Introduction to **tm** — Text Mining in R

Ingo Feinerer

January 12, 2007

Abstract

This vignette gives a short overview over available features in the **tm** package for text mining purposes in R.

Loading the Package

Before actually working we need to load the package:

```
> library("tm")
```

Data Import

The main structure for managing documents is a socalled text document collection (TextDocCol). Its constructor takes following arguments:

- `object`: a Source object which abstracts the input location
- `parser`: a parser which constructs a text document from a single element delivered by a source. A parser must have the argument signature `(elem, load, id)`. The first argument is the element provided from the source, the second indicates whether the user wants to load the documents immediately into memory, and the third is a unique identification string.
- `...`: formally if the passed over `parser` function is of class `FunctionGenerator`, it is assumed to be a function generating a parser. This way custom parsers taking various parameters (specified in `...`) can be built, which in fact must produce a valid parser signature but can access additional parameters via lexical scoping (i.e., by the including environment).

Available sources are DirSource, CSVSource, GmaneSource and ReutersSource which handle a directory, a mixed CSV, a Gmane mailing list archive RSS feed or a mixed Reuters file (mixed means several documents are in a single file). Except DirSource, which is designated solely for directories on a file system, all other implemented sources can take connections as input (a character string is interpreted as filename).

This package ships with several readers (`readPlain()` (default), `readRCV1()`, `readReut21578XML()`, `readGmane()` and `readNewsgroup()`). The default just reads in the whole input file and interprets the content as text.

Plain text files in a directory:

```
> txt <- system.file("texts", "txt", package = "tm")
> (Ovid.tdc <- TextDocCol(DirSource(txt), load = TRUE))

A text document collection with 5 text documents
```

A single comma separated values file:

```
> cars.csv <- system.file("texts", "cars.csv", package = "tm")
> TextDocCol(CSVSource(cars.csv))

A text document collection with 5 text documents
```

Reuters21578 files either in directory (one document per file) or a single file (several documents per file). Note that connections can be used as input:

```
> reut21578 <- system.file("texts", "reut21578", package = "tm")
> reut21578.xml <- system.file("texts", "reut21578.xml", package = "tm")
> reut21578.xml.gz <- system.file("texts", "reut21578.xml.gz",
+     package = "tm")
> (reut21578.tdc <- TextDocCol(DirSource(reut21578), readReut21578XML))

A text document collection with 10 text documents

> TextDocCol(ReutersSource(reut21578.xml), readReut21578XML)

A text document collection with 10 text documents

> TextDocCol(ReutersSource(gzfile(reut21578.xml.gz)), readReut21578XML)

A text document collection with 10 text documents
```

Analogously for files in the Reuters Corpus Volume 1 format:

```
> rcv1 <- system.file("texts", "rcv1", package = "tm")
> rcv1.xml <- system.file("texts", "rcv1.xml", package = "tm")
> TextDocCol(DirSource(rcv1), readRCV1, load = TRUE)

A text document collection with 2 text documents

> TextDocCol(ReutersSource(rcv1.xml), readRCV1)

A text document collection with 2 text documents
```

Or mails from newsgroups (as found in the UCI KDD newsgroup dataset):

```
> newsgroup <- system.file("texts", "newsgroup", package = "tm")
> TextDocCol(DirSource(newsgroup), readNewsgroup, load = TRUE)

A text document collection with 6 text documents
```

RSS feed as delivered by Gmane for the R mailing list archive:

```
> rss <- system.file("texts", "gmane.comp.lang.r.gr.rdf", package = "tm")
> TextDocCol(GmaneSource(rss), readGmane)

A text document collection with 21 text documents
```

