relates to the decomposition methods from Chapter 3. Each of the components we learn then corresponds to one topic, and the coefficients of the components in the representation of a document tell us how strongly related that document is to a particular topic. Often, when people talk about topic modeling, they refer to one particular decomposition method called *Latent Dirichlet Allocation* (often LDA for short).[9]

Latent Dirichlet Allocation

Intuitively, the LDA model tries to find groups of words (the topics) that appear together frequently. LDA also requires that each document can be understood as a "mixture" of a subset of the topics. It is important to understand that for the machine learning model a "topic" might not be what we would normally call a topic in everyday speech, but that it resembles more the components extracted by PCA or NMF (which we discussed in Chapter 3), which might or might not have a semantic meaning. Even if there is a semantic meaning for an LDA "topic", it might not be something we'd usually call a topic. Going back to the example of news articles, we might have a collection of articles about sports, politics, and finance, written by two specific authors. In a politics article, we might expect to see words like "governor," "vote," "party," etc., while in a sports article we might expect words like "team," "score," and "season." Words in each of these groups will likely appear together, while it's less likely that, for example, "team" and "governor" will appear together. However, these are not the only groups of words we might expect to appear together. The two reporters might prefer different phrases or different choices of words. Maybe one of them likes to use the word "demarcate" and one likes the word "polarize." Other "topics" would then be "words often used by reporter A" and "words often used by reporter B," though these are not topics in the usual sense of the word.

Let's apply LDA to our movie review dataset to see how it works in practice. For unsupervised text document models, it is often good to remove very common words, as they might otherwise dominate the analysis. We'll remove words that appear in at least 20 percent of the documents, and we'll limit the bag-of-words model to the 10,000 words that are most common after removing the top 20 percent:

In[41]:

```
vect = CountVectorizer(max_features=10000, max_df=.15)
X = vect.fit_transform(text_train)
```

9 There is another machine learning model that is also often abbreviated LDA: Linear Discriminant Analysis, a linear classification model. This leads to quite some confusion. In this book, LDA refers to Latent Dirichlet Allocation.

We will learn a topic model with 10 topics, which is few enough that we can look at all of them. Similarly to the components in NMF, topics don't have an inherent ordering, and changing the number of topics will change all of the topics.[10] We'll use the `"batch"` learning method, which is somewhat slower than the default (`"online"`) but usually provides better results, and increase `"max_iter"`, which can also lead to better models:

In[42]:

```
from sklearn.decomposition import LatentDirichletAllocation
lda = LatentDirichletAllocation(n_topics=10, learning_method="batch",
                                max_iter=25, random_state=0)
# We build the model and transform the data in one step
# Computing transform takes some time,
# and we can save time by doing both at once
document_topics = lda.fit_transform(X)
```

Like the decomposition methods we saw in Chapter 3, `LatentDirichletAllocation` has a `components_` attribute that stores how important each word is for each topic. The size of `components_` is `(n_topics, n_words)`:

In[43]:

```
lda.components_.shape
```

Out[43]:

```
(10, 10000)
```

To understand better what the different topics mean, we will look at the most important words for each of the topics. The `print_topics` function provides a nice formatting for these features:

In[44]:

```
# For each topic (a row in the components_), sort the features (ascending)
# Invert rows with [:, ::-1] to make sorting descending
sorting = np.argsort(lda.components_, axis=1)[:, ::-1]
# Get the feature names from the vectorizer
feature_names = np.array(vect.get_feature_names())
```

In[45]:

```
# Print out the 10 topics:
mglearn.tools.print_topics(topics=range(10), feature_names=feature_names,
                           sorting=sorting, topics_per_chunk=5, n_words=10)
```

10 In fact, NMF and LDA solve quite related problems, and we could also use NMF to extract topics.

Out[45]:

```
topic 0       topic 1       topic 2       topic 3       topic 4
--------      --------      --------      --------      --------
between       war           funny         show          didn
young         world         worst         series        saw
family        us            comedy        episode       am
real          our           thing         tv            thought
performance   american      guy           episodes      years
beautiful     documentary   re            shows         book
work          history       stupid        season        watched
each          new           actually      new           now
both          own           nothing       television    dvd
director      point         want          years         got


topic 5       topic 6       topic 7       topic 8       topic 9
--------      --------      --------      --------      --------
horror        kids          cast          performance   house
action        action        role          role          woman
effects       animation     john          john          gets
budget        game          version       actor         killer
nothing       fun           novel         oscar         girl
original      disney        both          cast          wife
director      children      director      plays         horror
minutes       10            played        jack          young
pretty        kid           performance   joe           goes
doesn         old           mr            performances  around
```

Judging from the important words, topic 1 seems to be about historical and war movies, topic 2 might be about bad comedies, topic 3 might be about TV series. Topic 4 seems to capture some very common words, while topic 6 appears to be about children's movies and topic 8 seems to capture award-related reviews. Using only 10 topics, each of the topics needs to be very broad, so that they can together cover all the different kinds of reviews in our dataset.

