CS 1671/2071
Human Language Technologies

Session 9: Text classification, tf-idf, PPMI

Michael Miller Yoder

February 10, 2025

School of Computing and Information

Course logistics: quiz and homework

- Quiz on Canvas due **this Thu Feb 13**
  - I will release it today
  - What readings it covers will be specified in the description on Canvas
- Homework 2 is due **next Thu Feb 20**
  - What you can work on now: look at scikit-learn `LogisticRegression` documentation. Try loading data, extract features (vectorize) the data and train a model from it. It's fine to use code from the clickbait classification exercises in class!
  - I will soon release the **optional** Kaggle competition for extra credit

Course logistics: project

- Next project milestone: project proposal due Feb 28 (stay tuned for more details on that)
- If I emailed your group about choosing different directions or datasets, please respond over email or book office hours to talk through by **this Fri Feb 14**
- Choose a communication chat platform for your group to collaborate (Teams, Discord, Signal, WhatsApp, etc)

Lecture overview: Text classification, tf-idf, PPMI

- Weighting bag-of-words text representations with:
  - Tf-idf
  - PPMI
- Text classification
- Evaluation of text classification
  - Precision, recall, f1-score
  - Train/dev/test and cross-validation sets
- Harms in classification
- Coding activity
  - tf-idf representations for documents
  - Clickbait classification evaluation

Tf-idf weighting

Raw frequency is a bad representation

- The co-occurrence matrices we have seen represent each cell by word frequencies
  - Whether in term-document or term-term matrices
- Frequency is clearly useful; if *sugar* appears a lot near *apricot*, that's useful information.
- But overly frequent words like *the*, *it*, or *they* are not very informative about the context
  - 2 documents that use a lot of *the* are not necessarily similar
- It's a paradox! How can we balance these two conflicting constraints?

Slide adapted from Jurafsky & Martin

Weighting words in term-document and term-term matrices

- **tf-idf** (term frequency-inverse document frequency)
  - For representing documents with their most unique words (for text classification, information retrieval)
  - Term-document matrix
- **PPMI** (positive pointwise mutual information)
  - For finding associations between words (which appear more often together than chance?)
  - Term-term matrix

Slide adapted from Jurafsky & Martin

Term frequency (tf)

$tf_{t,d}$ = count(t,d)

Instead of using raw count, we squash a bit:

$$tf_{t,d} = \begin{cases} 1 + \log count(t,d) & \text{if } count(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Slide adapted from Jurafsky & Martin

Document frequency (df)

df_t is the number of documents term *t* occurs in.

(note this is not collection frequency, which is the total count across all documents)

"Romeo" is very distinctive for one Shakespeare play:

	Collection Frequency	Document Frequency
Romeo	113	1
action	113	31

Slide adapted from Jurafsky & Martin

Inverse document frequency (idf)

$$\text{idf}_t = \log_{10}\left(\frac{N}{\text{df}_t}\right)$$

- N is the total number of documents in the collection
- Documents can be whatever you want! (Full documents, paragraphs, etc)

Word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.246
wit	34	0.037
fool	36	0.012
good	37	0
sweet	37	0

Slide adapted from Jurafsky & Martin

tf-idf Controls for Frequent but Uninformative Words

Some words are very common in a given document because they are common across all documents (e.g., *the*). They are not discriminative. `tf-idf` (product of term frequency and inverse document frequency) addresses this:

$$tf_{t,d} = \begin{cases} 1 + \log count(t,d) & \text{if } count(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\text{idf}_f = \log_{10} \frac{N}{\text{df}_t}$$

$$\text{tf–idf}(t, d) = \text{tf}_{t,d} \cdot \text{idf}_t$$

Slide adapted from Jurafsky & Martin, David Mortensen

Final tf-idf weighted values

Raw counts

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

tf-idf:

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

Positive pointwise mutual information (PPMI)

Problem with Raw Counts

- Raw word frequency is not a great measure of association between words.
- It is very skewed: “the” and “of” are very frequent, but maybe not the most discriminative.
- We would rather have a measure that asks whether a context word is **particularly informative** about the target word.

