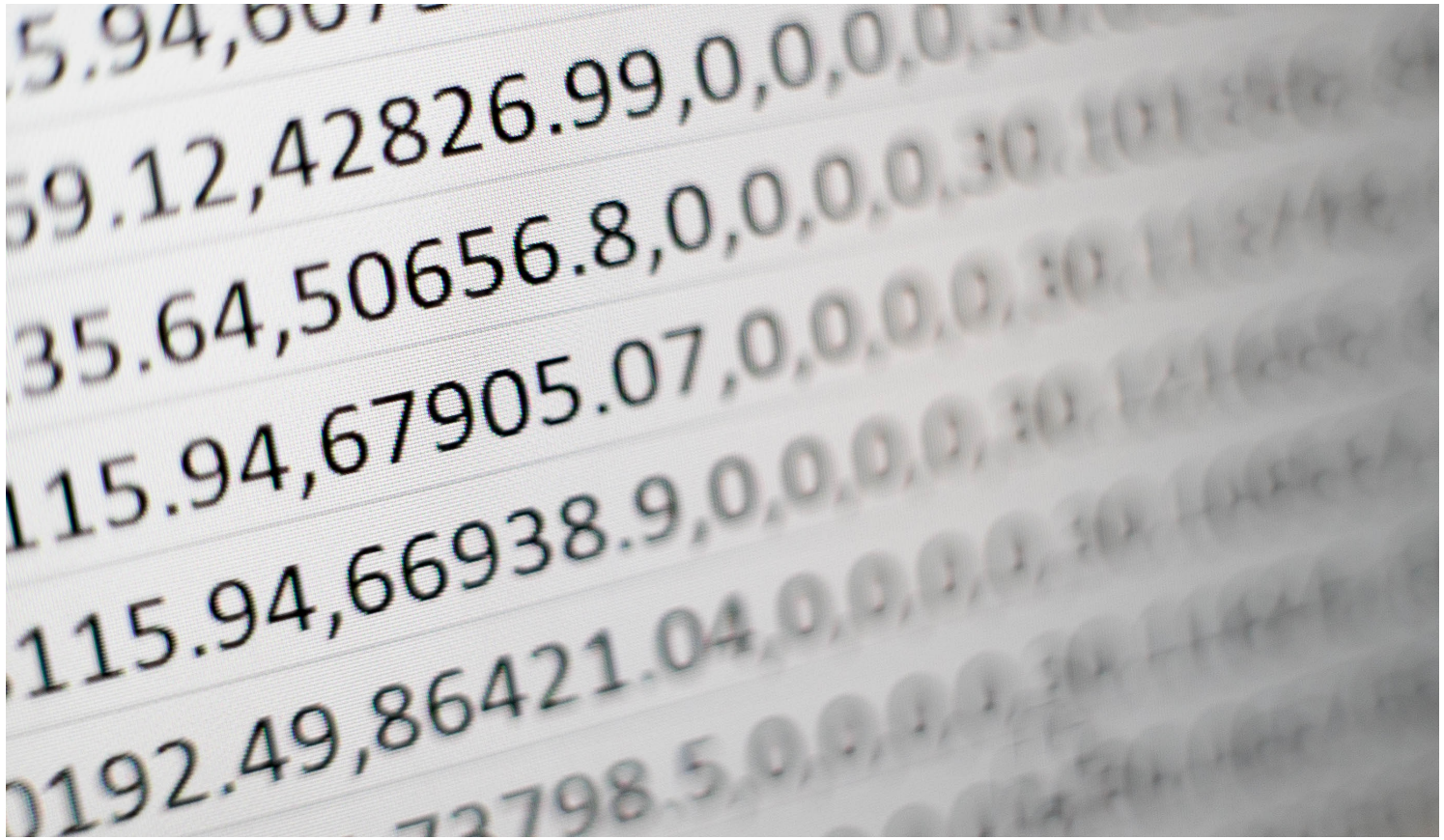


Naïve Bayes, Text Classification, and Evaluation Metrics

Natalie Parde, Ph.D.

Department of Computer Science

University of Illinois at Chicago

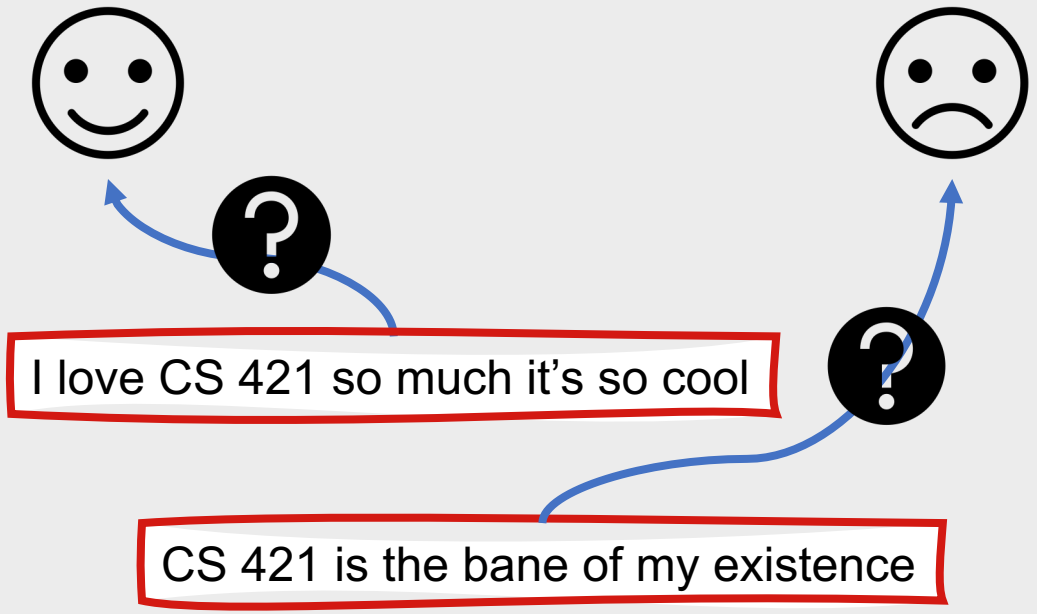
CS 421: Natural Language Processing

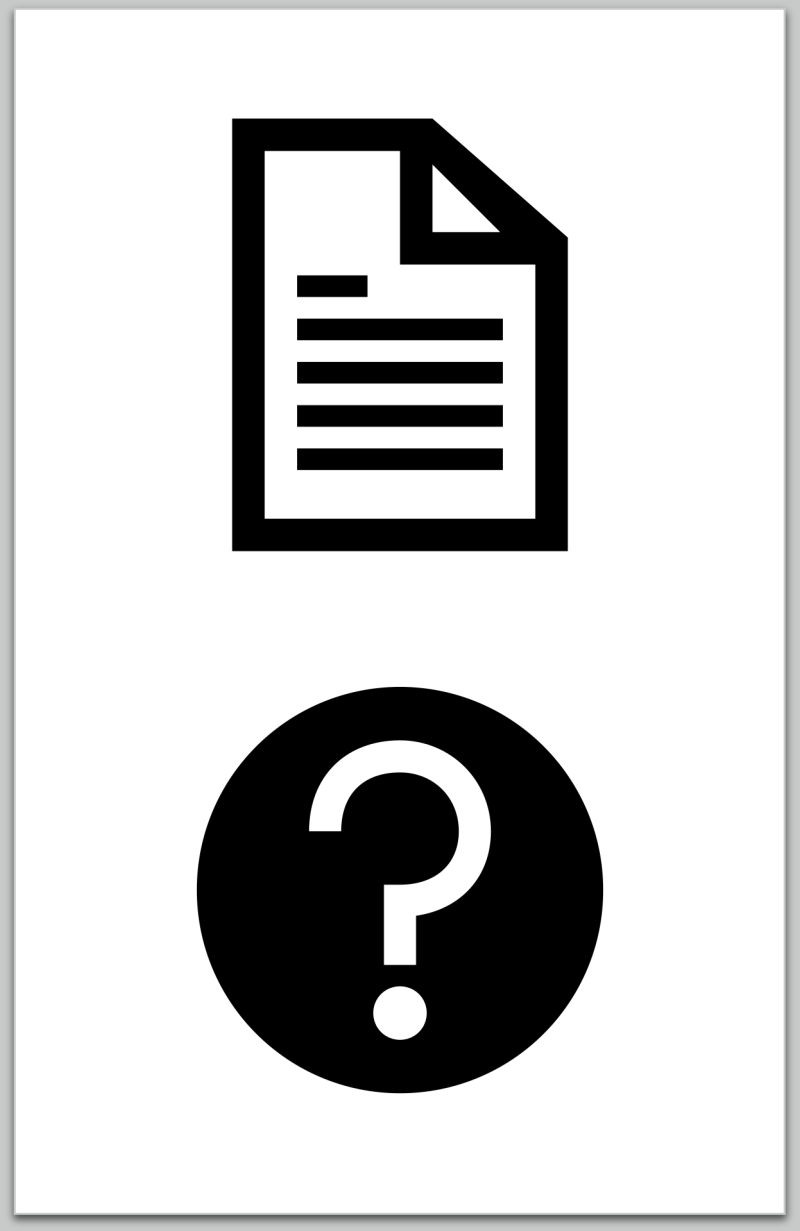
Fall 2019

Many slides adapted from Jurafsky and Martin (https://web.stanford.edu/~jurafsky/slp3/).

