13 *Text classification and Naive Bayes*

Thus far, this book has mainly discussed the process of *ad hoc retrieval*, where users have transient information needs that they try to address by posing one or more queries to a search engine. However, many users have ongoing information needs. For example, you might need to track developments in *multicore computer chips*. One way of doing this is to issue the query multicore AND computer AND chip against an index of recent newswire articles each morning. In this and the following two chapters we examine the question: How can this repetitive task be automated? To this end, many systems support *standing queries*. A standing query is like any other query except that it is periodically executed on a collection to which new documents are incrementally added over time.

STANDING QUERY

If your standing query is just multicore AND computer AND chip, you will tend to miss many relevant new articles which use other terms such as *multicore processors*. To achieve good recall, standing queries thus have to be refined over time and can gradually become quite complex. In this example, using a Boolean search engine with stemming, you might end up with a query like (multicore OR multi-core) AND (chip OR processor OR microprocessor).

To capture the generality and scope of the problem space to which standing queries belong, we now introduce the general notion of a *classification* problem. Given a set of *classes*, we seek to determine which class(es) a given object belongs to. In the example, the standing query serves to divide new newswire articles into the two classes: documents about multicore computer chips and documents not about multicore computer chips. We refer to this as *two-class classification*. Classification using standing queries is also called *routing* or *filtering*and will be discussed further in Section 15.3.1 (page 335).

CLASSIFICATION

ROUTING

FILTERING

A class need not be as narrowly focused as the standing query *multicore computer chips*. Often, a class is a more general subject area like *China* or *coffee*. Such more general classes are usually referred to as *topics*, and the classification task is then called *text classification*, *text categorization*, *topic classification*, or *topic spotting*. An example for *China* appears in Figure 13.1. Standing queries and topics differ in their degree of specificity, but the methods for

TEXT CLASSIFICATION

solving routing, filtering, and text classification are essentially the same. We therefore include routing and filtering under the rubric of text classification in this and the following chapters.

The notion of classification is very general and has many applications within and beyond information retrieval (IR). For instance, in computer vision, a classifier may be used to divide images into classes such as *landscape*, *portrait*, and *neither*. We focus here on examples from information retrieval such as:

- Several of the preprocessing steps necessary for indexing as discussed in Chapter 2: detecting a document's encoding (ASCII, Unicode UTF-8 etc; page 20); word segmentation (Is the white space between two letters a word boundary or not? page 24) ; truecasing (page 30); and identifying the language of a document (page 46).
- The automatic detection of spam pages (which then are not included in the search engine index).
- The automatic detection of sexually explicit content (which is included in search results only if the user turns an option such as SafeSearch off).

SENTIMENT DETECTION

- *Sentiment detection* or the automatic classification of a movie or product review as positive or negative. An example application is a user searching for negative reviews before buying a camera to make sure it has no undesirable features or quality problems.

EMAIL SORTING

- Personal *email sorting*. A user may have folders like *talk announcements*, *electronic bills*, *email from family and friends*, and so on, and may want a classifier to classify each incoming email and automatically move it to the appropriate folder. It is easier to find messages in sorted folders than in a very large inbox. The most common case of this application is a spam folder that holds all suspected spam messages.

VERTICAL SEARCH ENGINE

- Topic-specific or *vertical* search. *Vertical search engines* restrict searches to a particular topic. For example, the query computer science on a vertical search engine for the topic *China* will return a list of Chinese computer science departments with higher precision and recall than the query computer science China on a general purpose search engine. This is because the vertical search engine does not include web pages in its index that contain the term china in a different sense (e.g., referring to a hard white ceramic), but does include relevant pages even if they do not explicitly mention the term China.
- Finally, the ranking function in ad hoc information retrieval can also be based on a document classifier as we will explain in Section 15.4 (page 341).

This list shows the general importance of classification in IR. Most retrieval systems today contain multiple components that use some form of classifier. The classification task we will use as an example in this book is text classification.

A computer is not essential for classification. Many classification tasks have traditionally been solved manually. Books in a library are assigned Library of Congress categories by a librarian. But manual classification is expensive to scale. The *multicore computer chips* example illustrates one alternative approach: classification by the use of standing queries – which can be thought of as *rules* – most commonly written by hand. As in our example (multicore OR multi-core) AND (chip OR processor OR microprocessor), rules are sometimes equivalent to Boolean expressions.

RULES IN TEXT CLASSIFICATION

A rule captures a certain combination of keywords that indicates a class. Hand-coded rules have good scaling properties, but creating and maintaining them over time is labor intensive. A technically skilled person (e.g., a domain expert who is good at writing regular expressions) can create rule sets that will rival or exceed the accuracy of the automatically generated classifiers we will discuss shortly; however, it can be hard to find someone with this specialized skill.

Apart from manual classification and hand-crafted rules, there is a third approach to text classification, namely, machine learning-based text classification. It is the approach that we focus on in the next several chapters. In machine learning, the set of rules or, more generally, the decision criterion of the text classifier, is learned automatically from training data. This approach is also called *statistical text classification* if the learning method is statistical. In statistical text classification, we require a number of good example documents (or training documents) for each class. The need for manual classification is not eliminated because the training documents come from a person who has labeled them – where *labeling* refers to the process of annotating each document with its class. But labeling is arguably an easier task than writing rules. Almost anybody can look at a document and decide whether or not it is related to China. Sometimes such labeling is already implicitly part of an existing workflow. For instance, you may go through the news articles returned by a standing query each morning and give relevance feedback (cf. Chapter 9) by moving the relevant articles to a special folder like *multicore-processors*.

STATISTICAL TEXT CLASSIFICATION

LABELING

We begin this chapter with a general introduction to the text classification problem including a formal definition (Section 13.1); we then cover Naive Bayes, a particularly simple and effective classification method (Sections 13.2–13.4). All of the classification algorithms we study represent documents in high-dimensional spaces. To improve the efficiency of these algorithms, it is generally desirable to reduce the dimensionality of these spaces; to this end, a technique known as *feature selection* is commonly applied in text clas-

sification as discussed in Section 13.5. Section 13.6 covers evaluation of text classification. In the following chapters, Chapters 14 and 15, we look at two other families of classification methods, vector space classifiers and support vector machines.

13.1 The text classification problem

DOCUMENT SPACE
CLASS

In text classification, we are given a description $d \in \mathbb{X}$ of a document, where $\mathbb{X}$ is the *document space*; and a fixed set of *classes* $\mathbb{C} = \{c_1, c_2, \ldots, c_J\}$. Classes are also called *categories* or *labels*. Typically, the document space $\mathbb{X}$ is some type of high-dimensional space, and the classes are human defined for the needs of an application, as in the examples *China* and *documents that talk about multicore computer chips* above. We are given a *training set* $\mathbb{D}$ of labeled documents $\langle d, c \rangle$,where $\langle d, c \rangle \in \mathbb{X} \times \mathbb{C}$. For example:

TRAINING SET

$$\langle d, c \rangle = \langle \text{Beijing joins the World Trade Organization}, \textit{China} \rangle$$

for the one-sentence document *Beijing joins the World Trade Organization* and the class (or label) *China*.

LEARNING METHOD
CLASSIFIER

Using a *learning method* or *learning algorithm*, we then wish to learn a classifier or *classification function* γ that maps documents to classes:

$$\gamma : \mathbb{X} \rightarrow \mathbb{C} \tag{13.1}$$

SUPERVISED LEARNING

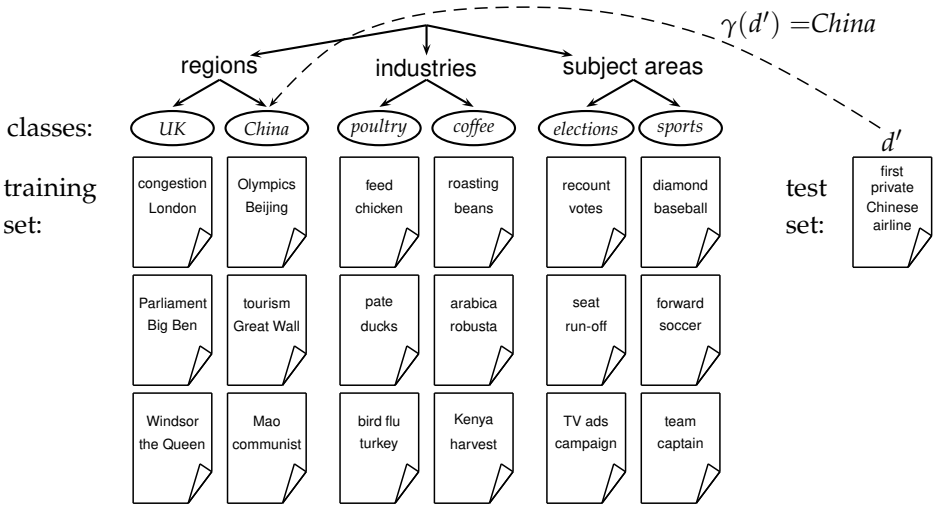
This type of learning is called *supervised learning* because a supervisor (the human who defines the classes and labels training documents) serves as a teacher directing the learning process. We denote the supervised learning method by Γ and write $\Gamma(\mathbb{D}) = \gamma$. The learning method Γ takes the training set $\mathbb{D}$ as input and returns the learned classification function γ.

Most names for learning methods Γ are also used for classifiers γ. We talk about the Naive Bayes (NB) *learning method* Γ when we say that "Naive Bayes is robust," meaning that it can be applied to many different learning problems and is unlikely to produce classifiers that fail catastrophically. But when we say that "Naive Bayes had an error rate of 20%," we are describing an experiment in which a particular NB *classifier* γ (which was produced by the NB learning method) had a 20% error rate in an application.

Figure 13.1 shows an example of text classification from the Reuters-RCV1 collection, introduced in Section 4.2, page 69. There are six classes (*UK*, *China*, ..., *sports*), each with three training documents. We show a few mnemonic words for each document's content. The training set provides some typical examples for each class, so that we can learn the classification function γ.

TEST SET

Once we have learned γ, we can apply it to the *test set* (or *test data*), for example, the new document *first private Chinese airline* whose class is unknown.



