JGloss User's Guide

Michael Koch

JGloss User's Guide

Michael Koch

Publication date 2012
Copyright © 2001-2012 Michael Koch

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

About JGloss

JGloss is an application for adding reading and translation annotations to words in a Japanese text document. This can be done automatically and manually. When a text document is first opened, words will be looked up in dictionary files and the first reading and translation (if any) is used to annotate the word. The user can then edit the annotations: choose among the readings and translations found in the dictionaries, enter own readings and translations, remove annotations and add new annotations. The document can be exported as plain text with annotations, HTML (with support for the *Ruby Annotation* [http://www.w3.org/TR/ruby/] XHTML specification) or LaTeX.

The application is designed as a translation aid for people learning Japanese. With some new document, you can first skim the text and change the annotations to match the likeliest meaning of the word. Then you can print/export the text with annotations and start working on the details of understanding the text without having to resort to a paper dictionary all the time.

JGloss is written in Java. It should work on any computer with support for the Java 2 Version 1.3/1.4 platform.

Credits

JGloss is written by Michael Koch (`<tensberg@gmx.net>`). It was inspired by Jim Breen's work, particularly the WWWJDIC [http://www.dgs.monash.edu.au/~jwb/wwwjdic.html] and XJDIC [http://www.csse.monash.edu.au/~jwb/xjdic/] . The kanji parser is based on ideas from the WWWJDIC [ftp://ftp.cc.monash.edu.au/pub/nihongo/www-jdict.ps.gz] . The character encoding detection uses code from Yasuhiro Tonooka's kcc (Kanji Code Converter). The French localization is contributed by Alexandre Beraud. Heinrich Künsting helped with the LaTeX export template design and wrote the LaTeX-CJK list template. Some of the file chooser icons are taken from the KDE project [http://artists.kde.org] .

License

JGloss is distributed under the terms of the GNU General Public License [http://www.gnu.org/licenses/licenses.html#TOCGPL] . It comes with ABSOLUTELY NO WARRANTY. This is free software, and you are welcome to redistribute it under certain conditions. Read the license for details.

Chapter 2. Setting up JGloss

What you need

JGloss is a Java application. To run it, you will need a Java implementation that conforms to the Java 2 Version 1.3 or 1.4 specifications, e. g. Sun's Java Runtime Environment 1.4 [http://java.sun.com/j2se/1.4/download.html] (JRE). Make sure that you install the international version.

Your computer system should already be configured to work with Japanese text. You must have a Japanese font installed. Having a Japanese input method installed is not absolutely necessary, but very useful. As a test, if your web browser can display Japanese text, it should be possible to set up Java do do the same. See also the section called “Making Java work with Japanese text” .

To use JGloss , you will need some dictionaries. Currently supported dictionary formats are:

EDICT [http://www.csse.monash.edu.au/~jwb/edict_doc.html]	EDICT dictionaries are Japanese to English word dictionaries. You can download them from the Monash Nihongo FTP Archive [http://ftp.cc.monash.edu.au/pub/nihongo/00INDEX.html#dic_fil] . Each dictionary also needs a JJDX index file. If no index file is found, it will be created automatically by JGloss and saved in the dictionary directory. Should the index file creation fail, for example because the directory is write-protected, the dictionary can't be used and an error message is displayed.
WadokuJT [http://www.bibiko.de/dlde.htm]	The WadokuJiten is an extensive Japanese-German dictionary. To use it, download the file `WadokuJT.zip` from this page [http://www.bibiko.de/dlen.htm] , unzip it and add the resulting file `WadokuJT.txt` to the list of dictionaries used by JGloss in the dictionary dialog. The WadokuJT dictionary format is only supported if you have Java Version 1.4 installed, otherwise JGloss will display a "not a dictionary file" error dialog. The WadokuJT dictionary is also available converted to EDICT format from the same location. The EDICT format however does not support the more detailed entry format of WadokuJT. WadokuJT dictionaries need a JJDX index file. If no index file is found, it will be created automatically by JGloss and saved in the dictionary directory. Should the index file creation fail, for example because the directory is write-protected, the dictionary can't be used and an error message is displayed.
KANJIDIC [http://ftp.cc.monash.edu.au/pub/nihongo/kanjidic_doc.html]	KANJIDIC dictionaries store information about individual kanji, among other things readings and translations. You can find KANJIDIC dictionaries at the same location as the EDICT dictionaries.
SKK [http://openlab.ring.gr.jp/skk/]	The SKK dictionaries provide a mapping from readings to words (no translations). They are usually used by Japanese input methods, but JGloss can use them to look up readings for kanji words.

Running JGloss

JGloss requires no installation. On a Windows system with the Java Runtime Environment installed, double-clicking the `jgloss.jar` file should start the application. To start JGloss from a shell, change to the directory which contains the JAR file and enter **`java -jar jgloss.jar`** . JGloss has some command line options: **`java -jar jgloss.jar [option] file ...`** .

`-h` , `--help` , `/?`	Shows a short help message with the list of options.
`-i` , `--createindex`	Creates JJDX index files for the dictionary files given on the command line. The index files will be saved in the current directory. JGloss tries to create a JJDX index file for a dictionary file automatically when none is found. If this fails, for example because a normal user has no write permissions for the dictionary directory, you can log in as a privileged user (e. g. administrator or root) and use this option to create the index files.
`-f` , `--format`	Prints the format of the dictionary files given on the command line.

