Text Classification and Naive Bayes

The Task of Text Classification
Is this spam?

Subject: Important notice!
From: Stanford University <newsforum@stanford.edu>
Date: October 28, 2011 12:34:16 PM PDT
To: undisclosed-recipients:;

Greats News!

You can now access the latest news by using the link below to login to Stanford University News Forum.


Click on the above link to login for more information about this new exciting forum. You can also copy the above link to your browser bar and login for more information about the new services.

© Stanford University. All Rights Reserved.
Who wrote which Federalist papers?

1787-8: anonymous essays try to convince New York to ratify the U.S. Constitution: Jay, Madison, Hamilton.

Authorship of 12 of the letters in dispute

1963: solved by Mosteller and Wallace using Bayesian methods

James Madison

Alexander Hamilton
What is the subject of this medical article?

MEDLINE Article

MeSH Subject Category Hierarchy
Antagonists and Inhibitors
Blood Supply
Chemistry
Drug Therapy
Embryology
Epidemiology
...

Syntactic frame and word bias in aphasia: Plausibility judgments of undergenerated sentences
Rosaella Colli, Linh Nguyen, Carl Rooijen, Daniel R. Jurafsky, Elizabeth闪闪, "Nancy Brenega, "and L. Daniel Jurafsky"

...
Positive or negative movie review?

+ ...zany characters and richly applied satire, and some great plot twists

− It was pathetic. The worst part about it was the boxing scenes...

+ ...awesome caramel sauce and sweet toasty almonds. I love this place!

− ...awful pizza and ridiculously overpriced...
Positive or negative movie review?

+ ...zany characters and richly applied satire, and some great plot twists

− It was pathetic. The worst part about it was the boxing scenes...

− ...awesome caramel sauce and sweet toasty almonds. I love this place!

+ ...awful pizza and ridiculously overpriced...
Why sentiment analysis?

Movie: is this review positive or negative?
Products: what do people think about the new iPhone?
Public sentiment: how is consumer confidence?
Politics: what do people think about this candidate or issue?
Prediction: predict election outcomes or market trends from sentiment
Scherer Typology of Affective States

**Emotion**: brief organically synchronized ... evaluation of a major event
- angry, sad, joyful, fearful, ashamed, proud, elated

**Mood**: diffuse non-caused low-intensity long-duration change in subjective feeling
- cheerful, gloomy, irritable, listless, depressed, buoyant

**Interpersonal stances**: affective stance toward another person in a specific interaction
- friendly, flirtatious, distant, cold, warm, supportive, contemptuous

**Attitudes**: enduring, affectively colored beliefs, dispositions towards objects or persons
- liking, loving, hating, valuing, desiring

**Personality traits**: stable personality dispositions and typical behavior tendencies
- nervous, anxious, reckless, morose, hostile, jealous
Scherer Typology of Affective States

**Emotion**: brief organically synchronized ... evaluation of a major event
- angry, sad, joyful, fearful, ashamed, proud, elated

**Mood**: diffuse non-caused low-intensity long-duration change in subjective feeling
- cheerful, gloomy, irritable, listless, depressed, buoyant

**Interpersonal stances**: affective stance toward another person in a specific interaction
- friendly, flirtatious, distant, cold, warm, supportive, contemptuous

**Attitudes**: enduring, affectively colored beliefs, dispositions towards objects or persons
- liking, loving, hating, valuing, desiring

**Personality traits**: stable personality dispositions and typical behavior tendencies
- nervous, anxious, reckless, morose, hostile, jealous
Basic Sentiment Classification

Sentiment analysis is the detection of attitudes

Simple task we focus on in this chapter
  ◦ Is the attitude of this text positive or negative?

We return to affect classification in later chapters
Summary: Text Classification

Sentiment analysis
Spam detection
Authorship identification
Language Identification
Assigning subject categories, topics, or genres...
...
Text Classification: definition

**Input:**
- a document \( d \)
- a fixed set of classes \( C = \{c_1, c_2, \ldots, c_j\} \)

**Output:** a predicted class \( c \in C \)
Classification Methods: Hand-coded rules

Rules based on combinations of words or other features
  ◦ spam: black-list-address OR ("dollars" AND "you have been selected")

Accuracy can be high
  ◦ If rules carefully refined by expert

But building and maintaining these rules is expensive
Classification Methods: 
Supervised Machine Learning

Input:
- a document $d$
- a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$
- A training set of $m$ hand-labeled documents $(d_1, c_1), \ldots, (d_m, c_m)$

Output:
- a learned classifier $\gamma: d \rightarrow c$
Classification Methods: Supervised Machine Learning

Any kind of classifier

- Naïve Bayes
- Logistic regression
- Neural networks
- k-Nearest Neighbors
- ...

