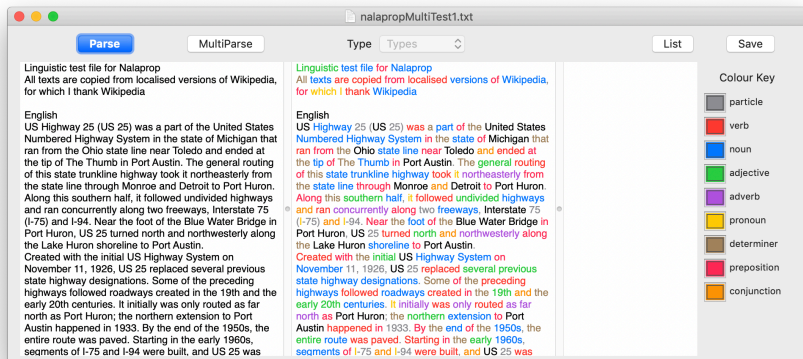


Start

Nalaprop uses natural language processing features in macOS 10.14 Mojave and later to parse text, mark it up according to detected parts of speech ('lexical classes'), generate classified frequency lists of words used, and more.

To open a plain text or rich text (RTF) file, use the **Open...** command in the **File** menu. To paste in plain text, if necessary create a new window using the **New** command, then paste that text into the lefthand pane of any document window. Rich text files are read in as plain text, as Nalaprop will impose its own styling to their text.

When you open an existing document, it will automatically be parsed into the centre pane, but word lists will not be generated at that stage. To re-parse the contents of the lefthand pane at any time, click on the **Parse** button or press Command-Return ⌘↵. Currently, parsing is performed in the main thread, which means that large documents such as a Charles Dickens novel can take some time to parse, and the spinning beachball cursor may appear for that period. A future version will perform parsing in the background to alleviate this issue.



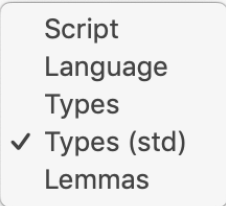
Nalaprop then uses macOS to parse the text and display the marked-up text in the centre.

This makes it easy to use Nalaprop interactively: adjust the window contents so that the left and centre views occupy as much of the width as possible, and type your text into the lefthand view, pressing ⌘↵ whenever you want to see it parsed. Unlike the centre and righthand views, you can change the font and size in the lefthand view to make editing most comfortable for you.

→ [MultiParse](#) → [Colour keys](#) → [Find](#) → [Word frequency counts](#) → [General](#) → [Updates](#) → [Technical Information](#)

MultiParse

To examine your text in more detail, click on the **MultiParse** button and Nalaprop will parse it several times over to extract fuller details about each word within it. Again, this is currently performed in the main thread, and for longer texts the spinning beachball cursor may appear while parsing.