Positive Pointwise Mutual Information (PPMI)

Slide credit: David Mortensen

Pointwise mutual information

- Do events *x* and *y* co-occur more than if they were independent?

$$\text{PMI}(X,Y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

- PMI between 2 words [Church+Hanks 1989]
  - Do words *x* and *y* co-occur more than if they were independent?

$$PMI(word_1, word_2) = \log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)}$$

Slide adapted from Jurafsky & Martin

- In computational linguistics, PMI has been used for finding collocations and associations between words.

word 1	*word 2*	*count word 1*	*count word 2*	count of co-occurrences	PMI
puerto	rico	1938	1311	1159	10.0349081703
hong	kong	2438	2694	2205	9.72831972408
los	angeles	3501	2808	2791	9.56067615065
carbon	dioxide	4265	1353	1032	9.09852946116
prize	laureate	5131	1676	1210	8.85870710982
san	francisco	5237	2477	1779	8.83305176711
nobel	prize	4098	5131	2498	8.68948811416
ice	hockey	5607	3002	1933	8.6555759741
star	trek	8264	1594	1489	8.63974676575
car	driver	5578	2749	1384	8.41470768304
it	the	283891	3293296	3347	-1.72037278119
are	of	234458	1761436	1019	-2.09254205335
this	the	199882	3293296	1211	-2.38612756961
is	of	565679	1761436	1562	-2.54614706831
and	of	1375396	1761436	2949	-2.79911817902
a	and	984442	1375396	1457	-2.92239510038
in	and	1187652	1375396	1537	-3.05660070757
to	and	1025659	1375396	1286	-3.08825363041
to	in	1025659	1187652	1066	-3.12911348956
of	and	1761436	1375396	1190	-3.70663100173

Slide credit: David Mortensen

Positive Pointwise Mutual Information

- PMI ranges from $-\infty$ to $+\infty$
- But the negative values are problematic:
  - Things are co-occurring less than we expect by chance
  - Unreliable without enormous corpora
    - Imagine w_1 and w_2 whose probability is each 10^{-6}.
    - Hard to be sure $p(w_1, w_2)$ is significantly different than 10^{-12}.
  - Furthermore it's not clear people are good at "unrelatedness".
- So we just replace negative PMI values by 0.

$$\text{PPMI}(w, c) = max\left(\log_2 \frac{p(w, c)}{p(w)p(c)}, 0\right)$$

Slide credit: David Mortensen

Computing PPMI on a Term-Context Matrix

- We have matrix F with V rows (words) and C columns (contexts) (in general $C = V$)
- Each cell contains the co-occurrence count of words w and c, f_{wc}
- Let S be the total sum of all word-context word counts

			C			
	0	0	1	0	1	2
	0	0	1	0	1	
	2	1	0	1	0	
V	1	6	0	4	0	
	3					19

$$PMI(w,c) = \frac{p(w,c)}{p(w)p(c)}$$

$$p(w,c) = \frac{f_{wc}}{S}$$

$$p(w) = \frac{\sum_{c \in C} f_w}{S}$$

$$p(c) = \frac{\sum_{w \in W} f_c}{S}$$

$$PPMI(w,c) = \max(PMI(w,c), 0)$$

Slide adapted from David Mortensen

Worked Example: Computing PPMI from Term-Context Matrix (Part I)

	computer	data	pinch	result	sugar	
apricot	0	0	1	0	1	2
pineapple	0	0	1	0	1	2
digital	2	1	0	1	0	4
information	1	6	0	4	0	11
	3	7	2	5	2	19

$$p(w = information, c = data) = \frac{6}{19} = 0.32$$

$$p(w = information) = \frac{11}{19} = 0.58 \quad p(c = data) = \frac{7}{19} = 0.37$$

$$pmi(information, data) = \log_2 \frac{0.32}{0.37 \cdot 0.58} \approx 0.58$$

Slide credit: David Mortensen

Worked Example: Computing PPMI from Term-Context Matrix (Part II)

$PPMI(w, c)$

	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pineapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.32	-	0.47	-

Slide credit: David Mortensen

Issues with PPMI

- PMI is biased toward infrequent events.
- Very rare words have very high PMI values.
- Two solutions:
  - Give rare words slightly higher probabilities
  - Use add-one smoothing (which has a similar effect)

Slide credit: David Mortensen

Text classification

Text classification

"My dear Mr. Bennet," said his lady to him one day, "have you heard that Netherfield Park is let at last?"