Text Classification and Naive Bayes

The Task of Text Classification
The Naive Bayes Classifier
Naive Bayes Intuition

Simple ("naive") classification method based on Bayes rule

Relies on very simple representation of document
- Bag of words
I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!
The bag of words representation

\[ \gamma(\text{seen}) = 2 \]
\[ \gamma(\text{sweet}) = 1 \]
\[ \gamma(\text{whimsical}) = 1 \]
\[ \gamma(\text{recommend}) = 1 \]
\[ \gamma(\text{happy}) = 1 \]
\[ \ldots \]
\[ \ldots \]
Bayes’ Rule Applied to Documents and Classes

• For a document $d$ and a class $c$

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}$$
Naive Bayes Classifier (I)

\[ c_{\text{MAP}} = \arg\max_{c \in C} P(c | d) \]

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

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

MAP is “maximum a posteriori” = most likely class

Bayes Rule

Dropping the denominator
Naive Bayes Classifier (II)

\[ c_{MAP} = \arg\max_{c \in C} P(d | c)P(c) \]

\[ = \arg\max_{c \in C} P(x_1, x_2, \ldots, x_n | c)P(c) \]
Naïve Bayes Classifier (IV)

\[ c_{MAP} = \arg\max_{c \in C} P(x_1, x_2, \ldots, x_n \mid c) P(c) \]

- O(|X|^n \cdot |C|) parameters
- Could only be estimated if a very, very large number of training examples was available.
- How often does this class occur?
- We can just count the relative frequencies in a corpus.
Multinomial Naive Bayes Independence Assumptions

\[ P(x_1, x_2, \ldots, x_n \mid c) \]

**Bag of Words assumption:** Assume position doesn’t matter

**Conditional Independence:** Assume the feature probabilities \( P(x_i \mid c_j) \) are independent given the class \( c \).

\[ P(x_1, \ldots, x_n \mid c) = P(x_1 \mid c) \cdot P(x_2 \mid c) \cdot P(x_3 \mid c) \cdot \ldots \cdot P(x_n \mid c) \]
Multinomial Naive Bayes Classifier

\[ c_{MAP} = \arg\max_{c \in C} P(x_1, x_2, \ldots, x_n \mid c) P(c) \]

\[ c_{NB} = \arg\max_{c \in C} P(c_j) \prod_{x \in X} P(x \mid c) \]
Applying Multinomial Naive Bayes Classifiers to Text Classification

\[ c_{NB} = \arg\max_{c_j \in C} P(c_j) \prod_{i \in \text{positions}} P(x_i | c_j) \]
Problems with multiplying lots of probs

There's a problem with this:

\[ c_{NB} = \arg \max_{c_j \in C} P(c_j) \prod_{i \in \text{positions}} P(x_i | c_j) \]

Multiplying lots of probabilities can result in floating-point underflow!

\[ .0006 \times .0007 \times .0009 \times .01 \times .5 \times .000008 \ldots \]

Idea: Use logs, because \( \log(ab) = \log(a) + \log(b) \)

We'll sum logs of probabilities instead of multiplying probabilities!
We actually do everything in log space

Instead of this:

\[
    c_{NB} = \arg\max_{c_j \in C} P(c_j) \prod_{i \in \text{positions}} P(x_i | c_j)
\]

This:

\[
    c_{NB} = \arg\max_{c_j \in C} \left[ \log P(c_j) + \sum_{i \in \text{positions}} \log P(x_i | c_j) \right]
\]

Notes:

1) Taking log doesn't change the ranking of classes!
   The class with highest probability also has highest log probability!

2) It's a linear model:
   Just a max of a sum of weights: a \textbf{linear} function of the inputs
   So naive bayes is a \textbf{linear classifier}
Text Classification and Naive Bayes

The Naive Bayes Classifier
Naive Bayes: Learning
Learning the Multinomial Naive Bayes Model

First attempt: maximum likelihood estimates
  • simply use the frequencies in the data

\[
\hat{P}(c_j) = \frac{N_{c_j}}{N_{total}}
\]

\[
\hat{P}(w_i | c_j) = \frac{\text{count}(w_i, c_j)}{\sum_{w \in V} \text{count}(w, c_j)}
\]
Parameter estimation

\[ \hat{P}(w_i \mid c_j) = \frac{\text{count}(w_i, c_j)}{\sum_{w \in V} \text{count}(w, c_j)} \]

fraction of times word \( w_i \) appears among all words in documents of topic \( c_j \)

Create mega-document for topic \( j \) by concatenating all docs in this topic

- Use frequency of \( w \) in mega-document
Problem with Maximum Likelihood

What if we have seen no training documents with the word *fantastic* and classified in the topic *positive (thumbs-up)*?

$$\hat{P}(\text{"fantastic" } | \text{positive}) = \frac{\text{count(\"fantastic", positive)}}{\sum_{w \in V} \text{count}(w, \text{positive})} = 0$$

Zero probabilities cannot be conditioned away, no matter the other evidence!

$$c_{MAP} = \text{argmax}_c \hat{P}(c) \prod_i \hat{P}(x_i | c)$$
Laplace (add-1) smoothing for Naïve Bayes

\[
\hat{P}(w_i | c) = \frac{\text{count}(w_i, c) + 1}{\sum_{w \in V} (\text{count}(w, c) + 1)}
= \frac{\text{count}(w_i, c) + 1}{\left(\sum_{w \in V} \text{count}(w, c)\right) + |V|}
\]
Multinomial Naïve Bayes: Learning