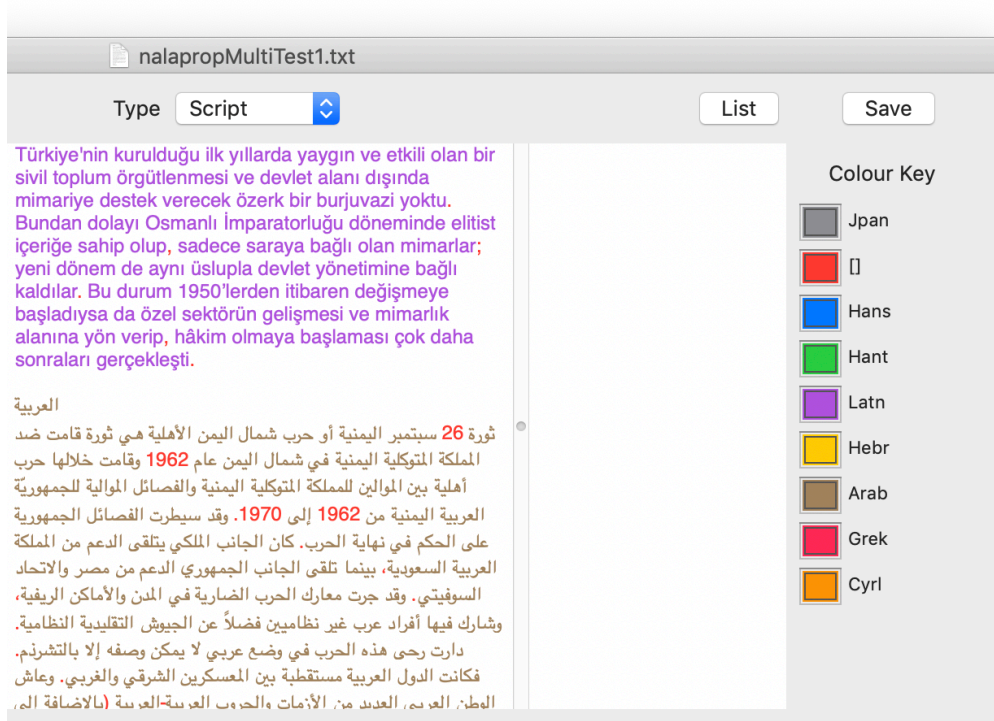


When you have MultiParsed the text, the **Type** popup menu is enabled, and offers five options for display of the centre pane:

- **Script** - this colours each word according to the script detected, e.g. Latn for Latin;
- **Language** - this colours each word according to the language detected, e.g. ru for Russian;
- **Types** - this colours each word according to part of speech using an arbitrary scheme derived from the types of word present;
- **Types (std)** - this colours each word by part of speech according to Nalaprop's standard scheme, to aid analysis;
- **Lemma** - this gives a stem or base form of those words which it can, e.g. in French, *le*, *la* and *les* are given the lemma *le*. Those stems are then coloured using the same scheme as Types. This option loses punctuation and text layout but can be valuable when reading a language which you know less well.

→ [Script](#) → [Language](#) → [Types](#) → [Types \(std\)](#) → [Lemmas](#)

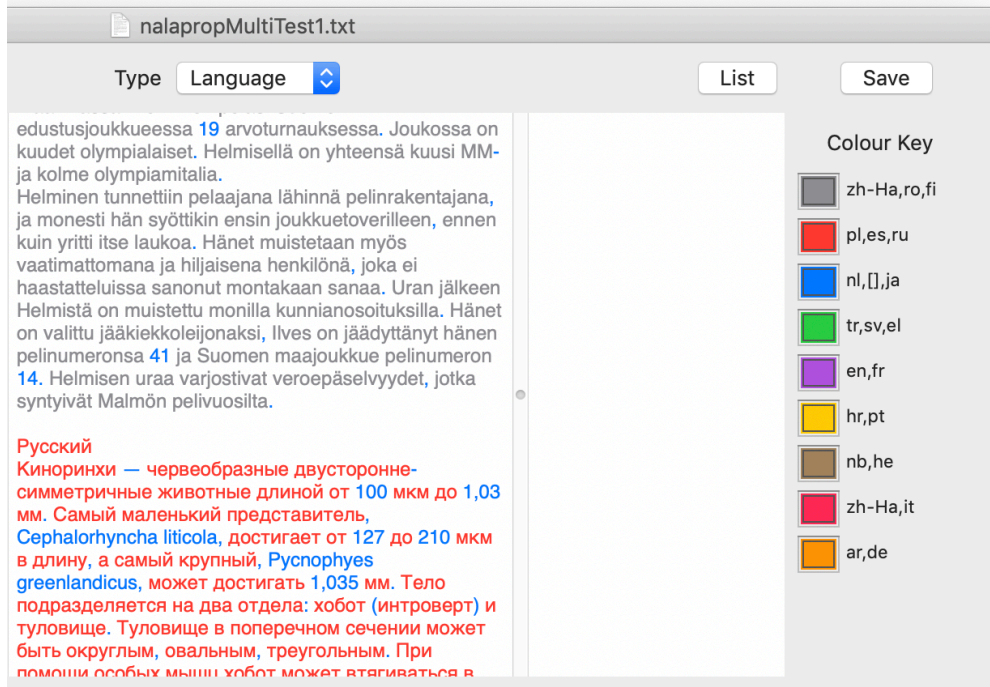
Script



The Script setting distinguishes each underlying script, here Latin and Arabic.