Making Java work with Japanese text

If Japanese is set as your computer system's default language, JGloss should work without any further setup. If not, you should at least have a Japanese font installed. A Japanese input method, while not required, should also be installed. If you are using Sun's Java Runtime Environment for Windows, make sure that you have the international version installed (the smaller download only has English locale settings).

On Windows and MacOS X systems no font configuration should be neccessary. The first time JGloss is run, it tries to determine which font to use for displaying Japanese characters. If the Java default fonts don't contain Japanese characters, it tests if one of the fonts *Arial Unicode MS* , *MS UI Gothic* , *MS Mincho* , *MS Gothic* or *Osaka* is available and configures itself to use it. If none of the fonts work, no configuration is done and the default settings are kept. In this case Japanese characters will not display correctly. Read the section called “Style” for information on how to select fonts manually, or read on for information on how to change the Java default fonts.

Ths following paragraphs describe how to change the Java default font settings to use Japanese fonts. The descriptions apply to Sun's Java Runtime, other implementations may differ. Following these instructions should not be neccessary on Windows and MacOS X Systems since a custom font can be used, but have to be used on X11 Systems (blame the X font handling for that).

The easiest way to get Java to use Japanese fonts is to set the `user.language` parameter to `ja` : start JGloss as **`java -Duser.language=ja -jar jgloss.jar`** . This will make Java read the font definition from the `font.properties.ja` file of the JVM installation, and only works if the fonts used in the file are available. Unfortunately, at least on my Linux box, this will not make Java use the Japanese input method. On Unix systems, a second way is to set the `LANG` environment variable to `ja_JP` in a shell and then run JGloss from that shell. This will also make Java recognize an installed Japanese input method.

An alternative method, and one that allows more customization of what fonts to use, is to edit the `font.properties` files. The `font.properties` files are stored in the `lib` folder in the install directory of the Java Runtime Environment . They contain the definition of the default fonts used in Java dialogs and input elements. Which of the `font.properties` is used depends on the locale setting of the operating system, e. g. if you are running in a Japanese environment, the file `font.properties.ja` will be used. If there is no matching file for the current locale, the `font.properties` file is used. This is the case for most western locales. The file contains no

definition for Japanese fonts, but you can edit it and add these fonts. The easiest way to do this is to simply rename the file `font.properties.ja` to `font.properties` (but make a backup of the original file first). You can also edit the file to make it use different fonts.

On Windows 2000 and XP , the JRE supports the Japanese input method regardless of the system language settings, but you will have to use one of the methods mentioned above to get Java to use Japanese fonts. I had no luck so far in getting the JRE to use the input method under Windows 98 .

Copying Japanese text between JGloss and other applications on X11 systems only works if JGloss is running in a Japanese environment. To achieve this, open a shell, set the environment variable `LANG` to `ja_JP` and start JGloss from that shell. Under MacOSX, copying Japanese text is problematic as well. To get copy and paste working properly, move nihongo to the first position in the Language tab of the System preference->International panel.

Java and X11 fonts

The `font.properties.ja` file of the Linux version of Java is configured for fonts contained in the Japanese distribution of Red Hat 6.1 (and the international Red Hat 7.x with Japanese support installed). It seems that these fonts (wadalab-gothic.ttf and watanabe-mincho.ttf) are not contained in most other Linux distributions. To make matters worse, Java only seems to accept TrueType fonts and not the Japanese bitmap fonts commonly installed with X11. This means that you will either have to install the mentioned TrueType fonts or edit the properties file to use a different already installed Japanese TrueType font.

Do It Yourself

This section describes how to build JGloss from the sources. You do not need to read it if you just want to use JGloss and have downloaded the binary release (no `-src` ending).

To build the JGloss JAR file, you have to have the Java Development Kit 1.4 [http://java.sun.com/j2se/1.4/] from Sun (or something compatible) installed. To create the documentation files from `jgloss.docbook` you need the DocBook 4.1 DTD and various DocBook tools. Look for them at the OASIS site [http://www.oasis-open.org/docbook/] , Norman Walsh [http://nwalsh.com/docbook/] 's site and the DocBook tools page [http://sourceware.cygnus.com/docbook-tools/] , or on your favorite Linux distribution's CD-ROMs.

The sources come with a `Makefile` which automates the build process. The Makefile is written for a GNU/Linux system, but should work on other systems with the GNU tools or equivalent commands installed. Here is an (incomplete) list of make targets:

jgloss	Creates the JGloss JAR archive. This is the default make target.
jgloss-www	Builds the JGloss-WWW servlet. The components will be placed in the `jgloss-www` directory. This is an experimental servlet which rewrites Japanese web pages. Words in the page are looked up in the dictionaries and annotated with the lookup results. JavaScript is used to display the result in the browser. You can find documentation for the servlet in `src/www/index.html` .
doc	Generates the documentation files from `doc.src/jgloss.docbook`
gen-javadoc	Generates the javadoc documentation for the JGloss source files.
dist	Creates the source and binary distribution zip and tgz files.

