Dolmen Documentation

Release 2.0.0

Julien Eychenne

04 May 2018

CONTENTS

CHAPTER ONE

OVERVIEW

Dolmen is a free, open-source software toolbox for the analysis of annotated speech. It offers a user-friendly interface to manage, annotate and query language corpora. It is particularly well suited for dealing with time-aligned data. The main features it offers are:

- Project management: organize files into projects and manage versions.
- Extensible metadata: files can be annotated with properties, which allow you to sort and organize your data.
- Interaction with Praat: Dolmen can read TextGrid files and open files directly in Praat.
- Powerful search engine: build and save complex queries; search patterns across tiers.
- Standard-based: Dolmen files are encoded in XML and Unicode.
- Scripting engine: Dolmen can be extended with plugins written in Lua and JSON.

Dolmen runs on all major platforms (Windows, Mac OS X and GNU/Linux) and is freely available under the terms of the GNU General Public License version 3 (GPL). The latest version of Dolmen can be downloaded from http://www.dolmen-ling.org. If you encounter any problem or bug, please write to jeychenne@gmail.com.

CHAPTER

TWO

DOWNLOAD

2.1 Dolmen 2 (current version)

Download version 2.0.0 (30/03/2018):

This version is recommended for all users.

- Windows 7 and later: dolmen_setup.exe
- MacOS 10.7 and later: dolmen.dmg
- Linux (Ubuntu 16.04 64-bit): dolmen-2.0.0.tar.bz2
- source code: available on GitHub

2.2 Dolmen 1.3 (legacy version)

Dolmen 1.3 can be downloaded from here.

2.3 Manual

Dolmen's documentation is available as a PDF file.

2.4 PFC plugin

The plugin for the PFC corpus which accompanies the book Varieties of Spoken French can be downloaded here. The version of Dolmen which is available on the book's companion website is the legacy version: it is recommended that you upgrade to the current version.

CHAPTER

THREE

TOPICS

3.1 Installation

3.1.1 Windows

On Windows, Dolmen is provided as a self-contained installer file. Simply double-click on 'dolmen_setup.exe' and follow the instructions.

The procedure will install Dolmen in your `Program Files` directory and will create a shortcut in the start menu (and optionally on the desktop).

If you wish to be able to open files in Praat from Dolmen, you will need to install Praat in Dolmen's installation directory, which should be either `C:\Program Files (x86)\Dolmen2\Tools` or `C:\Program Files\Dolmen2\Tools`, depending on your system. Alternatively, you can modify Praat's default path with the preference editor.

3.1.2 Mac OS

On Mac OS, Dolmen is provided as a standard DMG image disk. Mount the image by double-clicking on it and drag the application `Dolmen` into your `Applications` folder. If you want Dolmen to be able to interact with Praat, you will need to install it in the `Applications` folder too.

Currently, only Mac OS 10.7 (Snow Leopard) and later are "officially" supported. It does not work on earlier versions.

3.1.3 Linux (Debian/Ubuntu)

The official executable that is provided on the website is built on Debian 9 and is available for 64-bit architectures.

Since the program is available as a dynamically-linked executable, first make sure that the needed dependencies are installed (asound, libsndfile, speexdsp, Qt 5 and GTK 2). Most of these packages should already be installed, but you can issue the following command in a terminal to make sure they are:

```
sudo apt-get install libasound2 libsndfile1 libspeexdsp1 libgtk2.0-0 libqwt-qt5-6␣
↪libqt5sql5-sqlite
```

Next, assuming that you downloaded the archive in your `Downloads` directory, type the following commands in a terminal (replacing `XX` by the appropriate version number):

```
cd opt
sudo tar xvjpf ~/Downloads/dolmen-XX-linux.tar.bz2
sudo ln -s /opt/dolmen/bin/dolmen /usr/local/bin/
```

You can now run Dolmen by simply typing `dolmen &` from a terminal window.

If you get an error about a missing SQL plugin, try to add the following line to your `.bashrc` configuration file:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib/x86_64-linux-gnu/qt5/plugins/
↪sqldrivers
```

3.1.4 Compiling from source

You need to install the development packages for QT 5.3 or greater (including the sqlite plugin), GTK 2, ALSA (libasound2), libspeexdsp and libsndfile. You also need to manually build Qwt 6.1.0 (or later). Then, assuming that you have downloaded the source for version 1.3 in your `Downloads` directory, you can compile it by typing the following commands in the terminal:

```
unzip dolmen-2.0.zip
cd dolmen
qmake dolmen.pro; make
```

This will create an executable file called `dolmen` that you can put anywhere. To put it in `/usr/local/bin`, do:

```
sudo mv dolmen /usr/local/bin/
```

Assuming that `sudo` is installed and properly configured on your system. You can then run Dolmen by simply typing `dolmen` in the terminal.

In order to be able to read the documentation, you will also need to put the `html` directory somewhere on your disk, and adjust the `resources` path. To do this, go to `Edit > Preferences...` and in the `General` tab, adjust the path for the `Resources folder` to match your installation.

3.1.5 Known issues

On Mac OS, clicking on the sound scrollbar buttons after an item is selected in a tier results in the scollbar moving until an edge is reached.

3.2 Getting started

3.2.1 Main window

The left panel is the *file manager*: by default, it displays the hierarchical structure of the project, but it can also display bookmarks associated with the current project. The right panel is the *information panel*, which is used to display and edit metadata about the file(s) currently selected. The bottom panel is the *console*, which can be used to type commands using Dolmen's scripting engine. Finally, the central part of the user interface is the *viewer*, which displays views such as the result of a query. Each view is displayed as a tab, in a similar fashion to web pages in a modern browser. The default view, the start view, displays a few buttons for the most common operations a user may want to perform.

3.2.2 Corpus management

Several functions from the `File` menu let the user important files into a project, either individually or by importing a folder recursively. The logical structure of a project is independent from the physical organization of the files on the user's computer: once files have been added to a project, they can be moved around, merged into new folders or removed without affecting the files on disk. Dolmen supports several annotation formats, including TextGrid (Praat) and LAB (WaveSurfer). It also supports a number of audio formats, including WAV, AIFF and FLAC (the exact number of supported formats depends on the platform). By default, Dolmen will try to automatically bind an annotation and a sound file if they have the same base name but a different extension. If the names differ, it is possible to bind them manually, by right-clicking on them and choosing the corresponding option in the context menu, or semi-automatically using the `Import metadata...` feature from the file menu or using the scripting engine.

The hierarchical organization of a project is a matter of pure convenience to the user and is irrelevant for Dolmen. Instead, the program relies on metadata to keep files organized internally and to perform queries. File names represent the most basic type of metadata and for small projects (containing a dozen of files or so) this may be all that is needed. When one needs to sort and organize a larger collection of files, Dolmen offers a flexible mechanism called *properties*. A property is a typed key/value pair. Each file can be tagged with an arbitrary number of such properties: the key represents a category, which is always a text string, and the value may be either Boolean, textual or numeric. Typical examples of properties would be *Speaker* (where each unique speaker identifier represents a distinct value, for example *11ajp1*) and *Gender* (with the values *Male* and *Female*). Properties can be managed via the property editor, available from the information panel when a file selection is active.

In addition to properties, each file can be annotated with a description, a free-form string which can used to store any kind of information, and which is also exposed to the search engine to filter files.

3.3 Concordance

Dolmen offers a number of features to find concordances in a corpus. It also allows user to customize its search interface by creating search grammars specifically tailored for a project. Concordancing features are available in the `Search` menu.

3.3.1 Simple queries

To run a new query, click on `Search > Find concordances...` or use the shortcut `ctrl+F` (or `cmd+f` on macOS). This will open a new search window, which lets you search through all the *documents* (i.e. plain text files) or *annotations* in your corpus . We will focus on annotations here, but things work in a similar way for documents, mutatis mutandis.

The Files box

The `Files` box in the top left corner allows you to select the type of files to search in (documents or annotations). You can either select files individually if you want to restrict your query to a particular set of files, or leave all files unchecked, in which case Dolmen will try to search in all files.

The Search box

The `Search` box in the top right corner allows you to enter some text or a regular expression to search for. Next to the search field, a spin box lets you select the tier you want to search in. The default choice is `Any tier`, which means that Dolmen will try to find your pattern in all tiers of the selected files. You can restrict the search to a particular tier by selecting the appropriate tier number. Alternatively, you can specify a *tier name pattern* using a regular expression. If you do specify a tier name pattern, Dolmen will ignore the tier number and will search instead in any tier whose name matches the pattern.

By default, the text in the search field is interpreted as a regular expression. If you are looking for plain text string instead, you can select `plain text` instead of `regular expression` in the selector located under the + and − buttons. Whether you use a plain text string or a regular expression, the search will be case-insensitive by default, which means that strings like “foo”, “Foo” and “FOO” are treated as identical. To perform a case sensitive search, simply check the `case sensitive` box.

Concordances in a simple query follow the KWIC model (key word in context), which means that a match is extracted along with its left and right context. (The length of the context window can be adjusted in the preferences.) When the context window is longer than the context of the match in a given item (time point or interval), dolmen will extract additional text in the preceeding/following items until the left/right contexts have the expected length. If it cannot find enough items, it will pad the text with white space. By default, Dolmen will use one white space character to join the text from different intervals. You can specify a different string (including an empty string) in the `Separator` field.

Metadata

If you have added properties to your project, a set of property boxes will be added below the `Files` and `Search` box. Each property category is displayed as a group box containing a list of all the labels of this category. You can check or uncheck any label in any category (each category also has an `All labels` button to check/uncheck all labels at once). The search engine will filter files based on the conditions that you specify in the property box. Within a category, it uses the Boolean `OR` operator to find the subset of files that has either label. Across categories, it uses the `AND` operator to find the intersection of all the subsets defined by each category.

At the bottom of search window, an additional field lets you filter files based on their description. For example, it is possible to extract all the files that contain (or do not contain) a specific string.

Viewing results

Once you hit the `ok` button, the result of your query is presented as a new `query view` in the viewer. You can browse the results with the mouse wheel. The information panel on the right-hand side displays information about the selected token.

If an annotation is bound to a sound file, you can play a match by double-clicking on it or by pressing the space bar (you can also interrupt it by pressing `Esc`).

Right-clicking on item will display a context menu that allows you to perform a number of actions:

- `Play selection`: this will play the corresponding item if the annotation is bound to a sound file.
- `Open in annotation`: this will open the annotation in a new view, along with its sound file if it is bound to a sound.
- `Open selection in Praat`: if Praat is installed and configured to work with Dolmen, this will open the match in Praat. Dolmen will open the TextGrid (and the sound file if the annotation is bound) in Praat and will display the current match. (Note that you need to have Praat already running for this to work.)
- `Edit item text`: this allows you to modify the text of the item where the match was found. (Note that the query view is currently not updated to reflect this change.)
- `Create table view`: this will convert the concordance set, along with all its metadata, to a table that can be imported into a spreadsheet program.
