neural text models
with R package ruimtehol

Brussels and in Starspace

Jan Wijffels

BNOSAC - jwijffels@bnosac.be

Overzicht

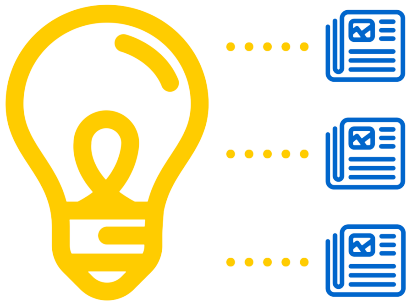
1. Starspace
2. Starspace for classification modelling
3. Word and Sentence embeddings
4. Article embeddings / article recommendation
5. More models
6. BNOSAC R consultancy

open analytical helpers

Starspace

Starspace / ruimtehol

- **The Starspace neural model** learns to map entities (e.g. text) to a set of numbers called embeddings
- An R wrapper called *ruimtehol* was developed by BNOSAC
  - `https://github.com/bnosac/ruimtehol`
  - wrapping the Starspace C++ library `https://github.com/facebookresearch/StarSpace`
- Example applications
  - Email / Customer feedback categorisation
  - Routing incoming customer questions to the right person
  - Enrich predictive models based on what is written in text
  - Chatbots
  - Automatic matching of questions of clients to possible answers
  - Recommending reading certain online help documents to clients
  - Matching CV to Job description
  - Sentiment scoring



- R package ruimtehol allows to do
  - Multi-label Text classification
  - Learning word, sentence or document level embeddings
  - Finding sentence or document similarity
  - Ranking web documents
  - Content-based or Collaborative filtering-based Recommendation

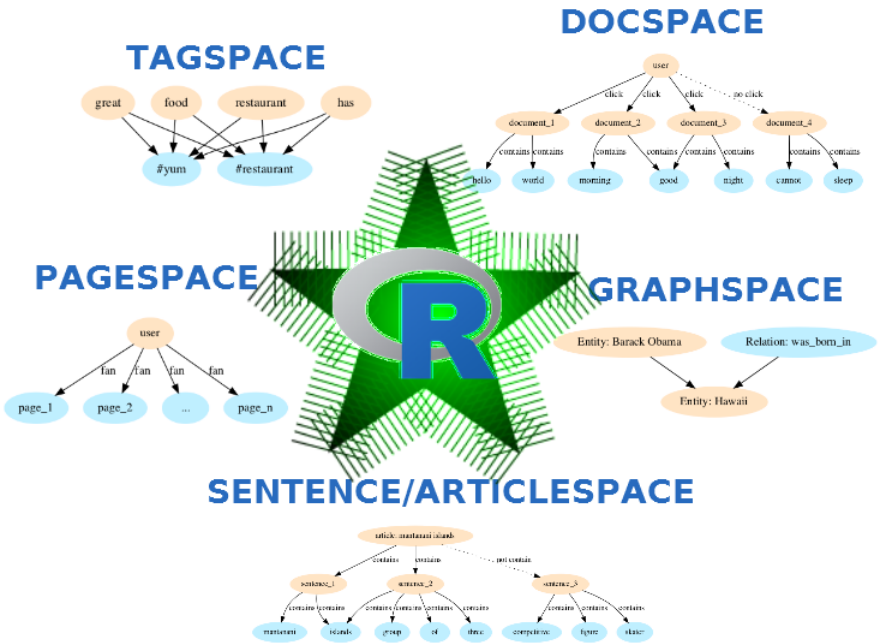


Figure: embed all the things with ruimtehol

Logic

- Starspace allows to **embed entities**
  - Entities are typically a customer mail, a webpage, a person, a type of category of customer answers, a department, a sentiment label, …
  - **Each entity can have several features** (e.g. words, ngrams, user clicked viewed a webpage which consists itself of words, an email was classified with a certain category)
- **Entity = sum of the embeddings of the features which are part of the entity**
  - $\sum_{feature_i \in entity} embedding_{feature_i}$
  - Example:
    - if embedding of the word 'field' is `[0.1, 0.2, 0.3, 0.4]`
    - if embedding of the word 'work' is `[-0.1, 0.1, 0.2, -0.2]`
    - embedding of bigram 'field work' is the sum: `[0, 0.3, 0.5, 0.2]`
  - Embeddings cover the meaning of text

- Model learns these embeddings **By comparing entities which are the same to entities which are not the same**
  - A set of positive pairs of entities (E^+) are selected from the data
  - A set of negative entities (E^-) are selected from the RHS of the data
- Minimises loss function over all entity similarities
  - $\sum_{(entity_a, entity_b \in E^+), (entity_{b-} \in E^-)} Loss_{batch}$
  - Loss = $(sim(a, b), sim(a, b_1^-), \ldots, , sim(a, b_k^-))$
  - similarities are cosine or dot product similarities of the embeddings
  - $entity_a$ is the left hand side, $entity_b$ / $entity_b^-$ are from the right hand side
- Embeddings are learned with stochastic gradient descent
- Learned embeddings can then be used to calculate similarities between entities

Example for classification

- left (*a* entity) would be the text of a customer email
- right (*b* entity) would be the email category
- you creating negatives by indicating *k* email categories that are not the right category (b^-)