If you can't use the Makefile, you can build the JGloss JAR archive by hand. Use **javac** to compile the `*.java` files in the `src` directory. Use **native2ascii** to convert every `.properties.in` file in the `src/resources` directory to a `.properties` file using the ASCII charset. The native charset of the file is specified in the first line of the input file. Then use **jar** to build a JAR archive, using the `MANIFEST.MF` file as manifest (see **jar** options). Make sure that the `resources` and `data` directories are included in the archive.

Chapter 3. Using JGloss

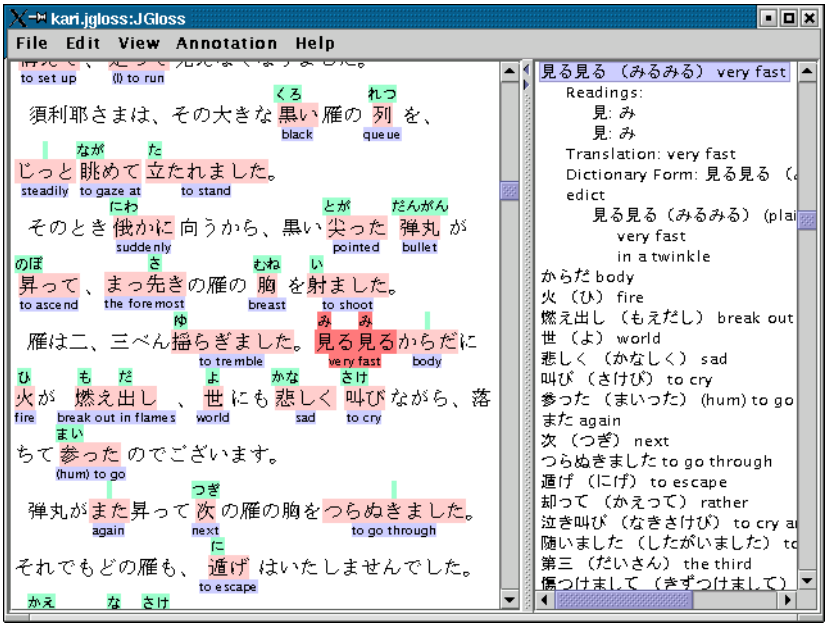
Quickstart

When you start JGloss for the first time, you should set the dictionary files which you want to use. Choose the Preferences item from the Edit menu, and select the Dictionaries tab in the newly opened window. Add the dictionaries you want to use and click OK . Note that dictionary entries will be chosen in the order of the dictionaries, so put your preferred dictionaries higher in the list.

Use the Import item from the File menu to open the import dialog. To import a file, select the File tab and enter the URL of the file you want to import. To choose a local file, click the Choose file button. Alternatively you can select the Text tab and paste the text to import in the area below.

Once you have made the text selection, click Import , and the text will be parsed and annotations will be added by looking up words in the dictionaries. After some time (this can take rather long for larger texts) the annotated document will be displayed in the window.

Figure 3.1. JGloss Main Window



The right part of the JGloss window shows the annotation editor (see the section called “The Annotation Editor”), which is used to manipulate the annotations in the document. The left part shows the annotated document (see the section called “The Document View”). An *annotated word* is highlighted in red, a *reading annotation* in green and a *translation annotation* in blue. The currently selected annotation is highlighted in deep red.

Figure 3.1, “JGloss Main Window” shows the JGloss window after some text is imported. The left part of the window shows the annotated document. An annotated word is shown in light red, with the reading annotation above and the translation annotation below the word. The right part of the window contains the annotation editor. For every annotated word, the word and its current reading and translation is shown. For the selected annotation at the top of the list, the annotation editor shows the current reading and translation, the dictionary form of the entry, and all readings and translations found in the dictionaries. You can use the annotation editor to change the reading and translation of a word, either by selecting one of the listed entries, or by selecting the current reading/translation item and entering the new text.

The heuristics used for generating the annotations are not perfect. For example, if you import a text using the edict dictionary file, 来る is assigned the reading きたる instead of くる and 一人 is assigned the reading いちにん instead of the more common ひとり . This happens because the application picks the first reading and translation found in the dictionary. Also, the algorithm used for verb/adjective deinflection can produce false results. What follows is that you should not trust the automatic annotations too much and that the document will require some editing.

To edit an annotation, select it in the annotation editor. The annotation editor will expand the annotation and show the dictionary lookup result, and the annotation will be highlighted red in the document view. Use the down key to select an entry with the appropriate reading or translation and hit the space bar. This will make the selected entry the reading/translation of the annotation. To manually edit the text of an annotation, select the Reading or Translation item and hit space to start editing.

When you have finished editing your document, you can export it to different formats. Select HTML from the Export entry in the File menu. In the file chooser you can select which type of annotations will be written. Select a file name and save the HTML file. If you are using a current version of the Microsoft Internet Explorer , which already (more or less) supports the *Ruby Annotation* XHTML specification, the ruby will be rendered above the annotated word by the browser. Translations will be shown in a popup window if you move the mouse over a word.

The Document View

The left part of the window shows the annotated document. Annotated words will have a colored background. You can change the colors or switch them off in the preferences dialog and toggle the display of reading and translation annotations in the View menu . If you left-click on an annotated word, the annotation will be selected in the annotation editor. A right-click will select the word and pop up a context menu with options for this annotation (see the section called “Annotation”). To look up the currently selected text in the dictionaries, select Word Lookup . To add an annotation, select some text and use Annotate Selection in the Edit menu (see the section called “Edit”).