- `Export results to tab-separated file (CSV)...`: this exports the concordance set, along with all its metadata, to a CSV file that can be imported into a spreadsheet program.
- `Bookmark search result`: this allows you to bookmark a matched item. Bookmarks are displayed in the bookmark panel, which can accessed by clicking on the star in the bottom left corner of the main window.

3.3.2 Complex queries

One performing a simple queries on a set of annotation files, Dolmen attempts to find a set of concordances in one item (point or interval) at a time. While it is possible for an item to match a given search pattern several times if several substrings match the pattern, matches are nevertheless limited to a single item.

Sometimes, however, we might want to match text in several items *simultaneously*. Such a query is called a *complex query* in Dolmen. There are 3 types of relations between items, detailed below: alignment, precedence and dominance.

Building a complex query

When you open a search window, two small buttons with a + and − sign appear below the main the search field. These buttons allow you to add and remove search items. Any query which has more than one search item is a complex query.

When you add one or more search items, you will notice that each of them (except the last one) is followed by a selector with 3 possible values: `is aligned with`, `precedes` and `dominates`. They correspond to the tier item relations `alignment`, `precedence` and `dominance`, respectively.

Contrary to simple queries, complex queries do not use the KWIC model to display results. Instead of displaying a matched string in its context, it lets the user select a `display tier`, which appears at the top of the search box. The text that is displayed is the concatenation of all the items contain within the time interval defined by simultaneous satisfaction of the constraints on each search item. Several examples are given below.

Alignment relation

Two items are aligned if they are on different tiers and their left and right boundaries coincide. Suppose that you have a word tier (tier 1), where each word was segmented, and a part-of-speech (POS) tier (tier 2) which is aligned with the word tier. To extract all the nouns in the corpus, you could do the following:

- set `NOUN` as the search pattern for tier 1, and choose the `is aligned with` value of the relation selector.
- set `.+` as the search pattern for tier 2
- set the display tier to tier 2

Dolmen will first look for all items whose text contains “NOUN” on tier 1, and will keep all those items which contain a non-empty label in an item of tier 2 which is exactly aligned with a NOUN item on tier 1. Dolmen will then return a list of the text labels on tier 2 which match the above criteria.

As another example, suppose you now want to extract all the adverbs that end with *-ly*. You could do the following:

- set `ADV` as the search pattern for tier 1, and choose the `is aligned with` value of the relation selector.
- set `.+ly$` as the search pattern for tier 2
- set the display tier to tier 2

Assuming that tier 2 contains exactly one word per interval, this will successfully extract all the adverbs on tier 2 that end with *-ly*.

Precedence relation

Two items are in a precedence relation if they immediately follow each other. You can search for arbitrarily long sequences by chaining search items on the same tier. When you specify a sequence, Dolmen will retrieve the text from the display tier that is included within the span defined by the sequence.

Suppose that you have a word tier (tier 1) and a POS tier (tier 2), as in the alignment examples. Instead of searching for a single word, you might be interested in looking for word sequences. To find all the `DET+NOUN` sequences, you could do the following:

- set `DET` as the search pattern for the first tier item in tier 1, and choose the `precedes` value of the relation selector.

- set NOUN as the search pattern for the second tier item, setting the tier number to 1 to ensure you are looking in the same tier
- set the display tier to tier 2

Dolmen will first look for all DET items on tier one, and will keep only those that are followed by a NOUN item on the same tier. It will then display the text that results from the concatenation of all the items on tier 2 within the span determined by the beginning of the DET item and by the end of NOUN item on tier 1.

Dominance relation

An item a dominates an item b if a and b are on different tier, the left boundary of b is greater or equal to that of a, and the right boundary of b is lesser or equal to that of a. Dominance relations typically encode hierarchical structures, for instance `word > syllable > segment`.

Suppose you have 3 tiers in your file: the first one contains spans which denote syllables, the second one contains syllabic constituents ("syll") ("Onset", "Nucleus", "Coda") and the last one individual segments ("p", "a", "t"...). In order to retrieve all syllables that end in a coda, you could do the following:

- set syll as the search pattern for tier 1, and choose the dominates value of the relation selector.
- set Coda as the search pattern for tier 2
- set the display tier to tier 3

This query will first get all the items that have a syll label on the first tier; then, for each of those, it will look for a label Coda on tier 2 within the limits of the span on tier 1; for each item which matches both conditions, it will display the concatenated text of the items on tier 3 that are dominated by the matching item on tier 1.

3.4 Scripting

3.4.1 Overview

Dolmen can be extended using the Lua scripting language. There are several ways this

- The console at the bottom of the main window accepts lua commands.
- *Plugins* are extensions, written in a JSON and Lua, which can add features to Dolmen

3.4.2 Modules and functions

Regular expressions

This page documents the regex module.

General concepts

Regular expressions are widely used in text processing to perform pattern matching and pattern substitution. Simply put, a regular expression (regex) is a string which describes a *set of strings*. Suppose that we want to any of the following strings: "petit", "petite", "petits", "petites". Instead of looking for each string separately, we can use a regular expression to look for any of them. The corresponding regular expression would be "petite? s?".

Syntax

Regular expressions always try to match a pattern from left to right; in their simplest form, they match a sequence of (non-special) characters and are equivalent in this case to a plain text search. Regular expressions provide a number of special symbols and operators that can match classes or sequences of characters. Here we only provide the most useful ones:

- `.` : match any character
- `^` : match the beginning of a string
- `$` : match the end of a string
- `[xyz]` : match either of the characters `x`, `y` or `z`
- `[^xyz]` : match any character except `x`, `y` or `z`
- `[a-z]` : match any character in the range from `a` to `z`
- `\b` : match a word boundary
- `\s` : match a white space character
- `\d` : match a digit character (equivalent to `[0-9]`)
- `\w` : match a word character, including digits and _ (underscore)

In addition, regular expressions offer a number of quantifiers:

- `E?` : match 0 or 1 occurrences of the expression E
- `E*` : match 0 or more occurrences of the expression E
- `E+` : match 1 or more occurrences of the expression E
- `E{n}` : match exactly n occurrences of the expression E
- `E{n,m}` : match between n and m occurrences of the expression E
- `E{n,}` : match at least n occurrences of the expression E
- `E{,m}` : match at most m occurrences of the expression E (and possibly 0)

In this context, an expression must be understood as either a character (e.g. `o{2,}` matches the string `"zoo"`) or a sequence of characters enclosed by parentheses (e.g. `(?:do){2}` matches the string `"fais dodo"`). Another useful character is `|`, which is used to combine expressions (logical OR). For example, the pattern `(?:est|était)` will find all occurrences of the strings est and était.

Regular expressions are “greedy” by default, which means they will match the longest string that satisfies the pattern. For instance, given the pattern `j.*e`, which matches the character `j` followed by zero or more characters followed by `e`, and the string `"je te l'ai dit"`, a non-greedy search will return the substring `"je te"` by default. Non-greedy search, on the other hand, will yield the substring `"je"` since it extracts the shortest string that satisfies the regular expression. To enable non-greedy behavior, we must use the quantifier `?` after the star (in this case, `"j.*?e"`).

Functions

`regex.`**`new`**`(`*pattern*`)`

Create and return a new regular expression (regex) from a string pattern. The regex can be matched against any string.

