Text Mining in R - das tm Paket

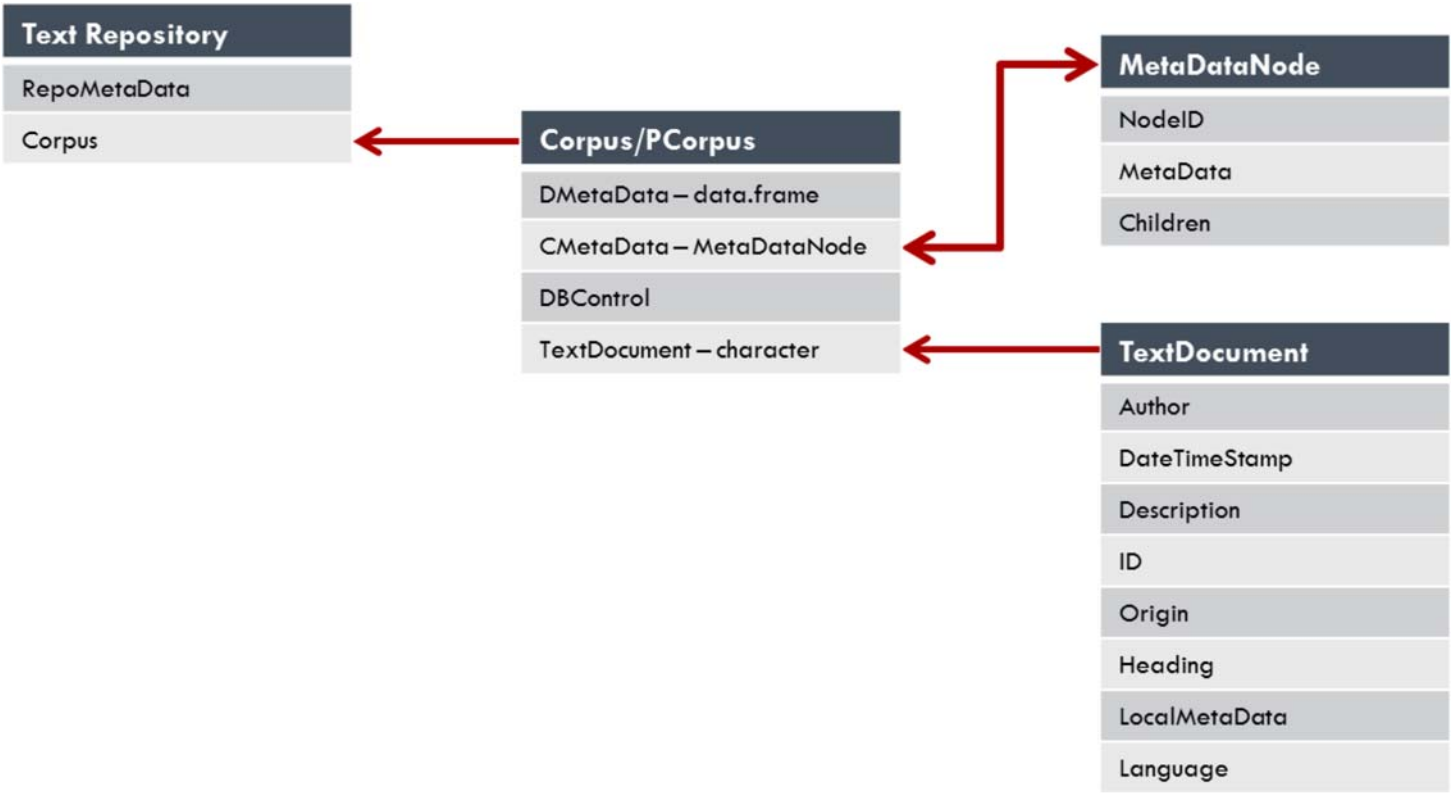
06.04.2014

Zielsetzung und Datenquellen

In dieser Seminararbeit bei Herrn Prof. Filzmoser (http://www.statistik.tuwien.ac.at/public/filz/) an der TU Wien (http://www.tuwien.ac.at/) werde ich die Funktionalität des R Pakets *tm — Text Mining Package* (http://tm.r-forge.r-project.org/) (v. 0.5-9.1)(Feinerer and Hornik 2013; Feinerer, Hornik, and Meyer 2008; Feinerer 2008) vorstellen und die wichtigsten Funktionen auf mitgelieferten Daten sowie Publikationen der *Verlag für die Deutsche Wirtschaft AG* (http://vnrag.de/) anwenden.