- From training corpus, extract *Vocabulary*

**Calculate $P(c_j)$ terms**
- For each $c_j$ in $C$ do
  
  $docs_j \leftarrow$ all docs with class $= c_j$

  $$P(c_j) \leftarrow \frac{|docs_j|}{|total \# \ documents|}$$

**Calculate $P(w_k \mid c_j)$ terms**
- $Text_j \leftarrow$ single doc containing all $docs_j$
- For each word $w_k$ in *Vocabulary*
  
  $n_k \leftarrow$ # of occurrences of $w_k$ in $Text_j$

  $$P(w_k \mid c_j) \leftarrow \frac{n_k + \alpha}{n + \alpha \mid Vocabulay \mid}$$
Unknown words

What about unknown words
- that appear in our test data
- but not in our training data or vocabulary?

We *ignore* them
- Remove them from the test document!
- Pretend they weren't there!
- Don't include any probability for them at all!

Why don't we build an unknown word model?
- It doesn't help: knowing which class has more unknown words is not generally helpful!
Stop words

Some systems ignore stop words

- **Stop words**: very frequent words like *the* and *a*.
  - Sort the vocabulary by word frequency in training set
  - Call the top 10 or 50 words the **stopword list**.
  - Remove all stop words from both training and test sets
    - As if they were never there!

But removing stop words doesn't usually help

- So in practice most NB algorithms use **all** words and **don't** use stopword lists
Naive Bayes: Learning
Text Classification and Naive Bayes

Sentiment and Binary Naive Bayes
Let's do a worked sentiment example!

<table>
<thead>
<tr>
<th></th>
<th>Cat</th>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>-</td>
<td>just plain boring</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>entirely predictable and lacks energy</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>no surprises and very few laughs</td>
</tr>
<tr>
<td></td>
<td>+</td>
<td>very powerful</td>
</tr>
<tr>
<td></td>
<td>+</td>
<td>the most fun film of the summer</td>
</tr>
<tr>
<td>Test</td>
<td>?</td>
<td>predictable with no fun</td>
</tr>
</tbody>
</table>
A worked sentiment example with add-1 smoothing

<table>
<thead>
<tr>
<th>Cat</th>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>- just plain boring</td>
</tr>
<tr>
<td></td>
<td>- entirely predictable and lacks energy</td>
</tr>
<tr>
<td></td>
<td>- no surprises and very few laughs</td>
</tr>
<tr>
<td></td>
<td>+ very powerful</td>
</tr>
<tr>
<td></td>
<td>+ the most fun film of the summer</td>
</tr>
<tr>
<td>Test</td>
<td>? predictable with no fun</td>
</tr>
</tbody>
</table>

### 1. Prior from training:

\[
P(c_j) = \frac{N_{c_j}}{N_{total}}
\]

\( P(-) = 3/5 \) \( P(+) = 2/5 \)

### 2. Drop "with"

### 3. Likelihoods from training:

\[
p(w_i|c) = \frac{\text{count}(w_i, c) + 1}{(\sum_{w\in V} \text{count}(w, c)) + |V|}
\]

\[
P(\text{"predictable"}|-) = \frac{1+1}{14+20} \quad P(\text{"predictable"}|+) = \frac{0+1}{9+20}
\]

\[
P(\text{"no"}|-) = \frac{1+1}{14+20} \quad P(\text{"no"}|+) = \frac{0+1}{9+20}
\]

\[
P(\text{"fun"}|-) = \frac{0+1}{14+20} \quad P(\text{"fun"}|+) = \frac{1+1}{9+20}
\]

### 4. Scoring the test set:

\[
P(-)P(S|-) = \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} = 6.1 \times 10^{-5}
\]

\[
P(+)P(S|+) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}
\]
Optimizing for sentiment analysis

For tasks like sentiment, word **occurrence** seems to be more important than word **frequency**.

- The occurrence of the word *fantastic* tells us a lot
- The fact that it occurs 5 times may not tell us much more.

**Binary multinominal naive bayes, or binary NB**

- Clip our word counts at 1
- Note: this is different than Bernoulli naive bayes; see the textbook at the end of the chapter.
Binary Multinomial Naïve Bayes: Learning

- From training corpus, extract *Vocabulary*

**Calculate $P(c_j)$ terms**
- For each $c_j$ in $C$
  - $docs_j \leftarrow$ all docs with class $= c_j$
  - $P(c_j) \leftarrow \frac{|docs_j|}{|\text{total # documents}|}$

**Calculate $P(w_k \mid c_j)$ terms**
- Remove single doc containing all $docs_j$
- For each word type $w_k$ in vocabulary
  - Retain only a single instance of $w_k$
  - $n_k \leftarrow$ # of occurrences of $w_k$ in $Text_j$
  - $P(w_k \mid c_j) \leftarrow \frac{n_k + \alpha}{n + \alpha |Vocabulary|}$
First remove all duplicate words from $d$

Then compute NB using the same equation:

$$c_{NB} = \arg\max_{c_j \in C} \prod_{i \in \text{positions}} P(w_i | c_j) \cdot P(c_j)$$
Binary multinomial naive Bayes

