CS 333: Natural Language Processing

Fall 2025

Prof. Carolyn Anderson
Wellesley College

Reminders

- HW 4 will be released today
- Quiz 4 next Tuesday: J&M Chapter 5
- ~~My next help hours: Monday 4-5:30~~ SHIFTED!
- My next help hours: Sunday 5-6

HW 4

- Goal: build a Naive Bayes classifier to identify book genres using a dataset from Goodreads
- Last homework before Midterm 1!

HW 2 Curiosity Points

- Explorations of segmentation techniques for other languages
- Detailed analysis of errors
- Improvements to make the segmenter more robust
- Read research papers on other segmentation approaches
- Tested our programs on another corpus (emails I sent to the class)

Multinomial Naive Bayes

Simplifying Assumptions:
- Bag-of-words
- Conditional independence of likelihoods (features are independent of each other)

positions ← all word positions in test document

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

likelihoods

$P(c_j)$: class

$P(x_i \mid c_j)$: Features

$$P(c \mid x) = P(c) \pi P(x_i \mid c)$$

Calculating Priors

Calculate *priors:*

- For each c_j in C:

$docs_j$ = n docs in class c

$$p(c_j) = \frac{|docs_j|}{|\text{total \# documents}|}$$

Calculating Likelihoods

Calculate *likelihoods:*

- Text_j = single doc containing all docs_j
- For each word w_k in V:

n_k = # of w_k in Text_j

$$p(w_k \mid c_j) = \frac{n_k + \alpha}{n + \alpha \mid Vocabulary \mid}$$

where α = smoothing parameter

often: $\alpha = 1$ for add-1

Naive Bayes: Bigger Picture

Naive Bayes Limitations

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

https://miguelmota.com/blog/naive-bayes-classifier-in-javascript/demo/

Limitation: 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
```

p(like|+)
p(like|-)

p(NOT_like|+)
p(NOT_like|-)

Das, Sanjiv and Mike Chen. 2001. Yahoo! for Amazon: Extracting market sentiment from stock message boards. In Proceedings of the Asia Pacific Finance Association Annual Conference (APFA).

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. EMNLP-2002, 79—86.

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

Other Tasks: Language ID

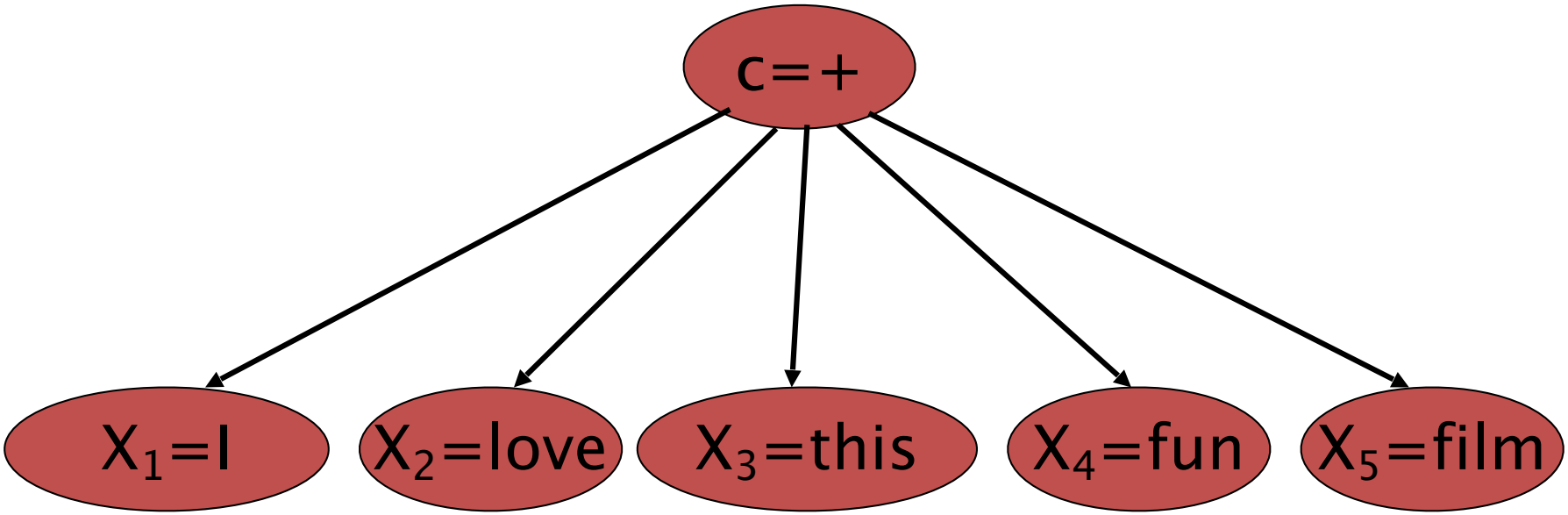
- Determining what language a piece of text is written in.
- Features based on character n-grams do very well
- Important to train on lots of varieties of each language (e.g., American English varieties like African-American English, or English varieties around the world like Indian English)

Naive Bayes Advantages

- Very Fast, low storage requirements
- Work wells 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
- Optimal if the independence assumptions hold
- A good dependable baseline for text classification
  - *But we will see other classifiers that give better accuracy in practice*

Relationship to Language Modeling

Generative Model for Naive Bayes



NB and and Language Modeling

Naive 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 can be viewed as a kind of language model.

Each class = unigram language model

Assigning each word: $P(word|c)$

Assigning each sentence: $P(s|c) = \pi P(word|c)$

class	pos
0.1	I
0.1	love
0.01	this
0.05	fun
0.1	film

I love this fun film

0.1 0.1 0.5 0.01 0.1

$P(s|pos) = 0.000\ 0005$

Each class = unigram language model