Techniques will be showcased on the following dataset containing questions/answers in Belgian parliament.

```
library(ruimtehol)
data("dekamer", package = "ruimtehol")
str(dekamer)
```

```
'data.frame':   3811 obs. of  12 variables:
 $ doc_id              : int  184501 184502 184503 184504 184505 184506 184507 184508 184509 184521 ...
 $ depotdat            : Date, format: "2017-01-04" "2017-01-04" ...
 $ aut_party           : Factor w/ 16 levels "CD&V","CDH","DEFI",..: 6 6 9 9 1 6 6 7 11 9 ...
 $ aut_person          : Factor w/ 156 levels "Almaci, Meyrem",..: 10 17 87 87 86 12 12 154 67 87 ...
 $ aut_language        : Factor w/ 2 levels "dutch","french": 2 2 1 1 1 2 2 1 2 1 ...
 $ question            : chr  "Lijn Brussel-Luxemburg. - Technisch incident. \n\n Op maandag 19 december 2016 kondigde de NMBS een uitb
 $ question_theme_main : Factor w/ 45 levels "ARBEID","BEGROTING",..: 43 43 39 39 15 NA NA NA 43 NA ...
 $ question_theme      : chr  "VERVOERSLIJN,VERVOERBELEID,VERVOER PER SPOOR,NMBS,DIENSTREGELING VAN HET VERVOER" "FRAUDE,CONTROLEORGAAN
 $ answer              : chr  "1. De noodrem werd geactiveerd; het ging om kwaad opzet. 2. De treinbestuurder moet in eerste instantie
 $ answer_deptpres     : int  13 13 6 3 9 4 16 13 13 16 ...
 $ answer_department   : chr  "Minister van Mobiliteit, belast met Belgocontrol en de Nationale Maatschappij der Belgische spoorwegen"
 $ answer_subdepartment: chr  "Mobiliteit, Belgocontrol en NMBS" "Mobiliteit, Belgocontrol en NMBS" "Justitie" "Veiligheid en Binnenlan
```

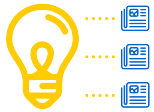
- Each question (field question) is labelled with several categories (field question_theme)
- Each question got an answer from the government (field answer)
- We know the name of the person who asked the question, the political party he/she belongs to and which department answered the questions

bnosac

open analytical helpers

Starspace for classification modelling

Starspace for classification



- Starspace allows to do **basic classification and multi-label classification**
- Use cases
  - sentiment classification
  - assignment of text to categories (e.g. incoming mail is assigned to right person, news is categorised alongside topics, customer answers are catgorised)

The sampling strategy

- left (*a* entity) would be the text of a customer email
- right (*b* entity) would be the email category or a set of categories
- you creating negatives by indicating *k* email categories that are not the right category (b^-)

How - first do basic data preparation

The following data preparation is needed

- make sure all the text is separated by spaces, this is the format which Starspace needs
- make sure the response is a list of all categories in case you do multi-label classification and each category should not contain spaces (in the below example spaces are replaced with a dash)

The Starspace C++ library internally uses spaces as a separator, make sure each token is separated by a space if you want to include it in the model and that the labels do not contain spaces.

```
dekamer$x <- strsplit(dekamer$question, "\\W")
dekamer$x <- lapply(dekamer$x, FUN = function(x) setdiff(x, ""))
dekamer$x <- sapply(dekamer$x, FUN = function(x) paste(x, collapse = " "))
dekamer$x <- tolower(dekamer$x)
dekamer$y <- strsplit(dekamer$question_theme, split = ",")
dekamer$y <- lapply(dekamer$y, FUN=function(x) gsub(" ", "-", x))
dekamer$x[1:2]
```

```
[1] "lijn brussel luxemburg technisch incident op maandag 19 december 2016 kondigde de nmbs een uitbreiding van het treinaanbod op brus
[2] "mystery shopping resultaten van de onderzoeken op een eerdere parlementaire vraag over met mysteryshoppers hebt u verwezen naar ei
```

```
dekamer$y[1:2]
```

```
[[1]]
[1] "VERVOERSLIJN"                  "VERVOERBELEID"
[3] "VERVOER-PER-SPOOR"             "NMBS"
[5] "DIENSTREGELING-VAN-HET-VERVOER"

[[2]]
[1] "FRAUDE"              "CONTROLEORGAAN"      "VERVOERBELEID"
```

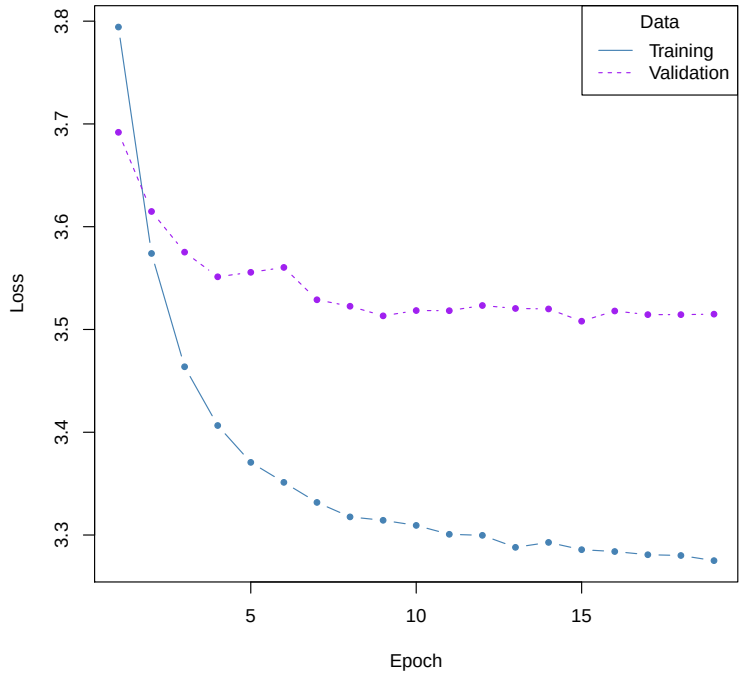
Model building

```
set.seed(321)
model <- embed_tagspace(x = dekamer$x, y = dekamer$y,
                        early_stopping = 0.8, validationPatience = 10,
                        dim = 50,
                        lr = 0.01, epoch = 40, loss = "softmax", adagrad = TRUE,
                        similarity = "cosine", negSearchLimit = 50,
                        ngrams = 2, minCount = 2)
model
```

