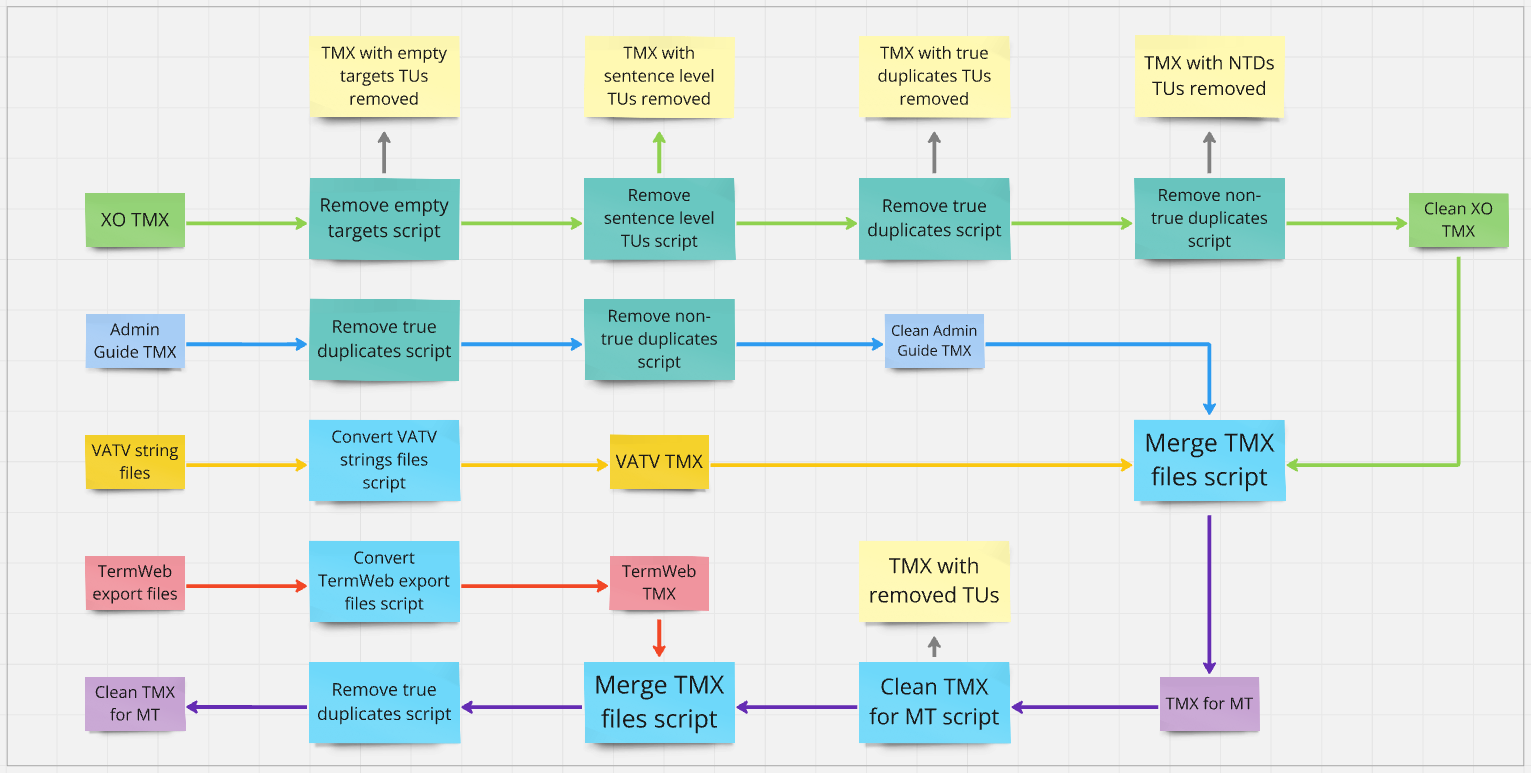
Using the Provided Scripts

Processing Analysis, and Quality Control are broken down into 2 folders:

* Processing
* and QA & Analysis

# Overview

Working with the Translation memories for Workday in an orchestrated sequence, language data is cleaned and made ready for re-integration in the TMS as well as assets that are prepped for use as Machine Translation engine training data.



## Dependencies

* Python 3.x   
  (Pylance is also recommended)
* To run the scripts, the following Python packages are required:
  + `PythonTmx`: A library for handling TMX files.
  + `glob`: A built-in library for file pattern matching.
  + `datetime`: A built-in library for manipulating dates and times.
  + `re`: A built-in library for regular expression operations.
  + `csv`: A built-in library for reading and writing CSV files.
  + `openpyxl`: A library for reading and writing Excel (XLSX) files.

