# Chapter 14

# Bags of Words

Possibly the simplest way to represent the "meaning" of a text is to assume that each word in the text contributes a little bit of meaning, which we can represent as the one-hot vector, and that we can combine the "meanings" of words simply by adding up their vectors.

$$\text{the cat sat on the mat} \rightarrow \{\text{cat} : 1, \text{mat} : 1, \text{on} : 1, \text{sat} : 1, \text{the} : 2\}$$

In information retrieval, this is called a *term vector*, but in NLP it's more commonly known as a *bag of words*. This representation completely ignores any semantic relationships between words (like synonmy) and any kind of structure of the text (like word order or syntax). Yet, in many situations, it can be very effective.

**Tokenization** We assume that all the sentences / documents have been tokenized so that the word boundaries are unambiguous. A commonly used English tokenizer is part of Stanford CoreNLP,[1] and LDC has a simple rule-based tokenizer.[2] Both are implemented / wrapped in Python by NLTK.[3]

**Stemming** It is also common in bag-of-word approaches to do morphological *stemming*, that is, removing affixes like *-ed*, *-ing*, etc. A classic stemmer for English is the Porter stemmer,[4] and another is the Lancaster (Paice / Husk) stemmer.[5] Both are implemented / wrapped in Python by NLTK.[6]

**Stop Words** Finally, it's also common to remove high-frequency but low-content words, like (Manning, Raghavan, and Schütze, 2008, p. 26):

a an and are as at be by for from has he in
is it its of on that the to was were will with

[1] `http://nlp.stanford.edu/software/corenlp.shtml`
[2] `http://www.cis.upenn.edu/~treebank/tokenizer.sed`
[3] `http://www.nltk.org/api/nltk.tokenize.html`
[4] `http://tartarus.org/martin/PorterStemmer/`
[5] `http://www.comp.lancs.ac.uk/computing/research/stemming/index.htm`
[6] `http://www.nltk.org/api/nltk.stem.html`

## 14.1 Classification

Suppose we have a collection of documents in different classes, and we want to learn to classify new documents. For example:

- We have a collection of e-mails that are labeled either as "spam" or "ham" and we want to learn to classify new e-mails.
- We have some texts whose author is known (e.g., Alexander Hamilton, James Madison, and John Jay) and want to classify anonymous texts (the Federalist Papers).
- We have a database of product reviews together with star ratings, and we want to be able to predict star ratings based on reviews alone.
- We have samples of (transcribed) speech from children diagnosed as autistic or not autistic, and we want to automatically diagnose other children using their speech.
- We have tweets that are known to be in various languages and want to automatically identify the language of new tweets.

**Question 12.** What are some other possible applications?

We'll continue working with the example of spam filtering, but just remember that this is only one of many possible applications. This is a fairly easy problem, but occasionally difficult even for humans. For example:

```
From: fas@nsf.gov
Subject: Payment
To: chiang@isi.edu

DFM has approved your requested payment and has asked the U.S. Treasury to
issue a payment to you within the next 4 working days. This payment is
being sent directly to the Bank or Financial Institution identified by
you for this purpose. If you are an NSF employee, this payment is being
sent directly to the bank/financial institution where your bi-weekly pay
is being deposited.

This payment for 560.00 is 1099 reportable if it totals $600 or more
for the year (i.e. You will receive an IRS 1099-Misc form from
NSF)P131318.

Head, Accounts Payable Section.
-------------------------------
```

More formally, we are given documents $d_1, \ldots, d_n$ together with their correct classes $k_1, \ldots, k_n$. We want to learn a model $P(k \mid d)$, where $k$ is a class and $d$ is a document, and given a new document $d$, we want to be able to find, with high accuracy,

$$k^* = \arg\max_k P(k \mid d). \tag{14.1}$$

### 14.1.1 Naïve Bayes

**Question 13.** What problem would you run into if you used the model

$$P(k \mid d) = \frac{c(k,d)}{c(d)}?$$

What might you do to repair the model?

This route turns out to be difficult. Instead of thinking about how to classify a document, let's think about how the document came to be. First, someone decided to write an e-mail; he was either a spammer or a "hammer." Then, this person authored a document. More formally:

$$\begin{aligned} \arg\max_k P(k \mid d) &= \arg\max_k P(d)P(k \mid d) && (14.2)\\ &= \arg\max_k P(k,d) && (14.3)\\ &= \arg\max_k P(k)P(d \mid k). && (14.4) \end{aligned}$$

The advantage of thinking about it this way is that it is much easier to write down a model for $P(d \mid k)$ than for $P(k \mid d)$.

We'll consider just the simplest, which is called a *naïve Bayes* classifier and looks like this:

$$P(k,d) = p(k) \prod_{w \in d} p(w \mid k), \tag{14.5}$$

where the $p(k)$ and $p(w \mid k)$ are the parameters of the model. (Let's adopt the convention that $P(\text{something})$ and $p(\text{something})$ both denote the probability of something, but $P(\text{something})$ might be defined in terms of other probabilities, whereas $p(\text{something})$ is a parameter of the model that needs to be estimated.)

