13 Text classification and Naive Bayes

Up until now, this book has principally discussed the process of ad-hoc retrieval where a user has a transient information need, which they try to address by posing one or more queries to a search engine. However, many users have ongoing information needs. For example, because of my job in the computer industry, I might need to track developments in multicore computer chips. One way of doing this is for me to issue the query multicore AND computer AND chip against an index of recent newswire articles each morning.

Rather than doing this by hand, it makes sense if this task can be automated. Therefore, many systems support having *standing queries* which can be run against new documents.

If my standing query is just multicore AND computer AND chip, I will tend to miss many relevant new articles which use other terms such as multicore processors. To achieve good recall, standing queries tend to be refined over time and to become quite complex. For example, for this information need, using a boolean search engine with stemming, I might end up with a query like (multicore OR multi-core) AND (chip OR processor OR microprocessor). However, rather than having to painstakingly build up such a standing query by hand, there might be better ways to address a long-term information need. For example, I might be able to just give the system some examples of news articles that are and aren’t relevant to my interests, and have it determine which documents I need to see. To be able to better appreciate the range of approaches and also the many applications of the technologies that we introduce in the next several chapters, we will introduce the general notion of a classification problem. In this example, I am wanting do do two-class classification: dividing all new newswire articles into the two classes ‘documents that talk about multicore computer chips’ and ‘documents not about multicore computer chips’.

In many text-processing applications, we wish to determine whether a document is in one of several pre-determined classes, which might be subject areas like China or coffee. These classes are usually referred to as topics, and the task is then called topic classification. An example appears in Fig-
In order to do topic classification, we need to build a topic classifier which can pick out documents on each topic of interest. A topic classifier for China will determine for each document whether it is relevant to China or not by looking for terms like China, Chinese and Beijing. These terms are more common in China than in non-China documents. The classification task is to determine such differences between the classes and to exploit them for assigning a document to its correct class.

A second application of topic classifiers is to help build a vertical search engine. Vertical search engines restrict searches to a particular topic. For example, the query computer science on a vertical search engine for the topic China might be expected to 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 word china in a different sense (e.g., referring to a hard white ceramic), but does include relevant pages even if they don’t explicitly mention the terms China or Chinese.

The notion of classifiers is very general. There are many applications of classification technology within and beyond information retrieval. Other IR examples include:

- Several of the preprocessing steps necessary for indexing as discussed in Chapter 2: detecting a document’s encoding (ASCII, Unicode UTF-8 etc; page 17); word segmentation (Is the gap between two letters a word boundary or not? page 22); truecasing (page 26); and identifying the language of a document (page 36)
- 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)

While the classification task we will use as an example in this book is topic classification, this list shows the general importance of classification in information retrieval. Most retrieval systems today contain multiple components that use some form of classifier.

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. Our initial example of a standing query illustrated the first alternative approach: classification by the use of rules, most commonly written by hand. Here are some rules one might use for the class UK:

- London $\rightarrow$ UK
- Queen AND NOT (freddie OR mercury) $\rightarrow$ UK
- Buckingham AND palace $\rightarrow$ UK
A rule captures a certain combination of keywords that indicates a class. Hand-coded rules have good scaling properties, but creating the rules 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 in accuracy the automatically generated classifiers we will discuss shortly. But it can be hard to find someone with this specialized skill.

The third approach to text classification is based on machine learning and it is the approach that we focus on in this book. In machine learning, the set of rules or, more generally, the decision criterion of the topic classifier is learned automatically from training data. This approach is also called statistical text classification if the learning algorithm is statistical. In statistical text classification, we require a number of good example documents (or training documents) from each class. The need for manual classification is not eliminated since the training documents come from a person who has labeled them. But labeling documents is arguably an easier task than writing rules. Almost anybody can look at a document and decide whether or not it is about the geographic region China. Sometimes such labeling is already implicitly part of an existing workflow, for instance, if it is a staffer’s job to pick out relevant news articles for a senator.

We begin this chapter with a general introduction to the text classification problem including a formal definition (Section 13.1); we then cover Naïve Bayes, a particularly simple and effective classification method (Sections 13.2–13.4). All of the classification algorithms we study view documents as vectors 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 classification. Section 13.5 introduces two of the most widely used feature selection techniques. Section 13.6 covers evaluation of text classification. In the following two chapters, Chapters 14 and 15, we look at two other classification methods, vector space classification and support vector machines.