► **Figure 13.1** Classes, training set, and test set in text classification .

In Figure 13.1, the classification function assigns the new document to class $\gamma(d) = China$, which is the correct assignment.

The classes in text classification often have some interesting structure such as the hierarchy in Figure 13.1. There are two instances each of region categories, industry categories, and subject area categories. A hierarchy can be an important aid in solving a classification problem; see Section 15.3.2 for further discussion. Until then, we will make the assumption in the text classification chapters that the classes form a set with no subset relationships between them.

Definition (13.1) stipulates that a document is a member of exactly one class. This is not the most appropriate model for the hierarchy in Figure 13.1. For instance, a document about the 2008 Olympics should be a member of two classes: the *China* class and the *sports* class. This type of classification problem is referred to as an *any-of* problem and we will return to it in Section 14.5 (page 306). For the time being, we only consider *one-of* problems where a document is a member of exactly one class.

Our goal in text classification is high accuracy on test data or *new data* – for example, the newswire articles that we will encounter tomorrow morning in the multicore chip example. It is easy to achieve high accuracy on the training set (e.g., we can simply memorize the labels). But high accuracy on the training set in general does not mean that the classifier will work well on

new data in an application. When we use the training set to learn a classifier for test data, we make the assumption that training data and test data are similar or from *the same distribution*. We defer a precise definition of this notion to Section 14.6 (page 308).

13.2 Naive Bayes text classification

MULTINOMIAL NAIVE BAYES

The first supervised learning method we introduce is the *multinomial Naive Bayes* or *multinomial NB* model, a probabilistic learning method. The probability of a document d being in class c is computed as

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c) \tag{13.2}$$

where $P(t_k|c)$ is the conditional probability of term t_k occurring in a document of class c.[1] We interpret $P(t_k|c)$ as a measure of how much evidence t_k contributes that c is the correct class. $P(c)$ is the prior probability of a document occurring in class c. If a document's terms do not provide clear evidence for one class versus another, we choose the one that has a higher prior probability. $\langle t_1, t_2, \ldots, t_{n_d} \rangle$ are the tokens in d that are part of the vocabulary we use for classification and n_d is the number of such tokens in d. For example, $\langle t_1, t_2, \ldots, t_{n_d} \rangle$ for the one-sentence document *Beijing and Taipei join the WTO* might be $\langle$Beijing, Taipei, join, WTO$\rangle$, with $n_d = 4$, if we treat the terms and and the as stop words.

MAXIMUM A POSTERIORI CLASS

In text classification, our goal is to find the *best* class for the document. The best class in NB classification is the most likely or *maximum a posteriori* (MAP) class c_{map}:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} \hat{P}(c|d) = \arg\max_{c \in \mathbb{C}} \hat{P}(c) \prod_{1 \leq k \leq n_d} \hat{P}(t_k|c). \tag{13.3}$$

We write $\hat{P}$ for P because we do not know the true values of the parameters $P(c)$ and $P(t_k|c)$, but estimate them from the training set as we will see in a moment.

In Equation (13.3), many conditional probabilities are multiplied, one for each position $1 \leq k \leq n_d$. This can result in a floating point underflow. It is therefore better to perform the computation by adding logarithms of probabilities instead of multiplying probabilities. The class with the highest log probability score is still the most probable; $\log(xy) = \log(x) + \log(y)$ and the logarithm function is monotonic. Hence, the maximization that is

1. We will explain in the next section why $P(c|d)$ is proportional to ($\propto$), not equal to the quantity on the right.

actually done in most implementations of NB is:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} [\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c)]. \tag{13.4}$$

Equation (13.4) has a simple interpretation. Each conditional parameter $\log \hat{P}(t_k|c)$ is a weight that indicates how good an indicator t_k is for c. Similarly, the prior $\log \hat{P}(c)$ is a weight that indicates the relative frequency of c. More frequent classes are more likely to be the correct class than infrequent classes. The sum of log prior and term weights is then a measure of how much evidence there is for the document being in the class, and Equation (13.4) selects the class for which we have the most evidence.

We will initially work with this intuitive interpretation of the multinomial NB model and defer a formal derivation to Section 13.4.

How do we estimate the parameters $\hat{P}(c)$ and $\hat{P}(t_k|c)$? We first try the maximum likelihood estimate (MLE; Section 11.3.2, page 226), which is simply the relative frequency and corresponds to the most likely value of each parameter given the training data. For the priors this estimate is:

$$\hat{P}(c) = \frac{N_c}{N}, \tag{13.5}$$

where N_c is the number of documents in class c and N is the total number of documents.

We estimate the conditional probability $\hat{P}(t|c)$ as the relative frequency of term t in documents belonging to class c:

$$\hat{P}(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}, \tag{13.6}$$

where T_{ct} is the number of occurrences of t in training documents from class c, including multiple occurrences of a term in a document. We have made the *positional independence assumption* here, which we will discuss in more detail in the next section: T_{ct} is a count of occurrences in all positions k in the documents in the training set. Thus, we do not compute different estimates for different positions and, for example, if a word occurs twice in a document, in positions k_1 and k_2, then $\hat{P}(t_{k_1}|c) = \hat{P}(t_{k_2}|c)$.

The problem with the MLE estimate is that it is zero for a term–class combination that did not occur in the training data. If the term WTO in the training data only occurred in *China* documents, then the MLE estimates for the other classes, for example *UK*, will be zero:

$$\hat{P}(\text{WTO}|UK) = 0.$$

Now, the one-sentence document *Britain is a member of the WTO* will get a conditional probability of zero for *UK* because we are multiplying the conditional probabilities for all terms in Equation (13.2). Clearly, the model should

TRAINMULTINOMIALNB($\mathbb{C}, \mathbb{D}$)
1 $V \leftarrow$ EXTRACTVOCABULARY($\mathbb{D}$)
2 $N \leftarrow$ COUNTDOCS($\mathbb{D}$)
3 **for each** $c \in \mathbb{C}$
4 **do** $N_c \leftarrow$ COUNTDOCSINCLASS($\mathbb{D}, c$)
5 $prior[c] \leftarrow N_c/N$
6 $text_c \leftarrow$ CONCATENATETEXTOFALLDOCSINCLASS($\mathbb{D}, c$)
7 **for each** $t \in V$
8 **do** $T_{ct} \leftarrow$ COUNTTOKENSOFTERM($text_c, t$)
9 **for each** $t \in V$
10 **do** $condprob[t][c] \leftarrow \frac{T_{ct}+1}{\sum_{t'}(T_{ct'}+1)}$
11 **return** $V, prior, condprob$

APPLYMULTINOMIALNB($\mathbb{C}, V, prior, condprob, d$)
1 $W \leftarrow$ EXTRACTTOKENSFROMDOC(V, d)
2 **for each** $c \in \mathbb{C}$
3 **do** $score[c] \leftarrow \log prior[c]$
4 **for each** $t \in W$
5 **do** $score[c] \mathrel{+}= \log condprob[t][c]$
6 **return** $\arg\max_{c \in \mathbb{C}} score[c]$

▶ **Figure 13.2** Naive Bayes algorithm (multinomial model): Training and testing.

assign a high probability to the *UK* class because the term Britain occurs. The problem is that the zero probability for WTO cannot be "conditioned away," no matter how strong the evidence for the class *UK* from other features. The estimate is 0 because of *sparseness*: The training data are never large enough to represent the frequency of rare events adequately, for example, the frequency of WTO occurring in *UK* documents.

SPARSENESS

ADD-ONE SMOOTHING

To eliminate zeros, we use *add-one* or *Laplace smoothing*, which simply adds one to each count (cf. Section 11.3.2):

$$\hat{P}(t|c) = \frac{T_{ct}+1}{\sum_{t' \in V}(T_{ct'}+1)} = \frac{T_{ct}+1}{(\sum_{t' \in V} T_{ct'}) + B}, \tag{13.7}$$

where $B = |V|$ is the number of terms in the vocabulary. Add-one smoothing can be interpreted as a uniform prior (each term occurs once for each class) that is then updated as evidence from the training data comes in. Note that this is a prior probability for the occurrence of a *term* as opposed to the prior probability of a *class* which we estimate in Equation (13.5) on the document level.

▶ **Table 13.1** Data for parameter estimation examples.

	docID	words in document	in $c =$ *China*?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

▶ **Table 13.2** Training and test times for NB.

mode	time complexity
training	$\Theta(\|\mathbb{D}\|L_{\text{ave}} + \|\mathbb{C}\||V\|)$
testing	$\Theta(L_a + \|\mathbb{C}\|M_a) = \Theta(\|\mathbb{C}\|M_a)$

We have now introduced all the elements we need for training and applying an NB classifier. The complete algorithm is described in Figure 13.2.

Example 13.1: For the example in Table 13.1, the multinomial parameters we need to classify the test document are the priors $\hat{P}(c) = 3/4$ and $\hat{P}(\overline{c}) = 1/4$ and the following conditional probabilities:

$$\begin{aligned}
\hat{P}(\textsf{Chinese}|c) &= (5+1)/(8+6) = 6/14 = 3/7 \\
\hat{P}(\textsf{Tokyo}|c) = \hat{P}(\textsf{Japan}|c) &= (0+1)/(8+6) = 1/14 \\
\hat{P}(\textsf{Chinese}|\overline{c}) &= (1+1)/(3+6) = 2/9 \\
\hat{P}(\textsf{Tokyo}|\overline{c}) = \hat{P}(\textsf{Japan}|\overline{c}) &= (1+1)/(3+6) = 2/9
\end{aligned}$$

The denominators are $(8+6)$ and $(3+6)$ because the lengths of $text_c$ and $text_{\overline{c}}$ are 8 and 3, respectively, and because the constant B in Equation (13.7) is 6 as the vocabulary consists of six terms.

We then get:

$$\begin{aligned}
\hat{P}(c|d_5) &\propto 3/4 \cdot (3/7)^3 \cdot 1/14 \cdot 1/14 \approx 0.0003. \\
\hat{P}(\overline{c}|d_5) &\propto 1/4 \cdot (2/9)^3 \cdot 2/9 \cdot 2/9 \approx 0.0001.
\end{aligned}$$

Thus, the classifier assigns the test document to $c =$ *China*. The reason for this classification decision is that the three occurrences of the positive indicator Chinese in d_5 outweigh the occurrences of the two negative indicators Japan and Tokyo.

