Chapter 20
Natural Language Processing/Text Mining

As we have seen in the previous chapters, traditional statistical analyses and classical data modeling are applied to *relational data* where the observed information is represented by tables, vectors, arrays, tensors, or data-frames containing binary, categorical, original, or numerical values. Such representations provide incredible advantages (e.g., quick reference and de-reference of elements, search, discovery, and navigation), but also limit the scope of applications. Relational data objects are quite effective for managing information that is based only on existing attributes. However, when data science inference needs to utilize attributes that are not included in the relational model, alternative non-relational representations are necessary. For instance, imagine that our data object includes a free text feature (e.g., physician/nurse clinical notes, biospecimen samples) that contains information about medical condition, treatment or outcome. It's very difficult, or sometimes even impossible, to include the raw text into the automated data analytics, using classical procedures and statistical models available for relational datasets.

Natural Language Processing (NLP) and Text Mining (TM) refer to automated machine-driven algorithms for semantically mapping, extracting information, and understanding of (natural) human language. Sometimes, this involves extracting salient information from large amounts of unstructured text. To do so, we need to build semantic and syntactic mapping algorithms for effective processing of heavy text. Related to NLP/TM, the work we did in Chap. 8 showed a powerful text classifier using the naive Bayes algorithm.

In this Chapter, we will present more details about various text processing strategies in R. Specifically, we will present simulated and real examples of text processing and computing document term frequency (TF), inverse document frequency (IDF), cosine similarity transformation, and machine learning based sentiment analysis.

Live demos will show various NLP tasks directly in the browser intermediate stages of processing, as well as full text processing performing complete text analysis.

I. D. Dinov, *Data Science and Predictive Analytics*,
https://doi.org/10.1007/978-3-319-72347-1_20

20.1 A Simple NLP/TM Example

Text mining or text analytics (TM/TA) examines large volumes of unstructured text (corpus), aiming to extract new information, discover context, identify linguistic motifs, or transform the text into a structured data format leading to derived quantitative data that can be further analyzed. Natural language processing (NLP) is one example of a TM analytical technique. Whereas TM's goal is to discover relevant contextual information, which may be unknown, hidden, or obfuscated, NLP is focused on linguistic analysis that trains a machine to interpret voluminous textual content. To decipher the semantics and ambiguities in human-interpretable language, NLP employs automatic summarization, tagging, disambiguation, extraction of entities and relations, pattern recognition, and frequency analyses. As of 2018, the total amount of information generated by the human race exceeds 6 zettabytes ($1ZB = 10^{21} = 2^{70}$ bytes), which is projected to top $50ZB$ by 2020. The amount of data we obtain, and record, doubles every 12–14 months (Kryder's law). A small fraction of this massive information (<0.0001% or $<1PB = 10^{15}$ bytes) represents newly written or transcribed text, including code. However, it is impossible (cf. efficiency, time, resources) for humans to read, synthesize, interpret and react to all this information without direct assistance of TM/NLP. The information content in text could be substantially higher than that of other information media. Remember that "a picture may be worth a thousand words", yet, "a word may also be worth a thousand pictures". As an example, the simple sentence "*The data science and predictive analytics textbook includes 23 Chapters*" takes 63 bytes to store as text; however, a color image showing this as printed text could reach 10 megabytes (MB), and an HD video of a speaker reading the same sentence could easily surpass 50 MB. Text mining and natural language processing may be used to automatically analyze and interpret written, coded, or transcribed content to assess news, moods, emotions, clinical notes, and biosocial trends related to specific topics.

In general, text analysis protocols involve:

- Construction of a document-term matrix (DTM) from the input documents, vectorizing the text, e.g., creating a map of single words or `n-grams` into a vector space. That is, the *vectorizer is a function mapping terms to indices*.
- Application of a model-based statistical analysis or a model-free machine learning technique for prediction, clustering, classification, similarity search, network/sentiment analysis, or forecasting using the DTM. This step also includes tuning and internally validating the performance of the method.
- Application and evaluation of the technique to new data.

We are going to demonstrate this protocol with a very simple example. Figure 20.1 points to a separate online demo.

http://www.conversational-technologies.com/nldemos/nlDemos.html

Fig. 20.1 Live demo: Dynamic NLP demonstration

20.1.1 Define and Load the Unstructured-Text Documents

Let's create some documents we can use to illustrate the use of the `tm` package for text mining. The five documents below represent portions of the syllabi of five recent courses taught by the author:

- HS650: Data Science and Predictive Analytics (DSPA)
- Bootcamp: Predictive Big Data Analytics using R
- HS 853: Scientific Methods for Health Sciences: Special Topics
- HS851: Scientific Methods for Health Sciences: Applied Inference, and
- HS550: Scientific Methods for Health Sciences: Fundamentals

We import the syllabi into several separate segments represented as `documents`.

- As an *exercise*, try to use the `rvest::read_html` method to load in the five course syllabi directly from the course websites listed above.

```
doc1 <-"HS650: The Data Science and Predictive Analytics(DSPA) course (offered
as a massive open online course, MOOC, as well as a traditional University of
Michigan class) aims to build computational abilities, inferential thinking,
and practical skills for tackling core data scientific challenges. It explores
foundational concepts in data management, processing, statistical computing,
and dynamic visualization using modern programming tools and agile web-
services. Concepts, ideas, and protocols are illustrated through examples of
real observational, simulated and research-derived datasets. Some prior quanti-
tative experience in programming, calculus, statistics, mathematical models,
or linear algebra will be necessary. This open graduate course will provide a
general overview of the principles, concepts, techniques, tools and services
for managing, harmonizing, aggregating, preprocessing, modeling, analyzing and
interpreting large, multisource, incomplete, incongruent, and heterogeneous
data (Big Data). The focus will be to expose students to common challenges
related to handling Big Data and present the enormous opportunities and power
associated with our ability to interrogate such complex datasets, extract useful
information, derive knowledge, and provide actionable forecasting. Biomedical,
healthcare, and social datasets will provide context for addressing specific
driving challenges. Students will learn about modern data analytic techniques
and develop skills for importing and exporting, cleaning and fusing, modeling
and visualizing, analyzing and synthesizing complex datasets. The collaborative
design, implementation, sharing and community validation of high throughput
analytic workflows will be emphasized throughout the course."
```

```
doc2 < -"Bootcamp: A week-longintensive Bootcamp focused on methods, techniques, tools, services and resources for big healthcare and biomedical data analytics using the open-source statistical computing software R. Morning sessions (3 hrs) will be dedicated to methods and technologies and applications. Afternoon sessions (3 hrs) will be for group-based hands-on practice and team work. Commitment to attend the full week of instruction (morning sessions) and self-guided work (afternoon sessions) is required. Certificates of completion will be issued only to trainees with perfect attendance that complete all work. This hands-on intensive graduate course (Bootcamp) will providea general overview of the principles, concepts, techniques, tools and services for managing, harmonizing, aggregating, preprocessing, modeling, analyzing and interpreting large, multi-source, incomplete, incongruent, and heterogeneous data (Big Data). The focus will be to expose students to common challenges related to handling Big Data and present the enormous opportunities and power associated with our ability to interrogate such complex datasets, extract useful information, derive knowledge, and provide actionable forecasting. Biomedical, healthcare, and social datasets will provide context for addressing specific driving challenges. Students will learn about modern data analytic techniques and develop skills for importing and exporting, cleaning and fusing, modeling and visualizing, analyzing and synthesizing complex datasets. The collaborative design, implementation, sharing and community validation of high-throughput analytic workflows will be emphasized throughout the course."
doc3 <- "HS 853: This course covers a number of modern analytical methods for advanced healthcare research. Specific focus will be on reviewing and using innovative modeling, computational, analytic and visualization techniques to address concrete driving biomedical and healthcare applications. The course will cover the 5 dimensions of Big-Data (volume, complexity, multiple scales,-multiple sources, and incompleteness). HS853 is a 4 credit hour course (3 lectures + 1 lab/discussion). Students will learn how to conduct research, employ and report on recent advanced health sciences analytical methods; read, comprehend and present recent reports of innovative scientific methods; apply a broad range of health problems; experiment with real Big-Data.Topics Covered include: Foundations of R, Scientific Visualization, Review of Multivariate and Mixed Linear Models, Causality/Causal Inference and Structural Equation Models, Generalized Estimating Equations, PCOR/CER methods Heterogeneity of Treatment Effects, Big-Data, Big-Science, Internal statistical cross-validation, Missing data, Genotype-Environment-Phenotype, associations, Variable selection (regularized regression and controlled/knockoff filtering), medical imaging, Databases/registries, Meta-analyses, classification methods, Longitudinal data and time-series analysis, Geographic Information Systems(GIS), Psychometrics and Rasch measurement model analysis, Bayesian inference, and Network Analysis."

doc3 <- "HS 851: This course introduces students to applied inference methods in studies involving multiple variables. Specific methods that will be discussed include linear regression, analysis of variance, and different regression models. This course will emphasize the scientific formulation, analytical modeling, computational tools and applied statistical inference in diverse health-sciences problems. Data interrogation, modeling approaches, rigorous interpretation and inference will be emphasized throughout. HS851 is a 4 credit hour course (3 lectures + 1 lab/discussion). Students will learn how to:, Understand the commonly used statistical methods of published scientific papers, Conduct statistical calculations/analyses on available data, Use software tools to analyze specific case-studies data, Communicate advanced statistical concepts/techniques, Determine, explain and interpret assumptions and limitations. Topics Covered include Epidemiology, Correlation/SLR, and slope inference, 1-2 samples, ROC Curve, ANOVA, Non-parametric
```

```
inference, Cronbach's $\alpha$, Measurement Reliability/Validity, Survival
Analysis, Decision theory, CLT/LLNs - limiting results and misconceptions,
Association Tests, Bayesian Inference, PCA/ICA/Factor Analysis, Point/
Interval Estimation (CI) - MoM, MLE, Instrument performance Evaluation,
Study/Research Critiques, Common mistakes and misconceptions in using
probability and statistics, identifying potential assumption violations, and
avoiding them."

doc5 <- "HS550: This course provides students withan introduction to probabil-
ity reasoning and statistical inference. Students will learn theoretical
concepts and apply analytic skills for collecting, managing, modeling,
processing, interpreting and visualizing (mostly univariate) data. Students
will learn the basic probability modeling and statistical analysis methods and
acquire knowledge to read recently published health research publications.
HS550 is a 4 credithour course (3 lectures + 1 lab/discussion). Students will
learn how to: Apply data management strategies to sample data files, Carryout
statistical tests to answer common healthcare research questions using appro-
priate methods and software tools, Understand the core analytical data mode-
ling techniques and their appropriate use Examples of Topics Covered,
EDA/Charts, Ubiquitous variation, Parametric inference, Probability Theory,
Odds Ratio/Relative Risk, Distributions, Exploratory data analysis, Resam-
pling/Simulation, Design of Experiments, Intro to Epidemiology, Estimation,
Hypothesis testing, Experiments vs. Observational studies, Data management
(tables, streams, cloud, warehouses, DBs, arrays, binary, ASCII, handling,
mechanics), Power, sample-size, effect-size, sensitivity, specificity, Bias/-
Precision, Association vs. Causality, Rate-of-change, Clinical vs. Stat
significance, Statistical Independence Bayesian Rule."
```