Inspecting the Text Document Collection

Custom `show` and `summary` methods are available, which hide the raw amount of information (consider a collection could consists of several thousand documents, like a database). `summary` gives more details on metadata than `show`, whereas in order to actually see the content of text documents use the command `inspect` on a collection.

```
> show(Ovid.tdc)

A text document collection with 5 text documents

> summary(Ovid.tdc)

A text document collection with 5 text documents

The metadata consists of 2 tag-value pairs and a data frame
Available tags are:
create_date creator
Available variables in the data frame are:
MetaID

> inspect(Ovid.tdc[1:2])

A text document collection with 2 text documents

The metadata consists of 2 tag-value pairs and a data frame
Available tags are:
create_date creator
Available variables in the data frame are:
MetaID

[[1]]
An object of class ■PlainTextDocument■
 [1] "    Si quis in hoc artem populo non novit amandi,"
 [2] "         hoc legat et lecto carmine doctus amet."
 [3] "    arte citae veloque rates remoque moventur,"
 [4] "         arte leves currus: arte regendus amor."
 [5] ""
 [6] "    curribus Automedon lentisque erat aptus habenis,"
 [7] "         Tiphys in Haemonia puppe magister erat:"
 [8] "    me Venus artificem tenero praefecit Amori;"
 [9] "         Tiphys et Automedon dicar Amoris ego."
[10] "    ille quidem ferus est et qui mihi saepe repugnet:"
[11] ""
[12] "         sed puer est, aetas mollis et apta regi."
[13] "    Phillyrides puerum cithara perfecit Achillem,"
[14] "         atque animos placida contudit arte feros."
[15] "    qui totiens socios, totiens exterruit hostes,"
[16] "         creditur annosum pertimuisse senem."
Slot "URI":
```

```
file("/home/feinerer/Rlibrary/tm/texts/txt/ovid_1.txt")

Slot "Cached":
[1] TRUE

Slot "Author":
[1] ""

Slot "DateTimeStamp":
[1] "2007-01-12 10:28:57 CET"

Slot "Description":
[1] ""

Slot "ID":
[1] "1"

Slot "Origin":
[1] ""

Slot "Heading":
[1] ""

Slot "LocalMetaData":
list()


[[2]]
An object of class ■PlainTextDocument■
 [1] "    quas Hector sensurus erat, poscente magistro"
 [2] "         verberibus iussas praebuit ille manus."
 [3] "    Aeacidae Chiron, ego sum praeceptor Amoris:"
 [4] "         saevus uterque puer, natus uterque dea."
 [5] "    sed tamen et tauri cervix oneratur aratro,"
 [6] ""
 [7] "         frenaque magnanimi dente teruntur equi;"
 [8] "    et mihi cedet Amor, quamvis mea vulneret arcu"
 [9] "         pectora, iactatas excutiatque faces."
[10] "    quo me fixit Amor, quo me violentius ussit,"
[11] "         hoc melior facti vulneris ultor ero:"
[12] ""
[13] "    non ego, Phoebe, datas a te mihi mentiar artes,"
[14] "         nec nos aëriae voce monemur avis,"
[15] "    nec mihi sunt visae Clio Cliusque sorores"
[16] "         servanti pecudes vallibus, Ascra, tuis:"
[17] "    usus opus movet hoc: vati parete perito;"
Slot "URI":
file("/home/feinerer/Rlibrary/tm/texts/txt/ovid_2.txt")

Slot "Cached":
```

```
[1] TRUE

Slot "Author":
[1] ""

Slot "DateTimeStamp":
[1] "2007-01-12 10:28:57 CET"

Slot "Description":
[1] ""

Slot "ID":
[1] "2"

Slot "Origin":
[1] ""

Slot "Heading":
[1] ""

Slot "LocalMetaData":
list()
```

Transformations

Once we have a text document collection one typically wants to modify the documents in it, e.g., stemming, stopword removal, et cetera. All this functionality is subsumed by the concept of *transformation*s in **tm**. Transformations are done via the `tmMap` function which applies a function to all elements of the collection. Basically, all transformations work on single text documents and `tmMap` just applies them to all documents in a document collection.

Loading Documents into Memory

If the source objects supports load on demand, but the user has not enforced the package to load the input content directly into memory, this can be done manually via `loadDoc`. Normally it is not necessary to call this explicitly, as other functions working on text corpora trigger this function for not-loaded documents (the corpus is automatically loaded if accessed via `[[`).

```
> reut21578.tdc <- tmMap(reut21578.tdc, loadDoc)
```

Converting to Plaintext Documents

The text document collection `reut21578.tdc` contains documents in XML format. We have no further use for the XML interna and just want to work with the text content. This can be done by converting the documents to plaintext documents. It is done by the generic `asPlain` in assistance by a converter function `convertReut21578XMLPlain` which knows how to actually do it.

```
> reut21578.tdc <- tmMap(reut21578.tdc, asPlain, convertReut21578XMLPlain)
```

Eliminating Extra Whitespace

Extra whitespace is eliminated by:

```
> reut21578.tdc <- tmMap(reut21578.tdc, stripWhitespace)
```

Convert to Lower Case

Conversion to lower case by:

```
> reut21578.tdc <- tmMap(reut21578.tdc, tmTolower)
```

Remove Stopwords

Removal of stopwords by:

```
> data(stopwords_en)
> reut21578.tdc <- tmMap(reut21578.tdc, removeWords, stopwords_en)
```

Stemming

Stemming is done by:

```
> if (require("Rstem")) tmMap(reut21578.tdc, stemDoc)

A text document collection with 10 text documents
```

Filters

Often it is of special interest to filter out documents satisfying given properties. For this purpose the function `tmFilter` is designated. It is possible to write custom filter functions, but for most cases the default filter does its job: it integrates a minimal query language to filter metadata. Statements in this query language are statements as used for subsetting data frames.