What is the time complexity of NB? The complexity of computing the parameters is $\Theta(|\mathbb{C}||V|)$ because the set of parameters consists of $|\mathbb{C}||V|$ conditional probabilities and $|\mathbb{C}|$ priors. The preprocessing necessary for computing the parameters (extracting the vocabulary, counting terms, etc.) can be done in one pass through the training data. The time complexity of this

component is therefore $\Theta(|\mathbb{D}|L_{ave})$, where $|\mathbb{D}|$ is the number of documents and L_{ave} is the average length of a document.

We use $\Theta(|\mathbb{D}|L_{ave})$ as a notation for $\Theta(T)$ here, where T is the length of the training collection. This is nonstandard; $\Theta(.)$ is not defined for an average. We prefer expressing the time complexity in terms of $\mathbb{D}$ and L_{ave} because these are the primary statistics used to characterize training collections.

The time complexity of APPLYMULTINOMIALNB in Figure 13.2 is $\Theta(|\mathbb{C}|L_a)$. L_a and M_a are the numbers of tokens and types, respectively, in the test document. APPLYMULTINOMIALNB can be modified to be $\Theta(L_a + |\mathbb{C}|M_a)$ (Exercise 13.8). Finally, assuming that the length of test documents is bounded, $\Theta(L_a + |\mathbb{C}|M_a) = \Theta(|\mathbb{C}|M_a)$ because $L_a < b|C|M_a$ for a fixed constant b.[2]

Table 13.2 summarizes the time complexities. In general, we have $|\mathbb{C}||V| < |\mathbb{D}|L_{ave}$, so both training and testing complexity are linear in the time it takes to scan the data. Because we have to look at the data at least once, NB can be said to have optimal time complexity. Its efficiency is one reason why NB is a popular text classification method.

13.2.1 Relation to multinomial unigram language model

The multinomial NB model is formally identical to the multinomial unigram language model (Section 12.2.1, page 242). In particular, Equation (13.2) is a special case of Equation (12.12) from page 243, which we repeat here for $\lambda = 1$:

$$P(d|q) \propto P(d) \prod_{t \in q} P(t|M_d). \tag{13.8}$$

The document d in text classification (Equation (13.2)) takes the role of the query in language modeling (Equation (13.8)) and the classes c in text classification take the role of the documents d in language modeling. We used Equation (13.8) to rank documents according to the probability that they are relevant to the query q. In NB classification, we are usually only interested in the top-ranked class.

We also used MLE estimates in Section 12.2.2 (page 243) and encountered the problem of zero estimates owing to sparse data (page 244); but instead of add-one smoothing, we used a mixture of two distributions to address the problem there. Add-one smoothing is closely related to add-$\frac{1}{2}$ smoothing in Section 11.3.4 (page 228).

? **Exercise 13.1**

Why is $|\mathbb{C}||V| < |\mathbb{D}|L_{ave}$ in Table 13.2 expected to hold for most text collections?

2. Our assumption here is that the length of test documents is bounded. L_a would exceed $b|C|M_a$ for extremely long test documents.

```
TRAINBERNOULLINB(ℂ, 𝔻)
1  V ← EXTRACTVOCABULARY(𝔻)
2  N ← COUNTDOCS(𝔻)
3  for each c ∈ ℂ
4  do N_c ← COUNTDOCSINCLASS(𝔻, c)
5     prior[c] ← N_c / N
6     for each t ∈ V
7     do N_ct ← COUNTDOCSINCLASSCONTAININGTERM(𝔻, c, t)
8        condprob[t][c] ← (N_ct + 1) / (N_c + 2)
9  return V, prior, condprob

APPLYBERNOULLINB(ℂ, V, prior, condprob, d)
1  V_d ← EXTRACTTERMSFROMDOC(V, d)
2  for each c ∈ ℂ
3  do score[c] ← log prior[c]
4     for each t ∈ V
5     do if t ∈ V_d
6           then score[c] += log condprob[t][c]
7           else score[c] += log(1 − condprob[t][c])
8  return arg max_{c∈ℂ} score[c]
```

▶ **Figure 13.3** NB algorithm (Bernoulli model): Training and testing. The add-one smoothing in Line 8 (top) is in analogy to Equation (13.7) with $B = 2$.

13.3 The Bernoulli model

There are two different ways we can set up an NB classifier. The model we introduced in the previous section is the multinomial model. It generates one term from the vocabulary in each position of the document, where we assume a generative model that will be discussed in more detail in Section 13.4 (see also page 237).

BERNOULLI MODEL

An alternative to the multinomial model is the *multivariate Bernoulli model* or *Bernoulli model*. It is equivalent to the binary independence model of Section 11.3 (page 222), which generates an indicator for each term of the vocabulary, either 1 indicating presence of the term in the document or 0 indicating absence. Figure 13.3 presents training and testing algorithms for the Bernoulli model. The Bernoulli model has the same time complexity as the multinomial model.

The different generation models imply different estimation strategies and different classification rules. The Bernoulli model estimates $\hat{P}(t|c)$ as the *fraction of documents* of class c that contain term t (Figure 13.3, TRAINBERNOULLI-

NB, line 8). In contrast, the multinomial model estimates $\hat{P}(t|c)$ as the *fraction of tokens* or *fraction of positions* in documents of class c that contain term t (Equation (13.7)). When classifying a test document, the Bernoulli model uses binary occurrence information, ignoring the number of occurrences, whereas the multinomial model keeps track of multiple occurrences. As a result, the Bernoulli model typically makes many mistakes when classifying long documents. For example, it may assign an entire book to the class *China* because of a single occurrence of the term China.

The models also differ in how nonoccurring terms are used in classification. They do not affect the classification decision in the multinomial model; but in the Bernoulli model the probability of nonoccurrence is factored in when computing $P(c|d)$ (Figure 13.3, APPLYBERNOULLINB, Line 7). This is because only the Bernoulli NB model models absence of terms explicitly.

Example 13.2: Applying the Bernoulli model to the example in Table 13.1, we have the same estimates for the priors as before: $\hat{P}(c) = 3/4$, $\hat{P}(\overline{c}) = 1/4$. The conditional probabilities are:

$$\begin{aligned}
\hat{P}(\text{Chinese}|c) &= (3+1)/(3+2) = 4/5\\
\hat{P}(\text{Japan}|c) = \hat{P}(\text{Tokyo}|c) &= (0+1)/(3+2) = 1/5\\
\hat{P}(\text{Beijing}|c) = \hat{P}(\text{Macao}|c) = \hat{P}(\text{Shanghai}|c) &= (1+1)/(3+2) = 2/5\\
\hat{P}(\text{Chinese}|\overline{c}) &= (1+1)/(1+2) = 2/3\\
\hat{P}(\text{Japan}|\overline{c}) = \hat{P}(\text{Tokyo}|\overline{c}) &= (1+1)/(1+2) = 2/3\\
\hat{P}(\text{Beijing}|\overline{c}) = \hat{P}(\text{Macao}|\overline{c}) = \hat{P}(\text{Shanghai}|\overline{c}) &= (0+1)/(1+2) = 1/3
\end{aligned}$$

The denominators are $(3+2)$ and $(1+2)$ because there are three documents in c and one document in $\overline{c}$ and because the constant B in Equation (13.7) is 2 – there are two cases to consider for each term, occurrence and nonoccurrence.

The scores of the test document for the two classes are

$$\begin{aligned}
\hat{P}(c|d_5) &\propto \hat{P}(c)\cdot\hat{P}(\text{Chinese}|c)\cdot\hat{P}(\text{Japan}|c)\cdot\hat{P}(\text{Tokyo}|c)\\
&\quad\cdot(1-\hat{P}(\text{Beijing}|c))\cdot(1-\hat{P}(\text{Shanghai}|c))\cdot(1-\hat{P}(\text{Macao}|c))\\
&= 3/4\cdot 4/5\cdot 1/5\cdot 1/5\cdot(1-2/5)\cdot(1-2/5)\cdot(1-2/5)\\
&\approx 0.005
\end{aligned}$$

and, analogously,

$$\begin{aligned}
\hat{P}(\overline{c}|d_5) &\propto 1/4\cdot 2/3\cdot 2/3\cdot 2/3\cdot(1-1/3)\cdot(1-1/3)\cdot(1-1/3)\\
&\approx 0.022
\end{aligned}$$

Thus, the classifier assigns the test document to $\overline{c}$ = *not-China*. When looking only at binary occurrence and not at term frequency, Japan and Tokyo are indicators for $\overline{c}$ ($2/3 > 1/5$) and the conditional probabilities of Chinese for c and $\overline{c}$ are not different enough ($4/5$ vs. $2/3$) to affect the classification decision.

13.4 Properties of Naive Bayes

To gain a better understanding of the two models and the assumptions they make, let us go back and examine how we derived their classification rules in Chapters 11 and 12. We decide class membership of a document by assigning it to the class with the maximum a posteriori probability (cf. Section 11.3.2, page 226), which we compute as follows:

$$\begin{aligned} c_{\text{map}} &= \arg\max_{c \in \mathbb{C}} P(c|d) \\ &= \arg\max_{c \in \mathbb{C}} \frac{P(d|c)P(c)}{P(d)} \quad (13.9) \\ &= \arg\max_{c \in \mathbb{C}} P(d|c)P(c), \quad (13.10) \end{aligned}$$

where Bayes' rule (Equation (11.4), page 220) is applied in (13.9) and we drop the denominator in the last step because $P(d)$ is the same for all classes and does not affect the argmax.