```lua
local re = regex.new("^(..)")
-- Do something with re...
```

See also: *pattern()*

regex.**match**(*re*, *subject*)

Match regular expression `re` against string `subject`. Returns `true` if there was a match, `false` otherwise.

See also: *count()*, *capture()*, *has_match()*

regex.**has_match**(*re*)

Returns `true` if the last call to `match` was sucessful, and `false` if was unsuccessful or if `match` was not called.

See also: *match()*

regex.**capture**(*re*, *nth*)

Returns the `nth` captured sub-expression in the last successful call to `match`. If `nth` equals `0`, the whole matched string is returned, even if no sub-expression was captured.

Note: This function returns an empty string if `nth` is greater than the number returned by the `count` function.

See also: *count()*, *match()*, *first()*, *last()*

regex.**count**(*re*)

Returns the number of captured sub-expressions in the last call to `match`. This function returns 0 if there was no captured sub-expression, if there was no match or if `match` was not called.

```lua
local re = regex.new("^a(...)(..)(..)")

-- Print "bra", "ca", "da"
if regex.match(re, "abracadabra") then
    for i=1, regex.count(re) do
        local text = regex.capture(re, i)
        print(text)
    end
end
```

See also: *capture()*, *match()*

regex.**pattern**(*re*)

Returns the pattern (as a `string`) from which the `re` regular expression was constructed.

See also: *new()*

regex.**first**(*re*, *nth*)

Returns the index of the first character of the `nth` capture. If `nth` equals `0`, it returns the index of the first character in the whole matched string.

See also: *capture()*, *last()*

`regex.`**`last`**`(`*re*, *nth*`)`

Returns the index of the last character of the `nth` capture. If `nth` equals `0`, it returns the index of the last character in the whole matched string.

See also: *`match()`*, *`first()`*

Shell

This page documents the `shell` module.

General concepts

The shell represents Dolmen's user interface. The following functions let you use user interface elements (such as dialogs) in order to facilitate interaction with users of your scripts.

Functions

`shell.`**`warning`**`(`*message*`)`

Displays a warning dialog.

See also: *`alert()`*

`shell.`**`alert`**`(`*message*`)`

Displays an error dialog. This can be used for critical errors.

See also: *`warning()`*

`shell.`**`open_file_dialog`**`(`*message*`)`

Displays a dialog that lets the user select a file.

See also: *`save_file_dialog()`*, *`open_directory_dialog()`*

`shell.`**`save_file_dialog`**`(`*message*`)`

Displays a dialog that lets the user choose a path to save a file.

See also: *`open_file_dialog()`*, *`open_directory_dialog()`*

`shell.`**`open_directory_dialog`**`(`*message*`)`

Displays a dialog that lets the user select a directory.

See also: *`save_file_dialog()`*, *`open_file_dialog()`*

`shell.`**`status`**`(`*message*, *timeout*`)`

Displays `message` in the status bar for `timeout` seconds. If `timeout` is `0`, the message is displayed until the next one appears.

Event handling

This page documents the `signal` module, which is responsible for event handling in Dolmen.

General concepts

Dolmen provides an event handling mechanism known as signal/slot. A signal corresponds to a unique identifier which can be triggered when an event occurs, for instance when a button is clicked. A signal can be associated with any number of functions called *slots*, which may or may not return a value. Whenever a signal is *emitted*, all the slots which are connected to it are executed (in an unspecified order).

This mechanism is used throughout Dolmen, as it provides hooks which plugins can use to react to events triggered by the program. For example, a signal is emitted whenever a file is loaded, which can be used to add custom metadata to each file, among other things.

Functions

`signal.`**`new`**`()`

Create and return a new signal identifier (id). Each id is guaranteed to be unique, such that two different calls to `new` will never yield the same id.

If you need to store an id for subsequent use, store it in a (preferably local) variable.

```lua
local my_event = signal.new()
-- Do something with my_event...
```

`signal.`**`connect`**`(`*id*, *slot*`)`

Connect signal `id` to function `slot`. The slot can take any number of arguments, and can return a value.

```lua
local e = signal.new()

local f = function(name)
    print("Hold the door, " .. name)
end

signal.connect(e, f)

-- Print "Hold the door, Hodor" to the standard output
signal.emit(e, "Hodor")
```

See also: *`disconnect()`*, *`emit()`*

`signal.`**`disconnect`**`(`*id*, *slot*`)`

Disconnect signal `id` from function `slot`. If `id` and `slot` are not connected, this function does nothing.

```lua
local e = signal.new()

local f = function(name)
    print("Hold the door, " .. name)
end
```

```
signal.connect(evt, f)

-- Print "Hold the door, Hodor" to the standard output
signal.emit(e, "Hodor")

signal.disconnect(e, f)

-- Do nothing since e and f are no longer connected
signal.emit(e, "Hodor")
```

See also: *connect()*, *emit()*

---

signal.**emit**(*id*, ...)

Emit signal `id`, followed by any number of arguments. The arguments are forwarded to all the slots which are connected to this signal (if any). Following Lua's function call conventions, if the slot receives less arguments than it expects, missing arguments have the value `nil`; if it receives more arguments than expected, additional arguments are ignored.

This function collects all the return values from the slots it called into a table, which it returns to the caller. (Keep in mind that if a slot doesn't explicitly return a value, its return value is `nil`.)

```
local e = signal.new()

local f1 = function(arg1)
    print("f1 received a " .. type(arg1))
end

local f2 = function(arg1, arg2)
    print("f2 received a " .. type(arg1) " and a " .. type(arg2))
end

signal.connect(e, f1)
signal.connect(e, f2)

-- Print "f1 received a number" and "f2 received a number and a string"
signal.emit(e, 3.14, "pi")
```

Note: the order in which slots are called is unspecified. In general, it will correspond to the order in which they were registered, but this should not be relied upon.

See also: *connect()*, *disconnect()*

String manipulation

This page documents the `string` module.

General concepts

A `string` is a sequence of characters enclosed between double quotes, such as `"this"`. Strings in Lua are immutable, which means that you cannot modify them directly. All functions which "modify" a string actually return a new (modified) copy of the string but leave the original string unchanged.

All string functions assume that strings are encoded according to the UTF-8 Unicode standard. A good tutorial about UTF-8 can be found at the following address: http://www.zehnet.de/2005/02/12/unicode-utf-8-tutorial. In the remainder of this document, the word *character* is used to mean *code point*, unless otherwise noted.

Functions

`string.`**`at`**`(`*str*, *pos*`)`

Get character at position `pos`.

`string.`**`len`**`(`*str*`)`

Returns the length of the string, in Unicode code points.

```
local size = string.len("안녕하세요")
print(size) -- Prints "5"
```

See also: *`byte_count()`*

`string.`**`byte_count`**`(`*str*`)`

Returns the length of the string, in bytes (or Unicode code units). For strings encoded in ASCII (mostly, strings of English text with no “special” character), each code unit is represented with 1 byte, such that `bytecount` and `len` return the same result. For most other languages, however, the number of bytes and the number of code points will be different.

