Text Mining of Social Media Content

Johan Blomme

april 2018

Johan Blomme
Leenstraat 11
8340 Damme-Sijsele
URL : www.johanblomme.net
Email : j.blomme@telenet.be

Text Mining of Social Media Content

Twitter is one of the most popular social networks through which millions of users share information and express views and opinions. The rapid growth of internet data is a driver for mining the huge amount of unstructured data that is generated to uncover insights from it.

In this study we explore different text mining tools. We collect tweets containing the "#MachineLearning" hashtag, prepare the data and run a series of diagnostics to mine the text that is contained in tweets. We also examine the issue of topic modeling that allows to estimate the similarity between documents in a larger corpus.

The analysis we present is not based on a theoretical framework. The main purpose is to explore a variety of tools to derive insights from data. The data to reproduce the analysis is available on my github.

The first step in the R session is to load all the necessary libraries we need for the analysis.

```
library(tidyverse)
library(tidyr)
library(tidytext)
library(lubridate)
library(scales)
library(readr)
library(stringr)
library(stringi)
library(ggplot2)
library(tm)
library(SnowballC)
library(twitteR)
library(igraph)
library(ggraph)
library(topicmodels)
library(reshape2)
library(igraph)
library(ggthemes)
library(FactoMineR)
library(factoextra)
library(cluster)
library(RColorBrewer)
library(ggrepel)
library(ape)
library(tsne)
library(Rtsne)
library(fpc)
library(wordcloud)
library(wordcloud2)
library(slam)
library(Rmpfr)
library(rgl)
library(ggalt)
library(widyr)
```

1. TWITTER AUTHENTICATION AND TWITTER DATA IMPORT

To extract data from Twitter, we need to create and register an app on twitter developers website for authentification[1]. To authorize our app to access Twitter, we need to use the OAuth interface. After setting up a Twitter authentification, we use the searchTwitter() function to extract tweets of hashtag "MachineLearning" .

```
api_key <- "insert key"
api_secret <- "insert key"
access_token <- "insert key"
access_token_secret <- "insert key"
setup_twitter_oauth(api_key,api_secret,access_token,access_token_secret)

ml <- searchTwitter("#MachineLearning",n=5000,lang="en")
# convert to a data frame
ml <- twListToDF(ml)
setwd()
save(ml,file="ml.Rdata")
write.csv(ml,"ml.csv")
```

2. TEXT CLEANING AND CORPUS CREATION

We import the data and perform some text cleaning to prepare the tweet documents for further processing. This involves the substitution of unnecessary characters, stripping whitespace, lowering text, removing stopwords and numbers. Then we define the tweets as a corpus that is repeatedly transformed by the clean.corpus() function[2].

```
text.df <- read.csv("ml.csv")
tweets <- data.frame(text=text.df$text)
tweets$text <- as.character(tweets$text)
tweets$text <- gsub('http\\S+\\s*', '', tweets$text)
tweets$text <- gsub('\\b+RT', '', tweets$text)
tweets$text <- gsub('#\\S+', '', tweets$text)
tweets$text <- gsub('@\\S+', '', tweets$text)
tweets$text <- gsub('[[:cntrl:]]', '', tweets$text)
tweets$text <- gsub("\\d", '', tweets$text)
tryTolower <- function(x){
 # return NA when there is an error
 y=NA
 # tryCatch error
 try_error= tryCatch(tolower(x),error=function(e) e)
 # if not an error
 if (!inherits(try_error, 'error'))
  y=tolower(x)
```

```
  return(y)
}
custom.stopwords <- c(stopwords('english'),'amp','machine','learning')
clean.corpus <- function(corpus) {
  corpus <- tm_map(corpus,
                content_transformer(tryTolower))
  corpus <- tm_map(corpus,removeWords,
                custom.stopwords)
  corpus <- tm_map(corpus,removePunctuation)
  corpus <- tm_map(corpus,stripWhitespace)
  corpus <- tm_map(corpus,removeNumbers)
  return(corpus)
}
corpus <- Corpus(VectorSource(tweets$text))
corpus <- clean.corpus(corpus)
```

We use a "bag of words" type of text mining, which means that the tokenization of the corpus takes place by splitting the text into smaller units (tokens), in this case words. In the bag of words approach to text mining, every word (or a group of words, n-grams) is treated as a unique feature of the document. Bag of words analysis can be done quickly and provides us with a term document matrix (tdm). A term document matrix is a matrix in which the terms (words) represent the rows and the documents (tweets) represent the columns.

```
tdm <- TermDocumentMatrix(corpus)
tdm
```

```
> tdm
<<TermDocumentMatrix (terms: 3341, documents: 5000)>>
Non-/sparse entries: 25862/16679138
Sparsity           : 100%
Maximal term length: 36
Weighting          : term frequency (tf)
```

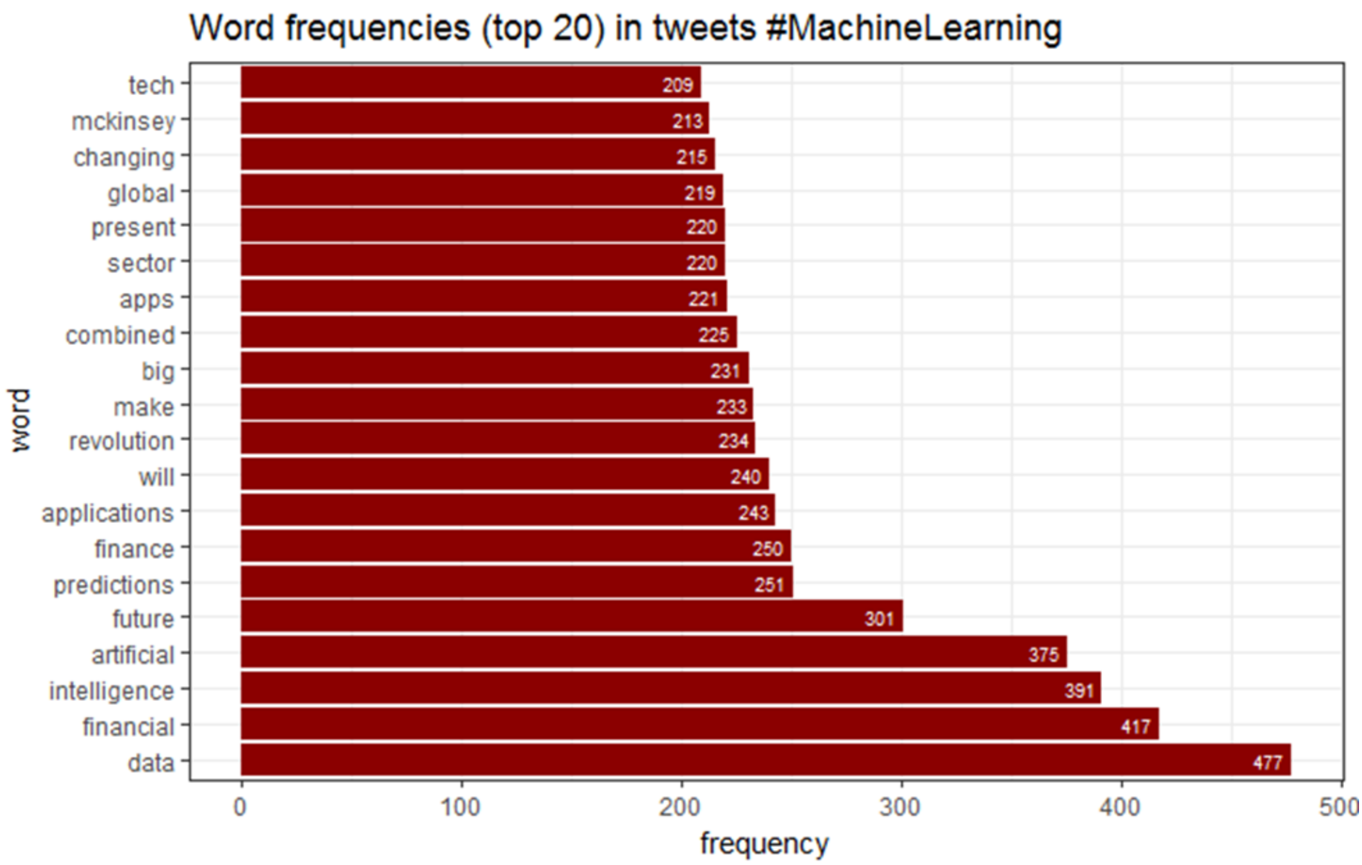
```
m <- as.matrix(tdm)
```

After the text data is cleaned and tokenized we can proceed with different text mining operations.

3. TEXT MINING

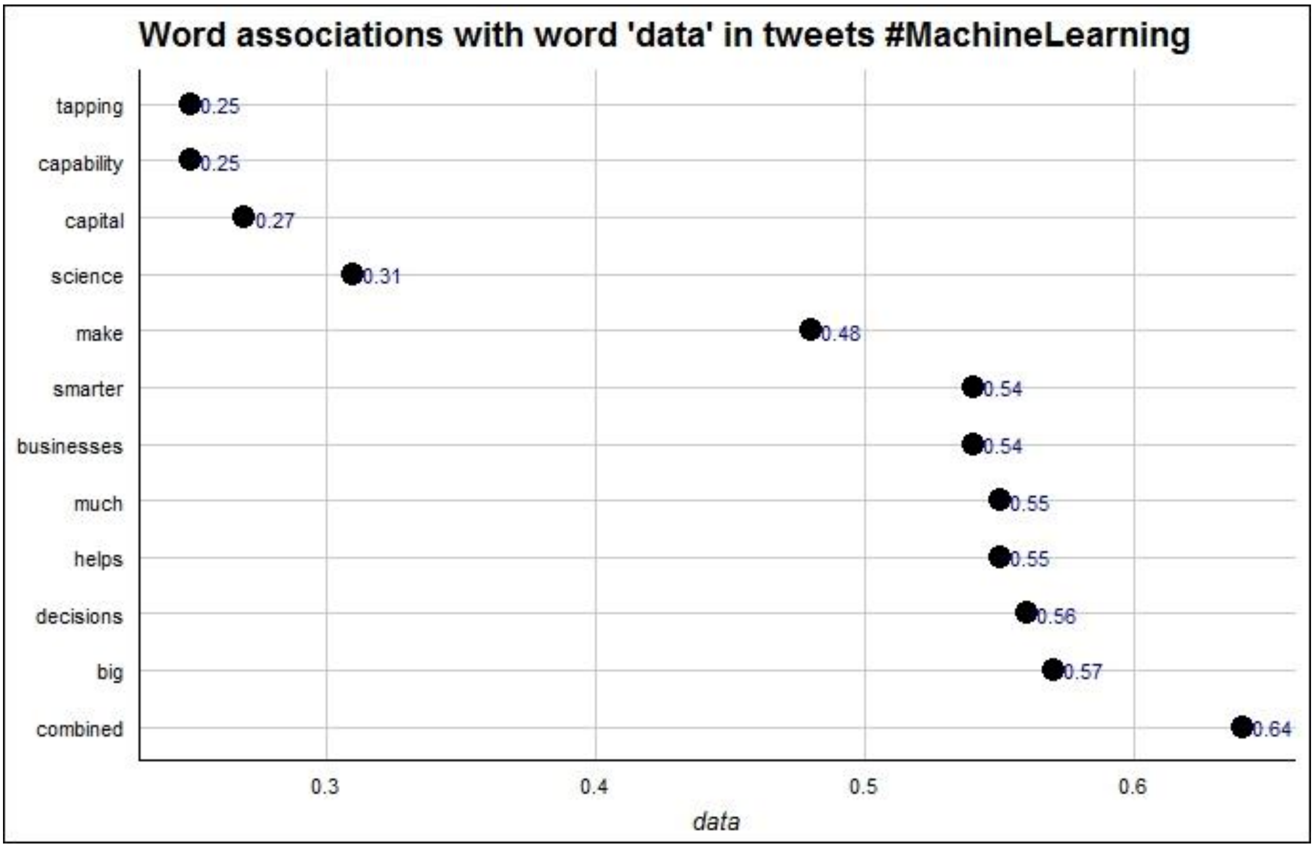
3.1. Word frequencies

```
term.freq <- rowSums(m)
freq.df <- data.frame(word=names(term.freq),frequency=term.freq)
freq.df <- freq.df[order(freq.df[,2],decreasing=T),]
freq.df$word <- factor(freq.df$word,levels=unique(as.character(freq.df$word)))
ggplot(freq.df[1:20,],aes(x=word,y=frequency)) +
  geom_bar(stat="identity",fill="darkred") +
  coord_flip() +
  theme_gdocs() +
  geom_text(aes(label=frequency),colour="white",hjust=1.25,size=2.5) +
  ggtitle("Word frequencies (top 20) in tweets #MachineLearning") +
  theme(plot.title = element_text(size = 10, face = "bold")) +
  theme_bw()
```



3.2. Word associations

```
associations <- findAssocs(tdm, "data", 0.25)
associations <- as.data.frame(associations)
associations$terms <- row.names(associations)
associations$terms <- factor(associations$terms,levels=associations$terms)
ggplot(associations,aes(y=terms)) +
  geom_point(aes(x=data),data=associations,size=4) +
  theme_gdocs() +
  geom_text(aes(x=data,label=data),colour="darkblue",hjust=-0.25,size=6) +
  theme(text=element_text(size=15),axis.title.y=element_blank())
```



3.3. Visualizing term relationships

3.3.1. Dimension reduction with t-sne

The matrix version of the term document matrix we created earlier (m) consists of 3341 terms and 5000 documents. The output below shows useful statistics, such as the sparsity of the matrix. This value reveals the level of emptiness or zero frequencies in the term document matrix. The matrix is very sparse and the majority of entries are zero.

```
> tdm
<<TermDocumentMatrix (terms: 3341, documents: 5000)>>
Non-/sparse entries: 25862/16679138
Sparsity           : 100%
Maximal term length: 36
Weighting          : term frequency (tf)
```

Another method to inspect the term document matrix is to find the words that occur frequently, e.g. 100 times or more.

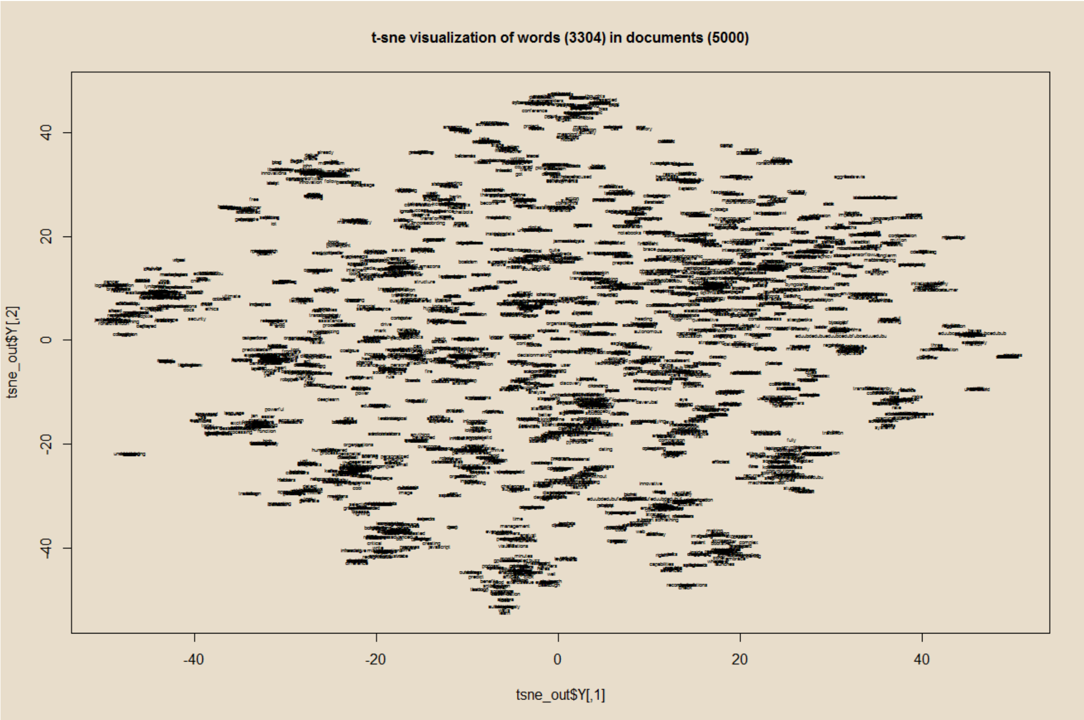
findFreqTerms(tdm,100)

```
> findFreqTerms(tdm,100)
 [1] "industry"      "using"         "tech"          "trends"        "approval"
 [6] "bleeds"        "brain"         "fda"           "gets"          "medymatch"
[11] "nod"           "pinpointing"   "quick"         "artificial"    "data"
[16] "finance"       "apps"          "year"          "changing"      "face"
[21] "financial"     "sector"        "will"          "intelligence"  "overview"
[26] "segment"       "driven"        "revolution"    "technological" "integration"
[31] "marketing"     "predictions"   "via"           "present"       "global"
[36] "institute"     "mckinsey"      "primer"        "servicessource" "future"
[41] "big"           "businesses"    "combined"      "decisions"     "helps"
[46] "make"          "much"          "smarter"       "can"           "deep"
[51] "new"           "applications"  "supervised"    "unsupervised"  "reinforcement"
[56] "ready"         "blocks"        "differences"   "illustrate"    "lego"
[61] "styles"
```

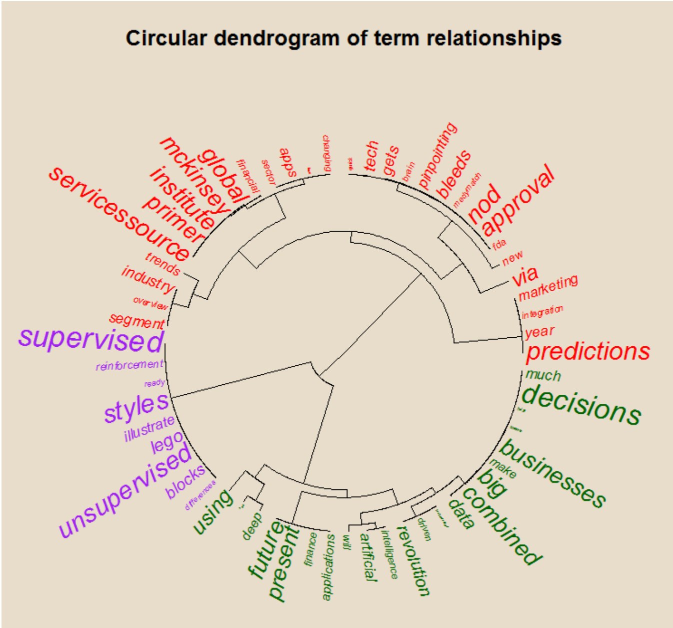
In order to make a visualization of token frequency relationships we used t-distributed stochastic neighbor embedding (t-sne)[3]. This technique visualizes high-dimensional data by giving each data point (in this case a word) a location in a two-dimensional space. In this way, t-sne is able to cluster similarities.

```
tsne_out <- Rtsne(m,dims=2,check_duplicates=F)
str(tsne_out)
plot(tsne_out$Y,t="n",main="t-sne visualization of words (3304) in documents (5000)",cex.main=1)
text(tsne_out$Y,labels=rownames(m),cex=0.35)
```

From the plot on the following page, we can see that (to a certain extent) words are grouped by t-sne. To further explore the relationships between words, we created a data frame from the two-dimensional coordinates and limited the analysis to frequently occurring words (>= 100). We then converted the matrix into a distance matrix and visualize the relationships between words in a circular dendrogram.