Next, we will learn another model, this time with 100 topics. Using more topics makes the analysis much harder, but makes it more likely that topics can specialize to interesting subsets of the data:

In[46]:

```
lda100 = LatentDirichletAllocation(n_topics=100, learning_method="batch",
                                   max_iter=25, random_state=0)
document_topics100 = lda100.fit_transform(X)
```

Looking at all 100 topics would be a bit overwhelming, so we selected some interesting and representative topics:

In[47]:

```
topics = np.array([7, 16, 24, 25, 28, 36, 37, 45, 51, 53, 54, 63, 89, 97])

sorting = np.argsort(lda100.components_, axis=1)[:, ::-1]
feature_names = np.array(vect.get_feature_names())
mglearn.tools.print_topics(topics=topics, feature_names=feature_names,
                           sorting=sorting, topics_per_chunk=7, n_words=20)
```

Out[48]:

```
topic 7       topic 16      topic 24      topic 25      topic 28
--------      --------      --------      --------      --------
thriller      worst         german        car           beautiful
suspense      awful         hitler        gets          young
horror        boring        nazi          guy           old
atmosphere    horrible      midnight      around        romantic
mystery       stupid        joe           down          between
house         thing         germany       kill          romance
director      terrible      years         goes          wonderful
quite         script        history       killed        heart
bit           nothing       new           going         feel
de            worse         modesty       house         year
performances  waste         cowboy        away          each
dark          pretty        jewish        head          french
twist         minutes       past          take          sweet
hitchcock     didn          kirk          another       boy
tension       actors        young         getting       loved
interesting   actually      spanish       doesn         girl
mysterious    re            enterprise    now           relationship
murder        supposed      von           night         saw
ending        mean          nazis         right         both
creepy        want          spock         woman         simple


topic 36      topic 37      topic 41      topic 45      topic 51
--------      --------      --------      --------      --------
performance   excellent     war           music         earth
role          highly        american      song          space
actor         amazing       world         songs         planet
cast          wonderful     soldiers      rock          superman
play          truly         military      band          alien
actors        superb        army          soundtrack    world
performances  actors        tarzan        singing       evil
played        brilliant     soldier       voice         humans
supporting    recommend     america       singer        aliens
director      quite         country       sing          human
oscar         performance   americans     musical       creatures
roles         performances  during        roll          miike
actress       perfect       men           fan           monsters
excellent     drama         us            metal         apes
screen        without       government    concert       clark
plays         beautiful     jungle        playing       burton
award         human         vietnam       hear          tim
work          moving        ii            fans          outer
playing       world         political     prince        men
gives         recommended   against       especially    moon
```

```
topic 53      topic 54      topic 63      topic 89      topic 97
--------      --------      --------      --------      --------
scott         money         funny         dead          didn
gary          budget        comedy        zombie        thought
streisand     actors        laugh         gore          wasn
star          low           jokes         zombies       ending
hart          worst         humor         blood         minutes
lundgren      waste         hilarious     horror        got
dolph         10            laughs        flesh         felt
career        give          fun           minutes       part
sabrina       want          re            body          going
role          nothing       funniest      living        seemed
temple        terrible      laughing      eating        bit
phantom       crap          joke          flick         found
judy          must          few           budget        though
melissa       reviews       moments       head          nothing
zorro         imdb          guy           gory          lot
gets          director      unfunny       evil          saw
barbra        thing         times         shot          long
cast          believe       laughed       low           interesting
short         am            comedies      fulci         few
serial        actually      isn           re            half
```

The topics we extracted this time seem to be more specific, though many are hard to interpret. Topic 7 seems to be about horror movies and thrillers; topics 16 and 54 seem to capture bad reviews, while topic 63 mostly seems to be capturing positive reviews of comedies. If we want to make further inferences using the topics that were discovered, we should confirm the intuition we gained from looking at the highest-ranking words for each topic by looking at the documents that are assigned to these topics. For example, topic 45 seems to be about music. Let's check which kinds of reviews are assigned to this topic:

In[49]:

```
# sort by weight of "music" topic 45
music = np.argsort(document_topics100[:, 45])[::-1]
# print the five documents where the topic is most important
for i in music[:10]:
    # pshow first two sentences
    print(b".".join(text_train[i].split(b".")[:2]) + b".\n")
```