→ [Language](#) → [Types](#) → [Types \(std\)](#) → [Lemmas](#)

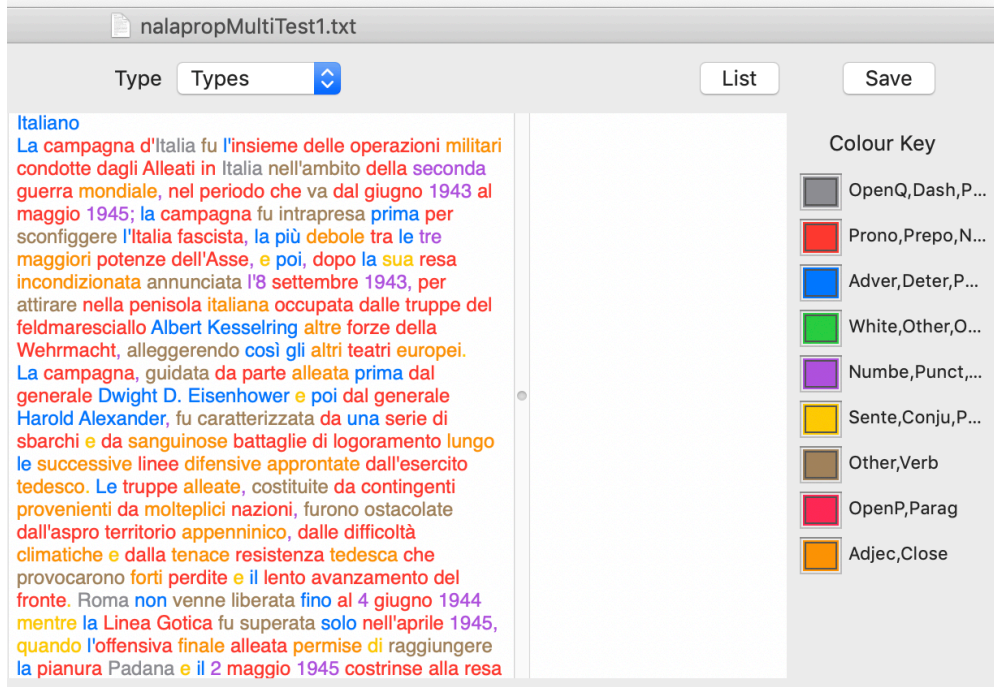
Language



The Language setting distinguishes each language. Note that numerals are given a type of [], indicating that they are independent of the underlying language.

→ [Script](#) → [Types](#) → [Types \(std\)](#) → [Lemmas](#)