Four original documents:
- it was pathetic the worst part was the boxing scenes
- no plot twists or great scenes
+ and satire and great plot twists
+ great scenes great film
Binary multinominal naive Bayes

Four original documents:

- it was pathetic the worst part was the boxing scenes
- no plot twists or great scenes
- and satire and great plot twists
- great scenes great film

<table>
<thead>
<tr>
<th>NB Counts</th>
<th>+</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>and</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>boxing</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>film</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>great</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>it</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>no</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>or</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>part</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>pathetic</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>plot</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>satire</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>scenes</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>the</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>twists</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>was</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>worst</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Four original documents:

- it was pathetic the worst part was the boxing scenes
- no plot twists or great scenes
+ and satire and great plot twists
+ great scenes great film

After per-document binarization:

- it was pathetic the worst part boxing scenes
- no plot twists or great scenes
+ and satire great plot twists
+ great scenes film

<table>
<thead>
<tr>
<th>Word</th>
<th>NB Counts</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>and</td>
<td>+2</td>
<td>-0</td>
</tr>
<tr>
<td>boxing</td>
<td>+0</td>
<td>-1</td>
</tr>
<tr>
<td>film</td>
<td>+1</td>
<td>-0</td>
</tr>
<tr>
<td>great</td>
<td>+3</td>
<td>-1</td>
</tr>
<tr>
<td>it</td>
<td>+0</td>
<td>-1</td>
</tr>
<tr>
<td>no</td>
<td>+0</td>
<td>-1</td>
</tr>
<tr>
<td>or</td>
<td>+0</td>
<td>-1</td>
</tr>
<tr>
<td>part</td>
<td>+0</td>
<td>-1</td>
</tr>
<tr>
<td>pathetic</td>
<td>+0</td>
<td>-1</td>
</tr>
<tr>
<td>plot</td>
<td>+1</td>
<td>-1</td>
</tr>
<tr>
<td>satire</td>
<td>+1</td>
<td>-0</td>
</tr>
<tr>
<td>scenes</td>
<td>+1</td>
<td>-2</td>
</tr>
<tr>
<td>the</td>
<td>+0</td>
<td>-2</td>
</tr>
<tr>
<td>twists</td>
<td>+1</td>
<td>-1</td>
</tr>
<tr>
<td>was</td>
<td>+0</td>
<td>-2</td>
</tr>
<tr>
<td>worst</td>
<td>+0</td>
<td>-1</td>
</tr>
</tbody>
</table>
## Binary multinominal naive Bayes

**Four original documents:**
- it was pathetic the worst part was the boxing scenes
- no plot twists or great scenes
- and satire and great plot twists
- great scenes great film

**After per-document binarization:**
- it was pathetic the worst part boxing scenes
- no plot twists or great scenes
- and satire great plot twists
- great scenes film

<table>
<thead>
<tr>
<th>Word</th>
<th>NB Counts</th>
<th>Binary Counts</th>
</tr>
</thead>
<tbody>
<tr>
<td>and</td>
<td>+ 2 - 0</td>
<td>+ 1 - 0</td>
</tr>
<tr>
<td>boxing</td>
<td>+ 0 - 1</td>
<td>+ 0 - 1</td>
</tr>
<tr>
<td>film</td>
<td>+ 1 - 0</td>
<td>+ 1 - 0</td>
</tr>
<tr>
<td>great</td>
<td>+ 3 - 1</td>
<td>+ 2 - 1</td>
</tr>
<tr>
<td>it</td>
<td>+ 0 - 1</td>
<td>+ 0 - 1</td>
</tr>
<tr>
<td>no</td>
<td>+ 0 - 1</td>
<td>+ 0 - 1</td>
</tr>
<tr>
<td>or</td>
<td>+ 0 - 1</td>
<td>+ 0 - 1</td>
</tr>
<tr>
<td>part</td>
<td>+ 0 - 1</td>
<td>+ 0 - 1</td>
</tr>
<tr>
<td>pathetic</td>
<td>+ 0 - 1</td>
<td>+ 0 - 1</td>
</tr>
<tr>
<td>plot</td>
<td>+ 1 - 1</td>
<td>+ 1 - 1</td>
</tr>
<tr>
<td>satire</td>
<td>+ 1 - 0</td>
<td>+ 1 - 0</td>
</tr>
<tr>
<td>scenes</td>
<td>+ 1 - 2</td>
<td>+ 1 - 2</td>
</tr>
<tr>
<td>the</td>
<td>+ 0 - 2</td>
<td>+ 0 - 1</td>
</tr>
<tr>
<td>twists</td>
<td>+ 1 - 1</td>
<td>+ 1 - 1</td>
</tr>
<tr>
<td>was</td>
<td>+ 0 - 2</td>
<td>+ 0 - 1</td>
</tr>
<tr>
<td>worst</td>
<td>+ 0 - 1</td>
<td>+ 0 - 1</td>
</tr>
</tbody>
</table>

Counts can still be 2! Binarization is within-doc!
Text Classification and Naive Bayes

Sentiment and Binary Naive Bayes
Text Classification and Naive Bayes

More on Sentiment Classification
Sentiment Classification: Dealing with Negation

I really like this movie
I really don't like this movie

Negation changes the meaning of "like" to negative.
Negation can also change negative to positive-ish
  ◦ Don't dismiss this film
  ◦ Doesn't let us get bored
Sentiment Classification: Dealing with Negation


Simple baseline method:
Add NOT_ to every word between negation and following punctuation:

didn’t like this movie, but I

didn’t NOT_like NOT_this NOT_movie but I
Sentiment Classification: Lexicons

Sometimes we don't have enough labeled training data.

