Statistical methods in NLP
Classification

Richard Johansson

UNIVERSITY OF GOTHENBURG

February 4, 2016

overview of today's lecture

- classification: general ideas
- Naive Bayes recap
  - formulation, estimation
  - Naive Bayes as a generative model
- other classifiers (that are not generative)
- practical matters

overview

introduction

Naive Bayes definition and generative models

estimation in the Naive Bayes model

discriminative models

the next few weeks

classifiers. . .

- given an object, assign a category
- such tasks are pervasive in NLP

example: classification of documents

- **assignment 1**: develop a program that groups customer reviews into positive and negative classes (given the text only)

Just plain lame., August 14, 2007

By Gary Smith "Editor, Handgun Hunter Magazine" (Texas) - See all my reviews

This review is from: Garden & Gun (Magazine)

This magazine has a catchy title and very nice graphics and photography. What the premier issue lacks is anything of any substance about guns or hunting. I wonder if they actually read their own title. In my opinion these guys are nothing more than posers from the guns/hunting standpoint and many of the photographs appear to be staged. In particular, there are a couple pictures of a woman shooting a bow and arrow. Not only is she showing extremely poor form she's using the equipment shown in the photographs incorrectly. This is tantamount to using spinning gear with the reel positioned over the top of the fishing pole. If they want to cover hunting they should at least hire a photo editor that knows what (s)he's looking at. If you want a hunting magazine buy something else...

- other examples:
  - Reuters, $\sim$ 100 hierarchical categories
  - classification according to a library system (LCC, SAB)
  - ...by target group (e.g. CEFR readability) or some property of the author (e.g. gender, native language)

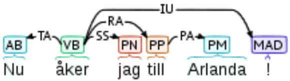
example: disambiguation of word meaning in context

A woman and child suffered minor injuries after the car they were riding in crashed into a ***rock*** *wall Tuesday morning.*

- what is the meaning of *rock* in this context?

- S: (n) **rock**, stone (a lump or mass of hard consolidated mineral matter) *"he threw a rock at me"*
- S: (n) **rock**, stone (material consisting of the aggregate of minerals like those making up the Earth's crust) *"that mountain is solid rock"; "stone is abundant in New England and there are many quarries"*
- S: (n) **Rock**, John Rock (United States gynecologist and devout Catholic who conducted the first clinical trials of the oral contraceptive pill (1890-1984))
- S: (n) **rock** ((figurative) someone who is strong and stable and dependable) *"he was her rock during the crisis"; "Thou art Peter, and upon this rock I will build my church"--Gospel According to Matthew*
- S: (n) rock candy, **rock** (hard bright-colored stick candy (typically flavored with peppermint))
- S: (n) rock 'n' roll, rock'n'roll, rock-and-roll, rock and roll, **rock**, rock music (a genre of popular music originating in the 1950s; a blend of black rhythm-and-blues with white country-and-western) *"rock is a generic term for the range of styles that evolved out of rock'n'roll."*
- S: (n) **rock**, careen, sway, tilt (pitching dangerously to one side)

example: classification of grammatical relations



- what is the grammatical relation between *åker* and *till*?
  - e.g. subject, object, adverbial, ...

example: classification of discourse relations

Mary had to study hard. Her exam was only one week away.

- what is the discourse/rhetorical relation between the two sentences?
  - e.g. IF, THEN, AND, BECAUSE, BUT, ...

features for classification

- to be able to classify an object, we must describe its properties: **features**
- useful information that we believe helps us tell the classes apart
- this is an art more than a science
- examples:
  - in document classification, typically the **words**
  - . . . but also stylistic features such as sentence length, word variation, syntactic complexity

representation of features

- depending on the task we are trying to solve, features may be viewed in different ways
- **bag of words**: `["I", "love", "this", "film"]`
- **attribute–value pairs**: `{"age"=63, "gender"="F", "income"=25000}`
- **geometric vector**: `[0, 0, 0, 0, 1, 0, 0, 2, 0, 0, 1]`
- in this lecture and in the assignments, we will use the bag of words representation