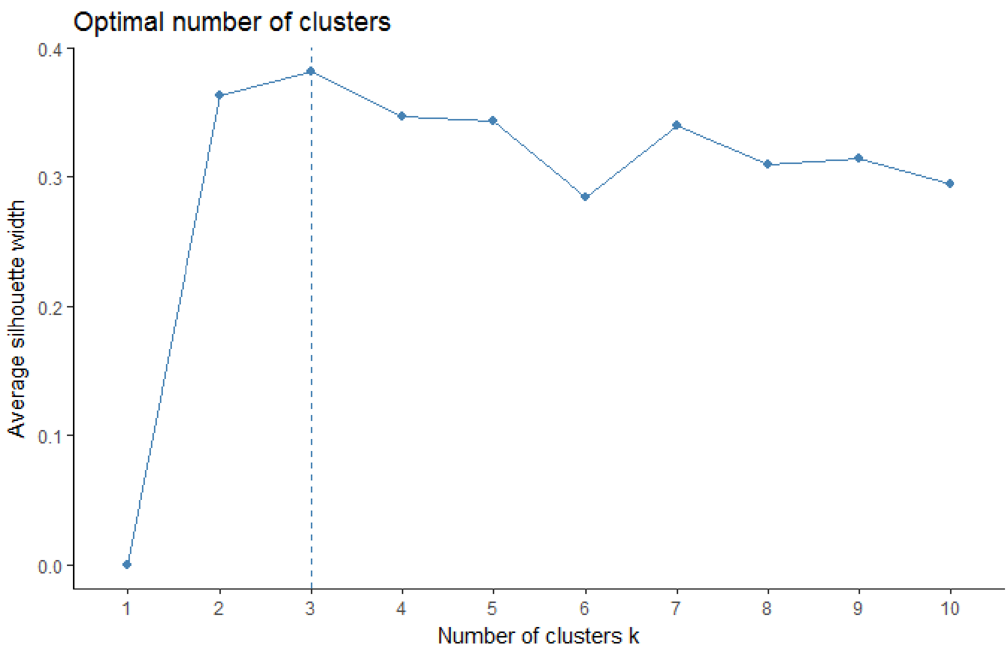



```
X1 <- tsne_out$Y[,1]
X2 <- tsne_out$Y[,2]
Fq <- rowSums(m,na.rm=TRUE,dims=1)
Rtsne <- data.frame(Fq,X1,X2)
str(Rtsne)
median(Rtsne$Fq)
head(Rtsne)
Rtsne2 <- subset(Rtsne,subset=Fq >= 100,select=c(X1,X2))
m2 <- as.matrix(Rtsne2)
distMatrix <- dist(scale(m2))
fit <- hclust(distMatrix,method="ward.D2")
p <- plot(fit)
rect.hclust(fit,k=3)
# vector of colors
mypal = c("red","darkgreen","purple")
# cut tree in 3 clusters
clus = cutree(fit, 3)
# plot
op = par(bg = "#E8DDCB")
# size reflects frequency
plot(as.phylo(fit), type = "fan", tip.color = mypal[clus], label.offset = 0.1, cex = log(Rtsne$Fq)/3,
    main="Circular dendrogram of term relationships",cex.main=1.5)
```

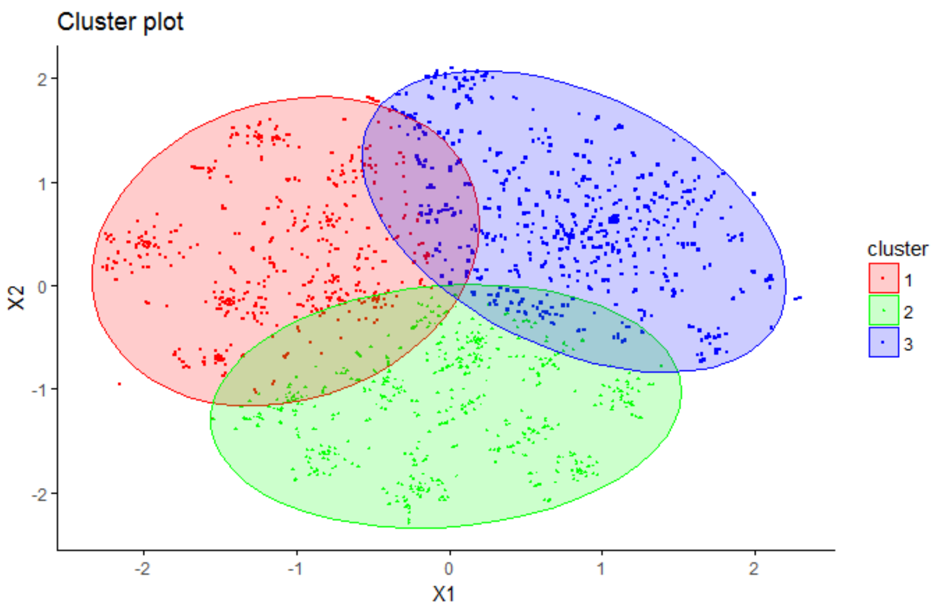


3.3.2. Cluster analysis

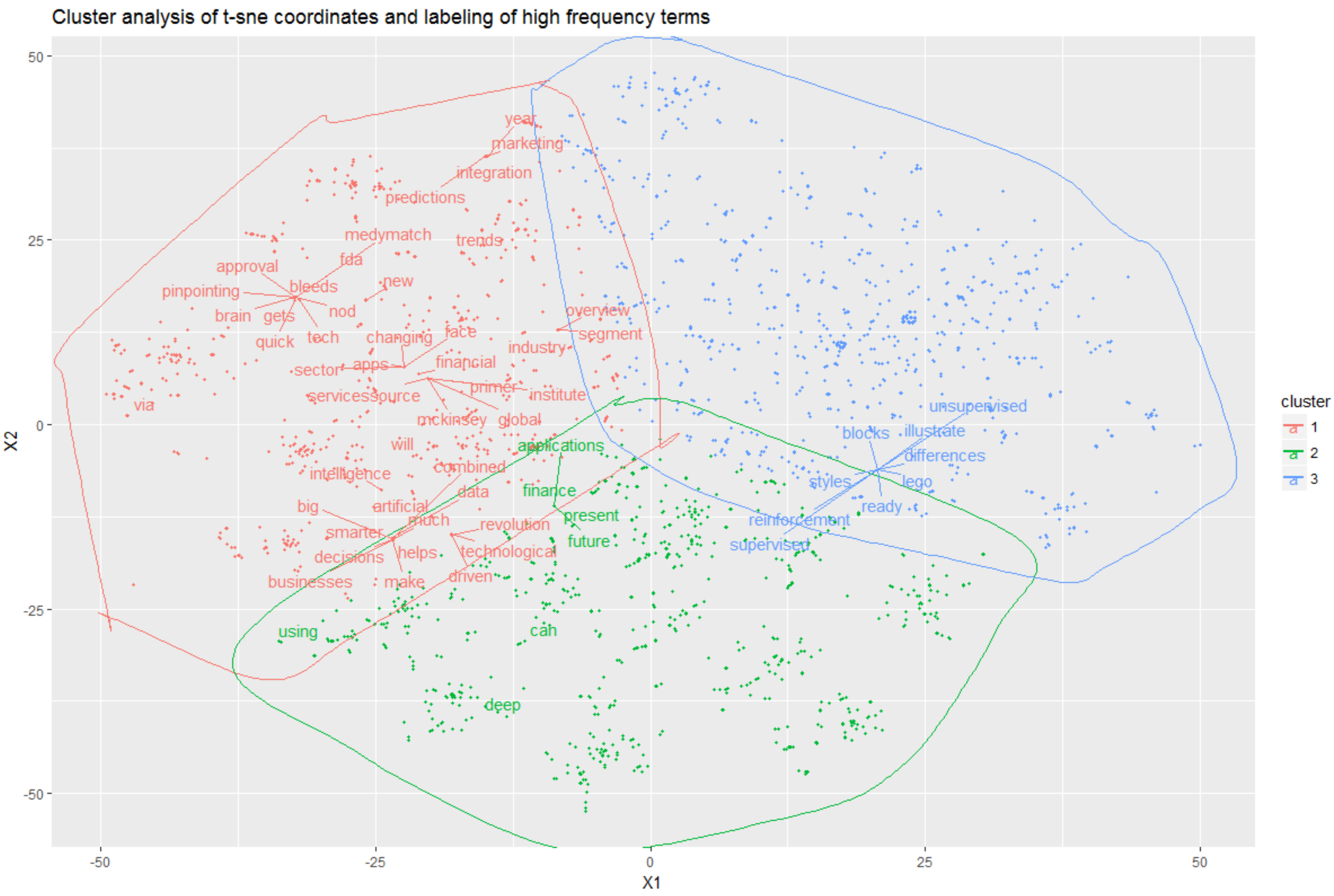
```
# clara clustering4
df <- data.frame(X1,X2)
fviz_nbclust(df,clara,method="silhouette") + theme_classic()
```



```
clara.res <- clara(df,3,samples=50,pamLike=TRUE)
fviz_cluster(clara.res,
        palette = c("red","green","blue"), # color palette
        ellipse.type = "t", # Concentration ellipse
        geom = "point", pointsize = 0.5,
        ggtheme = theme_classic())
```




```
clara.res2 <- cbind(clara.res$data,clara.res$clustering)
label <- rownames(m)
D <- cbind(clara.res2,label,Rtsne$Fq)
DX2 <- as.data.frame(D)
DX2$X1 <- as.numeric(as.character(DX2$X1))
DX2$X2 <- as.numeric(as.character(DX2$X2))
DX2$V5 <- as.numeric(as.character(DX2$V5))
names(DX2) <- c("X1","X2","cluster","label","frequency")
# select terms with frequency >= 100
DX3 <- subset(DX2,frequency >= 100)
P  <-  ggplot(DX2,aes(x=X1,y=X2,color=cluster)) + geom_point(size=0.8) + geom_encircle()
q  <-  p + geom_text_repel(data=DX3,aes(x=X1,y=X2,label=label,colour=cluster),size=4) +
  ggtitle("Cluster analysis of t-sne coordinates and labeling of high frequency terms")
q
```



3.4. Analyse text with the tidy text format

3.4.1. Create tidy text data from a document term matrix

Starting from a document term matrix, it is easy to convert this data with the tidytext package[5] into a one term per document per row data frame. This is done with the tidy() function.

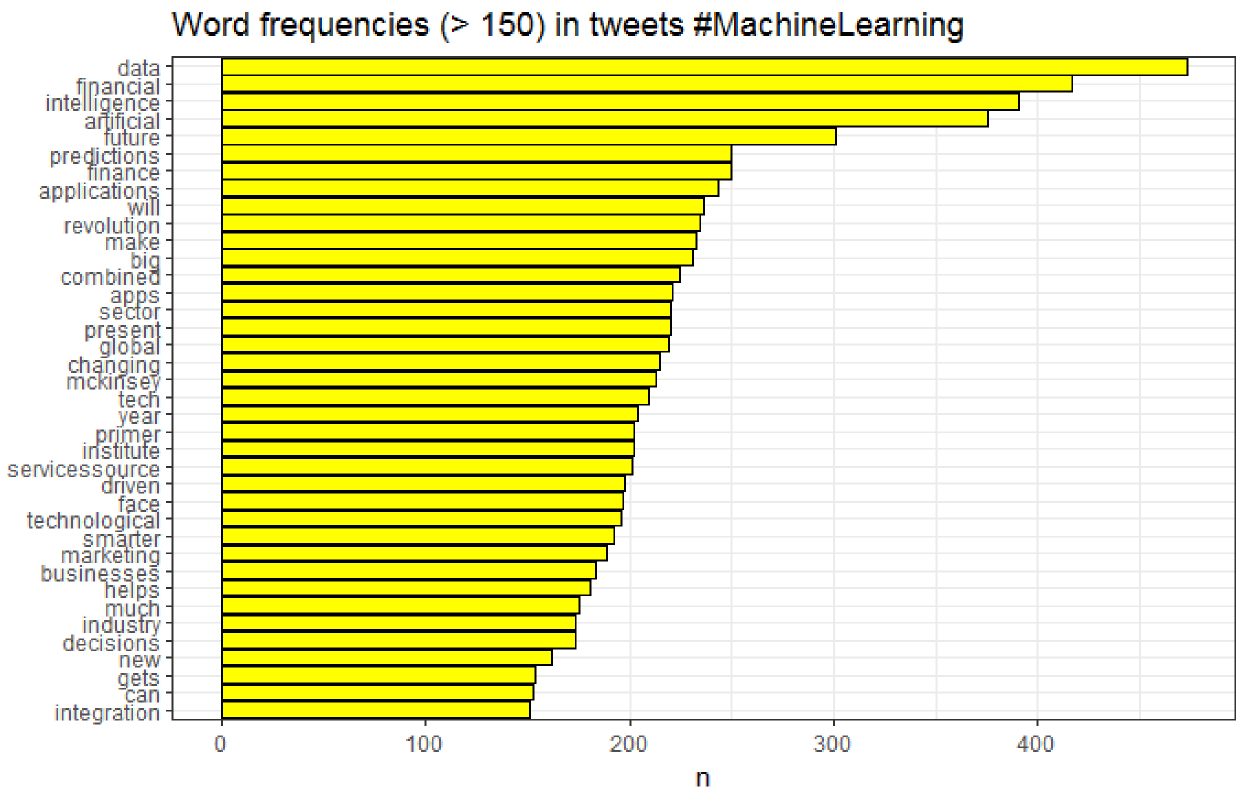
```
dtm <- DocumentTermMatrix(corpus)
str(dtm)
```

```
> str(dtm)
List of 6
 $ i       : int [1:25862] 1 1 1 1 1 1 2 2 2 2 ...
 $ j       : int [1:25862] 1 2 3 4 5 6 7 8 9 10 ...
 $ v       : num [1:25862] 1 1 1 1 1 1 1 1 1 1 ...
 $ nrow    : int 5000
 $ ncol    : int 3341
 $ dimnames:List of 2
  ..$ Docs : chr [1:5000] "1" "2" "3" "4" ...
  ..$ Terms: chr [1:3341] "aspects" "business" "disrupt" "embr" ...
 - attr(*, "class")= chr [1:2] "DocumentTermMatrix" "simple_triplet_matrix"
 - attr(*, "weighting")= chr [1:2] "term frequency" "tf"
```

tweets_td <- tidy(dtm)
tweets_td

```
> tweets_td
# A tibble: 25,862 x 3
   document         term count
      <chr>        <chr> <dbl>
 1        1      aspects     1
 2        1     business     1
 3        1      disrupt     1
 4        1         embr     1
 5        1     industry     1
 6        1 technologies     1
 7        2       automl     1
 8        2     generate     1
 9        2    pipelines     1
10        2         tpot     1
# ... with 25,852 more rows
```

```
tweets_td %>%
  count(term, sort = TRUE) %>%
  filter(n > 150) %>%
  mutate(word = reorder(term, n)) %>%
  ggplot(aes(word, n)) +
  geom_col(fill="yellow",col="black") +
  xlab(NULL) +
  coord_flip() +
  ggtitle("Word frequencies (> 150) in tweets #MachineLearning") +
  theme(plot.title = element_text(size = 10, face = "bold")) +  theme_bw()
```



3.4.2. Wordcloud and sentiments

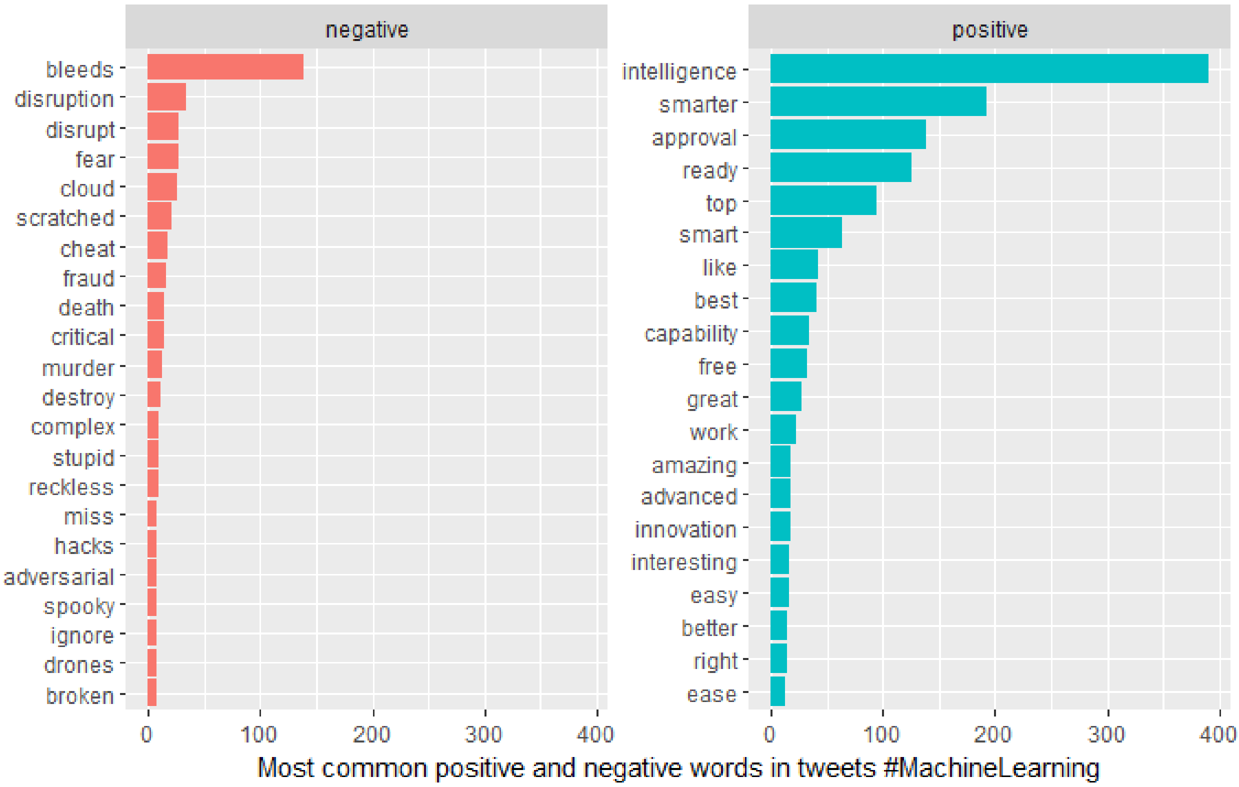
```
tweets_words <- tweets_td %>%
  count(term,sort=TRUE) %>%
  filter(n >= 50)
tweets_words
wordcloud2(data=tweets_words,size=0.4)
```

The tidytext package contains three sentiment lexicons. These three lexicons are based on single words (unigrams). The Bing lexicon categorizes words into positive or negative. The NRC lexicon categorizes words in a binary way into categories of positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise and trust. The AFINN lexicon assigns words a score that runs between -5 and 5, with negative scores indicating negative sentiment and positive scores indicating positive sentiment.

