Out[39]:

```
X_train_lemma.shape:  (25000, 21596)
X_train.shape:  (25000, 27271)
```

As you can see from the output, lemmatization reduced the number of features from 27,271 (with the standard `CountVectorizer` processing) to 21,596. Lemmatization can be seen as a kind of regularization, as it conflates certain features. Therefore, we expect lemmatization to improve performance most when the dataset is small. To illustrate how lemmatization can help, we will use `StratifiedShuffleSplit` for cross-validation, using only 1% of the data as training data and the rest as test data:

In[40]:

```
# build a grid search using only 1% of the data as the training set
from sklearn.model_selection import StratifiedShuffleSplit

param_grid = {'C': [0.001, 0.01, 0.1, 1, 10]}
cv = StratifiedShuffleSplit(n_iter=5, test_size=0.99,
                            train_size=0.01, random_state=0)
grid = GridSearchCV(LogisticRegression(), param_grid, cv=cv)
# perform grid search with standard CountVectorizer
grid.fit(X_train, y_train)
print("Best cross-validation score "
      "(standard CountVectorizer): {:.3f}".format(grid.best_score_))
# perform grid search with lemmatization
grid.fit(X_train_lemma, y_train)
print("Best cross-validation score "
      "(lemmatization): {:.3f}".format(grid.best_score_))
```

Out[40]:

```
Best cross-validation score (standard CountVectorizer): 0.721
Best cross-validation score (lemmatization): 0.731
```

In this case, lemmatization provided a modest improvement in performance. As with many of the different feature extraction techniques, the result varies depending on the dataset. Lemmatization and stemming can sometimes help in building better (or at least more compact) models, so we suggest you give these techniques a try when trying to squeeze out the last bit of performance on a particular task.

Topic Modeling and Document Clustering

One particular technique that is often applied to text data is *topic modeling*, which is an umbrella term describing the task of assigning each document to one or multiple *topics*, usually without supervision. A good example for this is news data, which might be categorized into topics like "politics," "sports," "finance," and so on. If each document is assigned a single topic, this is the task of clustering the documents, as discussed in Chapter 3. If each document can have more than one topic, the task

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.