THOMAS W. MILLER

FACULTY DIRECTOR OF NORTHWESTERN UNIVERSITY'S
PREDICTIVE ANALYTICS PROGRAM

MODELING TECHNIQUES IN PREDICTIVE ANALYTICS

BUSINESS PROBLEMS AND SOLUTIONS WITH R

REVISED AND EXPANDED EDITION

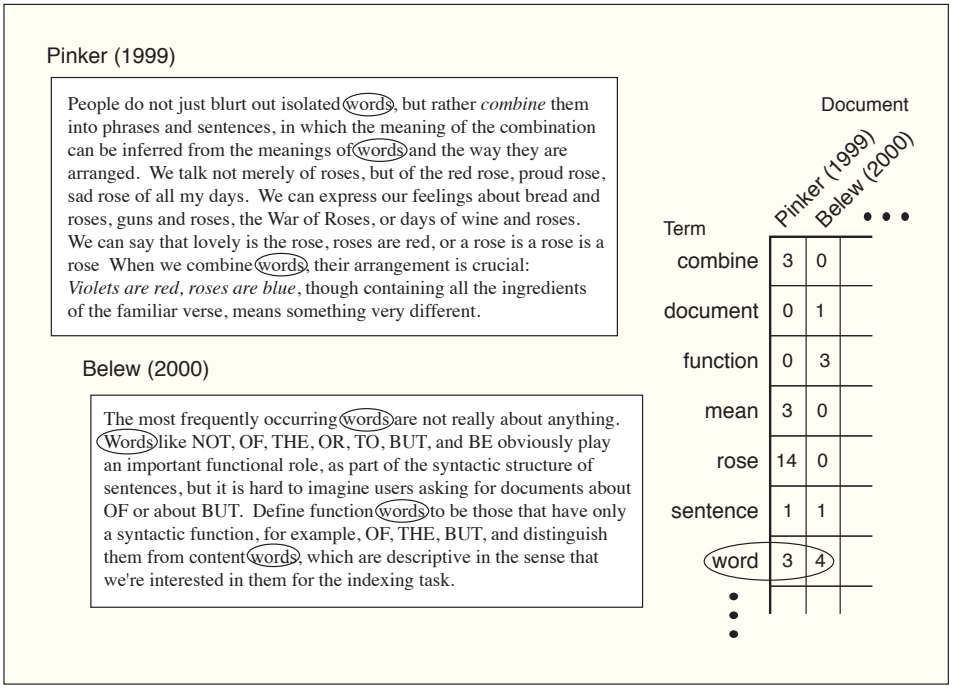
Modeling Techniques in Predictive Analytics

Business Problems and Solutions with R

Revised and Expanded Edition

THOMAS W. MILLER

Figure 7.9. *Creating a Terms-by-Documents Matrix*



Source: Adapted from Miller (2005).

and adjective form "functional." An alternative system might distinguish among parts of speech, permitting more sophisticated syntactic searches across documents. After being created, the terms-by-documents matrix is like an index, a mapping of document identifiers to terms (keywords or stems) and vice versa. For information retrieval systems or search engines we might also retain information regarding the specific location of terms within documents.

Typical text analytics applications have many more terms than documents, resulting in sparse rectangular terms-by-documents matrices. To obtain meaningful results for text analytics applications, analysts examine the distribution of terms across the document collection. Very low frequency terms, those used in few documents, are dropped from the terms-by-documents matrix, reducing the number of rows in the matrix.

Unsupervised text analytics problems are those for which there is no response or class to be predicted. Rather, as we showed with the movie taglines, the task is to identify common patterns or trends in the data. As part of the task, we may define text measures describing the documents in the corpus.

For supervised text analytics problems there is a response or class of documents to be predicted. We build a model on a training set and test it on a test set. Text classification problems are common. Spam filtering has long been a subject of interest as a classification problem, and many e-mail users have benefitted from the efficient algorithms that have evolved in this area. In the context of information retrieval, search engines classify documents as being relevant to the search or not. Useful modeling techniques for text classification include logistic regression, linear discriminant function analysis, classification trees, and support vector machines. Various ensemble or committee methods may be employed.

Automatic text summarization is an area of research and development that can help with information management. Imagine a text processing program with the ability to read each document in a collection and summarize it in a sentence or two, perhaps quoting from the document itself. Today's search engines are providing partial analysis of documents prior to their being displayed. They create automated summaries for fast information retrieval. They recognize common text strings associated with user requests. These applications of text analysis comprise tools of information search that we take for granted as part of our daily lives.

Programs with syntactic processing capabilities, such as IBM's Watson, provide a glimpse of what intelligent agents for text analytics are becoming. These programs perform grammatical parsing with an understanding of the roles of subject, verb, object, and modifier. They know parts of speech (nouns, verbs, adjective, adverbs). And, using identified entities representing people, places, things, and organizations, they perform relationship searches.