Daten-Struktur in R

Wichtigstes Objekt zur Speicherung von Dokumenten und Meta-Daten der Dokumente im *tm*-Paket ist der `corpus`, der die Dokumente als Textkorpus (http://de.wikipedia.org/wiki/Textkorpus) zusammenfasst und es einfach macht, sie mittels anderen Funktionen des *tm*-Pakets weiter zu verarbeiten. Der *corpus* ist wie folgt aufgebaut:



Im *tm*-Paket gibt es zwei Arten von Korpora: `Corpus` bzw. `VCorpus` und `PCorpus`. Der Unterschied ist, dass beim `PCorpus` so wenig Dokumente wie möglich in den RAM geladen werden um das Arbeiten mit sehr großen Dokumentensammlungen zu ermöglichen. Nachteil ist, dass das Zugreifen auf die einzelnen Dokumente verlangsamt wird. Realisiert wird dies durch eine Datenbank aus dem *filehash* (http://cran.r-project.org/web/packages/filehash/index.html)-Paket (Peng 2006). Auf nähere Details zur Realisierung werde ich nicht weiter eingehen. Der `Corpus` bzw. `VCorpus` lädt direkt alle Dokumente in den RAM.

Ein Korpus ist im Wesentlichen eine Liste mit oben aufgeführten Listenelementen, die folgende Informationen enthalten:

- **DMetaData**
  Spezifische Metadaten je Dokument (z.B. eine Klassifikation)
- **CMetaData**
  Kollektions-Metadaten (z.B. Autor und Erstellungsdatum)
- **DBControl** (nur bei PCorpus)

Welche Datenbank (aus dem *filehash*-Paket) wird benutzt

- **TextDocument**
  Für jedes Dokument im Korpus einen Listeneintrag vom Typ *TextDocument* mit den aufgeführten Attributen

Mehrere Korpora können in einem `TextRepository` zusammengefasst werden um Transformationen der Texte nachvollziehbar zu machen. Mit “Transformationen” sind z.B. das Löschen von Nummern und Sonderzeichen gemeint.

Dokumente einlesen, transformieren und filtern

Für das Einlesen, Transformieren und Filtern von Dokumenten nach bestimmten Charakteristika stellt das *tm*-Paket einige Funktionen zur Verfügung.

Einlesen

Zum Einlesen verwendet das *tm*-Paket zwei wichtige Funktionstypen: ***sources*** und ***reader***

Mittels sogenannter “sources” also Quellen gibt man dem Korpus an, wo er welche Dokumente in welchem Format findet. Durch die Funktion `getSources()` erhält man alle verfügbaren Quell-Typen. `DirSources` beispielsweise erstellt eine Liste mit allen Dateien in einem Ordner, die ein bestimmtes Muster erfüllen.

```
getSources()
```

```
## [1] "DataframeSource" "DirSource"       "GmaneSource"     "ReutersSource"
## [5] "URISource"       "VectorSource"
```

Sogenannte “reader” werden verwendet um Dokumente in verschiedensten Formaten in reinen Text umzuwandeln und Metadaten wie den Autor oder das Erstellungsdatum auszulesen. Mittels `getReaders()` erhält man eine Liste aller verfügbaren “reader”.

```
getReaders()
```

```
##  [1] "readDOC"                "readGmane"
##  [3] "readPDF"                "readReut21578XML"
##  [5] "readReut21578XMLasPlain" "readPlain"
##  [7] "readRCV1"               "readRCV1asPlain"
##  [9] "readTabular"            "readXML"
```

Auch kann man sehr einfach eigene reader schreiben.
Die Dokumente der *Verlag für die Deutsche Wirtschaft AG* sind beispielsweise im HTM-Format. Um sie einzulesen kann man beispielsweise die C++ - Routine html2text (http://manpages.ubuntu.com/manpages/hardy/man1/html2text.1.html)(Unkrig and Bayer 2004) wie folgt in einem reader verwenden:

```
readHTML<-function(elem, language, id){
  content <- system2("html2text", shQuote(elem$uri), stdout = TRUE)
  PlainTextDocument(content, id = id, language = language,
                    origin=sub("/.*", "", sub(".*Publikationen/*", "", elem$uri)))
}
```

Dieser einfache “reader” erstellt ein `PlainTextDocument` mit den Attributen *id*, *language* und *origin*.