### 13.1 The text classification problem

In text classification, we are given a description of an instance \( x \in X \), where \( X \) is the instance space; and a fixed set of classes (or, equivalently, categories) \( C = \{c_1, c_2, \ldots, c_J\} \). The instances are typically documents and the classes are typically human-defined for the needs of an application, as in the China example above. Given one or more training documents for each of the \( J \) classes, we wish to learn a classification function \( \gamma \) that maps instances to classes:

\[
\gamma : X \rightarrow C
\]
This type of learning is called \textit{supervised learning} since a "supervisor" (the human who defines the classes and labels training documents for each class) functions as a teacher directing the learning process.

Figure 13.1 shows an example of text classification from the Reuters-RCV1 collection, introduced in Section 4.1, page 52. There are six classes (UK, China, ..., sports), each with three training documents. We show a few mnemonic words for each document's content. The \textit{training data} provide some typical examples for each class, so that we can learn $\gamma$. Once we have learned it, we can apply $\gamma$ to \textit{test data}, for example the new document \textit{China's first private airline} whose class is unknown. The classification function assigns the new document to class $\gamma(d) = \text{China}$, which is the correct assignment in this case.

The classes often have some interesting structure such as the hierarchy in Figure 13.1. There are two instances each of region categories, industry categories and topic categories. A hierarchy can be an important aid in solving a classification problem (see Section 13.7). But we will make the simplifying assumption here that the classes form a set with no relationship between the classes.

\subsection*{13.2 Naive Bayes text classification}

\textbf{Naive Bayes} \quad The first machine learning method we look at is \textit{Naive Bayes} (often abbrevi-
13.2 Naive Bayes text classification

As indicated by its name, it is a Bayesian method. A generative model (Chapter 12, page 181) – a model that generates documents – is associated with each class. By comparing the text in a document \( d \) to the text that would be generated by the model associated with a class \( c_j \), we compute an estimate of the likelihood that \( d \) belongs to \( c_j \). Bayesian text classification provides a generative model that allows us to infer from the text in \( d \) the probability \( P(c_j|d) \) of being in each class \( c_j \), and hence its unknown (or hidden) class.

To perform this inference, we first define the prior class probability or prior \( P(c_j) \) of a class \( c_j \) as the probability that document \( d \) is in \( c_j \), if we knew nothing about the text in \( d \). (How do we compute the value of \( P(c_j) \)? This will be addressed shortly.) By multiplying \( P(c_j) \) by the probability \( P(d|c_j) \) that \( d \) was generated by \( c_j \), we get (a quantity proportional to) the posterior probability \( P(c_j|d) \). The posterior probability is the probability of class membership after we have taken into account the observable evidence. A classification decision is made by assigning the document to the class with the highest posterior probability. This classification principle is called maximum a-posteriori or MAP. It can be derived as follows.

\[
\epsilon_{\text{map}} = \arg\max_{c_j \in C} P(c_j|d)
\]

\[
= \arg\max_{c_j \in C} \frac{P(d|c_j)P(c_j)}{P(d)}
\]

\[
= \arg\max_{c_j \in C} P(d|c_j)P(c_j)
\]

where Bayes’ Rule (Equation (11.4), page 168) is applied in 13.2. We can drop the denominator in the last step since \( P(d) \) is the same for all classes and does not affect the argmax.

We have to define the instance space \( X \) in which documents are represented. The generic representation we will start with is a tuple of attributes or features \( x_i \); \( d = (x_1, x_2, \ldots, x_n) \). With this notation, 13.3 becomes:

\[
\epsilon_{\text{map}} = \arg\max_{c_j \in C} P(\langle x_1, x_2, \ldots, x_n \rangle|c_j)P(c_j)
\]

To determine the MAP class, we need to estimate the parameters of the model: the priors and the conditional probabilities. Unfortunately, the conditional probabilities in Equation 13.4 cannot be estimated reliably. Even if the \( n \) attributes were binary, our Bayesian model would have \( 2^n|C| \) different parameters, one for each possible combination of \( n \) attribute values and a class. This being a very large quantity, estimating these parameters reliably is challenging unless our training collection is astronomical in size.
To reduce the number of parameters, we make what is called the Naive Bayes Conditional Independence Assumption. We assume that attributes are independent of each other given the class:

$$P(\langle x_1, x_2, \ldots, x_n \rangle | c_j) = \prod_{1 \leq i \leq n} P(x_i | c_j)$$

The maximization we perform to determine the class then becomes:

$$c_{\text{map}} = \arg \max_{c_j \in C} P(c_j) \prod_{1 \leq i \leq n} P(x_i | c_j)$$ (13.5)