```
local english = "hello"
local korean  = "안녕하세요"

print(string.len(english)) -- prints "5"
print(string.len(korean))  -- prints "5"

print(string.byte_count(english)) -- prints "5"
print(string.byte_count(korean))  -- prints "15"
```

See also: *`len()`*

`string.`**`trim`**`(`*str*`)`

Returns a copy of the string with whitespace characters removed at both ends of the string.

```
local s = "\t  hello  \n"

s = string.trim(s)
print("$" .. s .. "$") -- prints "$hello$"
```

See also: *`ltrim()`*, *`rtrim()`*

`string.`**`ltrim`**`(`*str*`)`

Returns a copy of the string with whitespace characters removed at the left end of the string.

```lua
local s = "  hello  "

s = string.ltrim(s)
print("$" .. s .. "$") -- prints "$hello  $"
```

See also: *trim()*, *rtrim()*

string.**rtrim**(*str*)

Returns a copy of the string with whitespace characters removed at the right end of the string.

```lua
local s = "  hello  "

s = string.rtrim(s)
print("$" .. s .. "$") -- prints "$  hello$"
```

See also: *ltrim()*, *trim()*

string.**starts_with**(*str*, *prefix*)

Returns true if `str` starts with `prefix`, and `false` otherwise.

See also: *ends_with()*

string.**ends_with**(*str*, *suffix*)

Returns true if `str` ends with `suffix`, and `false` otherwise.

See also: *starts_with()*

string.**contains**(*str*, *substring*)

Returns true if `str` ends with `substring`, and `false` otherwise.

string.**count**(*str*, *substring*)

Returns the number of times `substring` appears in `str`.

```lua
local s = "cacococococa"
local count = string.count(s, "coco")

print(count) -- prints "2"
```

Note: matches don't overlap.

string.**to_upper**(*str*)

Returns a copy of `str` where each code point has been converted to upper case.

```lua
local s1 = "c'était ça"
local s2 = string.to_upper(s1)

print(s2) -- prints "C'ÉTAIT ÇA"
```

See also: *to_lower()*

---

string.**to_lower**(*str*)

Returns a copy of `str` where each code point has been converted to lower case.

```lua
local s1 = "C'ÉTAIT ÇA"
local s2 = string.to_lower(s1)

print(s2) -- prints "c'était ça"
```

See also: *to_upper()*

---

string.**replace**(*str*, *old*, *new*)

Returns a copy of `str` where all (non-overlapping) instances of the substring `old` have been replaced by `new`.

See also: *replace_at()*, *replace_first()*, *replace_last()*

---

string.**replace_at**(*str*, *at*, *count*, *new*)

Returns a copy of `str` where `count` code points, starting at position `at`, have been replaced by `new`.

See also: *replace()*, *replace_first()*, *replace_last()*

---

string.**replace_first**(*str*, *old*, *new*)

Returns a copy of `str` where the first instance of the substring `old` has been replaced by `new`.

See also: *replace_at()*, *replace()*, *replace_last()*

---

string.**replace_last**(*str*, *old*, *new*)

Returns a copy of `str` where the last instance of the substring `old` has been replaced by `new`.

See also: *replace_at()*, *replace()*, *replace_first()*

---

string.**concat**(*str1*, *str2*)

Create a new string which is the concatenation of `str1` and `str2`. In general, you should use Lua's built-in concatenation operator `..` instead of this function.

---

string.**contains**(*str*, *substr*)

Returns true if `str` contains `substr` and false otherwise. If `substr` is the empty string, the result is true.

---

string.**remove**(*str*, *substr*)

Returns a copy of `str` where all (non-overlapping) instances of the substring `substr` have been removed.

See also: *remove_at()*, *remove_first()*, *remove_last()*

`string.`**`remove_at`**(*str*, *at*, *count*)

Returns a copy of `str` where `count` code points, starting at position `at`, have been removed.

See also: *`remove()`*, *`remove_first()`*, *`remove_last()`*

`string.`**`remove_first`**(*str*, *substr*)

Returns a copy of `str` where the first instance of `substr` has been removed.

See also: *`remove_at()`*, *`remove()`*, *`remove_last()`*

`string.`**`remove_last`**(*str*, *substr*)

Returns a copy of `str` where the last instance of `substr` has been removed.

See also: *`remove_at()`*, *`remove()`*, *`remove_first()`*

`string.`**`reverse`**(*str*)

Returns a new string with all the characters in `str` in reversed order.

`string.`**`insert`**(*str*, *pos*, *other*)

Returns a copy of `str` with `other` inserted at position `pos`

`string.`**`substr`**(*str*, *from*, *to*)

Returns the substring of `str` starting at index `from` and ending at index `to` (inclusive). If `to` equals `-1`, returns the substring from `from` until the end of the string.

```lua
local s = "c'était ça"

print(string.substr(s, 3, 7)) -- "était"
print(string.substr(s, 3,-1)) -- "était ça"
```

`string.`**`left`**(*str*, *n*)

Get the substring corresponding to the `n` first characters of the string.

`string.`**`right`**(*str*, *n*)

Get the substring corresponding to the `n` last characters of the string.

`string.`**`first`**(*str*)

Get the first character of the string.

`string.`**`last`**(*str*)

Get the last character of the string.

`string.`**`join`**(*strings*, *delim*)

Returns a new string which is the result of the concatenation of the strings in table `strings`, separated by `delim`.

`string.`**`split`**(*str*, *delim*)

Returns a table of strings which have been split at each occurrence of the substring `delim`. If `delim` is the empty string, it returns a list of the characters in the string.

3.5 Plugins

Dolmen can be extended with plugins, which are written in JSON and the Lua scripting language. When it starts up, Dolmen loads all plugins which are located in the system plugin directory or in the user plugin directory. Plugins can be redistributed as ZIP files (the `.zip` extension is compulsory). To install a plugin, go to `File > Install plugin...` and choose the ZIP file. It will be installed in the current user's plugin directory.

See *Scripting* to learn more about scripting.

3.5.1 Structure of a plugin

To be valid, a plugin must adhere to a number of conventions: if they are not respected, Dolmen will silently ignore the plugin. The root directory of the plugin must contain the following:

- a description file, named description.json (compulsory)
- a `Scripts` sub-directory, which contains all your scripts (optional).
- a `Grammars` sub-directory, which contains all your
- a `Resources` sub-directory, which may contain anything (optional).

The description file contains all the information necessary to initialize the plugin. All declarative aspects of the plugin are written in the JSON format and must bear the extension `.json`. Scripts are written in Lua and must bear the extension `.lua`.

Here is an example of a basic `description.json` file:

```
{
    "PluginInfo": {
        "Name": "My first plugin",
        "Version": "0.1",
    },

    "Menu": {"Text": "Custom menu", "Actions":
        [
            {"Type": "Action", "Text": "Test script", "Script": "test.lua", "Shortcut
↪": "Ctrl+T"}
        ]
    }
}
```

The header PluginInfo is the only part that is compulsory. It contains essential information about the plugin. The Menu key lets you create a custom menu: each menu entry (called "action") is associated with a script which must be located in the Scripts sub-directory. When you click on an action in the menu, the corresponding script is executed. It is also possible to assign a shortcut to a given action.

3.5.2 Defining search grammars

If you have devised a coding scheme for your data, Dolmen lets you define a "search grammar". A search grammar is a description of your coding scheme which offers a user-friendly interface for querying your data; it tells Dolmen what to look for and how to present the information to the user. Dolmen will automatically load all valid search grammars that are located in the Grammars sub-directory of your plugin. It will create a submenu in the Conc menu, whose name is the name of your plugins. All search grammars will be be available as tabs in that submenu.

A search grammar defines a number of fields which can take on a number of values. The user is presented with a number of checkboxes for each field, and Dolmen converts the query to the corresponding regular expression, as defined by the grammar. Here is a simple yet realistic example, drawn from the PFC project:

```
{
    "Header" : {
        "Object" : "SearchGrammar",
        "DisplayName": "Schwa",
        "Version": "0.9",
    },

    "Separator": "",
    "FileType": "Annotation",
    "Tier": 2,
    "FieldsPerRow": 2,

    "Fields" : [

        {"Name": "Schwa", "MatchAll": "[0-2]",
         "Values": [
            {"Match": "0", "Text": "Absent"},
            {"Match": "1", "Text": "Present"},
            {"Match": "2", "Text": "Uncertain"},
         ]
        },

        {"Name": "Position", "MatchAll": "[1-5]",
         "Values": [
            {"Match": "1",  "Text": "monosyllable"},
            {"Match": "2",  "Text": "initial syllable"},
            {"Match": "3",  "Text": "median syllable"}
            {"Match": "4",  "Text": "final syllable"}
            {"Match": "5",  "Text": "metathesis"}
         ]
        },


        {"Name": "Left context", "MatchAll": "[1-5]",
         "Values": [
            {"Match": "1",  "Text": "vowel"},
            {"Match": "2",  "Text": "consonant"}
            {"Match": "3",  "Text": "start of an intonational phrase"}
            {"Match": "4",  "Text": "uncertain vowel"}
            {"Match": "5",  "Text": "simplified cluster"}
         ]
        },