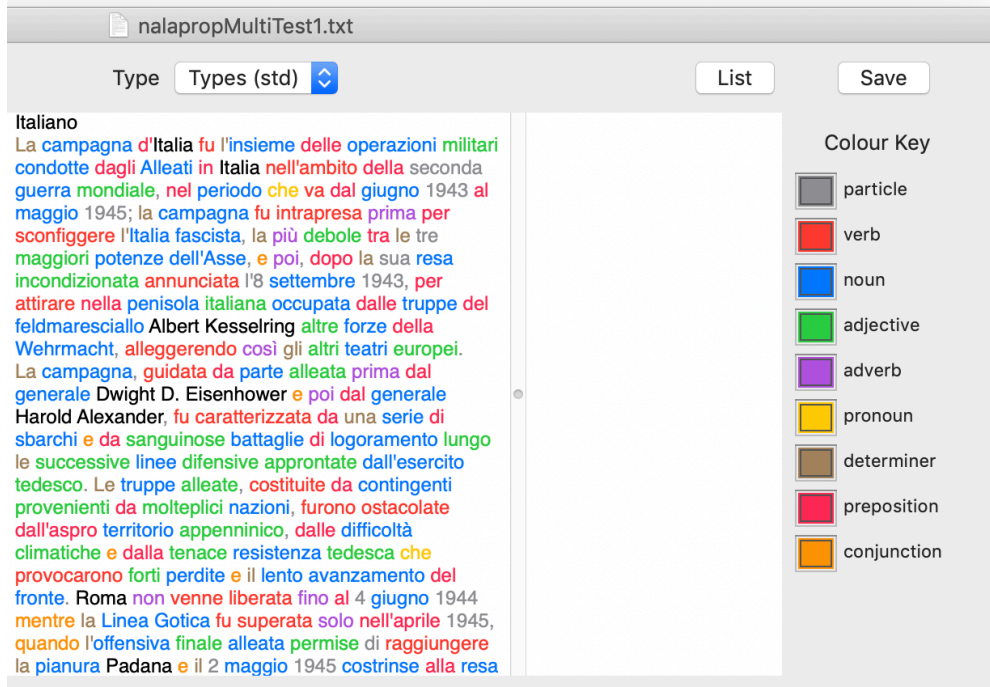
Types



The Types setting colours each word according to its lexical type (part of speech), using an arbitrary colour scheme. The colour key currently doesn't assign all the types detected.

→ [Script](#) → [Language](#) → [Types \(std\)](#) → [Lemmas](#)

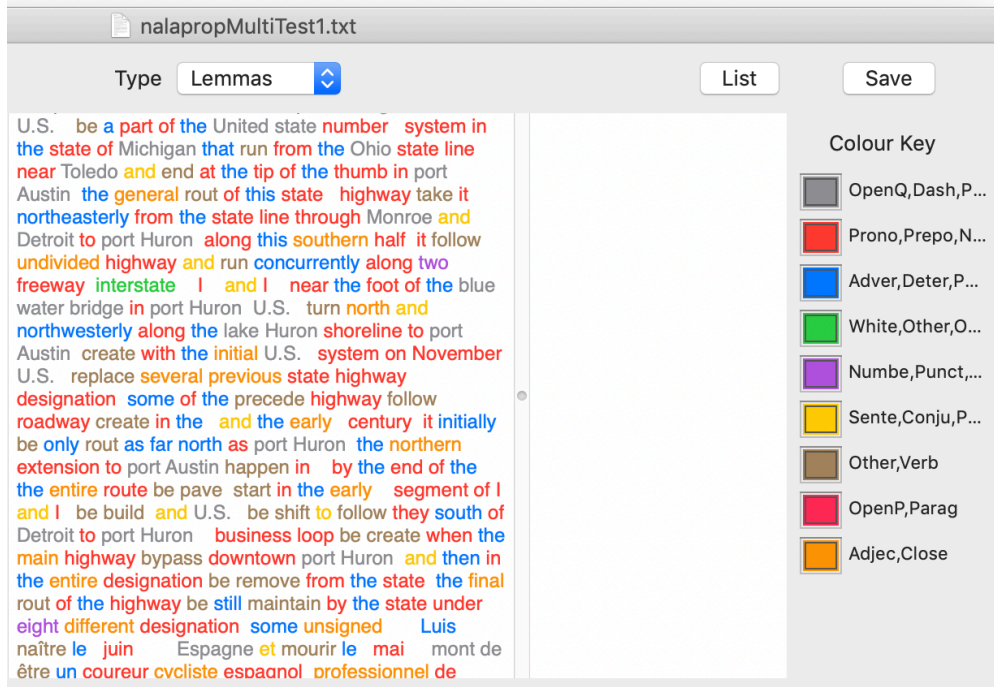
Types (std)



The Types (std) setting colours each word according to its lexical type, using an optimised colour scheme intended to make grammatical structure clear.

→ [Script](#) → [Language](#) → [Types](#) → [Lemmas](#)

Lemmas

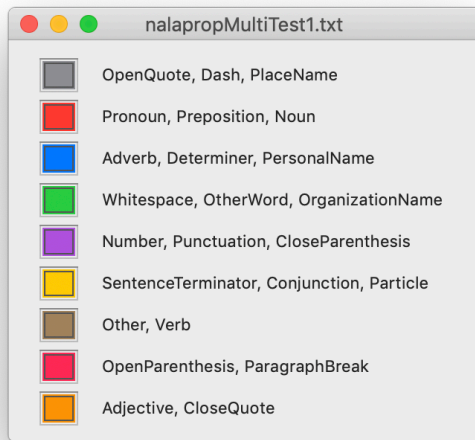


The Lemmas setting shows the ‘root’ forms of words which it knows, coloured according to the Types scheme.

