

Neural Text Models with R package ruimtehol

Jan Wijffels

Abstract

Ruimtehol is a comprehensive R package which wraps the StarSpace C++ library (<https://github.com/facebookresearch/StarSpace>). Starspace is a neural natural language modelling toolkit which allows you to calculate word, sentence, article, document, webpage, link and entity 'embeddings'. By using the 'embeddings', you can perform text based multi-label classification, find similarities between texts and categories, do collaborative-filtering based recommendation as well as content-based recommendation, find out relations between entities, calculate graph 'embeddings' as well as perform semi-supervised learning and multi-task learning on plain text. The techniques are explained in detail in the paper: *StarSpace: Embed All The Things!*, available at (<https://arxiv.org/abs/1709.03856>).

Keywords: Starspace, NLP, embed, embedding, neural, text.

1. Ground control to ruimtehol

1.1. Overview

The ruimtehol R package which wraps the StarSpace C++ library (<https://github.com/facebookresearch/StarSpace>), focussing on building and utilising embedding models for natural language. It allows you to do the following Natural Language Processing tasks

- Text classification
- Learning word, sentence or document level embeddings
- Finding sentence or document similarity
- Ranking web documents
- Content-based recommendation (e.g. recommend text/music based on the content)
- Collaborative filtering based recommendation (e.g. recommend text/music based on interest)
- Identification of entity relationships

Source code repository

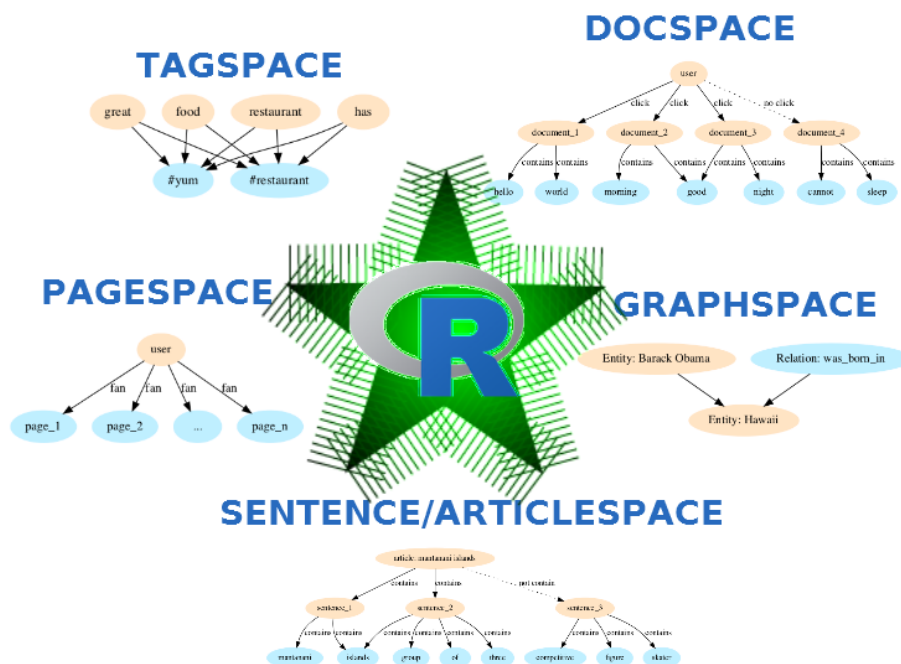
The source code of the package is on github at <https://github.com/bnosac/ruimtehol>. It uses Starspace version 'STARSPACE-2017-2' cloned in September 2018 to <https://github.com/facebookresearch/StarSpace>.

[com/bnosac-dev/StarSpace](https://github.com/bnosac-dev/StarSpace). Modifications were done in order to make it CRAN compliant and to make the usage more straightforward for R users. The R package is distributed under the Mozilla Public License version 2.0.

1.2. Functionalities

The R package allows you to have a low-level access to the C++ library for fine-grained control and provides as well high-level interfaces for commonly used tasks. Functionalities are also included for saving and loading models, using the models to predict and general embedding similarity and ranking functions.