Those interested in learning more about text analytics can refer to Jurafsky and Martin (2009), Weiss, Indurkhya, and Zhang (2010) and the edited volume by Srivastava and Sahami (2009). Reviews may be found in Miller (2005), Trybula (1999), Witten, Moffat, and Bell (1999), Meadow, Boyce,

and Kraft (2000), Sullivan (2001), Feldman (2002b), and Sebastiani (2002). Hausser (2001) gives an account of generative grammar and computational linguistics. Statistical language learning and natural language processing are discussed by Charniak (1993), Manning and Schütze (1999), and Indurkhya and Damerau (2010).

The writings of Steven Pinker (1994, 1997, 1999) provide insight into grammar and psycholinguistics. Maybury (1997) reviews data preparation for text analytics and the related tasks of source detection, translation and conversion, information extraction, and information exploitation. Detection relates to identifying relevant sources of information; conversion and translation involve converting from one medium or coding form to another.

Belew (2000), Meadow, Boyce, and Kraft (2000) and the edited volume by Baeza-Yates and Ribeiro-Neto (1999) provide reviews of computer technologies for information retrieval, which depend upon text classification, among other technologies and algorithms.

Authorship identification, a problem addressed a number of years ago in the statistical literature by Mosteller and Wallace (1984), continues to be an active area of research (Joula 2008). Merkl (2002) provides discussion of clustering techniques, which explore similarities between documents and the grouping of documents into classes. Dumais (2004) reviews latent semantic analysis and statistical approaches to extracting relationships among terms in a document collection.

Special topics from computational linguistics provide additional insights into working with text. Text tiling (Hearst 1997) involves the automatic division of text documents into blocks or units for further analysis. Adjacent blocks of text are more likely to have words in common with one another and thus be topically related. Text tiling can be used in text summarization and information retrieval, as well as stylistic analysis of literature and discourse (Youmans 1990; Youmans 1991).

Modeling techniques for unsupervised text analytics include multivariate methods such as multidimensional scaling and cluster analysis, which are based upon dissimilarity or distance measures, and principal components analysis, which works from covariance or correlation matrices. Dissimilarity and distance measures and cluster analysis are discussed by Meyer (2014a, 2014b) and Kaufman and Rousseeuw (1990) and Izenman (2008),

with R code provided by Maechler (2014a). For multidimensional scaling, see Davison (1992), Cox and Cox (1994), Everitt and Rabe-Hesketh (1997), Izenman (2008) and Borg and Groenen (2010). Principal components are reviewed in various multivariate texts (Manly 1994; Sharma 1996; Gnanadesikan 1997; Johnson and Wichern 1998; Everitt and Dunn 2001; Seber 2000; Izenman 2008), with biplots discussed by Gabriel (1971) and Gower and Hand (1996).

For an overview of text analytics in R, refer to Feinerer, Hornik, and Meyer (2008), Feinerer and Hornik (2014a), and Feinerer (2014). Of special interest are books that deal with the overlap between linguistics and statistics with R (Baayen 2008; Johnson 2008; Gries 2013). The work of Gries (2009) is of special note as it shows how to work with document corpuses. Text analytics utilities are provided by Grothendieck (2014a, 2014b) and Wickham (2010, 2014b). Understanding regular expression coding in R can be of special value to anyone interested in doing text analytics (Friedl 2006; Wickham 2014b). Dictionary capabilities are provided through an interface to WordNet (Miller 1995; Fellbaum 1998; Feinerer 2012; Feinerer and Hornik 2014b). For fun, we have word clouds with utilities for plotting non-overlapping text in scatter plots (Fellows 2014a; Fellows 2014b). An example is shown in exhibit D.1 on page 285 of appendix D.

Exhibit 7.1 shows the R program for working with the movie taglines data. There was initial data preparation, parsing of the partially structured text data, which was accomplished by a script in appendix D, exhibit D.5 on page D.5. These parsed data are used to define the corpus. Text measures are computed on the corpus. Results are presented using multivariate methods for data visualization. The program draws on programming packages by Feinerer and Hornik (2014a), Wickham (2014b), Wickham and Chang (2014), Sarkar and Andrews (2014), Maechler (2014a), and Meyer (2014c).