Nach dem Einlesen kann man dem Korpus noch *CMetaDaten* und *DMetaDaten* hinzufügen.

Beispiel

```
Dir<-DirSource(directory="/usr/lib/R/Publikationen",recursive = TRUE)
Publikationen <- Corpus(x=Dir,
                        readerControl = list(reader = readHTML,
                                             language="de"))

meta(Publikationen,"Themenbereich")<-unlist(meta(Publikationen,
                                                 tag = "Origin", type="local"))
TR<-TextRepository(Publikationen)
rm(Publikationen) # Da der Korpus ja jetzt im TextRepository gespeichert ist.
```

Transformieren

Um Transformations-Funktionen auf die Dokumente in einem Korpus anzuwenden verwendet das *tm*-Paket die Funktion `tm_map`. Mit der Funktion `getTransformations()` erhält man alle im *tm*-Paket implementierten Transformationen. Auch diese kann man natürlich selber schreiben und anpassen.

```
getTransformations()
```

```
## [1] "as.PlainTextDocument" "removeNumbers"        "removePunctuation"
## [4] "removeWords"          "stemDocument"         "stripWhitespace"
```

Beispiel

```
TR[[2]] <- tm_map(TR[[1]], stripWhitespace)
TR[[2]] <- tm_map(TR[[2]], removeNumbers)
TR[[2]] <- tm_map(TR[[2]], removePunctuation)
TR[[2]] <- tm_map(TR[[2]], tolower)
TR[[2]] <- tm_map(TR[[2]], function(x){removeWords(x,
                                    c("dass",stopwords("german")))})
TR[[2]] <- tm_map(TR[[2]], stripWhitespace)

ind <- which(Dir$FileList=="/usr/lib/R/Publikationen/Steuern/UST/UST_2012_01_Ausgabe.htm")

TR[[1]][[ind]][10:20] # Originaltext des erstes Dokuments: Zeilen 10-20
TR[[2]][[ind]][10:20] # Transformierter Text des erstes Dokumentes: Zeilen 10-20
```

```
##  [1] "Geschäftsräume vermieten reicht nicht für die"
##  [2] "Veräußerung eines Unternehmens im Ganzen Seite 3"
##  [3] "Bei Vermögensanlagen für Ihr Unternehmen:"
##  [4] "Wann sind Verwaltungsleistungen der Bank"
##  [5] "umsatzsteuerpflichtig? . . . . . Seite 6"
##  [6] "USt-Sonderprüfung: E-Mails sind für"
##  [7] "die Prüfer kein Tabu . . . . . Seite 8"
##  [8] "Generell gilt: Mit Ihrem Unternehmen"
##  [9] "sind Sie meldepflichtig, wenn Sie zum"
## [10] "Vorsteuerabzug berechtigt sind und in-"
## [11] "nerhalb der EU grenzüberschreitend"
```

```
##  [1] "geschäftsräume vermieten reicht "
##  [2] "veräußerung unternehmens ganzen seite "
##  [3] " vermögensanlagen unternehmen"
##  [4] "wann verwaltungsleistungen bank"
##  [5] "umsatzsteuerpflichtig seite "
##  [6] "ustsonderprüfung emails "
##  [7] " prüfer tabu seite "
##  [8] "generell gilt unternehmen"
##  [9] " meldepflichtig "
## [10] "vorsteuerabzug berechtigt "
## [11] "nerhalb eu grenzüberschreitend"
```

In den letzten beiden Zeile sieht man, dass das Wort “innerhalb” durch einen Bindestrich getrennt ist. Durch das Transformieren mit `removePunctuation` wird der Bindestrich entfernt und das Wort “in-” wird zu “in”. “in” wird nun durch `removeWords(x,stopwords("german"))` aus dem Text heraus gelöscht. Solche Fehler müsste man bei einer tiefergehenden Analyse natürlich vermeiden.

