Using the tm.plugin.koRpus Package for Corpus Analysis

m.eik michalke

November 13, 2017

The R package `tm.plugin.koRpus` is an extension to the `koRpus` package, enhancing its usability for actual corpus analysis. It adds new classes and methods to `koRpus`, which are designed to work with complete text corpora in both `koRpus` and `tm` formats. This vignette gives you a quick overview.

1 What is tm.plugin.koRpus?

While the `koRpus` package focusses mostly on analysis steps of individual texts, `tm.plugin.koRpus` adds several new object classes and respective methods, which can be used to analyse complete text corpora in a single step. These classes are also a first step to combine object classes of both, the `koRpus` and `tm` packages.

There are three basic classes, which are hierarchically nested:

- class `kRp.topicCorpus` holds a list (named by topics) of objects of
  - class `kRp.sourcesCorpus`, which in its `sources` slot holds a list of objects of
    - class `kRp.corpus`, which in turn contains objects of both `koRpus` and `tm` classes.

The idea behind this is to be able to categorize corpora on at least two levels. The default assumes that these levels are different *sources* and different *topics*, but apart from this naming (which is coded into the classes) you can actually use this for whatever levels you like.

If you don't need these levels, you can just use the function `simpleCorpus()` to create objects of class `kRp.corpus`. It represents a flat corpus of texts. To distinguish texts which came from different sources, use the function `sourcesCorpus()`, which will generate sub-corpora for each source given. And one level higher up, use the function `topicCorpus()`, to sort `kRp.sourcesCorpus` objects by different topics. Objects of this class will only be valid if there are texts of each topic from each source.

2 Tokenizing corpora

As with `koRpus`, the first step for text analysis is tokenizing and possibly POS tagging. This step is performed by the functions mentioned above, `simpleCorpus()`, `sourcesCorpus()`, or `topicCorpus()`, respectively. The package includes four sample texts taken from Wikipedia[1] in its `tests` directory we can use for an elaborate demonstration:

```
> library(tm.plugin.koRpus)
> # set tha root path to the sample files
> sampleRoot <- file.path(path.package("tm.plugin.koRpus"), "tests",
+   "testthat", "samples")
> # now we can define the topics (names of the vector elements)
> # and their main path
> samplePaths <- c(
>   C3S=file.path(sampleRoot, "C3S"),
>   GEMA=file.path(sampleRoot, "GEMA")
> )
> # we also define the sources
> sampleSources <- c(
>   wpa="Wikipedia_alt",
>   wpn="Wikipedia_neu"
> )
> # and finally, we can tokenize all texts
> sampleTexts <- topicCorpus(paths=samplePaths, sources=sampleSources,
+   tagger="tokenize", lang="de")
```

```
processing topic "C3S", source "Wikipedia_alt", 1 texts...
processing topic "C3S", source "Wikipedia_neu", 1 texts...
processing topic "GEMA", source "Wikipedia_alt", 1 texts...
processing topic "GEMA", source "Wikipedia_neu", 1 texts...
```

Should you need to get hold of the nested objects inside `kRp.sourcesCorpus` or `kRp.topicCorpus` class objects, or replace them with updated ones, you can do so by using the methods `corpusTagged()`, `corpusSources()`, or `corpusTopics()`:

```
> allC3SSources <- corpusSources(corpusTopics(sampleTexts, "C3S"))
> names(allC3SSources)
```

```
[1] "wpa" "wpn"
```

3 Analysing corpora

After the initial tokenizing, we can analyse the corpus by calling the provided methods, for instance lexical diversity:

[1] see the file `tests/testthat/samples/License_of_sample_texts.txt` for details

```
> sampleTexts <- lex.div(sampleTexts, char=FALSE, quiet=TRUE)
> corpusSummary(sampleTexts)

               text topic source CTTR HD-D (vocd-D) Herdan's C Maas a Maas lgV0
wpaC3S01   wpaC3S01   C3S    wpa 6.13         38.14       0.95   0.16      6.21
wpnC3S01   wpnC3S01   C3S    wpn 6.82         38.05       0.94   0.17      6.10
wpaGEMA01 wpaGEMA01  GEMA    wpa 7.07         37.61       0.94   0.17      6.11
wpnGEMA01 wpnGEMA01  GEMA    wpn 7.13         37.87       0.94   0.16      6.24
          MATTR MSTTR   MTLD MTLD-MA Root TTR Summer  TTR Uber index Yule's K
wpaC3S01   0.81  0.79 100.16      NA     8.68   0.93 0.78      39.92    49.92
wpnC3S01   0.82  0.76 123.01      NA     9.65   0.92 0.73      36.46    54.88
wpaGEMA01  0.80  0.78 106.94     192    10.00   0.92 0.71      35.96    65.08
wpnGEMA01  0.81  0.79 111.64      NA    10.08   0.92 0.73      37.47    60.14
```

As you can see, `corpusSummary()` fetches a data frame from the object which contains the summarised results of all texts below the given object level. That is, if wou are only interested in the results for texts of the first topic, simply apply `corpusSummary()` on the result of `corpusTopics()`:

```
> corpusSummary(corpusTopics(sampleTexts, "C3S"))

             text topic source CTTR HD-D (vocd-D) Herdan's C Maas a Maas lgV0
wpaC3S01 wpaC3S01   C3S    wpa 6.13         38.14       0.95   0.16      6.21
wpnC3S01 wpnC3S01   C3S    wpn 6.82         38.05       0.94   0.17      6.10
         MATTR MSTTR   MTLD MTLD-MA Root TTR Summer  TTR Uber index Yule's K
wpaC3S01  0.81  0.79 100.16      NA     8.68   0.93 0.78      39.92    49.92
wpnC3S01  0.82  0.76 123.01      NA     9.65   0.92 0.73      36.46    54.88
```

There are quite a number of `corpus*()` getter/setter methods for slots of these objects, e.g., `corpusReadability()` to get the `readability()` results from objects of class `kRp.corpus`.

Using the summary data.frame, you could now perform ANOVAs or plot interactions:

```
> library(sciplot)
> lineplot.CI(
+   x.factor=corpusSummary(sampleTexts)[["source"]],
+   response=corpusSummary(sampleTexts)[["MTLD"]],
+   group=corpusSummary(sampleTexts)[["topic"]],
+   type="l",
+   main="MTLD",
+   xlab="Media source",
+   ylab="Lexical diversity score",
+   col=c("grey", "black"),
+   lwd=2
+ )
```