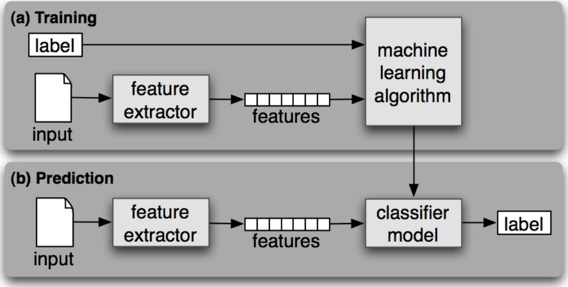
a note on terminology

- we want to develop some NLP system (a classifier, a tagger, a parser, ...) by getting some parameters from the data instead of hard-coding (**data-driven**)
- a statistician would say that we **estimate** parameters of a model
- a computer scientist would say that we **train** the model
  - or conversely, that we apply a **machine learning** algorithm
- in the machine learning course this fall, we will see several such algorithms
  - including algorithms that are not motivated by probabilities and statistical theory

training sets

- we are given a set of **examples** (e.g. reviews)
- each example comes with a **gold-standard** positive or negative class label
- we then use these examples to estimate the parameters of our statistical model
- the model can then be used to classify reviews we haven't seen before

overview


(a) Training
label
machine learning algorithm
feature extractor
features
input
(b) Prediction
feature extractor
features
classifier model
label
input

scientific hygiene in experiments

- in addition to the training set, we have a **test set** that we use when estimating the accuracy (or P, R, etc)
- like the training set, the test set also contains gold-standard labels
- the training and test sets should be distinct!
- also, **don't use the test set for optimization**!
  - use a separate development set instead

overview

introduction

Naive Bayes definition and generative models

estimation in the Naive Bayes model

discriminative models

the next few weeks

Naive Bayes

- Naive Bayes is a classification method based on a simple probability model
- recall from the NLP course:

$$\begin{aligned} P(f_1, \ldots, f_n, class) &= P(class) \cdot P(f_1, \ldots, f_n | class) \\ &= P(class) \cdot P(f_1 | class) \cdot \ldots \cdot P(f_n | class) \end{aligned}$$

- for instance: $f_1, \ldots, f_n$ are the words occurring in the document, and *class* is positive or negative
- if we have these probabilities, then we can guess the class of an unseen example (just find the class that maximizes P)

$$guess = \arg\max_{class} P(f_1, \ldots, f_n, class)$$

UNIVERSITY OF GOTHENBURG

Naive Bayes as a generative model

- Naive Bayes is an example of a **generative graphical model**
- a generative graphical model is defined in terms of a "generative story" that describes how the data was created
- a generative model computes the joint probability