Now we switch from a generic instance space $X$ to the one typical of Naive Bayes in text classification where each attribute corresponds to a word$^1$ – and is then more commonly called a feature. The conditional independence assumption can be illustrated with the graphical model in Figure 13.2. The class generates each of the five word attributes with a certain probability, independent of the absence or presence of the other words. The fact that a document in the class China contains the word Taipei does not make it more likely or less likely that it also contains Beijing. Of course this assumption does not really hold for text data. Words are conditionally dependent on each other. But NB models perform well despite the conditional independence assumption. Equation 13.5 can now be rewritten as follows, letting $w_i$ denote the $i$th word in the role of the $i$th feature:

$$c_{\text{map}} = \arg \max_{c_j \in C} P(c_j) \prod_{1 \leq i \leq n} P(w_i | c_j)$$ (13.6)

$^1$ It would be more accurate to talk about terms, not words since text classification systems often preprocess text using the same normalization procedures as indexers (such as downcasing and stemming). We follow convention in the three text classification chapters and use the terms word and term interchangeably.
With the independence assumption, we only need to estimate $O(n|C|)$ independent parameters $P(w_i|c_j)$, one for each word-class combination, rather than $O(2^n|C|)$. The independence assumption reduces the number of parameters to be estimated by several orders of magnitude.

How do we perform the estimation? We first try the maximum likelihood estimate (MLE), which corresponds to the most likely value of each parameter given the training data. For the priors this estimate is:

\[
\hat{P}(c_j) = \frac{N_j}{N}
\]

where $N_j$ is the number of documents of class $c_j$ and $N$ is the total number of documents.

To estimate the conditional probabilities, we first need to settle on a model of document generation. We choose the multinomial model here. Below we discuss a second model, the binomial model. The multinomial model was introduced for adhoc retrieval in Chapter 12: Equations (12.3) and (12.4) (page 185). There we estimated the probability of a document model ($M_d$) generating a query word ($q$). Here we want the probability of a class model generating a document word (or, more precisely, a word token). The MLE estimate for the conditional probabilities $\hat{P}(w_i|c_j)$ is then:

\[
\hat{P}(w_i|c_j) = \frac{T_{ji}}{\sum_j T_{ji}}
\]

where $T_{ji}$ is the number of occurrences of $w_i$ in training documents from class $c_j$, including multiple occurrences of a term in a document.

The problem with the MLE estimate is that it is zero for a word-class combination that did not occur in the training data. In the model in Figure 13.2, if occurrences of the word WTO in the training data only occurred in China documents, then the MLE estimates for the other classes, for example UK, will be zero:

\[
\hat{P}(\text{WTO}|\text{UK}) = 0
\]

Now a document consisting of the sentence Britain is a member of the WTO will get a conditional probability of zero for UK since we’re multiplying the conditional probabilities for all words in Equation 13.5. Clearly, the model should assign a high probability to the UK class since the word 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. This is the problem of data sparseness. We encountered the same problem when applying the multinomial model to adhoc retrieval in Chapter 12 (page 185).

To eliminate zeros, we use add-one or Laplace smoothing, which simply

---

Preliminary draft (c) 2007 Cambridge UP
adds one to each count:
\[ \hat{P}(w_i|c_j) = \frac{T_{ji} + 1}{\sum_i(T_{ji} + 1)} \]

Laplace smoothing can be interpreted as a uniform prior (each word occurs once for each class) that is then updated as evidence from the training data comes in.²

We have left the exact definition of the \( w_i \) in \( \hat{P}(w_i|c_j) \) vague so far. The index \( i \) might suggest that we have different random variable \( X_i \) for each position in the document. But the position of a word in a document by itself does not carry information about the class. There is a difference between China sues France and France sues China, but the occurrence of China in position 1 vs. 3 of the document is not useful for classification since we look at each word separately in classification. This way of processing the evidence is what we committed to when we made the conditional independence assumption.

Also, with different random variable \( X_i \) for each position \( i \), we would have to estimate a different set of parameters for each \( i \). The probability of bean appearing as the first word of a coffee document would be different from it appearing as the second word etc. This would again cause problems with data sparseness in estimation.

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

\[ P(X_{k_1} = w|c_j) = P(X_{k_2} = w|c_j) \]

for all positions \( k_1, k_2 \), words \( w \) and classes \( c_j \). \( X_k \) is a random variable whose values are words appearing in position \( k \) of the document. Thus, we have a single multinomial distribution of words that is valid for all positions \( k \) and can use \( X \) as the symbol for its random variable. Positional independence is equivalent to adopting the bag of words model, which we introduced in the context of adhoc retrieval in Chapter 6 (page 90).

We have now introduced all the elements we need for training and applying an NB classifier. The complete algorithm is described in Figure 13.3.

