POS-tagging with UDPipe

Filippo Chiarello, Ph.D.

UDPipe Introduction

- UDPipe is an R package which is an Rcpp wrapper around the UDPipe C++ library
- UDPipe provides language-agnostic tokenization, tagging, lemmatization and dependency parsing of raw text, which is an essential part in natural language processing.
- The techniques used are explained in detail in the paper: "Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe", available at http://ufal.mff.cuni.cz/~straka/papers/2017-conll_udpipe.pdf. I
- In that paper, you'll also find accuracies on different languages and process flow speed (measured in words per second).

General Features

The udpipe R package was designed with the following things in mind when building the Rcpp wrapper around the UDPipe C++ library:

- Give R users simple access in order to easily tokenize, tag, lemmatize or perform dependency parsing on text in any language
- Provide easy access to pre-trained annotation models
- Allow R users to easily construct your own annotation model based on data in CONLL-U format as provided in more than 100 treebanks available at http://universaldependencies.org/#ud-treebanks
- Don't rely on Python or Java so that R users can easily install this package without configuration hassle
- No external R package dependencies except the strict necessary (Rcpp and data.table, no tidyverse)

Installation & License

The package is available under the Mozilla Public License Version 2.0. Installation can be done as follows. Please visit the package documentation at https://bnosac.github.io/udpipe/en and look at the R package vignettes for further details.

```
# install.packages("udpipe")
# vignette("udpipe-tryitout", package = "udpipe")
# vignette("udpipe-annotation", package = "udpipe")
# vignette("udpipe-universe", package = "udpipe")
# vignette("udpipe-usecase-postagging-lemmatisation", package = "udpipe")
# # An overview of keyword extraction techniques: https://bnosac.github.io/udpipe/docs/doc7.htm
# vignette("udpipe-usecase-topicmodelling", package = "udpipe")
# vignette("udpipe-parallel", package = "udpipe")
# vignette("udpipe-train", package = "udpipe")
```

A first example

Currently the package allows you to do tokenisation, tagging, lemmatization and dependency parsing with one convenient function called udpipe.

```
library(udpipe)
```

```
## Warning: package 'udpipe' was built under R version 4.0.5
```

```
udmodel <- udpipe_download_model(language = "english")
# udmodel
```

```
x <- udpipe(x = "Competitive intelligence (CI) is the process and forward-looking
            practices used in producing knowledge about the competitive environment
            to improve organizational performance.",
            object = udmodel)
```

Inspect the content

```
str(x)
```

```
## 'data.frame':    26 obs. of  17 variables:
##  $ doc_id       : chr  "doc1" "doc1" "doc1" "doc1" ...
##  $ paragraph_id : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ sentence_id  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ sentence     : chr  "Competitive intelligence (CI) is the process and forward-looking pra
##  $ start        : int  1 13 26 27 29 31 34 38 46 50 ...
##  $ end          : int  11 24 26 28 29 32 36 44 48 56 ...
##  $ term_id      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ token_id     : chr  "1" "2" "3" "4" ...
##  $ token        : chr  "Competitive" "intelligence" "(" "CI" ...
##  $ lemma        : chr  "competitive" "intelligence" "(" "CI" ...
##  $ upos         : chr  "ADJ" "NOUN" "PUNCT" "PROPN" ...
##  $ xpos         : chr  "JJ" "NN" "-LRB-" "NNP" ...
##  $ feats        : chr  "Degree=Pos" "Number=Sing" NA "Number=Sing" ...
##  $ head_token_id: chr  "2" "8" "2" "2" ...
##  $ dep_rel      : chr  "amod" "nsubj" "punct" "appos" ...
##  $ deps         : chr  NA NA NA NA ...
##  $ misc         : chr  NA NA "SpaceAfter=No" "SpaceAfter=No" ...
```

Pre-trained models

- Pre-trained models build on Universal Dependencies treebanks are made available for more than 65 languages based on 101 treebanks.

- These have been made available easily to users of the package by using udpipe_download_model

How good are these models?

- Accuracy statistics of models provided by the UDPipe authors which you download with udpipe_download_model from the default repository are available at this link.

- Accuracy statistics of models trained using this R package which you download with udpipe_download_model from the bnosac/udpipe.models.ud repository are available at https://github.com/bnosac/udpipe.models.ud.

- For a comparison between UDPipe and spaCy visit https://github.com/jwijffels/udpipe-spacy-comparison

UDPipe - Basic Analytics

In order to get the most out of the package, let's enumerate a few things one can now easily do with your text annotated using the udpipe package using merely the Parts of Speech tags & the Lemma of each word.