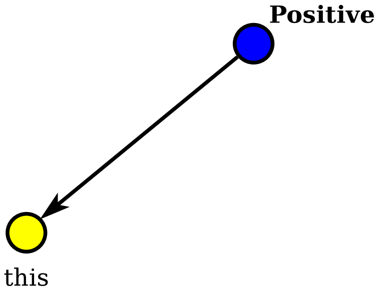
$$P(\text{input}, \text{output})$$

- we can draw them using **plate diagrams**

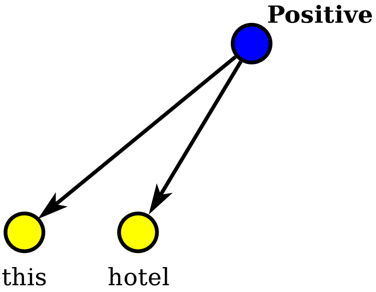
generative story in Naive Bayes

generative story in Naive Bayes

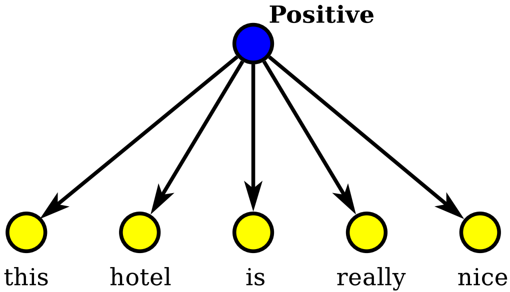
generative story in Naive Bayes



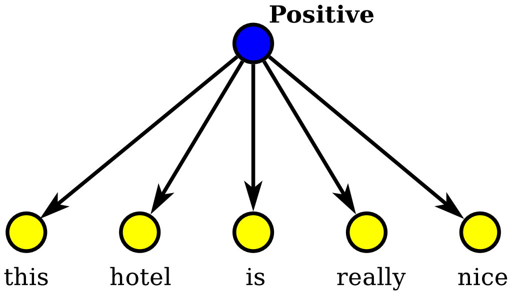
generative story in Naive Bayes



generative story in Naive Bayes

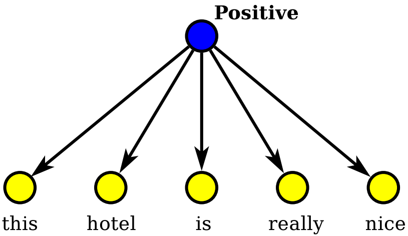


generative story in Naive Bayes

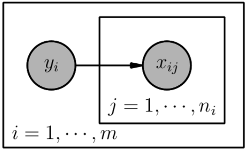


- the model gives us $P(\text{this hotel is really nice}, \text{Positive})$

a plate diagram for Naive Bayes

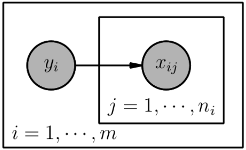


- this "story" can be represented using a plate diagram:

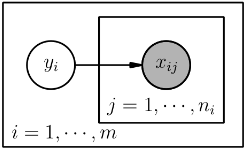


explanation of the plate diagram (1)

- grey balls represent observed variables and white balls unobserved
  - supervised NB: we see the words and the document classes

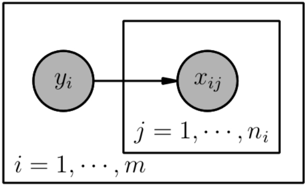


  - unsupervised NB: we don't see the document classes



explanation of the plate diagram (2)

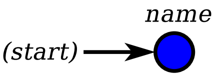
- the arrows represent how we model probabilities
  - the probability of a word x_{ij} is defined in terms of the document class y_i
- the rectangles (the "plates") represent repetion (a "for loop"):
  - the collection consists of documents $i = 1, \ldots, m$
  - each document consists of words $j = 1, \ldots, n_i$



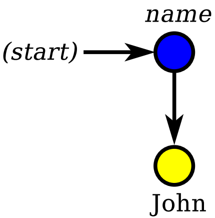
generative story in hidden Markov models

(start)

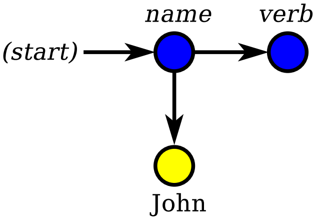
generative story in hidden Markov models



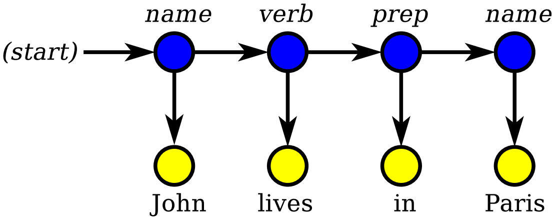
generative story in hidden Markov models



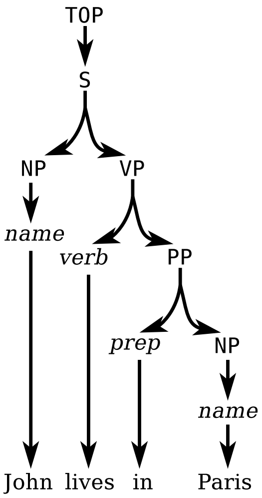
generative story in hidden Markov models