        {"Name": "Right context", "MatchAll": "[1-4]",
         "Values": [
            {"Match": "1", "Text": "vowel"},
            {"Match": "2", "Text": "consonant"},
            {"Match": "3", "Text": "weak prosodic boundary"},
            {"Match": "4", "Text": "strong prosodic boundary"}
         ]
        }
    ]
}
```

We first see a Header, which provides information about the file. The field Object indicates that the file is a search grammar, the Name corresponds to the name of the grammar, as it will be seen by the user, and Version corresponds to the version of the grammar.

Next, the Separator attribute indicates the separator to be used between fields. In this case, it is an empty string, which means that the fields are concatenated directly (e.g. 1412). If the separator was _, each field should be separated by this symbol (e.g. 1_4_1_2). Note that although a field may be only one digit (or one character), it does not need to be so; a field can be of any length, provided that it can be described by a regular expression.

Next the Tier attribute indicates the tier number in which codings should be searched for.

The following attribute, FieldsPerRow, lets us specify how many fields should be displayed in a row. In our case, since there are 4 fields, we decide to distribute them across 2 rows containing 2 fields each.

Finally, the Fields attribute contains a list of fields, each of them corresponding to a JSON object. The Name attribute provides a descriptive label for the field. The MatchAll attribute is a regular expression that should match all possible values for the field. If a user doesn't check any value for a field, this attribute will be used to retrieve all possible values. The Values attribute contains a list of value. Each of them contains (at least) a Match attribute, which is a string corresponding to the value, and a Text attribute which is the label that will be displayed in the user interface for the corresponding value, along with a check box. Note that checking all values has the same effect as leaving all values unchecked.

TO BE CONTINUED…

3.6 GNU General Public License

Version 3, 29 June 2007 Copyright © 2007 Free Software Foundation, Inc <http://fsf.org>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

3.6.1 Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: **(1)** assert copyright on the software, and **(2)** offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

3.6.2 TERMS AND CONDITIONS

0. Definitions

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that **(1)** displays an appropriate copyright notice, and **(2)** tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that **(a)** is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and **(b)** serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- **a)** The work must carry prominent notices stating that you modified it, and giving a relevant date.
- **b)** The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- **c)** You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- **d)** If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- **a)** Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

- **b)** Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either **(1)** a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or **(2)** access to copy the Corresponding Source from a network server at no charge.
- **c)** Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- **d)** Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- **e)** Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either **(1)** a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or **(2)** anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- **a)** Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- **b)** Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- **c)** Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- **d)** Limiting the use for publicity purposes of names of licensors or authors of the material; or
- **e)** Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- **f)** Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated **(a)** provisionally, unless and until the copyright holder explicitly and finally terminates your license, and **(b)** permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either **(1)** cause the Corresponding Source to be so available, or **(2)** arrange to deprive yourself of the benefit of the patent license for this particular work, or **(3)** arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license **(a)** in connection with copies of the covered work conveyed by you (or copies made from those copies), or **(b)** primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

3.6.3 How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author>
>
> This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

> <program> Copyright (C) <year> <name of author> This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'. This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands *show w* and *show c* should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

3.7 Acknowledgements

Dolmen uses the following open source components:

- wxWidgets, maintained by Julian Smart, Robert Roebling et al (LGPL with static linking exception), see www.wxwidgets.org
- libsndfile, maintained by Erik de Castro Lopo (LGPL), see www.mega-nerd.com
- RTAudio, maintained by Gary P. Scavone (MIT license), see www.music.mcgill.ca/~gary/rtaudio
- Speex, maintained by Jean-Marc Valin and contributors (BSD license), see www.speex.org
- JSON for Modern C++, maintained by Neils Lohmann (MIT), see https://github.com/nlohmann/json
- pugixml, maintained by Arseny Kapoulkine, see pugixml.org
- sendpraat, maintained by Paul Boersma, see www.praat.org
- FLAC, by Josh Coalson and contributors (BSD), see flac.sourceforge.net
- Inno Setup (Windows installer), by Jordan Russell (free), www.jrsoftware.org
- Lua, maintained by PUC-Rio (MIT), see lua.org
- sol2, maintained by Rapptz, ThePhD and contributors (MIT), see sol2.rtfd.io
- sphinx, maintained by the Sphinx team (BSD), see www.sphinx-doc.org
- utf8proc, maintained by the Public Software Group (MIT), see julialang.org/utf8proc
- PCRE2, maintained by Philip Hazel (BSD), see www.pcre.org
- icons from icons8.com (Creative Commons Attribution-NoDerivs 3.0 Unported).

Dolmen uses the following service for hosting its source code:

- GitHub

The development of search grammars was supported by the Japanese Society for the Promotion of Science (JSPS), Grant-in-Aid for Scientific Research (B) n°23320121 (2011-2014). Project title: *A corpus-based longitudinal study of the interphonological features of Japanese learners of French*. Project leader: Sylvain DETEY (Waseda University).

3.8 Release notes

3.8.1 Version 2.0.0 (30/3/2018)

Release date: - updated sol2 library - fixed Lua string -> QString

3.8.2 Version 1.9.5 (30/3/2018)

- Fixed a regression introduced in 1.9.4 which caused Dolmen to crash when there was no main plugin

3.8.3 Version 1.9.4 (16/3/2018)

- Restore Unicode support in regular expression character classes
- Lua scripts in plugins
- Initialization code can be added to a script named `init.lua` at the root of a plugin

3.8.4 Version 1.9.3 (23/11/2017)

- Fixed opening a TextGrid with Sound in Praat, which sometimes opened the wrong interval

3.8.5 Version 1.9.2 (23/9/2017)

- Updated RTAudio to version 5.0.0
- Improved sound playback on all platforms
- The last query is now remembered
- Small performance improvements in queries
- Fixed regression for search grammars introduced in 1.9.1
- Renamed `Conc` menu to `Search`

3.8.6 Version 1.9.1 (14/9/2017)

- Complex queries now recognize 3 relations: alignment, precedence and dominance
- Updated documentation for concordancing
- Better warning when a query doesn't return any match

3.8.7 Version 1.9.0 (25/8/2017)

- Small changes in the user interface
- lua scripting engine
- New regular expression engine
- New documentation

- Removed obsolete UNIX pattern search syntax
- Fixed freeze with open/save dialogs on Linux (Qt5)
- Added Conc menu
- Auto-detection of Unicode encoding (defaults to UTF-8 if no BOM is present)
- Metadata panel can now be resized
- Removed greedy search check box (use quantifiers in regular expressions instead)
- Bookmarks can now be opened in Praat
- Fixed bug which caused the content of an annotation to be loaded multiple times when there were bookmarks in a project
- Fixed crash which occurred when switching between projects
- Fixed minor memory leak (finalization of grammars)
- License is now GPL version 3, with an exception for plugins
- New icon theme (courtesy icons8.com)
- Replaced output tab with a (hideable) status bar

3.8.8 Version 1.3.3 (24/5/2016)

- fixed encoding when exporting search results to CSV
- minor bug fixes

3.8.9 Version 1.3.2 (8/5/2016)

- minor bug fixes

3.8.10 Version 1.3.1 (28/04/2016)

- query results are now sorted

3.8.11 Version 1.3 (12/9/2014)

- fixed bug in sound playing on Windows and Linux