What is Naïve Bayes?

- A **probabilistic classifier** that learns to make **predictions** from a predefined set of **labels** for new documents



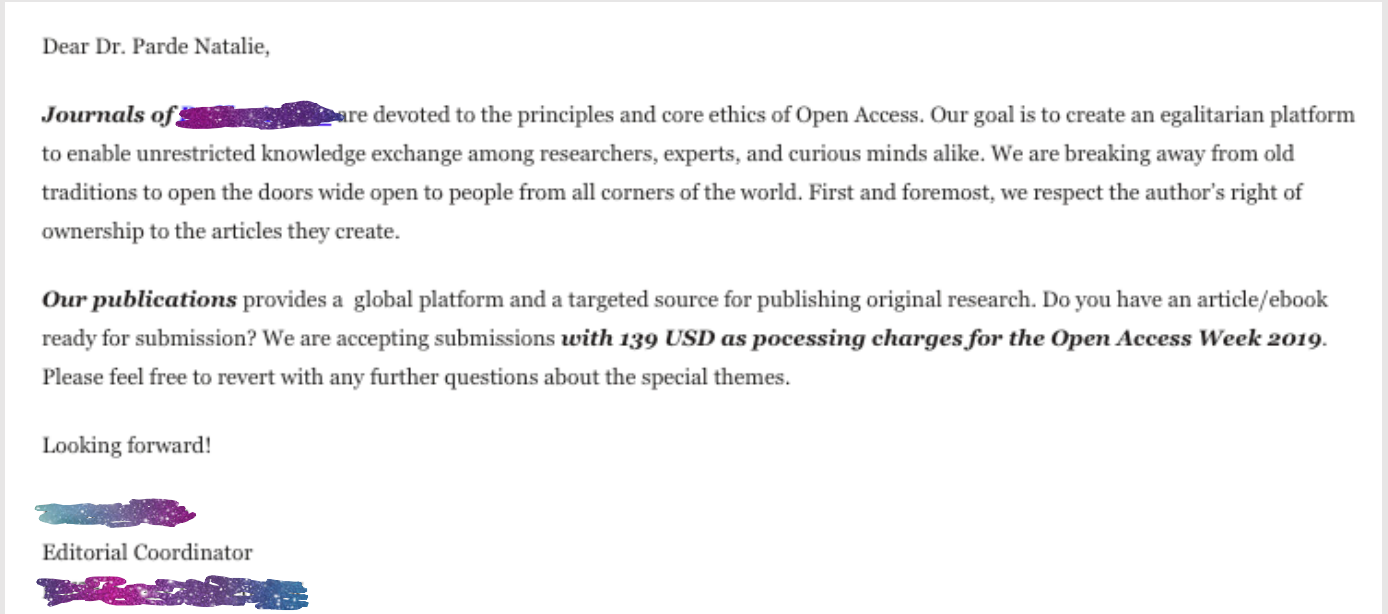


Classification

- Given an item (e.g., document, sentence, word, image, audio file, etc.), what is its **category**?
- Fundamental to natural language processing
- Focus in this course: **text categorization**
  - Assigning a label or category to an entire text or document

Common Applications of Text Categorization

Spam detection



Dear Dr. Parde Natalie,

Journals of [redacted] are devoted to the principles and core ethics of Open Access. Our goal is to create an egalitarian platform to enable unrestricted knowledge exchange among researchers, experts, and curious minds alike. We are breaking away from old traditions to open the doors wide open to people from all corners of the world. First and foremost, we respect the author's right of ownership to the articles they create.

Our publications provides a global platform and a targeted source for publishing original research. Do you have an article/ebook ready for submission? We are accepting submissions ***with 139 USD as pocessing charges for the Open Access Week 2019***. Please feel free to revert with any further questions about the special themes.

Looking forward!

[redacted]

Editorial Coordinator

[redacted]

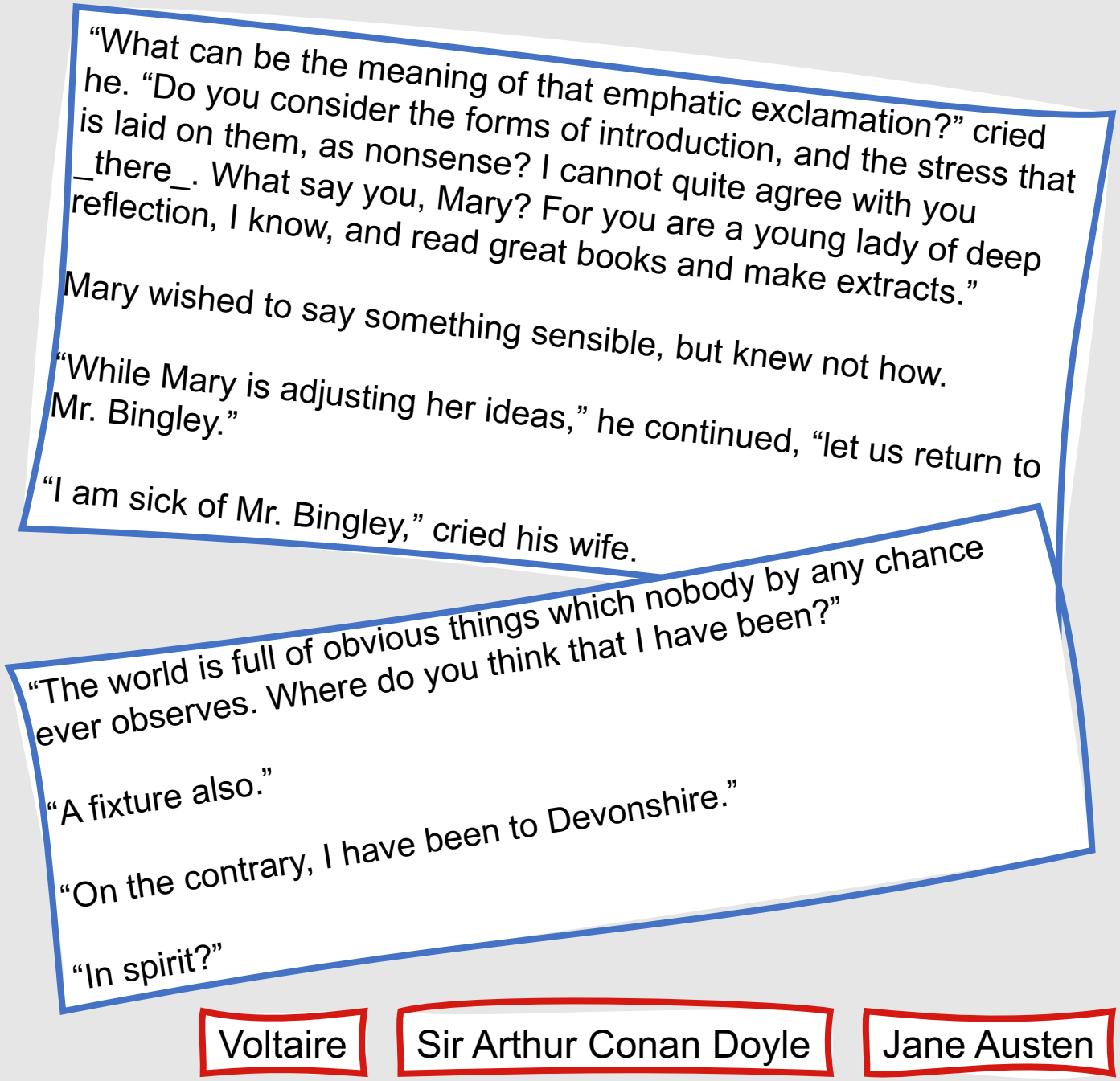
Spam

Not Spam

Common Applications of Text Categorization

Spam detection

Authorship attribution



Common Applications of Text Categorization

Spam detection

Authorship attribution

Sentiment analysis

Natalie's poem about Halloween was really dreadful. The word "Halloween" doesn't even rhyme with "trick or treat!" She should stick to writing NLP programs.