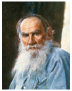
generative story in hidden Markov models



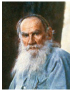
generative story in PCFGs

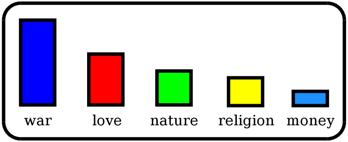


generative story in topic models (simplified)

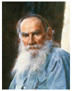


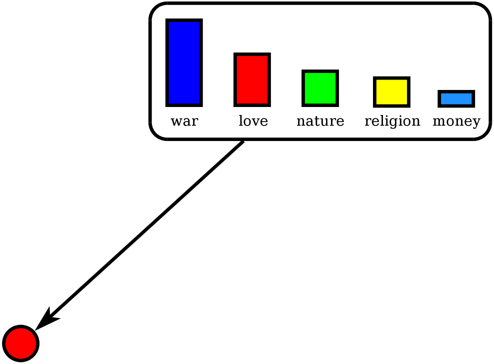
generative story in topic models (simplified)



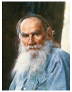


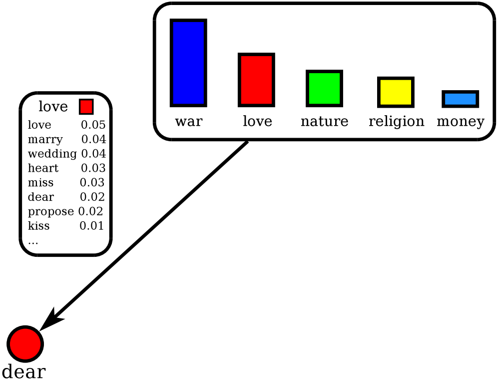
generative story in topic models (simplified)



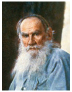


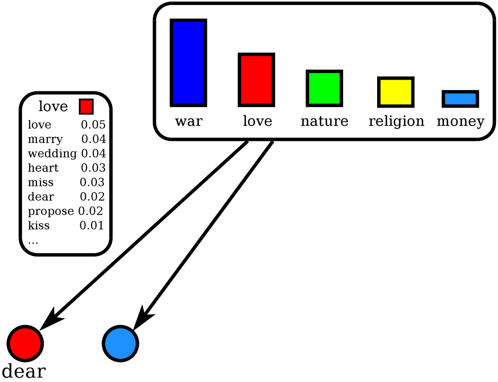
generative story in topic models (simplified)



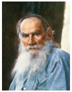


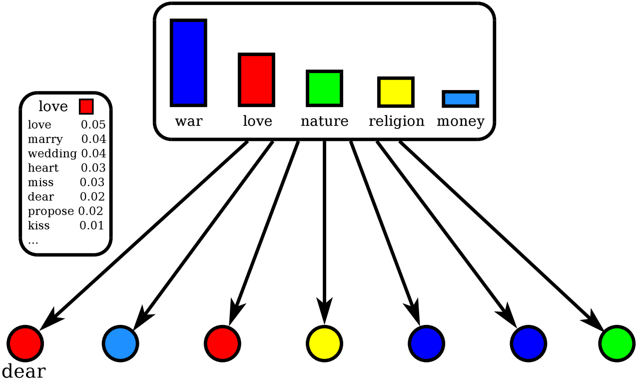
generative story in topic models (simplified)





generative story in topic models (simplified)





UNIVERSITY OF GOTHENBURG

overview

introduction

Naive Bayes definition and generative models

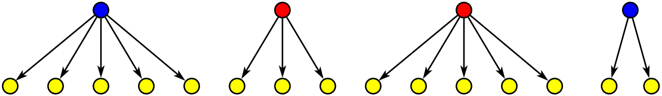
estimation in the Naive Bayes model

discriminative models

the next few weeks

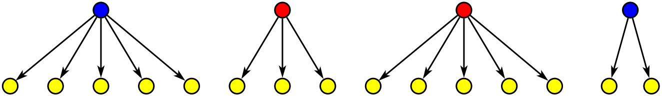
what kind of information is available?