20.1.2 Create a New VCorpus Object

The `VCorpus` object includes all the text and some meta-data (e.g., indexing) about the entire text.

```
docs<-c(doc1, doc2, doc3, doc4, doc5)

class(docs)

## [1] "character"
```

We can make a `VCorpus` object using `tm` package. To complete this task, we need to know the source type. Here, `docs` has a vector with "character" class, so we should use `VectorSource()`. If it is a dataframe, we should use `DataframeSource()` instead. `VCorpus()` creates a *volatile corpus*, which is the data type used by the `tm` package for text mining.

```
library(tm)

doc_corpus<-VCorpus(VectorSource(docs))
doc_corpus

## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 5

doc_corpus[[1]]$content

## [1] "HS650: The Data Science and Predictive Analytics (DSPA) course (offe
red as a massive open online course, MOOC, as well as a traditional Universi
ty of Michigan class) aims to build computational abilities, inferential thi
nking, … throughout the course."
```

This generates a list containing the information for the five documents we have created. Now we can apply `tm_map()` function on this object to preprocess the text. The goal is to automatically interpret the text and output more succinct information.

20.1.3 To-Lower Case Transformation

The text itself contains upper case letters as well as lower case letters. The first thing to do is to convert all characters to lower case.

```
doc_corpus<-tm_map(doc_corpus, tolower)
doc_corpus[[1]]

## [1] "hs650: the data science and predictive analytics (dspa) course (offe
red as a massive open online course, mooc, as well as a traditional universi
ty of michigan class) … community validation of high-throughput analytic wor
kflows will be emphasized throughout the course."
```

20.1.4 Text Pre-processing

Remove Stopwords

These documents contains a lot of "stopwords", or common words, that have important semantic meaning but low analytic value. We can remove these by the following command.

```
stopwords("english")

##   [1] "i"          "me"         "my"         "myself"     "we"
##   [6] "our"        "ours"       "ourselves"  "you"        "your"
##  [11] "yours"      "yourself"   "yourselves" "he"         "him"
##  [16] "his"        "himself"    "she"        "her"        "hers"
…
## [171] "so"         "than"       "too"        "very"
```

```
doc_corpus<-tm_map(doc_corpus, removeWords, stopwords("english"))
doc_corpus[[1]]

## [1] "hs650:  data science  predictive analytics (dspa) course (offered
massive open online course, mooc,  well  traditional university  michigan c
lass) aims  build …, sharing  community validation  high-throughput analytic
workflows will  emphasized throughout  course."
```

We removed all the stopwords specified in the `stopwords("english")` list. You can always make your own stopword list and just use `doc_corpus<-tm_map(doc_corpus, removeWords, your_own_words_list)` to apply this list.

From the output of `doc1` we notice the removal of stopwords creates extra blank spaces. Thus, the next step would be to remove them.

```
doc_corpus<-tm_map(doc_corpus, stripWhitespace)
doc_corpus[[1]]

## [1] "hs650: data science predictive analytics (dspa) course (offered mass
ive open online course, mooc, well traditional university michigan class) ai
ms build computational abilities, …, sharing community validation high-throu
ghput analytic workflows will emphasized throughout course."
```

Remove Punctuation

Now we notice the irrelevant punctuation in the text, which can be removed by using a combination of `tm_map()` and `removePunctuation()` functions.

```
doc_corpus<-tm_map(doc_corpus, removePunctuation)
doc_corpus[[2]]

## [1] "bootcamp weeklong intensive bootcamp focused methods techniques tool
s services resources big healthcare biomedical data analytics using opensour
ce statistical computing software r morning sessions 3 hrs … collaborative d
esign implementation sharing community validation highthroughput analytic wo
rkflows will emphasized throughout course"
```

The above `tm_map` commands changed the structure of our `doc_corpus` object. We may apply `PlainTextDocument` function if we need to convert it back to the original format.

```
doc_corpus<-tm_map(doc_corpus, PlainTextDocument)
```

Stemming: Removal of Plurals and Action Suffixes

Let's inspect the first three documents. We notice that there are some words ending with "ing", "es", or "s".

```
doc_corpus[[1]]$content

## [1] "hs650 data science predictive analytics dspa course offered massive
open online course mooc well traditional university michigan class aims buil
d computational abilities inferential … validation highthroughput analytic w
orkflows will emphasized throughout course"

doc_corpus[[2]]$content

## [1] "bootcamp weeklong intensive bootcamp focused methods techniques tool
s services resources big healthcare biomedical data analytics using opensour
ce statistical computing software r morning sessions 3 …  design implementat
ion sharing community validation highthroughput analytic workflows will emph
asized throughout course"

doc_corpus[[3]]$content

## [1] "hs 853 course covers number modern analytical methods advanced healt
hcare research specific focus will reviewing using innovative modeling compu
tational analytic visualization … information systems gis psychometrics rasc
h measurement model analysis mcmc sampling bayesian inference network analys
is"
```

If we have multiple terms that only differ in their endings (e.g., past, present, present-perfect-continuous tense), the algorithm will treat them differently because it does not understand language semantics, the way a human would. To make things easier for the computer, we can delete these endings by "stemming" documents. Remember to load the package `SnowballC` before using the function `stemDocument()`. The earliest stemmer was written by Julie Beth Lovins in 1968, which had great influence on all subsequent work. Currently, one of the most popular stemming approaches was proposed by Martin Porter and is used in `stemDocument()`, and you can read more on Porter algorithm online.

```
# install.packages("SnowballC")
library(SnowballC)
doc_corpus<-tm_map(doc_corpus, stemDocument)
doc_corpus[[1]]$content

## [1] "hs650 data scienc predict analyt dspa cours offer massiv open onlin
cours mooc well tradit univers michigan class aim build comput abil inferent
i think practic skill tackl core data scientif … fuse model visual analyz sy
nthes complex dataset collabor design implement share communiti valid highth
roughput analyt workflow will emphas throughout cours"
```

This stemming process has to be done after the `PlainTextDocument` function because `stemDocument` can only be applied to plain text.

20.1.5 Bags of Words

It's very useful to be able to tokenize text documents into `n-grams`, sequences of words, e.g., a `2-gram` represents two-word phrases that appear together in order. This allows us to form bags of words and extract information about word ordering.

The *bag of words model* is a common way to represent documents in matrix form based on their term frequencies (TFs). We can construct an $n \times t$ document-term matrix (DTM), where n is the number of documents, and t is the number of unique terms. Each column in the DTM represents a unique term. For instance, the $(i,j)^{th}$ cell represents how many of term j are present in document i.

The basic bag of words model is invariant to ordering of the words within a document. Once we compute the DTM, we can use machine learning techniques to interpret the derived signature information contained in the resulting matrices.

20.1.6 Document Term Matrix

Now that the `doc_corpus` object is quite clean, we can make a document-term matrix to explore all the terms in the five initial documents. The document term matrix includes dummy variables that tell us if a specific term appears in a specific document.

```
doc_dtm<-TermDocumentMatrix(doc_corpus)
doc_dtm

## <<TermDocumentMatrix (terms: 329, documents: 5)>>
## Non-/sparse entries: 540/1105
## Sparsity           : 67%
## Maximal term length: 27
## Weighting          : term frequency (tf)
```

The summary of document term matrix is informative. We have 329 different terms in the five documents. There are 540 non-zero and 1,105 sparse entries. Thus, the sparsity is $\frac{1105}{(540+1105)} \approx 67\%$, which measures the term sparsity across all documents. A high sparsity means that the terms are not repeated often among different documents.