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

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

Class *pos*

0.1	I
0.1	love
0.01	this
0.05	fun
0.1	film
…	

Each class = unigram language model

Which class assigns the higher probability to s?

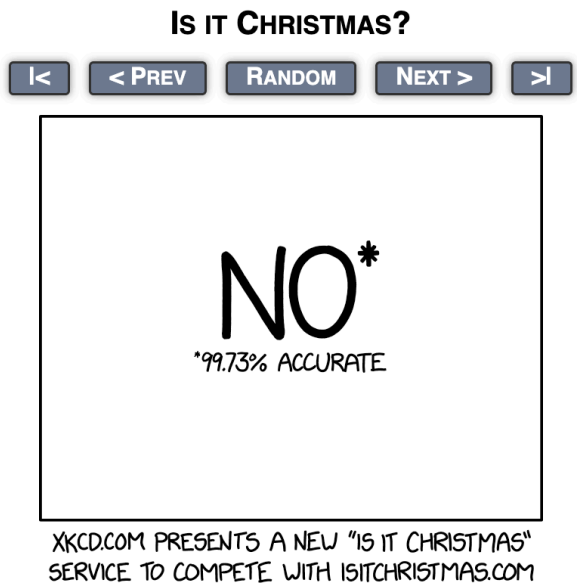
Model pos	
0.1	I
0.1	love
0.01	this
0.05	fun
0.1	film

Model neg	
0.2	I
0.001	love
0.01	this
0.005	fun
0.1	film

I	love	this	fun	film
0.1	0.1	0.01	0.05	0.1
0.2	0.001	0.01	0.005	0.1

$P(s \mid pos) > P(s \mid neg)$

Evaluation



https://xkcd.com/2236/

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 this paper, we present two variations on a semi-supervised approach to identify smell experiences in English literature. The combined set of patterns from both implementations offer significantly better performance than a keyword-based baseline.

Evaluation

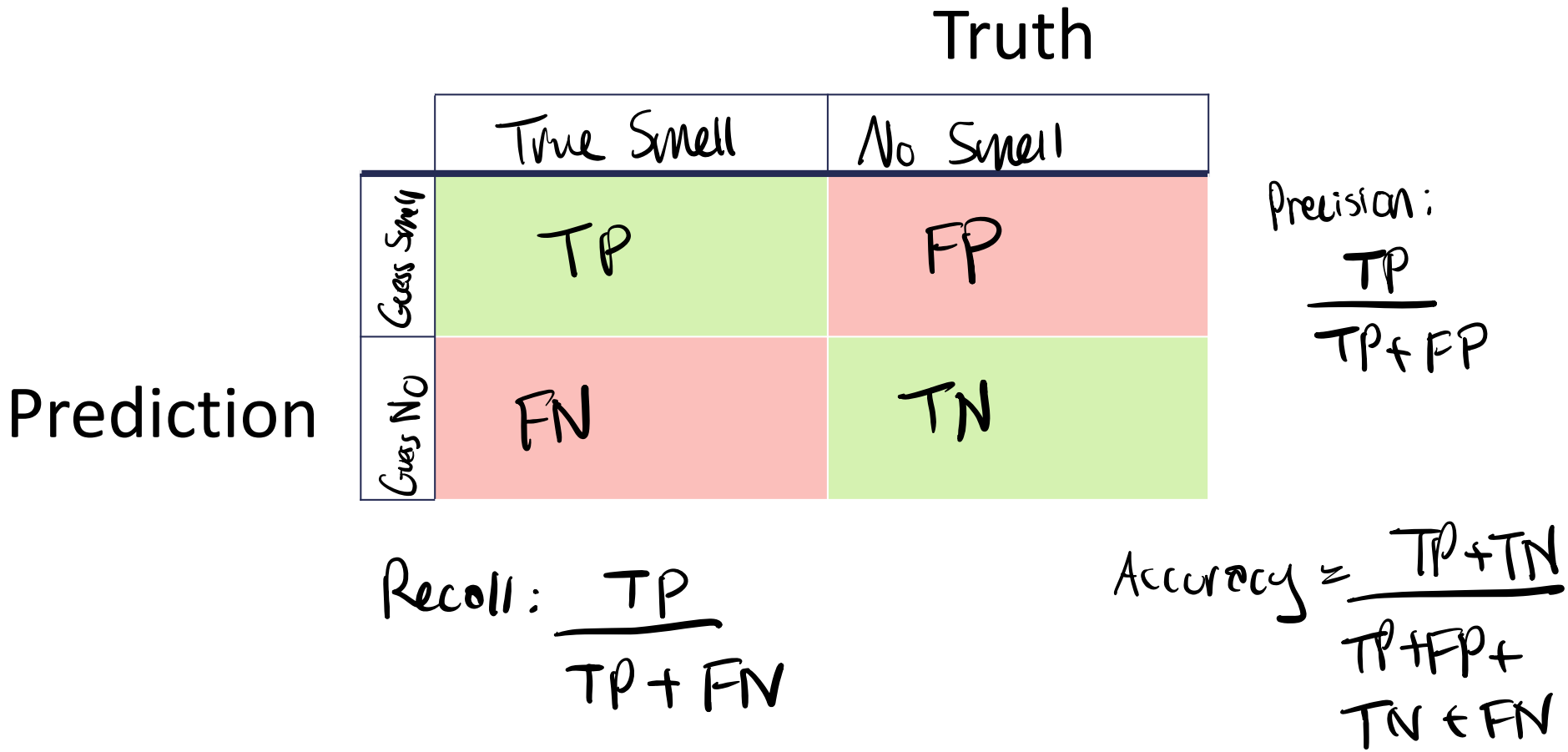
Consider a binary text classification task:

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

Evaluation


Truth
True Smell
No Smell
Guess Smell
Guess No
TP
FP
FN
TN
Prediction
Precision: TP / TP+FP
Recall: TP / TP+FN
Accuracy = TP+TN / TP+FP+TN+FN

Evaluation

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"

Why Not 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)

PRECISION =

Evaluation: Recall

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

RECALL =

Why Precision and Recall

Our no-smells classifier

- Labels nothing as "about smell"

Accuracy = 99.99%

Recall = 0 (no true positives)

Precision = undefined (division by 0)

A Combined Measure

F measure: a single number that combines Precision and Recall:

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

$\beta = 1$ is evenly balanced P & R

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

A Combined Measure

F measure: a single number that combines Precision and Recall:

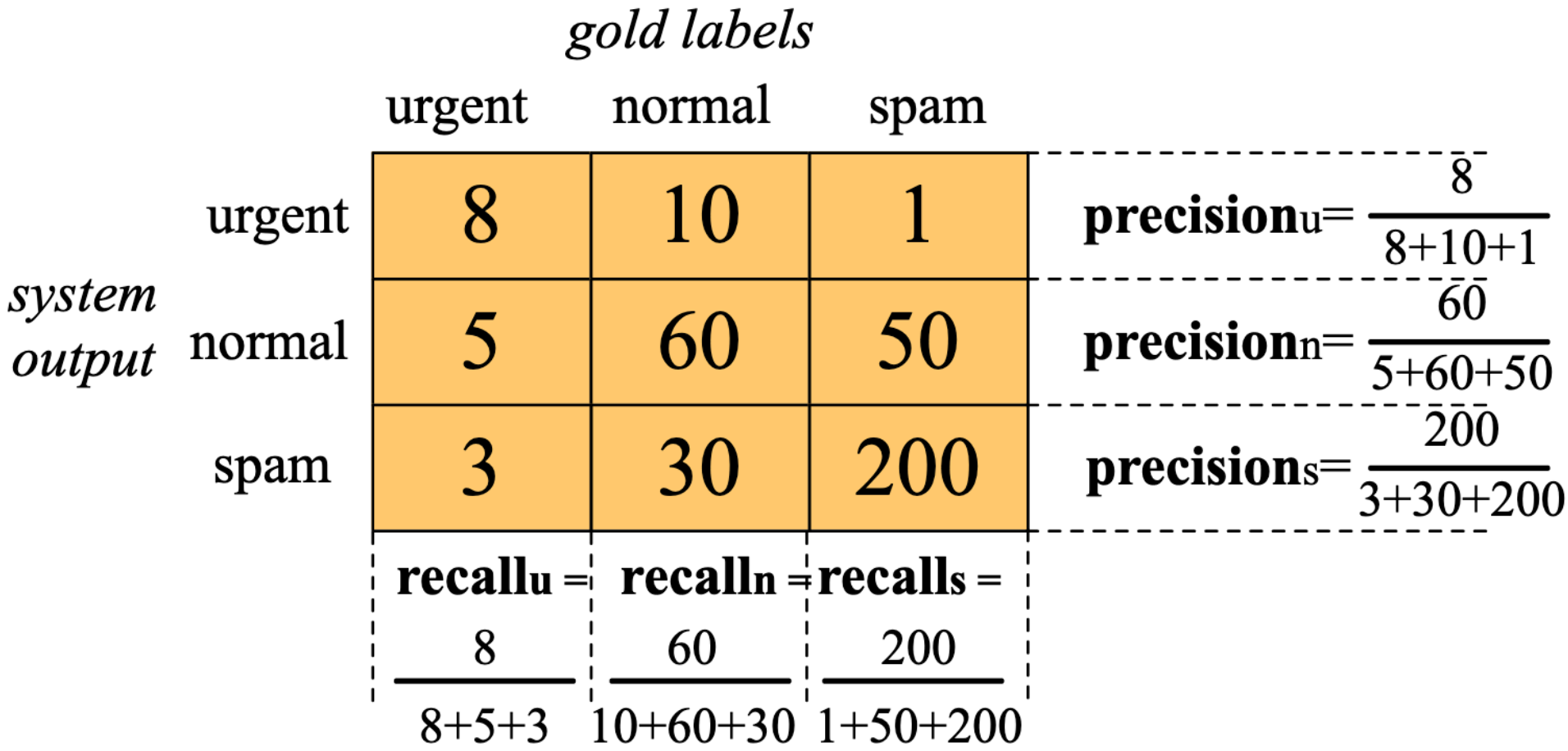
