

Text Classification with Naive Bayes

CS 485, Fall 2024

Applications of Natural Language Processing

Brendan O'Connor

College of Information and Computer Sciences
University of Massachusetts Amherst

upcoming

- .

roadmap

- Introduce text classification
- Method #1: Manually-defined rules and keywords
- Method #2: Supervised learning
 - Naive Bayes model
 - next week: logistic regression model

text classification

- input: some text **x** (e.g., sentence, document)
- output: a label **y** (from a finite, smallish, label set)
- goal: learn a mapping function f from **x** to **y**

text classification

- input: some text \mathbf{x} (e.g., sentence, document)
- output: a label \mathbf{y} (from a finite label set)
- goal: learn a mapping function f from \mathbf{x} to \mathbf{y}

fyi: basically every NLP problem reduces to learning a mapping function with various definitions of \mathbf{x} and \mathbf{y} !

problem

x

y

sentiment analysis

text from reviews (e.g.,
IMDB)

{positive, negative}

topic identification

documents

{sports, news, health, ...}

author identification

books

{Tolkien, Shakespeare,
...}

spam identification

emails

{spam, not spam}

... many more!

input \mathbf{x} :

From European Union <info@eu.org> ☆
Subject
Reply to [REDACTED] ☆

Please confirm to us that you are the owner of this very email address with your copy of identity card as proof.

YOU EMAIL ID HAS WON \$10,000,000.00 ON THE ONGOING EUROPEAN UNION COMPENSATION FOR SCAM VICTIMS. CONTACT OUR EMAIL:
CONTACT US NOW VIA EMAIL: [REDACTED] NOW TO CLAIM YOUR COMPENSATION

label \mathbf{y} : **spam** or **not spam**

we'd like to learn a mapping f such that
 $f(\mathbf{x}) = \mathbf{spam}$

Demo: Keyword count classifier

- Let's consider this task:
sentiment classification of movie reviews
- *Can manually defined* keyword lists be a useful indicator of text sentiment?
 - For each category, define set of words
 - Predict a category if many of its words are used
- Let's try manually defined keywords!
 - Sending link on Piazza/email

f can be hand-designed rules

- if “won \$10,000,000” in \mathbf{x} , $\mathbf{y} = \mathbf{spam}$
- if “CS485” in \mathbf{x} , $\mathbf{y} = \mathbf{not\ spam}$

what are the drawbacks of this method?

f can be learned from data

- given **training data** (already-labeled **\mathbf{x}, \mathbf{y}** pairs)
learn f by maximizing the likelihood of the training data
- this is known as **supervised learning**

training data:

x (email text)

y (spam or not spam)

learn how to fly in 2 minutes

spam

send me your bank info

spam

CS585 Gradescope consent poll

not spam

click here for trillions of \$\$\$

spam

... ideally many more examples!

heldout data:

x (email text)

y (spam or not spam)

CS485 important update

not spam

ancient unicorns speaking english!!!

spam

training data:

x (email text)

y (spam or not spam)

learn how to fly in 2 minutes

spam

send me your bank info

spam

CS585 Gradescope consent poll

not spam

click here for trillions of \$\$\$

spam

... ideally many more examples!

heldout data:

x (email text)

y (spam or not spam)

CS485 important update

not spam

ancient unicorns speaking english!!!

spam

learn mapping function on training data,
measure its accuracy on heldout data

probability review

- random variable X takes value x with probability $p(X = x)$; shorthand $p(x)$
- joint probability: $p(X = x, Y = y)$
- conditional probability: $p(X = x \mid Y = y)$
$$= \frac{p(X = x, Y = y)}{p(Y = y)}$$
- when does $p(X = x, Y = y) = p(X = x) \cdot p(Y = y)$?

probability of some input text

- goal: assign a probability to a sentence
 - sentence: sequence of *tokens*
 $p(w_1, w_2, w_3, \dots, w_n)$
 - $w_i \in V$ where V is the vocabulary (*types*)
- some constraints:

non-negativity for any $w \in V$, $p(w) \geq 0$

probability distribution, sums to 1 $\sum_{w \in V} p(w) = 1$

toy sentiment example

- vocabulary V : {i, hate, love, the, movie, actor}
- training data (movie reviews):
 - i hate the movie
 - i love the movie
 - i hate the actor
 - the movie i love
 - i love love love love love the movie
 - hate movie
 - i hate the actor i love the movie