In that case, we can make use of pre-built word lists called **lexicons**.

There are various publicly available lexicons.
MPQA Subjectivity Cues Lexicon


Home page: [https://mpqa.cs.pitt.edu/lexicons/subj_lexicon/](https://mpqa.cs.pitt.edu/lexicons/subj_lexicon/)

6885 words from 8221 lemmas, annotated for intensity (strong/weak)

- 2718 positive
- 4912 negative

+ : admirable, beautiful, confident, dazzling, ecstatic, favor, glee, great

- : awful, bad, bias, catastrophe, cheat, deny, envious, foul, harsh, hate
The General Inquirer


- Home page: http://www.wjh.harvard.edu/~inquirer
- List of Categories: http://www.wjh.harvard.edu/~inquirer/homecat.htm
- Spreadsheet: http://www.wjh.harvard.edu/~inquirer/inquirerbasic.xls

Categories:
- Positiv (1915 words) and Negativ (2291 words)
- Strong vs Weak, Active vs Passive, Overstated versus Understated
- Pleasure, Pain, Virtue, Vice, Motivation, Cognitive Orientation, etc

Free for Research Use
Using Lexicons in Sentiment Classification

Add a feature that gets a count whenever a word from the lexicon occurs

- E.g., a feature called "this word occurs in the positive lexicon" or "this word occurs in the negative lexicon"

Now all positive words (good, great, beautiful, wonderful) or negative words count for that feature.

Using 1-2 features isn’t as good as using all the words.

- But when training data is sparse or not representative of the test set, dense lexicon features can help
Naive Bayes in Other tasks: Spam Filtering

SpamAssassin Features:

- Mentions millions of (dollar) ((dollar) NN,NNN,NNN.NN)
- From: starts with many numbers
- Subject is all capitals
- HTML has a low ratio of text to image area
- "One hundred percent guaranteed"
- Claims you can be removed from the list
Naive Bayes in Language ID

Determining what language a piece of text is written in. Features based on character n-grams do very well. Important to train on lots of varieties of each language (e.g., American English varieties like African-American English, or English varieties around the world like Indian English).
Summary: Naive Bayes is Not So Naive

Very Fast, low storage requirements
Work well with very small amounts of training data
Robust to Irrelevant Features
  Irrelevant Features cancel each other without affecting results
Very good in domains with many equally important features
  Decision Trees suffer from *fragmentation* in such cases – especially if little data
Optimal if the independence assumptions hold: If assumed independence is correct, then it is the Bayes Optimal Classifier for problem
A good dependable baseline for text classification
  But we will see other classifiers that give better accuracy
Text Classification and Naive Bayes

More on Sentiment Classification
Naïve Bayes: Relationship to Language Modeling
Generative Model for Multinomial Naïve Bayes

\[ C = + \]

\[ X_1 = l \]
\[ X_2 = \text{love} \]
\[ X_3 = \text{this} \]
\[ X_4 = \text{fun} \]
\[ X_5 = \text{film} \]
Naïve Bayes and Language Modeling

Naïve bayes classifiers can use any sort of feature
  ◦ URL, email address, dictionaries, network features

But if, as in the previous slides
  ◦ We use only word features
  ◦ we use all of the words in the text (not a subset)

Then
  ◦ Naive bayes has an important similarity to language modeling.
Each class = a unigram language model

Assigning each word: \( P(\text{word} \mid c) \)

Assigning each sentence: \( P(s \mid c) = \prod P(\text{word} \mid c) \)

<table>
<thead>
<tr>
<th>Class pos</th>
<th>0.1</th>
<th>I</th>
<th>0.1</th>
<th>love</th>
<th>0.1</th>
<th>this</th>
<th>0.05</th>
<th>fun</th>
<th>0.01</th>
<th>0.1</th>
<th>film</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td></td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>film</td>
</tr>
<tr>
<td>0.1</td>
<td></td>
<td>love</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.01</td>
<td></td>
<td>this</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.05</td>
<td></td>
<td>fun</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td></td>
<td>film</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( P(s \mid \text{pos}) = 0.0000005 \)
Naïve Bayes as a Language Model

Which class assigns the higher probability to s?