What is the time complexity of Naive Bayes? The complexity of computing the parameters is \( O(|C||V|) \) since the set of parameters consists of \( |C||V| \) conditional probabilities and \( |C| \) priors. The preprocessing necessary for computing the parameters (extracting the vocabulary, counting words etc.) can be done in one pass through the training data. The time complexity of this component is therefore \( O(|D|L_d) \) where \(|D|\) is the number of documents and

---

² Note that this is a prior probability for the occurrence of a word as opposed to the prior probability of a class which we estimate in Equation (13.7) on the document level.
13.2 Naive Bayes text classification

**Training**

From training data $D$, extract vocabulary $V$

$N \leftarrow \text{number of documents in } D$

Calculate parameters $P(c_j)$ and $P(X = w_i | c_j)$

For each $c_j$ in $C$

$N_j \leftarrow \text{number of documents in } c_j$

$\hat{P}(c_j) \leftarrow \frac{N_j}{N}$

$text_j \leftarrow \text{the text of all documents in class } c_j$

For each word $w_i \in V$:

$T_{ji} \leftarrow \text{number of occurrences of } w_i \text{ in } text_j$

$\hat{P}(X = w_i | c_j) = \frac{T_{ji} + 1}{\sum (T_{ji} + 1)}$

**Testing**

Positions $S \leftarrow \text{all positions in current document that contain words in } V$

Return $c_{NB}$, where

$c_{NB} = \arg \max_{c_j \in C} P(c_j) \prod_{k \in S} P(X = w_k | c_j)$

◮ Figure 13.3 Naive Bayes algorithm (multinominal model): Training and testing

<table>
<thead>
<tr>
<th>mode</th>
<th>time complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>training</td>
<td>$O(</td>
</tr>
<tr>
<td>testing</td>
<td>$O(</td>
</tr>
</tbody>
</table>

◮ Table 13.1 Training and test times for Naive Bayes.

$L_d$ is the average length of a document. The time complexity of classifying a document is $O(|C|L_t)$ (where $L_t$ is the average length of a test document) as can be observed from Figure 13.3.

Table 13.1 summarizes the time complexities. In general, we have $|C||V| < |D|L_d$, so both training and testing complexity is linear in the time it takes to scan the data. Since we have to look at the data at least once, Naive Bayes can be said to have optimal time complexity. Its efficiency is one reason why Naive Bayes is such a popular text classification algorithm.

**Implementation note.** In NB classification, many conditional probabilities are multiplied, one for each vocabulary word in the document. 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 since $\log(xy) = \log(x) + \log(y)$ and the logarithm function is monotonic. Hence, the maximization that is actually done in most implementations of
<table>
<thead>
<tr>
<th>event model</th>
<th>multinomial model</th>
<th>binomial model</th>
</tr>
</thead>
<tbody>
<tr>
<td>random variable(s)</td>
<td>generation of token</td>
<td>generation of document</td>
</tr>
<tr>
<td>document representation</td>
<td>$X = w$ iff $w$ occurs at given pos</td>
<td>$U_w = 1$ iff $w$ occurs in doc</td>
</tr>
<tr>
<td>parameter estimation</td>
<td>$d = &lt;w_1, w_2, \ldots, w_{</td>
<td>d</td>
</tr>
<tr>
<td>decision rule: maximize multiple occurrences</td>
<td>$P(X = w</td>
<td>c_j)$</td>
</tr>
<tr>
<td>length of docs</td>
<td>$\hat{P}(c_j) \prod_{k \in S} P(X = w_k</td>
<td>c_j)$</td>
</tr>
<tr>
<td># features</td>
<td>taken into account</td>
<td>ignored</td>
</tr>
<tr>
<td>estimate for term the</td>
<td>can handle long docs</td>
<td>works best for short docs</td>
</tr>
<tr>
<td></td>
<td>can handle many</td>
<td>works best with few</td>
</tr>
<tr>
<td></td>
<td>$\hat{P}(X = \text{the}</td>
<td>c_j) \approx 0.05$</td>
</tr>
</tbody>
</table>

Table 13.2 Multinomial vs. Binomial model.

Naive Bayes is:

$$c_{NB} = \arg \max_{c_j \in C} \log P(c_j) + \sum_{k \in S} \log P(X = w_k | c_j)$$

### 13.3 The multinomial versus the binomial model

As we already mentioned there are two different ways we can set up an NB classifier. The model we have worked with so far is the **multinomial model**. It generates one word from the vocabulary in each position of the document. An alternative is the **multivariate binomial model** or **multivariate Bernoulli model** – or simply: **binomial model**. It is equivalent to the BIM of Section 11.3 (page 170), which generates an indicator for each word of the vocabulary, either 0 indicating absence or 1 indicating presence of the word in the document. We compare the two models in Table 13.2.