## TM Cleaning

Regular TM maintenance reduces leveraging times, reduces confusion by linguists, and the potential for mis-leveraged segments during translation. These scripts provide a comprehensive toolkit for processing TMX files, including cleaning, merging, and converting various formats.

### Process

The TM processing scripts are meant to be run in the following order:

* **1st** - remove\_empty\_target\_tus.py
  + This script removes any translation units (TUs) that have empty target segments from TMX files.
  + **Outcome**: Two TMX files are generated: one with non-empty target TUs (`clean\_tm.tmx`) and another with empty target TUs (`removed\_empty\_target\_entries.tmx`).
* **2nd** - remove\_sentence\_level\_tus.py
  + This script removes sentence-level TUs that are part of larger multi-sentence entries.
  + The x-context\_seg\_key is use as a key for the dictionary, and the value is a list, if any entry is part of a bigger entry, it is discarded, if it is not part of a larger entry and does not contain the current seg-key value, it assumes that is a new entry and adds it to the list.
  + **Outcome**: Two TMX files are generated: one with clean TUs (`clean\_tm.tmx`) and another with the removed sentence-level entries (`removed\_sentence\_level\_entries.tmx`).
* 3rd - remove\_true\_duplicates.py
  + This script identifies and removes true duplicates from TMX files based on source and target segments. Removes segments that have the same seg\_key and content.
  + **Outcome**: Two TMX files are created: one with unique TUs (`clean\_tm.tmx`) and another with the removed duplicates (`removed\_true\_duplicates\_entries.tmx`).
* 4th - extract\_ntds.py
  + This script extracts non-true duplicates from TMX files, identifying segments that are not unique.
  + **Outcome**: Two TMX files are produced: one with unique TUs (`clean\_tm.tmx`) and another with non-true duplicates (`ntds\_entries.tmx`).

## MT Cleaning

* clean\_tmx\_files\_for\_MT.py
  + This script cleans TMX files by removing true duplicates and entries that do not meet specific criteria (e.g., alphanumeric segments, segments with fewer than 5 words).
  + **Outcome**: Two TMX files are created: one with cleaned TUs (`clean\_tm.tmx`) and another with rejected segments (`rejects.tmx`).
* convert\_vatv\_files\_to\_tmx.py
  + This script converts VATV CSV files into TMX format, extracting source and target values.
  + **Outcome**: TMX files are created for each CSV file, named according to the original CSV file name.
* merge\_tmx\_files.py
  + This script merges multiple TMX files into a single TMX file.
  + **Outcome**: A single TMX file (`jointed\_tm.tmx`) is created, containing all TUs from the merged files.
* convert\_termweb\_files\_to\_tmx.py
  + This script converts TermWeb XLSX files into TMX format, using the main column as the source and the rest as targets.
  + **Outcome**: TMX files are created for each target language found in the TermWeb files.

## Ongoing maintenance

* remove\_pre\_date\_tus.py
  + This script removes entries from TMX files that were created before a specified date. The user is prompted to enter a date in the format MM/DD/YY.
  + Using this script against TMX exports later in time enables the data cleaned after this initial cleaning date to be extracted and will capture more recent data that needs to be cleaned.
  + **Outcome**: Two TMX files are created: one containing entries after the specified date (`clean\_tm.tmx`) and another with the removed entries (`removed\_old\_entries.tmx`).

## Testing

Several scripts are used for counting TUs based on metadata and extracting any duplicates into .CSV files for analysis.

* count creation dates.py
  + This script creates a .csv file that shows the number of TUs in the TMX file by date.
  + **Outcome**: The .csv file ‘creationdate\_counts.csv’ with columns for the UTC date, the human-readable date and the count of how many TUs were created in the TMX on each date.
* count lastusagedate.py
  + This script creates a .csv file that shows the number of TUs in the TMX file by date.
  + **Outcome**: The .csv file ‘lastusagedate\_counts.csv’ with columns for the UTC date, the human-readable date and the count of how many TUs were created in the TMX on each date.
* duplicates.py
  + This script creates a .csv file that shows the number of TUs in the TMX file by date.
  + **Outcome**: The .csv file ‘duplicates.csv’ with columns for Segment Key, Creation Date, Language, TU Content, and the Filename included in the metadata for the TU.
* ExtractfromTMX.py
  + This script creates a .csv file that contains the TUs in the TMX file for ease of analysis and review.
  + **Outcome**: The .csv file ‘output\_translations.csv’ with columns for the date, the segment source and target content as well as segment metadata.
* Locate\_TM\_in\_Proj\_Template.py
  + Used to audit what TMs are configured in which project templates to ensure the Project templates are covered by processing work.
  + **Outcome**: The expected output is a list of project templates that match the specified TM ID, formatted as:

Template ID: <template\_id>, Name: <template\_name>