- **supervised** learning: the desired output classes are given

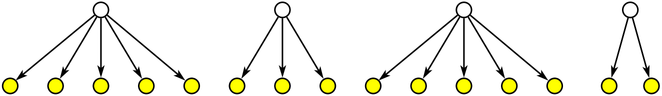


what kind of information is available?

- **supervised** learning: the desired output classes are given

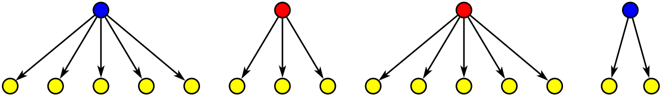


- **unsupervised** learning: the classes are not given

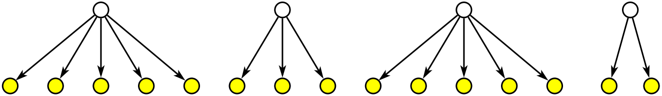


what kind of information is available?

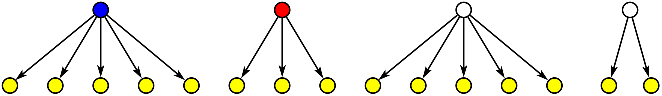
- **supervised** learning: the desired output classes are given



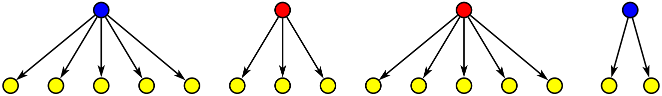
- **unsupervised** learning: the classes are not given



- **semisupervised** learning: some of the classes are given

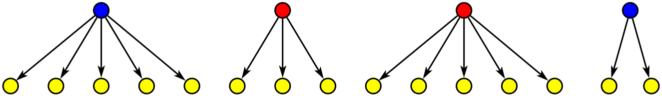


estimation in supervised Naive Bayes



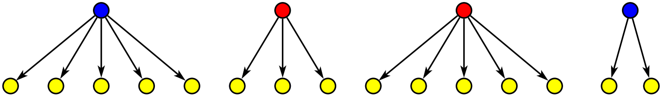
- we are given a set of documents labeled with classes
- to be able to guess the class of new unseen documents, we estimate the parameters of the model:
  - the probability of each class
  - the probabilities of the features (words) given the class
- in the supervised case, this is unproblematic

estimation of the class probabilities



- we observe two positive (blue) documents out of four
- how do we estimate $P(\text{positive})$?

estimation of the class probabilities

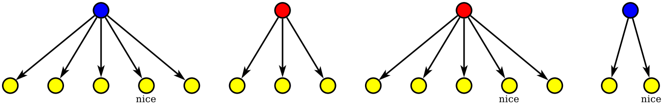


- we observe two positive (blue) documents out of four
- how do we estimate $P(\text{positive})$?
- maximum likelihood estimate

$$P_{\text{MLE}}(\text{positive}) = \frac{\text{count}(\text{positive})}{\text{count}(\text{all})} = \frac{2}{4}$$

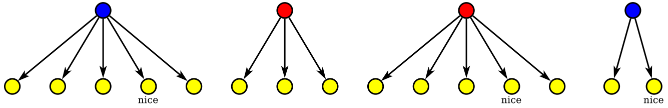
(four observations of a coin-toss variable)

estimation of the feature probabilities



- how do we estimate $P(\text{“nice”}|\text{positive})$?

estimation of the feature probabilities



- how do we estimate $P(\text{“nice”}|\text{positive})$?
- maximum likelihood estimate

$$P_{\mathrm{MLE}}(\text{“nice”}|\text{positive}) = \frac{\text{count}(\text{“nice”}, \text{positive})}{\text{count}(\textit{any word}, \text{positive})} = \frac{2}{7}$$