<table>
<thead>
<tr>
<th>Model pos</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>l</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>love</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.01</td>
<td>this</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.05</td>
<td>fun</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>film</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model neg</th>
<th>I</th>
<th>love</th>
<th>this</th>
<th>fun</th>
<th>film</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>0.1</td>
<td>0.001</td>
<td>0.01</td>
<td>0.05</td>
<td>0.1</td>
</tr>
<tr>
<td>0.1</td>
<td>0.1</td>
<td>0.01</td>
<td>0.005</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>0.01</td>
<td>0.001</td>
<td>0.01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.005</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$$P(s | \text{pos}) > P(s | \text{neg})$$
Naïve Bayes: Relationship to Language Modeling
Text Classification and Naïve Bayes

Precision, Recall, and F measure
Evaluation

Let's consider just binary text classification tasks

Imagine you're the CEO of Delicious Pie Company
You want to know what people are saying about your pies

So you build a "Delicious Pie" tweet detector
- Positive class: tweets about Delicious Pie Co
- Negative class: all other tweets
The 2-by-2 confusion matrix

<table>
<thead>
<tr>
<th>system output labels</th>
<th>gold standard labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>system positive</td>
<td>gold positive</td>
</tr>
<tr>
<td></td>
<td>true positive</td>
</tr>
<tr>
<td>system negative</td>
<td>gold negative</td>
</tr>
<tr>
<td></td>
<td>false positive</td>
</tr>
<tr>
<td></td>
<td>false negative</td>
</tr>
<tr>
<td></td>
<td>true negative</td>
</tr>
</tbody>
</table>

- **True Positive**: System correctly identifies the positive class.
- **False Positive**: System incorrectly identifies the positive class as negative.
- **False Negative**: System incorrectly identifies the negative class as positive.
- **True Negative**: System correctly identifies the negative class.

**Precision** = \( \frac{tp}{tp + fp} \)

**Recall** = \( \frac{tp}{tp + fn} \)

**Accuracy** = \( \frac{tp + tn}{tp + fp + tn + fn} \)

**Figure 4.4** A confusion matrix for visualizing how well a binary classification system performs against gold standard labels.

As it happens, the positive model assigns a higher probability to the sentence:

\[ P(s|\text{pos}) > P(s|\text{neg}) \]

Note that this is just the likelihood part of the naive Bayes model; once we multiply in the prior a full naive Bayes model might well make a different classification decision.
Why don't we use **accuracy** as our metric?

Imagine we saw 1 million tweets
- 100 of them talked about Delicious Pie Co.
- 999,900 talked about something else

We could build a dumb classifier that just labels every tweet "not about pie"
- It would get 99.99% accuracy!!! Wow!!!!
- But useless! Doesn't return the comments we are looking for!
- That's why we use **precision** and **recall** instead
Evaluation: Precision

% of items the system detected (i.e., items the system labeled as positive) that are in fact positive (according to the human gold labels)

\[
\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}
\]
Evaluation: Recall

% of items actually present in the input that were correctly identified by the system.

\[
\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}
\]
Why Precision and recall

Our dumb pie-classifier
  ◦ Just label nothing as "about pie"

Accuracy=99.99%
  but

Recall = 0
  ◦ (it doesn't get any of the 100 Pie tweets)

Precision and recall, unlike accuracy, emphasize true positives:
  ◦ finding the things that we are supposed to be looking for.
A combined measure: F

F measure: a single number that combines P and R:

\[ F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \]

We almost always use balanced \( F_1 \) (i.e., \( \beta = 1 \))

\[ F_1 = \frac{2PR}{P + R} \]
Development Test Sets ("Devsets") and Cross-validation

Train on training set, tune on devset, report on testset
- This avoids overfitting (‘tuning to the test set’)
- More conservative estimate of performance
- But paradox: want as much data as possible for training, and as much for dev; how to split?
Cross-validation: multiple splits

Pool results over splits, Compute pooled dev performance

Figure 4.6
Separate confusion matrices for the 3 classes from the previous figure, showing the pooled confusion matrix and the microaveraged and macroaveraged precision.

Training Iterations

<table>
<thead>
<tr>
<th>1</th>
<th>Training</th>
<th>Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Training</td>
<td>Dev</td>
</tr>
<tr>
<td>3</td>
<td>Training</td>
<td>Dev</td>
</tr>
<tr>
<td>4</td>
<td>Training</td>
<td>Dev</td>
</tr>
<tr>
<td>5</td>
<td>Training</td>
<td>Dev</td>
</tr>
<tr>
<td>6</td>
<td>Training</td>
<td>Dev</td>
</tr>
<tr>
<td>7</td>
<td>Training</td>
<td>Dev</td>
</tr>
<tr>
<td>8</td>
<td>Training</td>
<td>Dev</td>
</tr>
<tr>
<td>9</td>
<td>Training</td>
<td>Dev</td>
</tr>
<tr>
<td>10</td>
<td>Training</td>
<td>Dev</td>
</tr>
</tbody>
</table>

Testing

Test Set

Figure 4.7
10-fold cross-validation
Text Classification and Naive Bayes

Precision, Recall, and F measure
Text Classification and Naive Bayes

Evaluation with more than two classes
# Confusion Matrix for 3-class classification

The figure illustrates a confusion matrix for a three-class categorization task, showing for each pair of classes \((c_1, c_2)\), how many documents from \(c_1\) were (in)correctly assigned to \(c_2\). The matrix shows, for example, that the system mistakenly labeled one spam document as urgent, and we have shown how to compute a distinct precision and recall value for each class.