- ROMANCE
- *Pride and Prejudice*
- *DIALOG*

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.

http://www.123contactform.com/contact-form-StanfordNew1-236335.html

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.

Slide adapted from Jurafksy & Martin

What is the subject of this medical article?

MEDLINE Article

Syntactic frame and verb bias in aphasia: Plausibility judgments of undergoer-subject sentences

?

MeSH Subject Category Hierarchy

Antagonists and Inhibitors

Blood Supply

Chemistry

Drug Therapy

Embryology

Epidemiology

...

Slide adapted from Jurafksy & Martin

Text Classification

We have a set of documents that we want to *classify* into a small set *classes*.

Applications:

- **Topic classification:** you have a set of news articles that you want to classify as finance, politics, or sports.
- **Sentiment detection:** you have a set of movie reviews that you want to classify as good, bad, or neutral.
- **Language Identification:** you have a set of documents that you want to classify as English, Mandarin, Arabic, or Hindi.
- **Reading level:** you have a set of articles that you want to classify as kindergarten, 1st grade, ...12th grade.
- **Author identification:** you have a set of fictional works that you want to classify as Shakespeare, James Joyce, ...
- **Genre identification:** you have a set of documents that you want to classify as report, editorial, advertisement, blog, ...

Slide credit: David Mortensen

Example: Sentiment Detection

	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

Slide credit: David Mortensen

How to evaluate your classifier

Gold labels and predicted labels

Document	gold label	predicted label
just plain boring	–	–
entirely predictable	–	–
no surprises and very few laughs	–	+
very powerful	+	–
the most fun film of the summer	+	+

The **gold** label is the label that a human assigned to the document.

The **predicted** or **hypothesized** label is the label that the classifier assigned to the document.

Slide credit: David Mortensen

We Can Evaluate a Classifier Using Accuracy

Accuracy is our first shot.

- Accuracy:

$$\frac{\text{how many instances your system got right}}{\text{all instances in the test set}}$$

Slide credit: David Mortensen

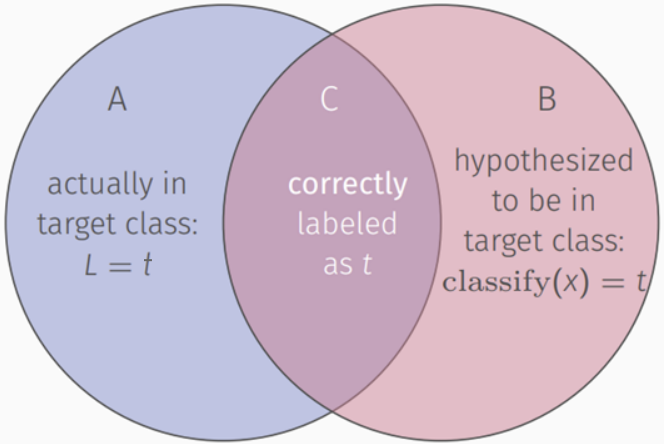
Issues with using test set accuracy

- Imagine an “important email” classifier that notifies you when you get an important email
- Suppose that 99% of the messages you receive are junk and not important (we’re being realistic here)
- An easy important email classifier: classify nothing as important
  - You would get lots of work done, because you wouldn’t be distracted by email
  - The email classifier would have an accuracy of ~99%
  - Everybody would be happy except for your boss
- You must take the relative importance of the classes into account, and the cost of the error types

