Text Classification and Naive Bayes

The Naive Bayes Classifier

Multinomial Naive Bayes Classifier

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} \underbrace{P(c_j)}_{\text{prior}} \prod_{x \in X} \underbrace{P(x \mid c)}_{\text{likelihood}}$$

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 $P(x_i|c_j)$ are independent given the class *c.*

$$P(x_1, \ldots, x_n \mid c) = P(x_1 \mid c) \bullet P(x_2 \mid c) \bullet P(x_3 \mid c) \bullet \ldots \bullet P(x_n \mid c)$$

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**

Slide from Chris Manning

Text Classification and Naïve Bayes

Precision, Recall, and F-measure

Evaluation

Consider a binary text classification task:

Is this passage from a book a "smell experience" or not?

Towards Olfactory Information Extraction from Text: A Case Study on Detecting Smell Experiences in Novels

Ryan Brate and **Paul Groth**
University of Amsterdam
Amsterdam, the Netherlands
r.brate@gmail.com
p.t.groth@uva.nl

Marieke van Erp
KNAW Humanities Cluster
Digital Humanities Lab
Amsterdam, the Netherlands
marieke.van.erp@dh.huc.knaw.nl

Abstract

Environmental factors determine the smells we perceive, but societal factors factors shape the importance, sentiment and biases we give to them. Descriptions of smells in text, or as we call them 'smell experiences', offer a window into these factors, but they must first be identified. To the best of our knowledge, no tool exists to extract references to smell experiences from text. In

Evaluation

Consider a binary text classification task:

Is this passage from a book a "smell experience" or not?

You build a "smell" detector

- Positive class: paragraph that involves a smell experience
- Negative class: all other paragraphs

The 2-by-2 confusion matrix

Truth = gold standard

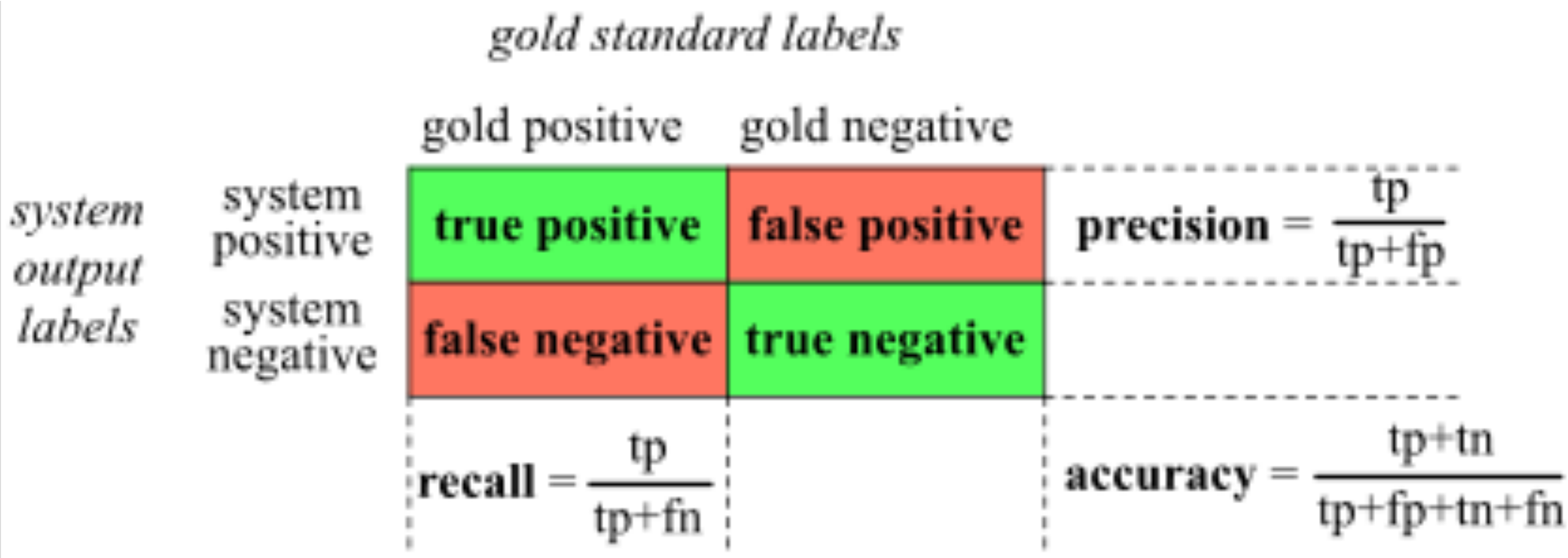
System output = Prediction

	gold positive	gold negative
	true positive	false positive
	false negative	true negative

Precision: $\frac{TP}{TP+FP}$

Recall: $\frac{TP}{TP+FN}$

The 2-by-2 confusion matrix



Evaluation: Accuracy

Why don't we use **accuracy** as our metric?