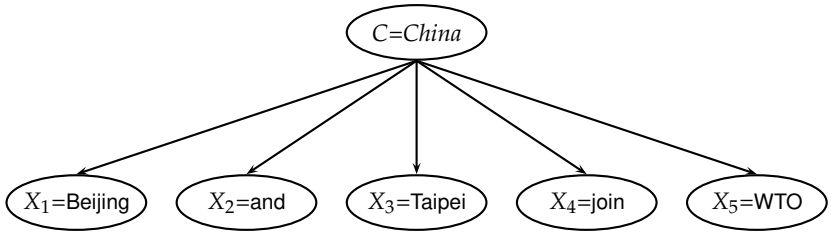
We can interpret Equation (13.10) as a description of the generative process we assume in Bayesian text classification. To generate a document, we first choose class c with probability $P(c)$ (top nodes in Figures 13.4 and 13.5). The two models differ in the formalization of the second step, the generation of the document given the class, corresponding to the conditional distribution $P(d|c)$:

$$\begin{aligned} \textbf{Multinomial} \quad P(d|c) &= P(\langle t_1, \ldots, t_k, \ldots, t_{n_d} \rangle | c) \quad (13.11) \\ \textbf{Bernoulli} \quad P(d|c) &= P(\langle e_1, \ldots, e_i, \ldots, e_M \rangle | c), \quad (13.12) \end{aligned}$$

where $\langle t_1, \ldots, t_{n_d} \rangle$ is the sequence of terms as it occurs in d (minus terms that were excluded from the vocabulary) and $\langle e_1, \ldots, e_i, \ldots, e_M \rangle$ is a binary vector of dimensionality M that indicates for each term whether it occurs in d or not.

It should now be clearer why we introduced the document space $\mathbb{X}$ in Equation (13.1) when we defined the classification problem. A critical step in solving a text classification problem is to choose the document representation. $\langle t_1, \ldots, t_{n_d} \rangle$ and $\langle e_1, \ldots, e_M \rangle$ are two different document representations. In the first case, $\mathbb{X}$ is the set of all term sequences (or, more precisely, sequences of term tokens). In the second case, $\mathbb{X}$ is $\{0, 1\}^M$.

We cannot use Equations (13.11) and (13.12) for text classification directly. For the Bernoulli model, we would have to estimate $2^M|\mathbb{C}|$ different parameters, one for each possible combination of M values e_i and a class. The number of parameters in the multinomial case has the same order of magni-



► **Figure 13.4** The multinomial NB model.

tude.[3] This being a very large quantity, estimating these parameters reliably is infeasible.

CONDITIONAL INDEPENDENCE ASSUMPTION

To reduce the number of parameters, we make the Naive Bayes *conditional independence assumption*. We assume that attribute values are independent of each other given the class:

$$\textbf{Multinomial} \quad P(d|c) = P(\langle t_1, \ldots, t_{n_d}\rangle|c) = \prod_{1\le k\le n_d} P(X_k = t_k|c) \tag{13.13}$$

$$\textbf{Bernoulli} \quad P(d|c) = P(\langle e_1, \ldots, e_M\rangle|c) = \prod_{1\le i\le M} P(U_i = e_i|c). \tag{13.14}$$

RANDOM VARIABLE X

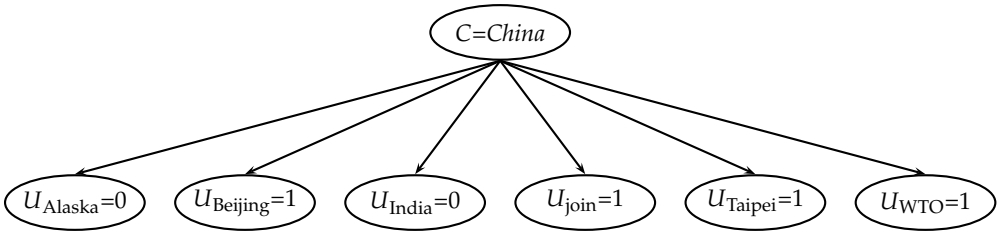
RANDOM VARIABLE U

We have introduced two random variables here to make the two different generative models explicit. X_k is the random variable for position k in the document and takes as values terms from the vocabulary. $P(X_k = t|c)$ is the probability that in a document of class c the term t will occur in position k. U_i is the random variable for vocabulary term i and takes as values 0 (absence) and 1 (presence). $\hat{P}(U_i = 1|c)$ is the probability that in a document of class c the term t_i will occur – in any position and possibly multiple times.

We illustrate the conditional independence assumption in Figures 13.4 and 13.5. The class *China* generates values for each of the five term attributes (multinomial) or six binary attributes (Bernoulli) with a certain probability, independent of the values of the other attributes. The fact that a document in the class *China* contains the term Taipei does not make it more likely or less likely that it also contains Beijing.

In reality, the conditional independence assumption does not hold for text data. Terms *are* conditionally dependent on each other. But as we will discuss shortly, NB models perform well despite the conditional independence assumption.

3. In fact, if the length of documents is not bounded, the number of parameters in the multinomial case is infinite.



► **Figure 13.5** The Bernoulli NB model.

Even when assuming conditional independence, we still have too many parameters for the multinomial model if we assume a different probability distribution for each position k in the document. The position of a term in a document by itself does not carry information about the class. Although there is a difference between *China sues France* and *France sues China*, the occurrence of China in position 1 versus position 3 of the document is not useful in NB classification because we look at each term separately. The conditional independence assumption commits us to this way of processing the evidence.

Also, if we assumed different term distributions for each position k, we would have to estimate a different set of parameters for each k. The probability of bean appearing as the first term of a *coffee* document could be different from it appearing as the second term, and so on. This again causes problems in estimation owing to data sparseness.

POSITIONAL INDEPENDENCE

For these reasons, we make a second independence assumption for the multinomial model, *positional independence*: The conditional probabilities for a term are the same independent of position in the document.

$$P(X_{k_1} = t|c) = P(X_{k_2} = t|c)$$

for all positions k_1, k_2, terms t and classes c. Thus, we have a single distribution of terms that is valid for all positions k_i and we can use X as its symbol.[4] Positional independence is equivalent to adopting the bag of words model, which we introduced in the context of ad hoc retrieval in Chapter 6 (page 117).

With conditional and positional independence assumptions, we only need to estimate $\Theta(M|\mathbb{C}|)$ parameters $P(t_k|c)$ (multinomial model) or $P(e_i|c)$ (Bernoulli

4. Our terminology is nonstandard. The random variable X is a categorical variable, not a multinomial variable, and the corresponding NB model should perhaps be called a *sequence model*. We have chosen to present this sequence model and the multinomial model in Section 13.4.1 as the same model because they are computationally identical.

▶ **Table 13.3** Multinomial versus Bernoulli model.

	multinomial model	Bernoulli model
event model	generation of token	generation of document
random variable(s)	$X = t$ iff t occurs at given pos	$U_t = 1$ iff t occurs in doc
document representation	$d = \langle t_1, \ldots, t_k, \ldots, t_{n_d} \rangle, t_k \in V$	$d = \langle e_1, \ldots, e_i, \ldots, e_M \rangle,$ $e_i \in \{0, 1\}$
parameter estimation	$\hat{P}(X = t\|c)$	$\hat{P}(U_i = e\|c)$
decision rule: maximize	$\hat{P}(c) \prod_{1 \le k \le n_d} \hat{P}(X = t_k\|c)$	$\hat{P}(c) \prod_{t_i \in V} \hat{P}(U_i = e_i\|c)$
multiple occurrences	taken into account	ignored
length of docs	can handle longer docs	works best for short docs
# features	can handle more	works best with fewer
estimate for term the	$\hat{P}(X = \text{the}\|c) \approx 0.05$	$\hat{P}(U_{\text{the}} = 1\|c) \approx 1.0$

model), one for each term–class combination, rather than a number that is at least exponential in M, the size of the vocabulary. The independence assumptions reduce the number of parameters to be estimated by several orders of magnitude.

RANDOM VARIABLE C

To summarize, we generate a document in the multinomial model (Figure 13.4) by first picking a class $C = c$ with $P(c)$ where C is a random variable taking values from $\mathbb{C}$ as values. Next we generate term t_k in position k with $P(X_k = t_k|c)$ for each of the n_d positions of the document. The X_k all have the same distribution over terms for a given c. In the example in Figure 13.4, we show the generation of $\langle t_1, t_2, t_3, t_4, t_5 \rangle = \langle \text{Beijing}, \text{and}, \text{Taipei}, \text{join}, \text{WTO} \rangle$, corresponding to the one-sentence document *Beijing and Taipei join WTO*.

For a completely specified document generation model, we would also have to define a distribution $P(n_d|c)$ over lengths. Without it, the multinomial model is a token generation model rather than a document generation model.

We generate a document in the Bernoulli model (Figure 13.5) by first picking a class $C = c$ with $P(c)$ and then generating a binary indicator e_i for each term t_i of the vocabulary ($1 \le i \le M$). In the example in Figure 13.5, we show the generation of $\langle e_1, e_2, e_3, e_4, e_5, e_6 \rangle = \langle 0, 1, 0, 1, 1, 1 \rangle$, corresponding, again, to the one-sentence document *Beijing and Taipei join WTO* where we have assumed that and is a stop word.

We compare the two models in Table 13.3, including estimation equations and decision rules.

Naive Bayes is so called because the independence assumptions we have just made are indeed very naive for a model of natural language. The conditional independence assumption states that features are independent of each other given the class. This is hardly ever true for terms in documents. In many cases, the opposite is true. The pairs hong and kong or london and en-

► **Table 13.4** Correct estimation implies accurate prediction, but accurate prediction does not imply correct estimation.

	c_1	c_2	class selected
true probability $P(c\|d)$	0.6	0.4	c_1
$\hat{P}(c)\prod_{1\leq k\leq n_d}\hat{P}(t_k\|c)$ (Equation (13.13))	0.00099	0.00001	
NB estimate $\hat{P}(c\|d)$	0.99	0.01	c_1

glish in Figure 13.7 are examples of highly dependent terms. In addition, the multinomial model makes an assumption of positional independence. The Bernoulli model ignores positions in documents altogether because it only cares about absence or presence. This bag-of-words model discards all information that is communicated by the order of words in natural language sentences. How can NB be a good text classifier when its model of natural language is so oversimplified?