Slide adapted from David Mortensen

Evaluation in the Two-class case

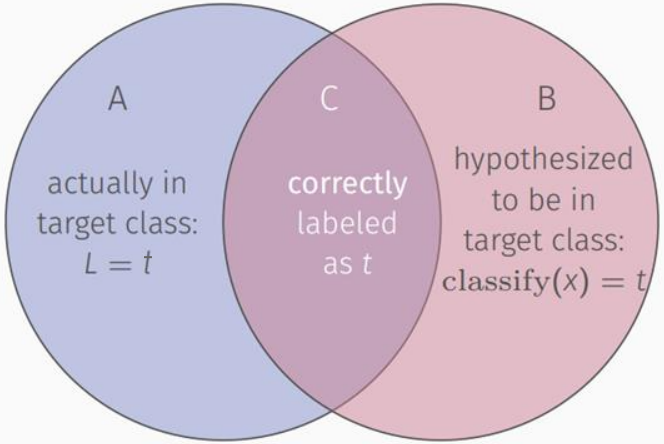
- Suppose we have one of the classes $t \in \mathcal{L}$ as the target class.
- We would like to identify documents with label t in the test data.
- We get



- Precision $\hat{P} = \frac{C}{B}$ (percentage of documents `classify` correctly labeled as t)
- Recall $\hat{R} = \frac{C}{A}$ (percentage of actual t labeled documents correctly labeled as t)
- $F_1 = 2\frac{\hat{P} \cdot \hat{R}}{\hat{P} + \hat{R}}$

Slide credit: David Mortensen

A Different View – Contingency Tables



	$L = t$	$L \neq t$	
$\texttt{classify}(X) = t$	C (true positives)	$B \backslash C$ (false positives)	B
$\texttt{classify}(X) \neq t$	$A \backslash C$ (false negatives)	(true negatives)	
	A		

precision = tp/(tp+fp)

recall = tp/(tp+fn)

Slide credit: David Mortensen

Why precision and recall

- 2-way precision and recall are specific to a target class
- Accuracy=99% on important email detection

  but

- Recall = 0 (out of all actually important emails, got none)
- Precision and recall, unlike accuracy, emphasize true positives: finding the things that we are supposed to be looking for