<code>starspace</code>	Low-level interface to build a Starspace model
<code>starspace_load_model</code>	Load a pre-trained model or a tab-separated file
<code>starspace_save_model</code>	Save a Starspace model
<code>starspace_embedding</code>	Get embeddings of documents/words/ngrams/labels
<code>starspace_knn</code>	Find k-nearest neighbouring information for new text
<code>starspace_dictionary</code>	Get words/labels part of the model dictionary
<code>predict.textspace</code>	Get predictions along a Starspace model
<code>as.matrix</code>	Get words and label embeddings
<code>embedding_similarity</code>	Cosine/dot product similarity between embeddings - top-n most similar text
<code>embed_wordspace</code>	Build a Starspace model which calculates word/ngram embeddings
<code>embed_sentencespace</code>	Build a Starspace model which calculates sentence embeddings
<code>embed_articlespace</code>	Build a Starspace model for embedding articles - sentence-article similarities
<code>embed_tagspace</code>	Build a Starspace model for multi-label classification
<code>embed_docspace</code>	Build a Starspace model for content-based recommendation
<code>embed_pagespace</code>	Build a Starspace model for interest-based recommendation
<code>embed_entityrelationspace</code>	Build a Starspace model for entity relationship completion



1.3. Example data

In what follows below, we showcase some of the use cases of the R package. In order to do that, we will use text with questions and answers from the Belgian parliament. This dataset was collected as open data under the CC0 license from <http://data.dekamer.be>

The data contains Dutch questions asked in the year 2017 by members in the National Belgian parliament. Each question was categorised alongside several themes and the dataset also contains the answer which was given by the department or minister who was responsible for that topic. We also know to which political party the person belonged to who was asking the question.

```
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 1845
 $ 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 15
 $ 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 maand
 $ question_theme_main : Factor w/ 45 levels "ARBEID","BEGROTING",...: 43 43 39 39 15 NA NA
 $ question_theme    : chr   "VERVOERSLIJN,VERVOERBELEID,VERVOER PER SPOOR,NMBS,DIENSTREG
 $ answer          : chr   "1. De noodrem werd geactiveerd; het ging om kwaad opzet. 2.
 $ answer_deptpres   : int   13 13 6 3 9 4 16 13 13 16 ...
 $ answer_department : chr   "Minister van Mobiliteit, belast met Belgocontrol en de Nati
 $ answer_subdepartment: chr   "Mobiliteit, Belgocontrol en NMBS" "Mobiliteit, Belgocontrol
```

2. Text classification

In this example, we will perform text classification. Each question in parliament can be labelled with one or more tags. A tag-space model is constructed which can be used to tag new questions with the learned tagset.

2.1. Data preparation

The following data preparation is done first

- 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 nmb
[2] "mystery shopping resultaten van de onderzoeken op een eerdere parlementaire vraag ove
```

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

```
[[1]]
```

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

```
[[2]]
```

```
[1] "FRAUDE"                    "CONTROLEORGaan"    "VERVOERBELEID"
[4] "PLAATSBEWIJS"             "VERVOER-PER-SPOOR" "NMBS"
```

2.2. Model building

Next a model is constructed, in this case a tag-space model which is a model which can be used for simple classification as well as multi-label classification (which is the case here).

```
set.seed(321)
model <- embed_tag_space(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
```

