**MultilingProfiler (ver. 20230523)**

Documentation by Laurence Anthony

Version 1.00 (2023-05-27)

1. **General Description**

The MultilingProfiler site is designed to create vocabulary profiles of French, German and Spanish texts to help users assess the suitability of written texts and listening transcripts for target learner groups.

A picture containing text, screenshot, font, website

Description automatically generated

Figure 1. MultilingProfiler Landing page

1. **Site structure**

The MultilingProfiler site offers the following main pages

* Home: A landing page with a direct link to the main vocabulary profiling tool
* MultilingProfiler: The main vocabulary profiling tool page
* Word Families: A page showing searchable word families for French, German and Spanish
* FAQ: A page answering the most frequent questions about the site
* About: A page with general information about the site development
* Contact: A page giving contact information if a user experiences any trouble with the site

A screenshot of a black screen

Description automatically generated with low confidenceAll files for the MultilingProfiler site are contained in a single ROOT folder on the target server. The file structure under ROOT is as follows:

* + css
    - ncelp.css
      * A custom css file for tweaking the w3.css base framework.
    - reset.css
      * A css 'reset' file to create a blank slate for later css processing. The file is probably unnecessary.
    - w3.css
      * The w3.css framework  
        (see https://www.w3schools.com/w3css/default.asp for details)
  + data
    - dictionary\_lists
      * Raw vocabulary lists used as reference lists by the "MultilingProfiler" tool when processing target texts.
      * JSON files mapping the raw word families to word families (levels 2 and 3) for display in the "Word Families" page.
    - gcse\_lists
      * Raw vocabulary lists used as reference lists by the "MultilingProfiler" tool when processing GSCE target texts.
    - ncelp\_lists\_ks3
      * Raw vocabulary lists used as reference lists by the "MultilingProfiler" tool when processing KS3 target texts.
    - ncelp\_lists\_ks4
      * Raw vocabulary lists used as reference lists by the "MultilingProfiler" tool when processing KS4 target texts.
    - readme\_gcse.txt
      * A text file explaining the structure of files that are exported when the "gcse\_lists" are used. This file is included in the export.
    - readme.txt
      * A text file explaining the structure of files that are exported when non "gcse\_lists" are used. This file is included in the export.
  + images
    - a folder containing all images used in the site
  + js
    - FileSaver
      * A JS library for exporting files (e.g. results from the MultilingProfiler tool).
    - jquery-3.6.0
      * A JS jquery library used to display tables in the main site.
    - jquery-ui
      * A JS jquery library used to add interactive buttons to tables in the main site.
    - vanillaSelectBoxAntLab
      * A customized version of the JS vanillaSelectBox library used for comboboxes in the main site.
    - app.js
      * The main (custom) JS library for handling all interactions in the main site.
    - handlebars-v4.0.10.js
      * A JS library for handling templates in the main site.
    - trunc\_fix\_for\_ie.js
      * A JS library for handling inconsistencies with old IE browsers.
  + templates
    - About. html
      * A handlebars library template for the About page.
    - Contact. html
      * A handlebars library template for the Contact page.
    - Dictionaries.html
      * A handlebars library template for the Word Families page.
    - Faq.html
      * A handlebars library template for the FAQ page.
    - Footer\_long.html
      * A handlebars library template for the footer of all pages (no longer used).
    - Footer.html
      * A handlebars library template for the footer of all pages.
    - Header.html
      * A handlebars library template for the header of all pages.
    - Home. html
      * A handlebars library template for the Home page.
    - Wordprofiler.html
      * A handlebars library template for the Wordprofiler page.
  + custom.txt
    - The dummy file used in place of standard level list when a 'custom list' option is used.
  + favicon.ico
    - The favicon icon displayed in the browser when the site is visited.
  + index.html
    - The home page of the site (with placeholders for the various templates.
  + manifest.json
    - A manifest file to determine the base color scheme of the site.

Additional JS libraries used by MultilingProfiler and loaded at launch are as follows:

* Quill
  + A JS library used for displaying and editing text supplied by the user
* Xrexexp
  + A JS library used for handling Perl-based regular expressions (regex)
* dataTables
  + A JS jquery library to show tables
* dataTables.buttons
  + A JS jquery library to add interactive button to dataTables tables
* Jszip
  + A JS library to create zip files used when exporting results
* jszip-utils
  + A JS library used to save files used when exporting results
* Pdfmake
  + A JS library used to save create PDF files (no longer currently used)

1. **Deployment**

The MultilingProfiler site is built only using HTML, CSS, and JAVASCRIPT files. There is no backend database to manage. Deployment of MultilingProfiler involves the simple process of copying all files described in the "Site structure" section to a suitable base folder on a server that has read access. This can be achieved most easily by creating a zip file of the entire website folder and sending this to the appropriate IT team member for upload to a server. It can also be achieved directly through an FTP upload tool such as FileZilla. All paths are described relative to this base folder. The site can then be accessed at the following URL (where the ‘/index.html’ part of the path is optional):

https://PATH\_TO\_BASE\_FOLDER/index.html.