The code above has trained a model with the following relevant arguments

1. **early_stopping**: data is split in a training (80%) and a validation set (20%)
2. **dim**: the dimension of the embedding is set to 50
3. optimisation is done with **adagrad**, during 40 **epochs**, starting with a learning rate (**lr**) of 0.01 and each time decreasing the learning rate by 1/epoch.
4. The **loss** which is optimised is softmax loss. If the loss does not stop decreasing after 10 iterations as set with **validationPatience**, training is stopped.
5. Similarity between positive and negative labels is done by **cosine** similarity
6. **negSearchLimit** indicates the number of negative samples which are taken (for each question we know which labels were given (positive) and we sample from the list of labels which were not given a bunch of negatives)
7. model is trained on bigrams (argument **ngrams**) and these should occur at least twice (argument **minCount**)

```
plot(model)
```



- Plot evolution of the loss over the epochs on the training set and the validation set
- This should generally decrease steadily and stabilise on the validation set which indicates it has learned the embeddings well
- If you don't see this behaviour, retrain with different hyperparameters

- the embeddings of the labels and the words are in the same embedding space allowing you to compute similarities across labels and text.
- dictionary of all terms in the model (words and labels)
- you can get embeddings of labels and words and ngrams

```
dict <- starspace_dictionary(model)
str(dict)
```

```
List of 6
 $ ntokens        : int 365654
 $ nwords         : int 14075
 $ nlabels        : int 1314
 $ labels         : chr [1:1314] "__label__VERVOERBELEID" "__label__GEZONDHEIDSBELEID" "__label__OVERHEID" "__label__OPENBARE-VEILIGHEI
 $ dictionary_size: int 15389
 $ dictionary     :'data.frame':    15389 obs. of  3 variables:
  ..$ term    : chr [1:15389] "de" "in" "het" "van" ...
  ..$ is_word : logi [1:15389] TRUE TRUE TRUE TRUE TRUE TRUE ...
  ..$ is_label: logi [1:15389] FALSE FALSE FALSE FALSE FALSE FALSE ...
```

```
emb_all    <- as.matrix(model)
emb_words  <- as.matrix(model, type = "words")
emb_labels <- as.matrix(model, type = "labels", prefix = FALSE)
e <- starspace_embedding(model, x = c("geld", "nationale loterij", "__label__VERVOERBELEID"), type = "ngram")
e
```

```
                             [,1]        [,2]       [,3]         [,4]
geld                  -0.009927476  0.06355097 0.02599729 -0.051911067
nationale loterij      0.016570840 -0.00510766 0.03713511  0.062561266
__label__VERVOERBELEID -0.023777248  0.04593002 0.01958371 -0.005254362
                             [,5]        [,6]        [,7]         [,8]
geld                  -0.038703699  0.04581665 -0.14283867  0.07054184
nationale loterij     -0.029120756 -0.02243782 -0.03028865  0.00239888
__label__VERVOERBELEID  0.006701535  0.02579976  0.01157977 -0.04964909
                             [,9]       [,10]       [,11]        [,12]
geld                  -0.104139268 -0.06040937 -0.13398242 -0.039323285
nationale loterij     -0.002617395  0.04821803  0.05865562 -0.076755270
__label__VERVOERBELEID  0.020059176  0.01897390  0.04063378  0.008211648
                            [,13]       [,14]       [,15]       [,16]
geld                  -0.10288447 -0.10941643 -0.01776431 0.01441684
```

- For retrieving embeddings of full text, use **type = 'document'** in starspace_embedding
- It aggregates the embeddings of the words which are part of the text

```
text <- c("de nmbs heeft het treinaanbod uitgebreid via onteigening",
          "de migranten komen naar europa de asielcentra")
emb_text <- starspace_embedding(model, text)
dim(emb_text)
```

```
[1]  2 50
```

- You can extract predictions and get embedding similarities for information retrieval and ranking.
- Find the closest label to some text

```
scores <- predict(model, "de migranten komen naar europa de asielcentra ...")
text   <- c("de nmbs heeft treinaanbod uitgebreid",
            "migranten komen naar europa, asielcentra")
scores <- embedding_similarity(starspace_embedding(model, text),
                               emb_labels, type = "cosine", top_n = 5)
scores[, c("term1", "term2", "rank")]
```

	term1	term2	rank
1	migranten komen naar europa, asielcentra	ILLEGALE-MIGRATIE	1
2	migranten komen naar europa, asielcentra	MIGRATIEBELEID	2
3	migranten komen naar europa, asielcentra	VLUCHTELING	3
4	migranten komen naar europa, asielcentra	OPENBARE-ORDE	4
5	migranten komen naar europa, asielcentra	UITWIJZING	5
6	de nmbs heeft treinaanbod uitgebreid	NMBS	1
7	de nmbs heeft treinaanbod uitgebreid	VERVOER-PER-SPOOR	2
8	de nmbs heeft treinaanbod uitgebreid	TRANSPORTINFRASTRUCTUUR	3
9	de nmbs heeft treinaanbod uitgebreid	VERVOERBELEID	4
10	de nmbs heeft treinaanbod uitgebreid	DIENSTREGELING-VAN-HET-VERVOER	5