The answer is that even though the *probability estimates* of NB are of low quality, its *classification decisions* are surprisingly good. Consider a document d with true probabilities $P(c_1|d) = 0.6$ and $P(c_2|d) = 0.4$ as shown in Table 13.4. Assume that d contains many terms that are positive indicators for c_1 and many terms that are negative indicators for c_2. Thus, when using the multinomial model in Equation (13.13), $\hat{P}(c_1)\prod_{1\leq k\leq n_d}\hat{P}(t_k|c_1)$ will be much larger than $\hat{P}(c_2)\prod_{1\leq k\leq n_d}\hat{P}(t_k|c_2)$ (0.00099 vs. 0.00001 in the table). After division by 0.001 to get well-formed probabilities for $P(c|d)$, we end up with one estimate that is close to 1.0 and one that is close to 0.0. This is common: The winning class in NB classification usually has a much larger probability than the other classes and the estimates diverge very significantly from the true probabilities. But the classification decision is based on which class gets the highest score. It does not matter how accurate the estimates are. Despite the bad estimates, NB estimates a higher probability for c_1 and therefore assigns d to the correct class in Table 13.4. *Correct estimation implies accurate prediction, but accurate prediction does not imply correct estimation.* NB classifiers estimate badly, but often classify well.

Even if it is not the method with the highest accuracy for text, NB has many virtues that make it a strong contender for text classification. It excels if there are many equally important features that jointly contribute to the classification decision. It is also somewhat robust to noise features (as defined in the next section) and *concept drift* – the gradual change over time of the concept underlying a class like *US president* from Bill Clinton to George W. Bush (see Section 13.7). Classifiers like kNN (Section 14.3, page 297) can be carefully tuned to idiosyncratic properties of a particular time period. This will then hurt them when documents in the following time period have slightly

CONCEPT DRIFT

► **Table 13.5** A set of documents for which the NB independence assumptions are problematic.

(1) He moved from London, Ontario, to London, England.
(2) He moved from London, England, to London, Ontario.
(3) He moved from England to London, Ontario.

different properties.

The Bernoulli model is particularly robust with respect to concept drift. We will see in Figure 13.8 that it can have decent performance when using fewer than a dozen terms. The most important indicators for a class are less likely to change. Thus, a model that only relies on these features is more likely to maintain a certain level of accuracy in concept drift.

NB's main strength is its efficiency: Training and classification can be accomplished with one pass over the data. Because it combines efficiency with good accuracy it is often used as a baseline in text classification research. It is often the method of choice if (i) squeezing out a few extra percentage points of accuracy is not worth the trouble in a text classification application, (ii) a very large amount of training data is available and there is more to be gained from training on a lot of data than using a better classifier on a smaller training set, or (iii) if its robustness to concept drift can be exploited.

In this book, we discuss NB as a classifier for text. The independence assumptions do not hold for text. However, it can be shown that NB is an *optimal classifier* (in the sense of minimal error rate on new data) for data where the independence assumptions do hold.

OPTIMAL CLASSIFIER

13.4.1 A variant of the multinomial model

An alternative formalization of the multinomial model represents each document d as an M-dimensional vector of counts $\langle \text{tf}_{t_1,d}, \ldots, \text{tf}_{t_M,d} \rangle$ where $\text{tf}_{t_i,d}$ is the term frequency of t_i in d. $P(d|c)$ is then computed as follows (cf. Equation (12.8), page 243);

$$P(d|c) = P(\langle \text{tf}_{t_1,d}, \ldots, \text{tf}_{t_M,d} \rangle | c) \propto \prod_{1 \leq i \leq M} P(X = t_i|c)^{\text{tf}_{t_i,d}} \tag{13.15}$$

Note that we have omitted the multinomial factor. See Equation (12.8) (page 243).

Equation (13.15) is equivalent to the sequence model in Equation (13.2) as $P(X = t_i|c)^{\text{tf}_{t_i,d}} = 1$ for terms that do not occur in d ($\text{tf}_{t_i,d} = 0$) and a term that occurs $\text{tf}_{t_i,d} \geq 1$ times will contribute $\text{tf}_{t_i,d}$ factors both in Equation (13.2) and in Equation (13.15).

```
SELECTFEATURES(𝔻, c, k)
1  V ← EXTRACTVOCABULARY(𝔻)
2  L ← []
3  for each t ∈ V
4  do A(t, c) ← COMPUTEFEATUREUTILITY(𝔻, t, c)
5     APPEND(L, ⟨A(t, c), t⟩)
6  return FEATURESWITHLARGESTVALUES(L, k)
```

▶ **Figure 13.6** Basic feature selection algorithm for selecting the k best features.

? **Exercise 13.2** [⋆]

Which of the documents in Table 13.5 have identical and different bag of words representations for (i) the Bernoulli model (ii) the multinomial model? If there are differences, describe them.

Exercise 13.3

The rationale for the positional independence assumption is that there is no useful information in the fact that a term occurs in position k of a document. Find exceptions. Consider formulaic documents with a fixed document structure.

Exercise 13.4

Table 13.3 gives Bernoulli and multinomial estimates for the word the. Explain the difference.

13.5 Feature selection

FEATURE SELECTION

Feature selection is the process of selecting a subset of the terms occurring in the training set and using only this subset as features in text classification. Feature selection serves two main purposes. First, it makes training and applying a classifier more efficient by decreasing the size of the effective vocabulary. This is of particular importance for classifiers that, unlike NB, are expensive to train. Second, feature selection often increases classification accuracy by eliminating noise features. A *noise feature* is one that, when added to the document representation, increases the classification error on new data. Suppose a rare term, say arachnocentric, has no information about a class, say *China*, but all instances of arachnocentric happen to occur in *China* documents in our training set. Then the learning method might produce a classifier that misassigns test documents containing arachnocentric to *China*. Such an incorrect generalization from an accidental property of the training set is called *overfitting*.

NOISE FEATURE

OVERFITTING

We can view feature selection as a method for replacing a complex classifier (using all features) with a simpler one (using a subset of the features).

It may appear counterintuitive at first that a seemingly weaker classifier is advantageous in statistical text classification, but when discussing the bias-variance tradeoff in Section 14.6 (page 308), we will see that weaker models are often preferable when limited training data are available.

The basic feature selection algorithm is shown in Figure 13.6. For a given class c, we compute a utility measure $A(t,c)$ for each term of the vocabulary and select the k terms that have the highest values of $A(t,c)$. All other terms are discarded and not used in classification. We will introduce three different utility measures in this section: mutual information, $A(t,c) = I(U_t;C_c)$; the χ^2 test, $A(t,c) = X^2(t,c)$; and frequency, $A(t,c) = N(t,c)$.

Of the two NB models, the Bernoulli model is particularly sensitive to noise features. A Bernoulli NB classifier requires some form of feature selection or else its accuracy will be low.

This section mainly addresses feature selection for two-class classification tasks like *China* versus *not-China*. Section 13.5.5 briefly discusses optimizations for systems with more than two classes.

13.5.1 Mutual information

MUTUAL INFORMATION

A common feature selection method is to compute $A(t,c)$ as the expected *mutual information* (MI) of term t and class c.[5] MI measures how much information the presence/absence of a term contributes to making the correct classification decision on c. Formally:

$$I(U;C) = \sum_{e_t \in \{1,0\}} \sum_{e_c \in \{1,0\}} P(U = e_t, C = e_c) \log_2 \frac{P(U = e_t, C = e_c)}{P(U = e_t)P(C = e_c)}, \tag{13.16}$$

where U is a random variable that takes values $e_t = 1$ (the document contains term t) and $e_t = 0$ (the document does not contain t), as defined on page 266, and C is a random variable that takes values $e_c = 1$ (the document is in class c) and $e_c = 0$ (the document is not in class c). We write U_t and C_c if it is not clear from context which term t and class c we are referring to.

ForMLEs of the probabilities, Equation (13.16) is equivalent to Equation (13.17):

$$\begin{aligned} I(U;C) = {} & \frac{N_{11}}{N} \log_2 \frac{NN_{11}}{N_{1.}N_{.1}} + \frac{N_{01}}{N} \log_2 \frac{NN_{01}}{N_{0.}N_{.1}} \\ & + \frac{N_{10}}{N} \log_2 \frac{NN_{10}}{N_{1.}N_{.0}} + \frac{N_{00}}{N} \log_2 \frac{NN_{00}}{N_{0.}N_{.0}} \end{aligned} \tag{13.17}$$

where the Ns are counts of documents that have the values of e_t and e_c that are indicated by the two subscripts. For example, N_{10} is the number of doc-

5. Take care not to confuse expected mutual information with *pointwise mutual information*, which is defined as $\log N_{11}/E_{11}$ where N_{11} and E_{11} are defined as in Equation (13.18). The two measures have different properties. See Section 13.7.

uments that contain t ($e_t = 1$) and are not in c ($e_c = 0$). $N_{1.} = N_{10} + N_{11}$ is the number of documents that contain t ($e_t = 1$) and we count documents independent of class membership ($e_c \in \{0, 1\}$). $N = N_{00} + N_{01} + N_{10} + N_{11}$ is the total number of documents. An example of one of the MLE estimates that transform Equation (13.16) into Equation (13.17) is $P(U = 1, C = 1) = N_{11}/N$.

Example 13.3: Consider the class *poultry* and the term export in Reuters-RCV1. The counts of the number of documents with the four possible combinations of indicator values are as follows:

	$e_c = e_{poultry} = 1$	$e_c = e_{poultry} = 0$
$e_t = e_{\text{export}} = 1$	$N_{11} = 49$	$N_{10} = 27{,}652$
$e_t = e_{\text{export}} = 0$	$N_{01} = 141$	$N_{00} = 774{,}106$

After plugging these values into Equation (13.17) we get:

$$\begin{aligned} I(U;C) &= \frac{49}{801{,}948} \log_2 \frac{801{,}948 \cdot 49}{(49+27{,}652)(49+141)} \\ &\quad + \frac{141}{801{,}948} \log_2 \frac{801{,}948 \cdot 141}{(141+774{,}106)(49+141)} \\ &\quad + \frac{27{,}652}{801{,}948} \log_2 \frac{801{,}948 \cdot 27{,}652}{(49+27{,}652)(27{,}652+774{,}106)} \\ &\quad + \frac{774{,}106}{801{,}948} \log_2 \frac{801{,}948 \cdot 774{,}106}{(141+774{,}106)(27{,}652+774{,}106)} \\ &\approx 0.0001105 \end{aligned}$$

To select k terms $t_1, \ldots, t_k$ for a given class, we use the feature selection algorithm in Figure 13.6: We compute the utility measure as $A(t, c) = I(U_t, C_c)$ and select the k terms with the largest values.