The different generation models imply different estimation strategies for the parameters. The binomial model estimates $P(U_w = 1 | c_j)$ as the *fraction of documents* of class $c_j$ that contain word $w$ where $W_w$ is a random variable whose values 0 and 1 indicate absence and presence, respectively, of word $w$. In contrast, the multinomial model estimates $P(X = w | c_j)$ as the *fraction of tokens or fraction of positions* in documents of class $c_j$ that contain word $w$. This difference between document generation vs. token generation as the underlying event affects how multiple occurrences are used in classification. The binomial model uses binary occurrence information, ignoring the number of occurrences, whereas the multinomial model keeps track of multiple occurrences. As a result, the binomial model typically makes many mistakes when classifying long documents. For example, it may assign a book to the class *China* because of a single occurrence of the word *China*. 

---

Preliminary draft (c) 2007 Cambridge UP
13.4 Properties of Naive Bayes

Naive Bayes is so called because of its “naive” independence assumptions. The conditional independence assumption states that features are independent of each other given the class. This is hardly ever true for words in documents. In many cases, the opposite is true. The pairs hong and kong or london and english in Figure 13.4 are examples of highly dependent words. In addition, the multinomial model makes an assumption of positional independence. The binomial model ignores positions in documents altogether since 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. These independence assumptions are so sweeping that Naive Bayes is sometimes called Idiot Bayes. How can Naive Bayes be a good text classifier when its model of natural language is so oversimplified?

The answer lies in a paradox. Even though the probability estimates of Naive Bayes are of low quality, its classification decisions are surprisingly good. Unless they are normalized, NB probability estimates tend to be close to 0 since multiplying large numbers of conditional probabilities produces numbers close to 0. The winning class usually has a much larger probability than the other classes, so after normalization its probability will be close to 1. In either case, the normalized NB probability is seldom a good estimate of the actual probability. But the classification decision is based on which class gets the highest score. It does not matter how accurate the probabilities are. An example is shown in Table 13.3. Even though Naive Bayes fails to correctly estimate the actual probabilities, it assigns a higher probability to $c_1$ and therefore assigns $d$ to the correct class. Correct estimation implies accurate prediction, but accurate prediction does not imply correct estimation. Naive Bayes is a classifier that estimates badly, but classifies well.

Even if it is not the method with the highest accuracy for text, Naive Bayes has many virtues that make it a strong contender for text classification. In contrast to methods like decision trees, it excels if there are many equally important features that jointly contribute to the classification decision. It is also somewhat robust to noise features 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). Its 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 Naive Bayes as a classifier for text. The independence assumptions do not hold for text. However, it can be shown that Naive Bayes is an optimal classifier in domains where the independence assumptions do hold.

13.5 Feature selection

Feature selection in text classification 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 Naive Bayes, are expensive to train. Secondly, feature selection often increases classification accuracy. By eliminating noise features, text classifiers avoid overfitting – basing decisions on rare events in the training set that do not generalize to test data. Without noise features, classification is more accurate. Of the two NB models, the binomial model is particularly sensitive to noise features. A well performing binomial NB classifier requires some form of feature selection.

This section addresses feature selection for a single binary classifier. Section 13.5.4 briefly discusses optimizations for operational systems with a large number of binary classifiers.

13.5.1 Mutual information

A common feature selection method is mutual information (MI):

\[
I(W; C) = \sum_{e_w \in \{0,1\}} \sum_{e_c \in \{0,1\}} P(W = e_w, C = e_c) \log_2 \frac{P(W = e_w, C = e_c)}{P(W = e_w)P(C = e_c)}
\]

(13.8)

3. Feature selection corresponds to two different formal processes in the two NB models. In the binomial model, the dimensionality of the underlying document representation is reduced. In the multinomial model, the sample space of the multinomial random variable is reduced – it has fewer possible outcomes after feature selection. As is customary, we call both of these processes feature selection here.
13.5 Feature selection

where the value of \( W \) indicates presence or absence of the word \( w \) in a document and the value of \( C \) indicates membership in class \( c \). To select \( k \) words \( w_1, \ldots, w_k \) for class \( c \), compute the values \( I(W_1; C), \ldots, I(W_M; C) \) and select the \( k \) words with the highest values. Mutual information measures how much information – in the information-theoretic sense – a word contains about the class. If a word’s distribution is the same in the class as it is in the collection as a whole, then \( I(W; C) = 0 \). MI reaches its maximum value if the word is a perfect indicator for class membership, that is, if the word is present in a document if and only if the document is in the class.