→ [Script](#) → [Language](#) → [Types](#) → [Types \(std\)](#)

Colour keys

The Colour Key shown in the main window gives an abbreviated list of the types corresponding to the colours, which should be sufficient for examining the script and language. In most samples of text there are more than nine word types which are detected. To make it easier to see the full colour key, use the **Custom colour key** command in the **Help** menu. This opens an accessory window showing in full all the current colour assignments for the window in which you last clicked the **MultiParse** button.



Because you can have many of these palette windows open at any time, the title of each matches that of its parent window. The key given within it remains frozen from the time that the window is opened, and shows the key for the current selected Type in that window. This may seem complicated, but gives you great flexibility in the colour keys that you have on display.

→ [Colour keys \(continued\)](#)

Colour keys (continued)

Nalaprop re-uses existing colours in a circular fashion for those additional types: the tenth word type, for example, uses the first colour again, and the eleventh the second colour. If you want to examine the lexical classes or parts of speech, you should find the **Types (std)** menu option preferable, as it allocates colours on an optimised and pre-determined basis. However, it doesn't offer as many classes as the other **Types** option.

The standard colour scheme for showing parts of speech (**Types (std)** in the menu, and that shown when you first open a document or click on the Parse button) uses the following colour mappings:

- nouns, blue
- verbs, red
- adjectives, green
- adverbs, purple
- pronouns, yellow
- determiners, brown
- particles, grey
- numbers, grey
- prepositions, pink
- conjunctions, orange
- classifiers, orange
- idioms, green
- interjections, orange
- personal, organisational, and place names, white or black (according to Light/Dark mode at the time)
- others, including punctuation and any unparsed text, grey.

You can view this colour key in full detail using the **Standard colour key** command in the **Help** menu. This opens a fixed-size window showing the colours and their significance.

→ [Colour keys \(concluded\)](#)

Colour keys (concluded)



Nalprop can fully parse English, French, Spanish, German, Italian, Portuguese, Russian, Turkish, and additional languages as supported by macOS. Many other languages are supported in part, and it can recognise most scripts in general use, and many languages. Support for lexical class analysis is currently limited to those eight languages, and for lemmas in even less.

→ [Word frequency counts](#)

Find

Nalaprop supports multiple Find features which effectively turn it into an interactive concordance. Because each of the three text views has quite different contents, Find works in two slightly different ways according to which is active.

To search the text in the **editable view at the left**, click on the text within it and use the **Find** command in the **Edit** menu. This adds a standard Find bar to the top of that view, into which you can type the text which you want to locate.

To search the **parsed text in the centre** view, which you cannot edit, click in that view and use the **Find** command in the **Edit** menu. This opens the standard Find window with its full set of tools.

To search the **wordlists in the righthand** view, which you also cannot edit, click in that view and use the **Find** command in the **Edit** menu. This adds a standard Find bar to the top of that view, into which you can type the text which you want to locate.

If you already have a **Find** window open, you can click in any of the three text views to perform that search within that view. You can't, though, use either Find bar to search either of the other text views.

To use Nalaprop as a concordance, select a word from the wordlists at the right, copy and paste it into the Find window. Then click in the centre view, and find each of its occurrences. If you want to search for lemmatised forms, remember to switch the **Types** menu to **Lemmas** first.

→ [Word frequency counts](#)

Word frequency counts

To perform word frequency analysis on the parsed text, click on the **List** button at the right. This runs through the parsed text, and displays the numbers of word occurrences according to the lexical class of the word.

Two types of frequency analysis are available: when you have just clicked on the **Parse** button, or have just read in a document, simple analysis is performed on the words found in the text. In this, each occurrence of *is*, for example, is counted as the verb form *is*.

The screenshot shows a software window titled 'ChristmasCarol.txt'. It features a 'Type' dropdown menu set to 'Types', a 'List' button, and a 'Save' button. The main area displays the text of 'A CHRISTMAS CAROL' by Charles Dickens, with words color-coded by their lexical class. A 'Colour Key' on the right lists the classes: particle (grey), verb (red), noun (blue), adjective (green), adverb (purple), pronoun (yellow), determiner (brown), preposition (pink), and conjunction (orange). A list of words and their counts is shown on the right side of the window, including 'bade (1)', 'baking (1)', 'balancing (1)', 'bared (1)', 'barred (2)', 'basin (1)', 'basking (1)', 'battled (1)', 'be (143)', 'beaming (2)', 'bear (5)', 'beat (4)', 'beaten (1)', 'beating (1)', 'became (10)', 'beckoned (1)', 'become (3)', 'becoming (2)', 'been (67)', 'beetling (1)', 'began (8)', 'begged (3)', 'begin (2)', 'beginning (1)', and 'begins (1)'.