Mutual information measures how much information – in the information-theoretic sense – a term contains about the class. If a term's distribution is the same in the class as it is in the collection as a whole, then $I(U;C) = 0$. MI reaches its maximum value if the term is a perfect indicator for class membership, that is, if the term is present in a document if and only if the document is in the class.

Figure 13.7 shows terms with high mutual information scores for the six classes in Figure 13.1.[6] The selected terms (e.g., london, uk, british for the class *UK*) are of obvious utility for making classification decisions for their respective classes. At the bottom of the list for *UK* we find terms like peripherals and tonight (not shown in the figure) that are clearly not helpful in deciding

6. Feature scores were computed on the first 100,000 documents, except for *poultry*, a rare class, for which 800,000 documents were used. We have omitted numbers and other special words from the top ten lists.

UK	
london	0.1925
uk	0.0755
british	0.0596
stg	0.0555
britain	0.0469
plc	0.0357
england	0.0238
pence	0.0212
pounds	0.0149
english	0.0126

China	
china	0.0997
chinese	0.0523
beijing	0.0444
yuan	0.0344
shanghai	0.0292
hong	0.0198
kong	0.0195
xinhua	0.0155
province	0.0117
taiwan	0.0108

poultry	
poultry	0.0013
meat	0.0008
chicken	0.0006
agriculture	0.0005
avian	0.0004
broiler	0.0003
veterinary	0.0003
birds	0.0003
inspection	0.0003
pathogenic	0.0003

coffee	
coffee	0.0111
bags	0.0042
growers	0.0025
kg	0.0019
colombia	0.0018
brazil	0.0016
export	0.0014
exporters	0.0013
exports	0.0013
crop	0.0012

elections	
election	0.0519
elections	0.0342
polls	0.0339
voters	0.0315
party	0.0303
vote	0.0299
poll	0.0225
candidate	0.0202
campaign	0.0202
democratic	0.0198

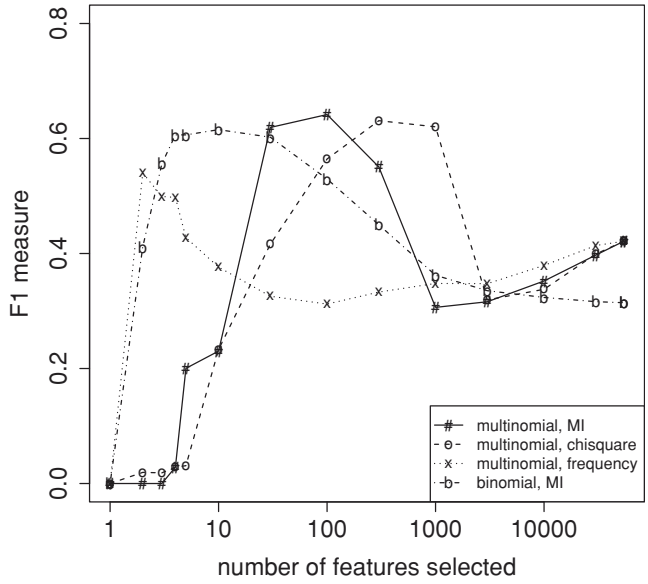
sports	
soccer	0.0681
cup	0.0515
match	0.0441
matches	0.0408
played	0.0388
league	0.0386
beat	0.0301
game	0.0299
games	0.0284
team	0.0264

► **Figure 13.7** Features with high mutual information scores for six Reuters-RCV1 classes.

whether the document is in the class. As you might expect, keeping the informative terms and eliminating the non-informative ones tends to reduce noise and improve the classifier's accuracy.

Such an accuracy increase can be observed in Figure 13.8, which shows F_1 as a function of vocabulary size after feature selection for Reuters-RCV1.[7] Comparing F_1 at 132,776 features (corresponding to selection of all features) and at 10–100 features, we see that MI feature selection increases F_1 by about 0.1 for the multinomial model and by more than 0.2 for the Bernoulli model. For the Bernoulli model, F_1 peaks early, at ten features selected. At that point, the Bernoulli model is better than the multinomial model. When basing a classification decision on only a few features, it is more robust to consider binary occurrence only. For the multinomial model (MI feature selection), the peak occurs later, at 100 features, and its effectiveness recovers somewhat at

7. We trained the classifiers on the first 100,000 documents and computed F_1 on the next 100,000. The graphs are averages over five classes.



► **Figure 13.8** Effect of feature set size on accuracy for multinomial and Bernoulli models.

the end when we use all features. The reason is that the multinomial takes the number of occurrences into account in parameter estimation and classification and therefore better exploits a larger number of features than the Bernoulli model. Regardless of the differences between the two methods, using a carefully selected subset of the features results in better effectiveness than using all features.

13.5.2 χ^2 Feature selection

χ^2 FEATURE SELECTION

INDEPENDENCE

Another popular feature selection method is χ^2. In statistics, the χ^2 test is applied to test the independence of two events, where two events A and B are defined to be *independent* if $P(AB) = P(A)P(B)$ or, equivalently, $P(A|B) = P(A)$ and $P(B|A) = P(B)$. In feature selection, the two events are occurrence of the term and occurrence of the class. We then rank terms with respect to the following quantity:

$$X^2(\mathbb{D}, t, c) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}} \tag{13.18}$$

where e_t and e_c are defined as in Equation (13.16). N is the *observed* frequency in $\mathbb{D}$ and E the *expected* frequency. For example, E_{11} is the expected frequency of t and c occurring together in a document assuming that term and class are independent.

Example 13.4: We first compute E_{11} for the data in Example 13.3:

$$\begin{aligned} E_{11} &= N \times P(t) \times P(c) = N \times \frac{N_{11}+N_{10}}{N} \times \frac{N_{11}+N_{01}}{N} \\ &= N \times \frac{49+141}{N} \times \frac{49+27652}{N} \approx 6.6 \end{aligned}$$

where N is the total number of documents as before.

We compute the other $E_{e_t e_c}$ in the same way:

	$e_{poultry}=1$		$e_{poultry}=0$	
$e_{\text{export}}=1$	$N_{11}=49$	$E_{11}\approx 6.6$	$N_{10}=27{,}652$	$E_{10}\approx 27{,}694.4$
$e_{\text{export}}=0$	$N_{01}=141$	$E_{01}\approx 183.4$	$N_{00}=774{,}106$	$E_{00}\approx 774{,}063.6$

Plugging these values into Equation (13.18), we get a X^2 value of 284:

$$X^2(\mathbb{D},t,c) = \sum_{e_t\in\{0,1\}} \sum_{e_c\in\{0,1\}} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}} \approx 284$$

X^2 is a measure of how much expected counts E and observed counts N deviate from each other. A high value of X^2 indicates that the hypothesis of independence, which implies that expected and observed counts are similar, is incorrect. In our example, $X^2 \approx 284 > 10.83$. Based on Table 13.6, we can reject the hypothesis that *poultry* and export are independent with only a 0.001 chance of being wrong.[8] Equivalently, we say that the outcome $X^2 \approx 284 > 10.83$ is *statistically significant* at the 0.001 level. If the two events are dependent, then the occurrence of the term makes the occurrence of the class more likely (or less likely), so it should be helpful as a feature. This is the rationale of χ^2 feature selection.

STATISTICAL SIGNIFICANCE

An arithmetically simpler way of computing X^2 is the following:

$$X^2(\mathbb{D},t,c) = \frac{(N_{11}+N_{10}+N_{01}+N_{00}) \times (N_{11}N_{00}-N_{10}N_{01})^2}{(N_{11}+N_{01}) \times (N_{11}+N_{10}) \times (N_{10}+N_{00}) \times (N_{01}+N_{00})} \tag{13.19}$$

This is equivalent to Equation (13.18) (Exercise 13.14).

8. We can make this inference because, if the two events are independent, then $X^2 \sim \chi^2$, where χ^2 is the χ^2 distribution. See, for example, Rice (2006).

► **Table 13.6** Critical values of the χ^2 distribution with one degree of freedom. For example, if the two events are independent, then $P(X^2 > 6.63) < 0.01$. So for $X^2 > 6.63$ the assumption of independence can be rejected with 99% confidence.

p	χ^2 critical value
0.1	2.71
0.05	3.84
0.01	6.63
0.005	7.88
0.001	10.83

Assessing χ^2 as a feature selection method

From a statistical point of view, χ^2 feature selection is problematic. For a test with one degree of freedom, the so-called Yates correction should be used (see Section 13.7), which makes it harder to reach statistical significance. Also, whenever a statistical test is used multiple times, then the probability of getting at least one error increases. If 1,000 hypotheses are rejected, each with 0.05 error probability, then $0.05 \times 1000 = 50$ calls of the test will be wrong on average. However, in text classification it rarely matters whether a few additional terms are added to the feature set or removed from it. Rather, the *relative* importance of features is important. As long as χ^2 feature selection only ranks features with respect to their usefulness and is not used to make statements about statistical dependence or independence of variables, we need not be overly concerned that it does not adhere strictly to statistical theory.

13.5.3 Frequency-based feature selection

A third feature selection method is *frequency-based feature selection*, that is, selecting the terms that are most common in the class. Frequency can be either defined as document frequency (the number of documents in the class c that contain the term t) or as collection frequency (the number of tokens of t that occur in documents in c). Document frequency is more appropriate for the Bernoulli model, collection frequency for the multinomial model.

Frequency-based feature selection selects some frequent terms that have no specific information about the class, for example, the days of the week (Monday, Tuesday, ...), which are frequent across classes in newswire text. When many thousands of features are selected, then frequency-based feature selection often does well. Thus, if somewhat suboptimal accuracy is acceptable, then frequency-based feature selection can be a good alternative to more complex methods. However, Figure 13.8 is a case where frequency-

based feature selection performs a lot worse than MI and χ^2 and should not be used.