Imagine we saw 1 million paragraphs

- 100 of them mention smells
- 999,900 talk about something else

We could build a classifier that labels every paragraph "not about smell"

Evaluation: Accuracy

Why don't we use **accuracy** as our metric?

Imagine we saw 1 million paragraphs

- 100 of them mention smells
- 999,900 talk about something else

We could build a classifier that labels every paragraph "not about smell"

- It would get 99.99% accuracy!!!
- But the whole point of the classifier is to help literary scholars find passages about smell to study--- so this is useless!
- 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: 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)

$$\textbf{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

Evaluation: Recall

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

$$\text{RECALL} = \frac{\text{True Positives}}{\text{True positives} + \text{False Negatives}}$$

Evaluation: Recall

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

$$\textbf{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

Why Precision and recall

Our no-smells classifier

- Labels nothing as "about smell"

Accuracy = 99.99%

Recall = $\frac{0}{100} = 0\%$

Precision = $\frac{0}{0}$

DivisionByZeroError

Why Precision and recall

Our no-smells classifier

- Labels nothing as "about smell"

Accuracy=99.99%

Precision = undefined (division by 0!)

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:

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

Typically, $\beta = 1$ (even balance)

$$F_1 = \frac{2PR}{P+R}$$

$FP = 100$

$TP = 100$

$$\text{Precision} = \frac{TP}{TP+FP} = \frac{100}{200}$$

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{100}{100}$$

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., $\beta = 1$)

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., $\beta = 1$)

$$\mathrm{F}_1 = \frac{2PR}{P+R}$$

Text Classification and Naive Bayes

Evaluation with more than two classes

Confusion Matrix for 3-class classification

= true positives

= FN (normal)

= FN (urgent)

= FN (spam)

gold labels

system output		urgent	normal	spam	
	urgent	8	10	1	**precisionu**= $\frac{8}{8+10+1}$
	normal	5	60	50	**precisionn**= $\frac{60}{5+60+50}$
	spam	3	30	200	**precisions**= $\frac{200}{3+30+200}$
		recallu = $\frac{8}{8+5+3}$	**recalln** = $\frac{60}{10+60+30}$	**recalls** = $\frac{200}{1+50+200}$	

How to combine P/R from 3 classes to get one metric

Macroaveraging:

Compute performance for each class

Average over classes

Microaveraging:

Collect decisions for all classes into one confusion matrix

Compute precision & recall from that table

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

Class 1: Urgent

	true urgent	true not
system urgent	8	11
system not	8	340

$$\text{precision} = \frac{8}{8+11} = .42$$

Class 2: Normal

	true normal	true not
system normal	60	55
system not	40	212

$$\text{precision} = \frac{60}{60+55} = .52$$

Class 3: Spam

	true spam	true not
system spam	200	33
system not	51	83

$$\text{precision} = \frac{200}{200+33} = .86$$

Pooled

	true yes	true no
system yes	268	99
system no	99	635

$$\begin{array}{c}\text{microaverage}\\ \text{precision}\end{array} = \frac{268}{268+99} = \mathbf{.73}$$

$$\begin{array}{c}\text{macroaverage}\\ \text{precision}\end{array} = \frac{.42+.52+.86}{3} = \mathbf{.60}$$

Macroaveraging and Microaveraging

Class 1: Urgent

	true urgent	true not
system urgent	8	11
system not	8	340

$$\text{precision} = \frac{8}{8+11} = .42$$

Class 2: Normal

	true normal	true not
system normal	60	55
system not	40	212

$$\text{precision} = \frac{60}{60+55} = .52$$

Class 3: Spam

	true spam	true not
system spam	200	33
system not	51	83

$$\text{precision} = \frac{200}{200+33} = .86$$

Pooled

	true yes	true no
system yes	268	99
system no	99	635

$$\text{microaverage precision} = \frac{268}{268+99} = \mathbf{.73}$$

$$\text{macroaverage precision} = \frac{.42+.52+.86}{3} = \mathbf{.60}$$

Text Classification and Naive Bayes

Statistical Significance Testing

How can we be sure that our results *generalize*?

Usually:

We care about how our system performs on data that is *similar* to the training data- not identical.