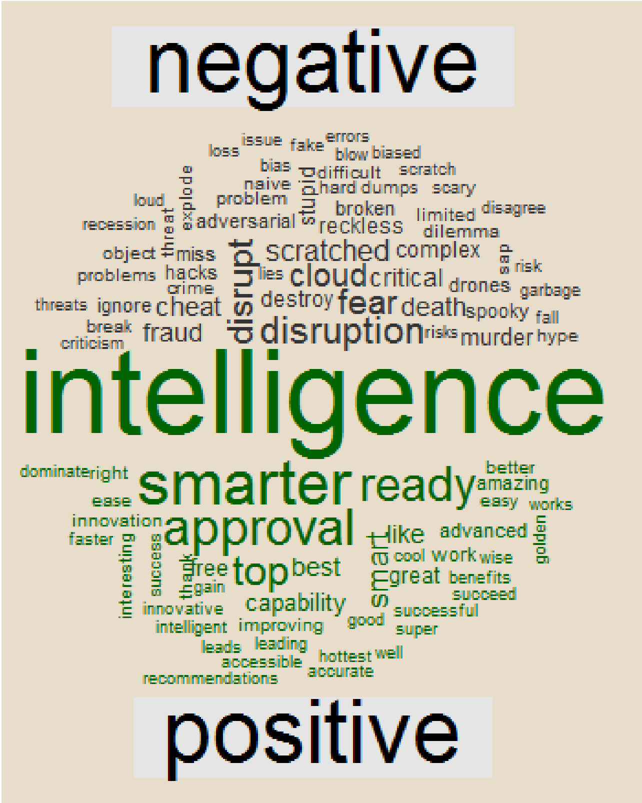
```
tweets_sentiment <- tweets_td %>%
  inner_join(get_sentiments("bing"),by=c(term="word"))
tweets_sentiment
```

```
> tweets_sentiment
# A tibble: 2,455 x 4
   document         term count sentiment
      <chr>        <chr> <dbl>     <chr>
 1        1      disrupt     1  negative
 2        5          top     1  positive
 3        6     approval     1  positive
 4        6       bleeds     1  negative
 5        8        cheat     1  negative
 6        9         work     1  positive
 7       10         ease     1  positive
 8       11       better     2  positive
 9       11       faster     1  positive
10       20 intelligence     1  positive
# ... with 2,445 more rows
```

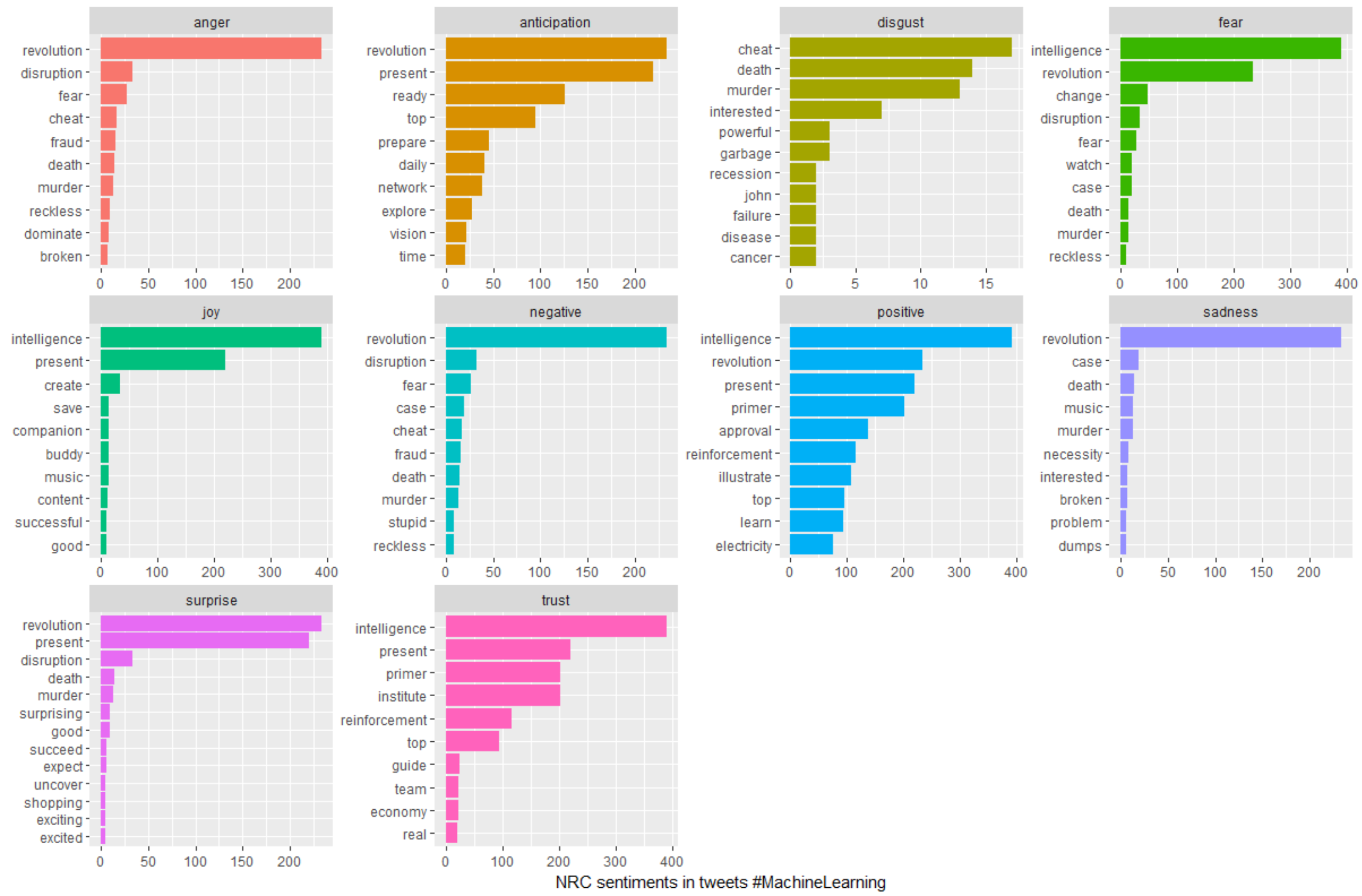
```
pos_neg %>%
  group_by (sentiment) %>%
  top_n(20) %>%
  ungroup() %>%
  mutate(term=reorder(term,n)) %>%
  ggplot(aes(term,n,fill=sentiment)) +
  geom_col(show.legend=FALSE) +
  facet_wrap(~ sentiment, scales="free_y") +
  labs(x=NULL,y="Most common positive and negative words in tweets #MachineLearning") +
  coord_flip()
```




```
# comparison workcloud positive/negative sentiments
pos_neg %>%
  acast(term ~ sentiment, value.var="n",fill= 0) %>%
  comparison.cloud(colors=c("gray20","darkgreen"),max.words=100)
```



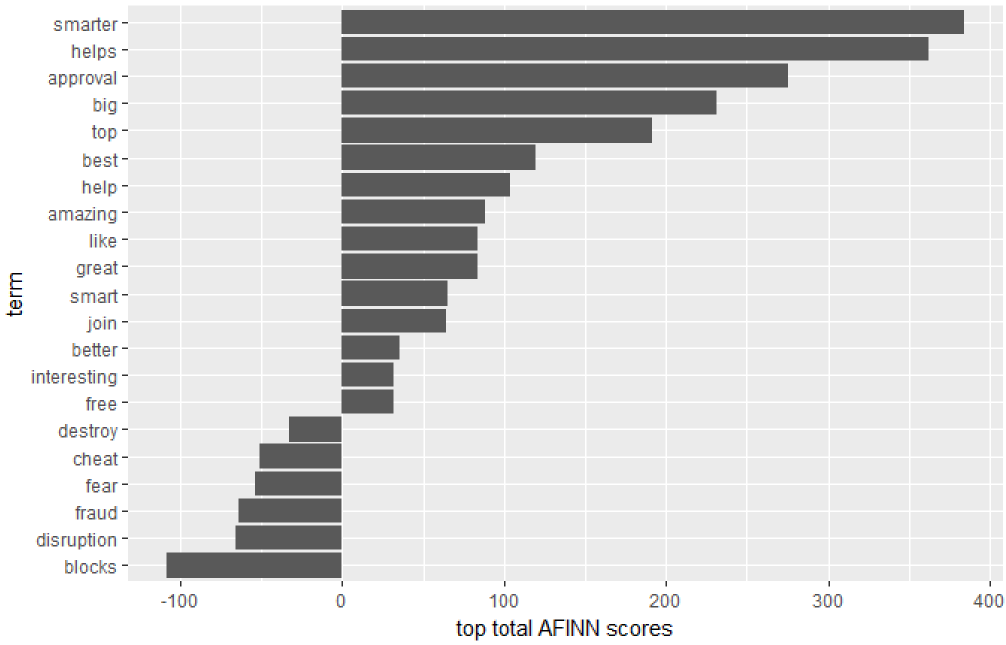

```
nrc_sentiments <- tweets_td %>%
  inner_join(get_sentiments("nrc"),by=c(term="word")) %>%
  count(term,sentiment) %>%
  group_by(sentiment) %>%
  top_n(10) %>%
  ungroup() %>%
  mutate(term=reorder(term,n)) %>%
  ggplot(aes(term,y=n,fill=sentiment)) +
  geom_col(show.legend=FALSE) +
  facet_wrap(~ sentiment, scales="free") +
  labs(x=NULL,y="NRC sentiments in tweets #MachineLearning") +
  coord_flip()
nrc_sentiments
```



# AFINN sentiment
afinn_sentiment <- tweets_td %>%
 inner_join(get_sentiments("afinn"),by=c(term="word"))
afinn_sentiment

```
> afinn_sentiment
# A tibble: 2,253 x 4
   document      term count score
      <chr>     <chr> <dbl> <int>
 1        3     reach     1     1
 2        4       cut     1    -1
 3        5       top     1     2
 4        6  approval     1     2
 5        8     cheat     1    -3
 6       10      ease     1     2
 7       11    better     2     2
 8       11  solution     1     1
 9       17 solutions     1     1
10       24    vision     1     1
# ... with 2,243 more rows
```

```
# calculate AFINN scores for terms
afinn_sentiment %>%
  group_by(term) %>%
  summarize(contribution=sum(count * score)) %>%
  top_n(20,abs(contribution)) %>%
  mutate(term=reorder(term,contribution)) %>%
  ggplot(aes(term,contribution)) +
  geom_col() +
  coord_flip() +
  labs(y="top total AFINN scores")
```



3.4.3. Term frequency

To examine term frequency, we first calculate the frequency of each token. Then we calculate the total number of words in all documents and perform a join of both.

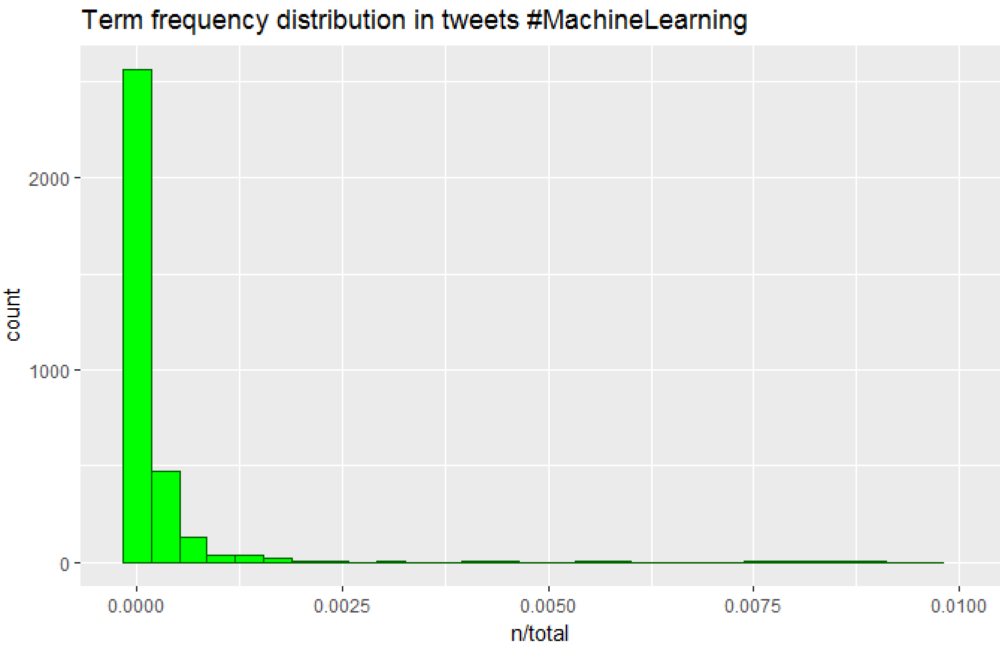
```
terms <- tweets_td %>%
  count(term,sort=T)
terms
```

```
> terms
# A tibble: 3,341 x 2
          term     n
         <chr> <int>
 1        data   473
 2   financial   417
 3 intelligence  391
 4  artificial   375
 5      future   301
 6     finance   250
 7  predictions  250
 8 applications  243
 9        will   236
10  revolution   234
# ... with 3,331 more rows
```

```
terms$var <- 1
totalterms <- terms %>%
  group_by(var) %>%
  summarize(total=sum(n))
terms_total <- left_join(terms,totalterms)
terms_total
```

```
> terms_total
# A tibble: 3,341 x 4
          term     n   var total
         <chr> <int> <dbl> <int>
 1        data   473     1 25862
 2   financial   417     1 25862
 3 intelligence  391     1 25862
 4  artificial   375     1 25862
 5      future   301     1 25862
 6     finance   250     1 25862
 7  predictions  250     1 25862
 8 applications  243     1 25862
 9        will   236     1 25862
10  revolution   234     1 25862
# ... with 3,331 more rows
```

```
ggplot(terms_total,aes(n/total)) +
  geom_histogram(fill="green",color="darkgreen",show.legend=FALSE) +
  xlim(NA,0.010) +
  ggtitle("Term frequency distribution in tweets #MachineLearning")
```