- Inspect what your model has learned
  - by looking at the closest word to a text or a label
  - starspace_knn returns the nearest term in the dictionary to the provide text

```
starspace_knn(model, "migranten", k = 2)
```

```
$input
[1] "migranten"

$prediction
          label similarity rank
1     migranten  1.0000000    1
2 vluchtelingen  0.9896093    2
```

```
starspace_knn(model, "__label__MIGRATIEBELEID", k = 5)
```

```
$input
[1] "__label__MIGRATIEBELEID"

$prediction
                    label similarity rank
1 __label__MIGRATIEBELEID  1.0000002    1
2                   sefor  0.9888631    2
3             derdelander  0.9869671    3
4  regularisatieaanvragen  0.9862695    4
5                verblijf  0.9841670    5
```

```
starspace_knn(model, "de migranten komen naar europa de asielcentra ...", k = 5)
```

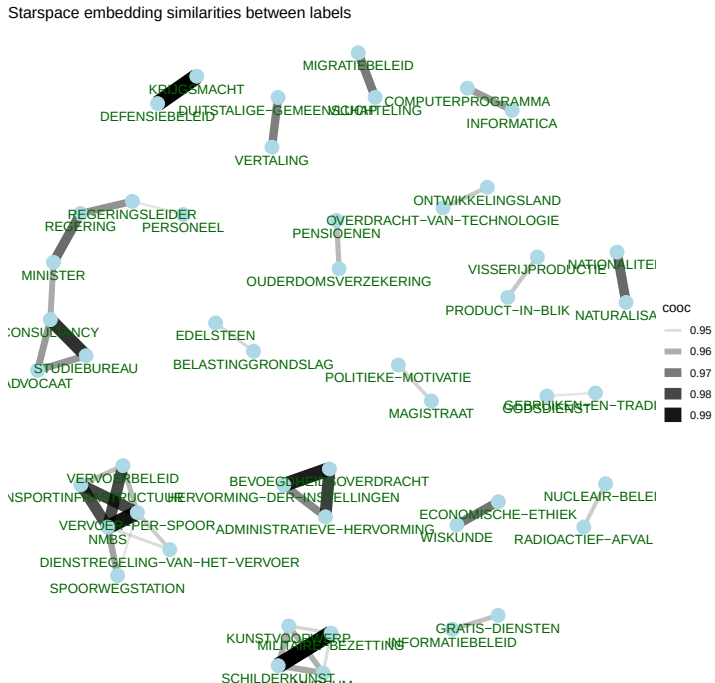
```
$input
[1] "de migranten komen naar europa de asielcentra ..."

$prediction
                   label similarity rank
1              toestroom  0.9207578    1
2              migranten  0.9192232    2
3                syriers  0.9186081    3
4                  bevel  0.9166192    4
5 hervestigingsprogramma  0.9120188    5
```

Visualise label embedding similarities

- Extract and visualise top 40 label embedding similarities

```
library(udpipe)
library(ggraph)
library(igraph)
emb_labels   <- as.matrix(model, type = "labels", prefix = FALSE)
similarities <- embedding_similarity(emb_labels, emb_labels)
similarities <- as_cooccurrence(similarities)
similarities <- subset(similarities, term1 < term2)
similarities <- similarities[order(similarities$cooc, decreasing = TRUE), ]
toplot <- head(similarities, 40)
toplot <- graph_from_data_frame(toplot)
ggraph(toplot, layout = "fr") +
  geom_edge_link0(aes(edge_alpha = cooc, edge_width = cooc)) +
  geom_node_point(color = "lightblue", size = 5) +
  geom_node_text(aes(label = name), vjust = 1.8, col = "darkgreen") +
  ggtitle("Starspace embedding similarities between labels") + theme_void()
```

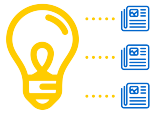


bnosac

open analytical helpers

Word and Sentence embeddings

Starspace for word and sentence embeddings



- Starspace allows to do **word and sentence embeddings**
- Use cases
  - find similar words / analogies
  - find similar sentences (e.g. what answer do we need to give to a mail, to which customer email does this email resemble)

The sampling strategy for word embeddings

- select (*a* entity) as a window of words (e.g., four words, two either side of a middle word), and (*b* entity) as the middle word
- negatives (*b*): words not in the middle

The sampling strategy for sentence embeddings

- Given a training set of unlabeled documents, each consisting of sentences, we select (*a* entity) and (*b* entity) as a pair of sentences both coming from the same document
- negatives (*b*) are sentences coming from other documents

Word embedding model

```
library(ruimtehol)
set.seed(321)
model <- embed_wordspace(dekamer$x, early_stopping = 0.8,
                         dim = 50, ws = 7, epoch = 10, ngrams = 2,
                         negSearchLimit = 10)
```