The Annotation Editor

The annotation editor is used to edit the reading and translation annotations of annotated words in the document. It is displayed in the right half of the JGloss window (see Figure 3.1, “JGloss Main Window”). Each annotated word has an entry in the editor, ordered by its appearance in the document. The entries are displayed as a list, with the currently selected annotation expanded to show the dictionary search results. If an entry is selected, the annotation will be highlighted in the document view.

You can use the keyboard to navigate in the editor. The **cursor up/down** keys will move the selection by one item. To move to the next annotation press **N** , to move to the previous annotation press **P** . Pressing **space** will do something useful, depending on what kind of item is currently selected. Look at the table below for the full list of keyboard commands. Right-clicking an entry with the mouse will show a context menu for the chosen item (see the section called “Annotation”).

Each annotated word has one or more readings. If a word has no kanji characters, the reading will span the whole word. Otherwise, a reading annotation is added to every kanji substring of the word. The annotation editor shows the reading, or list of readings. To edit a reading, select the item and hit **space** , or click it twice and wait a moment. When you are done with editing, press **return** , and the document view will be updated with the new reading. The second item shows the current translation. It is edited in the same way as a reading.

The dictionary form item shows the dictionary form of the annotated word. This form will be used for a user dictionary entry, the annotation list and the LaTeX export format. The dictionary form will be set when you select the reading from one of the dictionary entries. To edit it manually, hit

space when the dictionary entry item is selected. This will show the word and reading part of the entry, which you can edit like the reading/translation items.

Some text documents you can find on the net are already annotated with readings by placing the reading in brackets after a word. These readings can be recognized by the text parser during importing and used as reading annotation. They are shown in the Reading from document items. Selecting one and hitting **space** will make it the reading of the annotation. If you are using a dictionary which provides readings but no translations, entries from this dictionary will also be displayed this way. The ChaSen parser will also provide readings for some words.

For every dictionary in which an entry for the annotated word is found, an item will be generated showing the dictionary name. Under the name, for each entry the dictionary word and reading will be shown, and under that one item for every translation. If you select a reading and hit **space** , the reading will be made the new reading annotation, and likewise for a translation. If a dictionary entry was found using verb/adjective de-inflection, the item will show the dictionary form of the word and in brackets the grammatical form in which the word appears in the document. Note that this algorithm may show wrong grammatical forms and therefore this feature should not always be trusted.

Table 3.1. Annotation Editor Shortcut Keys

Key	Function
n	Move to the next annotation
p	Move to the previous annotation
space	Depending on selection:hide/use reading/use translation/start editing
r	Remove annotation
d	Remove annotation and duplicates
e	Add annotated word to exclusion list
u	Add annotated word to the user dictionary
a	Use the reading and translation of the annotated word for all identical words in the document
h	Hide/show annotation
del	Clear reading/translation annotation text (only when a reading/translation node is selected)

Importing text documents

To create a new annotated document, select Import or Import Clipboard from the File menu. Import Clipboard will create a new document from the content of the system clipboard, it can be configured in the General Preferences dialog.

Selecting Import will show the Import dialog. In the text field you can enter an HTTP URL or the file name of a local document. The document should be plain text. There is some support for importing HTML documents, but it will only work for documents with simple layout. If you click the Choose File button, a file chooser dialog will pop up and let you select a local file. The character encoding of the file can be selected in the Character encoding ... popup menu. You can usually leave it on the <auto> setting, which will make JGloss auto-detect the encoding of the file. If the auto-detection fails for a file, you can select the encoding manually. You can choose the text parser for the automatic annotation in the Text Parser part of the dialog (see the section called “Text Parser selection”). Clicking the Import button will import the selected file.

If a reading is supplied for a annotated word, this reading will be used for the reading annotation, and the first translation found with this reading will be used for the translation annotation. Otherwise, the first reading and translation found in a dictionary will be used for the annotation. You can change the annotations later manually.

Note

The JGloss application is quite resource intensive. It can take a rather long time for the newly imported document to be displayed. You should consider splitting a long text in several shorter files before you import it.

Exporting annotated documents

JGloss supports the exporting of annotated documents in several formats, described below. You can select one of the formats from the Export submenu in the File menu.

A common option for all formats is the character encoding of the generated file. What format you use depends mainly on what application you want to use the exported file with. Modern web browsers should support all of the encodings. For other applications, if you are working on Windows, you should try Shift-JIS and on Unix EUC-JP. If the document or your annotations contain characters not in ASCII or the Japanese character set (e. g. umlauts), you should use UTF-8, which takes more disk space but can represent all characters.

With the write "hidden" annotations switch you can control how to treat annotations which are marked as *hidden* in the annotation editor. If it is unchecked, the annotations for hidden words will not be written to the generated document.

HTML

If you export the document in HTML format, the document can viewed in any web browser that supports display of Japanese characters. The document title set in JGloss will be used as the title of the HTML document. The markup defined in the *Ruby Annotation* specification is used to embed the annotations, browsers which support it will render the annotations above/below the annotated words.