A screenshot of a computer

Description automatically generated

Figure 2. FileZilla upload to server

1. **Common Maintenance Activities**

The following descriptions assumes the website files are under and project folder with the following structure:

* MLP\_project (folder)
  + MLP\_webapp (folder)
    - css (folder)
    - data (folder)
    - images (folder)
    - js (folder)
    - templates (folder)
    - custom.txt
    - favicon.ico
    - index.html
    - manifest.json
  + dictionary\_data\_to\_json.py
  + verify\_data.py

**A screenshot of a black screen

Description automatically generated with low confidence4.1 Updating word lists**

For all new word lists with the exception of "dictionary lists", the lists can be added to the appropriate sub-folders under " MLP\_webapp /data" to update the site. No other changes are necessary. Sub-folders are consistently named "de", "es", and "fr" to reflect the three languages supported by the site. The format of the raw word list files to be placed in the "de", "es", and "fr" sub-folders is documented by the NCELP team.

In the case of " MLP\_webapp /data/dictionary\_lists", JSON files are used to compress the data to improve the site performance. Accordingly, two JSON files (dictionary\_wf2.json; dictionary\_wf3.json) need to be regenerated. A Python script to regenerate these JSON files is included with this documentation. Note the following:

* the Python script contains a hard-coded path name to the folder that contains the updated dictionary list files.
* the Python script contains a hard-coded variable name for the type of JSON file to create (“wf2” and “wf3”). The script should be run twice, once for “wf2” and once for “wf3”.

The site uses a JSON inventory file of all data lists to auto-generate the necessary options in the interface. The Python script that generates to the JSON files for the dictionary lists, also creates the main JSON inventory file.

A simple strategy to create the necessary JSON files is to copy the new “data” folder to the root of the website project (MLP\_ project), run the script “add\_json\_files\_to\_data.py” and then drag the revised “data” folder into the web folder “MLP\_webapp”, overwriting the existing data folder.

**4.2 Changing the wording of site pages**

Changes to the site page wording can be achieved by editing the corresponding template in the "templates" folder. Each template is written in HTML. In the current version, the templates contain no dynamic variable names. The templates make use of numerous CSS class names from both the "w3.css" framework file and the "ncelp.css" file. Changes and additions to these class names should be unnecessary.



A screen shot of a computer program

Description automatically generated with low confidence

Figure 3. Example of template file (Contact.html)

**4.3 Adding dynamic links to site pages**

Some site pages contain dynamic links to other pages in the site. To allow the site to display different site pages when a link is clicked, custom CSS names are used. Currently, the following CSS are used:

* multilingProfilerToolLink – links to the MultilingProfiler page
* aboutToolLink – links to the About page

Additional links can be included by adding a suitable CSS class name to a template HTML tag and updating the "app.js" to process this CSS class. (See the "setupHomeTool" function in "app.js" for examples)

****

A screen shot of a computer screen

Description automatically generated with low confidence

Figure 4. Example of dynamic link and associated code in setupHomeTool()

**4.4 Adding additional list types**

Adding additional list types to the system is a very complex process due to the many list-dependent parameters and processing conditions that are imposed on the system. The setup process is described in code in the "setupProfilerTool" function of "app.js". In this function, the following information is defined:



* the various lists to be displayed on the MultilingProfiler page
  + Comment out lists that you don’t want to be shown.
* the labels shown for the Global Coverage statistics table
* the labels shown for the other word-level statistics tables
* the labels shown for the other word-family-level statistics tables

It is important to note, however, that the choice of list type affects the code in multiple places. To identify where changes are necessary, it is advised to search through the app.js file for conditional cases based on "$("#listChoice").val()"

1. **Google Analytics**

The MultilingProfiler site is a single-page site, with the single page being updated dynamically depending on the interaction of the user. As a result, the Google Analytics tracking script needs to be placed only in the <head> tag of index.html. Currently, it is places at the beginning of the <head> tag. All further setup of Google Analytics happens within the Google Analytics management tool, which is beyond the scope of this documentation.

[Note that Google Analytics will likely need updating in mid 2023 to reflect changes to the Google Analytics API.]

1. **Data processing**
   1. **Flow of actions on site loading and individual page loading**

When the MultilingProfiler site loads for the first time, the following actions occur through the "app.js" script:

* tooltips are activated across the site
* the default page tab is opened
* the header and footer templates are added to the main page
* the "Home" page template is displayed (via the "openTab" function)
* dynamic links are created (via the "setupHomeTool" function)

When a tab for a particular page is selected, the following actions occur through the "app.js" script:

* the selected page template is displayed
* parameters and actions in the page are set (via the corresponding "setup[NAME]Tool" function)

Note that the page setup procedure is quite complex for the “Profiler” tool (as described in the "setupProfilerTool" function).

* 1. **Flow of actions during word list profiling**

In the MultilingProfiler page, after a user selects a word list and its associated parameters and then adds a target text to the "Profile Window" and clicks the "Profile Text" button, the following actions occur through the "app.js" script.

* The "loadFilesAndCreateProfile" function is called. This function loads raw text data and then serves as a wrapper function for the remaining functions below.
* The processExtendedEntries() function is called.
  + this function includes various conditional statements for language tokenization, inflection and derivation processing, and word filtering
* The "highlightStandardWordsAndMWUS" function is called, and the spans for MWUs are returned, which are highlighted in the next step.
  + this function includes various conditional statements for handling MWUs that come from the default lists or custom lists
* The "createProfile(offset, match\_pairs)" function is called, where all word highlighting and statistics generation takes place.
  + this function includes various conditional statements for list type loading, language tokenization, apostrophe use, hyphen processing, and word list processing
* The "setGAProfileEvent()" function is called for Google Analytics to track that a profile was generated.
* The "showOutOfListCount()" function is called to show global statistics.