- Check what it has learned with starspace_knn
- You can get the same with predict with type = 'knn'

```
x <- as.matrix(model)
starspace_knn(model, "luchthaven")
```

```
$input
[1] "luchthaven"

$prediction
       label similarity rank
1 luchthaven  0.9999999    1
2      notam  0.6075996    2
3   zaventem  0.5921771    3
4         38  0.5183266    4
5 technische  0.5169517    5
```

```
predict(model, c("brussel", "nmbs"), type = 'knn')
```

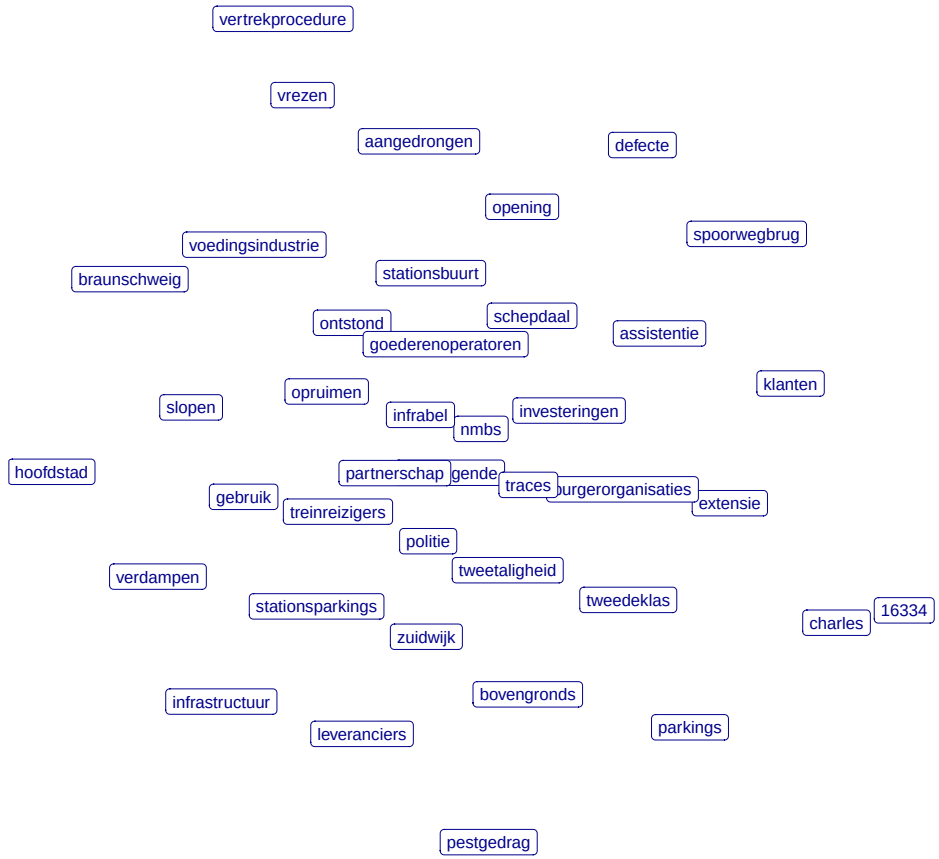
```
[[1]]
  doc_id    label similarity rank
1      1  brussel  1.0000001    1
2      1   weeder  0.7017010    2
3      1   perron  0.6184944    3
4      1     zuid  0.5887687    4
5      1 leefbaar  0.5706847    5

[[2]]
```

Visualise embedding in 2D using tSNE projection

```
library(projector)
library(ggplot2)
set.seed(321)
emb_words <- as.matrix(model)
annoy     <- get_annoy_model(emb_words, number_trees = 5)

toplot <- retrieve_neighbors(text = "nmbs", projection_type = "tsne", annoy_model = annoy, n = 40, perplexity = 5)
ggplot(toplot, aes(x = x, y = y)) +
  geom_label(label = toplot$text, size = 3, color = "darkblue") +
  theme_void() +
  ggtitle("Word embeddings in 2D using tSNE, centered around the word 'nmbs'")
```



Sentence embedding model

- For building the sentence embedding model, you need to have sentences
- Below we use the udpipe package to get these

```
library(udpipe)
library(ruimtehol)
data(dekamer, package = "ruimtehol")
questions <- udpipe(dekamer$question, "dutch", tagger = "none", parser = "none", trace = 250)
answers   <- udpipe(dekamer$answer, "dutch", tagger = "none", parser = "none", trace = 250)
saveRDS(questions, file = "dekamer_questions.rds")
saveRDS(answers,   file = "dekamer_answers.rds")
```

```
questions <- readRDS(file = "dekamer_questions.rds")
answers   <- readRDS(file = "dekamer_answers.rds")

set.seed(321)
model <- embed_sentencespace(questions, early_stopping = 0.8,
                             dim = 100, epoch = 20, minCount = 5, negSearchLimit = 10)
#plot(model)
```

- Find the most similar sentence to user input

```
sentences <- unique(questions$sentence)
sentences <- sentences[nchar(sentences) > 50]
scores <- embedding_similarity(
  starspace_embedding(model, "cijfers doorstroming"),
  starspace_embedding(model, sentences),
  top_n = 2)
scores
```

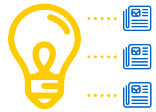
```
                  term1
1 cijfers doorstroming
2 cijfers doorstroming
                                                                                   term2
1 Beschikt u over cijfers van het aantal online verkochte DNA-kits in Belgie de afgelopen vijf jaren?4.
```

bnosac

open analytical helpers

Article embeddings / article recommendation

Starspace for article embeddings



- Starspace allows to do **sentence and article embeddings (articlespace)**
- Use cases
  - learning the *mapping between sentences and articles* (Given the embedding of one sentence, one can find the most relevant articles)
  - give the most appropriate answer from your existing knowledge base to a customer question
- Completely unsupervised