Development Test Sets and Cross-validation

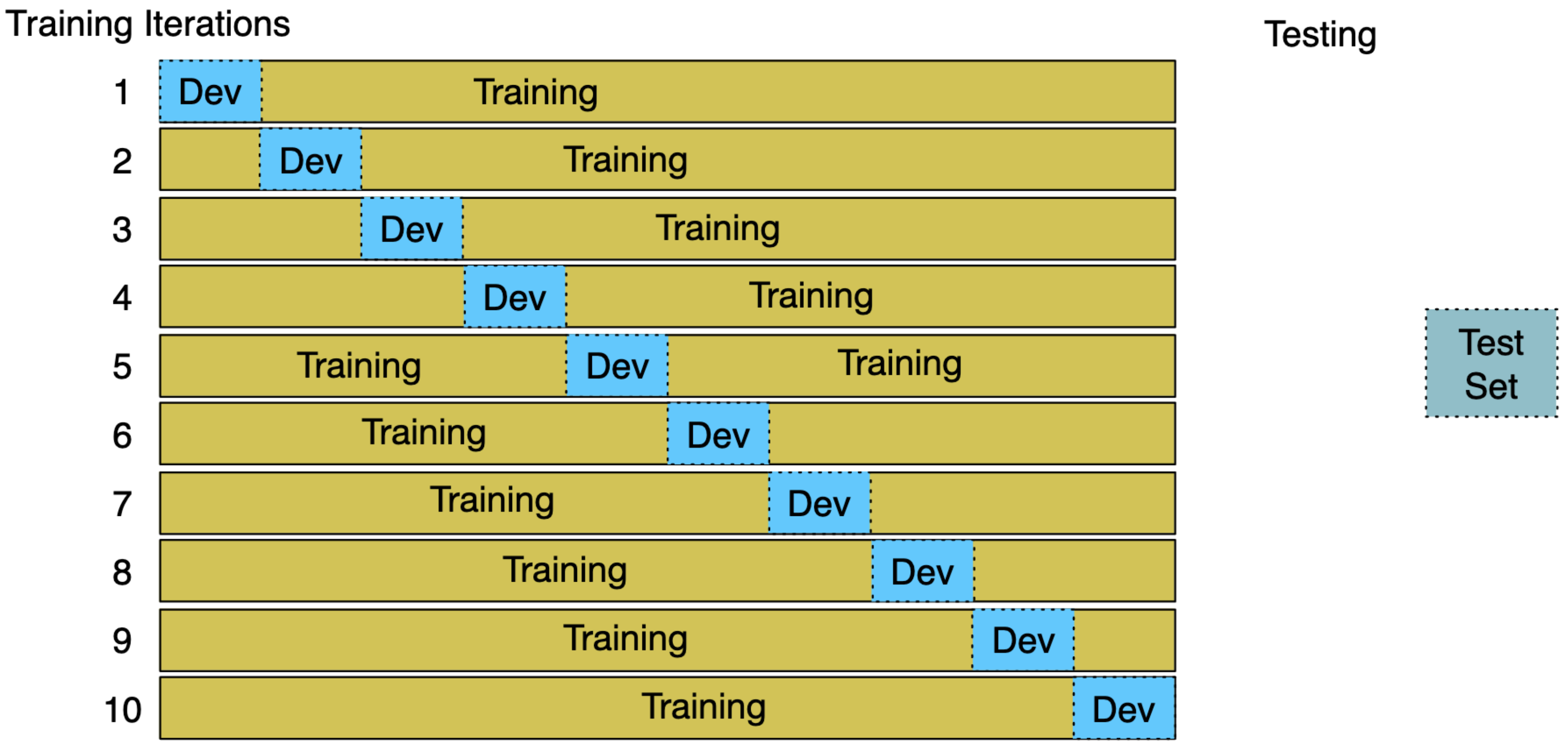
Training set | Development Test Set | Test Set

Train on training set, tune on devset, report on testset

- This avoids overfitting (‘training on test’)
- 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



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 testset *x*
- δ(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

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 testset *x*
- δ(x): the performance difference between A, B on x:
  - δ(x) = M(A,x) – M(B,x)

- We want to know if δ(x)>0, meaning A is better than B
- δ(x) is called the **effect size**
- Suppose we look and see that δ(x) is positive. Are we done?