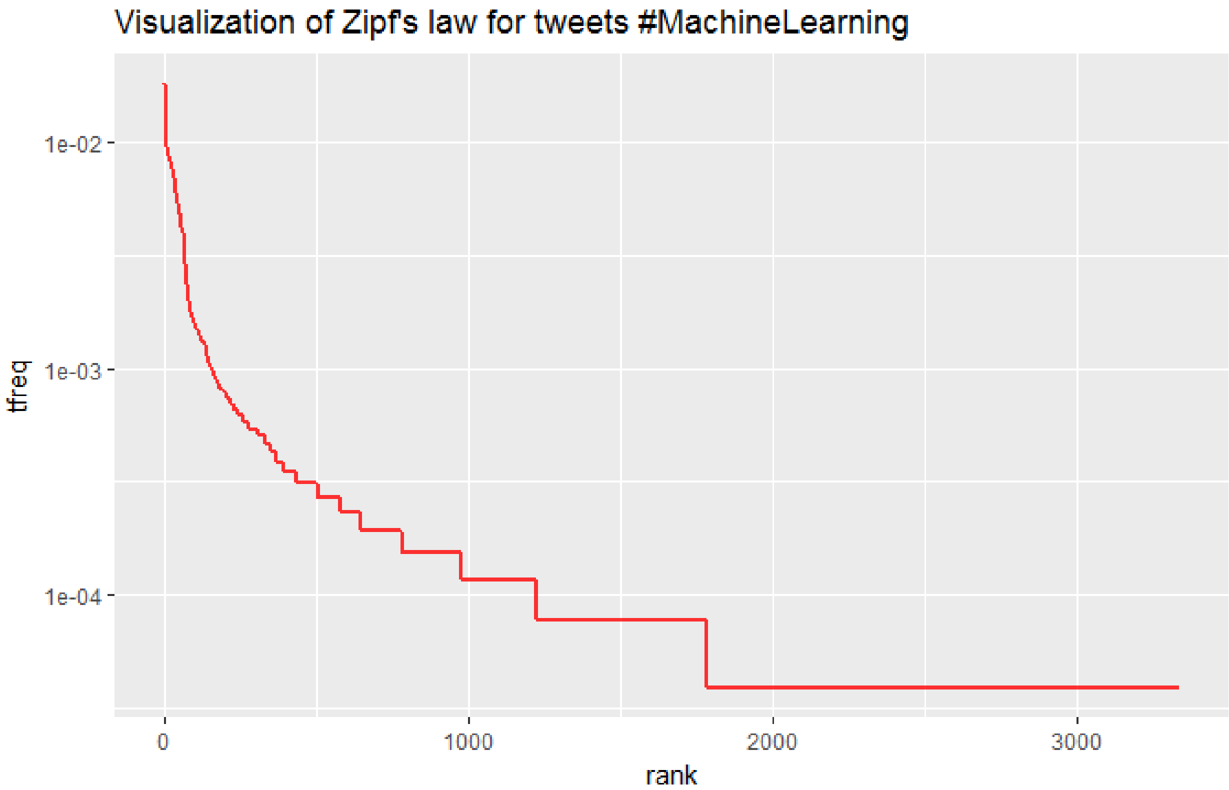


The above long-tail distribution is typical for the analysis of documents. The relationship between a term's frequency and its rank is called Zipf's law.

```
freq_by_rank <- terms_total %>%
  mutate(rank=row_number()) %>%
  mutate(tfreq=n/total)
freq_by_rank
```

```
> freq_by_rank
# A tibble: 3,341 x 6
          term     n   var total  rank       tfreq
         <chr> <int> <dbl> <int> <int>       <dbl>
 1        data   473     1 25862     1 0.018289382
 2   financial   417     1 25862     2 0.016124043
 3 intelligence  391     1 25862     3 0.015118707
 4  artificial   375     1 25862     4 0.014500039
 5      future   301     1 25862     5 0.011638698
 6     finance   250     1 25862     6 0.009666692
 7  predictions  250     1 25862     7 0.009666692
 8 applications  243     1 25862     8 0.009396025
 9        will   236     1 25862     9 0.009125358
10  revolution   234     1 25862    10 0.009048024
# ... with 3,331 more rows
```

```
p <- ggplot(freq_by_rank,aes(x=rank,y=tfreq)) +
  geom_line(size=1,col="red",alpha=.8,show.legend=FALSE) +
  scale_y_log10() +
  ggtitle("Visualization of Zipf's law for tweets #MachineLearning")
p
```



3.4.4. Term frequency – inverse document frequency

To quantify each document by giving weights to each of the terms inside the document, the concept "term frequency-inverse document frequency" (tf-idf) is used. The calculated tf-idf indicates the importance of each term to a document it belongs to in a context of the whole corpus. How many times a given word appears in a document it belongs to is the term frequency (tf) part of tf-idf. The higher the tf-value of a given term to a document the more important the term is to the document. But if a term appears in many documents of the corpus then it is not really important for a particular document. For example, if the term "applications" appears in a majority of all the documents then it not really important for a particular document. Therefore, we need a weighting system that would decrease the importance of a given term when the number of documents in which the term appears increases. This is the idf-part of the tf-idf.

Thus, tf-idf is a weighting system that quantifies the importance of each term to each document by increasing the importance based on the term frequency and decreasing the importance based on the document frequency.

3.4.4.1. Using tf-idf to cluster documents

We will explore the use of tf-idf to cluster tweet documents. First, a document term matrix is created weighted by term frequency-inverse document frequency.

dtm_tfidf <-DocumentTermMatrix(corpus,control = list(weighting = function(x) weightTfIdf(x, normalize = FALSE)))
str(dtm_tfidf)

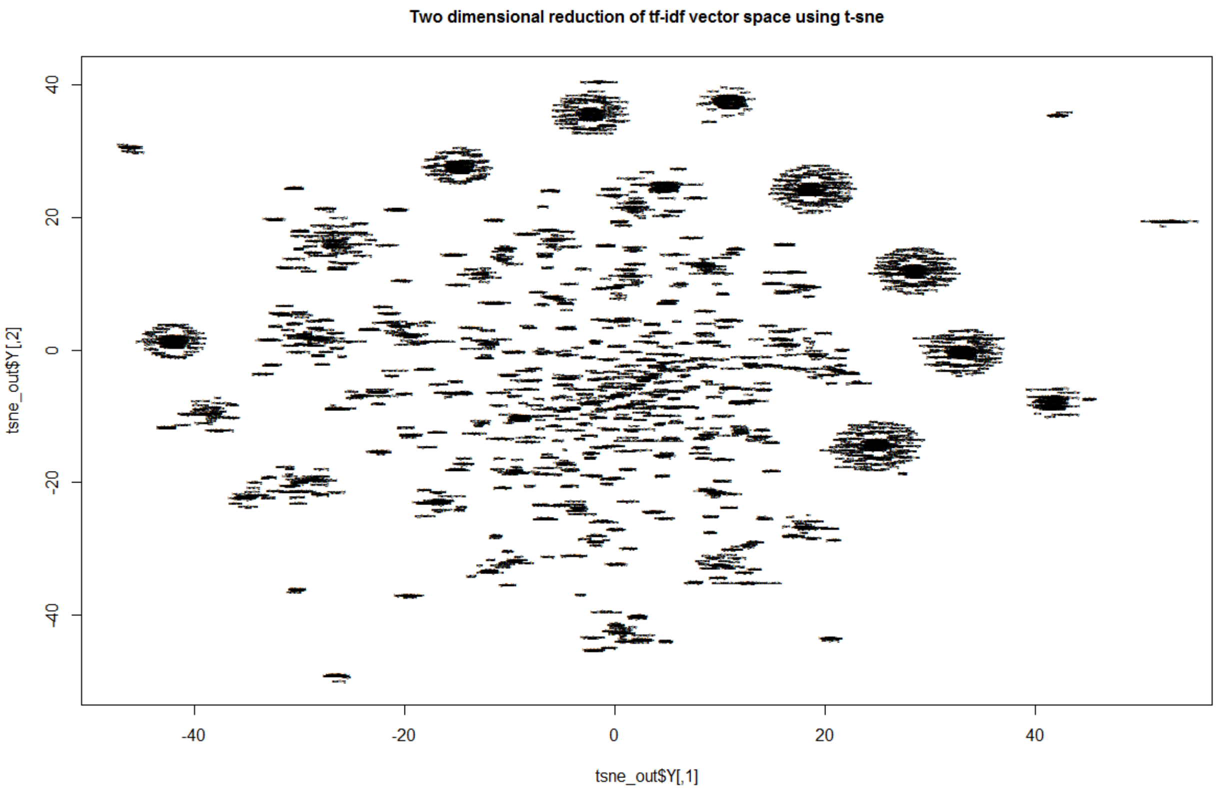
```
> str(dtm_tfidf)
List of 6
 $ i       : int [1:25862] 1 1 1 1 1 1 2 2 2 2 ...
 $ j       : int [1:25862] 1 2 3 4 5 6 7 8 9 10 ...
 $ v       : num [1:25862] 7.64 5.81 7.48 8.12 4.85 ...
 $ nrow    : int 5000
 $ ncol    : int 3341
 $ dimnames:List of 2
  ..$ Docs : chr [1:5000] "1" "2" "3" "4" ...
  ..$ Terms: chr [1:3341] "aspects" "business" "disrupt" "embr" ...
 - attr(*, "class")= chr [1:2] "DocumentTermMatrix" "simple_triplet_matrix"
 - attr(*, "weighting")= chr [1:2] "term frequency - inverse document frequency" "tf-idf"
```

inspect(terms)

```
> inspect(dtm_tfidf)
<<DocumentTermMatrix (documents: 5000, terms: 3341)>>
Non-/sparse entries: 25862/16679138
Sparsity           : 100%
Maximal term length: 36
Weighting          : term frequency - inverse document frequency (tf-idf)
Sample             :
      Terms
Docs   applications artificial     data finance financial future intelligence predictions revolution
  1124            0          0 0.000000       0         0      0            0           0          0
  1167            0          0 0.000000       0         0      0            0           0          0
  1676            0          0 0.000000       0         0      0            0           0          0
  2227            0          0 0.000000       0         0      0            0           0          0
  3034            0          0 3.402016       0         0      0            0           0          0
  3936            0          0 0.000000       0         0      0            0           0          0
  4241            0          0 0.000000       0         0      0            0           0          0
  4465            0          0 0.000000       0         0      0            0           0          0
  4821            0          0 0.000000       0         0      0            0           0          0
  734             0          0 0.000000       0         0      0            0           0          0
      Terms
Docs       will
  1124 0.000000
  1167 0.000000
  1676 0.000000
  2227 0.000000
  3034 0.000000
  3936 0.000000
  4241 0.000000
  4465 0.000000
  4821 0.000000
```

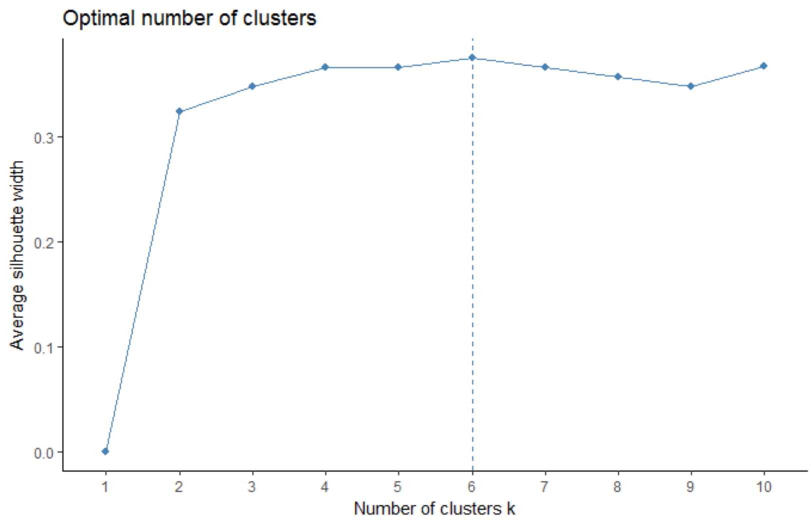
We converted the document term matrix into a matrix and as with the analysis of term frequencies (see . pages 7-10) we took the tf-idf vectors and performed a t-sne analysis to get a 2D-corpus visualization. From the plot we can see that t-sne is effectively able to cluster similar documents.

```
m2 <- as.matrix(dtm_tfidf)
tsne_out <- Rtsne(m2,dims=2,check_duplicates=F)
plot(tsne_out$Y,t="n",main="Two dimensional reduction of tf-idf vector space using t-sne",cex.main=1)
text(tsne_out$Y,labels=rownames(m),cex=0.25)
```

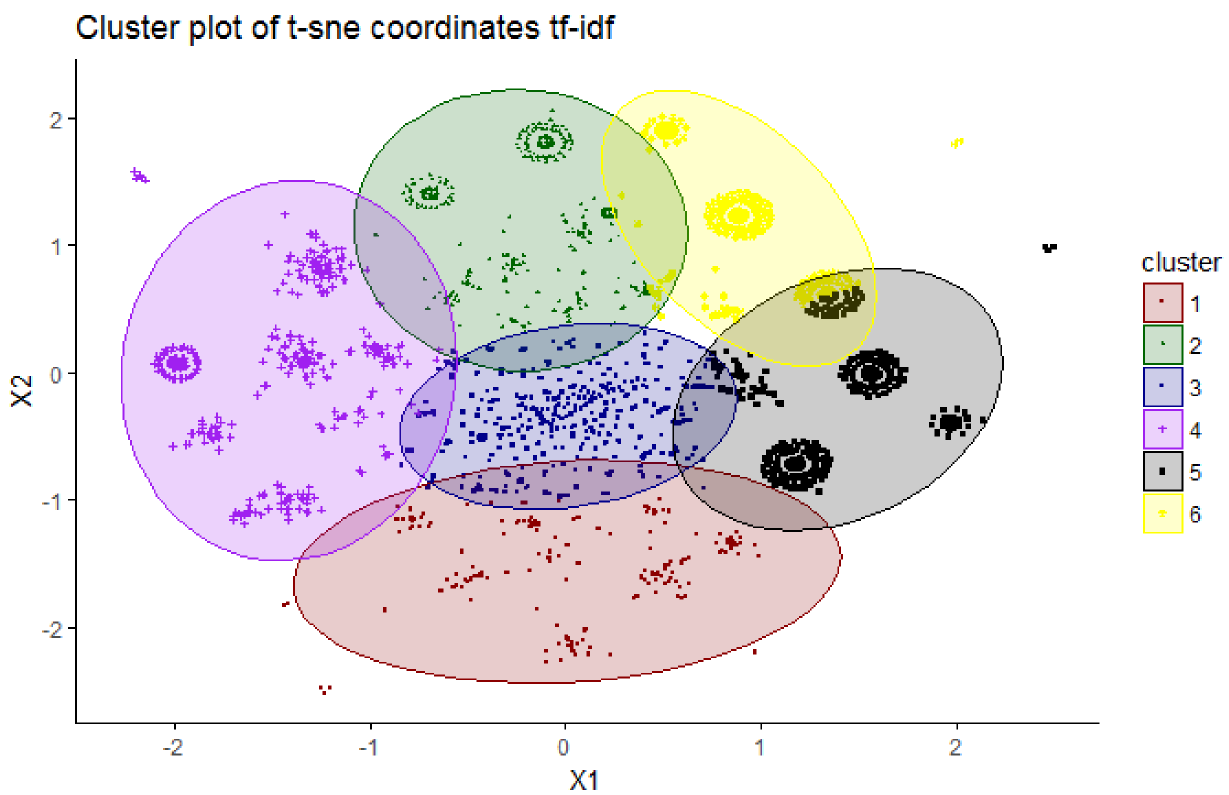


The t-sne coordinates are then fed to CLARA clustering, which resulted in the extraction of 6 clusters.

```
X1 <- tsne_out$Y[,1]
X2 <- tsne_out$Y[,2]
# clara clustering
df <- data.frame(X1,X2)
fviz_nbclust(df,clara,method="silhouette") + theme_classic()
```




```
clara.res <- clara(df,6,samples=50,pamLike=TRUE)
fviz_cluster(clara.res,
        palette = c("darkred","darkgreen","darkblue","purple","black","yellow"), # color palette
        ellipse.type = "t", # Concentration ellipse
        geom = "point", pointsize = 0.5,title="Cluster plot of t-sne coordinates tf-idf",
        ggtheme = theme_classic())
```



3.4.4.2. Word importance with the bind_tf_idf() function

To give some context to the clusters we derived, we examine which words are important to documents within each of the clusters. We quantify each term inside each document by giving weights based on the concept called “tf-inf” (term frequency-inverse document frequency).

The purpose of tf-idf is to find the important words by increasing the weight of words that are not very much used in a corpus of documents and by decreasing the weight of commonly used words. To do this, we first created a tidy format of the tweets and the cluster membership we gathered in the preceeding session.

```
dd <- cbind(df,cluster=clara.res$cluster)
dd$document <- as.numeric(rownames(dd))
dd$document <- as.character(dd$document)
dd$cluster <- as.factor(dd$cluster)
final <- left_join(tweets_td,dd)
final2 <- final %>%
 group_by(cluster) %>%
 count(cluster,term,sort=TRUE) %>%
 ungroup()
totals <- final2 %>%
 group_by(cluster) %>%
 summarize(total=sum(n))
cluster_words <- left_join(final2,totals)
cluster_words
```

```
> cluster_words
# A tibble: 4,974 x 4
   cluster           term     n total
    <fctr>          <chr> <int> <int>
 1       5   applications   220  3935
 2       5        finance   219  3935
 3       5         future   218  3935
 4       5        present   218  3935
 5       6      financial   201  2961
 6       6         global   201  2961
 7       6      institute   201  2961
 8       6       mckinsey   201  2961
 9       6         primer   201  2961
10       6 servicessource   201  2961
# ... with 4,964 more rows
```

```
ggplot(cluster_words,aes(n/total,fill=cluster)) +
  geom_histogram(show.legend=FALSE) +
  facet_wrap(~ cluster,ncol=2,scales="free_y") +
  ggtitle("Term frequency distribution in document clusters")
```

