Using igt-edit.

To load the program, run igt-edit.exe in Windows. There are a few support DLLs which must be located in the same directory as the executable. You do not necessarily have to run the program with the 'current directory' set to the location of the program. That is, you can create a shortcut to the program which specifies a different 'current directory.' But note that the configuration settings file 'app-settings.xaml' will be created in the current directory. So if you later start the program with a different current directory, your saved settings will not be loaded.

Settings file

A few administrative settings are stored in the app-settings.xaml file. Most of the settings can be changed via a corresponding item on the "Options" menu in the program. The app-settings.xaml file is a human-readable XML file, so you can inspect it to see what it contains, or to manually tweak settings. At this point, the following settings are saved:

"Escape key exits" – If selected, pressing the 'escape' key on the keyboard will exit the program. If 'Save on exit' is also selected, then all of your current IGT files will be saved. The default value is 'false' (escape key does not exit the program).

"Save on exit" – If selected, all of your current IGT files will be saved when exiting the program. This includes clicking on the close icon, selecting exit from the menu, or via pressing the escape key (if enabled). The default value is 'false' (Files are not saved upon exiting the program.)

"Reload Last session" – If selected, the IGT files you had open previously will be re-opened when the program starts. If possible, the specific IGT instance you were working on will also be re-selected. The default value is 'false' (Files are not reloaded when the program starts)

"Reset app settings" – This menu command deletes the app-settings.xaml file, withlee the result that default settings will be loaded the next time the program starts.

Also retained in the settings file is whether the overall application "window" was maximized on the screen or not. The previous state is restore the next time you run the app.

Corpora pane

The leftmost pane of the app shows the IGT files (each an "IGT corpus") that are loaded. You can have zero, one, or more IGT corpus files loaded. Selecting an IGT corpus file here causes all of the IGT instances it contains to be displayed in the Corpus pane. You can resize (or fully hide) the corpora pane using the mouse 'grip' along its right-hand vertical edge. You can move amongst the IGT corpus files using the 'up' and 'down' keyboard arrows. ***Pressing the 'right' keyboard arrow navigates to the corpus pane.***

Corpus pane

To the right of the corpora list is the Corpus pane, which lists each of the IGT instances available for editing in the currently selected IGT corpus. Select an IGT instance using the 'up' and 'down' keyboard keys. The contents of the selected IGT instance are displayed in the main editing area of the application. ***Pressing the 'left' keyboard arrow navigates to the corpora pane.***

IGT editing concepts

The editing of IGT instances is based on two concepts: Tiers and Parts. A Tier represents one or more lines of text. There are three basic types of tiers:

Simple **"text" tiers** represent one line of text which has a designated or "well-known" logical function. Basic tiers can also represent raw source text possibly spanning multiple lines of a source document. In either case, text tiers represent text as a whole, and thus serve as the basis to which the "standoff annotations" of all other tiers must ultimately refer.

Group tiers can be used to arbitrarily group other tiers together. In this way you can nest tiers with related logical functions together. To move a tier into a group, you can drag and drop it. To move a tier out of its group into its parent (or parent group), right click the tier and select "Promote."

Note that for certain operations, Tiers can only "see" other tiers in their group *plus* tiers in parent groups. For example, in the "Align with…" command, tiers which are nested below peers of the current tier's parent are not eligible for alignment unless they (or the current tier) are first promoted.

Parts tiers are tiers that refer to segmented parts. The most basic parts tier is the segmentation tier, which describes an arbitrary standoff segmentation of a text tier. Other types of parts tiers refer to these segmented parts. These tiers include the POS tag tier (which assigns POS tags to segmented parts), dependency tier (which defines dependency relationships between segmented parts), and alignment tier (which defines alignment relationships between two sets of segmented parts).

Text tiers

There are two types of text tier.

The **basic text tier** can show a single line of source text, or a section of multi-line source text. Currently basic text tiers cannot be entered in the program; they are already present in the IGT instances of the XAML file(s) which are loaded.

The "**multi-text**" tier is assembled from other source text tiers, which can be basic text tiers or other multi-text tiers. To create a multi-text tier, select "Join with" on the context menu for an eligible tier. Currently, this user interface only lets you create multi-text tiers which concatenate two other text tiers, but the process can be repeated to create longer concatenations. Alternatively, a multi-text tier with more than two parts can be created by directly editing the XAML file.

Segmenting a text tier

Either type of text tier can be segmented. This is the first step to take on raw data, in preparation for alignment, POS tagging, or assigning dependency structure.