Statistical Hypothesis Testing

Consider two hypotheses:

- Null hypothesis: A isn't better than B
- A is better than B

We want to rule out H_0

Null Hypothesis

$$H_0 : \delta(x) \leq 0$$

$$H_1 : \delta(x) > 0$$

Statistical Hypothesis Testing

Consider two hypotheses:

- Null hypothesis: A isn't better than B
- A is better than B

$$H_0 \;:\; \delta(x) \leq 0$$
$$H_1 \;:\; \delta(x) > 0$$

We want to rule out H_0

We create a random variable X ranging over test sets and ask, *among all these test sets, how likely are we to see $\delta(x)$ if H_0 is true?*

Statistical Hypothesis Testing

Consider two hypotheses:

- Null hypothesis: A isn't better than B
- A is better than B

$$H_0 \ : \ \delta(x) \leq 0$$
$$H_1 \ : \ \delta(x) > 0$$

We want to rule out H_0

We create a random variable X ranging over test sets and ask, *among all these test sets, how likely are we to see $\delta(x)$ if H_0 is true?*

- 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* is not better than *B)*.
  - If H_0 is true but $\delta(x)$ is huge, that is surprising! Very low probability!
- A small p-value means that the difference we observed is unlikely under the null hypothesis. We fail to find support for the null hypothesis.

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* is not better than *B)*.
  - If H_0 is true but $\delta(x)$ is huge, that is surprising! Very low probability!
- A small p-value means that the difference we observed is unlikely under the null hypothesis. We fail to find support for the null hypothesis.
- Conventionally, very small means $p < 0.05$ or 0.01

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* is not better than *B)*.
  - If H_0 is true but $\delta(x)$ is huge, that is surprising! Very low probability!
- A small p-value means that the difference we observed is unlikely under the null hypothesis. We fail to find support for the null hypothesis.
- Conventionally, very small means $p < 0.05$ or 0.01
- A result(e.g., "*A* is better than *B*") is **statistically significant** if the δ we saw has a probability that is below the threshold and we therefore reject this null hypothesis.

Statistical Hypothesis Testing

- 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'.

Statistical Hypothesis Testing

- 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

The Paired Bootstrap Test

Bootstrap test

Efron and Tibshirani, 1993

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

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.

Here are the results of systems A and B on x.

There are 4 outcomes (A & B both right, A & B both wrong, A right/B wrong, A wrong/B right):

1	2	3	4	5	6	7	8	9	10	A%	B%	d()
AB	A**B**	AB	**A**B	A**B**	**A**B	A**B**	AB	**AB**	A**B**	0.7	0.5	0.2

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:

1	2	3	4	5	6	7	8	9	10	A%	B%	d()
AB	A**B**	AB	**A**B	A**B**	**A**B	A**B**	AB	**AB**	A**B**	0.7	0.5	0.2
A**B**	AB	A**B**	**A**B	**A**B	A**B**	**A**B	AB	**AB**	AB	0.6	0.6	0.0
A**B**	AB	**AB**	**A**B	**A**B	AB	**A**B	A**B**	AB	AB	0.6	0.7	-0.1

Bootstrap example

We have a distribution! We check how often A has an **accidental** advantage, to see if the original δ(x) we saw was very common. If H_0 is true, we expect δ(x')=0.

Bootstrap example

We have a distribution! We check how often A has an **accidental** advantage, to see if the original δ(x) we saw was very common. If H_0 is true, we expect δ(x')=0.

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

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

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. What's the issue?

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 is biased (by .20) in favor of A.

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

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 is biased (by .20) in favor of *A*.

To measure how surprising our observed δ(*x*) is, we compute the p-value by counting how often δ(*x*′) exceeds the expected value of δ(*x*) by δ(*x*) or more:

$$
\begin{aligned}
\text{p-value}(x) &= \frac{1}{b}\sum_{i=1}^{b} \mathbb{1}\left(\boldsymbol{\delta}(x^{(i)}) - \boldsymbol{\delta}(x) \geq \boldsymbol{\delta}(x)\right) \\
&= \frac{1}{b}\sum_{i=1}^{b} \mathbb{1}\left(\boldsymbol{\delta}(x^{(i)}) \geq 2\boldsymbol{\delta}(x)\right)
\end{aligned}
$$

Bootstrap example

Suppose:

- We have 10,000 test sets $x(i)$ and a threshold of .01
- In 47 of the test sets we find that $\delta(x(i)) \geq 2\delta(x)$

Bootstrap example

Suppose:

- We have 10,000 test sets $x(i)$ and a threshold of .01
- In 47 of the test sets we find that $\delta(x(i)) \geq 2\delta(x)$
- The resulting p-value is .0047

Bootstrap example

Suppose:

- We have 10,000 test sets $x(i)$ and a threshold of .01
- In 47 of the test sets 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

Bootstrap example

Suppose:

- We have 10,000 test sets $x(i)$ and a threshold of .01
- In 47 of the test sets 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
- We reject the null hypothesis and conclude *A* is better than *B*.

Text Classification and Naive Bayes

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