Naïve Bayes is called naïve because it naïvely assumes that all the words in a document are independent of each other. All it captures is that spammers are more likely to use certain words and hammers are more likely to use certain words.

It is possible to count other things besides words. For example, if we're trying to classify a document as English or French, then the presence of an é is a pretty good sign that it's in French. So character probabilities like $p(\text{é} \mid k)$ might be useful in addition to word probabilities. Or, if we're trying to predict the positivity or negativity of a product review, then the occurrence of the word *bad* might be good sign of a negative review, unless it is immediately preceded by the word *not*. So probabilities of *bigrams* like $p(\text{not bad} \mid k)$ are important. Note that if we do this, however, the distribution $P(k,d)$ no longer sums to one, because there's no such thing as a document that contains the words $\{\text{dog}\}$ but the characters $\{\text{c}, \text{a}, \text{t}\}$. So, if we only sum over feasible documents $d$, then $P(k,d)$ sums to less than one, that is, it is *deficient*, which is in general not considered a good thing. This is even more naïve, but might work okay in practice.

**Question 14.** What are some other features that might be helpful?

### 14.1.2 Training

Training the model, or estimating the parameters $p(k)$ and $p(w \mid k)$, is easy. It's just:

$$\begin{aligned} p(k) &= \frac{c(k)}{\sum_k c(k)} \\ p(w \mid k) &= \frac{c(k,w)}{\sum_{w'} c(k,w')}. \end{aligned} \tag{14.6}$$

This is known as *relative frequency estimation* (or "count and divide"). It's so intuitive that it may seem odd to give it a name, but it's worth dwelling a bit on why we estimate probabilities this way.

Consider just $p(k)$. Suppose we have just two theories, or *models*, about how prevalent spam is. Model 1, or $m_1$, says that 10% of e-mails are spam, whereas Model 2, or $m_2$, says that 90% of e-mails are spam.

$$\begin{aligned} p(\text{spam} \mid m_1) &= \tfrac{1}{10} & \qquad p(\text{spam} \mid m_2) &= \tfrac{9}{10} \\ p(\text{ham} \mid m_1) &= \tfrac{9}{10} & \qquad p(\text{ham} \mid m_2) &= \tfrac{1}{10} \end{aligned}$$

Before looking at the data, suppose that we think these two models are equally valid.

$$p(m_1) = \tfrac{1}{2} \qquad\qquad p(m_2) = \tfrac{1}{2}$$

Then, suppose we look at some data and see that, out of 10 e-mails, 7 are spam and 3 are ham. Now we want to know which model is better, $m_1$ or $m_2$?

$$\begin{aligned} P(m_1 \mid \text{data}) &= \frac{p(m_1)P(\text{data} \mid m_1)}{P(\text{data})} \\ &= \frac{\frac{1}{2}\left(\frac{1}{10}\right)^7\left(\frac{9}{10}\right)^3}{P(\text{data})} \\ &= \frac{9^3}{2 \cdot 10^{10} \cdot P(\text{data})} \end{aligned} \qquad\qquad \begin{aligned} P(m_2 \mid \text{data}) &= \frac{p(m_2)P(\text{data} \mid m_2)}{P(\text{data})} \\ &= \frac{\frac{1}{2}\left(\frac{9}{10}\right)^7\left(\frac{1}{10}\right)^3}{P(\text{data})} \\ &= \frac{9^7}{2 \cdot 10^{10} \cdot P(\text{data})} \end{aligned}$$

Since $P(m_2 \mid \text{data})$ is bigger, we can conclude that $m_2$ is the better model.

Now suppose that we compare not two models, but an infinite number of models,

$$\begin{aligned} P(\text{spam} \mid \theta) &= \theta \\ P(\text{ham} \mid \theta) &= 1 - \theta \end{aligned}$$

for all $\theta \in [0, 1]$. The same reasoning still holds: we want to find the model that maximizes

$$\underbrace{P(\theta \mid \text{data})}_{\text{posterior}} = \frac{\overbrace{P(\theta)}^{\text{prior}}\overbrace{P(\text{data} \mid \theta)}^{\text{likelihood}}}{\underbrace{P(\text{data})}_{\text{evidence}}}. \tag{14.7}$$

Note that the denominator is independent of $\theta$, and if we assume that the prior $P(\theta)$ is uniform, then the only factor that matters is the likelihood. We therefore call this *maximum likelihood estimation*.

Going back to the full naïve Bayes model: we want to maximize the likelihood, which is

$$L = \prod_i P(k_i, d_i) \tag{14.8}$$

$$= \prod_i p(k_i) \prod_{w \in d_i} p(w \mid k_i). \tag{14.9}$$

And this maximization problem has a closed-form solution, namely (14.6). See Section **??** for more details on how to derive that.