Natalie's poem about Halloween was a true delight! The way she rhymed "Halloween" with "trick or treat" was artful and unexpected. I can't wait to read what she writes next!

Natalie wrote a poem about Halloween. She wrote it as if the words "Halloween" and "trick or treat" rhyme with one another. It was her first poem.

Positive

Negative

Neutral

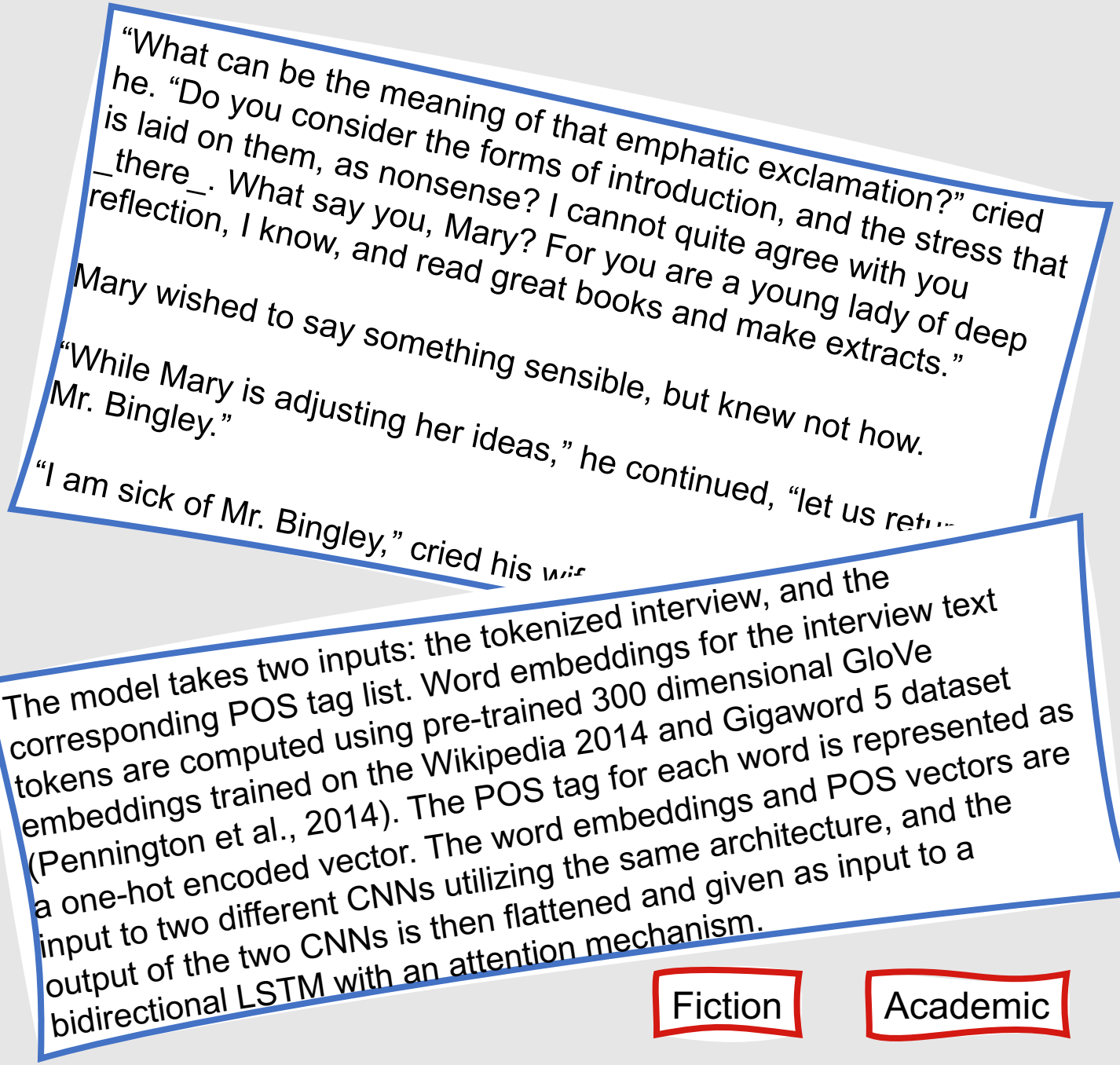
Common Applications of Text Categorization

Spam detection

Authorship attribution

Sentiment analysis

Domain identification



Common Applications of Text Categorization

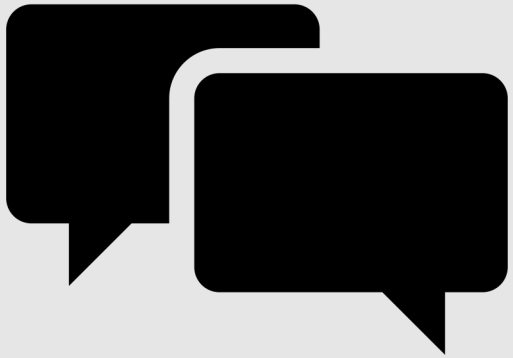
Spam detection

Authorship attribution

Sentiment analysis

Domain identification

…and many, many others!



Classification is also used for tasks below the document level.

Sentence segmentation

- Is this the beginning of a new sentence?

Character disambiguation

- Is this period marking the end of a sentence, or is it part of an acronym?

Tokenization

- Is this the last character in a word?

Language modeling

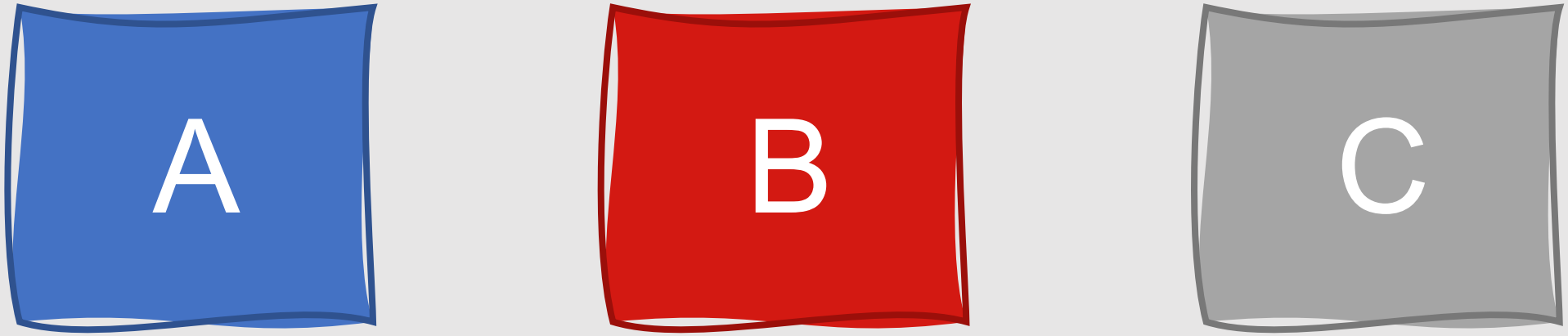
- Is the next word “the”?

Part-of-speech tagging

- Is this word a noun or a verb?

Classification

- Goal:
  - Take a single **observation**
  - Extract some useful **features**
  - Classify the observation into one of a set of discrete classes based on those features



How is classification performed?

Rule-based methods

- **Handwrite** a set of rules based on expected differences for features from different classes, and use that information to classify test data
  - Fiction will probably have more quotation marks than academic text
  - Positive text will probably contain “love” more frequently than negative text

Statistical methods

- **Learn** which features best distinguish different classes based on a collection of training data, and use that information to classify test data
  - In the training data, fiction text contained > 6 quotation marks
  - In the training data, 60% of positive texts contained the word “love”

Is rule-based or statistical classification better?

