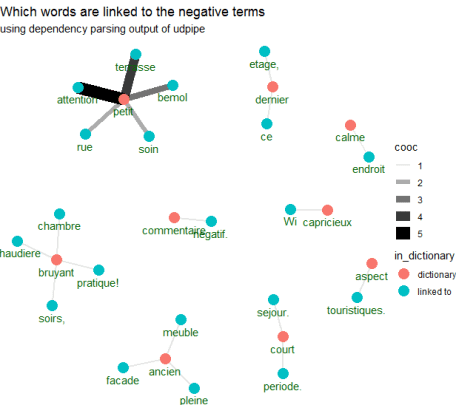
A small note on the growing list of users of the udpipe R package. In the last month of 2018, the package on CRAN with some noticeable changes:

* The default models which are now downloaded with the function udpipe\_download\_model are now models built on Universal Dependencies 2.3 (released on 2018-11-15)
* This means udpipe now has **models for 60 languages**. That’s right! And they provide tokenisation, parts of speech tagging, lemmatisation and dependency parsing built on all of these treebanks: afrikaans-afribooms, ancient\_greek-perseus, ancient\_greek-proiel, arabic-padt, armenian-armtdp, basque-bdt, belarusian-hse, bulgarian-btb, buryat-bdt, catalan-ancora, chinese-gsd, coptic-scriptorium, croatian-set, czech-cac, czech-cltt, czech-fictree, czech-pdt, danish-ddt, dutch-alpino, dutch-lassysmall, english-ewt, english-gum, english-lines, english-partut, estonian-edt, finnish-ftb, finnish-tdt, french-gsd, french-partut, french-sequoia, french-spoken, galician-ctg, galician-treegal, german-gsd, gothic-proiel, greek-gdt, hebrew-htb, hindi-hdtb, hungarian-szeged, indonesian-gsd, irish-idt, italian-isdt, italian-partut, italian-postwita, japanese-gsd, kazakh-ktb, korean-gsd, korean-kaist, kurmanji-mg, latin-ittb, latin-perseus, latin-proiel, latvian-lvtb, lithuanian-hse, maltese-mudt, marathi-ufal, north\_sami-giella, norwegian-bokmaal, norwegian-nynorsk, norwegian-nynorsklia, old\_church\_slavonic-proiel, old\_french-srcmf, persian-seraji, polish-lfg, polish-sz, portuguese-bosque, portuguese-br, portuguese-gsd, romanian-nonstandard, romanian-rrt, russian-gsd, russian-syntagrus, russian-taiga, sanskrit-ufal, serbian-set, slovak-snk, slovenian-ssj, slovenian-sst, spanish-ancora, spanish-gsd, swedish-lines, swedish-talbanken, tamil-ttb, telugu-mtg, turkish-imst, ukrainian-iu, upper\_sorbian-ufal, urdu-udtb, uyghur-udt, vietnamese-vtb.
* Although this was not intended originally we **added a sentiment scoring function** in the latest release (version 0.8 on CRAN). Combined with the output of the dependency parsing, this allows to answer questions like ‘WHAT IS CAUSING A NEGATIVE SENTIMENT’. Example showing below.
* If you want to use the udpipe models for commercial purposes, we have some nice extra pretrained models available for you

**Below we will showcase the new features of the R package by finding out what is causing a negative sentiment.**

*If I see some users of the* [*tidytext*](https://cran.r-project.org/package=tidytext) *sentiment R package I always wondered if they do sentiment scoring for the love of building reports as it looks like the main thing they report is frequency of occurrences of words which are part of a positive or negative dictionary. While probably their manager asked them. “Yeah but why is the sentiment negative or positive”.****You can answer this managerial question using dependency parsing*** *and that is exactly what udpipe provides (amongst other NLP annotations). Dependency parsing links each word to another word, allowing us the find out which words are linked to negative words giving you the context of why something is negative and what needs to be improved in your business. Let’s show how to get this easily done in R.*

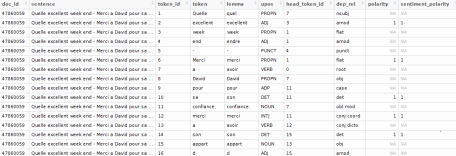
Below we get a sample of 500 AirBnb customer reviews in French, annotate it with, use the new sentiment scoring txt\_sentiment which is available in the new udpipe release using an online dictionary of positive / negative terms for French. Next we use the udpipe dependency parsing output by looking to the adjectival modifier ‘amod’ in the dep\_rel udpipe output and visualise all words which are linked the the negative terms of the dictionary. The result is this graph showing words of the dictionary in red and words which are linked to that word in another color.



