lvg Reference Guide

Kevin M. Rosenberg, M.D.

kevin@rosenberg.net
Albuquerque
New Mexico

lvg Reference Guide

by Kevin M. Rosenberg, M.D.

Table of Contents

Introduction to *lvg*

This reference guide describes *lvg*, a Lisp interface to the National Library of Medicine's (NLM's) Lexical Variant Generation library.

Background

The NLM Lexical Variant Generation library (NLMLVG) is a part of the *Unified Medical Language System* (http://umlsinfo.nlm.nih.gov) project. An important use of this library is to generate normalized terms for word indices. For example, the input terms *swim*, *swam*, and *swum* all normalize to the term *swim*.

NLMLVG contains a C Application Programming Interface (API) to this library. This interface library, *lvg*, provides a Common Lisp interface to the NLMLVG API.

Copyright

Recommended Configuration

The primary development platform for *lvg* consists of:

- The NLM Lexical Variant Generation library. (Required)
- The Berkeley B-tree database library. (Recommended by the NLM)
- Allegro Common Lisp compiler. (Other Common Lisp systems can be used by modifying the Foreign Function Interface declarations.)
- UNIX or LINUX operating system. (Microsoft Windows can be used by compiling the shared object libraries as dynamic link libraries).

Sample Usage

The installation of the *lvg* is described in the *appendix*. Once installed, the package can be loaded into Lisp.

```
cl-user(1): (load "lvg.cl")
; Loading /home/kevin/lvg/lvg.cl
;   Foreign loading /home/kevin/lvg/lvg.so.
t
cl-user(2): (lvg:set-process-options-to-default)
cl-user(3): (lvg:process-terms "leaves")
("leaf" "leave")
cl-user(4): (lvg:process-terms "swim swam swum")
("swim swim swim")
cl-user(5): (lvg:get-process-options)
"-fN"
cl-user(6): (lvg:set-process-options "-fi")   ;; generate all inflectional variants
cl-user(7): (lvg:process-terms "swim")
("swims" "swim" "swims" "swum" "swam" "swimming" "swim")
cl-user(8): (lvg:shutdown)
cl-user(9): (exit)
```

I. lvg Programming Reference

Overview

The *lvg* interface library is contained in the Common Lisp package `lvg`. This section documents the functions exported by the `lvg` package.

The functions documented in this section are:

- `process-terms`
- `set-process-options`
- `set-process-options-to-default`
- `get-process-options`
- `shutdown`

PROCESS-TERMS

Name

`PROCESS-TERMS` — Return the lexical variants of a string of terms.

Function

Syntax

```
process-terms
     terms => lv-list
```

Arguments and Values

`terms`

The input terms for which lexical variants are generated.

lv-list

A list of string lexical variants for the input *`terms`*.

Description

This function takes a string of input terms and returns a list of lexical variants. Th generation of these lexical variants is determined by the current process options as set by `set-process-options`.

Examples

```
(process-terms "leaves")
=> ("leave" "leaf")

(process-terms "left atriums")
```

```
=> ("atrium left" "atrium leave")
```

Side Effects

Transparently initializes the NLMLVG library the first time that this function is called.

Affected by

The current NLMLVG process options.

Exceptional Situations

If the NLMLVG shared object library was not able to be loaded, this function returns a value of `nil`.

See Also

```
set-process-options
set-process-options-to-default
```

Notes

The default NLMLVG process options performs the normalization used by the UMLS normalized word index `MRXNW.ENG`.

SHUTDOWN

Name

`SHUTDOWN` — Closes and frees resources for the *lvg* package.

Function

Syntax

```
shutdown
```

Description

This function closes and frees resources for the *lvg* package and the NLMLVG library.

Examples

```
(shutdown)
```

Side Effects

Closes the NLMLVG library freeing memory and closing files. Frees memory in the *lvg* C interface library.

Exceptional Situations

After the library is closed, it may not be re-opened. The NLMLVG library does not support re-initialization after it has been shutdown.

SET-PROCESS-OPTIONS

Name

`SET-PROCESS-OPTIONS` — Sets the NLMLVG process options.

Function

Syntax

```
set-process-options
      option-string
```

Arguments and Values

`option-string`

A NLMLVG process option string.

Description

This function sets the current NLMLVG process options. The process options governs the generation of lexical variants by the NLMLVG library. The documentation of valid values for the process option string is specified in the NLMLVG library's documentation.

Examples

```
(set-process-options "-fN")
```

Side Effects

The value set by this command determines the generation of lexical variants by the `process-terms` function.

Notes

If the process option string passed to this function is invalid, then the NLMLVG library either will not return any, or will return incorrect, lexical variants.

SET-PROCESS-OPTIONS-TO-DEFAULT

Name

`SET-PROCESS-OPTIONS-TO-DEFAULT` — Sets the NLMLVG process options to the default value.

Function

Syntax

```
set-process-options-to-default
```

Description

This function sets the NLMVLG library process options to the default value. The default value performs the normalization used by the UMLS normalized word index file.