If the backwards compatible switch is not selected, the generated HTML will use all options of the *Ruby Annotation* to embed both readings and translations. The generated document will only render correctly on browsers which fully support the specification (I don't know of a browser which currently does). Otherwise, the subset of the specification which the Internet Explorer 5.5 understands will be used for reading annotations. Translations are shown using JavaScript in a floating window and the status bar of the browser when the user moves the mouse over an annotated word. In Netscape 6/Mozilla and other browsers which don't support ruby annotations but support the neccessary JavaScript functions, reading annotations will be shown in a floating window above the annotated word and in the status bar. In other browsers, the reading annotations will be shown in the document after the annotated word, the translations are not available.

Plain Text

The plain text export function will generate a text document similar to the originally imported document. Annotations will be written after the annotated word, enclosed in brackets.

LaTeX

The LaTeX export function will generate a text document in LaTeX format. There are several document style variants you can choose from by selecting the corresponding template from the template chooser. A description of the currently selected template is shown below the template chooser. You can add new or modified templates by selecting Add... from the template chooser menu. A file selector dialog will be shown where you can select the new template file. See Appendix B, *LaTeX export template format* for the template format description. You can choose the document font size from the Font size menu.

Standard LaTeX can't handle Japanese documents. There are two ways of adding support for Japanese: using pLaTeX, a LaTeX variant modified to handle Japanese, or the CJK macro package,

which adds support for far eastern scripts to standard LaTeX. JGloss has templates for generating input files for both variants. Choose whatever option is easier for you to install and set up.

pTeX/pLaTeX is a modification of TeX/LaTeX, which can handle Japanese typesetting. To process a Japanese LaTeX document, you have to run the platex program instead of the normal latex command. pTeX is part of a standard TeTeX installation on Red Hat Linux, and may be available with other TeX distributions. You will also need some Japanese fonts for displaying the generated file. Ghostscript can supply fonts to TeX, Japanese fonts for Ghostscript are available at http://www.enel.ucalgary.ca/People/far/howto/gs-ttf.html [http://www.enel.ucalgary.ca/People/far/howto/gs-ttf.html] . dvips will also have to be configured to know about the Japanese fonts if you want to convert the generated dvi file to Postscript. To display ruby for reading anntations, the macro `ruby-annotation.sty` (in the folder `latex` of the JGloss distribution) must be installed.

CJK is a macro package for adding Chinese/Japanese/Korean script support to standard LaTeX. In order to use it, you will have to download and install the macro package and the corresponding font package. Once the package is installed, you can process the files generated from JGloss with the LaTeX-CJK templates with the commands latex or pdflatex .

Annotation List

The annotation list export function will write a text file listing all annotations in the document. The dictionary form and the selected translation of the annotated word is used. Annotations will only be written once, duplicate entries will be skipped. You can use the generated text as a basis for a vocabulary list.

XML

Export the JGloss document as XML document. Reading and translation annotations will be embedded in special tags. You can transform the XML document to other formats using XSL/XSLT stylesheets.

The User Dictionary

The user dictionary is a special dictionary to which you can add words from within the annotation editor. The user dictionary will always be shown in the list of dictionaries and cannot be deleted. You can use the user dictionary to add words which don't appear in any of the other dictionaries. Another use of the user dictionary is the selection of a preferred reading/translation for a word with several readings/translations. Since during a import the first reading/translation found for a word is choosen and the user dictionary is first in the list, entries in the user dictionary will be preferred.

To add an entry to the user dictionary, select an annotation in the annotation editor and choose Add to Dictionary from the Annotation menu. The dictionary form and current translation will be used for the new entry. There currently is no way to delete or edit an entry from within JGloss, but you can edit the dictionary file. The dictionary is saved in EDICT format to the file `.jgloss-userdictionary` in the user's home directory.

The Word Lookup Dialog

You can look up words in the dictionaries from the word lookup dialog. Enter a Japanese or English word in the Enter Expression text field and click the Search button. The results will be displayed in the lower area of the dialog window. The part of the entry which matched the expression will be highlighted blue for each result line. Hitting the Clear will delete the current search expression. If you select the arrow to the right of the text field, a popup menu listing previous search expressions will be shown.

You can select the search mode from the Search Options part of the dialog. Exact matches will only return entries where the entered expression is identical to the word, reading or one of the

translations of the entry. Starts with Expression and Ends with Expression will find entries where the word, reading or one of the translations have the search expression at the beginning/end. Any Match will return entries where the search expression appears anywhere within the result. Note that the dictionary formats may not support all of the search options. For example, the EDICT implementation does not do "ends with" searches for readings so you might not get all possible matches if you use this search mode.

The Best Match search option will automatically adapt the search mode such that the least number of results are found. If a Best Match search is performed, Exact matches will be tried first and if no results are found, the other search options are tried one after the other until a match is found or all options have been tried.

The Use Deinflection checkbox enables the use of the verb/adjective deinflection algorithm for the search expression. The algorithm tries to derive the base form of the search expression by looking at the hiragana ending. This might find entries for inflected word forms, but may also yield confusing results, since the algorithm does only work for some verb forms.

You can limit the search to a single dictionary by selecting the Search Dictionary radio button and selecting the dictionary from the popup menu to the right. If the Search all Dictionaries radio button is selected, all dictionaries made available to JGloss in the preferences will be searched.