labels:
positive
negative

bag-of-words representation

i hate the actor i love the movie

bag-of-words representation

i hate the actor i love the movie

word	count
i	2
hate	1
love	1
the	2
movie	1
actor	1

bag-of-words representation

i hate the actor i love the movie

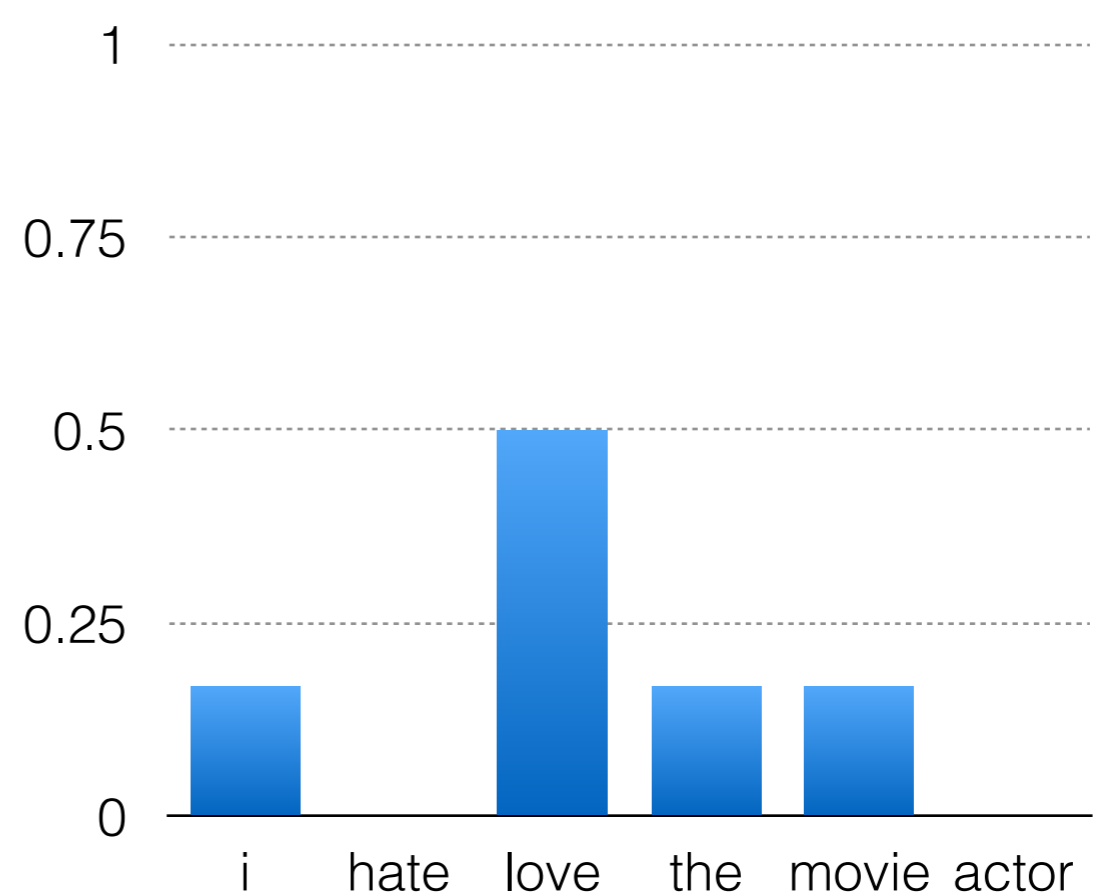
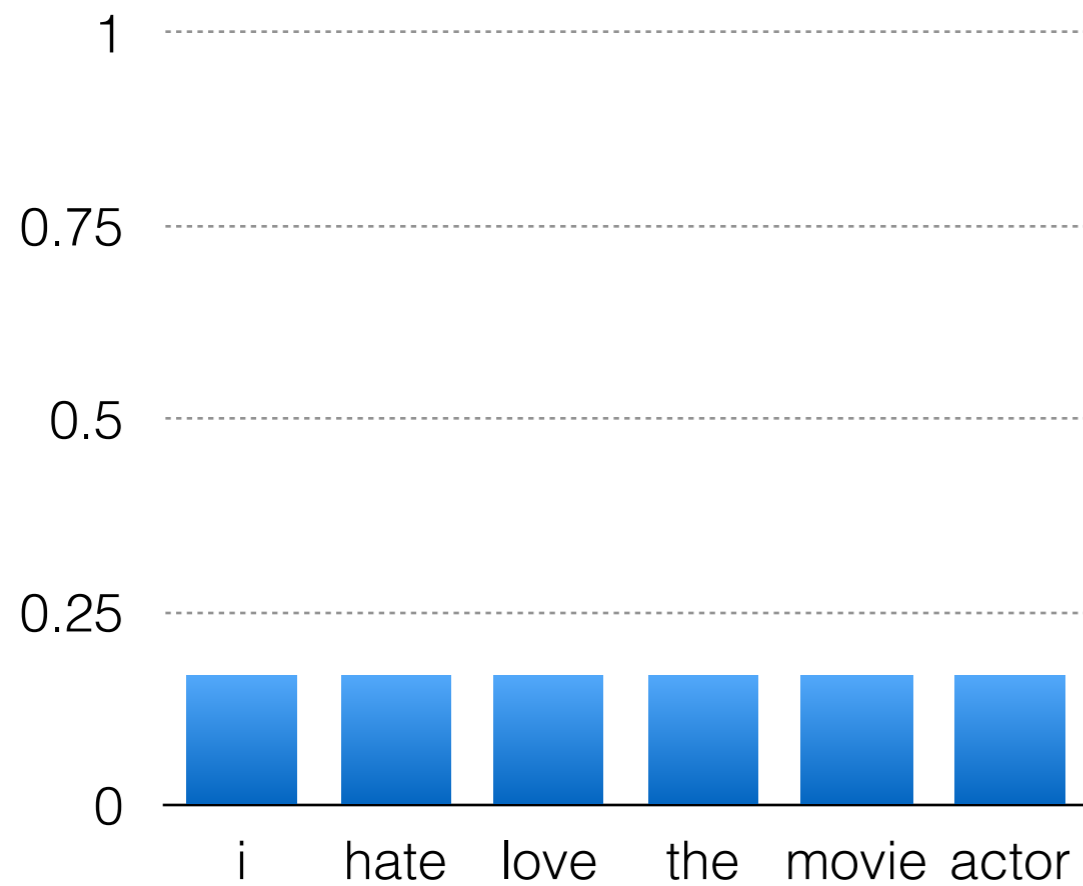
word	count
i	2
hate	1
love	1
the	2
movie	1
actor	1