When you have just clicked on the **MultiParse** button and the **Type** menu is enabled, the analysis is performed on *lemmatised* versions of each word, where they are available. So each occurrence of *is* will be counted as an occurrence of the lemma *be*. Where a lemmatised form is not available for a word, the word itself is used instead.

→ [Word frequency counts \(concluded\)](#)

Word frequency counts (concluded)

The screenshot shows a software window titled 'ChristmasCarol.txt'. It has a 'Type' dropdown menu set to 'Types' and two buttons: 'List' and 'Save'. The main area is divided into three sections:

- Left Panel:** Contains the text of 'A CHRISTMAS CAROL' and its 'PREFACE'. The text is color-coded by parts of speech. For example, in the preface, 'I' is blue, 'HAVE' is green, 'endeavoured' is red, 'in' is blue, 'this' is green, 'Ghostly' is red, 'little' is blue, 'book,' is green, 'to' is blue, 'raise' is red, 'the' is blue, 'Ghost' is red, 'of' is blue, 'an' is green, 'Idea,' is red, 'which' is blue, 'shall' is green, 'not' is red, 'put' is blue, 'my' is green, 'readers' is red, 'out' is blue, 'of' is green, 'humour' is red, 'with' is blue, 'themselves,' is red, 'with' is blue, 'each' is green, 'other,' is red, 'with' is blue, 'the' is green, 'season,' is red, 'or' is blue, 'with' is green, 'me.' is red, 'May' is blue, 'it' is green, 'haunt' is red, 'their' is blue, 'houses' is green, 'pleasantly,' is red, 'and' is blue, 'no' is green, 'one' is blue, 'wish' is red, 'to' is blue, 'lay' is green, 'it.' is red.
- Middle Panel:** A list of words and their frequencies, sorted alphabetically. The list includes: bake (1), balance (1), bar (2), bare (1), basin (1), bask (1), battle (1), be (1098), beam (2), bear (15), beat (6), beckon (1), become (15), beetle (1), beg (3), begin (14), beguile (1), behave (1), behold (4), believe (21), belong (4), bend (1), benefit (1), beseech (1), beseechina (1).
- Right Panel:** A 'Colour Key' with color-coded boxes and labels for parts of speech: Conju, Other, N... (grey), Deter, Organ, P... (red), Inter, Close, Op... (blue), Sente, Parag, A... (green), Adjec, Place, Ve... (purple), Other, White, N... (yellow), OpenP, Close, P... (brown), Dash, Perso (pink), and Punct, Prono (orange).

To switch between the two forms of analysis, use the **Parse** (for regular) and **MultiParse** (for lemmatised) buttons before clicking on **List**.

Word frequency analysis shown at the right is performed as follows:

- every word parsed into one of the major lexical classes is sorted into its class
- with the exception of names (which are left with case unchanged), those words are reduced to lower case
- an alphabetical listing of words is then given, with the number of occurrences of that word in the text, for each lexical class in turn (in the case of lemmatised forms, the lemma is substituted for the word, when available)
- at the end of each class listing, the total number of words of that class found in the the text is given
- at the end of all the class listings, the total number of words assigned to those classes is given.

At the moment, words from different languages which are used in the text are pooled together.

General

The **Save...** command has been disabled to help prevent you from overwriting original files. When you **Save as...**, select the Rich text (RTF) option to preserve colour coding in the parsed output. Saving only saves the text displayed in the centre panel, not the original text in the lefthand one. To save the frequency analysis in the righthand panel, use the **Save** button above it.

To print the text in any of the three panels, click on the text in that panel so that the insertion cursor is within it, then use the **Print** command.