$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

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

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

Evaluation with More Than Two Classes

Confusion Matrix for 3-class classification



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

Macroaveraging:

compute performance metric per class & then average over classes

Microaveraging:

Collect decisions for all classes into one confusion matrix

Compute P & R from the combined 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

Macroaverage : $\frac{0.42 + 0.52 + 0.86}{3}$

$= 0.6$

Microaverage : $\frac{268}{268 + 99} =$

0.73

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

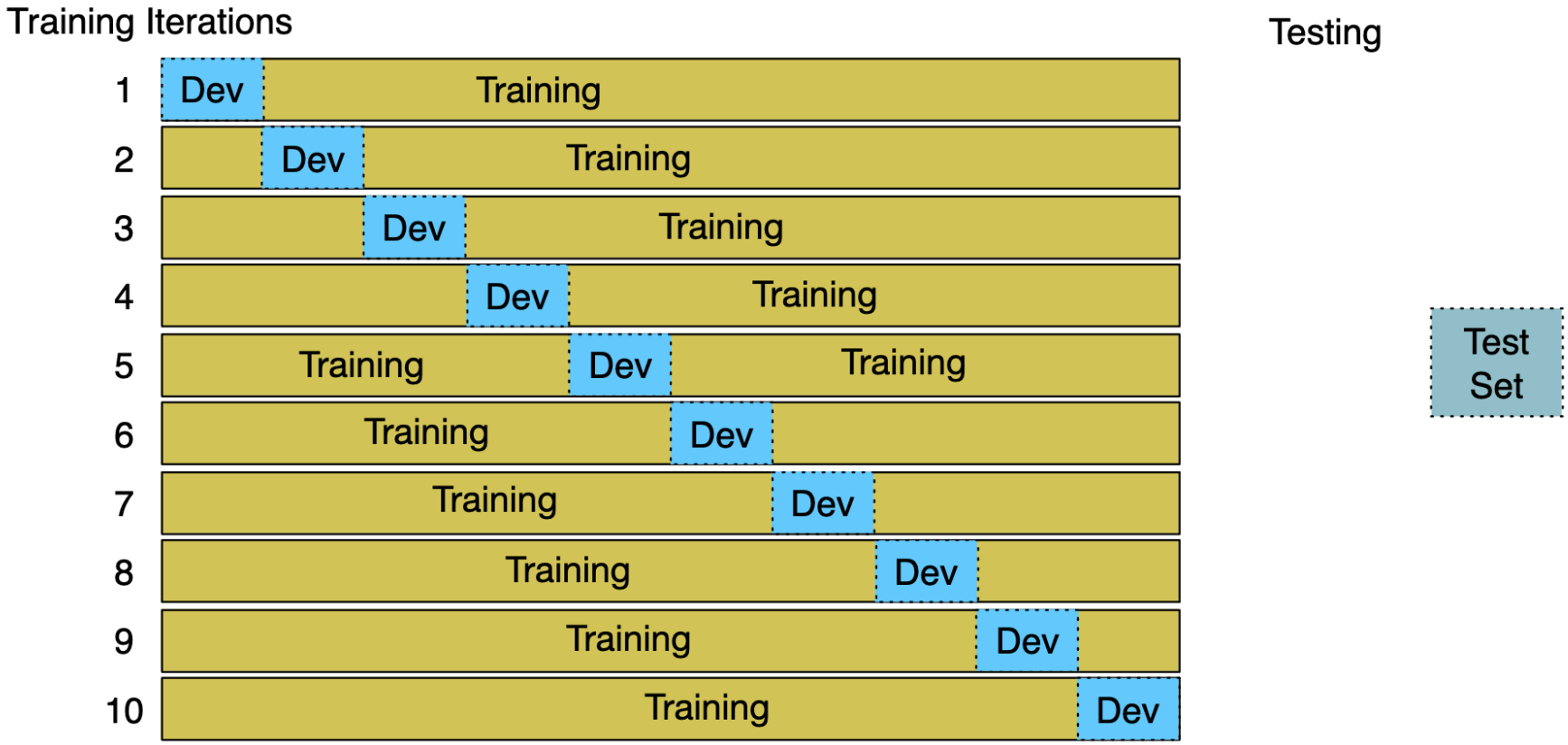
Training set	Development Test Set	Test Set

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 and compute pooled dev performance:



How to Check if Performance Difference is Reliable?

Given:

- Classifier A and B
- Metric M: M(A,x) is the performance of *A* on testset *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

How to Check if Performance Difference is Reliable?

Given:

- Classifier A and B
- Metric M: M(A,x) is the performance of *A* on testset *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?

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

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

Statistical Hypothesis Testing

Consider two hypotheses:

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

We want to rule out H_0

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

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

We want to rule out H_0

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

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

- 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

- 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

- 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 test sets 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 test sets 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 can 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

The Paired Bootstrap Test

Paired 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.

Paired Bootstrap Test

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	AB	AB	AB	AB	AB	AB	AB	AB	AB	0.7	0.5	0.2

Paired Bootstrap Test

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	AB	AB	AB	AB	AB	AB	AB	AB	AB	0.7	0.5	0.2
AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	0.6	0.6	0.0
AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	0.6	0.7	-0.1

Paired Bootstrap Test

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

Paired Bootstrap Test

We have a distribution! We check how often A has an **accidental** advantage, to see if the original $\delta(x)$ we saw was very common. If H_0 is true, 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:

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

Paired Bootstrap Test

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?

Paired Bootstrap Test

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:

Paired Bootstrap Test

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:

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

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

Paired Bootstrap Test

Suppose:

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

Paired Bootstrap Test

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

Paired Bootstrap Test

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

Paired Bootstrap Test

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*.

Avoiding Harms in Classification

Harms in Sentiment Analysis

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 Detection

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.

Harms in Classification

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