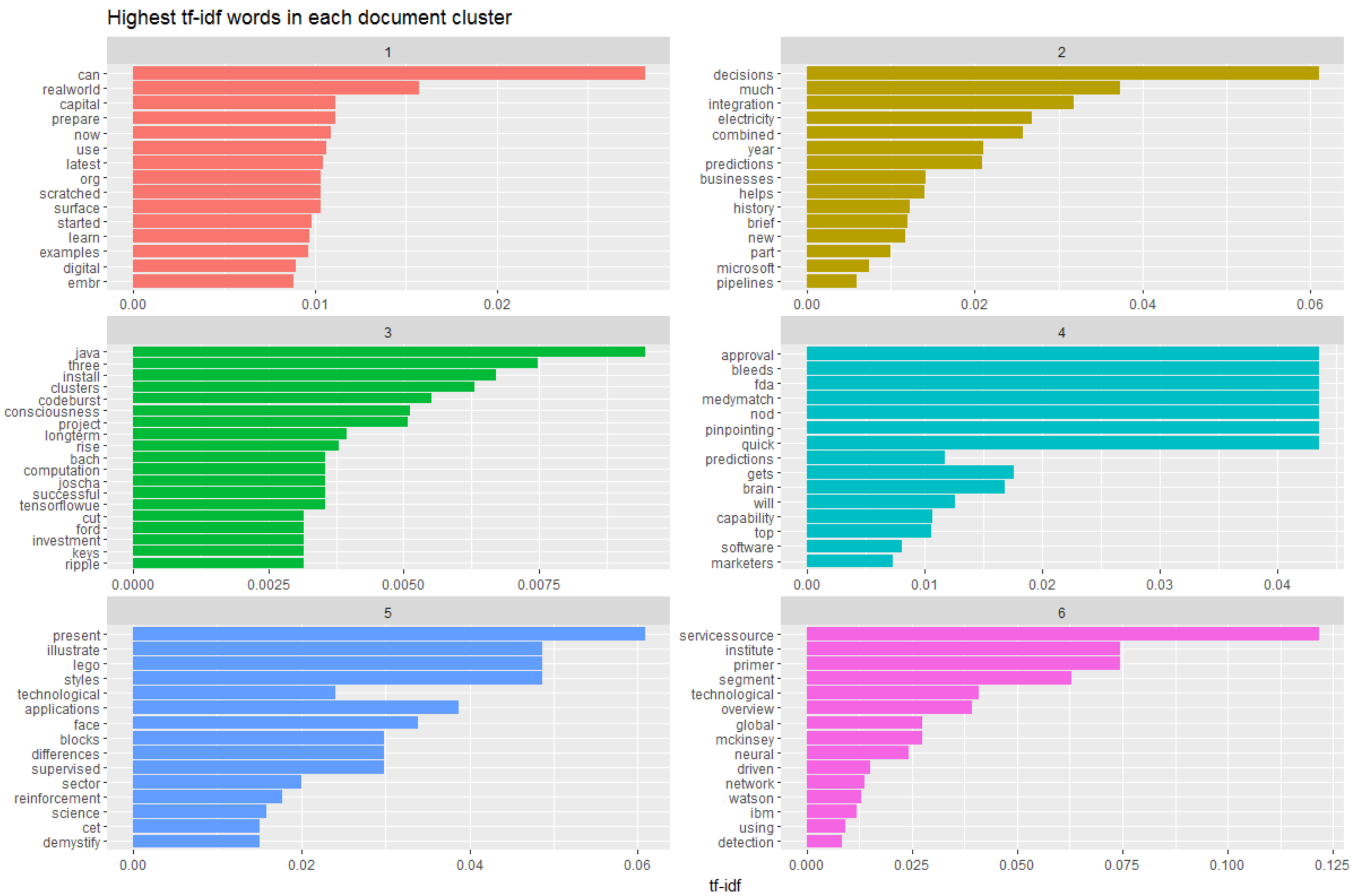
The bind_tf_idf() function in the tidytext package takes a tidy dataset as input with one row per token, per document. The first argument contains the tokens (term here), the second argument contains the documents (cluster) and a last necessary column contains the counts of how many times each document contains each token.

```
cluster_words <- cluster_words %>%
  bind_tf_idf(term,cluster,n)
cluster_words
```

```
> cluster_words
# A tibble: 4,974 x 7
   cluster          term     n total         tf       idf      tf_idf
    <fctr>         <chr> <int> <int>      <dbl>     <dbl>       <dbl>
 1        5  applications   220  3935 0.05590851 0.6931472 0.03875283
 2        5       finance   219  3935 0.05565438 0.1823216 0.01014699
 3        5        future   218  3935 0.05540025 0.0000000 0.00000000
 4        5       present   218  3935 0.05540025 1.0986123 0.06086340
 5        6     financial   201  2961 0.06788247 0.0000000 0.00000000
 6        6        global   201  2961 0.06788247 0.4054651 0.02752397
 7        6     institute   201  2961 0.06788247 1.0986123 0.07457652
 8        6      mckinsey   201  2961 0.06788247 0.4054651 0.02752397
 9        6        primer   201  2961 0.06788247 1.0986123 0.07457652
10        6 servicessource  201  2961 0.06788247 1.7917595 0.12162906
# ... with 4,964 more rows
```

The idf and tf-idf are zero for common words (e.g. “data”, “analytics”, “big”). These words appear in all clusters.

```
cluster_words %>%
  select(-total) %>%
  arrange(desc(tf_idf)) %>%
  mutate(term=factor(term,levels=rev(unique(term)))) %>%
  group_by(cluster) %>%
  top_n(15) %>%
  ungroup %>%
  ggplot(aes(term,tf_idf,fill=cluster)) +
  geom_col(show.legend=FALSE) +
  labs(x=NULL,y="tf-idf") +
  facet_wrap(~ cluster,ncol=2,scales="free") +
  coord_flip() +
  ggtitle("Highest tf-idf words in each document cluster")
```



3.4.5. Relationships between words

In the preceeding sections we analyzed words to infer sentiments or to consider the relationship between words and documents. Another topic of interest concerns the relationships between words.

To study relationships between words, we use the unnest_tokens() function to tokenize text into consecutive sequences of words (n-grams). We will explore pairs of words or bigrams. We start with a tidy dataset consisting of documents, terms and clusters.

```
doctermcl <- final  %>%
 select(document,term,cluster)
doctermcl
```

```
> doctermcl
# A tibble: 25,862 x 3
   document         term cluster
      <chr>        <chr>  <fctr>
 1        1      aspects       1
 2        1     business       1
 3        1      disrupt       1
 4        1         embr       1
 5        1     industry       1
 6        1 technologies       1
 7        2       automl       2
 8        2     generate       2
 9        2    pipelines       2
10        2         tpot       2
# ... with 25,852 more rows
```

```
# tokenizing by N-gram
tweets_bigrams <- doctermcl %>%
 unnest_tokens(bigram,term,token="ngrams",n=2)
tweets_bigrams
```

```
> tweets_bigrams
# A tibble: 20,913 x 3
   document cluster                bigram
      <chr>  <fctr>                 <chr>
 1        1       1      aspects business
 2        1       1      business disrupt
 3        1       1          disrupt embr
 4        1       1         embr industry
 5        1       1 industry technologies
 6       10       6        algorithms apps
 7       10       6          apps consume
 8       10       6        consume create
 9       10       6    create datascience
10       10       6      datascience ease
# ... with 20,903 more rows
```

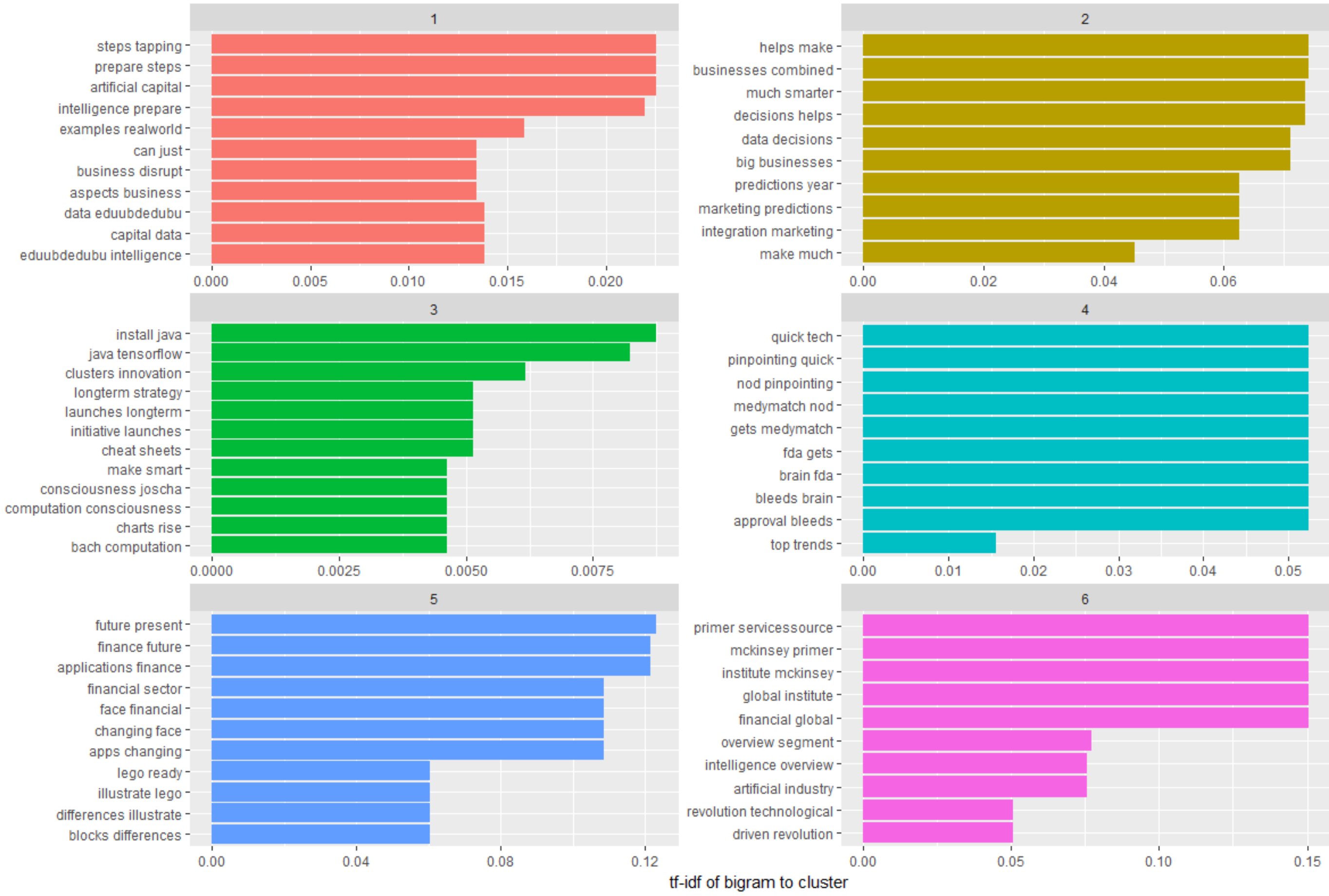
```
bigrams <- tweets_bigrams %>%
 count(cluster,bigram,sort=TRUE)
bigrams
```

```
> bigrams
# A tibble: 6,794 x 3
   cluster                bigram     n
    <fctr>                 <chr> <int>
 1       5        future present   218
 2       5  applications finance   215
 3       5        finance future   215
 4       6      financial global   201
 5       6      global institute   201
 6       6    institute mckinsey   201
 7       6        mckinsey primer   201
 8       6 primer servicessource   201
 9       5         apps changing   192
10       5         changing face   192
# ... with 6,784 more rows
```

```
bigrams_tfidf <- bigrams %>%
 bind_tf_idf(bigram,cluster,n) %>%
 arrange(desc(tf_idf))
bigrams_tfidf
```

```
> bigrams_tfidf
# A tibble: 6,794 x 6
   cluster                bigram     n         tf      idf    tf_idf
    <fctr>                 <chr> <int>      <dbl>    <dbl>     <dbl>
 1       6      financial global   201 0.08395990 1.791759 0.1504359
 2       6      global institute   201 0.08395990 1.791759 0.1504359
 3       6    institute mckinsey   201 0.08395990 1.791759 0.1504359
 4       6        mckinsey primer   201 0.08395990 1.791759 0.1504359
 5       6 primer servicessource   201 0.08395990 1.791759 0.1504359
 6       5        future present   218 0.06879142 1.791759 0.1232577
 7       5  applications finance   215 0.06784475 1.791759 0.1215615
 8       5        finance future   215 0.06784475 1.791759 0.1215615
 9       5         apps changing   192 0.06058694 1.791759 0.1085572
10       5         changing face   192 0.06058694 1.791759 0.1085572
# ... with 6,784 more rows
```

```
bigrams_tfidf %>%
  mutate(bigram = reorder(bigram, tf_idf)) %>%
  group_by(cluster) %>%
  top_n(10, tf_idf) %>%
  ungroup() %>%
  ggplot(aes(bigram
, tf_idf, fill = cluster)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ cluster, ncol = 2, scales = "free") +
  coord_flip() +
  labs(y = "tf-idf of bigram to cluster",
     x = "")
```



The above visualizations concentrate on the top terms in bigrams. To get an insight into the multiple relationships between terms, we make use of the igraph and ggraph package to visualize relationships between words with network diagrams.

With the igraph package we can arrange words (in this case bigrams) into a network. The tidy object (bigrams) has three variables (cluster, bigram and n) that we can use with the graph_from_data_frame() function to create a data frame of edges with columns “from” (the vertices, clusters), “to” (the edges, bigrams) and a weighting variable, in this case n.

```
bigram_graph <- bigrams %>%
  filter (n >=  25) %>%
  graph_from_data_frame()
bigram_graph
```

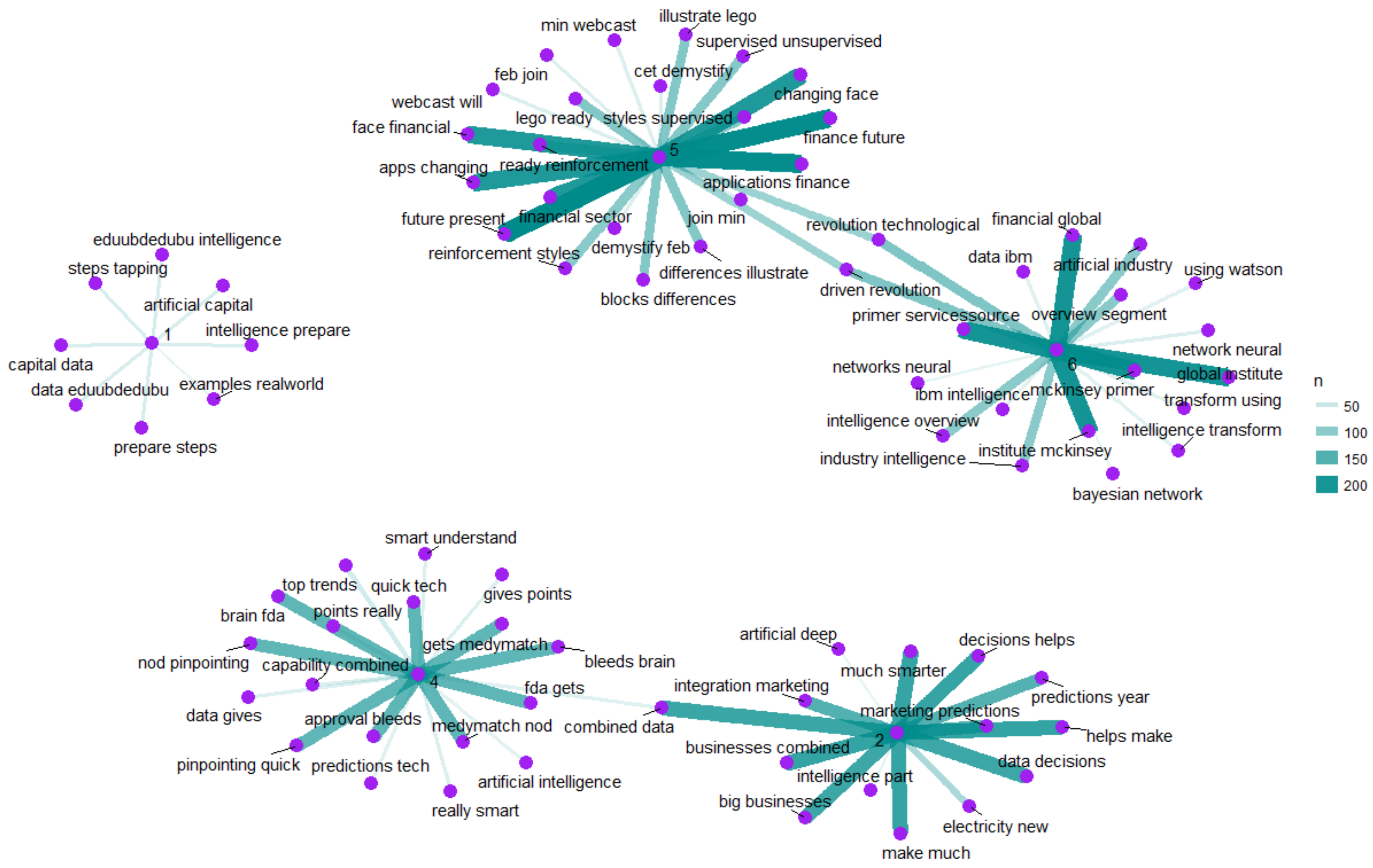
```
> bigram_graph
IGRAPH 00bf961 DN-- 82 80 --
+ attr: name (v/c), n (e/n)
+ edges from 00bf961 (vertex names):
 [1] 5->future present        5->applications finance  5->finance future        6->financial global
 [5] 6->global institute      6->institute mckinsey    6->mckinsey primer       6->primer servicessource
 [9] 5->apps changing         5->changing face         5->face financial        5->financial sector
[13] 2->businesses combined   2->helps make            2->decisions helps       2->make much
[17] 2->much smarter          2->combined data         2->big businesses        2->data decisions
[21] 2->integration marketing 2->marketing predictions 2->predictions year       4->approval bleeds
[25] 4->bleeds brain          4->brain fda             4->fda gets              4->gets medymatch
[29] 4->medymatch nod         4->nod pinpointing       4->pinpointing quick     4->quick tech
+ ... omitted several edges
```

The igraph object (bigram_graph) can be converted into a graph with the ggraph package. The graph has three layers : nodes, edges and text.

```
set.seed(123)
ggraph(bigram_graph,layout="fr") +
  geom_edge_link(aes(edge_alpha=n,edge_width=n),edge_colour="cyan4") +
  geom_node_point(color="purple",size=4) +
  geom_node_text(aes(label=name),repel=TRUE,point.padding=unit(0.2,"lines")) +
  theme_void() +
  ggtitle("Bigram network in tweets #MachineLearning")
```

The network diagram on the next page gives us a visualization of the relationships between co-occurring words within clusters. From the diagram it is clear that words in the dataset of machine learning tweets are organized into several clusters of word networks that are used together (note that no bigrams from cluster 3 were selected with the criteria (n >= 25) we used in the above code).

Co-occurrence network of words in tweets #MachineLearning



It is also possible to examine all the relationships between words through the use of the correlations between the words. From a tidy object that consists of the documents, terms and the clusters, we can create another tidy object with the correlations between terms within each cluster. The pairwise_cor() function from the widyr package lets us find the correlation between words based on how often they appear in the same cluster.

In a similar way as we used the ggraph() function to visualize bigrams, we can use this function to visualize the correlations within word clusters. In the graph on the page 32 we visualize networks of words where the correlation is relatively high (>= 0.50).