Recall that we applied `PlainTextDocument` function to your `doc_corpus` object. This removed all document meta data. To relabel the documents in the document term matrix, we can use the following commands:

```
doc_dtm$dimnames$Docs<-as.character(1:5)
inspect(doc_dtm)

## <<TermDocumentMatrix (terms: 329, documents: 5)>>
## Non-/sparse entries: 540/1105
## Sparsity           : 67%
## Maximal term length: 27
## Weighting          : term frequency (tf)
## Sample             :
##          Docs
## Terms     1 2 3 4 5
##   analyt  3 3 3 1 2
##   cours   4 2 3 3 2
##   data    7 5 2 3 6
##   infer   0 0 2 6 2
##   method  0 2 5 3 2
##   model   3 2 4 3 3
##   statist 2 1 1 5 4
##   student 2 2 1 2 4
##   use     2 2 1 3 2
##   will    6 8 3 4 3
```

We might want to find and report the frequent terms using this document term matrix.

```
findFreqTerms(doc_dtm, lowfreq = 2)
```

```
##   [1] "abil"          "action"        "address"       "advanc"
##   [5] "afternoon"     "aggreg"        "analysi"       "analyt"
##   [9] "analyz"        "appli"         "applic"        "appropri"
##  [13] "associ"        "assumpt"       "attend"        "bayesian"
##  [17] "big"           "bigdata"       "biomed"        "bootcamp"
##  [21] "challeng"      "clean"         "collabor"      "common"
##  [25] "communiti"     "complet"       "complex"       "comput"
##  [29] "concept"       "conduct"       "context"       "core"
##  [33] "cours"         "cover"         "credit"        "data"
##  [37] "dataset"       "deriv"         "design"        "develop"
##  [41] "drive"         "emphas"        "enorm"         "epidemiolog"
##  [45] "equat"         "estim"         "exampl"        "experi"
##  [49] "export"        "expos"         "extract"       "focus"
##  [53] "forecast"      "foundat"       "fuse"          "general"
##  [57] "graduat"       "hand"          "handl"         "harmon"
##  [61] "health"        "healthcar"     "heterogen"     "highthroughput"
##  [65] "hour"          "hrs"           "hs550"         "implement"
##  [69] "import"        "includ"        "incomplet"     "incongru"
##  [73] "infer"         "inform"        "innov"         "intens"
##  [77] "interpret"     "interrog"      "knowledg"      "labdiscuss"
##  [81] "larg"          "learn"         "lectur"        "limit"
##  [85] "linear"        "manag"         "measur"        "method"
##  [89] "misconcept"    "model"         "modern"        "morn"
##  [93] "multipl"       "multisourc"    "observ"        "open"
##  [97] "opportun"      "overview"      "power"         "practic"
## [101] "preprocess"    "present"       "principl"      "probabl"
## [105] "problem"       "process"       "program"       "provid"
## [109] "publish"       "read"          "real"          "recent"
## [113] "regress"       "relat"         "report"        "research"
## [117] "review"        "sampl"         "scienc"        "scientif"
## [121] "servic"        "session"       "share"         "skill"
## [125] "social"        "softwar"       "specif"        "statist"
## [129] "student"       "studi"         "synthes"       "techniqu"
## [133] "test"          "theori"        "throughout"    "tool"
## [137] "topic"         "understand"    "use"           "valid"
## [141] "variabl"       "visual"        "will"          "work"
## [145] "workflow"
```

This gives us the terms that appear in at least two documents. High-frequency terms like `comput`, `statist`, `model`, `healthcar`, `learn` make perfect sense to be included as these syllabi describe courses that cover modeling, statistical and computational methods with applications to health sciences.

The `tm` package also provides the correlation between terms. Here is a mechanism to determine the words that are highly correlated with `statist`, ($\rho(statist, ?) \geq 0.8$).

```
findAssocs(doc_dtm, "statist", corlimit = 0.8)
```

```
## $statist
## epidemiolog     publish       studi      theori  understand       appli
##        0.95        0.95        0.95        0.95        0.95        0.83
##        test
##        0.80
```

20.2 Case-Study: Job Ranking

Let's explore some real datasets. First, we will import the 2011 USA Jobs Ranking Dataset from SOCR data archive.

```
library(rvest)

wiki_url <- read_html("http://wiki.socr.umich.edu/index.php/SOCR_Data_2011_US_JobsRanking")
html_nodes(wiki_url, "#content")

## {xml_nodeset (1)}
## [1] <div id="content" class="mw-body-primary" role="main">\n\t<a id="top
...

job <- html_table(html_nodes(wiki_url, "table")[[1]])
head(job)

##   Index                Job_Title Overall_Score Average_Income(USD)
## 1     1        Software_Engineer            60                87140
## 2     2            Mathematician            73                94178
## 3     3                  Actuary           123                87204
## 4     4              Statistician           129                73208
## 5     5 Computer_Systems_Analyst           147                77153
## 6     6            Meteorologist           175                85210
##   Work_Environment Stress_Level Stress_Category Physical_Demand
## 1           150.00        10.40               1            5.00
## 2            89.72        12.78               1            3.97
## 3           179.44        16.04               1            3.97
## 4            89.52        14.08               1            3.95
## 5            90.78        16.53               1            5.08
## 6           179.64        15.10               1            6.98
##   Hiring_Potential
## 1            27.40
## 2            19.78
## 3            17.04
## 4            11.08
## 5            15.53
## 6            12.10
##
Description
## 1 Researches_designs_develops_and_maintains_software_systems_along_with_hardware_development_for_medical_scientific_and_industrial_purposes
## 2 Applies_mathematical_theories_and_formulas_to_teach_or_solve_problems_in_a_business_educational_or_industrial_climate
## 3  Interprets_statistics_to_determine_probabilities_of_accidents_sickness_and_death_and_loss_of_property_from_theft_and_natural_disasters
## 4   Tabulates_analyzes_and_interprets_the_numeric_results_of_experiments_and_surveys
## 5   Plans_and_develops_computer_systems_for_businesses_and_scientific_institutions
## 6   Studies_the_physical_characteristics_motions_and_processes_of_the_earth's_atmosphere
```

Note that low indices represent jobs that in 2011 were highly desirable. Thus, in 2011, the most desirable job among the top 200 common jobs would be Software

Engineer. The aim of our case study is to explore the difference between the top 30 desirable jobs and the last 100 jobs in the list.

We will go through the same procedure as we did for the simple course syllabi example. The documents we will be using include the `Description` column (a vector) in the dataset.

20.2.1 Step 1: Make a VCorpus Object

```
jobCorpus<-VCorpus(VectorSource(job[, 10]))
```

20.2.2 Step 2: Clean the VCorpus Object

```
jobCorpus<-tm_map(jobCorpus, tolower)
for(j in seq(jobCorpus)){
  jobCorpus[[j]]<-gsub("_", " ", jobCorpus[[j]])
}
```

Here we used a loop to substitute "_" with blank space. This is because when we use `removePunctuation`, all the underline characters will disappear and there will be no separation between terms. In this situation, `gsub` will be the best choice to use.

```
jobCorpus<-tm_map(jobCorpus, removeWords, stopwords("english"))
jobCorpus<-tm_map(jobCorpus, removePunctuation)
jobCorpus<-tm_map(jobCorpus, stripWhitespace)
jobCorpus<-tm_map(jobCorpus, PlainTextDocument)
jobCorpus<-tm_map(jobCorpus, stemDocument)
```

20.2.3 Step 3: Build the Document Term Matrix

The Document Term Matrix (DTM) objects (`tm::DocumentTermMatrix`) contains a sparse term-document matrix, or document-term matrix, and attribute weights of the matrix.

First, make sure that we got a clean VCorpus object.

```
jobCorpus[[1]]$content
## [1] "research design develop maintain softwar system along hardwar develo
p medic scientif industri purpos"
```

Then, we can start to build the DTM and reassign the labels to the `Docs`.

```
dtm<-DocumentTermMatrix(jobCorpus)
dtm
```

```
## <<DocumentTermMatrix (documents: 200, terms: 846)>>
## Non-/sparse entries: 1818/167382
## Sparsity           : 99%
## Maximal term length: 15
## Weighting          : term frequency (tf)
```

```
dtm$dimnames$Docs<-as.character(1:200)
inspect(dtm[1:10, 1:10])
```

```
## <<DocumentTermMatrix (documents: 10, terms: 10)>>
## Non-/sparse entries: 2/98
## Sparsity           : 98%
## Maximal term length: 7
## Weighting          : term frequency (tf)
## Sample             :
##     Terms
## Docs 16wheel abnorm access accid accord account accur achiev act activ
##   1        0      0      0     0      0       0     0      0   0     0
##   10       0      0      0     0      0       0     0      0   0     0
##   2        0      0      0     0      0       0     0      0   0     0
##   3        0      0      0     1      0       0     0      0   0     0
##   4        0      0      0     0      0       0     0      0   0     0
##   5        0      0      0     0      0       0     0      0   0     0
##   6        0      0      0     0      0       0     0      0   0     0
##   7        0      0      0     0      0       0     0      0   0     0
##   8        0      0      0     0      1       0     0      0   0     0
##   9        0      0      0     0      0       0     0      0   0     0
```

Let's subset the `dtm` into the top 30 jobs and the bottom 100 jobs.

```
dtm_top30<-dtm[1:30, ]
dtm_bot100<-dtm[101:200, ]
dtm_top30
```

```
## <<DocumentTermMatrix (documents: 30, terms: 846)>>
## Non-/sparse entries: 293/25087
## Sparsity           : 99%
## Maximal term length: 15
## Weighting          : term frequency (tf)
```

```
dtm_bot100
```

```
## <<DocumentTermMatrix (documents: 100, terms: 846)>>
## Non-/sparse entries: 870/83730
## Sparsity           : 99%
## Maximal term length: 15
## Weighting          : term frequency (tf)
```

In this case, since the sparsity is very high, we can try to remove some words that rarely appear in the job descriptions.

```
dtms_top30<-removeSparseTerms(dtm_top30, 0.90)
dtms_top30

## <<DocumentTermMatrix (documents: 30, terms: 19)>>
## Non-/sparse entries: 70/500
## Sparsity           : 88%
## Maximal term length: 10
## Weighting          : term frequency (tf)

dtms_bot100<-removeSparseTerms(dtm_bot100, 0.94)
dtms_bot100

## <<DocumentTermMatrix (documents: 100, terms: 14)>>
## Non-/sparse entries: 122/1278
## Sparsity           : 91%
## Maximal term length: 10
## Weighting          : term frequency (tf)
```

Now, instead of 846 terms, we only have 19 that appear in the top 30 job descriptions (JDs) and 14 that appear in the bottom 100 JDs.

Similar to what we did in **Chap. 8**, visualization of the terms-world clouds may be accomplished by combining the `tm` with `wordcloud` packages. First, we can count the term frequencies in the two document term matrices (Fig. 20.2).