Improved exploratory text visualisations

- Due to richer features
- Allowing to select easily words which you like to plot (e.g. nouns/adjectives or the subject of the text)
- look for co-occurrences between words which are relevant based on the POS tag
- look for correlations between words which are relevant based on the POS tag

Easy summarisation of text

- automatic keyword detection
- noun phrase extraction or chunking
- automatic text summarisation (e.g. using the textrank R package)

Improved topic modelling by

- taking only words with specific parts-of-speech tags in the topic model
- automation of topic modelling for all languages by using the right pos tags instead of working with stopwords
- using lemmatisation as a better replacement than stemming in topic modelling

Further processing

- Improved sentence or document similarities by using only the words of a specific POS tag
- Identification of authors based on grammatical patterns used

Let's Apply the tools

Let's start by reading some text. We have a set of patents on AI.

```
library(udpipe)
library(tidyverse)
```

```
## Warning: package 'tidyr' was built under R version 4.0.5
```

```
## Warning: package 'readr' was built under R version 4.0.5
```

```
## Warning: package 'dplyr' was built under R version 4.0.5
```

```
# Read in the folder of txt, saving in list_of_files all the names of the file we want to read
list_of_files <- list.files(path = "AI_patents_2020_2021_claim/",
                            pattern = "\\.txt$",
                            full.names = TRUE)
```

```r
# set how many patents we want to read (for time purposes)
n_patents <- 100

# Create an empty tibble, that will store all the information
patents <-tibble(id = rep("", n_patents),
                 title = rep("", n_patents),
                 abstract = rep("", n_patents))
```

Read in all files from a folder

```
for(i in 1:n_patents){

  raw_text <- read_file(list_of_files[[i]])

  patents[[i, 1]] <- str_remove_all(string = list_of_files[[i]],
                                pattern = "AI_patents_2020_2021_claim//|.txt")

  patents[[i, 2]] <- str_extract(string = raw_text,
                                pattern = "<title>\n(.*?)\n</title>") %>%
    str_remove_all("<title>\n|\n</title>")

  patents[[i, 3]] <- str_extract(string = raw_text,
                                pattern = "<abstract>\n(.*?)\n</abstract>") %>%
    str_remove_all("<abstract>\n|\n</abstract>")

}
```

Have a look at the table

```
head(patents)
```

```
## # A tibble: 6 × 3
##   id           title                                     abstract
##   <chr>        <chr>                                     <chr>
## 1 EP1712182_B1 Method of ultrasonic detection and local… A metho…
## 2 EP1941301_B1 SYSTEM AND METHODS FOR ENHANCING AN IMAG… A syste…
## 3 EP2126563_B1 METHOD FOR DETERMINING HEALTH STATUS BY … A metho…
## 4 EP2169547_B1 Compilation model for programmable logic… The cla…
## 5 EP2240878_B1 IDENTIFICATION AND VERIFICATION OF AN UN… Techniq…
## 6 EP2252889_B1 METHOD AND SYSTEM FOR ANALYSIS OF FLOW C… An auto…
```

Tag the text

```r
ud_model <- udpipe_download_model(language = "english")
ud_model <- udpipe_load_model(ud_model$file_model)

patents_tagged <- udpipe_annotate(ud_model, x = patents$abstract, doc_id = patents$id) %>%
  as_tibble()
```

Inspect the results

The resulting data.frame has a field called upos which is the Universal Parts of Speech tag and also a field called lemma which is the root form of each token in the text. These 2 fields give us a broad range of analytical possibilities.

```
head(patents_tagged)
```