Figure 13.4 shows words with high mutual information scores for the six classes in Figure 13.1. The selected words (e.g., London, UK, British for the class UK) are of obvious utility for making classifying decisions for their respective classes. At the bottom of the list for UK we find words like peripherals.

\footnote{Feature scores were computed on the first 100,000 documents, except for poultry, a rare class, for which 800,000 documents were used. Numbers and other special words were omitted from the top 10 lists.}
and tonight (not shown in the figure) that are clearly not helpful in deciding whether the document is in the class. As you might expect, keeping the informative terms and throwing away the non-informative ones tends to reduce noise and improve the classifier’s accuracy.

Such an accuracy increase can be observed in Figure 13.5 where classification accuracy (the fraction of documents classified correctly) is shown as a function of vocabulary size after feature selection. The data set used for this experiment is a collection of web pages from universities. Classes are student page, faculty page, department page etc. Accuracy initially goes up as we add features to the vocabulary. It then peaks at about 100 features for the binomial and at about 10,000 features for the multinomial. More importantly, accuracy goes down rather steeply for the binomial once it has reached its peak because it is sensitive to noise features. There is no such decline for the multinomial: it takes the number of occurrences into account in parameter estimation and classification and therefore better distinguishes between good and bad discriminators of the class.

13.5.2 $\chi^2$ feature selection

Another popular feature selection method is $\chi^2$ (pronounced chi-square with
### Table 13.4 Critical values of the $\chi^2$ distribution with one degree of freedom. For example, if $W$ and $C$ are independent, then $\Pr(X^2 > 6.63) < 0.01$. So for $X^2 > 6.63$ the assumption of independence can be rejected with 99% confidence.

<table>
<thead>
<tr>
<th>$p$</th>
<th>$\chi^2$ critical value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>2.71</td>
</tr>
<tr>
<td>0.05</td>
<td>3.84</td>
</tr>
<tr>
<td>0.01</td>
<td>6.63</td>
</tr>
<tr>
<td>0.005</td>
<td>7.88</td>
</tr>
<tr>
<td>0.001</td>
<td>10.83</td>
</tr>
</tbody>
</table>

In statistics, the $\chi^2$ test is applied to test the independence of two random variables. In feature selection, these two variables are occurrence of the word ($W$) and occurrence of the class ($C$). We then rank words with respect to the following quantity:

$$X^2(W, C) = \sum_{e_w \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(O_{e_we_c} - E_{e_we_c})^2}{E_{e_we_c}} \quad (13.9)$$

where, for example, $O_{11}$ is the observed frequency of $w$ and $c$ occurring together (number of documents in $c$ that contain $w$) and $E_{11}$ is the expected frequency of $w$ and $c$ occurring together assuming that word and class are independent.

The expected frequency is the product of the marginals (see definition on page 124) times the number of documents. Here is an example of computing $E_{11}$ and applying the test to the class *poultry* and the word *export*.

$$E_{11} = N \times P(w) \times P(c) = N \times \frac{O_{11} + O_{10}}{N} \times \frac{O_{11} + O_{01}}{N} \approx 6.6$$

where $N = O_{11} + O_{01} + O_{10} + O_{00} = 801,948$

\[
\begin{array}{ccc}
\text{poultry} = 1 & O_{11} = 49 & E_{11} \approx 6.6 \\
& O_{10} = 27,652 & E_{10} \approx 27,694.4 \\
\text{export} = 0 & O_{01} = 141 & E_{01} \approx 183.4 \\
& O_{00} = 774,106 & E_{00} \approx 774,063.6 \\
\end{array}
\]

$$X^2(W, C) = \sum_{e_w \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(O_{e_we_c} - E_{e_we_c})^2}{E_{e_we_c}} \approx 284$$

It can be shown that if $W$ and $C$ are independent, then $X^2 \sim \chi^2$, where $\chi^2$ is the $\chi^2$ distribution, one of the best known distributions in statistics.
Some critical values of the distribution are given in Table 13.4 for one degree of freedom, the number of degrees in this case. A value of $X^2$ that is unexpectedly high indicates that the hypothesis of independence is incorrect. In our example, $X^2 \approx 284 > 10.83$. Based on Table 13.4, we can reject the hypothesis that poultry and export are independent with only a 0.001 chance of being wrong. 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 one makes the occurrence of the other more likely (or less likely), so it should be helpful as a feature. This is the rationale of $\chi^2$ feature selection.

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

$$X^2(W, C) = \frac{(O_{11} + O_{10} + O_{01} + O_{00}) \times (O_{11}O_{00} - O_{10}O_{01})^2}{(O_{11} + O_{01}) \times (O_{11} + O_{10}) \times (O_{10} + O_{00}) \times (O_{01} + O_{00})}$$

This is equivalent to Equation 13.9 (Exercise 13.11).