Filtern

Um den Korpus nach verschiedenen Kriterien zu filtern stehen die Funktionen `tm_filter` und `tm_index` zur Verfügung. Erstere gibt einen Teilkorpus zurück, der alle Dokumente enthält, bei dem eine Filterfunktion *WAHR* war, zweitere einen Vektor mit *WAHR/FALSCH*-Argumenten je Dokument. Mit `getFilters()` erhält man analog alle verfügbaren Filterfunktionen:

```
getFilters()
```

```
## [1] "searchFullText" "sFilter"        "tm_intersect"
```

Beispiel

```
TR[[2]] # Der Ungefilterte Korpus
```

```
## A corpus with 614 text documents
```

```
tm_filter(TR[[2]], FUN = searchFullText, "mitarbeiter") # Der gefilterte Korpus
```

```
## A corpus with 437 text documents
```

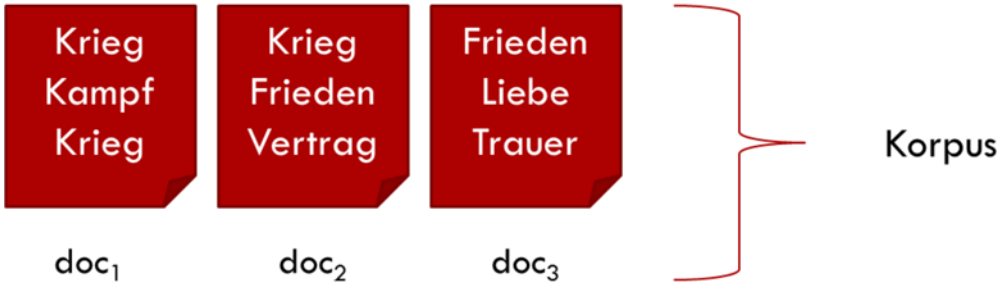
Es werden nur die Dokumente herausgefiltert, die das Wort “mitarbeiter” enthalten.

Anwendungsbeispiele

Die so aufbereiteten Daten kann man auf zahllose Arten weiterverarbeiten. Ein wichtiges Konzept ist dabei die Term-Document-Matrix. Diese kann man z.B. zur Clusteranalyse oder zur Klassifizierung von Dokumenten mittels *Support Vector Machines* (SVM) benutzen. Auch kann man die häufigsten Wörter eines Korpus in einer *Word Cloud* visualisieren.

Term-Document-Matrix

Das folgende Schaubild erklärt, was eine Document-Term-Matrix bzw. transponiert eine Term-Document-Matrix ist (Das Schaubild ist aus (Liegl 2009) übernommen)



Document-Term-Matrix

	Krieg	Kampf	Frieden	Vertrag	Liebe	Trauer
doc_1	2	1	0	0	0	0
doc_2	1	0	1	1	0	0
doc_3	0	0	1	0	1	1

Document Vector

Term Vector

Wie man erkennt ist es eine Matrix, die die Häufigkeiten der einzelnen Wörter je Dokument zählt. Diese Matrix ist im Allgemeinen schwach besetzt. Das *tm*-Paket erzeugt diese Matrizen mittels der `TermDocumentMatrixFixed`-Funktion. Diese Matrizen werden als schwachbesetzte Matrizen gespeichert (siehe `?slam:::simple_triplet_matrix`).
Einfache Funktionen die TermDocumentMatrix zu analysieren sind: `findFreqTerms`, `findAssocs`, `tm_tag_score` und `proxy:::dissimilarity`