UNIVERSITY OF GOTHENBURG

dealing with zeros

- zero counts are as usual a problem for MLE estimates!
- smoothing is needed

Laplace smoothing: add one to each count

- **Laplace smoothing**: add one to all counts

$$P_{Laplace}(word|class) = \frac{\text{count}(\text{word, class}) + 1}{\text{count}(\textit{any word}, \text{class}) + \text{ voc size}}$$

$$P_{Laplace}(\text{"nice"}|\text{positive}) = \frac{2+1}{7+12345}$$

overview

introduction

Naive Bayes definition and generative models

estimation in the Naive Bayes model

discriminative models

the next few weeks

UNIVERSITY OF GOTHENBURG

generative vs. discriminative models

- recall that a **generative** model computes the joint probability

$$P(\text{input}, \text{output})$$

  and is defined in terms of a "generative story"

- other types of classifiers are called **discriminative**:
  - they can compute some other probability instead – for instance $P(\text{output}|\text{input})$
  - or classify in some other way without probabilities!

some types of discriminative classifiers

- logistic regression: maximum likelihood of $P(\text{output}|\text{input})$ (read on your own)
- many other types of classifiers, e.g. decision trees (Simon's lecture)
- we will now study a very simple approach based on dictionary lookup in a weight table
  - we'll consider the use case of classifying reviews, like in your assignment

first idea: use a polarity wordlist

- ... for instance the MPQA list

```
type=strongsubj len=1 word1=wretchedly pos1=anypos stemmed1=n priorpolarity=negative
type=strongsubj len=1 word1=wretchedness pos1=noun stemmed1=n priorpolarity=negative
type=weaksubj len=1 word1=writhe pos1=verb stemmed1=y priorpolarity=negative
type=weaksubj len=1 word1=wrong pos1=adj stemmed1=n priorpolarity=negative
type=weaksubj len=1 word1=wrong pos1=anypos stemmed1=y priorpolarity=negative
type=weaksubj len=1 word1=wrongful pos1=adj stemmed1=n priorpolarity=negative
type=strongsubj len=1 word1=wrongly pos1=anypos stemmed1=y priorpolarity=negative
type=weaksubj len=1 word1=wrought pos1=adj stemmed1=n priorpolarity=negative
type=weaksubj len=1 word1=wrought pos1=noun stemmed1=n priorpolarity=negative
type=strongsubj len=1 word1=wry pos1=adj stemmed1=n priorpolarity=positive
type=strongsubj len=1 word1=yawn pos1=noun stemmed1=n priorpolarity=negative
type=strongsubj len=1 word1=yawn pos1=verb stemmed1=y priorpolarity=negative
type=strongsubj len=1 word1=yeah pos1=anypos stemmed1=y priorpolarity=neutral
type=strongsubj len=1 word1=yearn pos1=verb stemmed1=y priorpolarity=positive
type=strongsubj len=1 word1=yearning pos1=noun stemmed1=n priorpolarity=positive
type=strongsubj len=1 word1=yearningly pos1=anypos stemmed1=n priorpolarity=positive
type=strongsubj len=1 word1=yelp pos1=verb stemmed1=y priorpolarity=negative
type=strongsubj len=1 word1=yep pos1=anypos stemmed1=y priorpolarity=positive
type=strongsubj len=1 word1=yes pos1=anypos stemmed1=y priorpolarity=positive
type=weaksubj len=1 word1=youthful pos1=adj stemmed1=n priorpolarity=positive
type=strongsubj len=1 word1=zeal pos1=noun stemmed1=n priorpolarity=positive
type=strongsubj len=1 word1=zealot pos1=noun stemmed1=n priorpolarity=negative
type=strongsubj len=1 word1=zealous pos1=adj stemmed1=n priorpolarity=negative
type=strongsubj len=1 word1=zealously pos1=anypos stemmed1=n priorpolarity=negative
type=strongsubj len=1 word1=zenith pos1=noun stemmed1=n priorpolarity=positive
type=strongsubj len=1 word1=zest pos1=noun stemmed1=n priorpolarity=positive
```