- Both have cases in which they work better
- In modern computing environments (i.e., scenarios with plentiful data), **statistical classification is generally a better choice**

Why is statistical classification preferred?

Situations can change over time

- ghost = noun
- Emerging use: ghost = verb

So can data

- “I ship packages”
- “I ship them”

Humans aren’t necessarily good at coming up with rules

- “I don’t love it”

Supervised Machine Learning

- **Statistical classification with a labeled training set**
- Each input instance is associated with a known output (the label)
- Goal: Learn how to map from a new observation (with an unknown output) to a correct output
  - Compare predicted outputs with the correct outputs that we know from a labeled test set

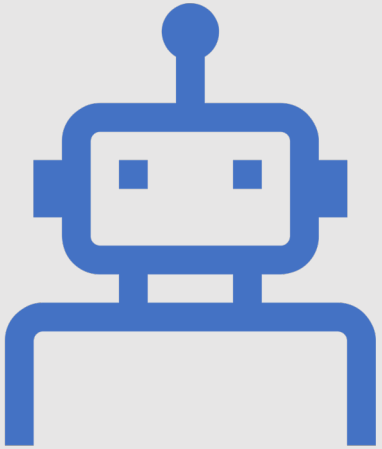
Formal Definition of Supervised Machine Learning

- Take an input x and a fixed set of output classes $Y = \{y_1, y_2, \ldots, y_M\}$
- Return a predicted class $y \in Y$
- In text classification, we often refer to x as d (for "document") and y as c (for "class")
- We have a training set of N documents, each of which have been manually labeled with a class: $\{(d_1, c_1), \ldots, (d_N, c_N)\}$
- Goal: Learn a classifier that is capable of mapping from a new document d to its correct class $c \in C$

Probabilistic Classifiers

- **Probabilistic classifiers:** Those that make their classification decisions based on probabilities
- In addition to making a prediction, probabilistic classifiers will tell us the probability of the observation (the data instance) belonging in each class
- Why is this useful?
  - Downstream decision-making!
  - If combining multiple classifiers for a task, having direct access to these probabilities can be useful for making our end decision

What are some common classification algorithms?



- **Naïve Bayes**
- Logistic Regression
- Support Vector Machines
- K-Nearest Neighbors
- Neural Networks (Multilayer Perceptrons)

Supervised classification algorithms can generally be divided into two categories.

- **Generative:** Build a model of how a class could generate some input data; given an observation, return the class most likely to have generated the observation
  - Example: Naïve Bayes
- **Discriminative:** Learn what features from the input are most useful to discriminate between the different possible classes
  - Example: Logistic Regression

Naïve Bayes Classifiers

Gaussian Naïve Bayes: Assumes the outcomes for the input data are normally distributed along a continuum

Multinomial Naïve Bayes: Assumes the outcomes for the input data follow a multinomial distribution (there is a discrete set of possible outcomes)

Binomial Naïve Bayes: Assumes the outcomes for the input data follow a binomial distribution (there are two possible outcomes)

Multinomial Naïve Bayes

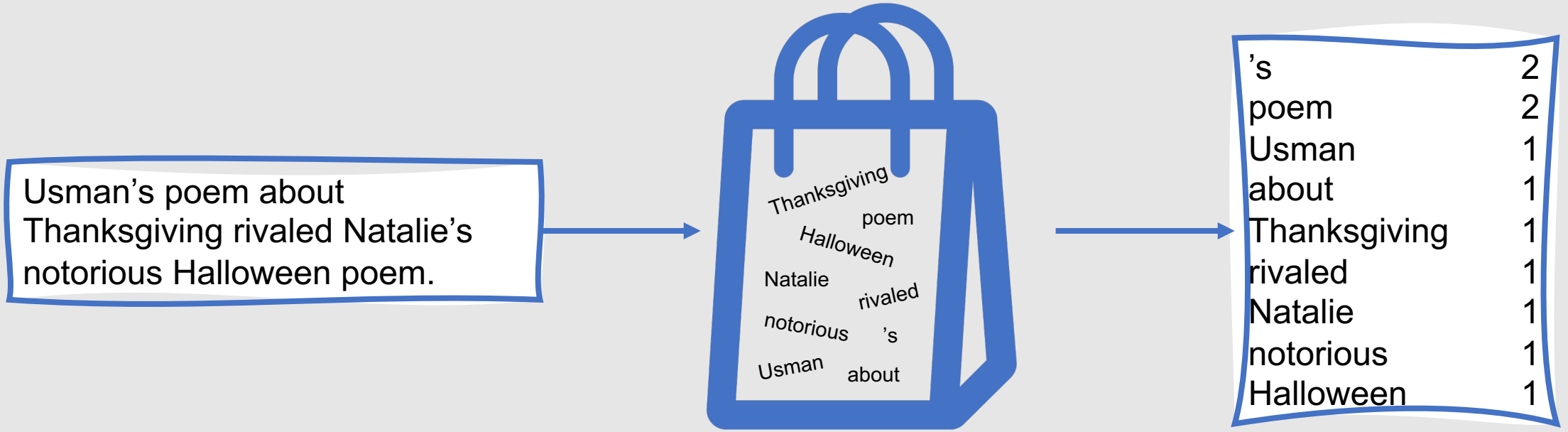
- Each instance falls into one of *n* classes
  - *n*=2 → Binomial Naïve Bayes
- Simple classification based on Bayes' rule
- Simple document representation
  - Technically, any features can be used
  - Traditionally, bag of words features are used

Why is it "Naïve" Bayes?

- Naïve Bayes classifiers make a naïve assumption about how features interact with one another: quite simply, they assume that they don't
- They instead **assume that all features are independent from one another**
- Is this really the case?
  - No---as already seen with language models, words are dependent on their contexts
  - However, Naïve Bayes classifiers still perform reasonably well despite adhering to this naïve assumption

Naïve Bayes Intuition

- Represent each document as a **bag of words**
  - Unordered set of words and their frequencies
- Decide how likely it is that a document belongs to a class based on its distribution of **word frequencies**



Naïve Bayes is a probabilistic classifier.

- For a document *d*, out of all classes $c \in C$ the classifier returns the class *c*’ which has the maximum **posterior probability**, given the document
  - $c' = \underset{c \in C}{\operatorname{argmax}} P(c|d)$

Naïve Bayes computes probabilities using Bayesian inference.

- Bayesian inference uses **Bayes' rule** to transform probabilities like those shown previously into other probabilities that are easier or more convenient to calculate
- Bayes' rule:
  - $P(x|y) = \frac{P(y|x)P(x)}{P(y)}$

Applying Bayesian inference to Naïve Bayes

- If we take Bayes' rule:
  - $P(x|y) = \frac{P(y|x)P(x)}{P(y)}$
- And substitute it into our previous equation:
  - $c' = \underset{c \in C}{\operatorname{argmax}} P(c|d)$
- We get the following:
  - $c' = \underset{c \in C}{\operatorname{argmax}} P(c|d)$

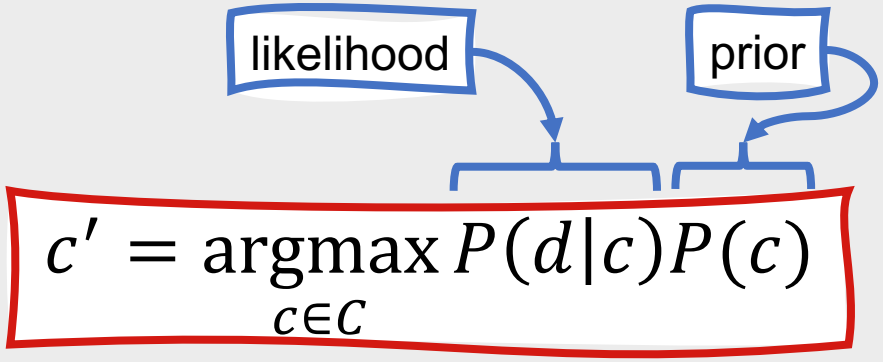
$$= \underset{c \in C}{\operatorname{argmax}} \frac{P(d|c)P(c)}{P(d)}$$

How can we simplify this?