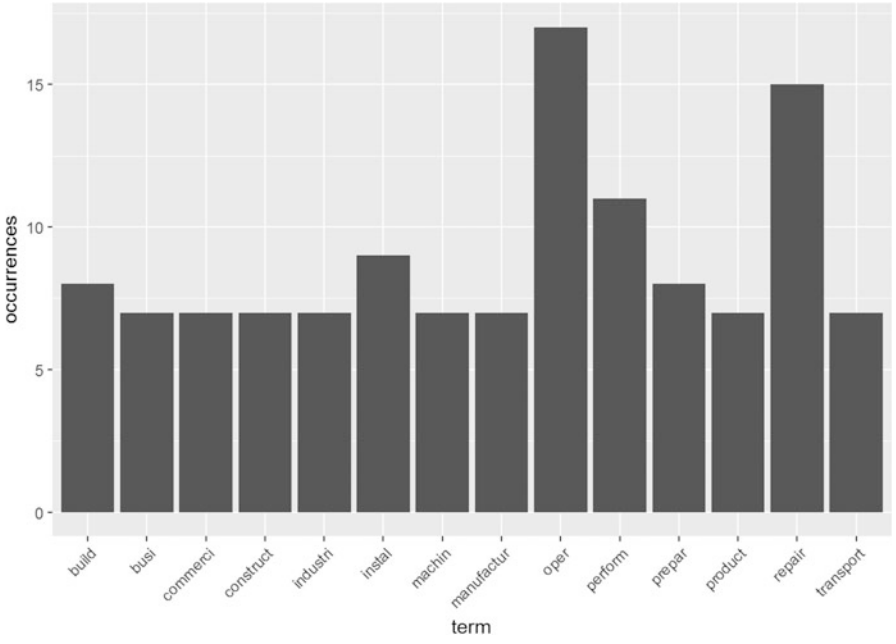

Fig. 20.2 Frequency plot of commonly occurring terms (bottom 100 jobs)

```
# Calculate the cumulative frequencies of words across documents and sort:
freq1<-sort(colSums(as.matrix(dtms_top30)), decreasing=T)
freq1

##    develop     assist      natur      studi     analyz    concern
##          6          5          5          5          4          4
##   individu   industri     physic       plan       busi     inform
##          4          4          4          4          3          3
##   institut    problem   research   scientif     theori  treatment
##          3          3          3          3          3          3
## understand
##          3

freq2<-sort(colSums(as.matrix(dtms_bot100)), decreasing=T)
freq2

##       oper     repair    perform     instal      build     prepar
##         17         15         11          9          8          8
##       busi   commerci  construct   industri     machin manufactur
##          7          7          7          7          7          7
##    product  transport
##          7          7
# Plot
wf=data.frame(term=names(freq2), occurrences=freq2)
library(ggplot2)

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:NLP':
##
##     annotate

p <- ggplot(subset(wf, freq2>2), aes(term, occurrences))
p <- p + geom_bar(stat="identity")
p <- p + theme(axis.text.x=element_text(angle=45, hjust=1))
p
```

Then, we apply the `wordcloud` function to the `freq` dataset (Figs. 20.3 and 20.4).

Fig. 20.3 Wordle plot of the frequently occurring terms in the top-30 jobs

Fig. 20.4 The appearance of the wordle plot may be customized as shown here for theh bottom-100 jobs

```
library(wordcloud)

set.seed(123)
wordcloud(names(freq1), freq1)
```

```
# Color code the frequencies using an appropriate color map:
# Sequential palettes names include:
# Blues BuGn BuPu GnBu Greens Greys Oranges OrRd PuBu PuBuGn PuRd Purples Rd
Pu Reds YlGn YlGnBu YlOrBr YlOrRd

# Diverging palettes include
# BrBG PiYG PRGn PuOr RdBu RdGy RdYlBu RdYlGn Spectral
wordcloud(names(freq2), freq2, min.freq=5, colors=brewer.pal(6, "Spectral"))
```

It is apparent that the top 30 jobs focus more on research or discovery of new things, and include frequent keywords like “study”, “nature”, and “analyze.” The bottom 100 jobs more focused of operating on existing objects, with frequent keywords like “operation”, “repair”, and “perform”.

20.2.4 Area Under the ROC Curve

In Chap. 14, we talked about the ROC curve. We can use document term matrices to build classifiers and use the area under the ROC curve (AUC) to evaluate those classifiers. Assume that we want to predict whether a job ranks in the top 30, i.e., the most desired jobs. The first task would be to create an indicator of high rank jobs (top 30). We can use the `ifelse()` function that we are already familiar with.

```
job$highrank<-ifelse(job$Index<30, 1, 0)
```

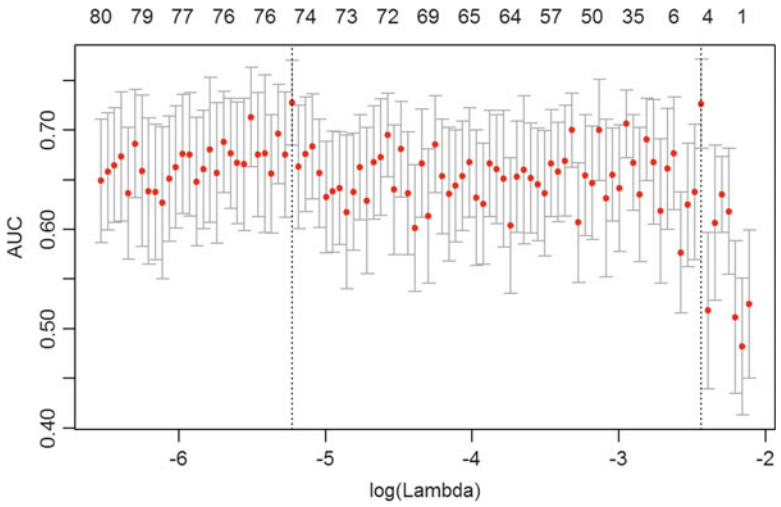

Fig. 20.5 The area under the curve (AUC) measures the performance of the cross-validated LASSO-regularized model of job-ranking against the magnitude of the regularization parameter (bottom axis), and the efficacy of the model selection, i.e., number of non-trivial coefficients (top axis). The vertical dash lines suggest an optimal range for the penalty term and the number of coefficients, see Chap. 18

Next we load the `glmnet` package to help us build the model and draw the corresponding graphs.

```
#install.packages("glmnet")
library(glmnet)
```

The function we will be using is the `cv.glmnet`, `cv` stands for cross-validation. Since the `highrank` variable is binary, we specify the option `family = 'binomial'`. Also, we want to use 10-fold CV method for re-sampling (Fig. 20.5).

```
set.seed(25)
fit <- cv.glmnet(x = as.matrix(dtm), y = job[['highrank']],
                 family = 'binomial',
                 # lasso penalty
                 alpha = 1,
                 # interested in the area under ROC curve
                 type.measure = "auc",
                 # 10-fold cross-validation
                 nfolds = 10,
                 # high value is less accurate, but has faster training
                 thresh = 1e-3,
                 # again lower number of iterations for faster training
                 maxit = 1e3)
plot(fit)
```

```
print(paste("max AUC =", round(max(fit$cvm), 4)))
## [1] "max AUC = 0.7276"
```

Here, x is a matrix and y is the response variable. The last line of code helps us select the best AUC among all models. The resulting $AUC \sim 0.73$ represents a relatively good prediction model for this small sample size.

20.3 TF-IDF

To enhance the performance of the DTM matrix, we introduce **TF-IDF (term frequency – inverse document frequency)**. Unlike pure frequency, TF-IDF measures the relative importance of a term. If a term appears in almost every document, the term will be considered common with a small weight. Alternatively, the rare terms would be considered more informational.

20.3.1 Term Frequency (TF)

TF is the ratio $\frac{\text{a term's occurrences in a document}}{\text{the number of occurrences of the most frequent word within the same document}}$. Symbolically,

$$TF(t,d) = \frac{f_d(t)}{\max\limits_{w \in d} f_d(w)}.$$

20.3.2 Inverse Document Frequency (IDF)

The **TF** definition may allow high scores for irrelevant words that naturally show up often in a long text, even after triaging common words in a prior preprocessing step. The **IDF** attempts to rectify that. **IDF** represents the inverse of the share of the documents in which the regarded term can be found. The lower the number of documents containing the term, relative to the size of the corpus, the higher the term factor.

IDF involves a logarithm function, to temper the effective scoring penalty of showing up in two documents, which othersize may be too extreme. Typically, the IDF for a term found in just one document is twice the IDF for another term found in two docs. The ln() function rectifies this bias of ranking in favor of rare terms, even if

the TF-factor may be high. It is rather unlikely that a term's relevance is only high in one doc and not all others.

$$IDF(t, D) = \ln\left(\frac{\mid D \mid}{\mid \{d \in D : t \in d\} \mid}\right).$$

20.3.3 TF-IDF

Both TF and IDF yield high scores for highly relevant terms. TF relies on local information (search over d), whereas IDF incorporates a more global perspective (search over D). The product $TF \times IDF$, gives the classical **TF-IDF** formula. However, alternative expressions may be formulated to get other univariate expressions using alternative weights for TF and IDF.