document sentiment polarity by summing word scores

- store all MPQA polarity values in a table as numerical values
- e.g. 2 points for strong positive, -1 point for weak negative
- predict the overall polarity value of the document by summing the scores of each word occurring

```
def guess_sentiment_polarity(document, weights):
    score = 0
    for word in document:
        score += weights[word]
    if score >= 0:
        return "pos"
    else:
        return "neg"
```

experiment

- we evaluate on 50% of a sentiment dataset
  http://www.cs.jhu.edu/~mdredze/datasets/sentiment/

```
def evaluate(labeled_documents, weights):
    ncorrect = 0
    for class_label, document in labeled_documents:
        guess = guess_sentiment_polarity(document, weights)
        if guess == class_label:
            ncorrect += 1
    return ncorrect / len(labeled_documents)
```

- this is a balanced dataset, coin-toss accuracy would be 50%
- with MPQA, we get an accuracy of 59.5%

can we do better?

- it's hard to set the word weights
- what if we don't even have a resource such as MPQA?
- can we set the weights automatically?

an idea for setting the weights automatically

- start with an empty weight table (instead of using MPQA)
- classify documents according to the current weight table
- each time we misclassify, change the weight table a bit
  - if a positive document was misclassified, add 1 to the weight of each word in the document
  - and conversely ...

an idea for setting the weights automatically

- start with an empty weight table (instead of using MPQA)
- classify documents according to the current weight table
- each time we misclassify, change the weight table a bit
  - if a positive document was misclassified, add 1 to the weight of each word in the document
  - and conversely ...

```
def train_by_errors(labeled_documents):
    weights = Counter()
    for class_label, document in labeled_documents:
        guess = guess_sentiment_polarity(document, weights)
        if class_label == "pos" and guess == "neg":
            for word in document:
                weights[word] += 1
        elif class_label == "neg" and guess == "pos":
            for word in document:
                weights[word] -= 1
    return weights
```

new experiment

- we compute the weights using 50% of the sentiment data and test on the other half
- the accuracy is 81.4%, up from the 59.5% we had when we used the MPQA
- `train_by_errors` is called the **perceptron** algorithm and is one of the most widely used machine learning algorithms

examples of the weights

amazing 171
easy 124
perfect 109
highly 108
five 107
excellent 104
enjoy 93
job 92
question 90
wonderful 90
performance 83
those 80
r&b 80
loves 79
best 78
recommended 77
favorite 77
included 76
medical 75
america 74

waste -175
worst -168
boring -154
poor -134
‘ -130
unfortunately -122
horrible -118
ok -111
disappointment -109
unless -108
called -103
example -100
bad -100
save -99
bunch -98
talk -96
useless -95
author -94
effort -94
oh -94

the same thing with scikit-learn

- to train a classifier:

```
vec = DictVectorizer()
clf = Perceptron(n_iter=20)
clf.fit(vec.fit_transform(train_docs),
        numpy.array(train_targets))
```

- to classify a new instance:

```
guess = clf.predict(vec.transform(doc))
```

- more about classification and scikit-learn in the course on **machine learning**

an aside: domain sensitivity

- a common problem with classifiers (and NLP systems in general) is **domain sensitivity**: they work best on the type of texts used when developing
  - a review classifier for book reviews won't work as well for health product reviews

	book	health
book	0.75	0.64
health	0.68	0.80

- it may depend on the domain which words are informative, and also what sentiment they have
  - for instance, **small** may be a good thing about a camera but not about a hotel room

overview

introduction

Naive Bayes definition and generative models

estimation in the Naive Bayes model

discriminative models

the next few weeks

the computer assignments

- assignment 1: implement a Naive Bayes classifier and use it to group customer reviews into positive and negative
  - optionally: implement the perceptron as well, or use scikit-learn
- February 9 and 11
- report deadline: February 25
- assignment 2: a statistical analysis of the performance of your classifier(s)

next lectures

- February 16 (in the lab): comparing classifiers
- February 23 (here): tagging with HMM models