- Drop the denominator P(d)
  - We'll be computing $\frac{P(d|c)P(c)}{P(d)}$ for each class, but P(d) doesn't change for each class
    - We're always asking about the most likely class for the same document d
- Thus:
  - $c' = \underset{c \in C}{\operatorname{argmax}} P(c|d)$
    $= \underset{c \in C}{\operatorname{argmax}} P(d|c)P(c)$

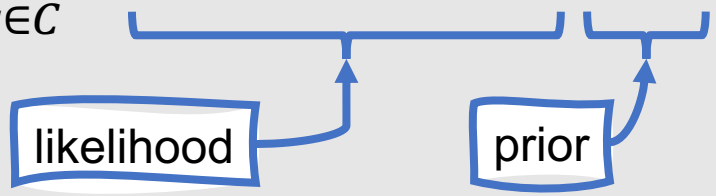
What does this mean?

- The most probable class *c*’ given some document *d* is the class that has the highest product of two probabilities
  - **Prior probability** of the class P(*c*)
  - **Likelihood** of the document P(*d*|*c*)



How do we represent a document?

- As already mentioned, documents are generally represented as **bags of words**
- Bags of words are simply sets of features $\{f_1, f_2, \ldots, f_n\}$, where each feature *f* corresponds to the frequency of one of the words in the vocabulary
- This means that:
  - $c' = \underset{c \in C}{\text{argmax}}\, P(d|c)P(c) = \underset{c \in C}{\text{argmax}}\, P(f_1, f_2, \ldots, f_n|c)\, P(c)$



As-is, this equation is still quite difficult to compute.

- It requires that we estimate the probability of every possible combination of features X class
  - For example, every possible set of words and their positions
- This would require a huge amount of training data
- This is why our two simplifying assumptions are useful:
  - A word's position doesn't matter
    - f_1, f_2, …, f_n encode only a word's identify, not its position
    - This is the general bag-of-words assumption
  - The probabilities P(f_i|c) are independent given the class c
    - This is the general Naïve Bayes assumption

The Naïve Bayes assumption means that we can "naïvely" multiply our probabilities for each feature together.

- Why?
  - They're assumed to be independent of one another!
- Therefore:
  - $P(f_1, f_2, \ldots, f_n|c) = P(f_1|c) * P(f_2|c) * \cdots * P(f_n|c)$

This brings us to our final equation.

$$c' = \underset{c \in C}{\operatorname{argmax}} P(d|c)P(c)$$

$$= \underset{c \in C}{\operatorname{argmax}} P(f_1, f_2, \ldots, f_n|c)\, P(c)$$

$$= \underset{c \in C}{\operatorname{argmax}} \boldsymbol{P(c) \prod_{f \in F} P(f|c)}$$

How do we apply our Naïve Bayes classifier to text?

- Simply walk through each word in the document and compute its probability
  - T ← all word (token) positions in a document
  - $c' = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i \in T} P(w_i|c)$
- To avoid underflow (the generation of numbers that are too tiny to be adequately represented) and increase speed, we usually do these computations in log space:
  - $c' = \underset{c \in C}{\operatorname{argmax}} \log P(c) + \sum_{i \in T} \log P(w_i|c)$

Linear Classifiers

- When we perform these computations in log space, we end up predicting a class as a linear function of the input features
  - $c' = \underset{c \in C}{\text{argmax}} \log P(c) + \sum_{i \in T} \log P(w_i|c)$
- Classifiers that use a linear combination of the inputs to make their classification decisions are called **linear classifiers**
  - Naïve Bayes
  - Logistic Regression

How do we train a Naïve Bayes classifier?

- More specifically, how do we learn P(c) and P($f_i|c$)?
- We need to use **maximum likelihood estimates**
- **Maximum likelihood estimation:** Maximizing a likelihood function such that for the observed instance, the probability of the observation occurring is most probable

We can find maximum likelihood estimates using frequencies from our text data.

- To compute P(c), we figure out what percentage of the documents in our training set are in class c
  - Let N_c be the number of documents in our training data with class c
  - Let N_{doc} be the total number of documents
  - $P(c)' = \frac{N_c}{N_{doc}}$

Remember, in our scenario we're assuming that a feature is just a word in a document's bag of words.

- Thus, to compute P($f_i|c$), we'll just need P($w_i|c$)
- We can just compute this as the fraction of times w_i appears among all words in all documents of class c
- How do we do this?
  - Concatenate all documents from class c into a big super-document of text
  - Find the frequency of w_i in this super-document to find the maximum likelihood estimate of the probability:
    - $P(w_i|c)' = \frac{count(w_i,c)}{\sum_{w \in V} count(w,c)}$
    - Note that V is the set of all word types across all classes, not just the words in class c

All good, right?

- Almost (but not quite)
- The current equations for estimating maximum likelihood do nothing to address zero probabilities

’s	2
poem	2
Usman	1
about	1
Thanksgiving	1
rivaled	1
Natalie	1
notorious	1
Halloween	1

$$P(\text{“prose”}|c) = \frac{count(\text{“prose”}, c)}{\sum_{w \in V} count(w, c)} = 0$$

Zero probabilities can be very problematic.

- Naïve Bayes naïvely multiplies all the feature likelihoods together
- This means that if there is a single zero probability when computing the word likelihoods, the entire probability for the class will be 0
  - $c' = \underset{c \in C}{\text{argmax}}\, P(c) \prod_{i \in T} P(w_i|c)$

How do we fix this issue?

- Simplest solution: Laplace (add-one) smoothing
- $P(w_i|c)' = \frac{count(w_i,c)+1}{\sum_{w \in V}(count(w,c)+1)} = \frac{\boldsymbol{count(w_i,c)+1}}{(\sum_{\boldsymbol{w \in V}}(\boldsymbol{count(w,c)})+\boldsymbol{|V|})}$
- Again, remember …*V* consists of the words from all classes, not just the words in class *c*

What about unknown words?

- Some words will inevitably occur in the test data despite never having occurred in the training data
- In Naïve Bayes, we can just go ahead and ignore those words
  - Remove them from the test document
  - Not include any probabilities for them at all in our calculation of *c*’

What about stop words?

- **Stop words** are very frequent words like *a* and *the*
- In some scenarios, it may make sense to ignore those words
  - Stop words may occur with equal frequency in all classes
  - However, this isn't always the case (e.g., spam detection)
- Stop words can be defined either automatically or using a predefined stop word list
  - Automatically:
    - Sort the vocabulary by frequency in the training set
    - Define the top 10-100 vocabulary entries as stop words
  - Predefined List:
    - Search online, or see if the package you're using (e.g., NLTK) already has one

Final, Formal Algorithm

Train Naïve Bayes

for each class c∈C: # Calculate P(c)

  N_{doc} ← |D|

  N_c ← number of d∈D from class c

  logprior[c] ← $\log(N_c/N_{doc})$

  V ← vocabulary of D

  superdoc[c] ← d∈D from class c

  for each word w in V:

    count(w,c) ← superdoc[c].count(w)

    loglikelihood[w,c] ← $\log(\frac{count(w_i,c)+1}{(\sum_{w\in V}(count(w,c))+|V|)})$

return logprior,loglikelihood,V

Test Naïve Bayes

for each class c∈C:

  sum[c] ← logprior[c]

  for each position i in testdoc:

    word ← testdoc[i]

    if word∈V:

      sum[c] ← sum[c]+loglikelihood[word,c]

return $\underset{c}{\text{argmax}}\, sum[c]$

Example: Naïve Bayes

Natalie was soooo thrilled that Usman had a famous new poem.

She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.

Usman was happy that his poem about Thanksgiving was so successful.

He congratulated Natalie for getting #2 on the bestseller list.

Natalie told Usman she was soooo totally happy for him.

Example: Naïve Bayes

Natalie was soooo thrilled that Usman had a famous new poem.

Sarcastic

She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.

Sarcastic

Usman was happy that his poem about Thanksgiving was so successful.

Not Sarcastic

He congratulated Natalie for getting #2 on the bestseller list.

Not Sarcastic

Natalie told Usman she was soooo totally happy for him.

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Usman had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Usman was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Usman she was soooo totally happy for him.	?

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Usman had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Usman was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Usman she was soooo totally happy for him.	?

- What is the prior probability for each class?
  - $P(c)' = \frac{N_c}{N_{doc}}$

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Usman had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Usman was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Usman she was soooo totally happy for him.	?