If you use the JGloss word lookup dialog to look up words copied from other programs, you might want to enable the automatically search clipboard content option in the dialog window. If the option is selected, every time the JGloss window is brought to the front, the content of the clipboard is entered in the Search Expression field and searched using the current search settings. This way you can look up words in other programs by simply selecting the word, copying it to the clipboard and bringing the word lookup dialog to the foreground.

Text Parser selection

JGloss can use two different parsers for the automatic annotation of Japanese text, the Kanji parser and the ChaSen parser. Select the parser by clicking the respective radio button. The ChaSen parser will only be available if the chasen program is installed (see below).

If the annotate first occurrence option is selected, each word in a document is only annotated the first time it appears. This decreases the RAM usage and the time it takes to display the document. The Guess paragraph breaks option controls how line breaks in the imported document are converted to paragraph breaks. If the option is selected, JGloss tries to determine if a line break in the imported document signifies the end of a paragraph or if it is used for formatting reasons only. In the second case, it will be ignored.

Some documents you can find on the internet already have reading annotations added to kanji words in the form of some hiragana enclosed in brackets after the kanji word. The parsers can generate reading annotation entries for these words. You can select the brackets used in the document for reading annotations with the Brackets used... box. If the document contains no reading annotations, you can select none or simply ignore this setting.

The Kanji parser

The Kanji parser is built into JGloss. A simple heuristic is used for choosing words to annotate: for a sequence of katakana characters, the whole sequence is treated as one word and looked up. For a sequence of kanji characters followed by hiragana characters, the algorithm first looks for possible inflected forms in the hiragana string and will try to find words that consist of the kanji word and the dictionary form of the inflected forms that appear in the hiragana string. If no match is found, only the kanji part is looked up. If still no match is found in any of the dictionaries, prefixes of the kanji word will be tried and if this leads to a match the process will be repeated with the remainder. A consequence of this method is that hiragana words will never be annotated automatically even if they are in the dictionaries.

The ChaSen parser

The ChaSen parser uses the ChaSen morphological analysis program to decompose Japanese text in words and to derive the base form of inflected words. It is slower than the Kanji parser, but will annotate hiragana words as well as kanji and katakana words. It also does a better job of deinflecting verbs and adjectives. You can download ChaSen from the ChaSen homepage [http://chasen.aist-nara.ac.jp/] . After installation, you have to set the path to the chasen executable in the preferences dialog . It usually is `/usr/local/bin/chasen` under Unix or `c:\Program Files\chasen21\chasen.exe` under Windows.

The ChaSen program is used to generate a list of words with their reading and base form from the parsed text. The words will be looked up in the dictionaries, and if an entry is found, an annotation will be generated. If no dictionary entry is found and the word is not inflected, kanji substrings will be tried. A reading annotation with the reading output by ChaSen is also added if the reading returned by ChaSen is different from the first reading found in the dictionaries. Since the ChaSen program uses its own set of dictionaries to detect words, it might not recognize words which are found in the dictionaries used by JGloss but not in the ChaSen dictionaries.

Note

If you use JRE 1.3 on Windows systems, an empty console window is visible while ChaSen is running, unless JGloss is started from the command line. Under Windows 98/ME, the ChaSen program does not terminate automatically. You will have to close the ChaSen window manually after the text is parsed. This does not happen with the JRE 1.4.

The Preferences Dialog

The preferences dialog contains four panels, one with general preferences, one for setting the visual appearance of the annotated document, one for managing the dictionaries and one for editing the list of words excluded from annotation. You can access the dialog by selecting Preferences from the Edit menu.

General

You can select the window opened on startup with the Open empty JGloss document and Open Word Lookup dialog radio buttons. The function of the left mouse button is changed in the Left-clicking an annotated word section of the dialog. If you select Show annotation tooltip , a popup window with the annotation text will be shown if you click on a word. Otherwise, the annotation will be selected in the annotation editor. The text parser used when importing the clipboard content can be selected in the Import Clipboard Parser section of the dialog. See the section called “Text Parser selection” for details. You can set the location of the ChaSen parser program in the ChaSen executable text field. It usually is `/usr/local/bin/chasen` under Unix or `c:\Program Files\chasen21\chasen.exe` under Windows. If the program cannot be found, the ChaSen parser will not be available.

Style

The Japanese User Interface Font lets you choose the font used in the display elements of JGloss. If the default Java fonts don't contain Japanese characters or you want to use a different font, select the Use this font radio button and choose a font from the list. Note that not all fonts in the list can display Japanese characters.

The other font selection options determine the fonts used in the word lookup result list and in the document view. Select a font by using the popup menu. The font size can be selected from the popup menu to the right, or you can enter the font size manually if it is not in the list. You can select a different background color by clicking on the button with the color label, or disable the use of a background color by unchecking the set background color checkbox. The Highlight color ...

button lets you select the color which is used for highlighting the currently selected annotation in the document view.

Dictionaries

In this dialog you can set the dictionaries which are used when importing a text or adding annotations to a document manually. Click on Add dictionary file to add one or more dictionaries to the list. JGloss currently supports dictionaries in EDICT, KANJIDIC and SKK format. Select a dictionary from the list and click Remove entry to remove it.

Since the automatic annotation process will search the dictionaries in the order in which they are displayed in the list and will use the first entry found as default annotation, you should put your preferred dictionary in the front. Select one of the dictionaries and click Move entry up or Move entry down do move it to the desired position.