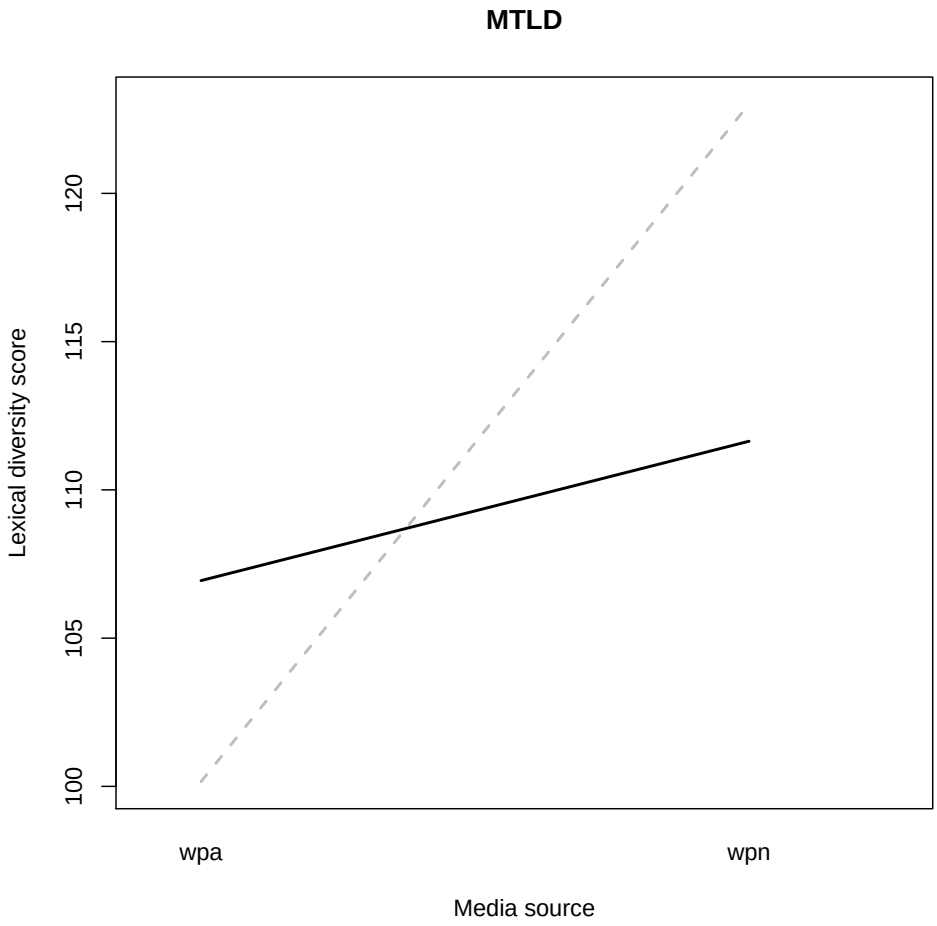

Figure 1: Well, the example texts aren't so impressive here, as there's not much variance in one text per source and topic.

3.1 Frequency analysis

The object classes make it quite comfortable to analyse type frequencies of corpora. There is a method `read.corp.custom()` for these classes, that will do this analysis recursively on all levels:

```
> sampleTexts <- read.corp.custom(sampleTexts, caseSens=FALSE)
> sampleTextsWordFreq <- query(corpusFreq(sampleTexts), var="wclass",
+   query=kRp.POS.tags(lang="de", list.classes=TRUE, tags="words"))
```

```
> head(sampleTextsWordFreq, 10)
```

	num	word	lemma	tag	wclass	lttr	freq	pct	pmio	log10
3	3	die		word.kRp	word	3	30	0.037220844	37220	4.570776
4	4	der		word.kRp	word	3	21	0.026054591	26054	4.415874
5	5	gema		word.kRp	word	4	17	0.021091811	21091	4.324097
6	6	und		word.kRp	word	3	17	0.021091811	21091	4.324097
7	7	einer		word.kRp	word	5	12	0.014888337	14888	4.172836
8	8	von		word.kRp	word	3	12	0.014888337	14888	4.172836
11	11	ist		word.kRp	word	3	10	0.012406948	12406	4.093632
12	12	bei		word.kRp	word	3	9	0.011166253	11166	4.047898
13	13	das		word.kRp	word	3	8	0.009925558	9925	3.996731
14	14	urheber		word.kRp	word	7	8	0.009925558	9925	3.996731

	rank.avg	rank.min	rank.rel.avg	rank.rel.min	inDocs	idf
3	263.0	263	99.24528	99.24528	4	0
4	262.0	262	98.86792	98.86792	4	0
5	260.5	260	98.30189	98.11321	4	0
6	260.5	260	98.30189	98.11321	4	0
7	258.5	258	97.54717	97.35849	4	0
8	258.5	258	97.54717	97.35849	4	0
11	256.0	255	96.60377	96.22642	4	0
12	254.0	254	95.84906	95.84906	4	0
13	252.5	252	95.28302	95.09434	4	0
14	252.5	252	95.28302	95.09434	2	0.301029995663981

In combination with the `wordcloud` package, this can directly be used to plot word clouds:

```
> require(wordcloud)
> colors <- brewer.pal(8, "RdGy")
> wordcloud(
+    head(sampleTextsWordFreq[["word"]], 200),
+    head(sampleTextsWordFreq[["freq"]], 200),
+    random.color=TRUE,
+    colors=colors
+  )
```

Figure 2: The 200 most frequent words in the example corpus