Slide adapted from Jurafksy & Martin

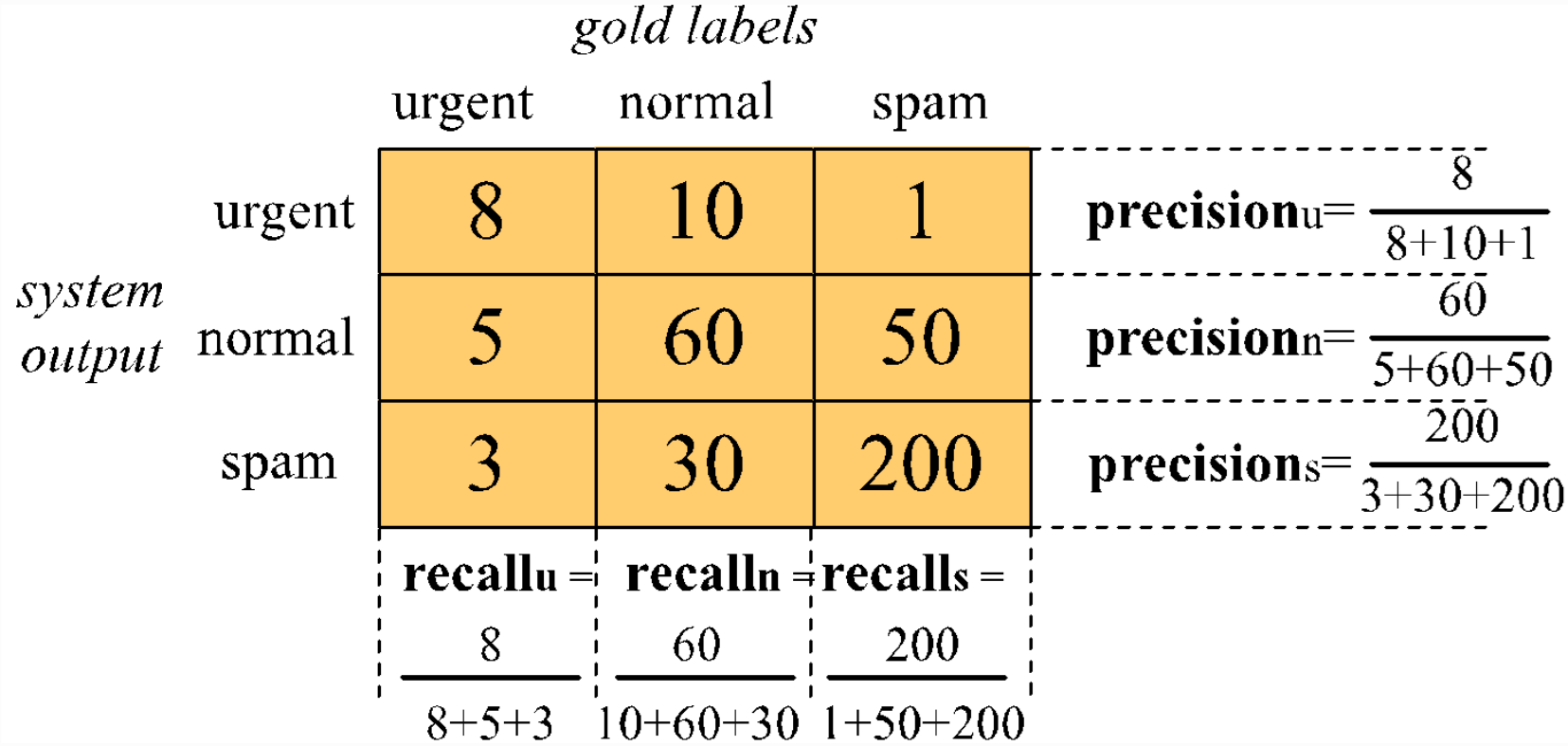
A combined measure: F1-score

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

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

Slide adapted from Jurafksy & Martin

Confusion matrix for 3-class classification

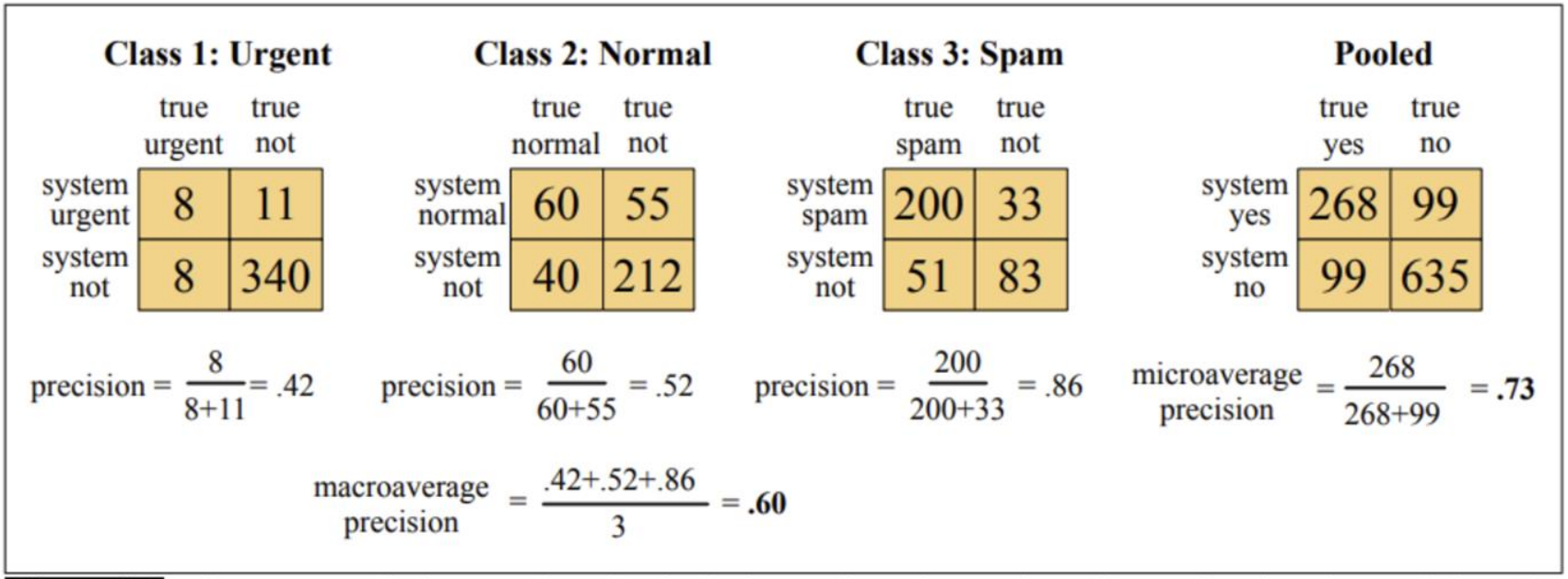


Slide adapted from Jurafksy & Martin

Evaluation with > 2 Classes

- **Macroaveraged precision and recall**: let each class be the target and report the average $\hat{P}$ and $\hat{R}$ across all classes.
- **Microaveraged precision and recall**: pool all one-vs.-rest decisions into a single contingency table, calculate $\hat{P}$ and $\hat{R}$ from that.

