String Operations - {sto}

The goal of {sto}[1] is to make easier some of the tedious operations involving strings and text analysis. It is under development, available on GitHub. The initial reason for this package was to easily create vectors of strings, like when creating stopword. It comes with three main types of functions: 1) to quickly create vectors from strings, load a series os packages more easily, and perform other operations like `grep`, `grepl` and `gsub` with native pipe. It also allows to concatenate strings in the same way as in Python `f()`. 2) Most of the existing stopwords packages only provides a vector of words, but with {sto} also provides functions for fine grain generate stopword lists by language and grammar categories. 3) Additionally, {sto} comes with a rule-based proper name/entity extractor that returns vectors, lists, or graphs representing binary co-occurrence in sentences or paragraphs, as well a function to plot it, according to frequency of terms and frequency of links between terms.

The initial reason for this package was to have an easy way to create vector of strings, like when creating stopwords or loading packages. It can be done with `s2v()`.

command	output	object type
s2v("a b c d")	"a" "b" "c" "d"	vector
s2v("a b c d", print = TRUE)	"c('a', 'b', 'c', 'd')"	string

To load many packages, instead of using `library(package_n)` for every package name, use:

```r
sto::ll("sto dplyr ggplot2 stringr tidyr")
```

To concatenate string like in Python, use `f()` and the variable inside curly brackets `{var}`.

```r
var1 <- 912 * 2
lorem <- "Lorem ipsum dolor sit amet"
f("Bla: {var1}. Ble: {lorem}") # concatenating the strings above
```

[1] https://soaresalisson.github.io/sto/

```
Bla: 1824. Ble: Lorem ipsum dolor sit amet
```

To use grep, grepl, gsub in a pipe, just add a “2” at the end of its names. Instead of using

command	output
`s2v("a b c d") |> grep2("a")`	`"a"`
`s2v("a b c d") |> grepl2("a")`	`TRUE FALSE FALSE FALSE`
`"a b c d" |> gsub2("a", "x")`	`"x b c d"`
`"a b c d" |> gsub2("a ")`	`"b c d"`

To generate a fine grained vector of stopwords, using grammatical categories:

```r
gen_stopwords(lang = "en", categories = "CC DT V")
```

```
 [1] "and"   "but"   "or"    "for"   "yet"   "so"    "a"     "the"   "this"
[10] "am"    "are"   "is"    "be"    "can"   "could" "did"   "do"    "have"
[19] "he"    "it"    "may"   "might" "must"  "need"  "no"    "not"   "now"
[28] "of"    "on"    "she"   "that"  "to"    "was"   "were"
```

To extract graphs using co-ocurrence of propen names - based on regex -

```r
"John Does lives in New York in United States of America." |>
  extract_entity()
```

```
[1] "John Does"                "New York"
[3] "United States of America"
```

```r
"João Ninguém mora em São José do Rio Preto, mas esteve antes em Sergipe" |>
  extract_entity(connect = connectors("pt"))
```

```
[1] "João Ninguém"          "São José do Rio Preto" "Sergipe"
```

To render this data in a graph, with frequency of each word and frequency of each link, you can use `plot_graph()`. For a more detailed description of this and the another functions, please visit the webpage of the project