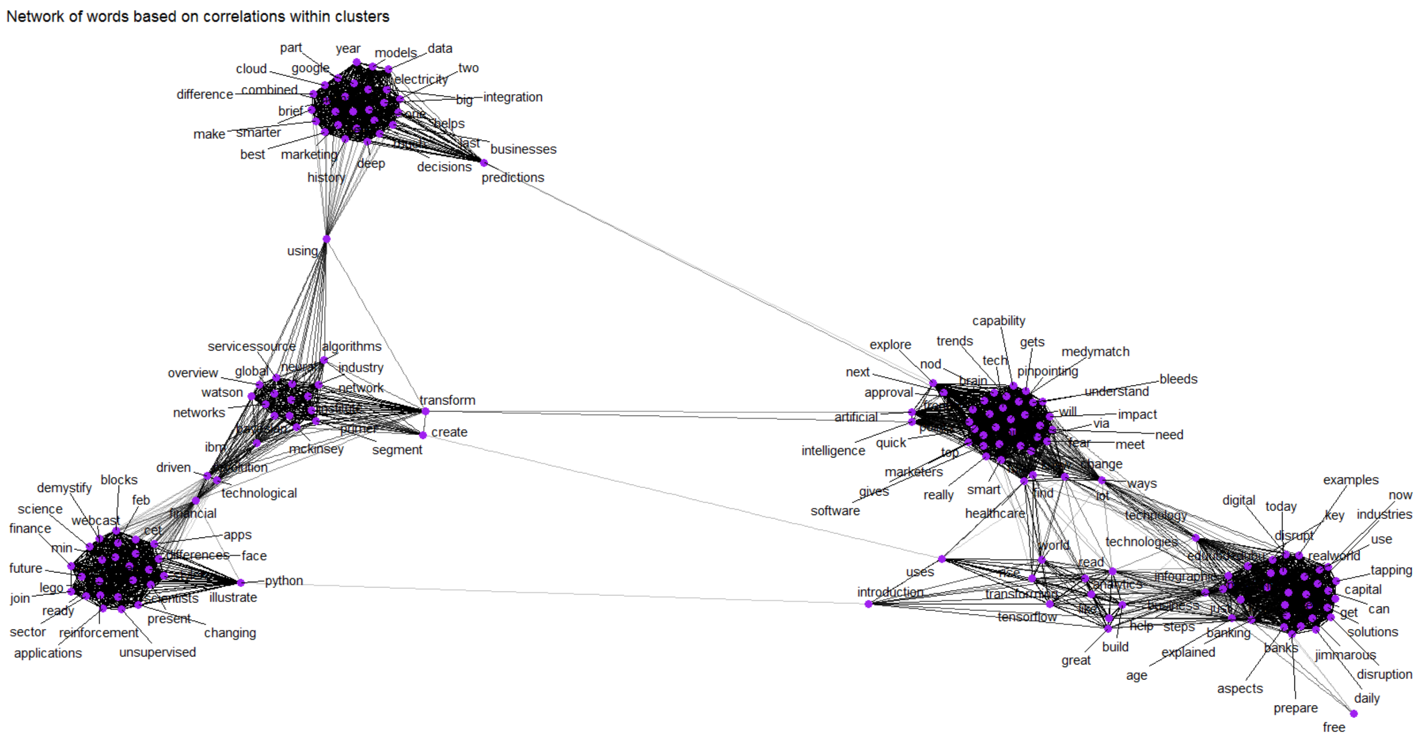
```
word_cor <- doctermcl %>%
  group_by(term) %>%
  pairwise_cor(term,cluster) %>%
  filter(!is.na(correlation))
word_cor
```

```
> word_cor
# A tibble: 11,158,940 x 3
          item1   item2 correlation
          <chr>   <chr>       <dbl>
 1     business aspects  0.88564420
 2      disrupt aspects  0.97986589
 3         embr aspects  0.99069827
 4     industry aspects -0.06079038
 5 technologies aspects  0.83469011
 6       automl aspects -0.23147156
 7     generate aspects  0.34295306
 8    pipelines aspects -0.23147156
 9         tpot aspects -0.23147156
10        using aspects -0.29493155
# ... with 11,158,930 more rows
```

```
word_graph <- doctermcl %>%
  group_by(term) %>%
  filter(n() >= 25) %>%
  pairwise_cor(term,cluster) %>%
  filter(!is.na(correlation),
       correlation >= 0.50) %>%
  graph_from_data_frame()
word_graph
```

```
> word_graph
IGRAPH 780daa6 DN-- 160 4836 --
+ attr: name (v/c), correlation (e/n)
+ edges from 780daa6 (vertex names):
 [1] business    ->aspects disrupt     ->aspects technologies->aspects learn       ->aspects
 [5] banks       ->aspects solutions   ->aspects use         ->aspects realworld   ->aspects
 [9] just        ->aspects help        ->aspects can         ->aspects get         ->aspects
[13] explained   ->aspects infographic ->aspects machines    ->aspects age         ->aspects
[17] digital     ->aspects key         ->aspects examples    ->aspects technology  ->aspects
[21] daily       ->aspects latest      ->aspects today       ->aspects steps       ->aspects
[25] prepare     ->aspects free        ->aspects now         ->aspects disruption  ->aspects
[29] eduubdedubu ->aspects tapping     ->aspects industries  ->aspects banking     ->aspects
+ ... omitted several edges
```

```
set.seed(1234)
ggraph(word_graph,layout="fr") +
geom_edge_link(aes(edge_alpha=correlation),show.legend=FALSE) +
geom_node_point(color="purple",size=3) +
geom_node_text(aes(label=name),repel=TRUE,point.padding=unit(0.1,"lines")) +
theme_void() +
ggtitle("Network of words based on correlations within clusters")
```



3.5. Topic modeling

So far we analyzed document similarity by applying cluster analysis on the two-dimensional coordinates we extracted from a document term matrix weighted by term frequency-inverse document frequency. An important alternative to this approach is to make use of topic modeling. Topic modeling is an unsupervised machine learning technique that identifies topics by grouping clusters of words that co-occur together in a large corpus. A fitted model allows to better estimate the similarity between documents. The basic idea of topic modeling is that every document in a collection is a mixture of several latent topics and each topic is a mixture of several words.

To estimate the coefficients of the document-topic and the topic-word distributions we use a topic modeling algorithm known as Latent Dirichlet Allocation (LDA)[6]. We use LDA for topic modeling with the R package "topicmodels".

3.5.1. Document term matrix and data preparation

The data set is already cleaned, but before fitting a model we need to prepare the document term matrix we created earlier. The document term matrix consists of all the terms (25862) in the cleaned corpus.

```
> str(dtm)
List of 6
 $ i       : int [1:25862] 1 1 1 1 1 1 2 2 2 2 ...
 $ j       : int [1:25862] 1 2 3 4 5 6 7 8 9 10 ...
 $ v       : num [1:25862] 1 1 1 1 1 1 1 1 1 1 ...
 $ nrow    : int 5000
 $ ncol    : int 3341
 $ dimnames:List of 2
  ..$ Docs : chr [1:5000] "1" "2" "3" "4" ...
  ..$ Terms: chr [1:3341] "aspects" "business" "disrupt" "embr" ...
 - attr(*, "class")= chr [1:2] "DocumentTermMatrix" "simple_triplet_matrix"
 - attr(*, "weighting")= chr [1:2] "term frequency" "tf"
```

Terms that occur frequently in the above document term matrix are associated a high value. As stated earlier, a high frequency of a term in a document is of less meaning when the term appears frequently in other documens in the corpus. Otherwise stated, terms that occur frequently within a document but not frequently in the corpus should receive a higher weighting as these words are assumed to contain more meaning in relation to the document. To achieve this, we downweight terms that occur frequently across documents by making use of tf-idf statistics. Therefore, in the next step we apply the term frequency inverse document frequency (tf-idf), a numerical statistic intended to reflect how important a word is to a document in a collection (corpus). The tf-idf value increases proportionally to the number of times a word appears in a document, but is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general.

We use the term frequency-inverse document frequency (tf-idf) as a weighting factor to reduce the number of terms in the document term matrix.

```
term_tfidf <- tapply(dtm$v/slam::row_sums(dtm)[dtm$i], dtm$j, mean) *
  log2(tm::nDocs(dtm)/slam::col_sums(dtm > 0))
summary(term_tfidf)
```

```
> summary(term_tfidf)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.5179  1.3148  1.6167  1.9290  2.0694 12.2877
```

We use the median (1.6167) as a lower limit, removing terms from the document term matrix with a tf-idf value smaller than 1.6167.

```
# keeping the rows with tfidf >= 1.6167
dtm <- dtm[,term_tfidf >= 1.6167]
summary(slam::col_sums(dtm))
```

```
> summary(slam::col_sums(dtm))
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   1.000   2.000   3.806   3.000 153.000
```

```
rowTotals<-apply(dtm,1,sum) #running this line takes time
empty.rows<-dtm[rowTotals==0,]$dimnames[1][[1]]
corpus<-corpus[-as.numeric(empty.rows)]
dtm <- DocumentTermMatrix(corpus)
str(dtm)
```

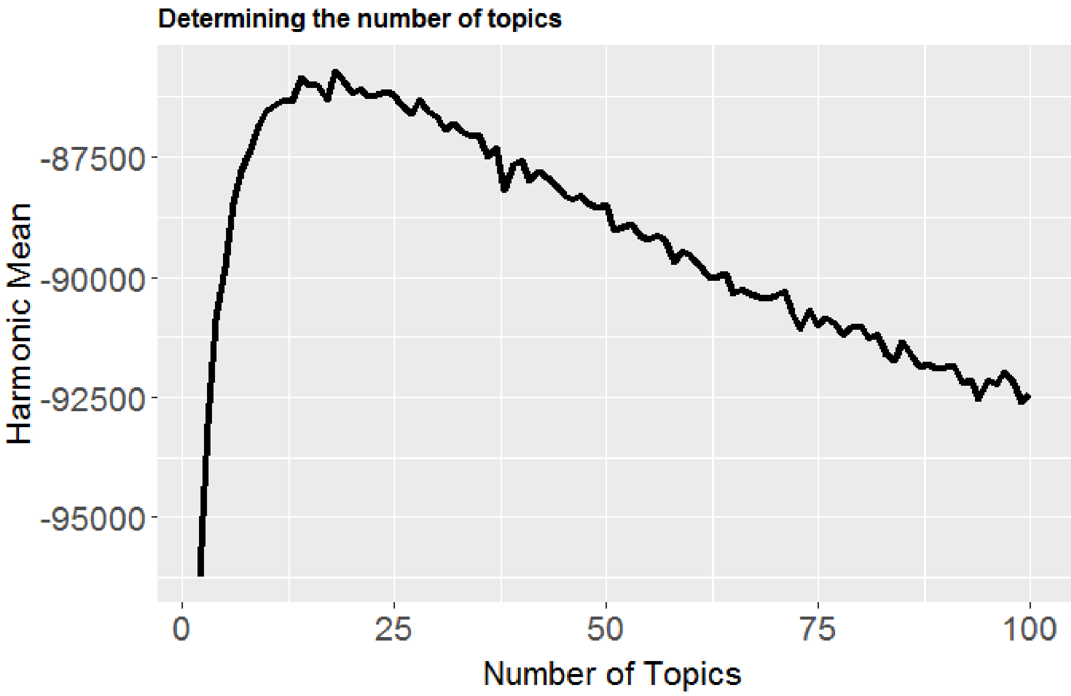
```
> str(dtm)
List of 6
 $ i       : int [1:13484] 1 1 1 1 1 2 2 2 2 2 ...
 $ j       : int [1:13484] 1 2 3 4 5 6 7 8 9 10 ...
 $ v       : num [1:13484] 1 1 1 1 1 1 1 1 1 1 ...
 $ nrow    : int 2911
 $ ncol    : int 3040
 $ dimnames:List of 2
  ..$ Docs : chr [1:2911] "1" "2" "3" "4" ...
  ..$ Terms: chr [1:3040] "automl" "generate" "pipelines" "tpot" ...
 - attr(*, "class")= chr [1:2] "DocumentTermMatrix" "simple_triplet_matrix"
 - attr(*, "weighting")= chr [1:2] "term frequency" "tf"
```

Working with tf-idf and excluding documents with zero terms substantially reduces the number of documents (from 5000 to 2911).

3.5.2. Determining the number of topics

One of the main issues in topic modeling is determining the number of topics[7]. Here we use a mathematical approach by calculating the harmonic mean over a sequence of topic models with different values for k (between 2 and 100). The optimal number of topics corresponds with the number of topics with the highest harmonic mean.

```
# parameters to produce a topic model
burnin <- 1000
iter <- 1000
keep <- 50
# determining k (the number of topics)
seqk <- seq(2,100,1)
system.time(fitted_many <- lapply(seqk,function(k) topicmodels::LDA(dtm,k=k,
                                    method="Gibbs",control=list(burnin=burnin,iter=iter,keep=keep))))
# extract logliks from each topic
logLiks_many <- lapply(fitted_many, function(L)  L@logLiks[-c(1:(burnin/keep))])
# compute harmonicMean
harmonicMean <- function(logLikelihoods, precision = 2000L) {
  llMed <- median(logLikelihoods)
  as.double(llMed - log(mean(exp(-mpfr(logLikelihoods,
                      prec = precision) + llMed))))
}
hm_many <- sapply(logLiks_many, function(h) harmonicMean(h))
ldaplot <- ggplot(data.frame(seqk, hm_many), aes(x=seqk, y=hm_many)) + geom_path(lwd=1.5) +
  theme(text = element_text(family= NULL),
      axis.title.y=element_text(vjust=1, size=16),
      axis.title.x=element_text(vjust=-.5, size=16),
      axis.text=element_text(size=16),
      plot.title=element_text(size=20)) +
  xlab('Number of Topics') +
  ylab('Harmonic Mean') +
  ggtitle("Determining the number of topics")
ldaplot
```




```
k <- seqk[which.max(hm_many)]
k
```

```
> k
[1] 18
```

3.5.3. LDA model

To run the model, the LDA() function from the topicmodels package is used. A seed number is added to replicate the analysis with the optimum number of topics (18) we determined in the preceeding step. The function extracts the most likely topic for each document.

```
# LDA model with k = seqk[which.max(hm_many)]
seedNum <- 50
system.time(ldaOut <- topicmodels::LDA(dtm,k = k,method="Gibbs",
                    control=list(burnin=burnin,keep=keep,iter=2000,seed=seedNum)))
```

str(ldaOut)

```
> str(ldaOut)
Formal class 'LDA_Gibbs' [package "topicmodels"] with 16 slots
  ..@ seedwords      : NULL
  ..@ z              : int [1:13610] 3 8 12 2 4 1 11 8 12 10 ...
  ..@ alpha          : num 2.78
  ..@ call           : language topicmodels::LDA(x = dtm, k = 18, method = "Gibbs", control = list(bu
rnin = burnin, keep = keep,      iter = 2000| __truncated__
  ..@ Dim            : int [1:2] 2911 3040
  ..@ control        :Formal class 'LDA_Gibbscontrol' [package "topicmodels"] with 14 slots
  .. .. ..@ delta        : num 0.1
  .. .. ..@ iter         : int 3000
  .. .. ..@ thin         : int 2000
  .. .. ..@ burnin       : int 1000
  .. .. ..@ initialize   : chr "random"
  .. .. ..@ alpha        : num 2.78
  .. .. ..@ seed         : int 50
  .. .. ..@ verbose      : int 0
  .. .. ..@ prefix       : chr "C:\\Users\\STUDIE~1\\AppData\\Local\\Temp\\RtmpGuzWEJ\\file145c4c9827
f5"
  .. .. ..@ save         : int 0
  .. .. ..@ nstart       : int 1
  .. .. ..@ best         : logi TRUE
  .. .. ..@ keep         : int 50
  .. .. ..@ estimate.beta: logi TRUE
  ..@ k              : int 18
  ..@ terms          : chr [1:3040] "automl" "generate" "pipelines" "tpot" ...
  ..@ documents      : chr [1:2911] "1" "2" "3" "4" ...
  ..@ beta           : num [1:18, 1:3040] -9.28 -9.24 -6.27 -5.57 -9.25 ...
  ..@ gamma          : num [1:2911, 1:18] 0.0505 0.064 0.0505 0.0471 0.0675 ...
  ..@ wordassignments:List of 5
  .. ..$ i    : int [1:13484] 1 1 1 1 1 2 2 2 2 2 ...
  .. ..$ j    : int [1:13484] 1 2 3 4 5 6 7 8 9 10 ...
  .. ..$ v    : num [1:13484] 4 4 12 2 18 1 11 8 9 10 ...
  .. ..$ nrow: int 2911
  .. ..$ ncol: int 3040
  .. ..- attr(*, "class")= chr "simple_triplet_matrix"
  ..@ loglikelihood  : num -85655
  ..@ iter           : int 3000
  ..@ logLiks        : num [1:60] -87031 -85807 -85558 -85206 -85337 ...
  ..@ n              : int 13610
```

The output of LDA consists of a word-topic matrix and a document_topic matrix.