Example of more than two classes



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

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

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

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.

Slide credit: David Mortensen, Jurafksy & Martin

Train/dev/test splits and cross-validation

Development Sets ("Devsets") and Cross-validation

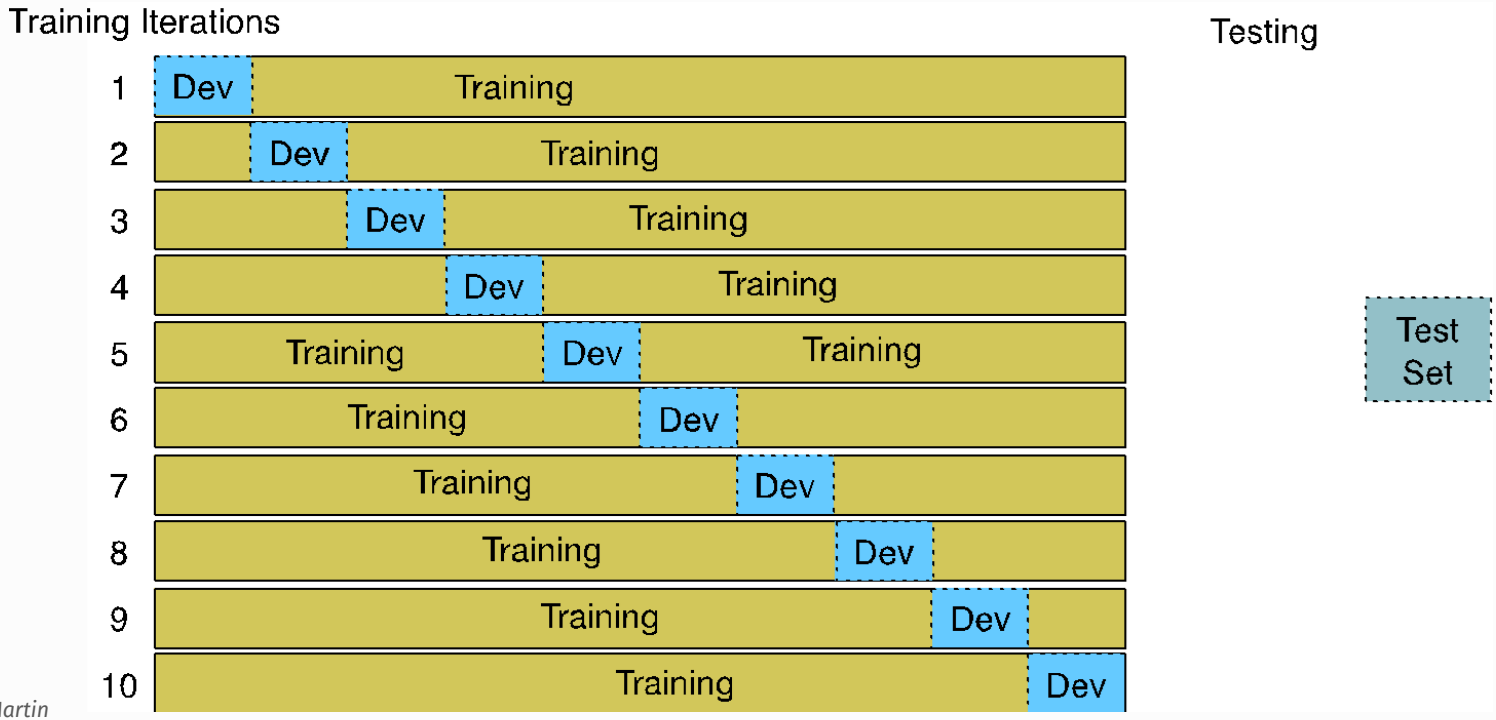
Train on training set, tune on dev set, report on test set

- **Do not look at test set**
- Using a dev set 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?