Exclusions

This dialog lets you manage the list of words excluded from automatic annotation. When you import a document, no annotations will be added for the words in this list. You can export and import the list by using the correspondent buttons. The format of the list is simply one word per line.

JGloss menus

File

Import	Creates a new document by importing a text file, annotating it on the fly. See the section called “Importing text documents” for details.
Import Clipboard	Creates a new document by importing the content of the clipboard. See the section called “Importing text documents” for details. This item will only be enabled if the clipboard currently contains some text.
Open	Open a JGloss annotated document.

Note

The application can take rather long to display a document for the first time after it is loaded, especially for larger documents.

Open Recent	Open a JGloss annotated document by selecting it from the list of recently opened files.
Save	Saves the current annotated document in the JGloss file format. If the file name has not yet been determined, a file chooser dialog will be shown.

Note

The JGloss file format is basically HTML with a few custom tags. You can open the file in a HTML editor to edit the text of the document, but be careful and don't change any of the special tags and attributes.

Save As	Saves the current document in the JGloss file format under a new name.
Export	The entries in this submenu let you export the current annotated document in several formats. See the section called “Exporting annotated documents” for details.

Print	Print the annotated document. The current settings for font sizes and colors will be used in the printed document. On Linux you can also use this option to generate a Postscript version of the document by selecting the print to file option in the print dialog. If you want to generate a Postscript file on Windows, you have to install a Postscript printer driver and use the print to file option of this driver.
Close	Closes the document window. If the document has unsaved changes, a warning dialog will be shown. If all document windows and the word lookup dialog are closed, the JGloss application will quit.

Edit

Cut/Copy/Paste	These have the standard functionality. Note that because the document view is not editable, the Cut and Paste items are always disabled.
Find	Search for some text in the document. The search will always start at the beginning of the document. If the text found is part of an annotation, the annotation will be selected.
Find Again	Searches for the next occurrence of the text from the last search. The search will start at the position where the last occurrence was found.
Annotate Selection	This will let you annotate to the currently selected text. The Word Lookup dialog will be opened and shows the selected word. You can edit the word and search it in the dictionaries. If you click the Annotate button, the word in the document will be annotated with the current search result. Click Cancel to not annotate the word. Annotations cannot overlap. If the selected text already contains annotations, these annotations will be deleted. Also, an annotation cannot span paragraphs. If text from more than one paragraph is selected, the annotation will end at the end of the first paragraph.
Document Title	Brings up a dialog which lets you set the title of the document. The title is used when the document is exported in HTML or LaTeX format.
Word Lookup	Selecting this item will show the Word Lookup dialog (see the section called “The Word Lookup Dialog”). If some text is selected in the current document, this text will be automatically searched using the current dialog settings.
Preferences	Selecting this item will show the preferences dialog (see the section called “The Preferences Dialog”).

View

Compact View	Enabling compact view will make each annotation take only as much space in the display as the annotated word needs, with the annotation text overlapping the surrounding text. This may look nicer, but will cause long annotations to overlap each other. You should try this with the display of translation annotations switched off.
Show Readings	This item toggles the display of reading annotations in the document view.
Show Translations	This item toggles the display of translation annotations in the document view.

Show Annotation Tooltips	If this item is selected, a window will pop up if you move the mouse over an annotation in the document view, showing all dictionary entries of this annotation.

Annotation

The items in this entry manipulate the annotations of the entry currently selected in the annotation editor. The items also appear in the context menu for annotation editor items and annotated words in the document view.

Use this Reading	Makes the currently selected item the reading of the annotation. The menu entry is only selectable if the item is a reading item or a dictionary entry word/reading item.
Use this Translation	Makes the currently selected item the translation of the annotation. The menu entry is only selectable if the item is a dictionary entry translation item.
Hide Annotation	Toggles the visibility of the reading and translation of this annotation. If an annotation is hidden, the document view will only display the word, but not the reading and annotation. This also influences how this annotation is treated when the document is exported (see the section called “Exporting annotated documents”).
Remove Annotation	Removes the annotation from the currently selected word. It will be removed from the annotation editor, and the word will be changed to normal text in the document view.
Remove Anno. & Duplicates	Removes the annotation from the currently selected word, and all duplicate annotations. An annotation is a duplicate if it has the same word, reading and translation as the selected entry.
Use for All	Changes all annotations with an annotated word identical to the currently selected word to use the same reading, translation and dictionary form.
Add to Exclusions	Adds the word of the currently selected annotation to the list of words excluded from automatic annotation (see the section called “Exclusions”).
Add to Dictionary	Adds the word of the currently selected annotation to the user dictionary (see the section called “The User Dictionary”).
Hide all Duplicates	This will hide all annotations which are duplicates of a previous annotation. An annotation is a duplicate if it has the same word, reading and translation as a previous entry. Note that if an annotation is edited, duplicate annotations will not be changed. This command will work on all annotations, not just the currently selected item.
Show all Annotations	This item will remove the hidden status from all annotations.

