to GeoHub endpoint for max records allowed
      2. Write/append raw GeoJSON response to file
      3. Write/append ["features"][0]["properties"]JSON objects to CSV file, including fields for date and time
      4. Write/append ["features"]["geometry"]JSON objects to CSV file, including fields for date and time
2. Transform data in CSV files (using ‘pandas’ and csv; libraries)
   1. Read CSV file into dataframe
   2. On the property rates and assessment CSV
      1. Split the ‘full\_address’ value into 4 individual parts
         1. Street\_no
         2. Street
         3. State\_highway
         4. Area
         5. Alt\_area
      2. Set the new values to uppercase
   3. Write data to CSV
   4. Return data frame
3. Retrieve Twitter messages (using ‘python-twitter’ and ‘pandas’ libraries)
   1. Create call to Twitter endpoint (GetUserTimeline)
   2. Download Twitter object data and assign to variable
   3. Load cleansed property rates and assessment data (created in previous step) into pandas dataframe
   4. Create a set using the ‘street’ column
   5. Begin loop assessing the list of street names compiled to ‘ful\_ text’ item field on each Tweet
      1. If the street name appears in a Tweet
         1. Add street name and tweet to dictionary (It is acknowledged that if a street name appears more than once, the last instance of the tweet will be captured in the dictionary. This step is performed as an example for being able to associate a Tweet with a street)
   6. Write dictionary to CSV file (this is performed using the standard file writer to avoid overcomplicating the dictionary)
4. Summarise Parks and Natural Areas data (using ‘pandas’ libraries)
   1. Read parks and natural spaces CSV into dataframe
   2. Calculate the number of parks and natural areas and the sum of the total area these spaces make up
   3. Reduce records by filtering out any records that do not have a LocationSite value
   4. Select on the LocationArea and LocationSite columns
   5. Set the LocationArea column to the standardised uppercase value for joining compatibility
   6. Group LocationArea
   7. Flatten LocationSite into a Tuple of values (producing a single row per LocationArea with a Tuple of LocationSites)
   8. Merge the aggregated data compiled in step ‘b’ with the list of LocationSites reated in step ‘g’
   9. Write dataframe to CSV
5. Produce file output (using ‘pandas’ library)
   1. Steps 2, 3, & 4 are called in this final step. Step 3 is dependant on step 2
   2. Load CSV created in step 2 into dataframe (Twitter data)
   3. Left merge (join) property rates data and summarised park data
   4. Left merge (join) Twitter data
   5. Apply 2 if statements
      1. If the tweet column is not null add tweet value to new ‘latest\_info’ column
      2. If the tweet column is null then add the record date value to the ‘latest\_info’ column
   6. Write data frame to CSV file