The sampling strategy for articlespace

- Each example from your knowledge base is an article which contains multiple sentences and the sentences are considered to be labels
- At training time, one sentence is picked at random as the input LHS (*a* entity), the remaining sentences in the article becomes the RHS label (*b* entity), other articles are picked as random negatives (*b* entity)

Article recommendation (articlespace)

- Input requires you to construct a tokenised data.frame with identifiers of the article and sentence
- Following function splits data in sentences and words

```
tokenize_article <- function(x){
  library(data.table)
  library(tokenizers)
  stopifnot(all(c("doc_id", "text") %in% colnames(x)))
  x <- na.exclude(x[, c("doc_id", "text")])
  ## split in sentences and words with tokenizers R package
  knowledgebase <- tokenizers::tokenize_sentences(x$text, lowercase = FALSE, strip_punct = FALSE)
  names(knowledgebase) <- x$doc_id
  knowledgebase <- lapply(knowledgebase, FUN=function(x){
    sentencelist <- tokenizers::tokenize_words(x, lowercase = TRUE, strip_punct = TRUE, strip_numeric = TRUE)
    sentencelist <- lapply(sentencelist, FUN=function(x) list(token = setdiff(x, "")))
    data.table::rbindlist(sentencelist, idcol = "sentence_id")
  })
  knowledgebase <- data.table::rbindlist(knowledgebase, idcol = "doc_id")
  knowledgebase
}
```

- Get all sentences of a knowledge base (sentence separated by tab)

```
knowledgebase <- data.frame(doc_id = dekamer$doc_id, text = dekamer$answer, stringsAsFactors = FALSE)
#knowledgebase <- udpipe(knowledgebase, "english")
knowledgebase <- tokenize_article(knowledgebase)
head(knowledgebase)
```

```
   doc_id sentence_id       token
1: 184501           2          de
2: 184501           2     noodrem
3: 184501           2        werd
4: 184501           2 geactiveerd
5: 184501           2         het
6: 184501           2        ging
```

- Use embed_articlespace to embed the knowledgebase

```
set.seed(321)
model <- embed_articlespace(knowledgebase,
                            dim = 150, lr = 0.05, epoch = 20, similarity = "cosine",
                            negSearchLimit = 5, maxNegSamples = 3, dropoutRHS = 0.5, adagrad = TRUE, minCount = 5)
```

- Use the Starspace model
  1. You can get the embeddings of the knowledgebase
  2. You can get the closest answer in the knowledgebase to a customer question (use argument basedoc for providing the list of possible answers)

```
emb <- predict(model, allarticles$text, type = "embedding")
# inspect that the model has learned the relevant things
closests <- starspace_knn(model, newdata = "nmbs")

# what is closest answer to this question
scores <- predict(model, "wat was de precieze oorzaak van de technische problemen", basedoc = allarticles$text)
scores <- predict(model, dekamer$question[1:2], basedoc = allarticles$text)
```

- The predict function returns the closest article and the words which are used to obtain that prediction.

```
str(scores[[1]])
```

```
List of 4
 $ doc_id    : int 1
 $ text      : chr "Lijn Brussel-Luxemburg. - Technisch incident. \n\n Op maandag 19 december 2016 kondigde de NMBS een uitbreiding van
 $ prediction:'data.frame': 5 obs. of  3 variables:
  ..$ label          : chr [1:5] "de proefprojecten in stations ciney marloie en libramont werden op januari ingevoerd\tde evaluatie za
  ..$ label_starspace: chr [1:5] "de proefprojecten in stations ciney marloie en libramont werden op januari ingevoerd\tde evaluatie za
  ..$ similarity     : num [1:5] 0.848 0.813 0.807 0.781 0.757
 $ terms     :List of 2
  ..$ basedoc_index: int [1:5] 247 2569 1859 2453 8
  ..$ basedoc_terms: chr [1:5] "de proefprojecten in stations marloie en libramont werden op januari ingevoerd de evaluatie zal gebeure
```

bnosac

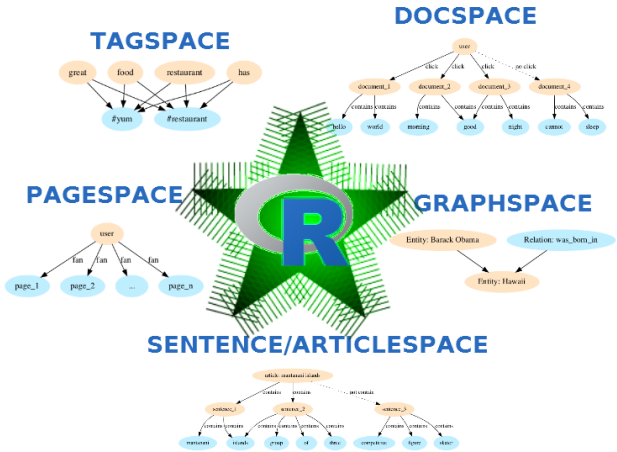
open analytical helpers

More models

Other models

- Interest-based or Document-based Recommendation
  1. **embed_pagespace**: Build a Starspace model for interest-based recommendation (e.g. user clicks on a web page)
  2. **embed_docspace**: Build a Starspace model for content-based recommendation (e.g. user clicks on a web page and you know the content of the web page)
- Entity relationship completion (graphspace)
  3. **embed_entityrelationspace**: Build a Starspace model for embedding the graph (e.g. you know 2 entities and the type of link the 2 entities have)
- Multi-task learning and semi-supervised learning and using other pre-trained word embeddings (e.g. fasttext/glove/…)
  4. **starspace**: Direct access to the Starspace C++ layer for experts

