How to use langnetr: A simple example

CSQ Siew

2017-10-12

Vignette Info

The `langnetr` package was built to help the user easily convert lists of words into network objects based on an edit distance of 1. This could be useful to any researcher who is working with linguistic data and wants to build a network based on character similarity between words. Hence this would work for phonological transcriptions and orthographic representations, but not so if one were interested in semantic relationships between words. This vignette will demonstrate how to use the functions in this package using a simple example.

First things first

This package is not hosted on CRAN, so the easiest way to download it is by installing the `devtools` package and downloading the `langnetr` package from my github page. Here's how to do it:

```r
install.packages('devtools')
library(devtools)
devtools::install_github("csqsiew/langnetr")
library(langnetr)
```

Not too painful, eh?

The tolangnet function

Let's generate a few sample datasets.

(Note that for this example the network will be built based on orthographic similarity. Of course you can convert the words to some kind of machine readable transcriptions for phonological similarity instead. E.g. cat --> k@t)

```r
words   <- c('cat', 'bat', 'cap', 'cape', 'door', 'cup', 'cut')
hermits <- c('spinach', 'brocoli', 'kale')
numbers <- c(1:10)
```

It is important that the data with the list of words is a character vector for the tolangnet function to work.

```r
# numbers.net <- langnetr::tolangnet(numbers)
# if you ran it, it would return an error
```

If no network is formed then nothing happens except for a useful message.

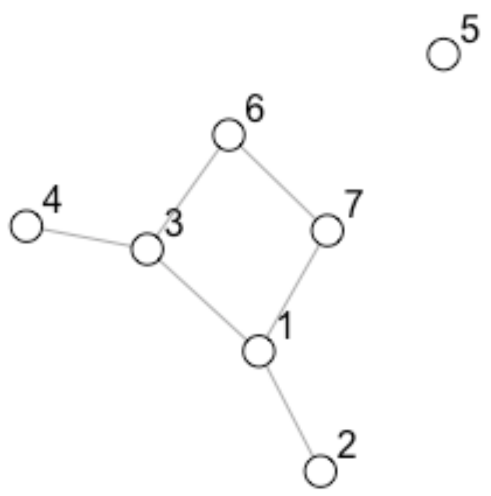
```r
hermits.net <- langnetr::tolangnet(hermits) # returns a message

## [1] "List of words given do not form a network. Might be hermits"
```

Make a language network!

```r
words.net <- langnetr::tolangnet(words)

plot(words.net,
     vertex.label.color='black',
     vertex.color = 'white',
     vertex.label.dist=2.5,
     vertex.label.family = 'Arial')
```



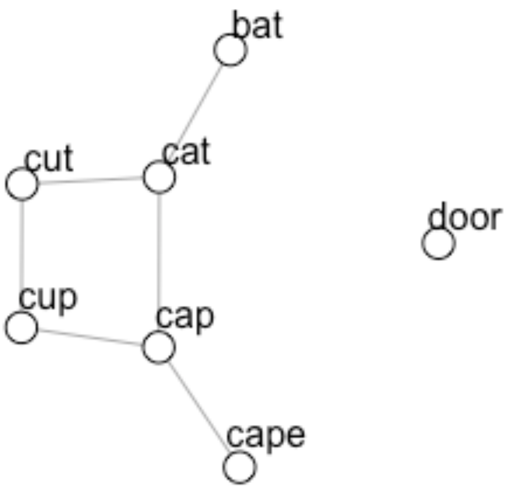
You can then analyze the network in R or output it into a different format to analyze in Pajek or Gephi. The `igraph` package has a bunch of useful functions for doing this, in particular the `write_graph` function link.

The nodeindex function

This function labels your nodes--which is more useful and informative than a bunch of numbers.

```
words.net.labels <- langnetr::nodeindex(words.net, words)

plot(words.net.labels,
     vertex.label.color='black',
     vertex.color = 'white',
     vertex.label.dist=2.5,
     vertex.label.family = 'Arial')
```



The "helper" functions

The `toedgelist` function returns a dataframe of the *labelled* edgelist of the language network (the standard `as_edgelist` function in `igraph` returns an edgelist with node IDs).

The `hermits` function returns a list of hermits from the language network.

The `giantc` function returns a list of nodes found in the giant component (largest connected component) of the language network.

These smaller functions just help make life a bit easier for a language researcher :)

```
words.net.edgelist <- langnetr::toedgelist(words.net.labels)
words.net.hermits <- langnetr::hermits(words.net.labels)
words.net.giantc <- langnetr::giantc(words.net.labels)
words.net.edgelist

##   word1 word2
## 1   cat   bat
## 2   cat   cap
## 3   cat   cut
## 4   cap  cape
## 5   cap   cup
## 6   cup   cut

words.net.hermits

## [1] "door"

words.net.giantc

## [1] "cat"  "bat"  "cap"  "cape" "cup"  "cut"
```

Comments, suggestions, bugs?

Email me at cynsiewsq at gmail dot com - I would love to hear from you! :)