3.5.3.1. Word-topic relationships

```
# keywords associated with each topic
ldaOut.terms <- as.matrix(terms(ldaOut,10))
ldaOut.terms[1:10,]
```

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9
1	Data	like	will	neural	new	latest	trends	software	history
2	Science	tech	use	network	electricity	future	top	next	one
3	Python	age	solutions	google	marketing	daily	technology	guide	deep
4	Features	difference	iot	bayesian	introduction	learn	via	writes	brief
5	scientists	technologies	banks	amazon	infographic	great	business	best	human
6	programming	tensorflow	get	networks	need	business	get	new	last
7	predictive	company	predictions	cloud	murder	industry	financial	frontier	part
8	Read	opportunities	ship	world	critical	going	strategic	will	two
9	Started	insights	technology	transforming	robot	free	using	rise	combined
10	Shaping	things	embedded	using	algorithms	store	watch	engineers	created

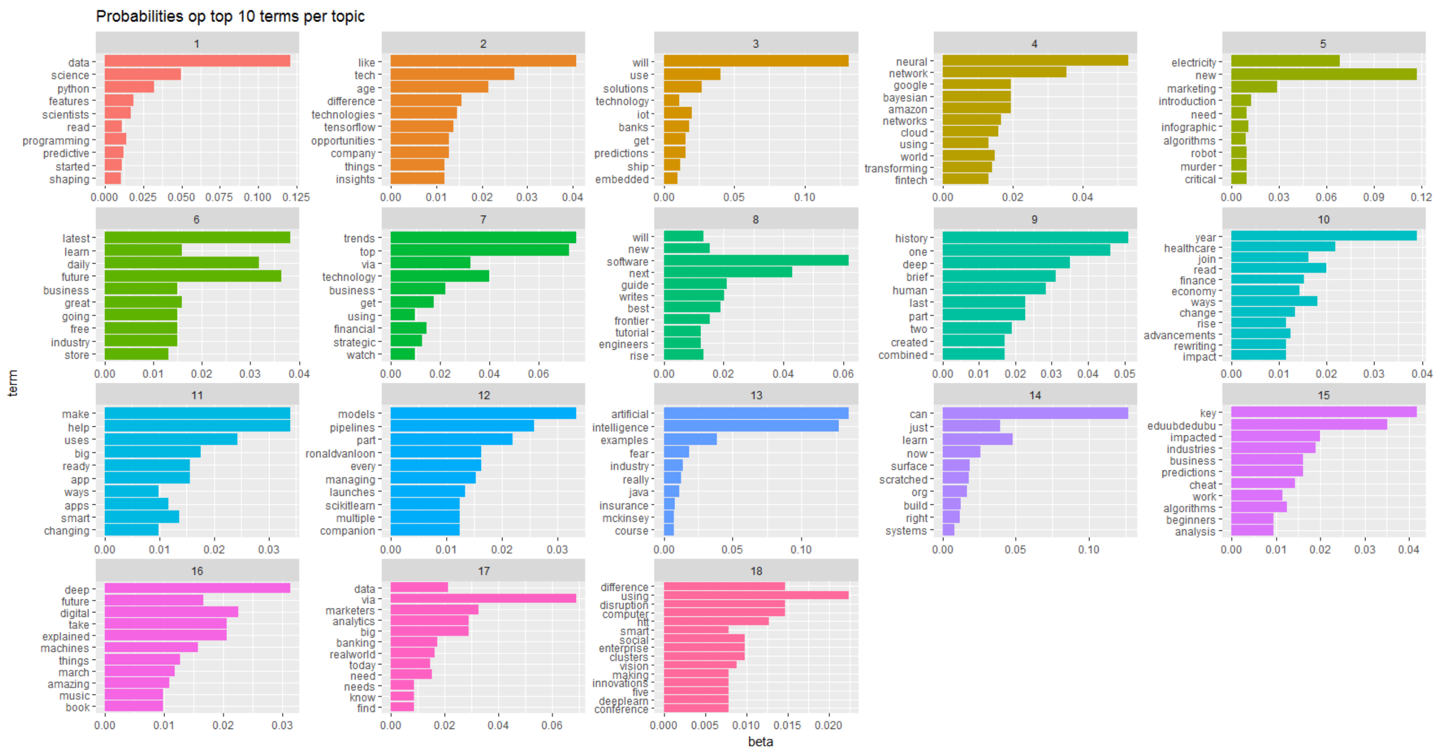
	Topic 10	Topic 11	Topic 12	Topic 13	Topic 14	Topic 15	Topic 16	Topic 17	Topic 18
1	Year	make	models	artificial	can	key	deep	via	using
2	healthcare	help	pipelines	intelligence	learn	eduubdedubu	digital	marketers	difference
3	Read	uses	part	examples	just	impacted	explained	analytics	disruption
4	Ways	big	ronaldvanloon	fear	now	industries	take	big	computer
5	Join	app	every	industry	surface	business	future	data	htt
6	Finance	ready	managing	really	scratched	predictions	machines	banking	enterprise
7	economy	smart	launches	java	org	cheat	things	realworld	clusters
8	Change	apps	multiple	insurance	build	algorithms	march	need	social
9	advancements	changing	scikitlearn	course	right	work	amazing	today	vision
10	Rise	ways	companion	mckinsey	systems	analysis	book	know	smart

```
write.csv (ldaOut.topics,file=paste("topic_model",k,"DocsToTopics.csv"))
```

The plot on the following page visualizes the probability distribution of the top 10 terms in each topic. Each bar represents the probability of a word to be selected when drawing terms from a specific topic.

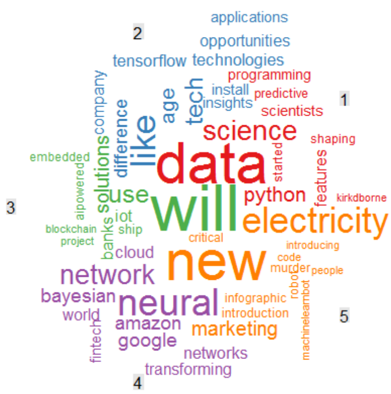
```
topics_beta <- tidy(ldaOut,matrix="beta")
top_terms <- topics_beta %>%
  group_by(topic) %>%
  top_n(10,beta) %>%
  ungroup() %>%
  arrange(topic, -beta)

top_terms %>%
    mutate(term=reorder(term,beta)) %>%
    ggplot(aes(term,beta,fill=factor(topic))) +
    geom_col(show.legend=FALSE) +
    facet_wrap(~ topic,scales="free") +
    coord_flip() +
    ggtitle("Probabilities op top 10 terms per topic")
```



Another way to visualise the top N words is using wordclouds to represent words that are informative, i.e. words that are uniquely associated with topics.

```
topic1_5 <- topics_beta %>%
  group_by(term) %>%
  top_n(1,beta) %>%
  group_by(topic) %>%
  top_n(10,beta) %>%
  filter(topic < 6) %>%
  acast(term ~ topic,value.var="beta",fill=0) %>%
  comparison.cloud(title.size=1,colors = brewer.pal(5,"Set1"))
```

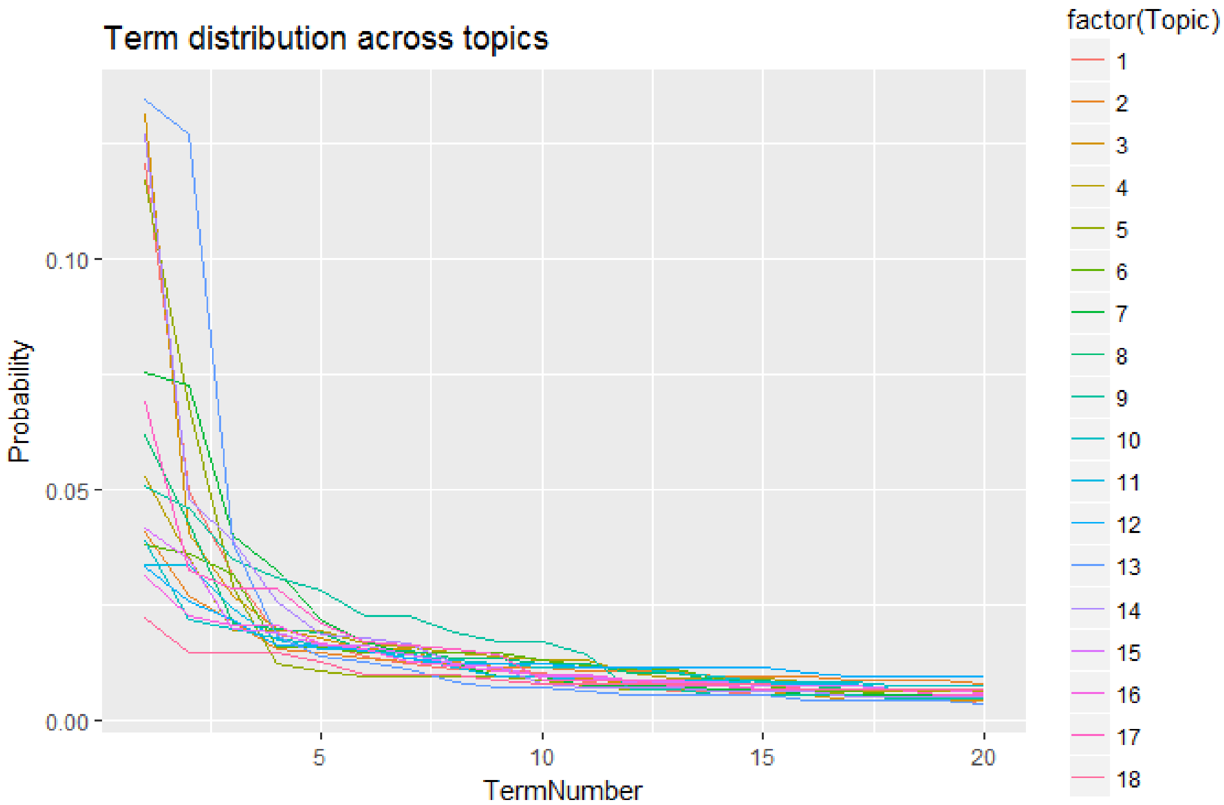



```
topics6_10 <- topics_beta %>%
  group_by(term) %>%
  top_n(1,beta) %>%
  group_by(topic) %>%
  top_n(10,beta) %>%
  filter(topic >= 6 & topic <= 10) %>%
  acast(term ~ topic,value.var="beta",fill=0) %>%
  comparison.cloud(title.size=1,colors = brewer.pal(5,"Set1"))
```

```
topics11_18 <- topics_beta %>%
  group_by(term) %>%
  top_n(1,beta) %>%
  group_by(topic) %>%
  top_n(10,beta) %>%
  filter(topic >= 11 & topic <= 18) %>%
  acast(term ~ topic,value.var="beta",fill=0) %>%
  comparison.cloud(title.size=1,colors = brewer.pal(8,"Set1"))
```

To examine the term distributions visually, the following can be used to create a quick plot of the term distributions across the topics[8].

```
termgenerator <- posterior(ldaOut)$terms
# relative probabilities of words in each topic
termimportance <- apply(termgenerator,1,
                    function(x) x[order(x,decreasing=T)[1:20]])
termimportance.longform <- melt(termimportance, value.name="Probability",
                          varnames=c("TermNumber","Topic"))
ggplot(data=termimportance.longform,
      aes(
       x=TermNumber,
       y=Probability,
       color=factor(Topic),
       group=Topic)) +
  geom_line() +
  ggtitle ("Term distribution across topics")
```



As can be seen, the first 5-10 words account for the majority of the probability that make up each topic, meaning that only a small number of words define the topic and therefore whether a document fits into it or not based on the words in that document.

To label the topics, the terms are ranked and a label is created by cancatenating the first three term per topic[9].

```
ML.topics <- topicmodels::topics(ldaOut, 1)
ML.terms <- as.data.frame(topicmodels::terms(ldaOut, 30), stringsAsFactors = FALSE)
topicTerms <- tidyr::gather(ML.terms, Topic)
topicTerms <- cbind(topicTerms, Rank = rep(1:30))
topTerms <- dplyr::filter(topicTerms, Rank < 4)
topTerms <- dplyr::mutate(topTerms, Topic = stringr::word(Topic, 2))
topTerms$Topic <- as.numeric(topTerms$Topic)
topicLabel <- data.frame()
for (i in 1:18){
  z <- dplyr::filter(topTerms, Topic == i)
  l <- as.data.frame(paste(z[1,2], z[2,2], z[3,2], sep = " " ), stringsAsFactors = FALSE)
  topicLabel <- rbind(topicLabel, l)

}
colnames(topicLabel) <- c("Label")
topicLabel
```

```
> topicLabel
                                   Label
1                data science python
2                      like tech age
3                  will use solutions
4               neural network google
5           new electricity marketing
6                latest future daily
7               trends top technology
8                 software next guide
9                    history one deep
10              year healthcare read
11                    make help uses
12              models pipelines part
13 artificial intelligence examples
14                    can learn just
15           key eduubdedubu impacted
16             deep digital explained
17            via marketers analytics
18       using difference disruption
```

We can also examine the tweets from a particular topic in greater detail. Let's take topic 13 ("artificial intelligence examples").

```
topicsProb <- read.csv("topic_model 18 DocsToTopics.csv")
topic13tweets <- which(topicsProb$V1==13)
tweetscorpus <- corpus
topic13TweetsText <- as.list(tweetscorpus[topic13tweets])
sampleTweets <- sample(topic13TweetsText,5)
sampleTweets
```

```
> sampleTweets
[[1]]
[1] " banks can ride artificial intelligence wave"
[[2]]
[1] "artificial intelligence trends will rule"
[[3]]
[1] "artificial intelligence used detect alzheimers early"
[[4]]
[1] " artificial intelligence fits cybersecurity"
[[5]]
[1] "oldline industry tools advanced technology solutions"
```

3.5.3.2. Document-topic relationships

Let's now examine the second part of the LDA output : the probabilities with which each topic is assigned to a document.

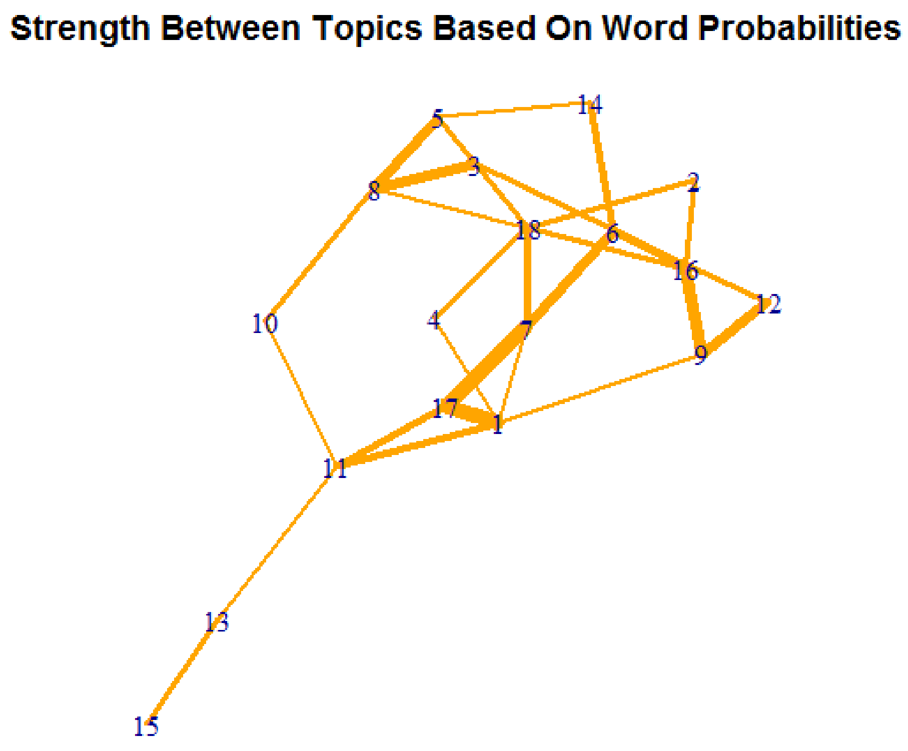
```
# probabilities associated with each topic assignment
topicProbabilities <- as.data.frame(ldaOut@gamma)
head(topicProbabilities)
```

```
> head(topicProbabilities)
          V1         V2         V3         V4         V5         V6         V7         V8         V9
1 0.05050505 0.06868687 0.06868687 0.06868687 0.05050505 0.05050505 0.05050505 0.06868687 0.05050505
2 0.06403013 0.04708098 0.04708098 0.04708098 0.04708098 0.04708098 0.04708098 0.06403013 0.04708098
3 0.05050505 0.05050505 0.05050505 0.05050505 0.05050505 0.05050505 0.05050505 0.05050505 0.08686869
4 0.04708098 0.08097928 0.04708098 0.04708098 0.06403013 0.06403013 0.06403013 0.04708098 0.04708098
5 0.06746032 0.04960317 0.04960317 0.04960317 0.04960317 0.04960317 0.04960317 0.06746032 0.04960317
6 0.05341880 0.05341880 0.05341880 0.05341880 0.05341880 0.05341880 0.05341880 0.05341880 0.05341880
         V10        V11        V12        V13        V14        V15        V16        V17        V18
1 0.05050505 0.05050505 0.06868687 0.05050505 0.05050505 0.05050505 0.05050505 0.05050505 0.05050505
2 0.06403013 0.08097928 0.06403013 0.06403013 0.04708098 0.04708098 0.04708098 0.08097928 0.04708098
3 0.05050505 0.06868687 0.05050505 0.05050505 0.06868687 0.05050505 0.06868687 0.05050505 0.05050505
4 0.04708098 0.04708098 0.06403013 0.04708098 0.06403013 0.04708098 0.04708098 0.04708098 0.08097928
5 0.04960317 0.06746032 0.04960317 0.06746032 0.04960317 0.04960317 0.08531746 0.04960317 0.04960317
6 0.05341880 0.07264957 0.05341880 0.05341880 0.05341880 0.07264957 0.05341880 0.05341880 0.05341880
```

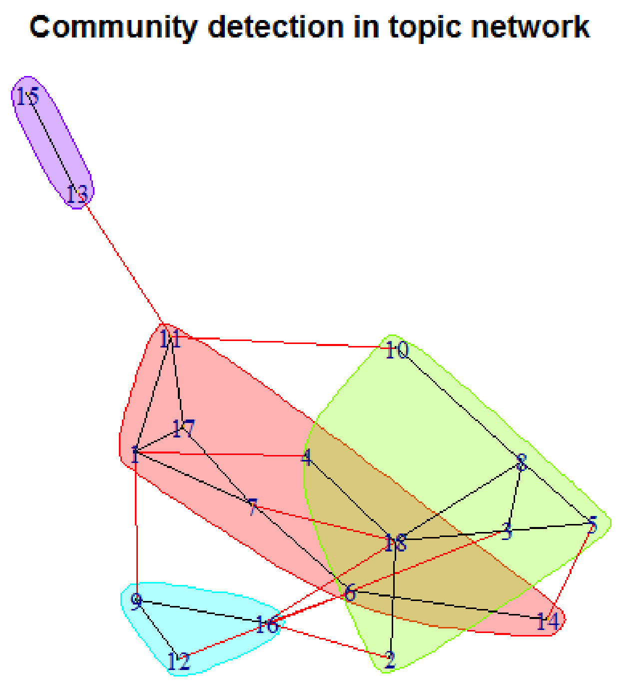
The above probabilities are evidence that many documents can be considered to be a mixture of several topics. For document 1, for example, the highest probabilities occur for topics 2, 3, 4, 8 and 12. Likewise, for document 6 the highest probabilities occur for topic 11 and 15. Topics are interrelated and cluster.