```
Object of class textspace
dimension of the embedding: 50
training arguments:
  loss: softmax
  margin: 0.05
  similarity: cosine
  epoch: 40
  adagrad: TRUE
  lr: 0.01
  termLr: 1e-09
  norm: 1
  maxNegSamples: 10
  negSearchLimit: 50
  p: 0.5
  shareEmb: TRUE
  ws: 5
  dropoutLHS: 0
  dropoutRHS: 0
  initRandSd: 0.001
```

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

- **early_stopping**: data is split in a training (80 %) and a validation set (20 %)
- **dim**: the dimension of the embedding is set to 50
- 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.
- 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.
- Similarity between positive and negative labels is done by **cosine** similarity

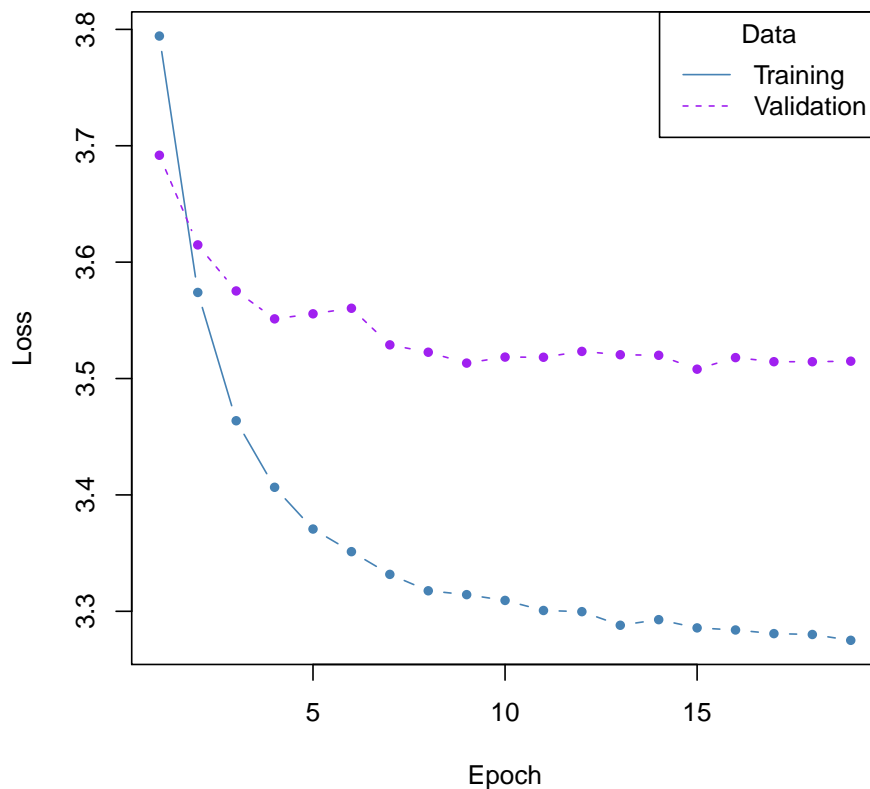
- **negSearchLimit** indicates the number of negative samples which are taken (for each question we know which labels were given (positive) and we sample from the list of labels which were not given a bunch of negatives)
- model is trained on bigrams (argument **ngrams**) and these should occur at least twice (argument **minCount**)

2.3. Model inspection

Loss evolution

The plot shows the 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.

```
plot(model)
```



Dictionary

Starspace models the embeddings of the labels and the words in the same embedding space

allowing you to compute similarities across labels and text. To get the dictionary of all terms in the model (words and labels), you can do the following.

```
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" "__
$ 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 ...
```

```
length(dict$labels)
```

```
[1] 1314
```

The dictionary element contains a data.frame with all words as well as all labels which were found in the data and for which embeddings are calculated.

Note that bigrams or ngrams are not stored and you'll also see that labels are prefixed with the text `__label__`, allowing them to be distinguished from plain words.

2.4. Embeddings of the dictionary

You can get the learned embeddings of the words and the labels which are part of the dictionary as follows.

```
emb <- as.matrix(model)
dim(emb)
```

```
[1] 15389    50
```

If you only want embeddings of the words or of the labels, you can set the *type* argument in the function *as.matrix* or you can use the function *starspace_embedding* directly. Below the embedding of the word 'geld' is retrieved as well as the embedding of the label 'VERVOERBELEID'.

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

If you trained a model with *ngrams* > 1, you can get the embedding of a bigram/ngram also. For this Starspace uses a hashing trick from *fastText* and gets - based on the words which define the bigram/ngram - the embedding of the hashed bucket of the combined term. If you specify to retrieve this type of embedding, you can not have more words than you specified *ngrams* when doing the training.

```
e <- starspace_embedding(model, c("nationale loterij"), type = "ngram")
```

2.5. Embeddings of full text

For retrieving embeddings of full text, use *type = 'document'* which is the default of *starspace_embedding*. It aggregates the embeddings of the words which are part of the text. Note that this aggregation is governed by the normalization parameter *p* which you can set when you build the model. The sum of the embeddings of the words which are in the text is normalised by dividing by $\#words^p$. If $p = 1$, this is equivalent to taking the average of the embeddings. When $p = 0$, this is equivalent to taking sum of the embeddings. The default is $p = 0.5$ indicating a mixture of both.

```
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
```

2.6. Predict/Similarities/Ranking

You can extract predictions and get embedding similarities for information retrieval and ranking. Function *predict* gets the document embeddings of the text and find the closest among the labels.