Full code showing how this is done is shown below.

library(udpipe)  
library(dplyr)  
library(magrittr)  
data(brussels\_reviews, package = "udpipe")  
x <- brussels\_reviews %>%   
  filter(language == "fr") %>%  
  rename(doc\_id = id, text = feedback) %>%  
  udpipe("french-spoken", trace = 10)  
##  
## Get a French sentiment dictionary lexicon with positive/negative terms, negators, amplifiers and deamplifiers  
##  
load(file("<https://github.com/sborms/sentometrics/raw/master/data-raw/FEEL_fr.rda>"))  
load(file("<https://github.com/sborms/sentometrics/raw/master/data-raw/valence-raw/valShifters.rda>"))  
polarity\_terms <- rename(FEEL\_fr, term = x, polarity = y)  
polarity\_negators <- subset(valShifters$valence\_fr, t == 1)$x  
polarity\_amplifiers <- subset(valShifters$valence\_fr, t == 2)$x  
polarity\_deamplifiers <- subset(valShifters$valence\_fr, t == 3)$x  
##  
## Do sentiment analysis based on that open French lexicon  
##  
sentiments <- txt\_sentiment(x, term = "lemma",   
                            polarity\_terms = polarity\_terms,  
                            polarity\_negators = polarity\_negators,   
                            polarity\_amplifiers = polarity\_amplifiers,  
                            polarity\_deamplifiers = polarity\_deamplifiers)  
sentiments <- sentiments$data

* Nothing fancy happened here above. We use udpipe for NLP annotation (tokenisation, lemmatisation, parts of speech tagging and dependency parsing). The sentiment scoring not only does a join with the sentiment dictionary but also looks for neighbouring words which might change the sentiment.
* The resulting dataset looks like this



**Now we can answer the question – why is something negative**

This is done by using the dependency relationship output of udpipe to find out which words are linked to negative words from our sentiment dictionary. Users unfamiliar with dependency relationships, have a look at definitions of possible tags for the dep\_rel field. In this case we only take ‘amod’ meaning we are looking for adjectives modifying a noun.

## Use cbind\_dependencies to add the parent token to which the keyword is linked  
reasons <- sentiments %>%   
  cbind\_dependencies() %>%  
  select(doc\_id, lemma, token, upos, sentiment\_polarity, token\_parent, lemma\_parent, upos\_parent, dep\_rel) %>%  
  filter(sentiment\_polarity < 0)  
head(reasons)

* Now instead of making a plot showing which negative words appear which tidytext users seem to be so keen of, we can make a plot showing the negative words and the words which these negative terms are linked to indicating the context of the negative term.
* We select the lemma’s of the negative words and the lemma of the parent word and calculate how many times they occur together

reasons <- filter(reasons, dep\_rel %in% "amod")  
word\_cooccurences <- reasons %>%   
  group\_by(lemma, lemma\_parent) %>%  
  summarise(cooc = n()) %>%  
  arrange(-cooc)  
vertices <- bind\_rows(  
  data\_frame(key = unique(reasons$lemma)) %>% mutate(in\_dictionary = if\_else(key %in% polarity\_terms$term, "in\_dictionary", "linked-to")),  
  data\_frame(key = unique(setdiff(reasons$lemma\_parent, reasons$lemma))) %>% mutate(in\_dictionary = "linked-to"))

* The following makes the visualisation using ggraph.

library(magrittr)  
library(ggraph)  
library(igraph)  
cooc <- head(word\_cooccurences, 20)  
set.seed(123456789)  
cooc %>%    
  graph\_from\_data\_frame(vertices = filter(vertices, key %in% c(cooc$lemma, cooc$lemma\_parent))) %>%  
  ggraph(layout = "fr") +  
  geom\_edge\_link0(aes(edge\_alpha = cooc, edge\_width = cooc)) +  
  geom\_node\_point(aes(colour = in\_dictionary), size = 5) +  
  geom\_node\_text(aes(label = name), vjust = 1.8, col = "darkgreen") +  
  ggtitle("Which words are linked to the negative terms") +  
  theme\_void()

This generated the image shown above, showing context of negative terms. Now go do this on your own data.