The following code is aimed to detect communities of correlated topics[10] :

```
bgg.post <- posterior(ldaOut)
bgg.cor_mat <- cor(t(bgg.post[["terms"]]))
bgg.cor_mat[bgg.cor_mat < .001 ] <- 0
diag(bgg.cor_mat) <- 0
bgg.graph <- graph.adjacency(bgg.cor_mat, weighted=TRUE, mode="lower")
bgg.graph <- delete.edges(bgg.graph, E(bgg.graph)[ weight < 0.05])
E(bgg.graph)$edge.width <- E(bgg.graph)$weight * 50
V(bgg.graph)$size <- colSums(bgg.post[["topics"]])/200
par(mar=c(0, 0, 3, 0))
set.seed(110)
plot.igraph(bgg.graph, edge.width = E(bgg.graph)$edge.width,
        main = "Strength Between Topics Based On Word Probabilities",
        edge.color = "orange",
        vertex.color = "orange",
        vertex.frame.color = NA)
```




```
clp <- cluster_label_prop(bgg.graph)
class(clp)
plot(clp,bgg.graph,main="Community detection in topic network")
```

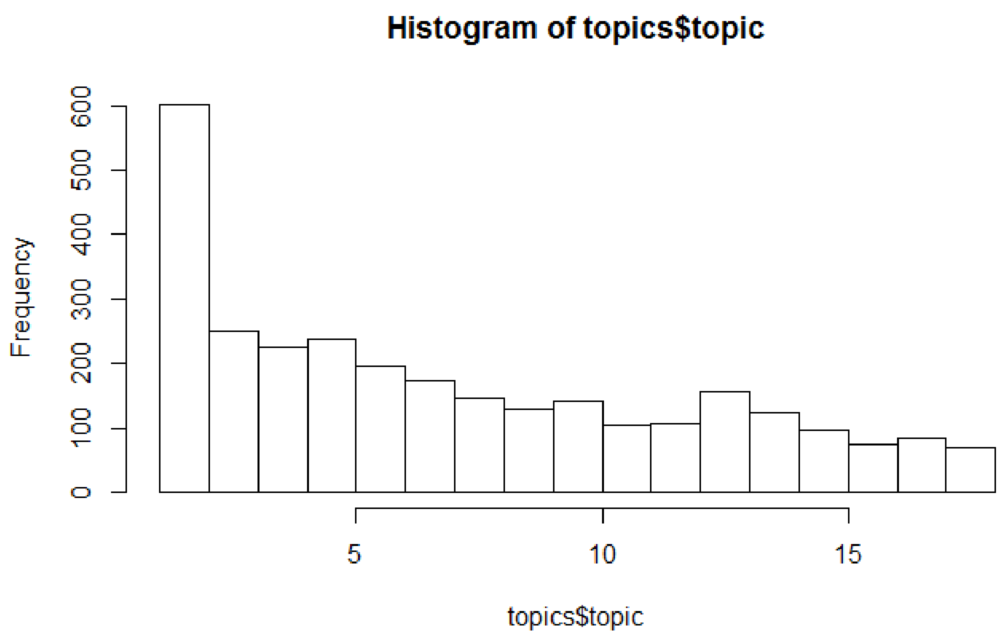


To visualise the LDA topic model, we create a data frame that consists of the topic to which each document is assigned to and the 18 topic probabilities. Then a t-sne analysis is performed on the topic probabilities of the 18 topics.

```
topics <- read.csv("topic_model 18 DocsToTopics.csv")
names(topics) <- c("X","topic")
str(topics)
```

```
> str(topics)
'data.frame':	2911 obs. of  2 variables:
 $ X    : int  1 2 3 4 5 6 7 8 9 10 ...
 $ topic: int  2 11 9 2 16 11 1 6 5 3 ...
```

```
hist(topics$topic)
```

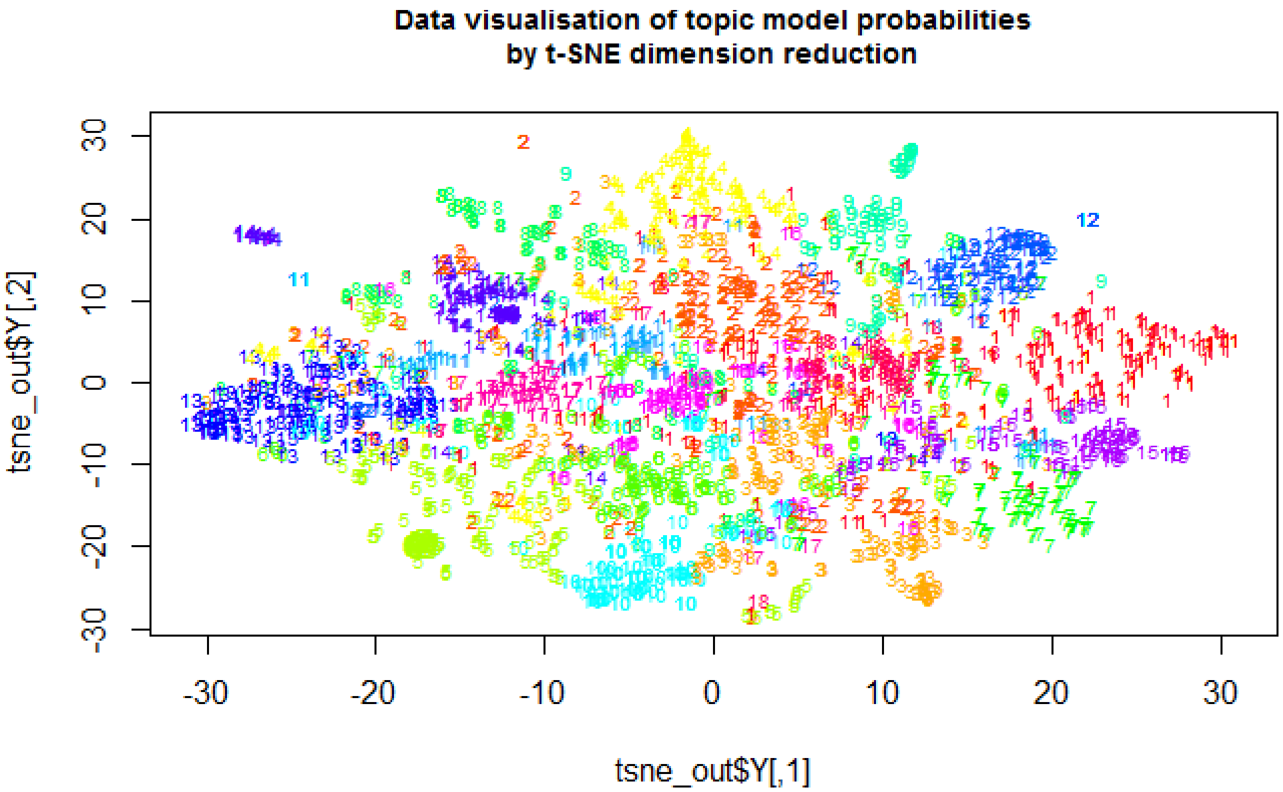



```
gamma <- read.csv("LDAGibbs 18 TopicProbabilities.csv")
all <- merge(topics,gamma,by="X",all.X=TRUE)
str(all)
mall <- as.matrix(all)
```

In order to visualize the tweets on a 2D map we used t-SNE as a dimensionality reduction algorithm to give each tweet an (x,y) coordinate in a 2D space which allows us to put the tweets on a scatter plot. The plot below shows the result of the dimension reduction applied on the (18) feature vectors that list the probabilities with which each topic is assigned to the documents (tweets).

```
set.seed(2018)
colors <- rainbow(length(unique(all$topic)))
tsne_out <- Rtsne(mall[,-(1:2)],dims=2,check_duplicates=F,perplexity=80,verbose=TRUE,max_iter=500)
plot(tsne_out$Y,t="n",main="Data visualisation of topic model probabilities\nby t-SNE dimension reduction",
   cex.main=0.9)
text(tsne_out$Y,labels=all$topic,cex=0.60,col=colors[all$topic])
```

The plot reveals complex relationships. On the one hand, tweets are separated quite nicely and cluster to homogeneous groups. The points that belong to the same topic are well separated. But on the other hand, topics also spread out.

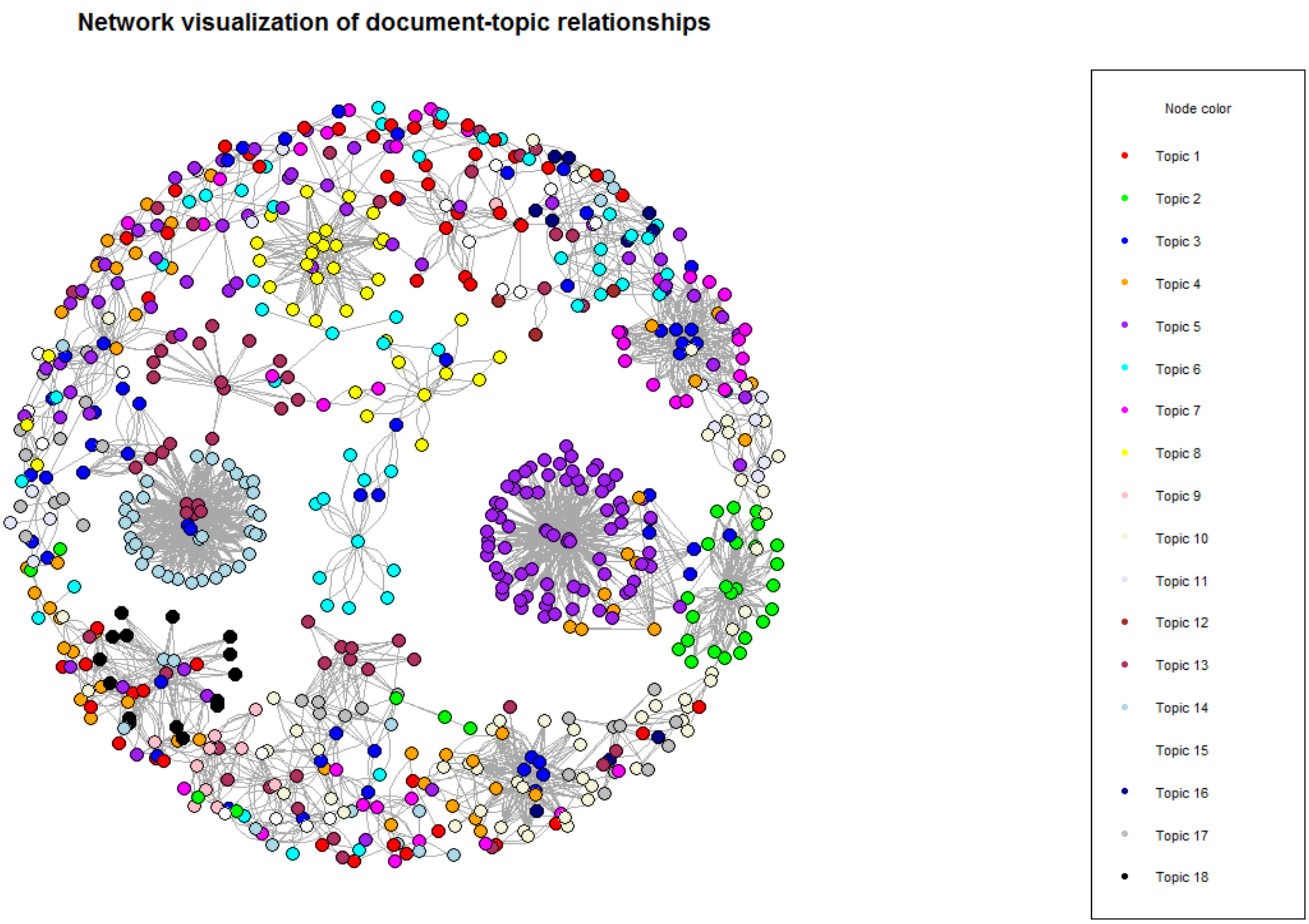


An alternative way to visualise distances between texts, is to calculate Euclidean distances between documents based on topic probabilities and visualise these distances in a network graph[11]. The following code calculates Euclidean distances between documents and converts the results into an adjacency matrix to be used for visualising a network structure.

```
topic_df <- as.data.frame(ldaOut@gamma)
topic_df_dist <- as.matrix(daisy(topic_df,metric="euclidean",stand=TRUE))
topic_df_dist[sweep(topic_df_dist,1,(apply(topic_df_dist,1,min) +
              apply(topic_df_dist,1,sd))) > 0] <- 0
g <- as.undirected(graph.adjacency(topic_df_dist))
bad.vs <- V(g)[degree(g)==0]
g.copy <- delete.vertices(g,bad.vs)
```

```
edge_table <- get.data.frame(g.copy)
node_table <- data.frame(name=unique(c(as.numeric(edge_table$from),
                                as.numeric(edge_table$to)))) %>%
  left_join(topics,by=c("name" = "X")) %>%
  unique()
graph <- graph_from_data_frame(edge_table,directed=FALSE,vertices=node_table)
V(graph)[V(graph)$topic==1]$color="Red"
V(graph)[V(graph)$topic==2]$color="Green"
V(graph)[V(graph)$topic==3]$color="Blue"
V(graph)[V(graph)$topic==4]$color="Orange"
V(graph)[V(graph)$topic==5]$color="Purple"
V(graph)[V(graph)$topic==6]$color="Cyan"
V(graph)[V(graph)$topic==7]$color="Magenta"
V(graph)[V(graph)$topic==8]$color="Yellow"
V(graph)[V(graph)$topic==9]$color="Pink"
V(graph)[V(graph)$topic==10]$color="Beige"
V(graph)[V(graph)$topic==11]$color="Lavender"
V(graph)[V(graph)$topic==12]$color="Brown"
V(graph)[V(graph)$topic==13]$color="Maroon"
V(graph)[V(graph)$topic==14]$color="Lightblue"
V(graph)[V(graph)$topic==15]$color="White"
V(graph)[V(graph)$topic==16]$color="Navy"
V(graph)[V(graph)$topic==17]$color="Grey"
V(graph)[V(graph)$topic==18]$color="Black"
plot(graph,layout=layout.kamada.kawai,vertex.size=3.5,vertex.color=V(graph)$color,vertex.label=NA)
title("Network visualization of document-topic relationships")
legend("topright",legend=c("Topic 1", "Topic 2", "Topic 3","Topic 4", "Topic 5", "Topic 6",
                    "Topic 7", "Topic 8", "Topic 9", "Topic 10", "Topic 11", "Topic 12",
                    "Topic 13","Topic 14", "Topic 15", "Topic 16", "Topic 17", "Topic 18"),
    pch=19,col=c("red","green","blue","orange","purple","cyan","magenta",
            "yellow","pink","beige","lavender","brown","maroon","lightblue",
            "white","navy","grey","black"),title="Node color",cex=0.70)
```

The graph on page 49 gives us a network visualization of how topics interlink due to their distribution in documents. From the graph it is clear that some topics uniquely group documents but a majority of documents share more topics. This is in line with the low topic probabilities in general (see page 44). Obviously, the brevity of each tweet (maximum 140 characters) is a reason why there is little to distinguish between some of the topics in the analysis. On the other hand, it is also possible that a smal(ler) number of topics reflects more accurately the structure of the data.



4. Conclusion

With the explosion of content generated by the use of social media, the analysis of short text documents has become an area of research in text mining. Using Twitter data, we applied various preprocessing techniques to transform a set of documents into a corpus. With the tidytext package we gained insights into the dimensions underlying the text in tweet messages. We also used a dimension reduction technique, t-sne, to cluster terms. Furthermore, topic modeling provides a way to classify a corpus of documents.

NOTES

[1] See for example I.A. Gray, *How to register a Twitter app in 8 easy steps*, https://iag.me/socialmedia/how-to-create-a-twitter-app-in-8-easy-steps.
[2] T. Kwartler, 2017, pp. 43.
[3] See L.J.P. van der Maaten, 2014.
[4] For a description of clara clustering, see http://www.sthda.com/english/articles/27-partitioning-clustering-essentials/89-clara-clustering-large-applications .
[5] See J. Silge & D. Robinson, 2017.
[6] See T. Graham & R. Ackland, 2015 for an introduction on topic modeling.
[7] See http://davidmeza1.github.io/2015/07/20/topic-modeling-in-R.html .
[8] Source code : P. Buckley, 2016, pp. 257.
[9] Source code : http://davidmeza1.github.io/2015/07/20/topic-modeling-in-R.html .
[10] Source code : R. Wesslen, 2016.
[11] https://stackoverflow.com/questions/16004847/visualise-distances-between-texts .

REFERENCES

K. Awati, *A gentle introduction to topic modeling using R*, https://eight2late.wordpress.com/2015/09/29/a-gentle-introduction-to-topic-modeling-using-r , 29 september 2015.

Chr. Benavent, *#IA à la moulinette du #ML*, https://benavent.fr/tag/rtsne ,9 août 2017.

P. Buckley, Topic modeling, in M. Hofmann & A. Chisholm (eds.), *Text mining and visualization. Case studies using open-source tools*, New York, CRC Press, 2016, pp. 241-264.

L. Davis & C. Sievert, *A topic model for movie reviews*, https://ldavis.cpsievert.me/reviews/reviews.html.

T. Graham & R. Ackland, Topic modeling of tweets in R : A tutorial and methodology, https://www.academia.edu/19255535/Topic_Modeling_of_Tweets_in_R_A_Tutorial_and_Methodology , 29 november 2015.

T. Kwartler, *Text mining in practice with R*, Oxford, John Wiley & Sons, 2017.

D. Meza, *Topic modeling in R*, http://davidmeza1.github.io/2015/07/20/topic-modeling-in-R.html.

H.T. Setiawan, Visualization of high dimensional data using tSNE with R, https://www.codeproject.com.tips/788739/visualization-of-high-dimensional-data-using-t-sne, 23 june 2014.

C. Sievert & K.E. Shirley, *LDAvis : A method for visualizing and interpreting topics*, Proceedings of the workshop on interactive language learning, visualization and interfaces, Baltimore, Maryland, USA, june 27, 2014, pp. 63-70.

J. Silge & D. Robinson, *Text mining with R*, Sebastopol, CA, O'Reilly Media, 2017.

L.J.P. van der Maaten, Accelerating t-SNE using Tree-Based Algorithms, *Journal of Machine Learning Research,* vol. 15, 2014, pp. :3221-3245.

R. Wesslen, *Using R to detect communities of correlated topics*, https://wesslen.github.io/text%20mining/topic-networks , 22 august 2016.

W.W.Xu, *What do twitter users talk about Mindhunter (2016) ? Leverage topic modeling for textual insights*, http://rstudio-pubs-static.s3.amazonaws.com/318764_e6d33551607f4dcda31c2e57d706d014.html, 15 october 2017.