### 14.1.3 Classification

Once we've trained the model, finding the most probable class of a new document is also easy:

$$k^* = \arg\max_k P(k \mid d) \tag{14.10}$$

$$= \arg\max_k P(k, d) \tag{14.11}$$

$$= \arg\max_k p(k) \prod_{w \in d} p(w \mid k). \tag{14.12}$$

### 14.1.4 Smoothing

What happens if we encounter a word we've never seen before? Then we would have $p(w \mid k) = 0$, and therefore $P(k, d) = 0$, for all $k$. The presence of a single unknown word zeros out the whole document probability and makes it impossible to choose a class.

The standard solution is *smoothing*. We'll talk about the simplest smoothing method now, and more advanced smoothing methods later (when we talk about language models). In *add-one smoothing*, invented by Laplace in the 18th century, we add one to the count of every event, including unseen events. Thus we would estimate $p(w \mid k)$ as:

$$p(w \mid k) = \frac{c(k, w) + 1}{\sum_{w'} c(k, w) + |V|} \tag{14.13}$$

where $|V|$ is the size of the vocabulary, including unseen words. But what if we don't know how many unseen words there are? A typical (though not entirely correct) thing to do is to set $|V|$ to the number of seen word types, plus one for a generic unseen word.

We can also add any value $\delta > 0$ to the counts instead of one:

$$p(w \mid k) = \frac{c(k, w) + \delta}{\sum_{w'} c(k, w') + |V|\delta}. \tag{14.14}$$

The increment that we add (whether 1 or $\delta$) is known as a *pseudocount*. Typically, $\delta$ would be set by trial and error. If you have a good reason to, you can also add a different pseudocount to different word types.

These methods are not hacks; they can all be thought of as different choices of the prior probability that we saw in Equation (14.7).

**Question 15.** If our training data is:

| ham | spam | spam |
|---|---|---|
| `please pass on to your groups` | `we deliver to your door within 24 hours` | `please update your account details with citibank` |

and we train with add-one smoothing, what are $P(\text{spam} \mid d)$ and $P(\text{ham} \mid d)$ for the document below?

```
please forward to
your groups
```

### 14.1.5 Experiment

A classic dataset for text classification is Pang and Lee's dataset of movie reviews.[7]

Here are some example sentences from positive reviews:

```
the rock is destined to be the 21st century's new " conan " and that
he's going to make a splash even greater than arnold schwarzenegger ,
jean-claud van damme or steven segal .

the gorgeously elaborate continuation of " the lord of the rings "
trilogy is so huge that a column of words cannot adequately describe
co-writer/director peter jackson's expanded vision of j . r . r
. tolkien's middle-earth .

effective but too-tepid biopic
```

And some example sentences from negative reviews:

```
simplistic , silly and tedious .

it's so laddish and juvenile , only teenage boys could possibly find
it funny .

exploitative and largely devoid of the depth or sophistication that
would make watching such a graphic treatment of the crimes bearable .
```

When we run the naive Bayes classifier on this data, we get an accuracy of 75.7%. Some of the

[7] `http://www.cs.cornell.edu/people/pabo/movie-review-data/`

model probabilities are:

$$\begin{aligned}
p(+) &= 0.5 \\
p(-) &= 0.5 \\
p(moving \mid +) &= 0.000582 \\
p(portrait \mid +) &= 0.000463 \\
p(touching \mid +) &= 0.000404 \\
p(powerful \mid +) &= 0.000394 \\
p(engrossing \mid +) &= 0.000296 \\
p(cinema \mid +) &= 0.000611 \\
p(beautiful \mid +) &= 0.000404 \\
p(culture \mid +) &= 0.000374 \\
p(enjoyable \mid +) &= 0.000453 \\
p(wonderful \mid +) &= 0.000286 \\
p(bad \mid -) &= 0.001862 \\
p(dull \mid -) &= 0.000657 \\
p(boring \mid -) &= 0.000488 \\
p(too \mid -) &= 0.003285 \\
p(flat \mid -) &= 0.000328 \\
p(tv \mid -) &= 0.000388 \\
p(worst \mid -) &= 0.000458 \\
p(unfunny \mid -) &= 0.000258 \\
p(jokes \mid -) &= 0.000378 \\
p(? \mid -) &= 0.001473
\end{aligned}$$

(The words shown above are *not* the most frequent words; those are rather boring. Instead, we selected the words with the highest $p(w \mid k)/(p(w) + 10)$.)

## 14.2 Clustering

Let's go back to the naïve Bayes model and consider the case where some or all of the classes $k_i$ are unobserved. For example, if we have 1,000 e-mails labeled as spam or ham, and 1 million more unlabeled e-mails, maybe we can learn a better classifier by combining both labeled and unlabeled data. Maybe from the labeled data we learn that "award" often implies spam, and from the unlabeled data we learn that "award" often occurs in the same documents as "deposit," then we can infer that "deposit," too, is associated with spam.