equivalent representation to:
actor i i the the love movie hate

naive Bayes

- assumption: each word is independent of all other words, conditional on document label
- given labeled data, we can use naive Bayes to estimate probabilities for unlabeled data
- goal: infer probability distribution that generated the labeled data for each label

which of the below word distributions looks like one found in **positive reviews**?



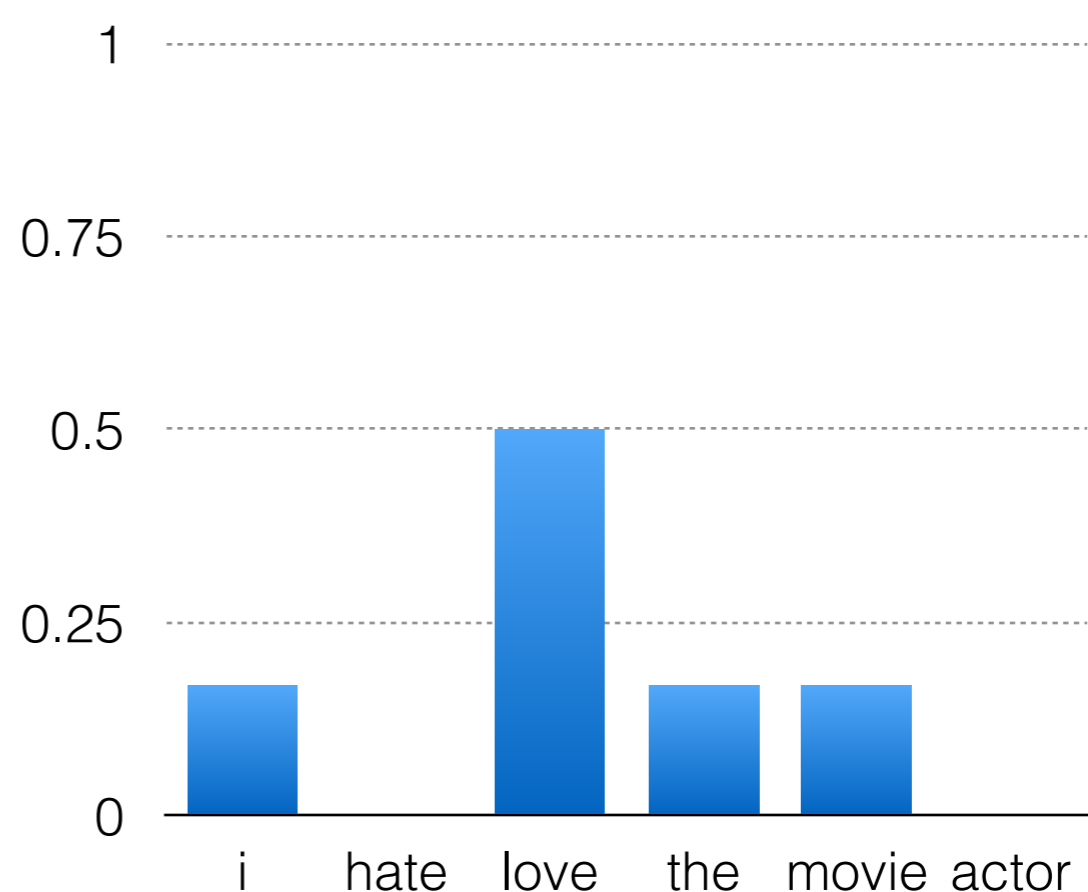
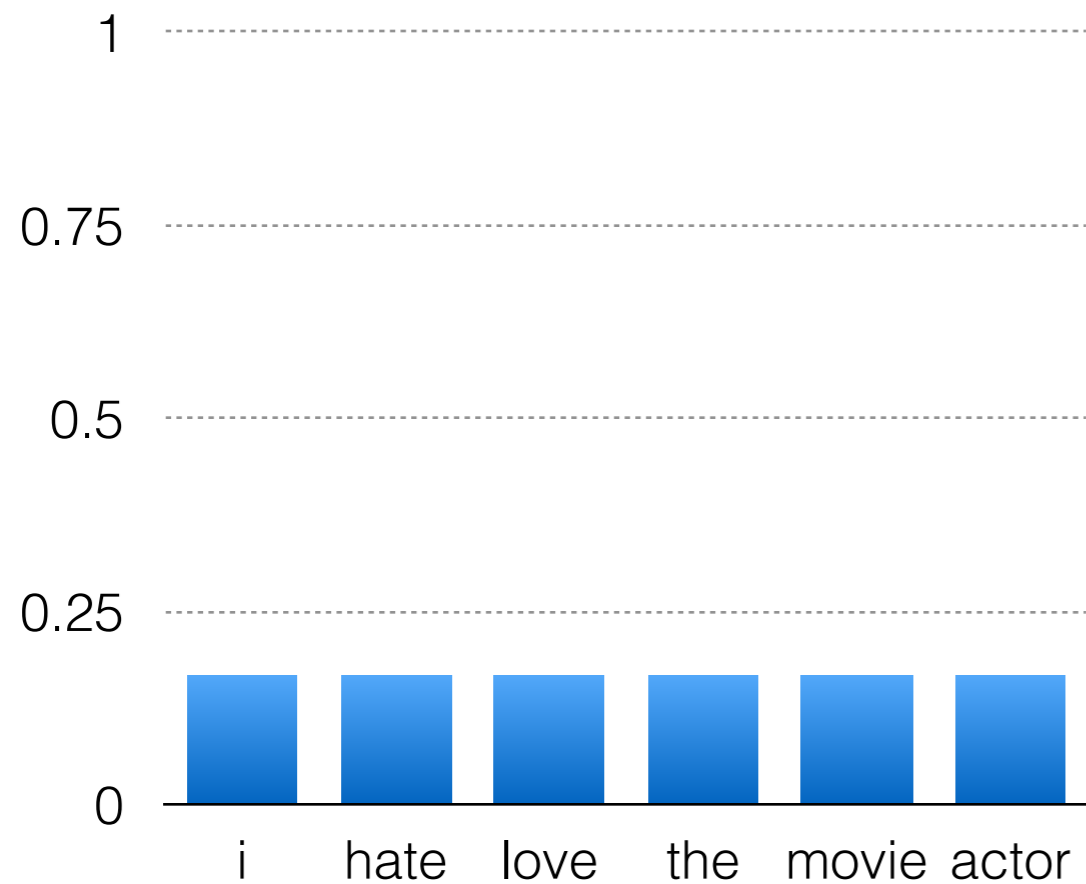
... back to our reviews