- What is the prior probability for each class?
  - $P(c)' = \frac{N_c}{N_{doc}}$
- P(Sarcastic) = 2/4 = 0.5
- P(Not Sarcastic) = 2/4 = 0.5

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Usman had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Usman was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Usman she was soooo totally happy for him.	?

- What is the prior probability for each class?
  - $P(c)' = \frac{N_c}{N_{doc}}$
- P(Sarcastic) = 2/4 = 0.5
- P(Not Sarcastic) = 2/4 = 0.5
- Note: This means we have a **balanced training set**
  - Balanced: An equal number of samples for each class

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Usman had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Usman was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Usman she was soooo totally happy for him.	?

- Taking a closer look at our test instance, let's remove:
  - Stop words
  - Unknown words

Natalie told Usman she was soooo totally happy for him.

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Usman had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Usman was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Usman she was soooo totally happy for him.	?

- Taking a closer look at our test instance, let's remove:
  - **Stop words**
  - Unknown words

Natalie told Usman she was soooo totally happy for him.

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Usman had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Usman was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Usman she was soooo totally happy for him.	?

- Taking a closer look at our test instance, let's remove:
  - Stop words
  - **Unknown words**

Natalie told Usman she was soooo totally happy for him.

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Usman had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Usman was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Usman she was soooo totally happy for him.	?

- What are the likelihoods from the training set for the remaining words in the test instance?
  - $P(w_i|c)' = \frac{count(w_i,c)}{\sum_{w \in V} count(w,c)}$

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Natalie told Usman she was soooo totally happy for him.

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Usman had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Usman was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Usman she was soooo totally happy for him.	?

- What are the likelihoods from the training set for the remaining words in the test instance?
  - $P(w_i|c)' = \frac{count(w_i,c)+1}{(\sum_{w \in V} count(w,c))+|V|}$
  - P("Natalie"|Sarcastic) = $\frac{1+1}{27+34} = 0.033$
  - P("Natalie"|Not Sarcastic) = $\frac{1+1}{21+34} = 0.036$

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Natalie told Usman she was soooo totally happy for him.

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Usman had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Usman was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Usman she was soooo totally happy for him.	?

- What are the likelihoods from the training set for the remaining words in the test instance?
  - $P(w_i|c)' = \frac{count(w_i,c)+1}{(\sum_{w \in V} count(w,c))+|V|}$
  - P("Natalie"|Sarcastic) = $\frac{1+1}{27+34} = 0.033$
  - P("Natalie"|Not Sarcastic) = $\frac{1+1}{21+34} = 0.036$
  - P("Usman"|Sarcastic) = $\frac{1+1}{27+34} = 0.033$
  - P("Usman"|Not Sarcastic) = $\frac{1+1}{21+34} = 0.036$

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Natalie told Usman she was soooo totally happy for him.

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Usman had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Usman was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Usman she was soooo totally happy for him.	?

- What are the likelihoods from the training set for the remaining words in the test instance?
  - $P(w_i|c)' = \frac{count(w_i,c)+1}{(\sum_{w \in V} count(w,c))+|V|}$
  - P("Natalie"|Sarcastic) = $\frac{1+1}{27+34} = 0.033$
  - P("Natalie"|Not Sarcastic) = $\frac{1+1}{21+34} = 0.036$
  - P("Usman"|Sarcastic) = $\frac{1+1}{27+34} = 0.033$
  - P("Usman"|Not Sarcastic) = $\frac{1+1}{21+34} = 0.036$
  - P("soooo"|Sarcastic) = $\frac{1+1}{27+34} = 0.033$
  - P("soooo"|Not Sarcastic) = $\frac{0+1}{21+34} = 0.018$

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Natalie told Usman she was soooo totally happy for him.

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Usman had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Usman was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Usman she was soooo totally happy for him.	?

- What are the likelihoods from the training set for the remaining words in the test instance?
  - $P(w_i|c)' = \frac{count(w_i,c)+1}{(\sum_{w \in V} count(w,c))+|V|}$
  - P("Natalie"|Sarcastic) = $\frac{1+1}{27+34} = 0.033$
  - P("Natalie"|Not Sarcastic) = $\frac{1+1}{21+34} = 0.036$
  - P("Usman"|Sarcastic) = $\frac{1+1}{27+34} = 0.033$
  - P("Usman"|Not Sarcastic) = $\frac{1+1}{21+34} = 0.036$
  - P("soooo"|Sarcastic) = $\frac{1+1}{27+34} = 0.033$
  - P("soooo"|Not Sarcastic) = $\frac{0+1}{21+34} = 0.018$
  - P("totally"|Sarcastic) = $\frac{1+1}{27+34} = 0.033$
  - P("totally"|Not Sarcastic) = $\frac{0+1}{21+34} = 0.018$

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Natalie told Usman she was soooo totally happy for him.

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Usman had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Usman was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Usman she was soooo totally happy for him.	?

- What are the likelihoods from the training set for the remaining words in the test instance?
  - $P(w_i|c)' = \frac{count(w_i,c)+1}{(\sum_{w \in V} count(w,c))+|V|}$
  - P("Natalie"|Sarcastic) = $\frac{1+1}{27+34} = 0.033$
  - P("Natalie"|Not Sarcastic) = $\frac{1+1}{21+34} = 0.036$
  - P("Usman"|Sarcastic) = $\frac{1+1}{27+34} = 0.033$
  - P("Usman"|Not Sarcastic) = $\frac{1+1}{21+34} = 0.036$
  - P("soooo"|Sarcastic) = $\frac{1+1}{27+34} = 0.033$
  - P("soooo"|Not Sarcastic) = $\frac{0+1}{21+34} = 0.018$
  - P("totally"|Sarcastic) = $\frac{1+1}{27+34} = 0.033$
  - P("totally"|Not Sarcastic) = $\frac{0+1}{21+34} = 0.018$
  - P("happy"|Sarcastic) = $\frac{0+1}{27+34} = 0.016$
  - P("happy"|Not Sarcastic) = $\frac{1+1}{21+34} = 0.036$

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Natalie told Usman she was soooo totally happy for him.

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Usman had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Usman was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Usman she was soooo totally happy for him.	?

- Given all of this information, how should we classify the test sentence?
  - $c' = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i \in T} P(w_i|c)$

Word	P(Word|Sarcastic)	P(Word|Not Sarcastic)
Natalie	0.033	0.036
Usman	0.033	0.036
soooo	0.033	0.018
totally	0.033	0.018
happy	0.016	0.036

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Natalie told Usman she was soooo totally happy for him.

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Usman had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Usman was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Usman she was soooo totally happy for him.	?

- Given all of this information, how should we classify the test sentence *s*?
  - $c' = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i \in T} P(w_i|c)$
  - P(Sarcastic)*P(s|Sarcastic) = 0.5 * 0.033 * 0.033 * 0.033 * 0.033 * 0.016 = $9.487 * 10^{-9}$

Word	P(Word|Sarcastic)	P(Word|Not Sarcastic)
Natalie	0.033	0.036
Usman	0.033	0.036
soooo	0.033	0.018
totally	0.033	0.018
happy	0.016	0.036

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Natalie told Usman she was soooo totally happy for him.

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Usman had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Usman was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Usman she was soooo totally happy for him.	?

- Given all of this information, how should we classify the test sentence *s*?
  - $c' = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i \in T} P(w_i|c)$
  - P(Sarcastic)*P(*s*|Sarcastic) = 0.5 * 0.033 * 0.033 * 0.033 * 0.033 * 0.016 = $9.487 * 10^{-9}$
  - P(Not Sarcastic)*P(*s*|Not Sarcastic) = 0.5 * 0.036 * 0.036 * 0.018 * 0.018 * 0.036 = $7.558 * 10^{-9}$

Word	P(Word|Sarcastic)	P(Word|Not Sarcastic)
Natalie	0.033	0.036
Usman	0.033	0.036
soooo	0.033	0.018
totally	0.033	0.018
happy	0.016	0.036

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Natalie told Usman she was soooo totally happy for him.

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Usman had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Usman was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Usman she was soooo totally happy for him.	?