If all of the $k_i$ are unobserved, then there's no way that the computer can learn to label emails as spam or ham. The best that we can do is tell the computer that there are two kinds of emails, and it should try to find some way of classifying them into class 1 and class 2. Then, just maybe, class 1 will be ham and class 2 will be spam, or class 1 will be spam and class 2 will be ham.

Then again, maybe the computer will find some other possibly meaningful way of classifying the emails instead. We can also choose more than two classes and see what happens. This is known as text *clustering*. Some of the most interesting NLP problems are unsupervised learning problems, although unsupervised learning also often fails.

### 14.2.1 Expectation-Maximization: "hard" version

Assume that all of the $k_i$ are unobserved. We have to choose a number of classes in advance. Then we want to maximize the likelihood, that is, the probability of the observed data. Since we only observe words without classes, the likelihood looks different from before:

$$L = \prod_i P(d_i) \tag{14.15}$$

$$= \prod_i \sum_k p(k) \prod_{w \in d_i} p(w \mid k). \tag{14.16}$$

The bad news is this does not have a closed-form solution, and we have to use an iterative method. While we could use a generic method like gradient ascent, it's far more common to use a specialized method called *expectation-maximization* (Dempster, Laird, and Rubin, 1977).

We start with the "hard" (as opposed to "soft") version. The basic idea is pretty intuitive. If we have labels, then we know how train the model (this is just like training in the supervised case). And if we have a model, we know how to predict the labels (this is just like testing in the supervised case). So, we can start by initializing the model parameters to random values; then guess labels for all the documents, then use those labels to retrain the model, and repeat.

initialize parameters $p(k)$ and $p(w \mid k)$ randomly
**repeat**
  **for** each $i$ **do**
    predict $k_i^* = \arg\max_k P(k \mid d_i)$
  **end for**
  estimate the parameters from the $d_i$ and $k_i^*$
**until** done

It seems that one of three things could happen: nothing, or the model will get worse and worse as noise takes over, or somehow the model will get better.

If you consider the partially supervised case (where some of the documents are labeled and some are not labeled), it's easier to see how it might actually get better. Suppose, as in the example from the beginning, that we've seen emails labeled as spam that contain the word "award" but never the word "deposit." But we've seen unlabeled emails containing both words; when we guess the label for those emails, they're likely to be labeled as spam. If they are, then when we retrain the model, we'll learn that "deposit" is mildly associated with spam, which is good.

In the fully unsupervised case, it's a little harder to see, but let's work out a simple example.

```
award notification
enron canada
enron america
award payment
```

Initialize the model randomly:

$$p(1) = 0.5 \qquad p(2) = 0.5$$
$$p(\texttt{america} \mid 1) = 0.1 \qquad p(\texttt{america} \mid 2) = 0.2$$
$$p(\texttt{award} \mid 1) = 0.1 \qquad p(\texttt{award} \mid 2) = 0.1$$
$$p(\texttt{canada} \mid 1) = 0.1 \qquad p(\texttt{canada} \mid 2) = 0.2$$
$$p(\texttt{enron} \mid 1) = 0.2 \qquad p(\texttt{enron} \mid 2) = 0.2$$
$$p(\texttt{notification} \mid 1) = 0.4 \qquad p(\texttt{notification} \mid 2) = 0.2$$
$$p(\texttt{payment} \mid 1) = 0.1 \qquad p(\texttt{payment} \mid 2) = 0.1$$

Guess a class for each example:

| | $P(1, d)$ | $P(2, d)$ | $k^*$ |
|---|---|---|---|
| `award notification` | 0.02 | 0.01 | 1 |
| `enron canada` | 0.01 | 0.02 | 2 |
| `enron america` | 0.01 | 0.02 | 2 |
| `award payment` | 0.005 | 0.005 | 1 (arbitrary) |

For the last document, there was a tie, which we broke arbitrarily. Retrain the model:

$$p(1) = 0.5 \qquad p(2) = 0.5$$
$$p(\texttt{america} \mid 1) = 0 \qquad p(\texttt{america} \mid 2) = 0.25$$
$$p(\texttt{award} \mid 1) = 0.5 \qquad p(\texttt{award} \mid 2) = 0$$
$$p(\texttt{canada} \mid 1) = 0 \qquad p(\texttt{canada} \mid 2) = 0.25$$
$$p(\texttt{enron} \mid 1) = 0 \qquad p(\texttt{enron} \mid 2) = 0.5$$
$$p(\texttt{notification} \mid 1) = 0.25 \qquad p(\texttt{notification} \mid 2) = 0$$
$$p(\texttt{payment} \mid 1) = 0.25 \qquad p(\texttt{payment} \mid 2) = 0$$

Guess a class for each example:

| | $P(1, d)$ | $P(2, d)$ | $k^*$ |
|---|---|---|---|
| `award notification` | 0.0625 | 0 | 1 |
| `enron canada` | 0 | 0.0625 | 2 |
| `enron america` | 0 | 0.0625 | 2 |
| `award payment` | 0.0625 | 0 | 1 |

And it's not hard to see that nothing changes after this. The algorithm clustered the documents by noticing that two documents had the word "award" in common and two documents had the word "enron" in common.