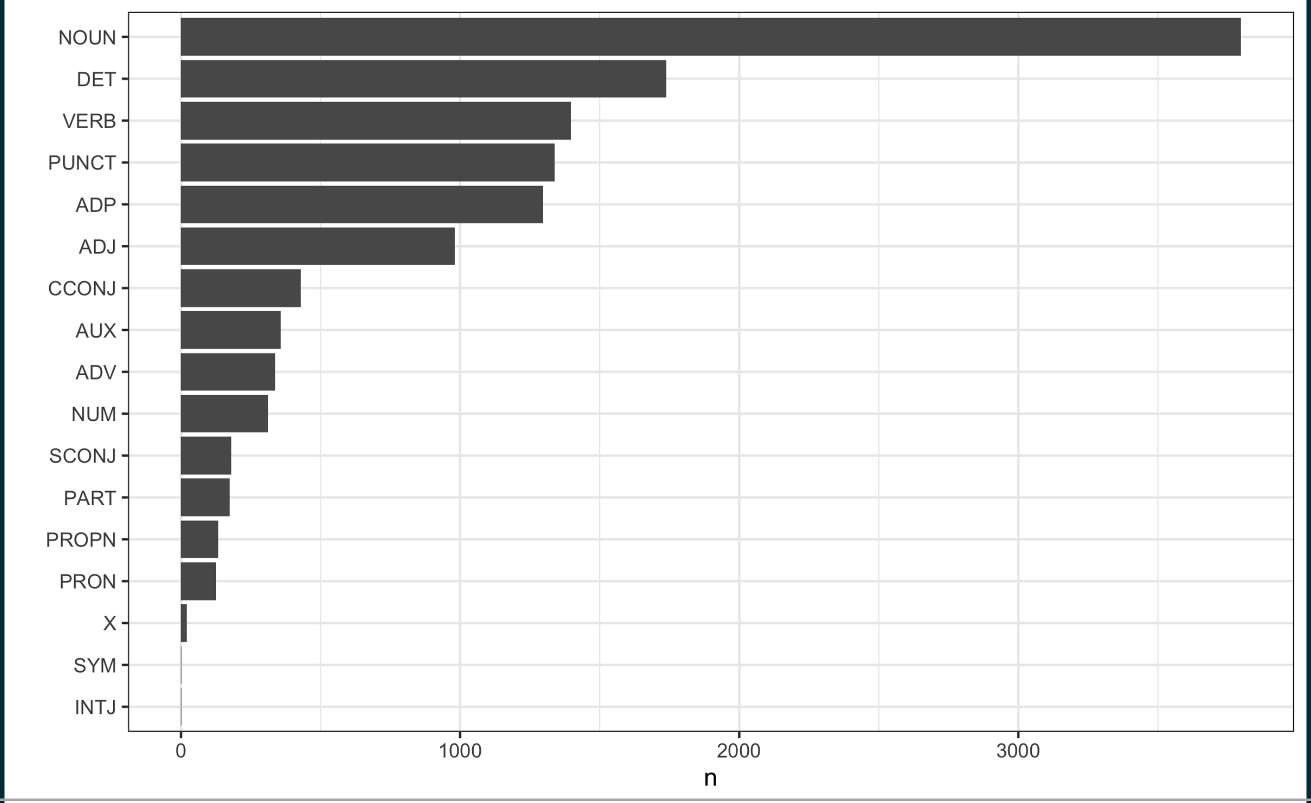
```
## # A tibble: 6 × 14
##   doc_id   paragraph_id sentence_id sentence token_id token lemma
##   <chr>           <int>       <int> <chr>    <chr>    <chr> <chr>
## 1 EP17121…            1           1 A metho… 1        A     a
## 2 EP17121…            1           1 A metho… 2        meth… meth…
## 3 EP17121…            1           1 A metho… 3        of    of
## 4 EP17121…            1           1 A metho… 4        ultr… ultr…
## 5 EP17121…            1           1 A metho… 5        dete… dete…
## 6 EP17121…            1           1 A metho… 6        and   and
## # … with 7 more variables: upos <chr>, xpos <chr>, feats <chr>,
## #   head_token_id <chr>, dep_rel <chr>, deps <chr>, misc <chr>
```

Basic frequency statistics

In most languages, nouns (NOUN) are the most common types of words, next to verbs (VERB) and these are the most relevant for analytical purposes, next to the adjectives (ADJ) and proper nouns (PROPN).

For a detailed list of all POS tags: visit https://universaldependencies.org/u/pos/index.html.

```
patents_tagged %>%
  count(upos) %>%
  ggplot(aes(x = reorder(upos, n), y = n)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  xlab("") +
  theme_bw()
```