$p(\text{i love love love love love the movie})$

$$= p(\text{i}) \cdot p(\text{love})^5 \cdot p(\text{the}) \cdot p(\text{movie})$$

$$= 5.95374181e-7$$

$$= 1.4467592e-4$$



logs to avoid underflow

$$p(w_1) \cdot p(w_2) \cdot p(w_3) \dots \cdot p(w_n)$$

can get really small esp. with large n

$$\log \prod p(w_i) = \sum \log p(w_i)$$

$$p(i) \cdot p(\text{love})^5 \cdot p(\text{the}) \cdot p(\text{movie}) = 5.95374181e-7$$

$$\log p(i) + 5 \log p(\text{love}) + \log p(\text{the}) + \log p(\text{movie})$$

$$= -14.3340757538$$

[This implementation trick is very common in ML and NLP]

class conditional probabilities

Bayes rule (ex: x = sentence, y = label in {pos, neg})

$$p(y | x) = \frac{p(y) \cdot P(x | y)}{p(x)}$$

our predicted label is the one with the highest posterior probability, i.e.,

class conditional probabilities

Bayes rule (ex: x = sentence, y = label in {pos, neg})

$$\text{posterior } p(y | x) = \frac{\text{prior } p(y) \cdot \text{likelihood } P(x | y)}{p(x)}$$

our predicted label is the one with the highest posterior probability, i.e.,

$$\hat{y} = \arg \max_{y \in Y} p(y) \cdot P(x | y)$$

what happened to the denominator???

argmax notation

computing the prior...

- i hate the movie
- i love the movie
- i hate the actor
- the movie i love
- i love love love love love the movie
- hate movie
- i hate the actor i love the movie

$p(y)$ lets us encode inductive bias about the labels
we can estimate it from the data by simply counting...

label y	count	$p(Y=y)$	$\log(p(Y=y))$
POS	3	0.43	-0.84
NEG	4	0.57	-0.56

computing the likelihood...

$$p(X | y=\text{POS})$$

word	count	$p(w y)$
i	3	0.19
hate	0	0.00
love	7	0.44
the	3	0.19
movie	3	0.19
actor	0	0.00
total	16	

$$p(X | y=\text{NEG})$$

word	count	$p(w y)$
i	4	0.22
hate	4	0.22
love	1	0.06
the	4	0.22
movie	3	0.17
actor	2	0.11
total	18	

$p(X | y=\text{POS})$

word	count	$p(w y)$
i	3	0.19
hate	0	0.00
love	7	0.44
the	3	0.19
movie	3	0.19
actor	0	0.00
total	16	

$p(X | y=\text{NEG})$

word	count	$p(w y)$
i	4	0.22
hate	4	0.22
love	1	0.06
the	4	0.22
movie	3	0.17
actor	2	0.11
total	18	

new review X_{new} : love love the movie

$$\log p(X_{\text{new}} | \text{POS}) = \sum_{w \in X_{\text{new}}} \log p(w | \text{POS}) = -4.96$$

$$\log p(X_{\text{new}} | \text{NEG}) = -8.91$$

posterior probs for X_{new}

$$\begin{aligned}\log p(\text{POS} | X_{\text{new}}) &\propto \log P(\text{POS}) + \log p(X_{\text{new}} | \text{POS}) \\ &= -0.84 - 4.96 = -5.80\end{aligned}$$

$$\log p(\text{NEG} | X_{\text{new}}) \propto -0.56 - 8.91 = -9.47$$

What does NB predict?

Naive Bayes

- Assumptions
- Steps to use
 - 1. Training: learn $p(y)$ and $p(w|y)$ parameters for all classes and words, based on their counts in labeled training data
 - 2. Prediction: given learned parameters, for new doc, use Bayes Rule to predict posterior probability of class labels

what if we see no positive training documents containing the word “awesome”?

$$p(\text{awesome} \mid \text{POS}) = 0$$

Add- α (pseudocount) smoothing

$$\text{unsmoothed } P(w_i | y) = \frac{\text{count}(w_i, y)}{\sum_{w \in V} \text{count}(w, y)}$$

$$\text{smoothed } P(w_i | y) = \frac{\text{count}(w_i, y) + \alpha}{\sum_{w \in V} \text{count}(w, y) + \alpha |V|}$$

what happens if we do
add- α smoothing as α increases?

Example: Training

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

Example: Prediction

Model Parameters

New doc $x =$

$P(+)$ =

$P(-)$ =

w	$P(w +)$	$P(w -)$
I	0.1	0.2
love	0.1	0.001
this	0.01	0.01
fun	0.05	0.005
film	0.1	0.1
...

Other details

- Binarization
 - Issue: overcounting word repetitions
 - Solution:

- Negation handling
 - Issue:
 - Solution: heuristic

Evaluation

- Must assess accuracy on held-out data.
 - Train/test split
 - (Alternative: cross-validation)
- Must tune hyperparameters (e.g. pseudocount) on a "development" or "tuning" set.
 - Train/dev/test split