$$TF\_IDF(t, d, D) = TF(t, d) \times IDF(t, D).$$

An example of an alternative TF-IDF metric can be defined by:

$$TF\_IDF'(t, d, D) = \frac{IDF(t, D)}{\mid D \mid} + TF\_IDF(t, d, D).$$

Let's make another DTM with TF-IDF weights and compare the differences between the *unweighted* and *weighted* DTM.

```
dtm.tfidf<-DocumentTermMatrix(jobCorpus, control = list(weighting=weightTfId
f))
dtm.tfidf

## <<DocumentTermMatrix (documents: 200, terms: 846)>>
## Non-/sparse entries: 1818/167382
## Sparsity           : 99%
## Maximal term length: 15
## Weighting          : term frequency - inverse document frequency (normali
zed) (tf-idf)

dtm.tfidf$dimnames$Docs <- as.character(1:200)
inspect(dtm.tfidf[1:9, 1:10])

## <<DocumentTermMatrix (documents: 9, terms: 10)>>
## Non-/sparse entries: 2/88
## Sparsity           : 98%
## Maximal term length: 7
## Weighting          : term frequency - inverse document frequency (normali
zed) (tf-idf)
## Sample             :
##     Terms
## Docs 16wheel abnorm access     accid    accord account accur achiev act
##    1       0      0      0 0.0000000 0.0000000       0     0      0   0
##    2       0      0      0 0.0000000 0.0000000       0     0      0   0
```

```
##    3       0      0      0 0.5536547 0.0000000       0     0      0   0
##    4       0      0      0 0.0000000 0.0000000       0     0      0   0
##    5       0      0      0 0.0000000 0.0000000       0     0      0   0
##    6       0      0      0 0.0000000 0.0000000       0     0      0   0
##    7       0      0      0 0.0000000 0.0000000       0     0      0   0
##    8       0      0      0 0.0000000 0.4321928       0     0      0   0
##    9       0      0      0 0.0000000 0.0000000       0     0      0   0
##     Terms
## Docs activ
##    1     0
##    2     0
##    3     0
##    4     0
##    5     0
##    6     0
##    7     0
##    8     0
##    9     0
```

```
inspect(dtm[1:9, 1:10])
```

```
## <<DocumentTermMatrix (documents: 9, terms: 10)>>
## Non-/sparse entries: 2/88
## Sparsity           : 98%
## Maximal term length: 7
## Weighting          : term frequency (tf)
## Sample             :
##     Terms
## Docs 16wheel abnorm access accid accord account accur achiev act activ
##    1       0      0      0     0      0       0     0      0   0     0
##    2       0      0      0     0      0       0     0      0   0     0
##    3       0      0      0     1      0       0     0      0   0     0
##    4       0      0      0     0      0       0     0      0   0     0
##    5       0      0      0     0      0       0     0      0   0     0
##    6       0      0      0     0      0       0     0      0   0     0
##    7       0      0      0     0      0       0     0      0   0     0
##    8       0      0      0     0      1       0     0      0   0     0
##    9       0      0      0     0      0       0     0      0   0     0
```

An inspections of the two different DTMs suggests that TF-IDF is not only counting the frequency but also assigning different weights to each term according to the importance of the term. Next, we are going to fit another model with this new DTM (`dtm.tfidf`) (Fig. 20.6).

```
set.seed(2)
fit1 <- cv.glmnet(x = as.matrix(dtm.tfidf), y = job[['highrank']],
                  family = 'binomial',
                  # lasso penalty
                  alpha = 1,
                  # interested in the area under ROC curve
                  type.measure = "auc",
                  # 10-fold cross-validation
                  nfolds = 10,
                  # high value is less accurate, but has faster training
                  thresh = 1e-3,
                  # again lower number of iterations for faster training
                  maxit = 1e3)
plot(fit1)
```

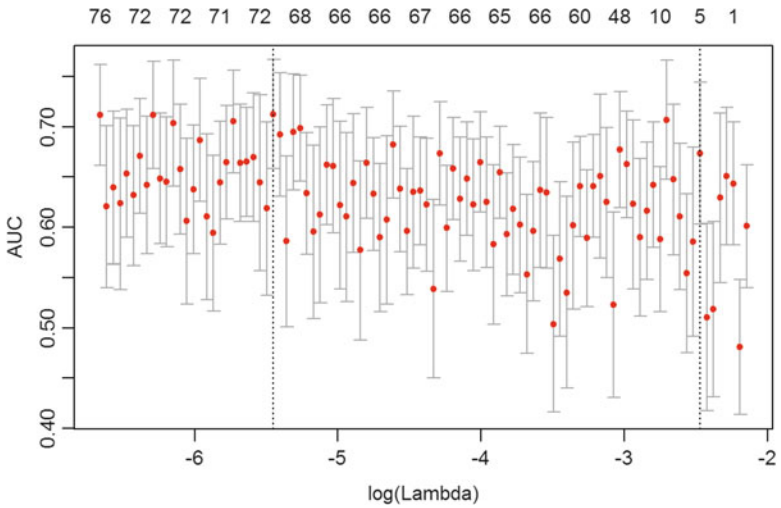

Fig. 20.6 AUC-based performance of the cross-validated LASSO-regularized model of job-ranking based on the new DTM (`dtm.tfidf`), see Fig. 20.5 and Chap. 18

```
print(paste("max AUC =", round(max(fit1$cvm), 4)))

## [1] "max AUC = 0.7125"
```

This output is about the same as the previous jobs ranking prediction classifier (based on the unweighted DTM). Due to random sampling, each run of the protocols may generate slightly different results. The idea behind using TF-IDF is that one would expect to get more unbiased estimates of word importance. If the document includes stopwords, like "the" or "one", the DTM may distort the results, but TF-IDF may resolve some of these problems.

Next, we can report a more intuitive representation of the job ranking prediction reflecting the agreement of the binary (top-30 or not) classification between the real labels and the predicted labels. Notice that this applies only to the training data itself.

```
# Binarize the LASSO probability prediction
preffit1 <- predict(fit1, newx=as.matrix(dtm.tfidf), s="lambda.min", type =
"class")
binPredfit1 <- ifelse(preffit1<0.5, 0, 1)
table(binPredfit1, job[['highrank']])

##
## binPredfit1   0   1
##           0 171   0
##           1   0  29
```

Let's try to predict the job ranking of several new (testing or validation) job descriptions (JDs). There are many job descriptions provided online that we can extract text from to predict the job ranking of the corresponding positions. Trying

several alternative job categories, e.g., some high-tech or fin-tech, and some manufacturing or construction jobs, may provide some intuition to the power of the jobs-classifier we built. Below, we will compare the JDs for the positions of *accountant*, *attorney*, and *machinist*.

```
# install.packages("text2vec"); install.packages("data.table")
library(text2vec)
library(data.table)

# Choose the JD for a PUBLIC ACCOUNTANTS 1430 (https://www.bls.gov/ocs/ocsjobde.htm)
xTestAccountant <- "Performs professional auditing work in a public accounting firm. Work requires at least a bachelor's degree in accounting. Participates in or conducts audits to ascertain the fairness of financial representations made by client companies. May also assist the client in improving accounting procedures and operations. Examines financial reports, accounting records, and related documents and practices of clients. Determines whether all important matters have been disclosed and whether procedures are consistent and conform to acceptable practices. Samples and tests transactions, internal controls, and other elements of the accounting system(s) as needed to render the accounting firm's final written opinion. As an entry level public accountant, serves as a junior member of an audit team. Receives classroom and on-the-job training to provide practical experience in applying the principles, theories, and concepts of accounting and auditing to specific situations. (Positions held by trainee public accountants with advanced degrees, such as MBA's are excluded at this level.) Complete instructions are furnished and work is reviewed to verify its accuracy, conformance with required procedures and instructions, and usefulness in facilitating the accountant's professional growth. Any technical problems not covered by instructions are brought to the attention of a superior. Carries out basic audit tests and procedures, such as: verifying reports against source accounts and records; reconciling bank and other accounts; and examining cash receipts and disbursements, payroll records, requisitions, receiving reports, and other accounting documents in detail to ascertain that transactions are properly supported and recorded. Prepares selected portions of audit working papers"

xTestAttorney <-"Performs consultation, advisory and/or trail work and carries out the legal processes necessary to effect the rights, privileges, and obligations of the organization. The work performed requires completion of law school with an L.L.B. degree or J.D. degree and admission to the bar. Responsibilities or functions include one or more of the following or comparable duties:
1. Preparing and reviewing various legal instruments and documents, such as contracts, leases, licenses, purchases, sales, real estate, etc.;
2. Acting as agent of the organization in its transactions;
3. Examining material (e.g., advertisements, publications, etc.) for legal implications; advising officials of proposed legislation which might affect the organization;
4. Applying for patents, copyrights, or registration of the organization's products, processes, devices, and trademarks; advising whether to initiate or defend law suits;
5. Conducting pretrial preparations; defending the organization in lawsuits;
```

```
6. Prosecuting criminal cases for a local or state government or defending
the general public (for example, public defenders and attorneys rendering
legal services to students); or
7. Advising officials on tax matters, government regulations, and/or legal
rights.
Attorney jobs are matched at one of six levels according to two factors:
1. Difficulty level of legal work; and
2. Responsibility level of job.
Attorney jobs which meet the above definitions are to be classified and
coded in accordance with a chart available upon request.
Legal questions are characterized by: facts that are well-established;
clearly applicable legal precedents; and matters not of substantial
importance to the organization. (Usually relatively limited sums of money,
e.g., a few thousand dollars, are involved.)
a. legal investigation, negotiation, and research preparatory to defending
the organization in potential or actual lawsuits involving alleged
negligencewhere the facts can be firmly established and there are precedent
cases directly applicable to the situation;
b. searching case reports, legal documents, periodicals, textbooks, and other
legal references, and preparing draft opinions on employee compensation or
benefit questions where there is a substantial amount of clearly applicable
statutory, regulatory, and case material;
c. drawing up contracts and other legal documents in connection with real
property transactions requiring the development of detailed information but
not involving serious questions regarding titles to property or other major
factual or legal issues.
d. preparing routine criminal cases for trial when the legal or factual
issues are relatively straight forward and the impact of the case is limited;
and
e. advising public defendants in regard to routine criminal charges or compl-
aints and representing such defendants in court when legal alternatives and
facts are relatively clear and the impact of the outcome is limited primarily
to the defendant.

Legal work is regularly difficult by reason of one or more of the following:
the absence of clear and directly applicable legal precedents; the different
possible interpretations that can be placed on the facts, the laws, or the
precedents involved; the substantial importance of the legal matters to the
organization (e.g., sums as large as $100,000 are generally directly or indir-
ectly involved); or the matter is being strongly pressed or contested in
formal proceedings or in negotiations by the individuals, corporations, or
government agencies involved.
a. advising on the legal implications of advertising representations when the
facts supporting the representations and the applicable precedent cases are
subject to different interpretations;
b. reviewing and advising on the implications of new or revised laws
affecting the organization;
c. presenting the organization's defense in court in a negligence lawsuit
which is strongly pressed by counsel for an organized group;
d. providing legal counsel on tax questions complicated by the absence of
precedent decisions that are directly applicable to the organization's
situation;
e. preparing and prosecuting criminal cases when the facts of the cases are
complex or difficult to determine or the outcome will have a significant
impact within the jurisdiction; and
```

```
f. advising and representing public defendants in all phases of criminal
proceedings when the facts of the case are complex or difficult to determine,
complex or unsettled legal issues are involved, or the prosecutorial
jurisdiction devotes substantial resources to obtaining a conviction."

xTestMachinist <- "Produces replacement parts and new parts in making repairs
of metal parts of mechanical equipment. Work involves most of the following:
interpreting written instructions and specifications; planning and laying
out of work; using a variety of machinist's handtools and precision measuring
instruments; setting up and operating standard machine tools; shaping of
metal parts to close tolerances; making standard shop computations relating
to dimensions of work, tooling, feeds, and speeds of machining; knowledge of
the working properties of the common metals; selecting standard materials,
parts, and equipment required for this work; and fitting and assembling parts
into mechanical equipment. In general, the machinist's work normally requires
a rounded training in machine-shop practice usually acquired through a formal
apprenticeship or equivalent training and experience. Industrial machinery
repairer. Repairs machinery or mechanical equipment. Work involves most of
the following: examining machines and mechanical equipment to diagnose source
of trouble; dismantling or partly dismantling machines and performing repairs
that mainly involve the use of handtools in scraping and fitting parts;
replacing broken or defective parts with items obtained from stock; ordering
the production of a replacement part by a machine shop or sending the machine
to a machine shop for major repairs; preparing written specifications for
major repairs or for the production of parts ordered from machine shops;
reassembling machines; and making all necessary adjustments for operation.
In general, the work of a machinery maintenance mechanic requires rounded
training and experience usually acquired through a formal apprenticeship or
equivalent training and experience. Excluded from this classification are
workers whose primary duties involve setting up or adjusting machines. Vehicle
and mobile equipment mechanics and repairers. Repairs, rebuilds, or overhauls
major assemblies of internal combustion automobiles, buses, trucks, or tract
tractors. Work involves most of the following: Diagnosing the source of trouble
and determining the extent of repairs required; replacing worn or broken parts
such as piston rings, bearings, or other engine parts; grinding and adjusting
valves; rebuilding carburetors; overhauling transmissions; and repairing
fuel injection, lighting, and ignition systems. In general, the work of the
motor vehicle mechanic requires rounded training and experience usually acqu-
ired through a formal apprenticeship or equivalent training and experience"

testJDs <- c(xTestAccountant, xTestAttorney, xTestMachinist)

# define the preprocessing (tolower case) function
# preproc_fun = tolower

# define the tokenization function
token_fun = text2vec::word_tokenizer

# loop to substitute "_" with blank space
for(j in seq(job[, 10])){
  job[j, 10] <- gsub("_", " ", job[j, 10])
}

# iterator for Job training and testing JDs
iter_Jobs = itoken(job[, 10],
              preprocessor = preproc_fun,
              tokenizer = token_fun,
              progressbar = TRUE)
iter_testJDs = itoken(testJDs,
              preprocessor = preproc_fun,
              tokenizer = token_fun,
              progressbar = TRUE)
```

```
jobs_Vocab= create_vocabulary(iter_Jobs,stopwords=tm::stopwords("english"),
ngram = c(1L, 2L))

jobsVectorizer = vocab_vectorizer(jobs_Vocab)

dtm_jobsTrain = create_dtm(iter_Jobs, jobsVectorizer)

dtm_testJDs = create_dtm(iter_testJDs, jobsVectorizer)

dim(dtm_jobsTrain); dim(dtm_testJDs)
## [1]  200 2675

## [1]    3 2675

set.seed(2)
fit1 <- cv.glmnet(x = as.matrix(dtm_jobsTrain), y = job[['highrank']],
                  family = 'binomial',
                  # lasso penalty
                  alpha = 1,
                  # interested in the area under ROC curve
                  type.measure = "auc",
                  # 10-fold cross-validation
                  nfolds = 10,
                  # high value is less accurate, but has faster training
                  thresh = 1e-3,
                  # again lower number of iterations for faster training
                  maxit = 1e3)
print(paste("max AUC =", round(max(fit1$cvm), 4)))

## [1] "max AUC = 0.7934"
```