In order to derive a single metric that tells us how well the system is doing, we can combine these values in two ways. In **macroaveraging**, we compute the performance for each class, and then average over classes. In **microaveraging**, we collect the decisions for all classes into a single confusion matrix, and then compute precision and recall from that table.

Fig. 4.5 shows the confusion matrix for each class separately, and shows the computation of microaveraged and macroaveraged precision.

As the figure shows, a microaverage is dominated by the more frequent class (in this case spam), since the counts are pooled. The macroaverage better reflects the statistics of the smaller classes, and so is more appropriate when performance on all the classes is equally important.

## Test sets and Cross-validation

The training and testing procedure for text classification follows what we saw with language modeling (Section ??): we use the training set to train the model, then use the **development test set** (also called a **devset**) to perhaps tune some parameters, **development test set** **devset**
How to combine P/R from 3 classes to get one metric

Macroaveraging:
- compute the performance for each class, and then average over classes

Microaveraging:
- collect decisions for all classes into one confusion matrix
- compute precision and recall from that table.
Macroaveraging and Microaveraging

<table>
<thead>
<tr>
<th>System</th>
<th>Class 1: Urgent</th>
<th>Class 2: Normal</th>
<th>Class 3: Spam</th>
<th>Pooled</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>true urgent</td>
<td>true normal</td>
<td>true spam</td>
<td>true yes</td>
</tr>
<tr>
<td></td>
<td>false urgent</td>
<td>false not</td>
<td>false not</td>
<td>false no</td>
</tr>
<tr>
<td>System urgent</td>
<td>8</td>
<td>60</td>
<td>200</td>
<td>268</td>
</tr>
<tr>
<td>System not</td>
<td>11</td>
<td>55</td>
<td>33</td>
<td>99</td>
</tr>
<tr>
<td>System urgent</td>
<td>8</td>
<td>60</td>
<td>200</td>
<td>268</td>
</tr>
<tr>
<td>System not</td>
<td>340</td>
<td>212</td>
<td>83</td>
<td>635</td>
</tr>
</tbody>
</table>

Precision:

- Class 1: \( \text{precision} = \frac{8}{8+11} = .42 \)
- Class 2: \( \text{precision} = \frac{60}{60+55} = .52 \)
- Class 3: \( \text{precision} = \frac{200}{200+33} = .86 \)
- Macroaverage precision: \( \text{macroaverage precision} = \frac{.42+.52+.86}{3} = .60 \)

Microaverage precision:

\( \text{microaverage precision} = \frac{268}{268+99} = .73 \)
Text Classification and Naive Bayes

Evaluation with more than two classes
Text Classification and Naive Bayes

Statistical Significance Testing
How do we know if one classifier is better than another?

Given:
- Classifier A and B
- Metric $M$: $M(A, x)$ is the performance of $A$ on test set $x$
- $\delta(x)$: the performance difference between $A$, $B$ on $x$:
  - $\delta(x) = M(A, x) - M(B, x)$

- We want to know if $\delta(x) > 0$, meaning $A$ is better than $B$
- $\delta(x)$ is called the **effect size**
- Suppose we look and see that $\delta(x)$ is positive. Are we done?
- No! This might be just an accident of this one test set, or circumstance of the experiment. Instead:
Statistical Hypothesis Testing

Consider two hypotheses:

- Null hypothesis: A isn't better than B \( H_0 : \delta(x) \leq 0 \)
- A is better than B \( H_1 : \delta(x) > 0 \)

We want to rule out \( H_0 \)

We create a random variable \( X \) ranging over test sets

And ask, how likely, if \( H_0 \) is true, is it that among these test sets we would see the \( \delta(x) \) we did see?

- Formalized as the p-value:
  \[
P(\delta(X) \geq \delta(x)|H_0 \text{ is true})
  \]
Statistical Hypothesis Testing

\[ P(\delta(X) \geq \delta(x)|H_0 \text{ is true}) \]

- In our example, this p-value is the probability that we would see \( \delta(x) \) assuming \( H_0 (=A \text{ is not better than } B) \).
- If \( H_0 \) is true but \( \delta(x) \) is huge, that is surprising! Very low probability!
- A very small p-value means that the difference we observed is very unlikely under the null hypothesis, and we can reject the null hypothesis.
- Very small: .05 or .01
- A result(e.g., “A is better than B”) is statistically significant if the \( \delta \) we saw has a probability that is below the threshold and we therefore reject this null hypothesis.
How do we compute this probability?

In NLP, we don't tend to use parametric tests (like t-tests)

Instead, we use non-parametric tests based on sampling: artificially creating many versions of the setup.

For example, suppose we had created zillions of testsets $x'$.

Now we measure the value of $\delta(x')$ on each test set

That gives us a distribution

Now set a threshold (say .01).

So if we see that in 99% of the test sets $\delta(x) > \delta(x')$

We conclude that our original test set delta was a real delta and not an artifact.
Statistical Hypothesis Testing

Two common approaches:
- approximate randomization
- bootstrap test

Paired tests:
- Comparing two sets of observations in which each observation in one set can be paired with an observation in another.
- For example, when looking at systems A and B on the same test set, we can compare the performance of system A and B on each same observation $x_i$.
Text Classification and Naive Bayes

Statistical Significance Testing
Text Classification and Naive Bayes

The Paired Bootstrap Test
Bootstrap test

Efron and Tibshirani, 1993

Can apply to any metric (accuracy, precision, recall, F1, etc).