Slide adapted from Jurafksy & Martin

Cross-validation: multiple splits

- Pool results over splits, Compute pooled dev performance
- Good for when you don't have much data (<10k instances rule of thumb)



Slide adapted from Jurafksy & Martin

Harms in classification in NLP

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

Slide adapted from Jurafksy & Martin

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.

Slide adapted from Jurafksy & Martin

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

Can't fully "remove" bias because exists in societies that produced texts we use

So need to be explicit about what those biases may be through **data statements** and **model cards**

Slide adapted from Jurafksy & Martin

Data statements [Bender & Friedman 2018]

For each dataset you release, document:

- Curation rationale: why were certain texts selected
- Language variety
- Speaker demographic
- Annotator demographic
- Speech situation
  - Time and place, modality, scripted vs spontaneous, intended audience
- Text characteristics
  - Genre, topic
- Recording quality (for speech)

Slide adapted from Jurafksy & Martin

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

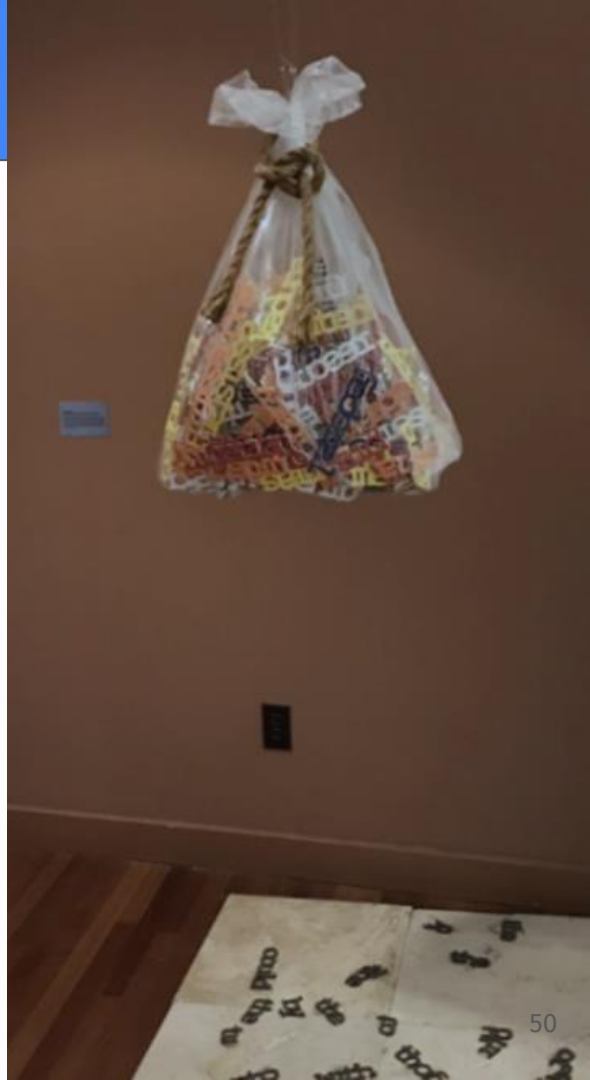
Coding activity

Class Jupyter notebook

- Click on this nbgitpuller link
  - Or find the link on the course website
- Open **session9_tfidf_clickbait_eval.ipynb**

Conclusion

- Downweighting words that appear frequently in term-document and term-term matrices
  - **tf-idf** for document representations
    - Downweight terms that appear across many documents
  - **PPMI** for word associations
    - Downweight words that appear with many other words
- Text classification is an NLP task learning a mapping from texts to a set of discrete labels
- Classifiers are evaluated with accuracy, precision, recall and F1-score
- Cross-validation is an alternative to train/dev/test split to estimate performance
- Text classification systems can be biased against the language or references to marginalized groups



Questions?