- Given all of this information, how should we classify the test sentence *s*?
  - $c' = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i \in T} P(w_i|c)$
  - **P(Sarcastic)*P(s|Sarcastic) = 0.5 * 0.033 * 0.033 * 0.033 * 0.033 * 0.016 = 9.487 * 10^{-9}**
  - P(Not Sarcastic)*P(*s*|Not Sarcastic) = 0.5 * 0.036 * 0.036 * 0.018 * 0.018 * 0.036 = 7.558 * 10^{-9}

Word	P(Word|Sarcastic)	P(Word|Not Sarcastic)
Natalie	0.033	0.036
Usman	0.033	0.036
soooo	0.033	0.018
totally	0.033	0.018
happy	0.016	0.036

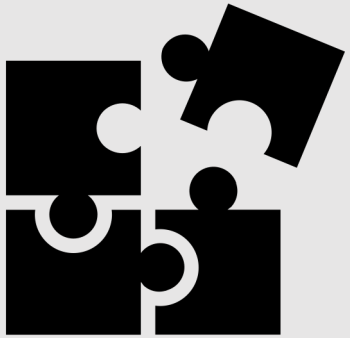
P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Natalie told Usman she was soooo totally happy for him.

Sarcastic

Optimizing for Specific Tasks

- Standard Naïve Bayes text classification (such as that in the previous example) can work well for a variety of tasks
- However, often there are also task-specific ways to improve performance for a particular task



Optimizing for Specific Tasks

- For some tasks, whether or not a word occurs tends to matter more than its frequency
  - Rather than include frequency counts, just use binary values indicating whether each word occurs in the data
- Performance on many tasks is also heavily influenced by the presence of **negation**

The students did not like having a surprise midterm.

Handling Negation

Negation alters the inferences drawn from a statement

- The students did like having a surprise midterm.
  - Let's make them all surprises from now on!
- The students did not like having a surprise midterm.
  - Let's schedule the midterms in advance.

Negation can change the correct class in tasks like sentiment analysis.

- I like surprise midterms. 😀
- I do not like surprise midterms. 😢

Handling Negation

- Simple Baseline:
  - During text normalization, add the prefix "NOT_" to every word after a token of logical negation (n't, not, no, never) until the next punctuation mark
    - I do not like surprise midterms. → I do not NOT_like NOT_surprise NOT_midterms.
- Thus, we have new "words" like *NOT_like* that will (hopefully!) occur more often in negative text and act as cues for negative sentiment, and new words like *NOT_unhappy* that will (again, hopefully!) occur more often in positive text and act as cues for positive sentiment

What if we don't have enough labeled training data to train an accurate Naïve Bayes classifier for a given task?

- For some tasks, we can derive alternate/additional features (not word counts) from external **lexicons**
- **Lexicons** generally contain annotated characteristics (e.g., sentiment labels) for a list of words
- For sentiment analysis:
  - Linguistic Inquiry and Word Count (http://liwc.wpengine.com/)
  - Opinion Lexicon (https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#lexicon)
  - MPQA Subjectivity Lexicon (https://mpqa.cs.pitt.edu/lexicons/subj_lexicon/)

What does a lexicon look like?

It varies depending on which lexicon you're using!

MPQA Lexicon:

- type=strongsubj len=1 word1=love pos1=noun stemmed1=n priorpolarity=positive
  - a. type - either strongsubj or weaksubj
  - b. len - length of the clue in words
  - c. word1 - token or stem of the clue
  - d. pos1 - part of speech of the clue, may be anypos (any part of speech)
  - e. stemmed1 - y (yes) or n (no)
  - f. priorpolarity - positive, negative, both, neutral

How are lexicons incorporated in Naïve Bayes classifiers?

- Many different ways, depending on the application
- A few strategies:
  - Add a feature that is counted whenever a word from the lexicon occurs
    - InMPQA=1
  - Add several features corresponding to different labels in the lexicon
    - IsStronglySubjective=1
    - IsPositive=0

These strategies will likely differ depending on data sparsity.

Large dataset:

- Using many features will work better than just using a few binary features (allows for the classifier to learn more complex ways to discriminate between classes)

Small dataset:

- Using a smaller number of more general features may work better (allows for the classifier to learn meaningful differences, rather than making predictions based on one or two occurrences of a given feature)

Summary: Naïve Bayes Essentials

- **Naïve Bayes** is a **probabilistic classification algorithm** that learns to make predictions based on **labeled training data**
- When making predictions, a classifier takes a test observation, extracts a set of features from it, and assigns a label to the observation based on similarities between its feature values and those of observations in the training dataset
- Naïve Bayes is a **supervised classification algorithm**
- **Multinomial Naïve Bayes** assumes that there is a discrete set of possible classes for the data
- Naïve Bayes is "naïve" because it makes the simplifying assumption that **all features are independent of one another**
- Naïve Bayes classifiers generally use **bag of words** features, but may use other features (e.g., those from external **lexicons**) depending on the task

What are some other ways we can use Naïve Bayes?

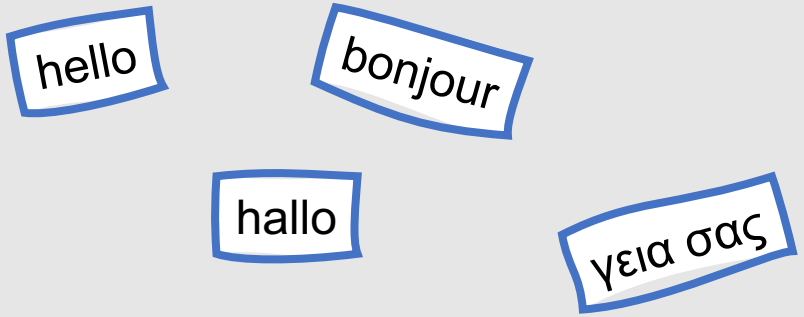
- We saw previously that we can extend our feature sets to incorporate information from external lexicons …what else can we do?
  - Use likely sets or groups of words as features
  - Use non-linguistic features
  - Use character n-grams

When would these extensions be useful?

- Spam detection
  - Contains_OneHundredPercentGuaranteed
    - *You are one hundred percent guaranteed to love our gluten-free celery water!*
  - Matches_Regex_OneMillion
    - *Call the number below to claim your $1,000,000 prize!*
  - Contains_UnbalancedHTMLHeadTags
    - (Non-linguistic)

When would these extensions be useful?

- Language identification
  - th → English
  - te → French
  - ei → German
  - ησ → Greek



Naïve Bayes can also be viewed as a language model.

Use only individual word features (unigrams)

Use all words in the text (not a subset)

Don't remove stop words or unknown words

This means that the model learned for each class is a class-specific unigram language model

- Letting S be the list of all tokens in a sentence:
  - $P(S|c) = \prod_{i \in S} P(w_i|c)$

This means that not only can we get likelihoods for individual words belonging to a class …we can also get likelihoods for entire sentences.

Computing Sentence Likelihood

Training	
Document	**Class**
Natalie was soooo thrilled that Usman had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Usman was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Usman she was soooo totally happy for him.	?

Word	P(Word|Sarcastic)	P(Word|Not Sarcastic)
Natalie	0.033	0.036
Usman	0.033	0.036
soooo	0.033	0.018
totally	0.033	0.018
happy	0.016	0.036

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Natalie told Usman she was soooo totally happy for him.

Computing Sentence Likelihood

Training	
Document	**Class**
Natalie was soooo thrilled that Usman had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Usman was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Usman she was soooo totally happy for him.	?

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Natalie told Usman she was soooo totally happy for him.

Word	P(Word|Sarcastic)	P(Word|Not Sarcastic)
Natalie	0.033	0.036
Usman	0.033	0.036
soooo	0.033	0.018
totally	0.033	0.018
happy	0.016	0.036
told	$\frac{0+1}{27+34}$	$\frac{0+1}{21+34}$
she	$\frac{1+1}{27+34}$	$\frac{0+1}{21+34}$
was	$\frac{2+1}{27+34}$	$\frac{2+1}{21+34}$
for	$\frac{0+1}{27+34}$	$\frac{1+1}{21+34}$
him	$\frac{0+1}{27+34}$	$\frac{0+1}{21+34}$

Computing Sentence Likelihood

Training	
Document	**Class**
Natalie was soooo thrilled that Usman had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Usman was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Usman she was soooo totally happy for him.	?

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Natalie told Usman she was soooo totally happy for him.