13.5.4 Feature selection for multiple classifiers

In an operational system with a large number of classifiers, it is desirable to select a single set of features instead of a different one for each classifier. One way of doing this is to compute the X^2 statistic for an $n \times 2$ table where the columns are occurrence and nonoccurrence of the term and each row corresponds to one of the classes. We can then select the k terms with the highest X^2 statistic as before.

More commonly, feature selection statistics are first computed separately for each class on the two-class classification task c versus $\overline{c}$ and then combined. One combination method computes a single figure of merit for each feature, for example, by averaging the values $A(t, c)$ for feature t, and then selects the k features with highest figures of merit. Another frequently used combination method selects the top k/n features for each of n classifiers and then combines these n sets into one global feature set.

Classification accuracy often decreases when selecting k common features for a system with n classifiers as opposed to n different sets of size k. But even if it does, the gain in efficiency owing to a common document representation may be worth the loss in accuracy.

13.5.5 Comparison of feature selection methods

Mutual information and χ^2 represent rather different feature selection methods. The independence of term t and class c can sometimes be rejected with high confidence even if t carries little information about membership of a document in c. This is particularly true for rare terms. If a term occurs once in a large collection and that one occurrence is in the *poultry* class, then this is statistically significant. But a single occurrence is not very informative according to the information-theoretic definition of information. Because its criterion is significance, χ^2 selects more rare terms (which are often less reliable indicators) than mutual information. But the selection criterion of mutual information also does not necessarily select the terms that maximize classification accuracy.

Despite the differences between the two methods, the classification accuracy of feature sets selected with χ^2 and MI does not seem to differ systematically. In most text classification problems, there are a few strong indicators and many weak indicators. As long as all strong indicators and a large number of weak indicators are selected, accuracy is expected to be good. Both methods do this.

Figure 13.8 compares MI and χ^2 feature selection for the multinomial model.

Peak effectiveness is virtually the same for both methods. χ^2 reaches this peak later, at 300 features, probably because the rare, but highly significant features it selects initially do not cover all documents in the class. However, features selected later (in the range of 100–300) are of better quality than those selected by MI.

GREEDY FEATURE SELECTION

All three methods – MI, χ^2 and frequency based – are *greedy* methods. They may select features that contribute no incremental information over previously selected features. In Figure 13.7, kong is selected as the seventh term even though it is highly correlated with previously selected hong and therefore redundant. Although such redundancy can negatively impact accuracy, non-greedy methods (see Section 13.7 for references) are rarely used in text classification due to their computational cost.

? **Exercise 13.5**

Consider the following frequencies for the class *coffee* for four terms in the first 100,000 documents of Reuters-RCV1:

term	N_{00}	N_{01}	N_{10}	N_{11}
brazil	98,012	102	1835	51
council	96,322	133	3525	20
producers	98,524	119	1118	34
roasted	99,824	143	23	10

Select two of these four terms based on (i) χ^2, (ii) mutual information, (iii) frequency.

13.6 Evaluation of text classification

] Historically, the classic Reuters-21578 collection was the main benchmark for text classification evaluation. This is a collection of 21,578 newswire articles, originally collected and labeled by Carnegie Group, Inc. and Reuters, Ltd. in the course of developing the CONSTRUE text classification system. It is much smaller than and predates the Reuters-RCV1 collection discussed in Chapter 4 (page 69). The articles are assigned classes from a set of 118 topic categories. A document may be assigned several classes or none, but the commonest case is single assignment (documents with at least one class received an average of 1.24 classes). The standard approach to this *any-of* problem (Chapter 14, page 306) is to learn 118 two-class classifiers, one for each class, where the *two-class classifier* for class c is the classifier for the two classes c and its complement $\overline{c}$.

TWO-CLASS CLASSIFIER

MODAPTE SPLIT

For each of these classifiers, we can measure recall, precision, and accuracy. In recent work, people almost invariably use the *ModApte split*, which includes only documents that were viewed and assessed by a human indexer,

► **Table 13.7** The ten largest classes in the Reuters-21578 collection with number of documents in training and test sets.

class	# train	# test	class	# train	# test
earn	2877	1087	*trade*	369	119
acquisitions	1650	179	*interest*	347	131
money-fx	538	179	*ship*	197	89
grain	433	149	*wheat*	212	71
crude	389	189	*corn*	182	56

and comprises 9,603 training documents and 3,299 test documents. The distribution of documents in classes is very uneven, and some work evaluates systems on only documents in the ten largest classes. They are listed in Table 13.7. A typical document with topics is shown in Figure 13.9.

In Section 13.1, we stated as our goal in text classification the minimization of classification error on test data. Classification error is 1.0 minus classification accuracy, the proportion of correct decisions, a measure we introduced in Section 8.3 (page 155). This measure is appropriate if the percentage of documents in the class is high, perhaps 10% to 20% and higher. But as we discussed in Section 8.3, accuracy is not a good measure for "small" classes because always saying no, a strategy that defeats the purpose of building a classifier, will achieve high accuracy. The always-no classifier is 99% accurate for a class with relative frequency 1%. For small classes, precision, recall and F_1 are better measures.

EFFECTIVENESS

We will use *effectiveness* as a generic term for measures that evaluate the quality of classification decisions, including precision, recall, F_1, and accuracy. *Performance* refers to the computational efficiency of classification and IR systems in this book. However, many researchers mean effectiveness, not efficiency of text classification when they use the term performance.

PERFORMANCE
EFFICIENCY

When we process a collection with several two-class classifiers (such as Reuters-21578 with its 118 classes), we often want to compute a single aggregate measure that combines the measures for individual classifiers. There are two methods for doing this. *Macroaveraging* computes a simple average over classes. *Microaveraging* pools per-document decisions across classes, and then computes an effectiveness measure on the pooled contingency table. Table 13.8 gives an example.

MACROAVERAGING
MICROAVERAGING

The differences between the two methods can be large. Macroaveraging gives equal weight to each class, whereas microaveraging gives equal weight to each per-document classification decision. Because the F_1 measure ignores true negatives and its magnitude is mostly determined by the number of true positives, large classes dominate small classes in microaveraging. In the example, microaveraged precision (0.83) is much closer to the precision of

```
<REUTERS TOPICS=''YES'' LEWISSPLIT=''TRAIN''
CGISPLIT=''TRAINING-SET'' OLDID=''12981'' NEWID=''798''>
<DATE> 2-MAR-1987 16:51:43.42</DATE>
<TOPICS><D>livestock</D><D>hog</D></TOPICS>
<TITLE>AMERICAN PORK CONGRESS KICKS OFF TOMORROW</TITLE>
<DATELINE> CHICAGO, March 2 - </DATELINE><BODY>The American Pork
Congress kicks off tomorrow, March 3, in Indianapolis with 160
of the nations pork producers from 44 member states determining
industry positions on a number of issues, according to the
National Pork Producers Council, NPPC.
Delegates to the three day Congress will be considering 26
resolutions concerning various issues, including the future
direction of farm policy and the tax law as it applies to the
agriculture sector. The delegates will also debate whether to
endorse concepts of a national PRV (pseudorabies virus) control
and eradication program, the NPPC said. A large
trade show, in conjunction with the congress, will feature
the latest in technology in all areas of the industry, the NPPC
added. Reuter
\&\#3;</BODY></TEXT></REUTERS>
```

► **Figure 13.9** A sample document from the Reuters-21578 collection.

c_2 (0.9) than to the precision of c_1 (0.5) because c_2 is five times larger than c_1. Microaveraged results are therefore really a measure of effectiveness on the large classes in a test collection. To get a sense of effectiveness on small classes, you should compute macroaveraged results.

In one-of classification (Section 14.5, page 306), microaveraged F_1 is the same as accuracy (Exercise 13.6).

Table 13.9 gives microaveraged and macroaveraged effectiveness of Naive Bayes for the ModApte split of Reuters-21578. To give a sense of the relative effectiveness of NB, we compare it with linear SVMs (rightmost column; see Chapter 15), one of the most effective classifiers, but also one that is more expensive to train than NB. NB has a microaveraged F_1 of 80%, which is 9% less than the SVM (89%), a 10% relative decrease (row "micro-avg-L (90 classes)"). So there is a surprisingly small effectiveness penalty for its simplicity and efficiency. However, on small classes, some of which only have on the order of ten positive examples in the training set, NB does much worse. Its macroaveraged F_1 is 13% below the SVM, a 22% relative decrease (row "macro-avg (90 classes)").

The table also compares NB with the other classifiers we cover in this book:

► **Table 13.8** Macro- and microaveraging. "Truth" is the true class and "call" the decision of the classifier. In this example, macroaveraged precision is $[10/(10+10)+90/(10+90)]/2 = (0.5+0.9)/2 = 0.7$. Microaveraged precision is $100/(100+20) \approx 0.83$.

class 1

	truth: yes	truth: no
call: yes	10	10
call: no	10	970

class 2

	truth: yes	truth: no
call: yes	90	10
call: no	10	890

pooled table

	truth: yes	truth: no
call: yes	100	20
call: no	20	1860

► **Table 13.9** Text classification effectiveness numbers on Reuters-21578 for F_1 (in percent). Results from Li and Yang (2003) (a), Joachims (1998) (b: kNN) and Dumais et al. (1998) (b: NB, Rocchio, trees, SVM).

(a)

	NB	Rocchio	kNN	SVM
micro-avg-L (90 classes)	80	85	86	89
macro-avg (90 classes)	47	59	60	60

(b)

	NB	Rocchio	kNN	trees	SVM
earn	96	93	97	98	98
acq	88	65	92	90	94
money-fx	57	47	78	66	75
grain	79	68	82	85	95
crude	80	70	86	85	89
trade	64	65	77	73	76
interest	65	63	74	67	78
ship	85	49	79	74	86
wheat	70	69	77	93	92
corn	65	48	78	92	90
micro-avg (top 10)	82	65	82	88	92
micro-avg-D (118 classes)	75	62	n/a	n/a	87

DECISION TREES

Rocchio and kNN. In addition, we give numbers for *decision trees*, an important classification method we do not cover. The bottom part of the table shows that there is considerable variation from class to class. For instance, NB beats kNN on *ship*, but is much worse on *money-fx*.