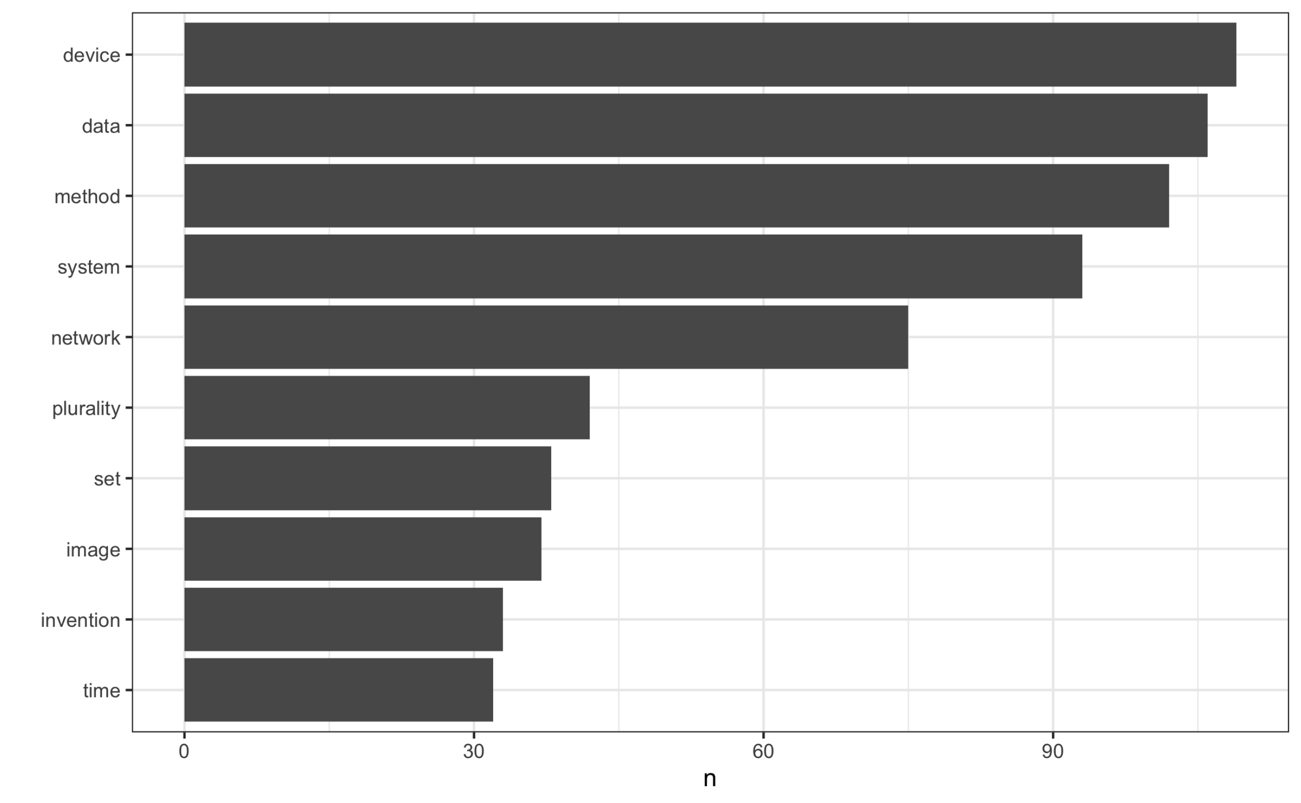


Using POS to filter

Parts of Speech tags are really interesting to extract easily the words you like to plot. You really don't need stopwords for doing this, just select nouns / verbs or adjectives and you have already the most relevant parts for basic frequency analysis.

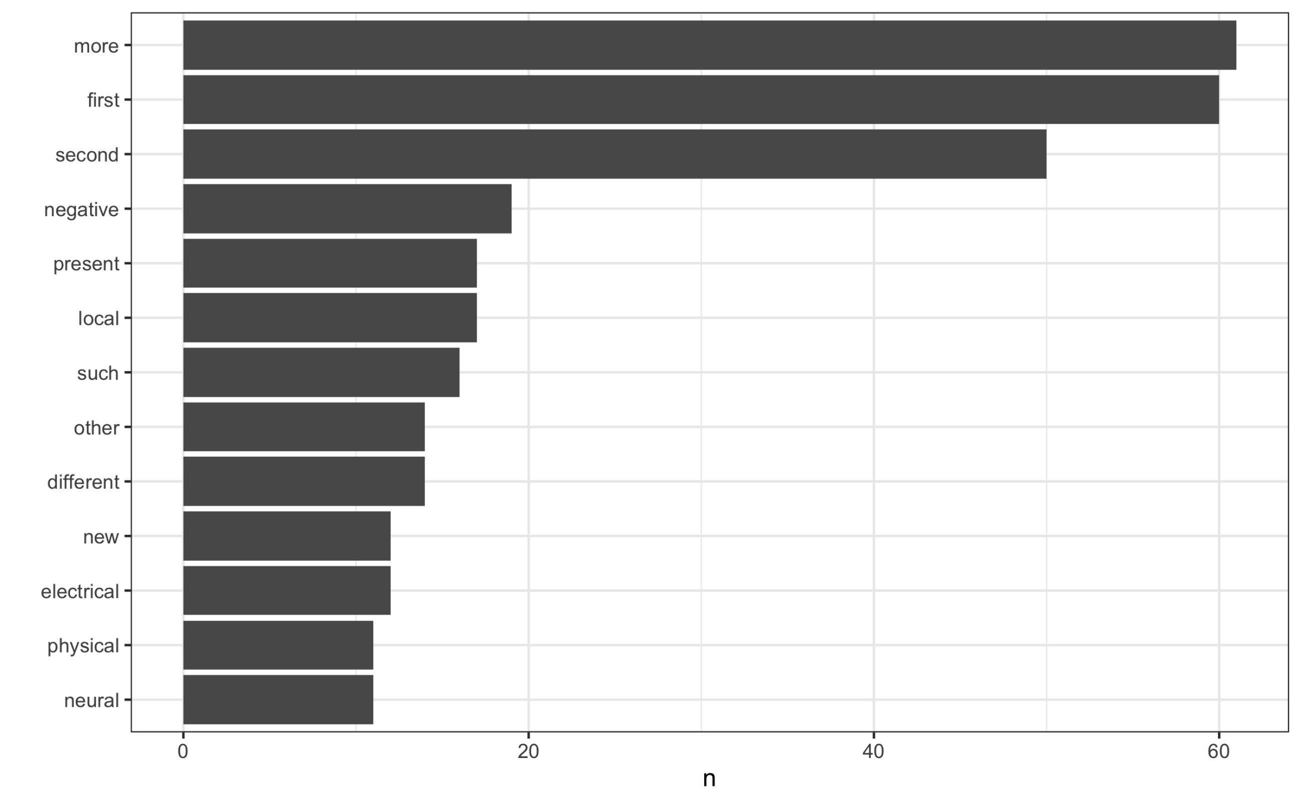
```
graph_noun <- patents_tagged %>%
  filter(upos == "NOUN") %>%
  count(lemma) %>%
  filter(n > 30) %>%
  ggplot(aes(x = reorder(lemma, n), y = n)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  xlab("") +
  theme_bw()
```