- import metadata from CSV file
- fixed a bug introduced in version 1.2 that made Dolmen crash when loading a project that contains bookmarks
- now built on top of Qt5
- better compatibility with recent version of Mac OS X (now requires 10.7 or later)
- plugin tabs are no longer displayed in random order
- updated RtAudio to version 4.1.1
- minor fixes and improvements

3.8.12 Version 1.2 (1/9/2013)

- improvements to sound and annotation views
- sound can be browsed with keyboard arrows in sound and annotation views
- faster loading of projects (annotations are now opened lazily)
- updated RtAudio to version 4.0.12
- RtAudio and libsndfile versions are now displayed in Help > Sound information
- the source code can now be compiled with Qt4 or Qt5
- search by tier name now uses regular expressions instead of string comparison
- search grammars can now define a "tier selecting" field to search a pattern across different tiers
- fixed compiler warnings
- minor fixes and enhancements

3.8.13 Version 1.1 (1/7/2013)

- updated code to use Qwt 6.1 to prepare the transition to Qt 5 (Qwt is no longer included and must be provided externally)
- sound can now be played/paused with the space bar and stopped with Esc in sound and annotation views
- plots are now antialiased by default (this can be turned off in the preference editor)
- minor code clean-up's
- new icon

3.8.14 Version 1.0.4 (21/02/2013)

- Windows only: added msvcp100.dll, for systems that don't have it.

3.8.15 Version 1.0.3 (16/02/2013)

- prevent Dolmen from crashing when displaying a short sound file in a view

3.8.16 Version 1.0.2 (15/02/2013)

- Fixes audio playback of stereo files on Windows and Linux

3.8.17 Version 1.0.1 (16/12/2012)

- fixed a bug that happened when opening intervals shorter than 100 ms in Praat
- the tier name fied is now properly hidden in the search window when the files are Documents

3.8.18 Version 1.0 (15/12/2012)

- annotator comparison for search grammars (requires an “Annotator” property)
- Search by tier name instead of tier number
- code modified so that it compiles with MSVC2010
- updated libsndfile to 1.0.25 on Windows

3.8.19 Version 0.9.9 (21/10/2012)

- Qwt is no longer required as an external dependency (v5.2.2 has been included in Dolmen’s source code)
- updated Qt to v 4.8.3 on Windows

3.8.20 Version 0.9.8 (18/10/2012)

- improved sound playback on Mac OS X and Linux (updated RTAudio to v 4.0.11)
- fixed selection rectangle in waveforms on Linux
- updated Qt to v 4.8.3 on Mac OS X
- refactoring of the output console

3.8.21 Version 0.9.7.2 (04/05/2012)

- fixed a deployment bug on Windows (SQL plugin)
- error message if the database fails to open

3.8.22 Version 0.9.7 (23/04/2012)

- data tables (DMT files). Data tables can be created from a query view (right click on a query match > Create table view)
- option for greedy search in the main search window
- minor improvements and bug fixes

3.8.23 Version 0.9.6 (14/04/2012)

- support for user-defined plugins (written in JavaScript/JSON)
- cleaner check list boxes (mostly used in search window and gauge tool)
- disabled internal debug messages

3.8.24 Version 0.9.5 (08/04/2012)

This version brings a significant number of changes and improvements: * metadata is now displayed in a permanent widget on the right-hand side. * metadata is now stored in an SQLite database instead of DMM files * tags have been renamed to "properties". * redesigned property editor which makes editing properties much simpler * on Mac, the application data folder has been moved from "~/Library" to "~/Application Support" * resampling now uses speex instead of libsamplerate * support for Document files (plain text files) along with Annotations * bug fixes and usability improvements

3.8.25 Version 0.9.4 (03/03/2012)

- prevent a crash when using the arrow keys in a query view
- tags are now properly refreshed in the metadata editor

3.8.26 Version 0.9.3 (02/03/2012)

- fixes bugs in regular expression and plain text search
- prevent Dolmen from crashing when playing a new search match while the previous one hasn't finished playing
- search results can now be browsed using keyboard arrows

3.8.27 Version 0.9.2 (28/01/2012)

- new version (mac only): fixes a crash when playing a sound
- updated RTAudio to 4.0.10
- the Mac version is now linked against Qt 4.8

3.8.28 Version 0.9.1 (21/08/2011)

- fixed a bug that made Dolmen crash when loading a new project if the previous one had bookmarks
- added icons for surveys and speakers in the file browser
- items can now be edited from query views (right click, then "Edit item text…")
- the Linux version is now distributed as Debian packages (Debian stable)
- minor bug fixes and improvements

3.8.29 Version 0.9.0 (19/08/2011)

- implementation of tiers in annotation views (can read and edit text of Praat tiers)
- much smoother scrolling of waveform
- waveforms can now be scrolled with the mouse wheel
- new metadata editor, available from annotation and sound views, as well as from the file browser (right click on an item)
- new shortcuts to hide the left sidebar (ctrl+shift+L) and the bottom tabs (ctrl+shift+b)
- updated support for Praat point tiers (if you have old files with points, it is best to open and resave them in Praat)

- minor improvements to the side bar on Mac OS X

3.8.30 Version 0.8.3 (08/08/2011)

- Sound views now indicate the current position while a file is playing
- Removed Webkit dependency
- The Mac version is now built statically and is much smaller as a result

3.8.31 Version 0.8.2 (06/08/2011)

- Sound can now be paused in sound views
- Improved audio output of mono files on Mac OS X

3.8.32 Version 0.8.1 (23/07/2011)

- Fixed a bug that caused Dolmen to crash on Mac and Linux when playing a sound file containing non-ASCII characters in its path
- Fixed a bug that caused Dolmen to freeze on Windows when interrupting playing

3.8.33 Version 0.8.0 (16/07/2011)

- the sound stack has been rewritten and now uses libsndfile and RTAudio instead of libsox.
- initial support for waveforms in sound and annotation views (mono files)
- new tool “Compare speakers or subjects…”
- export to CSV (spreadsheet format)
- bookmarks
- open files in Praat from the file browser
- new command “save project as…”
- the system tab “metadata” has been removed (metadata is now shown in tool tips in the file browser or in the viewer when opening a file)
- simplified search window

3.8.34 Version 0.7.2 (20/03/2011)

- fixed display of results in query view when hovering the mouse cursor over items
- better Linux support

3.8.35 Version 0.7.1 (05/03/2011)

- read WaveSurfer label files (*.lab)
- export tiers of DMF files to TextGrid
- TextGrid files that contain text fields spanning over several lines are now properly parsed

3.8.36 Version 0.7.0 (04/03/2011)

This version brings many improvements across the board:

- sendpraat and sox have been included in the source code and are no longer neeeded as external dependencies
- cross-tier search
- the preference editor has been simplified and reorganized
- HTML rendering now uses the WebKit engine
- the documentation has been overhauled and is now available in HTML from within Dolmen, instead of a PDF file.
- many more sound file formats are supported out of the box (still no MP3 nor OGG though)
- the most recent project can now be opened with ctrl+shift+o
- minor bug fixes

3.8.37 Version 0.6.5 (19/02/2011)

Features:

- added project versioning (Edit > Project Properties…)
- can now open TextGrid without sound in Praat
- added search variables %LINE, %WORD, and symbols '#' (word boundary), '#*' (prefix) and '*#' (suffix).
- documentation for regular expressions

Bug fixes/improvements:

- removed "Open selection in Praat" from Tools menu (which has been disabled for now)
- parentheses can now be used in search pattern (they still need to be escaped in regular expressions)
- fixed identification of the tier when opening in Praat

3.8.38 Version 0.6.4 (12/02/2011)

- The description field is now searchable in the search window.
- Tags can be hidden in the search window
- PFC/PAC goodies (see documentation)

3.8.39 Version 0.6.3 (09/02/2011)

- First public version.

Development of the C++ version started in early August 2010. A python proof-of-concept was sketched out in April/May 2010. Dolmen is a complete redesign of the PFC platform (2006/2008), a concordancer implemented in Python and specifically written for the PFC project (www.projet-pfc.net).

CHAPTER

FOUR

HOW TO CITE?

To cite Dolmen, you can use the following citation:

You can also cite the following paper:

BIBLIOGRAPHY

[EYC2018] Eychenne, Julien (2018). Dolmen: a program for the analysis of speech corpora [Computer program]. Version 2.0.0, retrieved 04 May 2018 from http://www.dolmen-ling.org

[EYC2016] Eychenne, J. & R. Paternostro (2016). "Analyzing transcribed speech with Dolmen". In S. Detey, J. Durand, B. Laks & C. Lyche (eds) *Varieties of Spoken French*, Oxford: Oxford University Press, D35-D52.

LUA MODULE INDEX

INDEX