TFIDF is efficient in that the calculations are simple and straightforward, and it does not require knowledge of the underlying meanings of the text. But this approach also reveals little of the inter-document or intra-document statistical structure. The next section shows how topic models can address this shortcoming of TFIDF.

9.6 Categorizing Documents by Topics

With the reviews collected and represented, the data science team at ACME wants to categorize the reviews by topics. As discussed earlier in the chapter, a topic consists of a cluster of words that frequently occur together and share the same theme.

The topics of a document are not as straightforward as they might initially appear. Consider these two reviews:

1. The bPhone5x has coverage everywhere. It's much less flaky than my old bPhone4G.
2. While I love ACME's bPhone series, I've been quite disappointed by the bEbook. The text is illegible, and it makes even my old NBook look blazingly fast.

Is the first review about bPhone5x or bPhone4G? Is the second review about bPhone, bEbook, or NBook? For machines, these questions can be difficult to answer.

Intuitively, if a review is talking about bPhone5x, the term `bPhone5x` and related terms (such as `phone` and `ACME`) are likely to appear frequently. A document typically consists of multiple themes running through the text in different proportions—for example, 30% on a topic related to `phones`, 15% on a topic related to `appearance`, 10% on a topic related to `shipping`, 5% on a topic related to `service`, and so on.

Document grouping can be achieved with clustering methods such as *k*-means clustering [24] or classification methods such as support vector machines [25], *k*-nearest neighbors [26], or naïve Bayes [27]. However, a more feasible and prevalent approach is to use ***topic modeling***. Topic modeling provides tools to automatically organize, search, understand, and summarize from vast amounts of information. ***Topic models*** [28, 29] are statistical models that examine words from a set of documents, determine the themes over the text, and discover how the themes are associated or change over time. The process of topic modeling can be simplified to the following.

1. Uncover the hidden topical patterns within a corpus.
2. Annotate documents according to these topics.
3. Use annotations to organize, search, and summarize texts.

A ***topic*** is formally defined as a distribution over a fixed vocabulary of words [29]. Different topics would have different distributions over the same vocabulary. A topic can be viewed as a cluster of words with related meanings, and each word has a corresponding weight inside this topic. Note that a word from the vocabulary can reside in multiple topics with different weights. Topic models do not necessarily require prior knowledge of the texts. The topics can emerge solely based on analyzing the text.

The simplest topic model is ***latent Dirichlet allocation*** (LDA) [29], a generative probabilistic model of a corpus proposed by David M. Blei and two other researchers. In generative probabilistic modeling, data

is treated as the result of a generative process that includes hidden variables. LDA assumes that there is a fixed vocabulary of words, and the number of the latent topics is predefined and remains constant. LDA assumes that each latent topic follows a Dirichlet distribution [30] over the vocabulary, and each document is represented as a random mixture of latent topics.

Figure 9-4 illustrates the intuitions behind LDA. The left side of the figure shows four topics built from a corpus, where each topic contains a list of the most important words from the vocabulary. The four example topics are related to problem, policy, neural, and report. For each document, a distribution over the topics is chosen, as shown in the histogram on the right. Next, a topic assignment is picked for each word in the document, and the word from the corresponding topic (colored discs) is chosen. In reality, only the documents (as shown in the middle of the figure) are available. The goal of LDA is to infer the underlying topics, topic proportions, and topic assignments for every document.