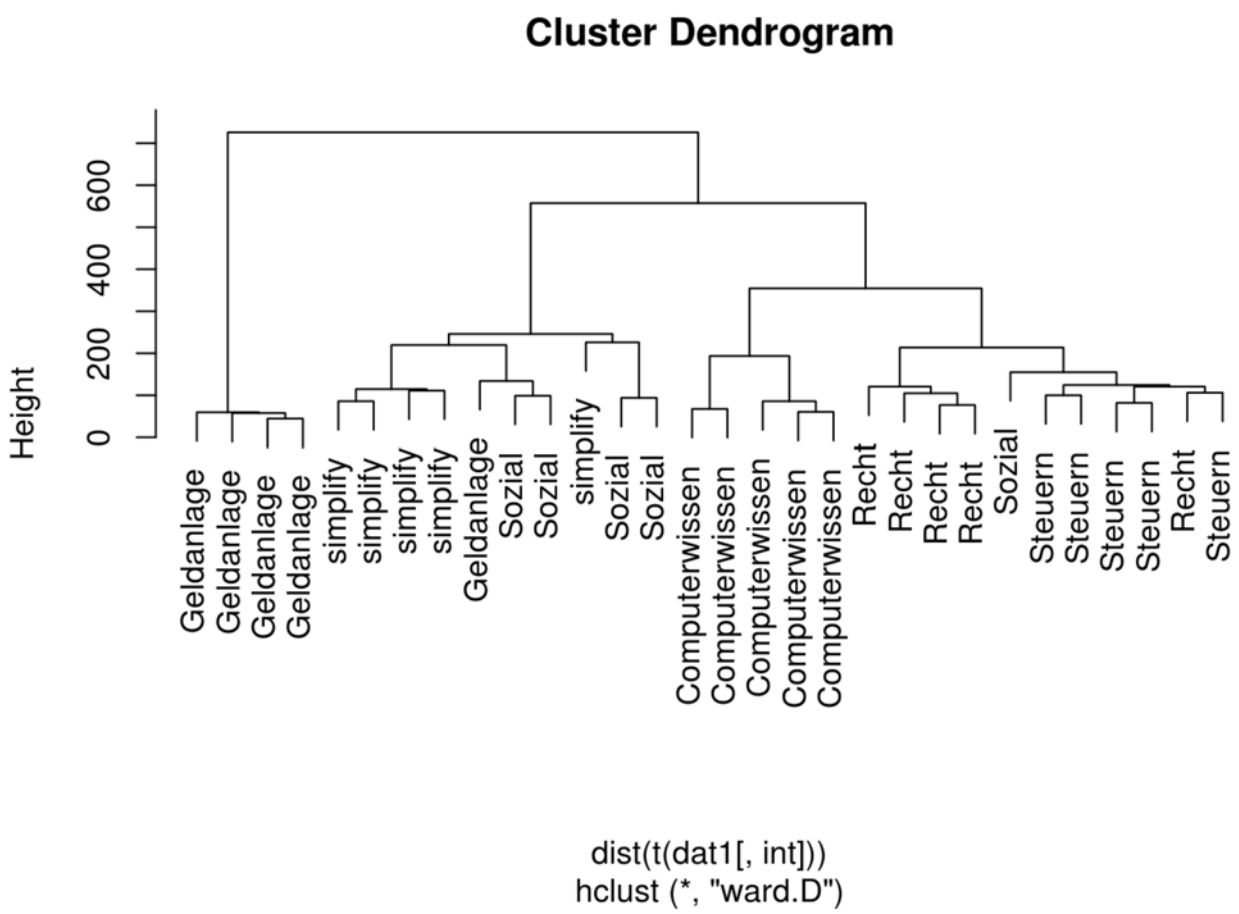
Erstellen der Term-Document-Matrizen

```
TDM1<-TermDocumentMatrix(TR[[1]]) # Original Texte
TDM2<-TermDocumentMatrix(TR[[2]]) # Transformierte Texte

# Gruppieren nach Themenbereich der Publikation

dat1<-as.matrix(TDM1) # Speichern als normale Matrix
# Reduktion auf die Wörter, die mehr als 600 mal vorkommen
dat1<-dat1[which(rownames(dat1)%in%findFreqTerms(TDM1,600,Inf)),]

dat2<-as.matrix(TDM2) # Speichern als normale Matrix
# Reduktion auf die Wörter, die mehr als 600 mal vorkommen
dat2<-dat2[which(rownames(dat2)%in%findFreqTerms(TDM2,600,Inf)),]
# Das Speichern der schwach besetzten Matrizen als "normale" Matrizen ist
# keineswegs speichereffizient. Dies dient lediglich der Lesbarkeit
# der kommenden Codes.
```

