

Embed Text with BERT models and Friends

Jan Wijffels

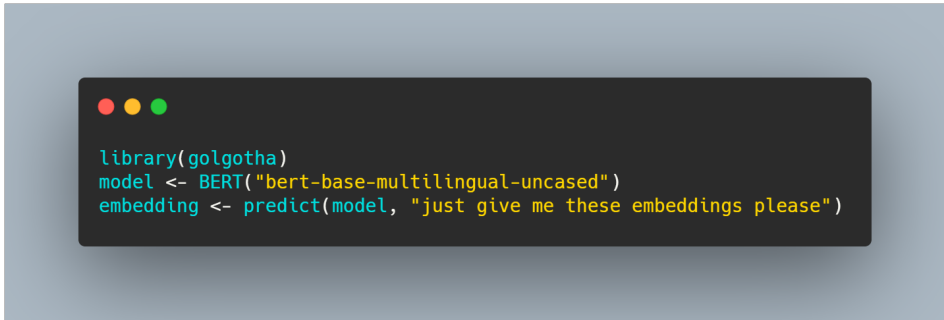
Abstract

The objective of the *golgotha* package is to easily get sentence embeddings using BERT-like models in R. This is particularly usefull if you are a practitioner in Natural Language Processing familiar with R and you want to use these embeddings for downstream modelling, notably as input to Support Vector Machines, for Sentiment Labelling models, for Classification or Regression purposes or in order to find text Similarities

Keywords: BERT, Transformers, NLP, embed, embedding, neural, text.

1. Usage

BERT models are Transformer-based models which are constructed on huge volumes of text and allow to get the embedding of a token within its context or the embedding of sentences. The transformers (<https://github.com/huggingface/transformers>) module gives access to these pretrained models and provides ready-made functionalities to work with these. **This R package facilitates working with these heavy-weight models and provides easy functions to get sentence and token embeddings.** Just that, nothing more, nothing less. It pretty much looks like this.



```
library(golgotha)
model <- BERT("bert-base-multilingual-uncased")
embedding <- predict(model, "just give me these embeddings please")
```

The NLP community has created many types of Transformer architectures. A list of these can be found at <https://github.com/huggingface/transformers#model-architectures>. In this example we will use a model which was trained on a diverse set of Wikipedia dumps, covering 102 languages. The model is named *bert-base-multilingual-uncased*.

2. BERT example

2.1. Model

The following code will download the *bert-base-multilingual-uncased* model once. By default it downloads the model in the models folder of the package, if this was the first time you were referencing this model. Note model size is approx 600Mb. Next time you reference the model, it will just download it from your hard disk directly.

```
library(golgotha)
model <- BERT("bert-base-multilingual-uncased")
```

Downloading model to C:/Users/Jan/Documents/R/win-library/3.5/golgotha/models/bert-base-mu

```
model
```

Object of class Transformer

```
model name: bert-base-multilingual-uncased
model stored at: C:/Users/Jan/Documents/R/win-library/3.5/golgotha/models/bert-base-mult
```

2.2. Embeddings

Once you have the model, you can start getting the embeddings of either

- Sentences
- The subword tokens which are part of the sentences

These embeddings can next be used in your downstream tasks. In case of *embed-sentence* you get a matrix back containing the average of the embeddings of the last layer of the BERT model, in case of *embed-token*, you get a list of matrices back, containing the contextualised embeddings of the tokens and special tokens which were detected using the Wordpiece tokeniser in the sentences.

Note that getting these embeddings can take a while as these models are huge. To print a trace of the evolution, use the *trace* argument.

```
x <- data.frame(
  doc_id = c("doc_1", "doc_2"),
  text = c("give me back my money or i'll call the police.",
           "talk to the hand because the face don't want to hear it any more."),
  stringsAsFactors = FALSE)
embedding <- predict(model, x, type = "embed-sentence", trace = 10)
```

```

class(embedding)

[1] "matrix"

str(embedding)

num [1:2, 1:768] -0.2137 -0.1197 0.0108 -0.0937 0.4037 ...
- attr(*, "dimnames")=List of 2
..$ : chr [1:2] "doc_1" "doc_2"
..$ : NULL

dim(embedding)

[1] 2 768

embedding <- predict(model, x, type = "embed-token", trace = FALSE)
str(embedding)

List of 2
 $ doc_1: num [1:15, 1:768] -0.1588 0.0311 -0.0542 -0.6092 -0.2788 ...
 $ doc_2: num [1:19, 1:768] -0.0869 0.2561 -0.5539 -0.2934 0.1358 ...

```

In order to get the subword tokens, you can use the `predict` function as well to get the wordpieces which were used to construct the embedding. Note that BERT models use as first and last token special symbols, namely [CLS] and [SEP], that is why the embeddings you get with *embed-token* also include the embeddings of these 2 special tokens.

```

tokens <- predict(model, x, type = "tokenise")
tokens

$doc_1
[1] "give"    "me"      "back"    "my"      "money"   "or"      "i"       ""
[9] "ll"      "call"    "the"     "police"  "."

$doc_2
[1] "talk"    "to"      "the"     "hand"    "because" "the"     "face"
[8] "don"     ""        "t"       "want"    "to"      "hear"    "it"
[15] "any"     "more"    "."

```

2.3. Other Transformer models

The package allows to get embeddings alongside other Transformer architectures as well. To get these models, use the `transformer` function and provide the appropriate model name and architecture. The package works with the following architectures: 'BERT', 'GPT', 'GPT-2', 'CTRL', 'Transformer-XL', 'XLNet', 'XLM', 'DistilBERT', 'RoBERTa' and 'XLM-RoBERTa'.

Some examples are given below. More model names can be found at <https://github.com/huggingface/transformers#model-architectures>.

```

model <- transformer("bert-base-multilingual-uncased")
model <- transformer("bert-base-multilingual-cased")
model <- transformer("bert-base-dutch-cased")
model <- transformer("bert-base-uncased")
model <- transformer("bert-base-cased")
model <- transformer("bert-base-chinese")
model <- transformer("xlm-roberta-base", architecture = "XLM-RoBERTa")
model <- transformer("distilbert-base-german-cased", architecture = "DistilBERT")
model <- transformer("distilbert-base-multilingual-cased", architecture = "DistilBERT")
model <- transformer("distilroberta-base", architecture = "DistilBERT")
model <- transformer("distilbert-base-cased", architecture = "DistilBERT")
embedding <- predict(model, "Just give me these embeddings please.")

```

2.4. Fun

I basically said 'embeddings, nothing more, nothing less'. Unfortunately I couldn't resist in also wrapping an example of the *generate* functionality to see what a GPT model would provide as extra text following your sequence of words.

```

model <- transformer("distilgpt2", architecture = "GPT-2-LMHead")

newtxt <- predict(model, c("I am thinking about ",
                          "Roses are red, violets are",
                          "R is a programming language for"),
                  type = "generate", max_length = 15L)

newtxt

```

```

$`1`
[1] "I am thinking about the future of the world. I am thinking about the"

```

```

$`2`
[1] "Roses are red, violets are red, and the red is"

```

```

$`3`
[1] "R is a programming language for programming languages. It is a programming language f

```

Enjoy!

Affiliation:

BNOSAC - Open Analytical Helpers

E-mail: jwijffels@bnosac.be

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