But what if we had broken that tie in the other direction? Retrain the model:

$$p(1) = 0.25 \qquad p(2) = 0.75$$
$$p(\texttt{america} \mid 1) = 0 \qquad p(\texttt{america} \mid 2) = 0.167$$
$$p(\texttt{award} \mid 1) = 0.5 \qquad p(\texttt{award} \mid 2) = 0.167$$
$$p(\texttt{canada} \mid 1) = 0 \qquad p(\texttt{canada} \mid 2) = 0.167$$
$$p(\texttt{enron} \mid 1) = 0 \qquad p(\texttt{enron} \mid 2) = 0.333$$
$$p(\texttt{notification} \mid 1) = 0.5 \qquad p(\texttt{notification} \mid 2) = 0$$
$$p(\texttt{payment} \mid 1) = 0 \qquad p(\texttt{payment} \mid 2) = 0.167$$

Guess a class for each example:

| | $P(d, 1)$ | $P(d, 2)$ | $k^*$ |
|---|---|---|---|
| `award notification` | 0.0625 | 0 | 1 |
| `enron canada` | 0 | 0.0417 | 2 |
| `enron america` | 0 | 0.0417 | 2 |
| `award payment` | 0 | 0.0208 | 2 |

So hard EM is able to discover some things, but is also kind of brittle.

### 14.2.2 Expectation-Maximization: real version

The real version of EM has a slight difference that makes it possible to actually prove that the likelihood (14.15) gets better, or is at a local maximum.

Before, we used the model to predict the best label for each document, by using the model's best guess for each. But its guesses aren't certain. For a document $d$, it only gives probabilities $P(1 \mid d)$ and $P(2 \mid d)$. So, it's more reasonable to say that we observed $d$ a fraction of a time in class 1, and a fraction of a time in class 2.

We do this for all the documents and add up all the counts to get *expected counts*, which may be fractional. Then, just like before, we train the model from those counts. It doesn't matter that the counts are fractional; we just count and divide like before.

Let's try this with our example, with the same random initialization as before. Instead of guessing the best class for each document, we compute the distribution over classes:

| | $P(1, d)$ | $P(2, d)$ | $P(1 \mid d)$ | $P(2 \mid d)$ |
|---|---|---|---|---|
| `award notification` | 0.02 | 0.01 | 0.667 | 0.333 |
| `enron canada` | 0.01 | 0.02 | 0.333 | 0.667 |
| `enron america` | 0.01 | 0.02 | 0.333 | 0.667 |
| `award payment` | 0.005 | 0.005 | 0.5 | 0.5 |

Note that we didn't have to do any arbitrary tie-breaking. Retrain the model:

$$p(1) = 0.458 \qquad p(2) = 0.542$$
$$p(\texttt{america} \mid 1) = 0.0909 \qquad p(\texttt{america} \mid 2) = 0.154$$
$$p(\texttt{award} \mid 1) = 0.318 \qquad p(\texttt{award} \mid 2) = 0.192$$
$$p(\texttt{canada} \mid 1) = 0.0909 \qquad p(\texttt{canada} \mid 2) = 0.154$$
$$p(\texttt{enron} \mid 1) = 0.182 \qquad p(\texttt{enron} \mid 2) = 0.308$$
$$p(\texttt{notification} \mid 1) = 0.182 \qquad p(\texttt{notification} \mid 2) = 0.0768$$
$$p(\texttt{payment} \mid 1) = 0.136 \qquad p(\texttt{payment} \mid 2) = 0.115$$

And so on. The computations are tedious to do by hand, but hopefully you can see where this is going—eventually the words "award", "notification", and "payment" will dominate class 1, and the words "enron", "america", and "canada" will dominate class 2.

The algorithm looks like this:

initialize parameters $p(k)$ and $p(w \mid k)$ randomly
**repeat**
  ▷ E step:
  **for** each $i$, $k$ **do**
    compute $P(k \mid d_i) = \dfrac{P(k, d_i)}{\sum_{k'} P(k, d_i)}$
    $E[c(k)] \leftarrow E[c(k)] + P(k \mid d_i)$
    **for** each $w \in d_i$ **do**
      $E[c(k, w)] \leftarrow E[c(k, w)] + P(k \mid d_i) \cdot c(w \in d_i)$
    **end for**
  **end for**
  ▷ M step:
  $p(k) = \dfrac{E[c(k)]}{\sum_{k'} E[c(k')]}$
  $p(w \mid k) = \dfrac{E[c(k, w)]}{\sum_{w'} E[c(k, w')]}$
**until** done

This version is guaranteed to make $L$ increase or, if at a local maximum, stay the same.