Note that we somewhat improved the $AUC \sim 0.79$. Below, we will assess the JD predictive model using the three out of bag job descriptions (Fig. 20.7).

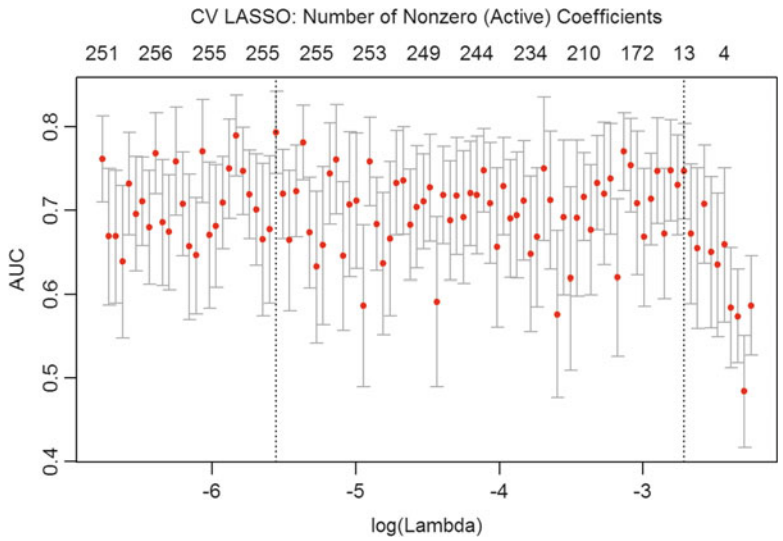

Fig. 20.7 AUC-based performance of the cross-validated LASSO-regularized model of job-ranking based on `dtm_jobsTrain`, see Figs. 20.5 and 20.6

```
plot(fit1)
# plot(fit1, xvar="lambda", label="TRUE")
mtext("CV LASSO: Number of Nonzero (Active) Coefficients", side=3, line=2.5)

predTestJDs <- predict(fit1, s = fit1$lambda.1se,
                       newx = dtm_testJDs, type="response"); predTestJDs

##           1
## 1 0.2153011
## 2 0.2200925
## 3 0.1257575

predTrainJDs <- predict(fit1, s = fit1$lambda.1se, newx = dtm_jobsTrain, typ
e="response"); predTrainJDs

##             1
## 1   0.3050636
## 2   0.4118190
## 3   0.1288656
## 4   0.1493051
## 5   0.6432706
## 6   0.1257575
## 7   0.1257575
## 8   0.2561290
## 9   0.3866247
## 10  0.1262752
…
## 196 0.1257575
## 197 0.1257575
## 198 0.1257575
## 199 0.1257575
## 200 0.1257575

# Type can be: "link", "response", "coefficients", "class", "nonzero"
```

The output of the predictions shows that:

- On the *training data*, the predicted probabilities rapidly decrease with the indexing of the jobs, corresponding to the *overall job ranking* (highly ranked/desired jobs are listed on the top).
- On the three *testing job description data* (accountant, attorney, and machinist), there is a clear ranking difference between the machinist and the other two professions.

Also see the discussion in Chap. 18 about the different *types of predictions* that can be generated as outputs of `cv.glmnet` regularized forecasting methods.

20.4 Cosine Similarity

As we mentioned above, text data are often *transformed* in terms of Term Frequency-Inverse Document Frequency (TF-IDF), which offers a better input than the raw frequencies for many text-mining methods. An alternative transformation can be represented as a different distance measure such as the *cosine distance*, which is defined by:

$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\|_2 \|B\|_2},$$

where θ represents the angle between the pair of vectors A and B in the Euclidean space spanned by the DTM matrix (Fig. 20.8).

```
cos_dist = function(mat){
  numer = tcrossprod(mat)
  denom1 = sqrt(apply(mat, 1, crossprod))
  denom2 = sqrt(apply(mat, 1, crossprod))
  1 - numer / outer(denom1,denom2)
}

dist_cos = cos_dist(as.matrix(dtm))

set.seed(2000)
fit_cos <- cv.glmnet(x = dist_cos, y = job[['highrank']],
                     family = 'binomial',
                     # lasso penalty
                     alpha = 1,
```

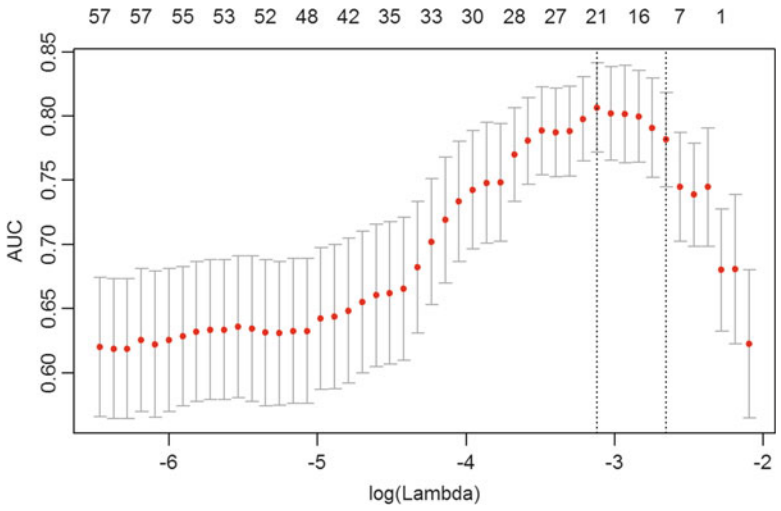

Fig. 20.8 AUC-based performance of the cross-validated LASSO-regularized model of job-ranking based on cosine-similarity distance (`dist_cos`), see Figs. 20.5, 20.6, and 20.7

```
                # interested in the area under ROC curve
                type.measure = "auc",
                # 10-fold cross-validation
                nfolds = 10,
                # high value is less accurate, but has faster training
                thresh = 1e-3,
                # again lower number of iterations for faster training
                maxit = 1e3)
plot(fit_cos)
```

```
print(paste("max AUC =", round(max(fit_cos$cvm), 4)))
```

```
## [1] "max AUC = 0.8065"
```

The AUC now is greater than 0.8, which is a pretty good result; even better than what we obtained from DTM or TF-IDF. This suggests that our machine "understanding" of the textual content, i.e., the natural language processing, leads to a more acceptable content classifier.