graph_noun


device
data
method
system
network
plurality
set
image
invention
time
0
30
60
90
n

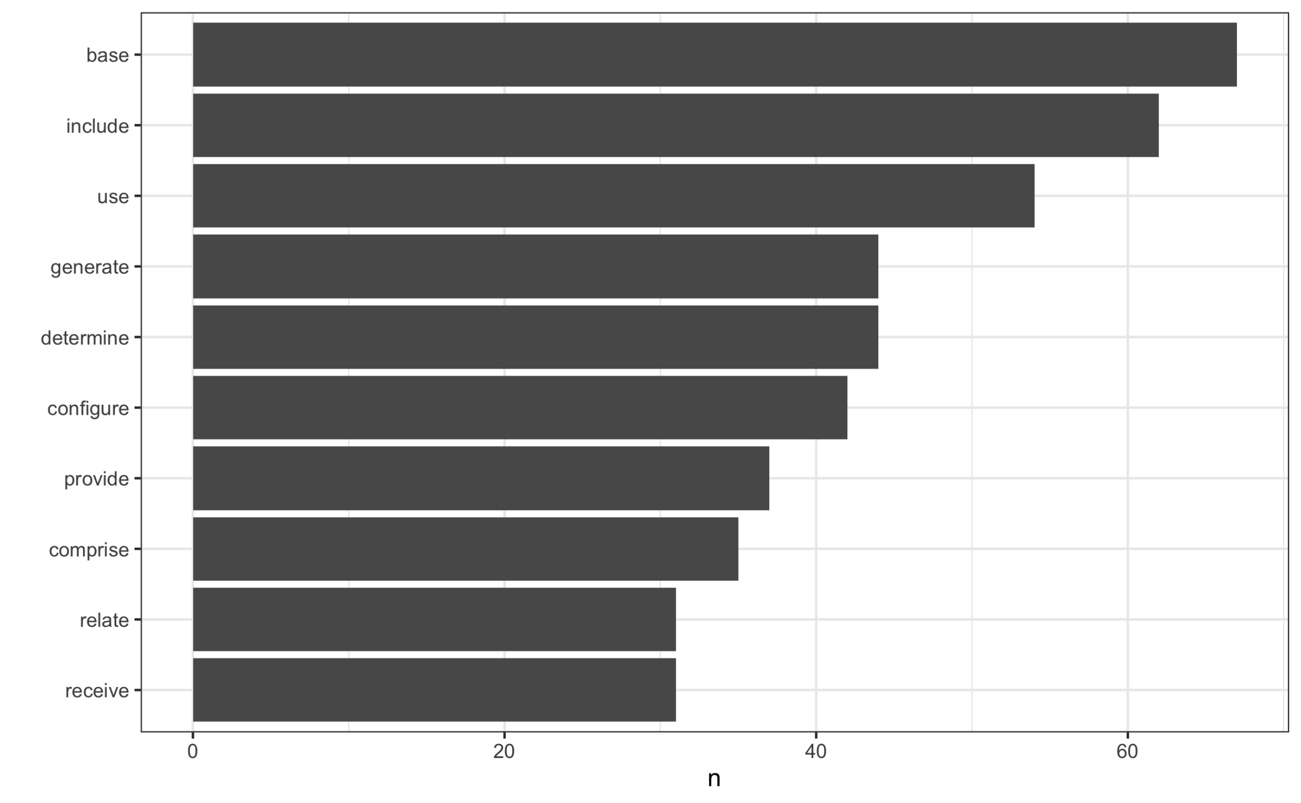
```r
graph_adj <- patents_tagged %>%
  filter(upos == "ADJ") %>%
  count(lemma) %>%
  filter(n > 10) %>%
  ggplot(aes(x = reorder(lemma, n), y = n)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  xlab("") +
  theme_bw()
```

graph_adj


more
first
second
negative
present
local
such
other
different
new
electrical
physical
neural
0
20
40
60
n

```r
graph_vb <- patents_tagged %>%
  filter(upos == "VERB") %>%
  count(lemma) %>%
  filter(n > 30) %>%
  ggplot(aes(x = reorder(lemma, n), y = n)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  xlab("") +
  theme_bw()
```

`graph_vb`


base
include
use
generate
determine
configure
provide
comprise
relate
receive
0
20
40
60
n

Finding keywords

Frequency statistics of words are nice but most of the time, you are getting stuck in words which only make sense in combination with other words. This is typical of technical documents, where the most of the key concepts are multi-words. Hence you want to find keywords which are a combination of words.