More examples are provided in the R package



bnosac

open analytical helpers

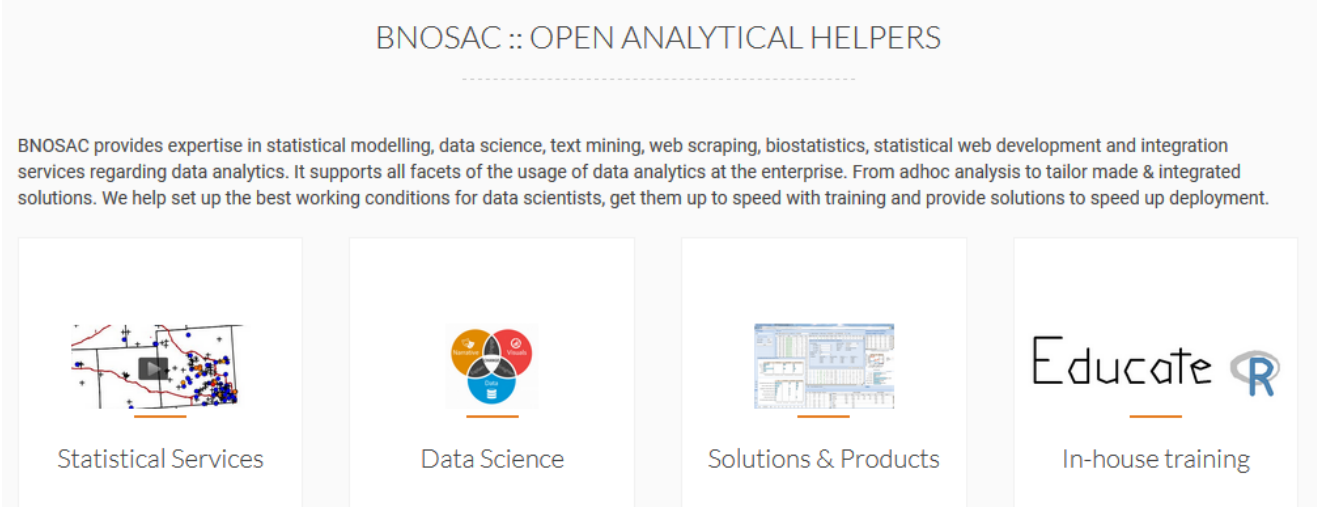
BNOSAC R consultancy

www.bnosac.be

www.bnosac.be: open source analytics experts

Providing consultancy services in **open source analytical engineering**

- Support for R / Oracle R Enterprise / Microsoft R / PostgreSQL / Python / ExtJS / Hadoop / …
- Expertise in predictive data mining, biostatistics, geostats, R + Python programming, GUI building, artificial intelligence, process automation, analytical web development
- R/Python implementations, application maintenance & training/consulting
- RStudio Server Pro / Shiny Pro / RStudio Connect reseller
- Hosting CRAN at www.datatailor.be
- Contributing to the R community with R packages / R training