In the initialization step, it's very important to set the parameters randomly.

**Question 16.** What happens if we initialize the parameters uniformly?

**Question 17.** What happens if the number of classes is greater than or equal to the number of documents?

### 14.2.3 Details

**Number of iterations** Theoretically we should keep performing iterations of EM until the likelihood stops increasing. In practice, it's extremely common just to set a fixed number of iterations.

**Random restarts** Because (real) EM is only guaranteed to converge to a local maximum, if we run it again we might get a different result. So to try to get the best result, we might run EM many times and choose the one with the best likelihood.

**Smoothing** In the supervised case, we used add-one or add-$\delta$ smoothing. We can use it here too, during the M step. However, the likelihood is no longer guaranteed to increase at every iteration; it could decrease. There's another objective function that *is* guaranteed to increase, but it's not so common to explicitly compute it.

### 14.2.4 Experiment

We ran the unsupervised naïve Bayes clustering algorithm on the movie review data from before, using two classes and 20 iterations.

Whatever the algorithm learns, it isn't positive/negative:

| $k$ | $+$ | $-$ | total |
|---|---|---|---|
| 0 | 2330 | 2546 | 4876 |
| 1 | 2468 | 2252 | 4720 |

| class | words |
|---|---|
| 0 | `i know my how want you're you will don't ?` |
| 1 | `nearly * suffers master fine quiet satire solid appealing murder` |

If we increase the number of classes to ten, we get:

| $k$ | words |
|---|---|
| 0 | `strength faith magnificent actresses writer/director rhythm stone subtle tout bride` |
| 1 | `sisterhood tour ya-ya grand heartfelt force la secrets baffling flamboyant` |
| 2 | `reign sequence fire century mad artificial saga oddly woody pulls` |
| 3 | `know someone joke sign theaters see check company unless ?` |
| 4 | `chicago superficial feelings loose blade blood amount michael commentary speaking` |
| 5 | `* / nowhere martin hill card tough weight winner cheesy` |
| 6 | `[a] extremely painfully effective well-a ted e clunker exquisitely honesty stilted` |
| 7 | `wilder significant van amy's main hold shadow ingredients mann driven` |
| 8 | `niro murphy showtime de husband en secret es discovery shocks` |
| 9 | `fat bag my greek liked ecks ! alabama sever cell` |

So, fully unsupervised learning isn't magic; sometimes its results can be difficult to interpret, or sometimes its results can be garbage.

But note that class 8 contains a number of foreign words (de, en, es). This is pretty common, and suggests that naive Bayes clustering might be good at automatically categorizing documents according to language. If we run the clusterer on text in 10 European languages (500 sentences per language) from the WiLI-2018 dataset,[8] we get the following results (showing the most frequent words in each class):

[8] `https://doi.org/10.5281/zenodo.841984`

| $k$ | words |
|---|---|
| 0 | de a e o do em da que no um |
| 1 | de la en y el a que le et un |
| 2 | the in and of a to ja on was de |
| 3 | de a in the la der e en et des |
| 4 | der und die in von den des im de wurde |
| 5 | de i en og the der af at in er |
| 6 | the of a and in to di i for at |
| 7 | the de in of a and la en i di |
| 8 | de la di e del a il en con in |
| 9 | i ä den med och og en som de av |

It's not perfect, but many of the classes are pretty clear: 0 = Portuguese, 1 = Spanish, 2 = English, 4 = German, 5 = Danish, 8 = Italian, 9 = Swedish.

## 14.3 Topic Modeling

Counting words can be a pretty good way of quantifying what a document is about, but not perfect. Suppose that we are trying to categorize documents into different genres. If a document contains the word "laser" or "robot" or other similar words, that is a pretty good sign that it's an engineering article or a science-fiction story. Suppose further that many of these technology-related words just occur one time – say, "photon" occurs only in an engineering article and "ramjet" appears only in a science-fiction story. That wouldn't be very good evidence that documents mentioning "photon" are engineering and documents mentioning "ramjet" are science-fiction. It would be more reasonable for a model to learn that these are all technology-related words, and that technology-related words are indicative of both engineering and science fiction. That is, all these words belong to the *topic* of technology, and both engineering and science fiction use words in the technology topic. How might we learn such topics automatically? It turns out that these topics aren't all that helpful for classification; nevertheless, topics are interesting in their own right.

### 14.3.1 Naïve Bayes with topics

Let's modify the NBC as follows:

$$P(d) = p(k) \prod_{w \in d} \sum_{t} p(t \mid k) p(w \mid t). \tag{14.17}$$

What's new is the introduction of the variable $t$ (topic), which is drawn from a finite set of predetermined size. We are given training data consisting of documents $d_i$ and classes $k_i$, but we are not given the topics of the words in the $d_i$. The topics are called *hidden* variables.