Currently, udpipe provides 3 methods to identify keywords in text:

- RAKE (Rapid Automatic Keyword Extraction)
- Collocation ordering using Pointwise Mutual Information
- Parts of Speech phrase sequence detection

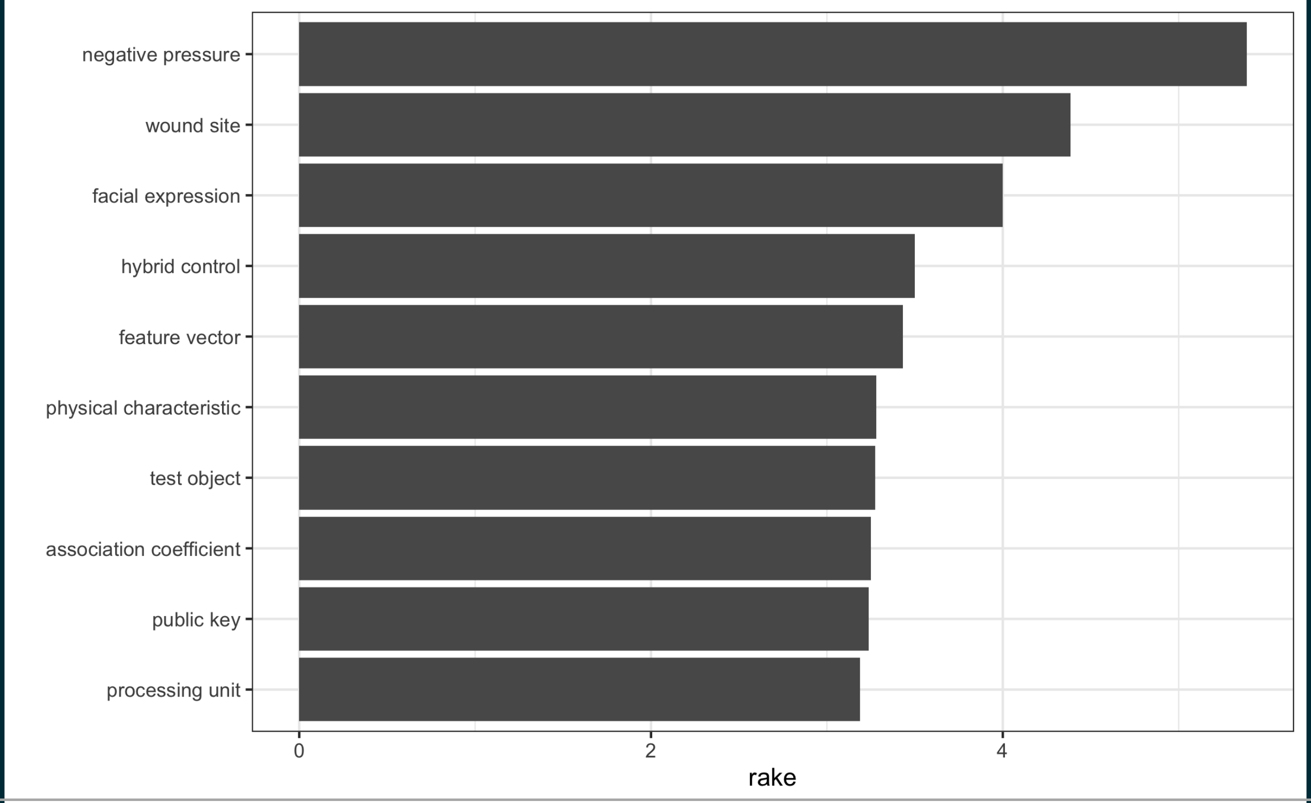
Using RAKE

```
pat_stats <- keywords_rake(x = patents_tagged, term = "lemma", group = "doc_id",
                       relevant = patents_tagged$upos %in% c("NOUN", "ADJ")) %>%
  as_tibble()
```

```
head(pat_stats)
```

```
## # A tibble: 6 × 4
##   keyword                 ngram  freq  rake
##   <chr>                   <int> <int> <dbl>
## 1 negative pressure           2     5  5.39
## 2 wound site                  2     2  4.38
## 3 facial expression           2     2  4   
## 4 hybrid control              2     2  3.5 
## 5 feature vector              2     7  3.43
## 6 physical characteristic     2     2  3.28
```

```
pat_stats %>%
  top_n(10, rake) %>%
  ggplot(aes(x = reorder(keyword, rake), y = rake)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  xlab("") +
  theme_bw()
```