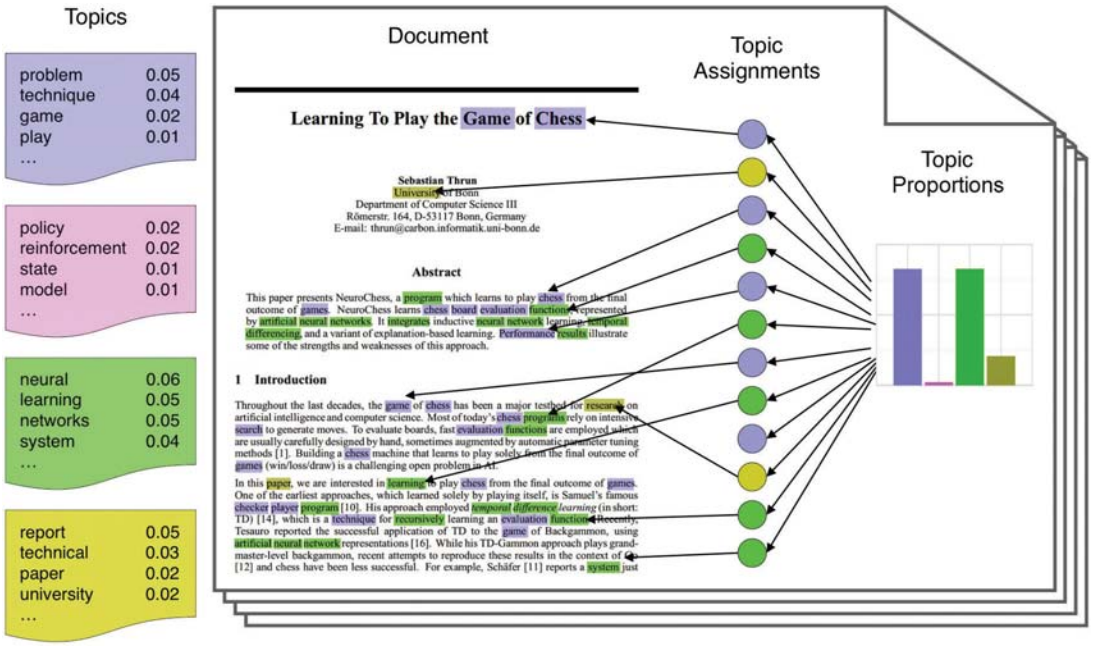

FIGURE 9-4 *The intuitions behind LDA*

The reader can refer to the original paper [29] for the mathematical detail of LDA. Basically, LDA can be viewed as a case of hierarchical Bayesian estimation with a posterior distribution to group data such as documents with similar topics.

Many programming tools provide software packages that can perform LDA over datasets. R comes with an `lda` package [31] that has built-in functions and sample datasets. The `lda` package was developed by David M. Blei's research group [32]. Figure 9-5 shows the distributions of ten topics on nine scientific documents randomly drawn from the `cora` dataset of the `lda` package. The `cora` dataset is a collection of 2,410 scientific documents extracted from the Cora search engine [33].

FIGURE 9-5 *Distributions of ten topics over nine scientific documents from the Cora dataset*

The code that follows shows how to generate a graph similar to Figure 9-5 using R and add-on packages such as `lda` and `ggplot`.

```
require("ggplot2")
require("reshape2")
require("lda")

# load documents and vocabulary
data(cora.documents)
data(cora.vocab)

theme_set(theme_bw())

# Number of topic clusters to display
K <- 10

# Number of documents to display
N <- 9
```

```
result <- lda.collapsed.gibbs.sampler(cora.documents,
                                      K,  ## Num clusters
                                      cora.vocab,
                                      25,  ## Num iterations
                                      0.1,
                                      0.1,
                                      compute.log.likelihood=TRUE)

# Get the top words in the cluster
top.words <- top.topic.words(result$topics, 5, by.score=TRUE)

# build topic proportions
topic.props <- t(result$document_sums) / colSums(result$document_sums)

document.samples <- sample(1:dim(topic.props)[1], N)
topic.props <- topic.props[document.samples,]

topic.props[is.na(topic.props)] <-  1 / K

colnames(topic.props) <- apply(top.words, 2, paste, collapse=" ")

topic.props.df <- melt(cbind(data.frame(topic.props),
                             document=factor(1:N)),
                       variable.name="topic",
                       id.vars = "document")

qplot(topic, value*100, fill=topic, stat="identity",
      ylab="proportion (%)", data=topic.props.df,
      geom="histogram") +
  theme(axis.text.x = element_text(angle=0, hjust=1, size=12)) +
  coord_flip() +
  facet_wrap(~ document, ncol=3)
```

Topic models can be used in document modeling, document classification, and collaborative filtering [29]. Topic models not only can be applied to textual data, they can also help annotate images. Just as a document can be considered a collection of topics, images can be considered a collection of image features.

9.7 Determining Sentiments

In addition to the TFIDF and topic models, the Data Science team may want to identify the sentiments in user comments and reviews of the ACME products. ***Sentiment analysis*** refers to a group of tasks that use statistics and natural language processing to mine opinions to identify and extract subjective information from texts.

Early work on sentiment analysis focused on detecting the polarity of product reviews from Epinions [34] and movie reviews from the Internet Movie Database (IMDb) [35] at the document level. Later work handles sentiment analysis at the sentence level [36]. More recently, the focus has shifted to phrase-level [37] and short-text forms in response to the popularity of micro-blogging services such as Twitter [38, 39, 40, 41, 42].