Appendix A. Links

- The JGloss Homepage [http://jgloss.sourceforge.net/] is the place where you can find information about JGloss and download the latest release.
- The JGloss project page [http://sourceforge.net/projects/jgloss/] at SourceForge [http://sourceforge.net/] has information about JGloss development.
- Jim Breen's Japanese Page [http://www.dgs.monash.edu.au/~jwb/Japanese.html] : here you will find lots of interesting links, among them links to dictionaries in EDICT and KANJIDIC format.
- A direct link to the EDICT and KANJIDIC dictionaries [http://ftp.cc.monash.edu.au/pub/nihongo/00INDEX.html#dic_fil] .
- The ChaSen [http://chasen.aist-nara.ac.jp/] morphological analysis tool can be used by JGloss to parse Japanese text.
- The Blue Sky Collection [http://www.aozora.gr.jp/] of electronic books.
- Sun's Java download page [http://java.sun.com/j2se/download.html] .

Appendix B. LaTeX export template format

The documents generated by the LaTeX export function can be customized by creating template files. A template file is a text file containing special sections which are replaced by JGloss with the document content. Per convention, template files use the `.tmpl` file ending. Example template files can be found in the `latex/templates` folder of the JGloss distribution.

The first line in the template file is the *encoding declaration* .

```
%encoding: encoding_name
```

Valid encoding names are all encodings supported by the Java Runtime which is used. (Not all encodings can represent Japanese characters or are supported by LaTeX-CJK/pLaTeX). Some useful encodings are `EUC-JP` , `SJIS` or `UTF-8` . The encoding declaration determines both the character encoding of the template file itself and the encoding used for the file generated from the template. The encoding declaration line will not be copied to the generated file.

The second line gives a short description of the template. This line is optional.

```
%short-description: Text describing the template
```

The description will be displayed in the export dialog template chooser. The description line will not be copied to the generated file.

A longer description of the template can be given in a description block.

```
%description
Template description line 1
Template description line 2
...
%end description
```

The description can appear once anywhere in the template, but it is suggested to place it after the short description. The text will be displayed in the export dialog below the template chooser when the template is selected. `%` and whitespace are stripped from the beginning of a description line. Normal line breaks are ignored, to force a line break, insert an empty line in the description block. The description block will not be copied to the generated file.

The rest of the template is made of normal text, *variables* and *insertion sections* . Normal text is copied unchanged from the template file to the generated file. Variables have the form *%variable-name%* . In the generated file, variables are replaced by their values. A variable may appear anywhere in normal text and in *pattern definitions* (see below). In all text generated through variable substitution and for insertion sections, LaTeX special characters are properly escaped.

In the generated file, an *insertion section* in the template file is replaced by the document content. Insertion sections contain *pattern definitions* , which control the text generated for document elements like annotations or paragraphs. An insertion section has the following form:

```
%insertion-section-name
%pattern-name1: pattern definition text
%pattern-name2: pattern definition text
...
%end insertion-section-name
```

The two insertion section types are `document-text` , which inserts the text of the document with reading and translation annotations, and `annotation-list` , which inserts the list of annotations.

The *pattern definitions* control the text generation in insertion sections. Pattern definitions can be given for readings, translations or paragraphs. The pattern definition text is inserted for every occurrence of a pattern in the insertion section, and special variables have different values for every application of the pattern. To support insertion of line breaks and other special characters, the pattern definition text may contain C-string-style escape sequences: `\n` inserts a line break, `\t` a tabulator and `\\` a single backslash.

For example, a pattern for occurrences of reading annotations may be defined in a `document-text` insertion section.

```
%document-text
%reading: \\ruby{%word%}{%reading%}
... more pattern definitions ...
%end document text
```

When the document text is inserted in the generated file, for every occurrence of a reading annotation in the document, the variable `%word%` is set to the annotated word and `%reading%` is set to the reading annotation. The pattern is then applied, and the pattern text with variables substituted is written to the generated file.

Table B.1. Pattern definitions

Name	Function	Defined in	Default
`reading`	Applied to every reading annotation in the document.	`document-text`	(none)
`translation`	Applied to every translation annotation in the document.	`document-text` , `annotation-list`	(none)
`line-break`	Applied to every line break in the document which is not a paragraph break.	`document-text`	`\\\\\n`
`paragraph-start`	Inserted at the start of every paragraph. Paragraphs in a JGloss document are assumed to be seperated by one or more empty lines.	`document-text` , `annotation-list`	(none)
`paragraph-end`	Inserted at the end of every paragraph. Paragraphs in a JGloss document are assumed to be seperated by one or more empty lines.	`document-text` , `annotation-list`	`\n\n`

Table B.2. Variables

Name	Description	Defined in pattern
`document-filename`	Replaced by the file name of the exported JGloss document.	(global)
`generation-time`	Replaced by the current time and date.	(global)
`document-title`	Replaced by title of the JGloss document.	(global)
`font-size`	Replaced by the font size selected in the export dialog.	(global)

Name	Description	Defined in pattern
`longest-word`	Replaced by the longest annotated Japanese word (dictionary form) in the JGloss document.	(global)
`longest-reading`	Replaced by the longest reading (dictionary form) in the JGloss document.	(global)
`word`	Replaced by the annotated word.	`reading`
`reading`	Replaced by the reading annotation of an annotated word.	`reading`
`dictionary-word`	Replaced by the dictionary form of an annotated word.	`translation`
`dictionary-reading`	Replaced by the dictionary form of the reading of an annotated word.	`translation`
`translation`	Replaced by the translation text of an annotated word.	`translation`
`paragraph-number`	Replaced by the current paragraph number.	`paragraph-start`