E.g., the following statement filters out those documents having `COMPUTER TERMINAL SYSTEMS <CPML> COMPLETES SALE` as their heading and an `ID` equal to 10 (both are metadata slot variables of the text document).

```
> query <- "identifier == '10' & heading == 'COMPUTER TERMINAL SYSTEMS <CPML> COMPLETES SALE
> tmFilter(reut21578.tdc, query)

A text document collection with 1 text document
```

There is also a full text search filter available which accepts regular expressions:

```
> tmFilter(reut21578.tdc, FUN = searchFullText, "partnership",
+     doclevel = TRUE)

A text document collection with 1 text document
```

Adding Data or Metadata

Text documents or metadata can be added to text document collections with `appendElem` and `appendMeta`, respectively. The text document collection has two types of metadata: one is the metadata on the document collection level (`cmeta`), the other is the metadata related to the documents (e.g., clusterings) (`dmeta`) in form of a dataframe. For the method `appendElem` it is possible to give a row of values in the dataframe for the added data element.

```
> data(crude)
> reut21578.tdc <- appendElem(reut21578.tdc, crude[[1]], 0)
> reut21578.tdc <- appendMeta(reut21578.tdc, cmeta = list(test = c(1,
+     2, 3)), dmeta = list(cl1 = 1:11))
> summary(reut21578.tdc)

A text document collection with 11 text documents

The metadata consists of 3 tag-value pairs and a data frame
Available tags are:
create_date creator test
Available variables in the data frame are:
MetaID cl1

> CMetaData(reut21578.tdc)

An object of class ■MetaDataNode■
Slot "NodeID":
[1] 0

Slot "MetaData":
$create_date
[1] "2007-01-12 10:28:57 CET"

$creator
   LOGNAME
"feinerer"

$test
[1] 1 2 3


Slot "children":
list()

> DMetaData(reut21578.tdc)

   MetaID cl1
1       0   1
2       0   2
3       0   3
4       0   4
```

```
5       0   5
6       0   6
7       0   7
8       0   8
9       0   9
10      0  10
11      0  11
```

Removing Metadata

The metadata of text document collections can be easily modified or removed:

```
> data(crude)
> reut21578.tdc <- removeMeta(reut21578.tdc, cname = "test",
+     dname = "cl1")
> CMetaData(reut21578.tdc)

An object of class ■MetaDataNode■
Slot "NodeID":
[1] 0

Slot "MetaData":
$create_date
[1] "2007-01-12 10:28:57 CET"

$creator
   LOGNAME
"feinerer"


Slot "children":
list()

> DMetaData(reut21578.tdc)

   MetaID
1       0
2       0
3       0
4       0
5       0
6       0
7       0
8       0
9       0
10      0
11      0
```

Operators

Most standard operators ([, [<-, [[, [[<-, c, length) are available for text document collections with semantics similar to standard R routines. E.g. c concatenates two (or more) text document collections. Applied to several text documents it returns a text document collection. The metadata is automatically updated, if text document collections are concatenated (i.e., merged).

Note also the custom element-of operator—it checks whether a text document is already in a text document collection (metadata is not checked, only the corpus):

```
> crude[[1]] %IN% reut21578.tdc

[1] TRUE

> crude[[2]] %IN% reut21578.tdc

[1] FALSE
```