```
predict(model, "de migranten komen naar europa de asielcentra ...")
```

```
[[1]]
[[1]]$doc_id
[1] 1
```

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

```
[[1]]$prediction
      label      label_starspace similarity
1 OPENBARE-ORDE __label__OPENBARE-ORDE 0.8768471
2  VLUCHTELING  __label__VLUCHTELING 0.8725885
3 MIGRATIEBELEID __label__MIGRATIEBELEID 0.8678574
```



```

4      ARRESTATIE      __label__ARRESTATIE  0.8512142
5          SYRIE          __label__SYRIE  0.8414013

```

```

[[1]]$terms
[[1]]$terms$basedoc_index
[1] 581  21   9 394 199

```

```

[[1]]$terms$basedoc_terms
character(0)

```

The same can also be achieved with the function *embedding_similarity* which provides cosine and dot product similarities. They give the same numbers as the predict functionality.

```
embedding_similarity(emb_text, emb_labels, type = "cosine", top_n = 5)
```

```

                                                    term1
1 de nmbs heeft het treinaanbod uitgebreid via onteigening ...
2 de nmbs heeft het treinaanbod uitgebreid via onteigening ...
3 de nmbs heeft het treinaanbod uitgebreid via onteigening ...
4 de nmbs heeft het treinaanbod uitgebreid via onteigening ...
5 de nmbs heeft het treinaanbod uitgebreid via onteigening ...
6       de migranten komen naar europa de asielcentra ...
7       de migranten komen naar europa de asielcentra ...
8       de migranten komen naar europa de asielcentra ...
9       de migranten komen naar europa de asielcentra ...
10      de migranten komen naar europa de asielcentra ...
      term2 similarity rank
1      VERVOERBELEID  0.9388207   1
2              NMBS  0.9348487   2
3  TRANSPORTINFRASTRUCTUUR  0.9241427   3
4      ROLLEND-MATERIEEL  0.9163696   4
5      VERVOER-PER-SPOOR  0.9108901   5
6      OPENBARE-ORDE  0.8768471   1
7      VLUCHTELING  0.8725885   2
8      MIGRATIEBELEID  0.8678574   3
9      ARRESTATIE  0.8512143   4
10     SYRIE  0.8414015   5

```

Shorthands for the knn Starspace functionality is also provided. This function allows you to answer things like 'What does this look like'. It shows the nearest neighbour of text to the dictionary.

```
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	0.9207584	1
2	migranten	0.9192230	2
3	syriers	0.9186083	3
4	bevel	0.9166191	4
5	hervestigingsprogramma	0.9120197	5

2.7. Customising

As the words and labels are in the same embedding space, you can interpret the predict and similarity functionalities in a broad way. This is shown below where the target documents to compare with are changed.

```
targetdocs <- c("__label__FISCALITEIT",
                "__label__OVERHEIDSADMINISTRATIE",
                "__label__MIGRATIEBELEID",
                "__label__POLITIE",
                "__label__BUITENLANDS-BELEID",
                "__label__ECONOMISCH-BELEID",
                "de migranten komen naar europa ZZZ",
                "__label__PERSONEEL")
predict(model, "de migranten komen naar europa de asielcentra ...",
        basedoc = targetdocs)
embedding_similarity(
  starspace_embedding(model, "de migranten komen naar europa de asielcentra ..."),
  starspace_embedding(model, targetdocs), top_n = 3)
```

2.8. Save/Load model

Saving the model consists of saving the embeddings of the words and the labels as well as storing the model parameters. The saved model can next be loaded back in and used for information retrieval. The following approach is the advised approach to save and reload models.

```
starspace_save_model(model, file = "textspace.ruimtehol")
model <- starspace_load_model("textspace.ruimtehol")
```

3. Other models

The `ruimtehol` package contains many more models. It is advised to just inspect the help of the functions listed up in section 1.2.

- Embeddings of word, sentences, articles, documents, webpages, links and entities: See the examples in the package.
- Ranking and information retrieval: See the examples in the package.
- Collaborative filtering: See the examples in the package.

Affiliation:

BNOSAC - Open Analytical Helpers

E-mail: jwijffels@bnosac.be

URL: <http://www.bnosac.be>