Exhibit 7.1. *Text Analysis of Movie Taglines*

```
# Text Analysis of Movie Tag Lines (R)
# Note. Results from this program may differ from those published
#       in the book due to changes in the tm package.
#       The original analysis used the tm Dictionary() function,
#       which is no longer available in tm. This function has
#       been replaced by c(as.character()) to set the dictionary
#       as a character vector. Another necessary change concerns
#       the tolower() function, which must now be embedded within
#       the tm content_transformer() function.
#       The original analysis used the tm dissimilarity() function
#       to compute cosine similarities. This is no longer available,
#       so we use the proxy package and its dist() function.

# install these packages before bringing them in by library()
library(tm)  # text mining and document management
library(proxy)  # dissimilarity calculations by dist()
library(stringr)  # character manipulation with regular expressions
library(grid)  # grid graphics utilities
library(ggplot2)  # graphics
library(latticeExtra)  # package used for text horizon plot
library(wordcloud)  # provides utility for plotting non-overlapping text
library(cluster)  # cluster analysis
# R preliminaries to get the user-defined utilities for plotting
# place the plotting code file <R_utility_program_3.R>
# in your working directory and execute it by
#     source("R_utility_program_3.R")
# Or if you have the R binary file in your working directory, use
#     load("mtpa_split_plotting_utilities.Rdata")
load("mtpa_split_plotting_utilities.Rdata")

# standardization needed for text measures
standardize <- function(x) {(x - mean(x)) / sd(x)}

# to begin with the original movie taglines data, use the utility
# program for reading those data in and preparing the data for analysis
#     source(R_utility_program_7.R)
# otherwise read in the comma-delimited text file with the parsed data
# creating the movies data frame for analysis
movies = read.csv(file = "movie_tagline_data_parsed.csv", stringsAsFactors = FALSE)

# plot frequency of movies by year
pdf(file = "fig_text_movies_by_year_histogram.pdf", width = 11, height = 8.5)
ggplot.object <- ggplot(data = movies, aes(x = year)) +
  geom_histogram(binwidth = 1, fill = "blue", colour = "black") +
    labs(x = "Year of Release",
         y = "Number of Movies in Database") +
         coord_fixed(ratio = 1/50)
ggplot.print.with.margins(ggplot.object.name = ggplot.object,
  left.margin.pct=10, right.margin.pct=10,
  top.margin.pct=10,bottom.margin.pct=10)
dev.off()
```

```
# let us work with movies from 1974 to 2013
# creating an aggregate tagline_text collection for each year of interest
years.list <- 1974:2013
document.collection <- NULL  # initialize
for (index.for.year in seq(along=years.list)) {
  working.year.data.frame =
    subset(movies, subset = (year == years.list[index.for.year]))
  tagline_text <- NULL
  for(index.for.movie in seq(along = working.year.data.frame$movie))
    tagline_text <-
      paste(tagline_text, working.year.data.frame$tagline[index.for.movie])
  document <- PlainTextDocument(x = tagline_text, author = "Tom",
    description = paste("movie taglines for ",
    as.character(years.list[index.for.year]),sep = ""),
    id = paste("movies_",as.character(years.list[index.for.year]),sep=""),
    heading = "taglines",
    origin = "IMDb", language = "en_US",
    localmetadata = list(year = years.list[index.for.year]))

# give each created document a unique name
  if (years.list[index.for.year] == 1974) Y1974 <- document
  if (years.list[index.for.year] == 1975) Y1975 <- document
  if (years.list[index.for.year] == 1976) Y1976 <- document
  if (years.list[index.for.year] == 1977) Y1977 <- document
  if (years.list[index.for.year] == 1978) Y1978 <- document
  if (years.list[index.for.year] == 1979) Y1979 <- document
  if (years.list[index.for.year] == 1980) Y1980 <- document
  if (years.list[index.for.year] == 1981) Y1981 <- document
  if (years.list[index.for.year] == 1982) Y1982 <- document
  if (years.list[index.for.year] == 1983) Y1983 <- document
  if (years.list[index.for.year] == 1984) Y1984 <- document
  if (years.list[index.for.year] == 1985) Y1985 <- document
  if (years.list[index.for.year] == 1986) Y1986 <- document
  if (years.list[index.for.year] == 1987) Y1987 <- document
  if (years.list[index.for.year] == 1988) Y1988 <- document
  if (years.list[index.for.year] == 1989) Y1989 <- document
  if (years.list[index.for.year] == 1990) Y1990 <- document
  if (years.list[index.for.year] == 1991) Y1991 <- document
  if (years.list[index.for.year] == 1992) Y1992 <- document
  if (years.list[index.for.year] == 1993) Y1993 <- document
  if (years.list[index.for.year] == 1994) Y1994 <- document
  if (years.list[index.for.year] == 1995) Y1995 <- document
  if (years.list[index.for.year] == 1996) Y1996 <- document
  if (years.list[index.for.year] == 1997) Y1997 <- document
  if (years.list[index.for.year] == 1998) Y1998 <- document
  if (years.list[index.for.year] == 1999) Y1999 <- document
  if (years.list[index.for.year] == 2000) Y2000 <- document
  if (years.list[index.for.year] == 2001) Y2001 <- document
  if (years.list[index.for.year] == 2002) Y2002 <- document
  if (years.list[index.for.year] == 2003) Y2003 <- document
  if (years.list[index.for.year] == 2004) Y2004 <- document
  if (years.list[index.for.year] == 2005) Y2005 <- document
  if (years.list[index.for.year] == 2006) Y2006 <- document
```