Comparing parts (a) and (b) of the table, one is struck by the degree to which the cited papers' results differ. This is partly due to the fact that the numbers in (b) are break-even scores (cf. page 161) averaged over 118 classes, whereas the numbers in (a) are true F_1 scores (computed without any know-

ledge of the test set) averaged over ninety classes. This is unfortunately typical of what happens when comparing different results in text classification: There are often differences in the experimental setup or the evaluation that complicate the interpretation of the results.

These and other results have shown that the average effectiveness of NB is uncompetitive with classifiers like SVMs when trained and tested on *independent and identically distributed* (*i.i.d.*) data, that is, uniform data with all the good properties of statistical sampling. However, these differences may often be invisible or even reverse themselves when working in the real world where, usually, the training sample is drawn from a subset of the data to which the classifier will be applied, the nature of the data drifts over time rather than being stationary (the problem of concept drift we mentioned on page 269), and there may well be errors in the data (among other problems). Many practitioners have had the experience of being unable to build a fancy classifier for a certain problem that consistently performs better than NB.

Our conclusion from the results in Table 13.9 is that, although most researchers believe that an SVM is better than kNN and kNN better than NB, the ranking of classifiers ultimately depends on the class, the document collection, and the experimental setup. In text classification, there is always more to know than simply which machine learning algorithm was used, as we further discuss in Section 15.3 (page 334).

When performing evaluations like the one in Table 13.9, it is important to maintain a strict separation between the training set and the test set. We can easily make correct classification decisions on the test set by using information we have gleaned from the test set, such as the fact that a particular term is a good predictor in the test set (even though this is not the case in the training set). A more subtle example of using knowledge about the test set is to try a large number of values of a parameter (e.g., the number of selected features) and select the value that is best for the test set. As a rule, accuracy on new data – the type of data we will encounter when we use the classifier in an application – will be much lower than accuracy on a test set that the classifier has been tuned for. We discussed the same problem in ad hoc retrieval in Section 8.1 (page 153).

In a clean statistical text classification experiment, you should never run any program on or even look at the test set while developing a text classification system. Instead, set aside a *development set* for testing while you develop your method. When such a set serves the primary purpose of finding a good value for a parameter, for example, the number of selected features, then it is also called *held-out data*. Train the classifier on the rest of the training set with different parameter values, and then select the value that gives best results on the held-out part of the training set. Ideally, at the very end, when all parameters have been set and the method is fully specified, you run one final experiment on the test set and publish the results. Because no informa-

DEVELOPMENT SET

HELD-OUT DATA

► **Table 13.10** Data for parameter estimation exercise.

	docID	words in document	in $c = China$?
training set	1	Taipei Taiwan	yes
	2	Macao Taiwan Shanghai	yes
	3	Japan Sapporo	no
	4	Sapporo Osaka Taiwan	no
test set	5	Taiwan Taiwan Sapporo	?

tion about the test set was used in developing the classifier, the results of this experiment should be indicative of actual performance in practice.

This ideal often cannot be met; researchers tend to evaluate several systems on the same test set over a period of several years. But it is nevertheless highly important to not look at the test data and to run systems on it as sparingly as possible. Beginners often violate this rule, and their results lose validity because they have implicitly tuned their system to the test data simply by running many variant systems and keeping the tweaks to the system that worked best on the test set.

?

Exercise 13.6 [⋆⋆]

Assume a situation where every document in the test collection has been assigned exactly one class, and that a classifier also assigns exactly one class to each document. This setup is called one-of classification (Section 14.5, page 306). Show that in one-of classification (i) the total number of false positive decisions equals the total number of false negative decisions and (ii) microaveraged F_1 and accuracy are identical.

Exercise 13.7

The class priors in Figure 13.2 are computed as the fraction of *documents* in the class as opposed to the fraction of *tokens* in the class. Why?

Exercise 13.8

The function APPLYMULTINOMIALNB in Figure 13.2 has time complexity $\Theta(L_a + |\mathbb{C}|L_a)$. How would you modify the function so that its time complexity is $\Theta(L_a + |\mathbb{C}|M_a)$?

Exercise 13.9

Based on the data in Table 13.10, (i) estimate a multinomial Naive Bayes classifier, (ii) apply the classifier to the test document, (iii) estimate a Bernoulli NB classifier, (iv) apply the classifier to the test document. You need not estimate parameters that you don't need for classifying the test document.

Exercise 13.10

Your task is to classify words as English or not English. Words are generated by a source with the following distribution:

event	word	English?	probability
1	ozb	no	4/9
2	uzu	no	4/9
3	zoo	yes	1/18
4	bun	yes	1/18

(i) Compute the parameters (priors and conditionals) of a multinomial NB classifier that uses the letters b, n, o, u, and z as features. Assume a training set that reflects the probability distribution of the source perfectly. Make the same independence assumptions that are usually made for a multinomial classifier that uses terms as features for text classification. Compute parameters using smoothing, in which computed-zero probabilities are smoothed into probability 0.01, and computed-nonzero probabilities are untouched. (This simplistic smoothing may cause $P(A) + P(\overline{A}) > 1$. Solutions are not required to correct this.) (ii) How does the classifier classify the word zoo? (iii) Classify the word zoo using a multinomial classifier as in part (i), but do not make the assumption of positional independence. That is, estimate separate parameters for each position in a word. You only need to compute the parameters you need for classifying zoo.

Exercise 13.11

What are the values of $I(U_t; C_c)$ and $X^2(\mathbb{D}, t, c)$ if term and class are completely independent? What are the values if they are completely dependent?

Exercise 13.12

The feature selection method in Equation (13.16) is most appropriate for the Bernoulli model. Why? How could one modify it for the multinomial model?

Exercise 13.13

INFORMATION GAIN

Features can also be selected according to *information gain* (IG), which is defined as:

$$\text{IG}(\mathbb{D}, t, c) = H(p_{\mathbb{D}}) - \sum_{x \in \{\mathbb{D}_{t^+}, \mathbb{D}_{t^-}\}} \frac{|x|}{|\mathbb{D}|} H(p_x)$$

where H is entropy, $\mathbb{D}$ is the training set, and $\mathbb{D}_{t^+}$, and $\mathbb{D}_{t^-}$ are the subset of $\mathbb{D}$ with term t, and the subset of $\mathbb{D}$ without term t, respectively. p_A is the class distribution in (sub)collection A, e.g., $p_A(c) = 0.25$, $p_A(\overline{c}) = 0.75$ if a quarter of the documents in A are in class c.

Show that mutual information and information gain are equivalent.

Exercise 13.14

Show that the two X^2 formulas (Equations (13.18) and (13.19)) are equivalent.

Exercise 13.15

In the χ^2 example on page 276 we have $|N_{11} - E_{11}| = |N_{10} - E_{10}| = |N_{01} - E_{01}| = |N_{00} - E_{00}|$. Show that this holds in general.

Exercise 13.16

χ^2 and mutual information do not distinguish between positively and negatively correlated features. Because most good text classification features are positively correlated (i.e., they occur more often in c than in $\overline{c}$), one may want to explicitly rule out the selection of negative indicators. How would you do this?

13.7 References and further reading

General introductions to statistical classification and machine learning can be found in (Hastie et al. 2001), (Mitchell 1997), and (Duda et al. 2000), including many important methods (e.g., decision trees and boosting) that we do not cover. A comprehensive review of text classification methods and results is (Sebastiani 2002). Manning and Schütze (1999, Chapter 16) give an accessible introduction to text classification with coverage of decision trees, perceptrons and maximum entropy models. More information on the superlinear time complexity of learning methods that are more accurate than Naive Bayes can be found in (Perkins et al. 2003) and (Joachims 2006a).

Maron and Kuhns (1960) described one of the first NB text classifiers. Lewis (1998) focuses on the history of NB classification. Bernoulli and multinomial models and their accuracy for different collections are discussed by McCallum and Nigam (1998). Eyheramendy et al. (2003) present additional NB models. Domingos and Pazzani (1997), Friedman (1997), and Hand and Yu (2001) analyze why NB performs well although its probability estimates are poor. The first paper also discusses NB's optimality when the independence assumptions are true of the data. Pavlov et al. (2004) propose a modified document representation that partially addresses the inappropriateness of the independence assumptions. Bennett (2000) attributes the tendency of NB probability estimates to be close to either 0 or 1 to the effect of document length. Ng and Jordan (2001) show that NB is sometimes (although rarely) superior to discriminative methods because it more quickly reaches its optimal error rate. The basic NB model presented in this chapter can be tuned for better effectiveness (Rennie et al. 2003;Kołcz and Yih 2007). The problem of concept drift and other reasons why state-of-the-art classifiers do not always excel in practice are discussed by Forman (2006) and Hand (2006).

Early uses of mutual information and χ^2 for feature selection in text classification are Lewis and Ringuette (1994) and Schütze et al. (1995), respectively. Yang and Pedersen (1997) review feature selection methods and their impact on classification effectiveness. They find that *pointwise mutual information* is not competitive with other methods. Yang and Pedersen refer to expected mutual information (Equation (13.16)) as information gain (see Exercise 13.13, page 285). (Snedecor and Cochran 1989) is a good reference for the χ^2 test in statistics, including the Yates' correction for continuity for 2×2 tables. Dunning (1993) discusses problems of the χ^2 test when counts are small. Nongreedy feature selection techniques are described by Hastie et al. (2001). Cohen (1995) discusses the pitfalls of using multiple significance tests and methods to avoid them. Forman (2004) evaluates different methods for feature selection for multiple classifiers.

POINTWISE MUTUAL INFORMATION

David D. Lewis defines the ModApte split at www.daviddlewis.com/resources/testcollections/reuters215 based on Apté et al. (1994). Lewis (1995) describes *utility measures* for the

UTILITY MEASURE

evaluation of text classification systems. Yang and Liu (1999) employ significance tests in the evaluation of text classification methods.

Lewis et al. (2004) find that SVMs (Chapter 15) perform better on Reuters-RCV1 than kNN and Rocchio (Chapter 14).