Out[49]:

```
b'I love this movie and never get tired of watching. The music in it is great.\n'
b"I enjoyed Still Crazy more than any film I have seen in years. A successful
  band from the 70's decide to give it another try.\n"
b'Hollywood Hotel was the last movie musical that Busby Berkeley directed for
  Warner Bros. His directing style had changed or evolved to the point that
  this film does not contain his signature overhead shots or huge production
  numbers with thousands of extras.\n'
b"What happens to washed up rock-n-roll stars in the late 1990's?
  They launch a comeback / reunion tour. At least, that's what the members of
  Strange Fruit, a (fictional) 70's stadium rock group do.\n"
```

```
b'As a big-time Prince fan of the last three to four years, I really can\'t
  believe I\'ve only just got round to watching "Purple Rain". The brand new
  2-disc anniversary Special Edition led me to buy it.\n'
b"This film is worth seeing alone for Jared Harris' outstanding portrayal
  of John Lennon. It doesn't matter that Harris doesn't exactly resemble
  Lennon; his mannerisms, expressions, posture, accent and attitude are
  pure Lennon.\n"
b"The funky, yet strictly second-tier British glam-rock band Strange Fruit
  breaks up at the end of the wild'n'wacky excess-ridden 70's. The individual
  band members go their separate ways and uncomfortably settle into lackluster
  middle age in the dull and uneventful 90's: morose keyboardist Stephen Rea
  winds up penniless and down on his luck, vain, neurotic, pretentious lead
  singer Bill Nighy tries (and fails) to pursue a floundering solo career,
  paranoid drummer Timothy Spall resides in obscurity on a remote farm so he
  can avoid paying a hefty back taxes debt, and surly bass player Jimmy Nail
  installs roofs for a living.\n"
b"I just finished reading a book on Anita Loos' work and the photo in TCM
  Magazine of MacDonald in her angel costume looked great (impressive wings),
  so I thought I'd watch this movie. I'd never heard of the film before, so I
  had no preconceived notions about it whatsoever.\n"
b'I love this movie!!! Purple Rain came out the year I was born and it has had
  my heart since I can remember. Prince is so tight in this movie.\n'
b"This movie is sort of a Carrie meets Heavy Metal. It's about a highschool
  guy who gets picked on alot and he totally gets revenge with the help of a
  Heavy Metal ghost.\n"
```

As we can see, this topic covers a wide variety of music-centered reviews, from musicals, to biographical movies, to some hard-to-specify genre in the last review. Another interesting way to inspect the topics is to see how much weight each topic gets overall, by summing the `document_topics` over all reviews. We name each topic by the two most common words. Figure 7-6 shows the topic weights learned:

In[50]:

```
fig, ax = plt.subplots(1, 2, figsize=(10, 10))
topic_names = ["{:>2} ".format(i) + " ".join(words)
               for i, words in enumerate(feature_names[sorting[:, :2]])]
# two column bar chart:
for col in [0, 1]:
    start = col * 50
    end = (col + 1) * 50
    ax[col].barh(np.arange(50), np.sum(document_topics100, axis=0)[start:end])
    ax[col].set_yticks(np.arange(50))
    ax[col].set_yticklabels(topic_names[start:end], ha="left", va="top")
    ax[col].invert_yaxis()
    ax[col].set_xlim(0, 2000)
    yax = ax[col].get_yaxis()
    yax.set_tick_params(pad=130)
plt.tight_layout()
```