Bootstrap means to repeatedly draw large numbers of smaller samples with replacement (called bootstrap samples) from an original larger sample.
Bootstrap example

Consider a baby text classification example with a test set $x$ of 10 documents, using accuracy as metric.

Suppose these are the results of systems A and B on $x$, with 4 outcomes (A & B both right, A & B both wrong, A right/B wrong, A wrong/B right):

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>A%</th>
<th>B%</th>
<th>δ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>.70</td>
<td>.50</td>
<td>.20</td>
</tr>
</tbody>
</table>
Bootstrap example

Now we create, many, say, $b=10,000$ virtual test sets $x(i)$, each of size $n = 10$.

To make each $x(i)$, we randomly select a cell from row $x$, with replacement, 10 times:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>A%</th>
<th>B%</th>
<th>δ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>.70</td>
<td>.50</td>
<td>.20</td>
</tr>
<tr>
<td>x(1)</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>.60</td>
<td>.60</td>
<td>.00</td>
</tr>
<tr>
<td>x(2)</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>.60</td>
<td>.70</td>
<td>-.10</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x(b)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bootstrap example

Now we have a distribution! We can check how often A has an accidental advantage, to see if the original $\delta(x)$ we saw was very common.

Now assuming $H_0$, that means normally we expect $\delta(x')=0$

So we just count how many times the $\delta(x')$ we found exceeds the expected 0 value by $\delta(x)$ or more:

$$p\text{-value}(x) = \sum_{i=1}^{b} \mathbb{1} \left( \delta(x^{(i)}) - \delta(x) \geq 0 \right)$$
Bootstrap example

Alas, it's slightly more complicated.

We didn't draw these samples from a distribution with 0 mean; we created them from the original test set $x$, which happens to be biased (by .20) in favor of $A$.

So to measure how surprising is our observed $\delta(x)$, we actually compute the p-value by counting how often $\delta(x')$ exceeds the expected value of $\delta(x)$ by $\delta(x)$ or more:

$$p\text{-value}(x) = \sum_{i=1}^{b} \mathbb{1} \left( \delta(x^{(i)}) - \delta(x) \geq \delta(x) \right)$$

$$= \sum_{i=1}^{b} \mathbb{1} \left( \delta(x^{(i)}) \geq 2\delta(x) \right)$$
Bootstrap example

Suppose:
- We have 10,000 test sets $x(i)$ and a threshold of .01
- And in only 47 of the test sets do we find that $\delta(x(i)) \geq 2\delta(x)$
- The resulting p-value is .0047
- This is smaller than .01, indicating $\delta(x)$ is indeed sufficiently surprising
- And we reject the null hypothesis and conclude $A$ is better than $B$. 
**Paired bootstrap example**

*After Berg-Kirkpatrick et al (2012)*

```plaintext
function BOOTSTRAP(test set $x$, num of samples $b$) returns $p$-value$(x)$

Calculate $\delta(x)$  # how much better does algorithm A do than B on $x$

$s = 0$

for $i = 1$ to $b$ do

  for $j = 1$ to $n$ do  # Draw a bootstrap sample $x^{(i)}$ of size $n$

    Select a member of $x$ at random and add it to $x^{(i)}$

    Calculate $\delta(x^{(i)})$  # how much better does algorithm A do than B on $x^{(i)}$

  $s \leftarrow s + 1$ if $\delta(x^{(i)}) > 2\delta(x)$

$p$-value$(x) \approx \frac{s}{b}$  # on what % of the $b$ samples did algorithm A beat expectations?

return $p$-value$(x)$  # if very few did, our observed $\delta$ is probably not accidental
```
Text Classification and Naive Bayes

The Paired Bootstrap Test
Avoiding Harms in Classification
Harms in sentiment classifiers

Kiritchenko and Mohammad (2018) found that most sentiment classifiers assign lower sentiment and more negative emotion to sentences with African American names in them.

This perpetuates negative stereotypes that associate African Americans with negative emotions.
Harms in toxicity classification

Toxicity detection is the task of detecting hate speech, abuse, harassment, or other kinds of toxic language. But some toxicity classifiers incorrectly flag as being toxic sentences that are non-toxic but simply mention identities like blind people, women, or gay people. This could lead to censorship of discussion about these groups.
What causes these harms?

Can be caused by:

- Problems in the training data; machine learning systems are known to amplify the biases in their training data.
- Problems in the human labels
- Problems in the resources used (like lexicons)
- Problems in model architecture (like what the model is trained to optimized)

Mitigation of these harms is an open research area

Meanwhile: **model cards**
Model Cards

(Mitchell et al., 2019)

For each algorithm you release, document:

- training algorithms and parameters
- training data sources, motivation, and preprocessing
- evaluation data sources, motivation, and preprocessing
- intended use and users
- model performance across different demographic or other groups and environmental situations
Text Classification and Naive Bayes

Avoiding Harms in Classification