20.5 Sentiment Analysis

Let's use the `text2vec::movie_review` dataset, which consists of 5,000 movie reviews dichotomized as `positive` or `negative`. In the subsequent predictive analytics, this *sentiment* will represent our output feature:

$$Y = Sentiment = \begin{cases} 0, & negative \\ 1, & positive \end{cases}.$$

20.5.1 Data Preprocessing

The `data.table` package will also be used for some data manipulation. Let's start with splitting the data into *training* and *testing* sets.

```
# install.packages("text2vec"); install.packages("data.table")
library(text2vec)
library(data.table)

# Load the movie reviews data
data("movie_review")

# coerce the movie reviews data to a data.table (DT) object
setDT(movie_review)

# create a key for the movie-reviews data table
setkey(movie_review, id)
```

```
# View the data
# View(movie_review)
head(movie_review); dim(movie_review); colnames(movie_review)
```

```
##         id sentiment
## 1: 10000_8         1
## 2: 10001_4         0
## 3: 10004_3         0
## 4: 10004_8         1
## 5: 10006_4         0
## 6: 10008_7         1
##
review
## 1: Homelessness (or Houselessness as George Carlin stated) has been an is
sue for years but never a plan to help those on the street that were once co
nsidered human who did everything from going to school ... Maybe they should
give it to the homeless instead of using it like Monopoly money.<br /><br />
Or maybe this film will inspire you to help others.
## 2:
This film lacked something I couldn't put my finger on at first: charisma on
the part of the leading actress. This inevitably translated to lack of chemi
stry when she shared the screen with her leading man. Even the romantic scen
es came across as being merely the actors at play … I was disappointed in th
is movie. But, don't forget it was nominated for an Oscar, so judge for your
self.
## 3:
\\"It appears that many critics find the idea of a Woody Allen drama unpalat
able.\\" And for good reason: they are unbearably wooden and pretentious imi
tations of Bergman. And let's … \\"ripping off\\" Hitchcock in his suspense/
horror films? In Robin Wood's view, it's a strange form of cultural snobbery
. I would have to agree with that.
## 4:
This isn't the comedic Robin Williams, nor is it the quirky/insane Robin Wil
liams of recent thriller fame. This is a hybrid of the classic drama without
over-dramatization, mixed with Robin's new love of the thriller. But this is
n't a thriller, per se … <br /><br />All in all, it's worth a watch, though
it's definitely not Friday/Saturday night fare.<br /><br />It rates a 7.7/10
from...<br /><br />the Fiend :.
## 5:
I don't know who to blame, the timid writers or the clueless director. It se
emed to be one of those movies where so much was paid to the stars (Angie, C
harlie, Denise, Rosanna and Jon) … If they were only looking for laughs why
not cast Whoopi Goldberg and Judy Tenuta instead? This was so predictable I
was surprised to find that the director wasn't a five year old. What a waste
, not just for the viewers but for the actors as well.
## 6:
You know, Robin Williams, God bless him, is constantly shooting himself in t
he foot lately with all these dumb comedies he has done this decade (with pe
rhaps the exception of \\"Death To Smoochy … It's incredible that there is a
t least one woman in this world who is like this, and it's scary too.<br /><
br />This is a good, dark film that I highly recommend. Be prepared to be un
settled, though, because this movie leaves you with a strange feeling at the
end.
```

```
## [1] 5000    3
```

```
## [1] "id"        "sentiment" "review"
```

```
# Generate 80-20% training-testing split of the reviews
all_ids = movie_review$id
set.seed(1234)
train_ids = sample(all_ids, 5000*0.8)
test_ids = setdiff(all_ids, train_ids)
train = movie_review[train_ids, ]
test = movie_review[test_ids, ]
```

Next, we will vectorize the reviews by creating terms to *termID* mappings. Note that terms may include arbitrary *n-grams*, not just single words. The set of reviews will be represented as a sparse matrix, with rows and columns corresponding to reviews/reviewers and terms, respectively. This vectorization may be accomplished in several alternative ways, e.g., by using the corpus vocabulary, feature hashing, etc.

The vocabulary-based DTM, created by the `create_vocabulary()` function, relies on all unique terms from all reviews, where each term has a unique ID. In this example, we will create the review vocabulary using an *iterator* construct abstracting the input details and enabling *in memory* processing of the (training) data by chunks.

```
# define the text preprocessing
# either a simple (tolower case) function
preproc_fun = tolower

# or a more elaborate "cleaning" function
preproc_fun = function(x)                    # text data
{ require("tm")
  x  =  gsub("<.*?>", " ", x)                # regex removing HTML tags
  x  =  iconv(x, "latin1", "ASCII", sub="") # remove non-ASCII characters
  x  =  gsub("[^[:alnum:]]", " ", x)     # remove non-alpha-numeric values
  x  =  tolower(x)                          # convert to lower case characters
  # x  =  removeNumbers(x)                  # removing numbers
  x  =  stripWhitespace(x)                  # removing white space
  x  =  gsub("^\\s+|\\s+$", "", x)  # remove leading and trailing white space
  return(x)
}

# define the tokenization function
token_fun = word_tokenizer

# iterator for both training and testing sets
iter_train = itoken(train$review,
             preprocessor = preproc_fun,
             tokenizer = token_fun,
             ids = train$id,
             progressbar = TRUE)
```

```
iter_test = itoken(test$review,
             preprocessor = preproc_fun,
             tokenizer = token_fun,
             ids = test$id,
             progressbar = TRUE)
reviewVocab = create_vocabulary(iter_train)

# report the head and tail of the reviewVocab
reviewVocab

## Number of docs: 4000
## 0 stopwords:  ...
## ngram_min = 1; ngram_max = 1
## Vocabulary:
##                terms terms_counts doc_counts
##     1:       lowlife            1          1
##     2:         sorin            1          1
##     3:         ewell            1          1
##     4:   negligence            1          1
##     5:      stribor            1          1
##    ---
## 35661: landscaping            1          1
## 35662:        bikes            1          1
## 35663:       primer            1          1
## 35664:      loosely           26         25
## 35665:      cycling            1          1
```

Next, we can compute the *document term matrix* (DTM).

```
reviewVectorizer = vocab_vectorizer(reviewVocab)
t0 = Sys.time()
dtm_train = create_dtm(iter_train, reviewVectorizer)

dtm_test = create_dtm(iter_test, reviewVectorizer)

t1 = Sys.time()
print(difftime(t1, t0, units = 'sec'))

## Time difference of 3.844368 secs

# check the DTM dimensions
dim(dtm_train); dim(dtm_test)

## [1]  4000 35665

## [1]  1000 35665

# confirm that the training data review DTM dimensions are consistent
# with training review IDs, i.e., #rows = number of documents, and
# #columns = number of unique terms (n-grams), dim(dtm_train)[[2]]
identical(rownames(dtm_train), train$id)

## [1] TRUE
```

20.5.2 NLP/TM Analytics

We can now fit statistical models or derive machine learning model-free predictions. Let's start by using `glmnet()` to fit a *logit model* with LASSO (L_1) regularization and 10-fold cross-validation, see Chap. 18 (Fig. 20.9).

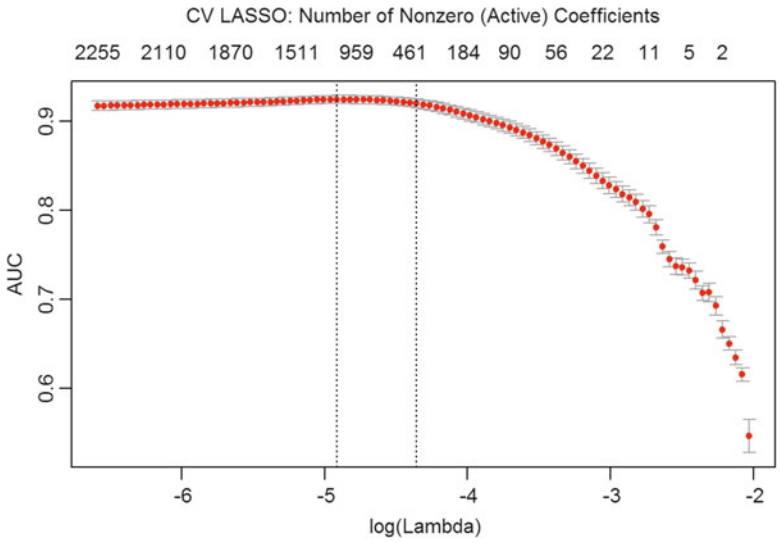

Fig. 20.9 AUC-based performance of the cross-validated LASSO-regularized model of movie sentiment analysis training data (`dtm_train`), see Fig. 20.8

```
library(glmnet)
nFolds = 10
t0 = Sys.time()
glmnet_classifier = cv.glmnet(x = dtm_train, y = train[['sentiment']],
        family = "binomial",
        # LASSO L1 penalty
        alpha = 1,
        # interested in the area under ROC curve or MSE
        type.measure = "auc",
        # n-fold internal (training data) stats cross-validation
        nfolds = nFolds,
        # threshold: high value is less accurate / faster training
        thresh = 1e-2,
        # again lower number of iterations for faster training
        maxit = 1e3
      )

Lambda.best <- glmnet_classifier$lambda.min
Lambda.best

## [1] 0.007344319
# report execution time
t1 = Sys.time()
print(difftime(t1, t0, units = 'sec'))

## Time difference of 5.923289 secs

# some prediciton plots
plot(glmnet_classifier)
# plot(glmnet_classifier, xvar="lambda", label="TRUE")
mtext("CV LASSO: Number of Nonzero (Active) Coefficients", side=3, line=2.5)
```