R support by BNOSAC

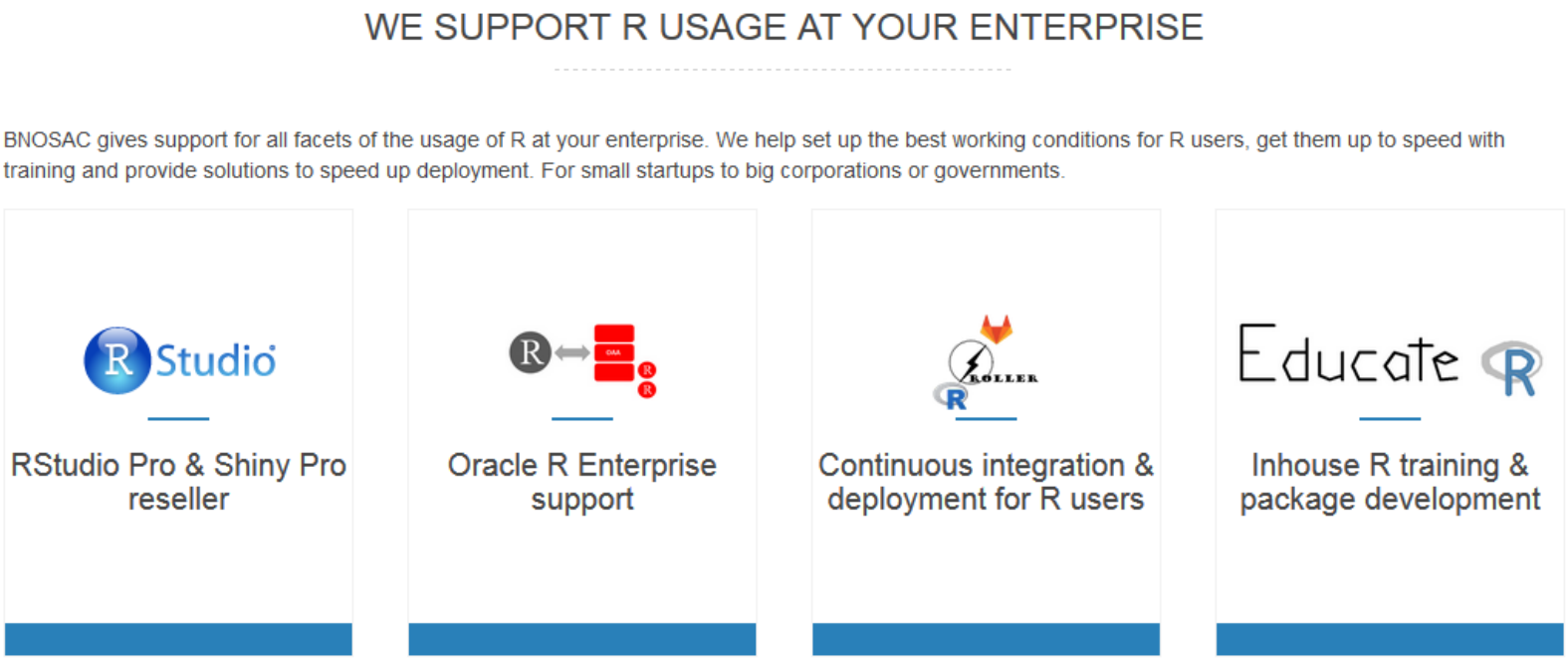


Figure: R support by BNOSAC

R training by BNOSAC: www.bnosac.be/training

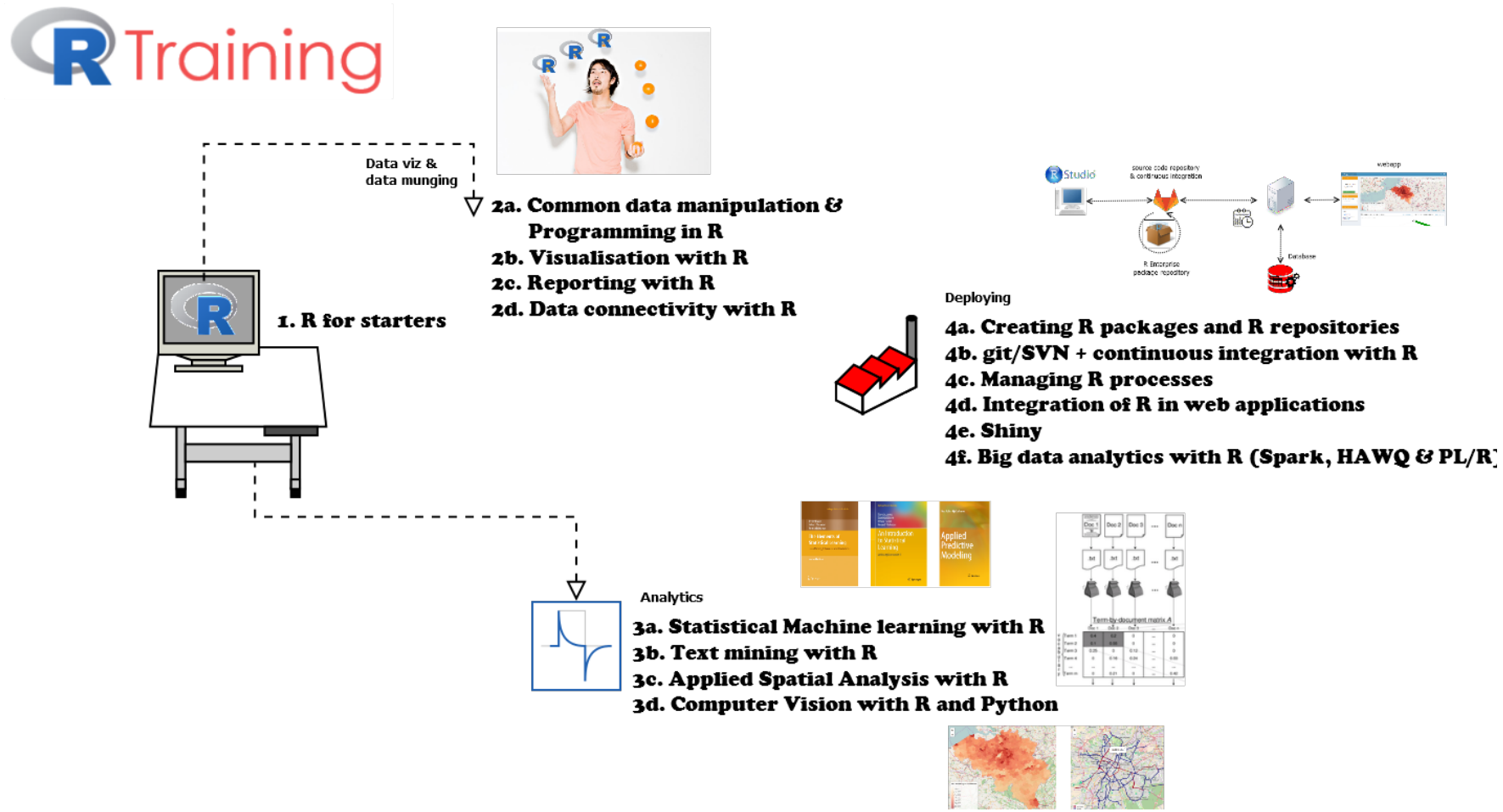


Figure: Training on R / Data Science

www.bnosac.be

Need support, send a message at www.bnosac.be/index.php/contact/get-in-touch

Figure: Contact www.bnosac.be