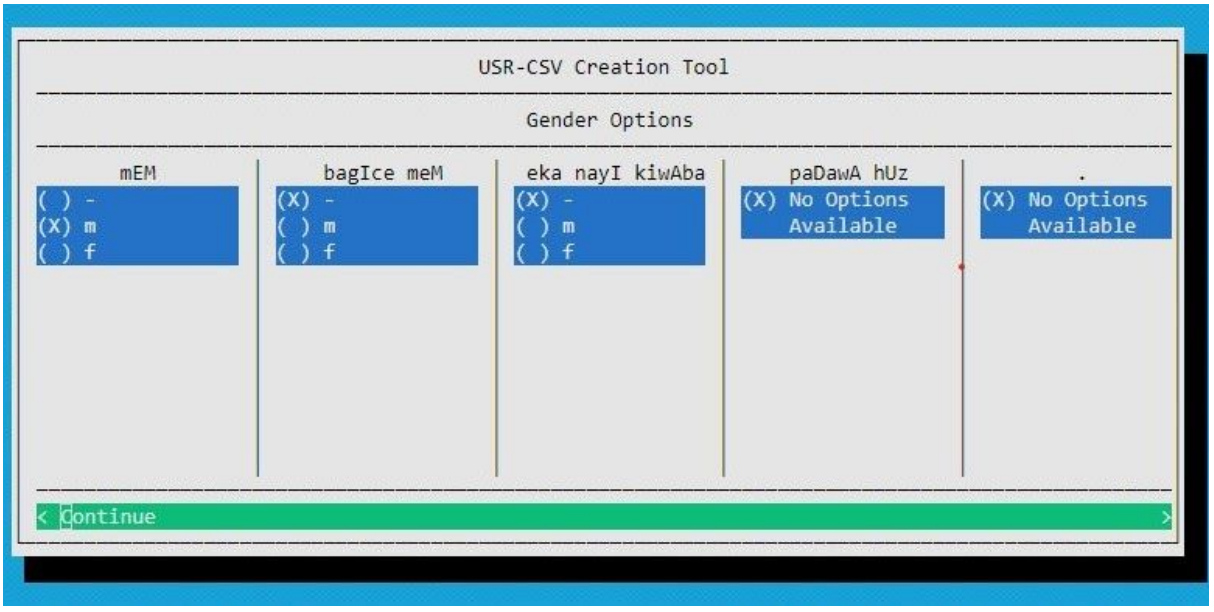
The Textual User Interface (TUI)

A Textual User Interface (TUI), built using the python package urwid helps ease the USR-CSV creation process. The TUI addresses the cases where multiple candidates are available for a given key in a dictionary. It allows the user to select TAMs, Hindi concepts and GNP information from a UI containing a list of options.

The process of creation of the TUI is handled by a class *UsrCreationTUI*. To best understand the code, please go through urwid's tutorial here and examples here. Finally, go through the example code *graph.py* here. Although the class is quite different from the example code, going through the mentioned links is enough to understand its working.



For displaying any screen of the TUI, we instantiate it with three inputs, namely *title*, *OptionViewsTitles*, *Options* and *OptionsSelected* in the following way:
UsrCreationTUI('Gender Options', myGrps, genderOptions, genderOptionsSelected).main()

The input details are as follows:

- *title* : Represents the title of the TUI.
- *OptionViewsTitles* : Represents a list of strings containing the titles of the subviews inside the TUI, where each subview is independent and contains multiple options. In the above image, *mEM*, *bagIce meM*, *eka nayI kiwAba* etc. are some of the elements of OptionViewsTitles list.
- *Options* : Represents a list of lists of strings, whose each element list contains the options available for the subview it represents.
- *OptionsSelected* : Represents a list of indices representing the options selected by default when opening the TUI window for each of its subview.

Minor change: Argparse

Added argparse to document the command line options and assist the user to run the script correctly. This basic documentation is shown when a user tries to run the script with the wrong arguments or runs it with the *--help* switch. This also adds the *--edit* switch whose presence runs the TUI and absence runs the original script itself.

Minor Change: Regular expression for parsing input files

Parsed the groups of the input file using the command *re.split('^\s*\(\s*|\s*\)\s*\(\s*|\s*\)\s*$', sent)*. This use of a regular expression helps parse the input in a single pass and more importantly makes the input *not space sensitive*, i.e. there can be any number of spaces between the tokens.
The regex ^\s*\(\s*|\s*\)\s*\(\s*|\s*\)\s*$ is a union of the regular languages "^\s*\(\s*", "\s*\)\s*\(\s*" and "\s*\)\s*$", note that "|" is the union operator in RegEx.
The first regex matches the patterns with any number of spaces in the beginning of the string followed by an open parentheses followed by any number of spaces.
The second regex matches the patterns with any number of spaces in the middle of the string followed by a close parentheses followed by any number of spaces followed by an open parentheses followed by any number of spaces.
The third regex matches the patterns with any number of spaces in the middle of the string followed by a close parentheses followed by any number of spaces.

Minor Change: Rectified the algorithm to select the longest TAM.
This algorithm was wrong, rectified it. The logic is simple, just maintain a running longest TAM among all the TAMs seen yet.

The install script (install-project.sh)

The script is quite straightforward, the only notable thing is the way *USR-csv.tgz* file is downloaded from Google Drive using *wget*. In case the location of the file changes, take note of the steps on this site.