From a statistical point of view, $\chi^2$ 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 1000 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. The typical application of $\chi^2$ (and MI) is to select the top $k$ features where $k$ is chosen based on some knowledge about the problem or by cross-validation. As long as $\chi^2$ 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, i.e., selecting the words that are most common in the class. This method will select some frequent words 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 hundreds or thousands of features are selected, then frequency-based feature selection does surprisingly well, with a typical relative accuracy decrease of around 5% (for example, from 80% to 76% accuracy). If somewhat suboptimal accuracy is acceptable, then
frequency-based feature selection is a good alternative to more complex methods. However, this does not apply when only a few features (on the order of 10) are selected. In this case, mutual information and $\chi^2$ perform much better than frequency-based selection.

13.5.4 Comparison of feature selection methods

The selection criteria of mutual information and $\chi^2$ are quite different. The independence of term and class can sometimes be rejected with high confidence even if the term carries little information useful for classification. This is particularly true for rare terms. If a word 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, $\chi^2$ 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 $\chi^2$ 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.

All three methods – MI, $\chi^2$ and frequency-based – are greedy methods. They may select features that contribute no information in addition to previously selected features. In Figure 13.4, kong is selected as the seventh word 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.

In an operational system with a large number of binary classifiers, it is desirable to select a single set of features that works well for all classifiers. One way of doing this is to compute the $\chi^2$ statistic for an $n \times 2$ table where the columns are occurrence and non-occurrence of the word and each row corresponds to one of the classes (or, more precisely, each classifier’s positive class). We can then select the $k$ words with the highest $\chi^2$ statistic as before.

More commonly, feature selection statistics are first computed separately for each classifier and then combined (for example, by averaging) into a single figure of merit. Classification accuracy almost always decreases when selecting $k$ common features for a system with $n$ classifiers as opposed to $n$ different sets of size $k$. But the gain in efficiency due to a common document representation is often worth such a decrease.
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 categorization system. It
is much smaller than and predates the Reuters RCV1 collection discussed
in Chapter 4 (page 52). 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 or multivalue problem (Chapter 14, page 232) is to learn 118 binary classifiers, one for each class. 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 with at least one class, and comprises 9603 training documents and 3299 test documents. The distribution of documents in classes is quite uneven, and some work classifies only for the 10 largest classes. They are listed in Table 13.5. A typical document with topics is shown in Figure 13.6.

We already saw one evaluation measure for text classifiers, accuracy, in Figure 13.5. We know from Section 8.3 (page 116) that classification accuracy is the proportion of correct decisions. This measure is appropriate if the population rate of the class is high, perhaps 10–20% and higher. But as was discussed in Section 8.3, accuracy is not a good measure for “small” classes since always saying “no”, a strategy that defeats the purpose of building a classifier, will achieve high accuracy. The “always no” classifier achieves 99% accuracy for a class with population rate 1%. For small classes, precision, recall and $F_1$ are better measures. We will use effectiveness as a generic term for measures that evaluate the quality of classification decisions, including precision, recall, $F_1$ and accuracy. In this book, performance refers to the efficiency of classification and information retrieval systems. However, many researchers mean quality, not efficiency of text classification when they use the term performance.

As in the case of Reuters-21578 and Reuters-RCV1, most other evaluation benchmarks for text classification also contain a set of classes rather than a single class. This means that measures for individual classes have to be combined into one aggregate measure. 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 based on the average of the metrics for each class.
Table 13.7 Experimental results for $F_1$ on Reuters-21578 (all classes).

<table>
<thead>
<tr>
<th>Method</th>
<th>$F_1$ micro-avg.</th>
<th>$F_1$ macro-avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>multinomial NB</td>
<td>0.80</td>
<td>0.47</td>
</tr>
<tr>
<td>SVM</td>
<td>0.89</td>
<td>0.60</td>
</tr>
</tbody>
</table>

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. Since 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 $c_2$ (0.9) 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, compute macroaveraged results.

Table 13.7 gives microaveraged and macroaveraged effectiveness of Naive Bayes for 90 classes in Reuters-21578. To give a sense of the relative effectiveness of Naive Bayes, we compare it to support vector machines (Chapter 15). Naive Bayes has a microaveraged $F_1$ of 80% which is 9% less than the SVM (89%), a 10% relative decrease. So there is a surprisingly small effectiveness penalty for its simplicity and efficiency. However, on small classes, many of which only have on the order of ten positive examples in the training set, Naive Bayes does much worse. Its macroaveraged $F_1$ is 13% below optimal, a 22% relative decrease.