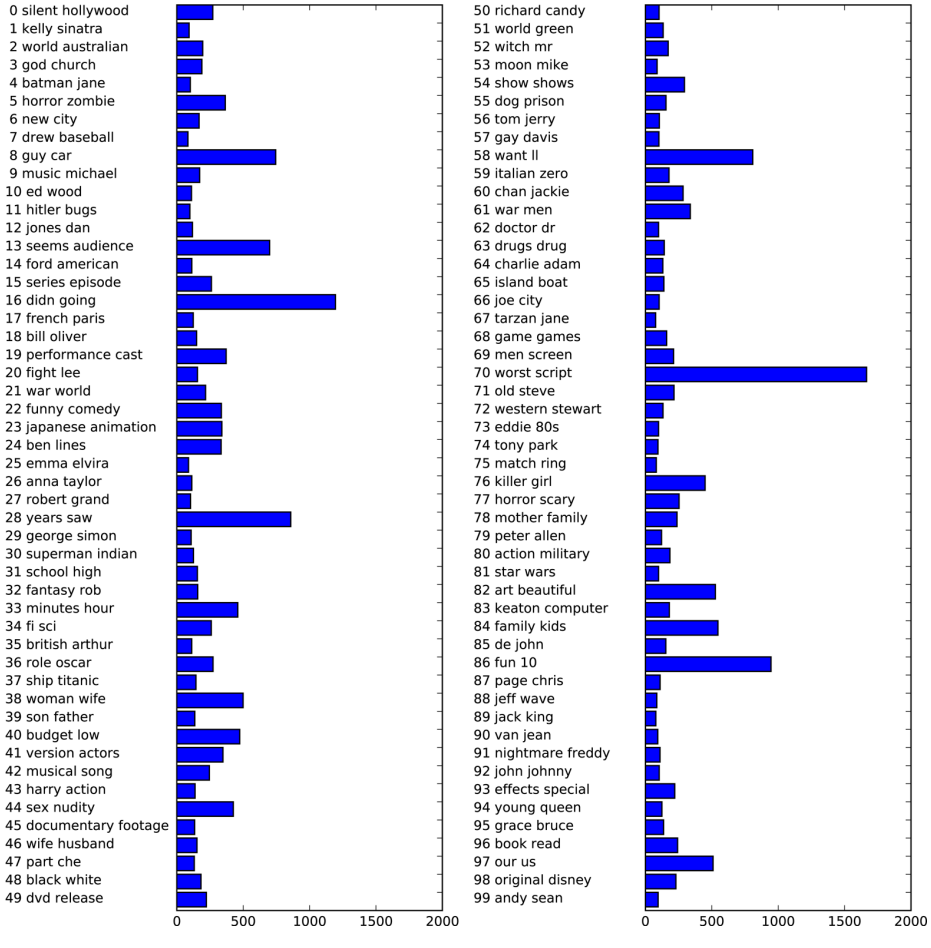

Figure 7-6. Topic weights learned by LDA

The most important topics are 97, which seems to consist mostly of stopwords, possibly with a slight negative direction; topic 16, which is clearly about bad reviews; followed by some genre-specific topics and 36 and 37, both of which seem to contain laudatory words.

It seems like LDA mostly discovered two kind of topics, genre-specific and rating-specific, in addition to several more unspecific topics. This is an interesting discovery, as most reviews are made up of some movie-specific comments and some comments that justify or emphasize the rating.

Topic models like LDA are interesting methods to understand large text corpora in the absence of labels—or, as here, even if labels are available. The LDA algorithm is randomized, though, and changing the `random_state` parameter can lead to quite

different outcomes. While identifying topics can be helpful, any conclusions you draw from an unsupervised model should be taken with a grain of salt, and we recommend verifying your intuition by looking at the documents in a specific topic. The topics produced by the `LDA.transform` method can also sometimes be used as a compact representation for supervised learning. This is particularly helpful when few training examples are available.

Summary and Outlook

In this chapter we talked about the basics of processing text, also known as *natural language processing* (NLP), with an example application classifying movie reviews. The tools discussed here should serve as a great starting point when trying to process text data. In particular for text classification tasks such as spam and fraud detection or sentiment analysis, bag-of-words representations provide a simple and powerful solution. As is often the case in machine learning, the representation of the data is key in NLP applications, and inspecting the tokens and *n*-grams that are extracted can give powerful insights into the modeling process. In text-processing applications, it is often possible to introspect models in a meaningful way, as we saw in this chapter, for both supervised and unsupervised tasks. You should take full advantage of this ability when using NLP-based methods in practice.

Natural language and text processing is a large research field, and discussing the details of advanced methods is far beyond the scope of this book. If you want to learn more, we recommend the O'Reilly book *Natural Language Processing with Python* by Steven Bird, Ewan Klein, and Edward Loper, which provides an overview of NLP together with an introduction to the `nltk` Python package for NLP. Another great and more conceptual book is the standard reference *Introduction to Information Retrieval* by Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze, which describes fundamental algorithms in information retrieval, NLP, and machine learning. Both books have online versions that can be accessed free of charge. As we discussed earlier, the classes `CountVectorizer` and `TfidfVectorizer` only implement relatively simple text-processing methods. For more advanced text-processing methods, we recommend the Python packages `spacy` (a relatively new but very efficient and well-designed package), `nltk` (a very well-established and complete but somewhat dated library), and `gensim` (an NLP package with an emphasis on topic modeling).

There have been several very exciting new developments in text processing in recent years, which are outside of the scope of this book and relate to neural networks. The first is the use of continuous vector representations, also known as word vectors or distributed word representations, as implemented in the `word2vec` library. The original paper "Distributed Representations of Words and Phrases and Their Compositionality" by Thomas Mikolov et al. is a great introduction to the subject. Both `spacy`