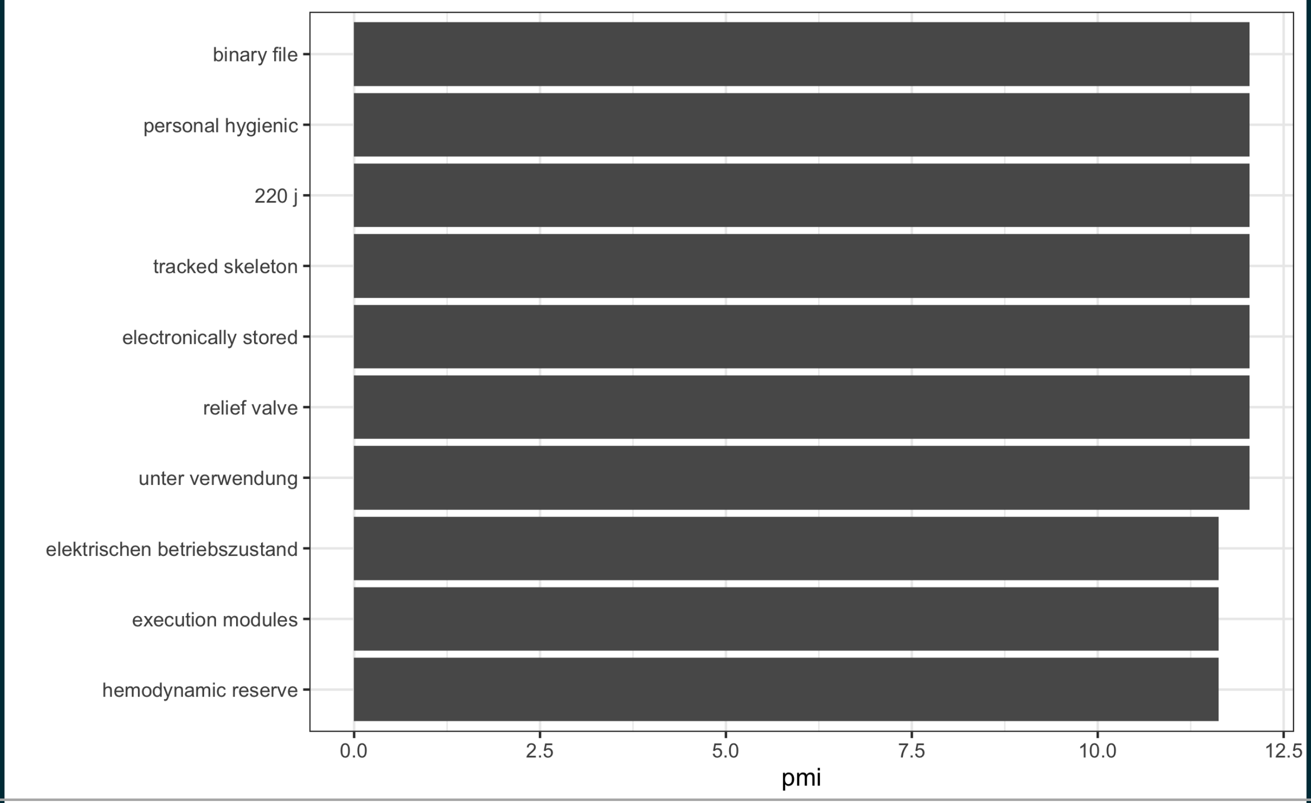
Using Pointwise Mutual Information Collocations

```
patents_tagged$word <- tolower(patents_tagged$token)
pat_stats <- keywords_collocation(x = patents_tagged, term = "word", group = "doc_id")
pat_stats$key <- factor(pat_stats$keyword, levels = rev(pat_stats$keyword))
```

```
head(pat_stats)
```

```
##                  keyword ngram           left    right freq
## 1           binary file     2         binary     file    3
## 2     personal hygienic     2       personal hygienic    3
## 3                 220 j     2            220        j    3
## 4      tracked skeleton     2        tracked skeleton    3
## 5 electronically stored     2 electronically   stored    3
## 6          relief valve     2         relief    valve    3
##   freq_left freq_right     pmi md     lfmd                   key
## 1         3          3 12.0388  0 -12.0388           binary file
## 2         3          3 12.0388  0 -12.0388     personal hygienic
## 3         3          3 12.0388  0 -12.0388                 220 j
## 4         3          3 12.0388  0 -12.0388      tracked skeleton
## 5         3          3 12.0388  0 -12.0388 electronically stored
## 6         3          3 12.0388  0 -12.0388          relief valve
```

```
pat_stats %>%
  top_n(10, pmi) %>%
  ggplot(aes(x = reorder(key, pmi), y = pmi)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  xlab("") +
  theme_bw()
```