### 13.7 References and further reading

Sebastiani (2002) gives a comprehensive review of text classification methods and results. Lewis (1998) focuses on the history of Naive Bayes classification. McCallum and Nigam (1998) discuss binomial and multinomial models and their accuracy for different collections. Figure 13.5 is from their paper. Friedman (1997) and Domingos and Pazzani (1997) analyze why Naive Bayes performs well although its probability estimates are poor. Ng and Jordan (2001) show that Naive Bayes is sometimes (though rarely) superior to discriminative methods because it more quickly reaches its optimal error rate. The problem of concept drift is discussed by Forman (2006) and Hand (2006).

Yang and Pedersen (1997) review feature selection methods and their impact on classification effectiveness. They find that pointwise mutual information...

Table 13.8 A set of documents for which the Naive Bayes independence assumptions are problematic.

A number of approaches for hierarchical classification have been developed in order to deal with the common situation where the classes to be assigned have a natural hierarchical organization (Koller and Sahami 1997, Weigend et al. 1999, Dumais and Chen 2000). In a recent large study using the Yahoo! directory, Liu et al. (2005) conclude that hierarchical classification noticeably if still modestly outperforms flat classification.

13.8 Exercises

Exercise 13.1
Which of the documents in Table 13.8 have identical and different bag of words representations for (a) the binomial model (b) the multinomial model?

Exercise 13.2
The rationale for the positional independence assumption is that there is no useful information in the fact that a word occurs in position \( k \) of a document. Try to find exceptions. Consider formulaic documents with a fixed document structure.

Exercise 13.3
The class priors in Figure 13.3 are computed as the fraction of documents in the class as opposed to the fraction of tokens in the class. Why?

Exercise 13.4
Why is \(|C||V| < |D|L_d\) in Table 13.1 expected to hold for most text collections?

Exercise 13.5
Why would a more complete name for the multinomial model be univariate multinomial model?

Exercise 13.6
Table 13.2 gives binomial and multinomial estimates for the word the. Explain the difference.
Exercise 13.7
What are the values of $I(W; C)$ and $X^2(W, C)$ if word and class are completely independent? What are the values if they are completely dependent?

Exercise 13.8
The feature selection method in Equation 13.8 is most appropriate for the binomial model. Why? How could one modify it for the multinomial model?

Exercise 13.9
Compute $I(\text{export}; \text{poultry})$ for the data on page 207.

Exercise 13.10
Features can also be selected according to information gain (IG). Information gain is defined as:

$$IG(D, f) = H(p_D) - \sum_{x \in \{D_f, D_{f^c}\}} \frac{|x|}{|D|} H(p_x)$$

where $H$ is entropy and $D,f,+,$ and $D,f^c$ are the entire collection, the subcollection of documents with feature $f$, and the subcollection of documents without feature $f$, respectively. $p_A$ is the class distribution in (sub)collection $A$, e.g., $p_A(c) = 0.25, p_A(\neg c) = 0.75$ if a quarter of the documents in $A$ are in class $c$.

Show that mutual information and information gain are equivalent.

Exercise 13.11
Show that the two $X^2$ formulae (Equations (13.9) and (13.10)) are equivalent.

Exercise 13.12
In the $\chi^2$ example on page 207 we have $|O_{11} - E_{11}| = |O_{10} - E_{10}| = |O_{01} - E_{01}| = |O_{00} - E_{00}|$. Show that this holds in general.

Exercise 13.13
$\chi^2$ and mutual information do not distinguish between positively and negatively correlated features. Since most good text classification features are positively correlated (i.e., they occur more often in the class than outside), one may want to explicitly rule out the selection of “negative” indicators. How would you do this?

Exercise 13.14
Your task is to classify words as English or not English. Words are generated by a source with the following distribution:

<table>
<thead>
<tr>
<th>event</th>
<th>word</th>
<th>English?</th>
<th>probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ozb</td>
<td>0</td>
<td>4/9</td>
</tr>
<tr>
<td>2</td>
<td>uzu</td>
<td>0</td>
<td>4/9</td>
</tr>
<tr>
<td>3</td>
<td>zoo</td>
<td>1</td>
<td>1/18</td>
</tr>
<tr>
<td>4</td>
<td>bun</td>
<td>1</td>
<td>1/18</td>
</tr>
</tbody>
</table>

(i) Compute the parameters (priors and conditionals) of a multinomial Naive Bayes 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 words 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 $P(A) + P(\neg A) > 1$, which can be corrected if we correspondingly smooth all complementary probability-1 values into probability 0.99. For this exercise, solutions may omit this correction to simplify arithmetic.) (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.