Side Effects

Sets the NLMLVG process options string to the value "-fN".

GET-PROCESS-OPTIONS

Name

`GET-PROCESS-OPTIONS` — Returns the current NLMLVG process options string.

Function

Syntax

```
get-process-options
        => option-string
```

Arguments and Values

option-string

The current NLMLVG process options string

Description

This function returns the current option string used by the NLMLVG library. This string controls the generation of lexical variants.

This function can be used by an application to save the current process options before the application modifies the process options.

Affected by

```
set-process-options
set-process-options-to-default
```

Appendix A. Installation

The NLM Lexical Variant Generation Library

Download the NLMLVG Package

NLM's lexical variant generation tools can be downloaded from the UMLS *Knowledge Source Server* (http://umlsks3.nlm.nih.gov/KSS/LVG_01/). *lvg* has been successfully tested with NLMLVG version `1.84h6`.

Download and Compile the Berkeley B-tree library

The NLM strongly recommends using the Berkeley B-tree library with the NLMLVG library. The library can be download it from the *Sleepycat* (http://www.sleepycat.com) web site. Version `2.2.7` of the library has been found to work well with the NLMLVG source code.

By default, compiling this library produces a statically linked library. However, in order to be loaded into a Lisp system, a shared object version of this library must be built. Instructions for the creation of a shared object library are included in the documentation directory of this library.

Compile the Shared Object Library

NLMLVG includes its own documentation on building and installing the package. The package that you download may or may not include precompiled binaries and database. If your package does not include these precompiled components, then you'll want to build the binary programs and database.

Version `1.86h6` of the NLMLVG library is preconfigured to be built on Solaris. To compile the library on a system such as Linux or FreeBSD which uses the GCC compiler, the NLMLVG `Makefile` must be edited. After editing the `Makefile`, the binary programs and databases can be built with the command:

```
make all
```

After building the binary programs and database, the shared library can be built with the command:

```
make sharedLibs
```

Install Into an Accessible Directory

After building the shared library, the shared object library and C header files need to be installed into a location accessible to *lvg*. By default, the *lvg* expects the NLMLVG shared object library to be in the directory `/usr/local/lv/lib` and the NLMLVG C header files to be in the directory `/usr/local/lv/include`.

Set the NLS Environment Variable

The shell environment variable NLS must be set for the NLMLVG library to operate correctly.

If you use the `bash` shell and you have installed the NLMLVG library in `/usr/local/lv`, you may add the following code to your `.bashrc`. With this code, each time you login to the shell the NLS shell environment variable will be defined and the binary program directory will be added to your path.

```
if [ -d /usr/local/lv ]; then
  export NLS=/usr/local/lv
  export PATH=${PATH}:${NLS}/bin
fi
```

Test Using the NLMLVG Programs

You should test the NLMLVG binary applications. Unless these applications work correctly, then the Lisp interface to the library will not work correctly.

As a quick test, if you give this command at the shell prompt,

```
echo "leaves" | norm
```

Then you should get this output,

```
leaves|leaf
leaves|leave
```

And if you give this command,

```
echo "swim" |lvg -fi
```

Then you should get this output,

```
swim|swims|128|8|i|1|
swim|swim|128|1|i|1|
swim|swims|1024|128|i|1|
swim|swum|1024|64|i|1|
swim|swam|1024|32|i|1|
swim|swimming|1024|16|i|1|
swim|swim|1024|1|i|1|
```

The *lvg* Library

Download the *lvg* Lisp Distriubtion

The package can be downloaded from the *UMLisp* (http://www.umls.org) web site.

Customize the Makefile For Your System

The `Makfile` included with *lvg* is pre-configured for use with the GCC compiler and assumes that the NLMLVG library is installed in `/usr/local/lv`. If these assumptions don't apply to your system, you'll need to edit the `Makefile`. Recommended settings for the Solaris operating system are included in the `Makefile`.

Compile the *lvg* C Interface Library

lvg includes a C source file, `lvg.c`. to interface to the NLMLVG library. The functions in this file provide a simplifying layer to the NLMLVG library. To be loaded into Lisp, it must be compiled into a shared object library that is linked to the NLMLVG shared object library. This can be accomplished with command:

```
make lib
```

Build the Documentation (Optional)

The documentation file for the *lvg* package, `doc/lvg.sgml`, is written in the *DocBook* (http://www.docbook.org) markup language. Included in the *lvg* distribution are pre-compiled Postscript, Portable Document Format (PDF) and HTML documentation. If you wish to build your own formatted documentation files, you’ll need to install and use several applications and data files to process the SGML DocBook source.

Load the *lvg* package into Lisp

After compiling the shared library, you can start Lisp and load the Lisp `lvg.cl` file. Before starting Lisp, make sure that you have the NLS shell environment variable set. In Lisp, load the *lvg* package and shared library file with the command:

```
(load "lvg.cl")
```

Th `lvg.cl` file will automatically load the `lvg.so` shared library as well as define the `lvg` Common Lisp package.