#### Training

The class distribution $p(k)$ is estimated by counting and dividing:

$$p(k) = \frac{c(k)}{\sum_k c(k)}. \tag{14.18}$$

The rest of the model is trained using EM. In the E step, we compute, for each document $d_i$ and word $w \in d_i$,

$$P(t \mid k_i, w) = \frac{p(t \mid k_i)p(w \mid t)}{\sum_{t'} p(t' \mid k_i)p(w \mid t')}. \tag{14.19}$$

Then, for each topic $t$,

$$E[c(k_i, t)] \leftarrow E[c(k_i, t)] + P(t \mid k_i, w) \tag{14.20}$$

$$E[c(t, w)] \leftarrow E[c(t, w)] + P(t \mid k_i, w). \tag{14.21}$$

The M step is just count and divide as usual:

$$p(t \mid k) = \frac{E[c(k, t)]}{\sum_{t'} E[c(k, t')]} \tag{14.22}$$

$$p(w \mid t) = \frac{E[c(t, w)]}{\sum_{t'} E[c(t', w)]}. \tag{14.23}$$

**Experiment**

We ran this model on the movie dataset, using 2 topics, 25 iterations, and add-$10^{-6}$ smoothing, and got an accuracy of 77.4%, which is a small improvement over the original model (75.7%).

Somewhat more interesting are the topics learned by the model.

| topic | words |
|---|---|
| 0 | `stupid boring dull flat routine bad mess unfunny fails neither` |
| 1 | `funny both drama us look performances romantic entertaining world those` |
| 2 | `too lack bad silly title worst already ? mess tedious` |
| 3 | `compelling ride fascinating both entertaining works surprisingly heart family experience` |
| 4 | `made up through ' all never out comes audience than` |
| 5 | `worst too tries bad gross-out things ? title premise tv` |
| 6 | `engrossing beautiful refreshingly terrific portrait quiet riveting thoughtful moving enjoyable` |
| 7 | `poorly dull loud bad flat boring jokes generic numbers badly` |
| 8 | `long just isn't where when be doesn't plot movie or` |
| 9 | `de drama film love with your and their you life` |
| 10 | `who often story their picture cast and funny movies you` |
| 11 | `that's up quite as lot for little which it's out` |
| 12 | `remarkable quiet our relationships study moving fascinating culture performances powerful` |
| 13 | `compelling smart performances personal family portrait world intelligent different our` |
| 14 | `far can one but that in little , itself feel` |
| 15 | `unfunny flat boring bad mediocre dull plodding tv product badly` |
| 16 | `many not never it into has time ( from but` |
| 17 | `vividly lively provides touching warm wonderful disturbing powerful engrossing riveting` |
| 18 | `film gives picture makes without you with at first and` |
| 19 | `too bad title jokes fails feels problem nothing silly tv` |

Many of the topics are clearly dominated by positive or negative words. There are also seem to be some topics with neutral words (4, 8, 9, 10, 11, 14, 16, 18). It's hard to guess what might further differentiate the topics (if anything).

### 14.3.2 Probabilistic LSA

We can think of a document represented as a bag of words as a vector, where each component of the vector is a different word type. In many applications, this vector representation is useful in its own right. For example, we can compute the similarity between two documents using the cosine of the angle between the two document vectors.

As argued above, however, words might be too fine-grained a way of quantifying what a document is about. So what if we use vectors where each component of the vector is a different *topic*? This is the idea behind *probabilistic latent semantic analysis* (PLSA) or *probabilistic latent semantic indexing* (PLSI) (Hofmann, 1999). The only difference between PLSA and the NBC with topics is that in PLSA, every document is in *its own class*. Instead of training the model to predict a label (e.g., positive or negative), we're training the model to look at a test document and guess which individual training document it's most similar to. This task is not very useful, but what we get from it is the probabilities $p(t \mid k_i)$, which can be thought of as a vector representation of document $d_i$, and the probabilities $p(t \mid w)$, which can be thought of as a vector representation of word $w$.

Training PLSA is identical to above. But after training a PLSA model, there isn't any inference step (because we don't care about the document-similarity task). The end product of PLSA training is the parameters themselves. We ran PLSA using the same settings as before, but for more iterations. This time, the topics we got were:

| topic | words |
|---|---|
| 0 | `made last since fans once passion again perfect cinema animated` |
| 1 | `? don't man video stuff satisfying problem what's how why` |
| 2 | `far journey summer getting job surprises surprise water setting crush` |
| 3 | `works worth genre idea seeing cliches already appeal add female` |
| 4 | `documentary heart mind fascinating do difficult satire strong light filmmakers` |
| 5 | `her romantic truly formula girl start delightful major puts play` |
| 6 | `there's those lot women enjoy need beautiful close something real` |
| 7 | `your you'll flat deep throughout murder forgettable coming-of-age seriously [a]` |
| 8 | `" other ! comes year silly ending horror less five` |
| 9 | `if fun because watch mr you're beyond hilarious sad witty` |
| 10 | `too quite moving place amusing visually human somewhat important appealing` |
| 11 | `performance keep sometimes romance melodrama compelling wrong wonderful strange straight` |
| 12 | `entertaining family comic mostly surprisingly touching ride leaves brilliant quiet` |
| 13 | `de short hard psychological que falls y la ideas o` |
| 14 | `bad study sort rarely gives entire modern fire ugly awful` |
| 15 | `piece part fine beautifully humor pleasure storytelling did usual solid` |
| 16 | `doesn't care lacks power else yet work they rich impossible` |
| 17 | `moments my thing dramatic neither nor mess there funny career` |
| 18 | `i like feels me pretty still looks left long worst` |
| 19 | `direction seem everyone opera help tedious soap nature alone predictable` |

Note how different these topics look from before. None are strongly positive or negative, because the model is no longer receiving information about positivity or negativity. But there doesn't seem to be that much coherence in general in these topics.

### 14.3.3 Optional: Latent Dirichlet Allocation

No one uses PLSA anymore. *Latent Dirichlet Allocation* (Blei, Ng, and Jordan, 2003) is a further development of PLSA that is by far the most widely known and used topic model. LDA is quite math-heavy and we don't give a full description here, but just a little bit of the basic idea.

In Section 14.1.2, we introduced the notion of a *prior* distribution:

$$\underbrace{P(\theta \mid \text{data})}_{\text{posterior}} = \frac{\overbrace{P(\theta)}^{\text{prior}} \overbrace{P(\text{data} \mid \theta)}^{\text{likelihood}}}{\underbrace{P(\text{data})}_{\text{evidence}}}, \tag{14.24}$$

and we just assumed that the prior was uniform. But we can choose other distributions too. The easiest choice of prior, other than uniform, is the (symmetric) *Dirichlet distribution*:

$$\text{Dir}(\theta; \alpha) \propto \prod_i \theta_i^{\alpha - 1}. \tag{14.25}$$

The parameter $\alpha$ is called the *concentration*.

Now we have a number of options. First, we can try to find the best model ($\theta$) just like before. Since we are now taking the prior into account, we're not doing maximum likelihood estimation anymore; we're doing *maximum a posteriori (MAP) estimation*. It's not too hard to see that the resulting estimate is exactly what the MLE estimate would have been if there were pseudocounts of $(\alpha - 1)$ for each outcome. In other words, MAP estimation with a Dirichlet prior with concentration $\alpha$ is equivalent to MLE estimation with add-$(\alpha - 1)$ smoothing.

But another way of thinking about it is, if we're only interested in the topics, and not $\theta$, then we shouldn't try to estimate $\theta$. Instead, we should integrate over all possible values of $\theta$:

$$P(\text{topics} \mid \text{data}) = \frac{P(\text{topics}, \text{data})}{P(\text{data})} \tag{14.26}$$

$$= \frac{\int P(\text{topics}, \text{data}, \theta)\, \mathrm{d}\theta}{\int P(\text{data}, \theta) \mathrm{d}\theta} \tag{14.27}$$

That looks bad, and in all but the simplest cases, it is bad, but that doesn't stop people from trying to do it anyway.

Asuncion et al. (2009) provide a good overview and comparison of both PLSA and LDA. The LDA model looks like this:

$$P(\text{data} \mid \alpha, \beta) = \underbrace{\prod_t P(\phi_t \mid \beta) \prod_i P(\theta_i \mid \alpha)}_{\text{prior}} \underbrace{\prod_{w \in d_i} \sum_t p(t \mid i, \theta) p(w \mid t, \phi)}_{\text{likelihood}}. \tag{14.28}$$

The part labeled "likelihood" is more or less the same as PLSA. The part labeled "prior" has two parts; $P(\phi_t \mid \beta)$ is a Dirichlet prior on the word distributions $p(w \mid t)$, and $P(\theta_i \mid \alpha)$ is a Dirichlet prior on the topic distributions $p(t \mid i)$.

From here, we can use one of several different approximation methods to proceed. One of these (Variational Bayes or mean-field approximation) is operationally quite similar to EM. For further information, see the references listed below.

## References

Asuncion, Arthur et al. (2009). "On Smoothing and Inference for Topic Models". In: *Proc. UAI*, pp. 27–34.

Blei, David M., Andrew Y. Ng, and Michael I. Jordan (2003). "Latent Dirichlet Allocation". In: *J. Machine Learning Research* 3, pp. 993–1022.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society, Series B* 39.1, pp. 1–38.

Hofmann, Thomas (1999). "Probabilistic Latent Semantic Analysis". In: *Uncertainty in Artificial Intelligence (UAI)*, pp. 289–296.

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze (2008). *Introduction to Information Retrieval*. Cambridge University Press. URL: `http://www-nlp.stanford.edu/IR-book/`.