Now let's look at external validation, i.e., testing the model on the independent 20% of the reviews we kept aside. The performance of the binary prediction (binary

sentiment analysis of these movie reviews) on the test data is roughly the same as we had from the internal statistical 10-fold cross-validation.

```
# report the mean internal cross-validated error
print(paste("max AUC =", round(max(glmnet_classifier$cvm), 4)))
## [1] "max AUC = 0.9246"

# report TESTING data prediction accuracy
xTest = dtm_test
yTest = test[['sentiment']]
predLASSO <- predict(glmnet_classifier,
              s = glmnet_classifier$lambda.1se, newx = xTest)
testMSE_LASSO <- mean((predLASSO - yTest)^2); testMSE_LASSO

## [1] 2.869523

# Binarize the LASSO probabiliuty prediction
binPredLASSO <- ifelse(predLASSO<0.5, 0, 1)
table(binPredLASSO, yTest)
```

```
##             yTest
## binPredLASSO   0   1
##            0 455 152
##            1  40 353

# and testing data AUC
glmnet:::auc(yTest, predLASSO)

## [1] 0.9175598

# report the top 20 negative and positive predictive terms
summary(predLASSO)

##        1
##  Min.   :-12.80392
##  1st Qu.: -0.94586
##  Median :  0.14755
##  Mean   : -0.07101
##  3rd Qu.:  1.01894
##  Max.   :  6.60888

sort(predict.cv.glmnet(glmnet_classifier, s = lambda.best, type = "coefficie
nts"))[1:20]

## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient

##  [1] -4.752272 -2.199304 -1.987171 -1.966585 -1.902009 -1.866655 -1.84496
6
##  [8] -1.750693 -1.518148 -1.502966 -1.436081 -1.405963 -1.349566 -1.34285
6
## [15] -1.320218 -1.283414 -1.270231 -1.257663 -1.242869 -1.209449

rev(sort(predict.cv.glmnet(glmnet_classifier, s = lambda.best, type = "coeff
icients")))[1:20]

## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient
##  [1] 2.559487 2.416333 2.101371 1.913529 1.899846 1.684176 1.600367
##  [8] 1.530320 1.519663 1.435103 1.430446 1.376056 1.343108 1.309902
## [15] 1.300156 1.287921 1.131859 1.078685 1.074059 1.015887
```

The (external) prediction performance, measured by AUC, on the testing data is about the same as the internal 10-fold stats cross-validation we reported above.

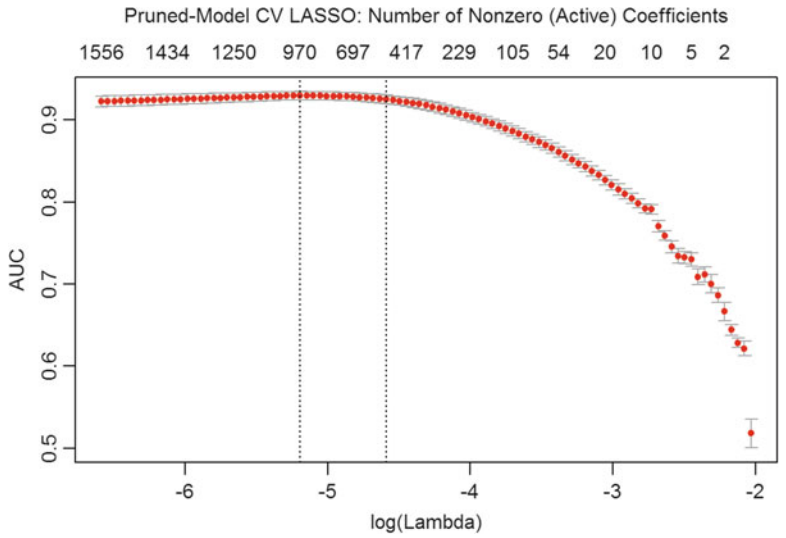

Fig. 20.10 AUC-based performance of the pruned cross-validated LASSO-regularized model of movie sentiment analysis (`glmnet_prunedClassifier`), see Fig. 20.9

20.5.3 Prediction Optimization

Earlier, we saw that we can also prune the vocabulary and perhaps improve prediction performance, e.g., by removing non-salient terms like stopwords and by using *n-grams* instead of single words (Fig. 20.10).

```
reviewVocab = create_vocabulary(iter_train,
stopwords=tm::stopwords("english"), ngram = c(1L, 2L))

prunedReviewVocab = prune_vocabulary(reviewVocab,
                term_count_min = 10,
                doc_proportion_max = 0.5,
                doc_proportion_min = 0.001)
prunedVectorizer = vocab_vectorizer(prunedReviewVocab)

t0 = Sys.time()
dtm_train = create_dtm(iter_train, prunedVectorizer)

dtm_test = create_dtm(iter_test, prunedVectorizer)

t1 = Sys.time()
print(difftime(t1, t0, units = 'sec'))

## Time difference of 3.778152 secs
```

Next, let's refit the model and report its performance. Would there be an improvement in the prediction accuracy?

```
glmnet_prunedClassifier=cv.glmnet(x=dtm_train,
        y=train[['sentiment']],
        family = "binomial",
        # LASSO L1 penalty
        alpha = 1,
        # interested in the area under ROC curve or MSE
        type.measure = "auc",
        # n-fold internal (training data) stats cross-validation
        nfolds = nFolds,
        # threshold: high value is less accurate / faster training
        thresh = 1e-2,
        # again lower number of iterations for faster training
        maxit = 1e3
      )

lambda.best <- glmnet_prunedClassifier$lambda.min
lambda.best

## [1] 0.005555708
# report execution time
t1 = Sys.time()
print(difftime(t1, t0, units = 'sec'))

## Time difference of 6.978195 secs

# some prediction plots
plot(glmnet_prunedClassifier)
mtext("Pruned-Model CV LASSO: Number of Nonzero (Active) Coefficients",
side=3, line=2.5)

# report the mean internal cross-validated error
print(paste("max AUC =", round(max(glmnet_prunedClassifier$cvm), 4)))

## [1] "max AUC = 0.9295"

# report TESTING data prediction accuracy
xTest = dtm_test
yTest = test[['sentiment']]
predLASSO = predict(glmnet_prunedClassifier,
          dtm_test, type = 'response')[,1]

testMSE_LASSO <- mean((predLASSO - yTest)^2); testMSE_LASSO

## [1] 0.1194583

# Binarize the LASSO probabiliuty prediction
binPredLASSO <- ifelse(predLASSO<0.5, 0, 1)
table(binPredLASSO, yTest)
##             yTest
## binPredLASSO   0   1
##            0 416  60
##            1  79 445

# and testing data AUC
glmnet:::auc(yTest, predLASSO)

## [1] 0.9252405

# report the top 20 negative and positive predictive terms
summary(predLASSO)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000014 0.2518000 0.5176000 0.5026000 0.7604000 0.9995000
```

```
sort(predict.cv.glmnet(glmnet_classifier, s = lambda.best, type = "coefficie
nts"))[1:20]
```

```
## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient
```

```
## [1] -5.695082 -2.774694 -2.756099 -2.540456 -2.508213 -2.474586 -2.432767
## [8] -2.429874 -1.999731 -1.941299 -1.934803 -1.929788 -1.819220 -1.774936
##[15] -1.765978 -1.737596 -1.717957 -1.661592 -1.611752 -1.599558
```

```
rev(sort(predict.cv.glmnet(glmnet_classifier, s = lambda.best, type = "coeff
icients")))[1:20]
```

```
## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient
```

```
##  [1] 3.276620 2.695083 2.575524 2.436630 2.366057 2.139067 2.087892
##  [8] 2.027113 1.980694 1.894909 1.839621 1.777573 1.743082 1.599660
## [15] 1.579711 1.569817 1.533461 1.509555 1.453862 1.425065
```

```
# Binarize the LASSO probability prediction
# and construct an approximate confusion matrix
binPredLASSO <- ifelse(predLASSO<0.5, 0, 1)
table(binPredLASSO, yTest)
```

```
##             yTest
## binPredLASSO   0   1
##            0 416  60
##            1  79 445
```

Using `n-grams` improved a bit the sentiment prediction model.

Try these NLP techniques to other data like:

- MIMIC-III, a freely accessible critical care database. Johnson AEW, Pollard TJ, Shen L, Lehman L, Feng M, Ghassemi M, Moody B, Szolovits P, Celi LA, and Mark RG. Scientific Data (2016). DOI: 10.1038/sdata.2016.35. Available from: http://www.nature.com/articles/sdata201635
- Other data from the list of our Case-Studies.
- Your own free text.

20.6 Assignment: 20. Natural Language Processing/Text Mining

20.6.1 Mining Twitter Data

Use these R Data Mining Twitter data to apply NLP/TM methods and investigate the Twitter corpus.

- Construct a VCorpus object
- Clean the VCorpus object

- Build document term matrix (DTM)
- Compute the TF-IDF(term frequency - inverse document frequency
- Use the DTM to construct a wordcloud.

20.6.2 Mining Cancer Clinical Notes

Use Head and Neck Cancer Medication Data to apply NLP/TM methods and investigate the corpus. You have already seen this data in Chap. 8; now we can go a step further.

- Use `MEDICATION_SUMMARY` to construct a VCorpus object.
- Clean the VCorpus object.
- Build the document term matrix (DTM).
- Add a column to indicate early and later stage according to `seer_stage` (refer to Chap. 8).
- Use the DTM to construct a word cloud for early stage, later stage and whole.
- Interpret according to the word cloud.
- Compute the TF-IDF (Term Frequency - Inverse Document Frequency).
- Apply LASSO on the unweighted and weighted DTM respectively and evaluate the results according to AUC.
- Try cosine similarity transformation, apply LASSO and compare the result.
- Use other measures such as "class" for `cv.glmnet()`.
- Does it appear that these classifiers understand well human language?

References

Kumar, E. (2011) *Natural Language Processing*, I. K. International Pvt Ltd, ISBN 9380578776, 9789380578774.

Kao, A, Poteet, SR (eds.) (2007) Natural Language Processing and Text Mining, Springer Science & Business Media, ISBN 1846287545, 9781846287541.

https://github.com/kbenoit/spacyr

https://tartarus.org/martin/PorterStemmer/