The text file nalapropMultiTest1.txt contains sample passages of twenty different languages, copied from Wikipedia's localised versions, for which I am very grateful to Wikipedia.

→ [Technical information](#)

Updates

Whenever you open Nalaprop, it may check to see if an update is available. This *doesn't* use the popular Sparkle mechanism for updating in place, but works as detailed here.

Once Nalaprop has successfully completed its integrity check, it checks whether update checking has been turned off in its preferences file. If that has, it abandons any attempt to check for updates. If checking is allowed, it then checks when it last checked for updates. If that was more than 12 hours ago, it continues to perform the check. It then connects to my GitHub server, from where it downloads a list of current versions of my apps. It doesn't upload any data to the GitHub server at all, and no statistics beyond GitHub normal connection figures are collected either: no personal identifiers are recorded. If there is an update available, Nalaprop then checks that its location is on this WordPress blog, and posts a dialog which invites you to download the update.

If you click on the **Download** button, it then points your default browser at that update, which should trigger the update to be downloaded to your normal downloads folder. The update is received as a regular Zip archive, and is exactly the same as you would download from the Downloads page here. It also carries a quarantine flag, so that when you unZip it and install the app inside, it undergoes normal first run 'Gatekeeper' security checks. If you click on the **Ignore** button, Nalaprop won't remind you about it again for another 12 hours.

An additional item at the end of the **Help** menu explains the update status. If no update check is performed, or the check fails, the last item reads **Update not checked**. If the check is performed and update information is obtained, even when no update is available or you decline to download it, that menu item reads **Checked for update** and is ticked (but still disabled).

You can customise this behaviour by changing Nalaprop's preferences. The keys to use are:

- `noUpdateCheck`, a Boolean. When set to `true`, this disables all update checking. Default is `false`.
- `updateCheckInt`, a real number (Double). When set to a value greater than 1.0, the minimum time interval between checks, in seconds. Default is 43200, which is 12 hours. If you set it to any value less than 1, Nalaprop will reset it automatically to that default.

To change either of these, use a Terminal command of the form

```
defaults write co.electiclight.Nalaprop updateCheckInt '10'
```

which works properly through the preferences server `cfprefsd`.

Technical Information

Swift 5 code used to achieve the parsing includes:

```
import NaturalLanguage
let theMASOut = NSMutableAttributedString()
let tagger = NLTagger(tagSchemes: [.nameTypeOrLexicalClass])
tagger.string = theStr
let options: NLTagger.Options = []
tagger.enumerateTags(in: theStr.startIndex..
```

Why the name Nalaprop? It's a pun on Mrs Malaprop, and **Natural Language Processing**.

→ [Change list](#)

Change list

1.0 release:

- Universal App, first full release.

1.0b10:

- added Help
- added code integrity check each time it's opened
- added auto-update support
- added product support link
- ported to Swift 5 in Xcode 10.2.1.

1.0b9:

- added support for Find in each of the three text views
- added palette view for colour keys to script, language, lexical class
- miscellaneous interface tweaks.

1.0b8:

- added Cmd-Enter shortcut to Parse button to make parse-while-edit simple.

1.0b7 (unreleased):

- reorganised button logic and added List button
- ported to Swift 4.2.1 in Xcode 10.1
- improved labels on colour swatches
- added lemmatised wordlist.

1.0b6:

- major structural changes: nows opens with parsed Types view
- Parse an opened document now only adds wordlists
- MultiParse to support multiple parsed views, e.g. Script, Language
- added in-window colour key
- added internal multi-tagged version of text to support these.

1.0b5:

- fixed typos in the word frequency analyses.

1.0b4 (unreleased):

- added third scrolling text view
- added classified word frequency analysis to righthand view, and Save button.

1.0b3:

- tested against 20 different language passages, from Arabic to Turkish (included)
- added Colour Key help window.

1.0b2:

- added support for opening RTF files as plain text
- enabled printing support
- built using Xcode 10.0 release.

1.0b1:

- tidied the interface, removing superfluous controls
- notarized for Mojave
- built using Xcode 10β5.

1.0a2:

- fixed a crash when language not recognised/supported yet
- added second stage parsing with popup menu.

1.0a1:

- initial release
- crashes when trying to parse Chinese.

17 August 2020.