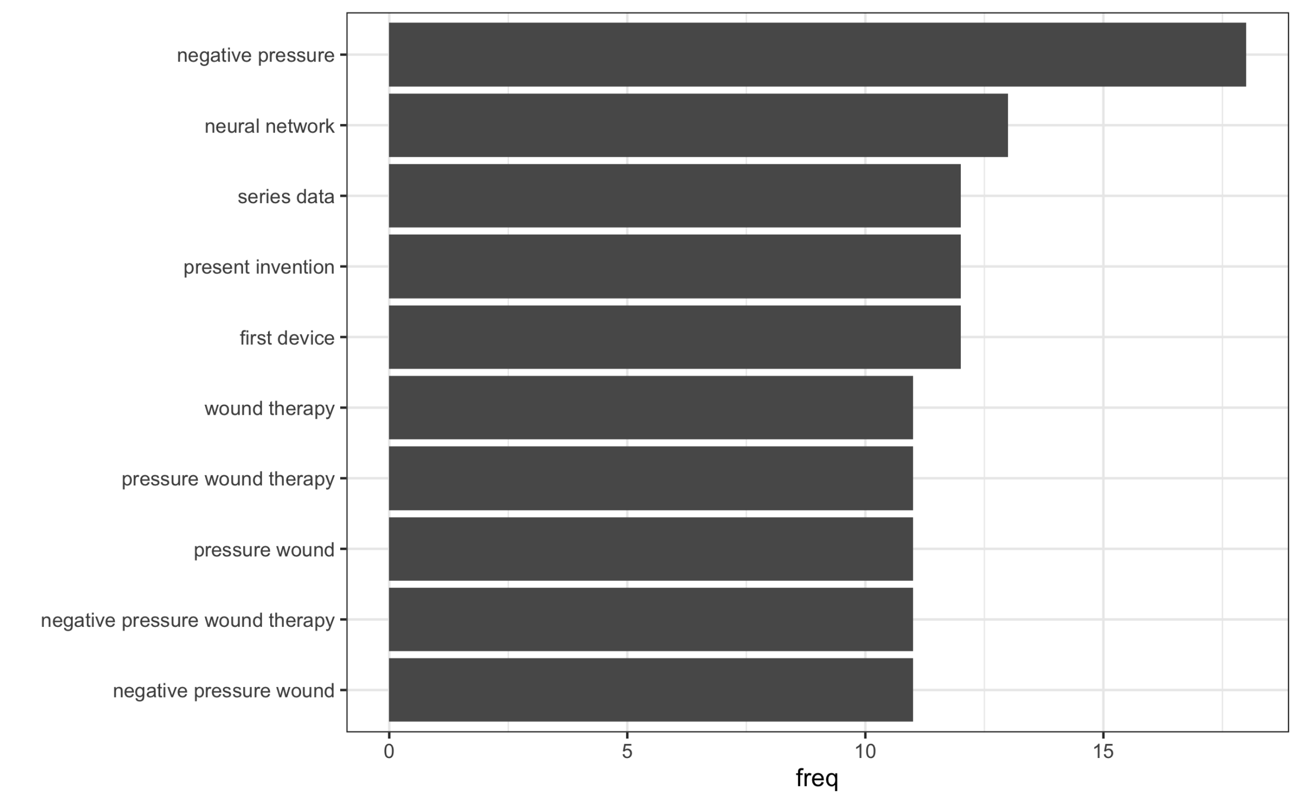
Using a sequence of POS tags (noun phrases / verb phrases)

```
# transform the pos in letters, to be readed by the tool
patents_tagged$phrase_tag <- as_phrasemachine(patents_tagged$upos, type = "upos")

# extract exact patterns of POS, using regex
stats <- keywords_phrases(x = patents_tagged$phrase_tag, term = tolower(patents_tagged$token),
                          pattern = "(A|N)*N(P+D*(A|N)*N)*",
                          is_regex = TRUE, detailed = FALSE)

head(stats)
```

```
##   keyword ngram freq
## 1    data     1  106
## 2  device     1  100
## 3  method     1   87
## 4  system     1   82
## 5 network     1   69
## 6    that     1   57
```

```r
chunked_graph <- stats %>%
  filter(ngram >1 ) %>%
  top_n(10, freq) %>%
  ggplot(aes(x = reorder(keyword, freq), y = freq)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  xlab("") +
  theme_bw()
```

chunked_graph


negative pressure
neural network
series data
present invention
first device
wound therapy
pressure wound therapy
pressure wound
negative pressure wound therapy
negative pressure wound
freq