Clusteranalyse

Sehr einfach kann man z.B. hierarchisch clustern, wie folgendes Beispiel zeigt. Andere Cluster-Verfahren kann man genau so einfach anwenden.

```
int <- NULL
for (i in unique(DMetaData(TR[[1]])$Themenbereich)){
  int<-c(int,sample(which(DMetaData(TR[[1]])$Themenbereich==i),5))
}
plot(hclust(dist(t(dat1[,int])),
            method = "ward"),
     labels=DMetaData(TR[[1]])$Themenbereich[int])
```




```
hc.class1<-cutree(hclust(dist(t(dat1)), method = "ward"),6)
```

```
## The "ward" method has been renamed to "ward.D"; note new "ward.D2"
```

```
table(DMetaData(TR[[1]])$Themenbereich,hc.class1)
```

```
##                 hc.class1
##                    1   2   3   4   5   6
##   Computerwissen  62  60   0   0   0   0
##   Geldanlage       0   0  22 109   0   0
##   Recht            0   0   0   0  61   0
##   Sozial           0   0   0   0  34  76
##   Steuern          3   0  52   1  27   0
##   simplify         0   0  29   0  14  64
```

Support Vector Machines

SVM lernt an einem Trainings-Datensatz der Größe 100 und prognostiziert daraufhin die Klasse der übrigen 514 Datensätze.

```
require(kernlab)
train.ind <- sample(ncol(dat2),100)
train.class <- DMetaData(TR[[2]])$Themenbereich[train.ind]
test.class <- DMetaData(TR[[2]])$Themenbereich[-train.ind]

train2 <- t(dat2[,train.ind])
test2 <- t(dat2[,-train.ind])

ksvmTrain2 <- ksvm(train.class ~ . ,data = train2, kernal="rbfdot")
```

```
## Using automatic sigma estimation (sigest) for RBF or laplace kernel
```

```
svmCl2 <- predict(ksvmTrain2, test2)
(svmTable2 <- table(SVM = svmCl2, Richtig = test.class))
```

```
##                 Richtig
## SVM              Computerwissen Geldanlage Recht Sozial Steuern simplify
##   Computerwissen            102          0     0       0       0        0
##   Geldanlage                  0        112     0       0       0        1
##   Recht                       0          0    48       0       0        0
##   Sozial                      0          0     0      91       0        1
##   Steuern                     0          0     0       0      70        0
##   simplify                    0          0     0       0       1       88
```

Eine Missklassifikation von 0.0058 ist doch sehr ansehnlich, wenn man bedenkt, dass der Trainings-Datensatz weniger als 1/6 der Daten enthält. Um hier wirklich eine Aussage über die Prognose mittels SVM machen zu können, muss man dies natürlich öfter wiederholen. Für nähere Informationen über das *kernlab*-Paket siehe (Karatzoglou et al. 2004).

Word Cloud

Um die Term-Document-Matrizen zu visualisieren kann sie beispielsweise als Wordcloud darstellt werden:

```
require(wordcloud)

v<-sort(rowSums(dat2),decreasing=TRUE)
WC.df<-data.frame(word=names(v),freq=v)
RGB<-function(red, green, blue,...){ rgb(red, green, blue,maxColorValue = 255,...)}
set.seed(2)
wordcloud(WC.df$word,WC.df$freq,min.freq=1000,
          colors=c(brewer.pal(5, "Set1"),RGB(0,0,0)),
          scale=c(3,.01),
          use.r.layout=FALSE,
          rot.per=.15,max.words=Inf)
```

Leider ist die Visualisierung mittels des `wordcloud`-Paketes (Fellows 2013) nicht sehr gut implementiert, so dass es oft zu Fehlern kommt und wichtige Worte nicht dargestellt werden.

Beschreibung der Betriebsmittel

Das Projekt wurde auf einer *Amazon EC2-Instanz c1.xlarge* (http://aws.amazon.com/de/ec2/instance-types/) durchgeführt. Als Betriebssystem wurde *Ubuntu 12.04 LTS* (http://releases.ubuntu.com/12.04/) verwendet, als Webserver *nginx 1.1.19* (http://nginx.org/), als Datenbank *MariaDB 5.5.35* (https://mariadb.org/), als Scriptsprache *PHP 5.3.10-1ubuntu3.9* (http://www.php.net/) und um `R` (R Core Team 2013) zu verwenden *RStudio Server 0.98.484* (http://www.rstudio.com/ide/server/) (RStudio Team 2012). Dieses Dokument wurde mit *rmarkdown* (https://github.com/rstudio/rmarkdown) (RStudio and Inc. 2014), *pandoc* (http://johnmacfarlane.net/pandoc/) und *Tex Live* (https://www.tug.org/texlive/) erstellt.