Keeping Track of Text Document Collections

There is a mechanism available for managing text document collections. It is called TextRepository. A typical use would be to save different states of a text document collection. A repository has metadata in list format which can be either set with appendElem as additional argument (e.g., a date when a new element is added), or directly with appendMeta.

```
> data(acq)
> repo <- TextRepository(reut21578.tdc)
> repo <- appendElem(repo, acq, list(modified = date()))
> repo <- appendMeta(repo, list(moremeta = 5:10))
> summary(repo)

A text repository with 2 text document collections

The repository metadata consists of 3 tag-value pairs
Available tags are:
created modified moremeta

> RepoMetaData(repo)

$created
[1] "2007-01-12 10:29:00 CET"

$modified
[1] "Fri Jan 12 10:29:00 2007"

$moremeta
[1]  5  6  7  8  9 10

> summary(repo[[1]])
```

```
A text document collection with 11 text documents

The metadata consists of 2 tag-value pairs and a data frame
Available tags are:
create_date creator
Available variables in the data frame are:
MetaID
```

```
> summary(repo[[2]])
```

```
A text document collection with 50 text documents

The metadata consists of 2 tag-value pairs and a data frame
Available tags are:
create_date creator
Available variables in the data frame are:
MetaID
```

Creating Term-Document Matrices

A common approach in text mining is to create a term-document matrix for given texts. In this package the class TermDocMatrix handles this for text document collections.

```
> tdm <- TermDocMatrix(reut21578.tdc)
> tdm[1:8, 50:55]
```

```
    terms
docs convertible covertible crop cruzados cumulative currency
   1           1          1    5        1          1        2
   2           0          0    0        0          0        0
   3           0          0    0        0          0        1
   4           1          0    0        0          0        0
   5           0          0    0        0          0        0
   6           0          0    1        0          0        0
   7           0          0    0        0          0        0
   8           0          0    0        0          0        0
```

Operations on Term-Document Matrices

Besides the fact that on this matrix a huge amount of R functions (like clustering, classifications, etc.) is possible, this package brings some shortcuts. Consider we want to find those terms that occur at least 5 times:

```
> findFreqTerms(tdm, 5, Inf)
```

```
 [1] "bags"        "cocoa"       "comissaria"  "crop"        "dec"
 [6] "dlrs"        "july"        "mln"         "sales"       "sept"
[11] "smith"       "times"       "york"        "analysts"    "bankamerica"
[16] "debt"        "stock"       "price"       "level"       "total"
[21] "apr"         "feb"         "mar"         "nil"         "prev"
[26] "computer"    "terminal"    "oil"         "the"
```

Or we want to find associations (i.e., terms which correlate) with at least 0.97 correlation for the term `crop`:

```
> findAssocs(tdm, "crop", 0.97)
```

crop	155	221	325	340
1.00	0.98	0.98	0.98	0.98
345	350	351	375	380
0.98	0.98	0.98	0.98	0.98
400	415	450	480	750
0.98	0.98	0.98	0.98	0.98
753	780	785	850	870
0.98	0.98	0.98	0.98	0.98
875	880	995	alleviating	areas
0.98	0.98	0.98	0.98	0.98
argentina	arrivals	arroba	aug	bags
0.98	0.98	0.98	0.98	0.98
bahia	+bahia	bean	booked	butter
0.98	0.98	0.98	0.98	0.98
buyers	cake	carnival	certificates	cocoa
0.98	0.98	0.98	0.98	0.98
comissaria	consignment	continued	covertible	cruzados
0.98	0.98	0.98	0.98	0.98
cumulative	currently	dec	delivered	destinations
0.98	0.98	0.98	0.98	0.98
dificulties	doubt	doubts	drought	dry
0.98	0.98	0.98	0.98	0.98
end	estimated	estimates	experiencing	exporters
0.98	0.98	0.98	0.98	0.98
farmers	final	fit	fob	hands
0.98	0.98	0.98	0.98	0.98
harvesting	held	humidity	hundred	improving
0.98	0.98	0.98	0.98	0.98
included	june	kilos	levels	liquor
0.98	0.98	0.98	0.98	0.98
may	means	midday	middlemen	named
0.98	0.98	0.98	0.98	0.98
nearby	normal	obtaining	period	ports
0.98	0.98	0.98	0.98	0.98
practically	processors	prospects	published	quality
0.98	0.98	0.98	0.98	0.98
registered	reluctant	restored	review	rose
0.98	0.98	0.98	0.98	0.98
routine	sales	season	selling	sept
0.98	0.98	0.98	0.98	0.98
shipment	shippers	showers	spot	stage
0.98	0.98	0.98	0.98	0.98
superior+	temporao	thousand	throughout	times
0.98	0.98	0.98	0.98	0.98
tonne	trade	uruguay	view	weekly
0.98	0.98	0.98	0.98	0.98

```
york         zone
0.98         0.98
```

The function also accepts a matrix as first argument (which does not inherit from a term-document matrix). This matrix is then interpreted as a correlation matrix and directly used. With this approach different correlation measures can be employed.