Die `sessionInfo()` von `R` ist die folgende:

```
## R version 3.1.0 beta (2014-03-28 r65330)
## Platform: x86_64-pc-linux-gnu (64-bit)
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8 LC_NUMERIC=C         LC_TIME=C
##  [4] LC_COLLATE=C         LC_MONETARY=C        LC_MESSAGES=C
##  [7] LC_PAPER=C           LC_NAME=C            LC_ADDRESS=C
## [10] LC_TELEPHONE=C       LC_MEASUREMENT=C     LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] wordcloud_2.4        RColorBrewer_1.0-5   Rcpp_0.10.6
## [4] kernlab_0.9-19       tm_0.5-9.1           rmarkdown_0.1.65
## [7] knitrBootstrap_1.0.0
##
## loaded via a namespace (and not attached):
##  [1] codetools_0.2-8 digest_0.6.4    evaluate_0.5.1  formatR_0.10
##  [5] knitr_1.5.25    markdown_0.6.5  parallel_3.1.0  slam_0.1-30
##  [9] stringr_0.6.2   tools_3.1.0     yaml_2.1.11
```

Literaturverzeichnis

Feinerer, Ingo. 2008. “A Text Mining Framework in R and Its Applications.” PhD thesis, Department of Statistics; Mathematics, Vienna University of Economics; Business Administration. http://epub.wu-wien.ac.at/dyn/openURL?id=oai:epub.wu-wien.ac.at:epub-wu-01_e09 (http://epub.wu-wien.ac.at/dyn/openURL?id=oai:epub.wu-wien.ac.at:epub-wu-01_e09).

Feinerer, Ingo, and Kurt Hornik. 2013. *tm: Text Mining Package*. http://CRAN.R-project.org/package=tm (http://CRAN.R-project.org/package=tm).

Feinerer, Ingo, Kurt Hornik, and David Meyer. 2008. “Text Mining Infrastructure in R.” *Journal of Statistical Software* 25 (5) (March): 1–54. http://www.jstatsoft.org/v25/i05/ (http://www.jstatsoft.org/v25/i05/).

Fellows, Ian. 2013. *wordcloud: Word Clouds*. http://CRAN.R-project.org/package=wordcloud (http://CRAN.R-project.org/package=wordcloud).

Karatzoglou, Alexandros, Alex Smola, Kurt Hornik, and Achim Zeileis. 2004. “kernlab – an S4 Package for Kernel Methods in R.” *Journal of Statistical Software* 11 (9): 1–20. http://www.jstatsoft.org/v11/i09/ (http://www.jstatsoft.org/v11/i09/).

Liegl, Johannes. 2009. “Topic Classification in R.” december. http://wwwu.edu.uni-klu.ac.at/jliegl/TopicClassificationInR/Topic%20Classification%20in%20R.pptx (http://wwwu.edu.uni-klu.ac.at/jliegl/TopicClassificationInR/Topic%20Classification%20in%20R.pptx).

Peng, Roger D. 2006. “Interacting with Data Using the Filehash Package.” *R News* 6 (4): 19–24. http://CRAN.R-project.org/doc/Rnews/ (http://CRAN.R-project.org/doc/Rnews/).

R Core Team. 2013. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. http://www.R-project.org/ (http://www.R-project.org/).

RStudio Team. 2012. *RStudio: Integrated Development Environment for R*. Boston, MA: RStudio, Inc. http://www.rstudio.com/ (http://www.rstudio.com/).

RStudio, and Inc. 2014. *rmarkdown: R Markdown Document Conversion*.

Unkrig, Arno, and Martin Bayer. 2004. “html2text.” http://manpages.ubuntu.com/manpages/hardy/man1/html2text.1.html.

Author dieser Seminararbeit Richard Rentrop (https://plus.google.com/114282442713554874087?rel=author)