Word	P(Word|Sarcastic)	P(Word|Not Sarcastic)
Natalie	0.033	0.036
Usman	0.033	0.036
soooo	0.033	0.018
totally	0.033	0.018
happy	0.016	0.036
told	$\frac{0+1}{27+34} = 0.016$	$\frac{0+1}{21+34} = 0.018$
she	$\frac{1+1}{27+34} = 0.033$	$\frac{0+1}{21+34} = 0.018$
was	$\frac{2+1}{27+34} = 0.049$	$\frac{2+1}{21+34} = 0.055$
for	$\frac{0+1}{27+34} = 0.016$	$\frac{1+1}{21+34} = 0.036$
him	$\frac{0+1}{27+34} = 0.016$	$\frac{0+1}{21+34} = 0.018$

Computing Sentence Likelihood

Training	
Document	**Class**
Natalie was soooo thrilled that Usman had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Usman was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Usman she was soooo totally happy for him.	?

Word	P(Word|Sarcastic)	P(Word|Not Sarcastic)
Natalie	0.033	0.036
Usman	0.033	0.036
soooo	0.033	0.018
totally	0.033	0.018
happy	0.016	0.036
told	0.016	0.018
she	0.033	0.018
was	0.049	0.055
for	0.016	0.036
him	0.016	0.018

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Natalie told Usman she was soooo totally happy for him.

Computing Sentence Likelihood

Training	
Document	**Class**
Natalie was soooo thrilled that Usman had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Usman was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Usman she was soooo totally happy for him.	?

Word	P(Word|Sarcastic)	P(Word|Not Sarcastic)
Natalie	0.033	0.036
Usman	0.033	0.036
soooo	0.033	0.018
totally	0.033	0.018
happy	0.016	0.036
told	0.016	0.018
she	0.033	0.018
was	0.049	0.055
for	0.016	0.036
him	0.016	0.018

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Natalie told Usman she was soooo totally happy for him.

Computing Sentence Likelihood

Natalie told Usman she was soooo totally happy for him.

$$P(S|c) = \prod_{i \in S} P(w_i|c)$$

P("Natalie told Usman she was soooo totally happy for him"|Sarcastic) = 0.033 * 0.016 * 0.033 * 0.033 * 0.049 * 0.033 * 0.033 * 0.016 * 0.016 * 0.016 = **$1.26 * 10^{-16}$**

Word	P(Word|Sarcastic)	P(Word|Not Sarcastic)
Natalie	0.033	0.036
Usman	0.033	0.036
soooo	0.033	0.018
totally	0.033	0.018
happy	0.016	0.036
told	0.016	0.018
she	0.033	0.018
was	0.049	0.055
for	0.016	0.036
him	0.016	0.018

Computing Sentence Likelihood

Natalie told Usman she was soooo totally happy for him.

$$P(S|c) = \prod_{i \in S} P(w_i|c)$$

P("Natalie told Usman she was soooo totally happy for him"|Sarcastic) = 0.033 * 0.016 * 0.033 * 0.033 * 0.049 * 0.033 * 0.033 * 0.016 * 0.016 * 0.016 = **$1.26 * 10^{-16}$**

P("Natalie told Usman she was soooo totally happy for him"|Not Sarcastic) = 0.036 * 0.018 * 0.036 * 0.018 * 0.055 * 0.018 * 0.018 * 0.036 * 0.036 * 0.018 = **$1.75 * 10^{-16}$**

Word	P(Word|Sarcastic)	P(Word|Not Sarcastic)
Natalie	0.033	0.036
Usman	0.033	0.036
soooo	0.033	0.018
totally	0.033	0.018
happy	0.016	0.036
told	0.016	0.018
she	0.033	0.018
was	0.049	0.055
for	0.016	0.036
him	0.016	0.018

Computing Sentence Likelihood

Natalie told Usman she was soooo totally happy for him.

$$P(S|c) = \prod_{i \in S} P(w_i|c)$$

P("Natalie told Usman she was soooo totally happy for him"|Sarcastic) = 0.033 * 0.016 * 0.033 * 0.033 * 0.049 * 0.033 * 0.033 * 0.016 * 0.016 * 0.016 = **$1.26 * 10^{-16}$**

P("Natalie told Usman she was soooo totally happy for him"|Not Sarcastic) = 0.036 * 0.018 * 0.036 * 0.018 * 0.055 * 0.018 * 0.018 * 0.036 * 0.036 * 0.018 = **$1.75 * 10^{-16}$**

Word	P(Word|Sarcastic)	P(Word|Not Sarcastic)
Natalie	0.033	0.036
Usman	0.033	0.036
soooo	0.033	0.018
totally	0.033	0.018
happy	0.016	0.036
told	0.016	0.018
she	0.033	0.018
was	0.049	0.055
for	0.016	0.036
him	0.016	0.018

Slightly higher likelihood of the sentence being **not sarcastic**, but this isn't the full Naïve Bayes model …once we multiply in the prior probability, we might make a different classification decision.

Computing Sentence Likelihood

Natalie told Usman she was soooo totally happy for him.

$$P(S|c) = \prod_{i \in S} P(w_i|c)$$

P("Natalie told Usman she was soooo totally happy for him"|Sarcastic) = 0.033 * 0.016 * 0.033 * 0.033 * 0.049 * 0.033 * 0.033 * 0.016 * 0.016 * 0.016 = **$1.26 * 10^{-16}$**

P("Natalie told Usman she was soooo totally happy for him"|Not Sarcastic) = 0.036 * 0.018 * 0.036 * 0.018 * 0.055 * 0.018 * 0.018 * 0.036 * 0.036 * 0.018 = **$1.75 * 10^{-16}$**

Word	P(Word|Sarcastic)	P(Word|Not Sarcastic)
Natalie	0.033	0.036
Usman	0.033	0.036
soooo	0.033	0.018
totally	0.033	0.018
happy	0.016	0.036
to		
s		
w		
for	0.016	0.036
him	0.016	0.018

Although, note that in this case we do not! 🙃 This is a good example of how stop words can be problematic in text classification, particularly with extremely tiny datasets.

Slightly higher likelihood of the sentence being **not sarcastic**, but this isn't the full Naïve Bayes model …once we multiply in the prior probability, we might make a different classification decision.

Moving on to evaluation….

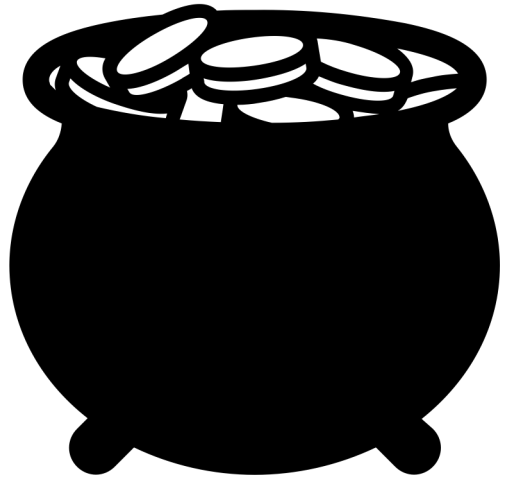
How do we determine how well our classification models work?

When can we say that our performance is good?

When can we say that our model is better than others?

Gold Labels

- Before determining anything, we need some sort of basis upon which to make our comparisons
  - *Is "Sarcastic" the correct label for "Natalie told Usman she was soooo totally happy for him." ?*
- We can acquire **gold standard labels** from human annotators



Does it matter who our annotators are?

- Depends on the task
- For complex tasks, you may want to recruit experts in the desired subject area
  - Rating translation quality
  - Labeling pedagogical strategies in teacher-student interactions
- For simpler tasks, you can probably recruit non-experts
  - Deciding whether text is sarcastic or non-sarcastic
  - Deciding whether a specified event takes place before or after a second event
- Common sources of annotators:
  - Amazon Mechanical Turk: https://www.mturk.com/
  - Figure Eight: https://www.figure-eight.com/
  - Friends and family

Contingency Tables

- Once we have our gold standard labels (either from an existing dataset, or after collecting our own), we can begin comparing **predicted** and **actual** labels
- To do this, we can create a **contingency table**
  - Often also referred to as a **confusion matrix**

Contingency Tables

- In a contingency table, each cell labels a set of possible outcomes
- These outcomes are generally referred to as:
  - **True positives**
    - Predicted true and actually true
  - **False positives**
    - Predicted true and actually false
  - **True negatives**
    - Predicted false and actually false
  - **False negatives**
    - Predicted false and actually true

Predicted \ Actual		
	True Positive (TP)	False Positive (FP)
	False Negative (FN)	True Negative (TN)

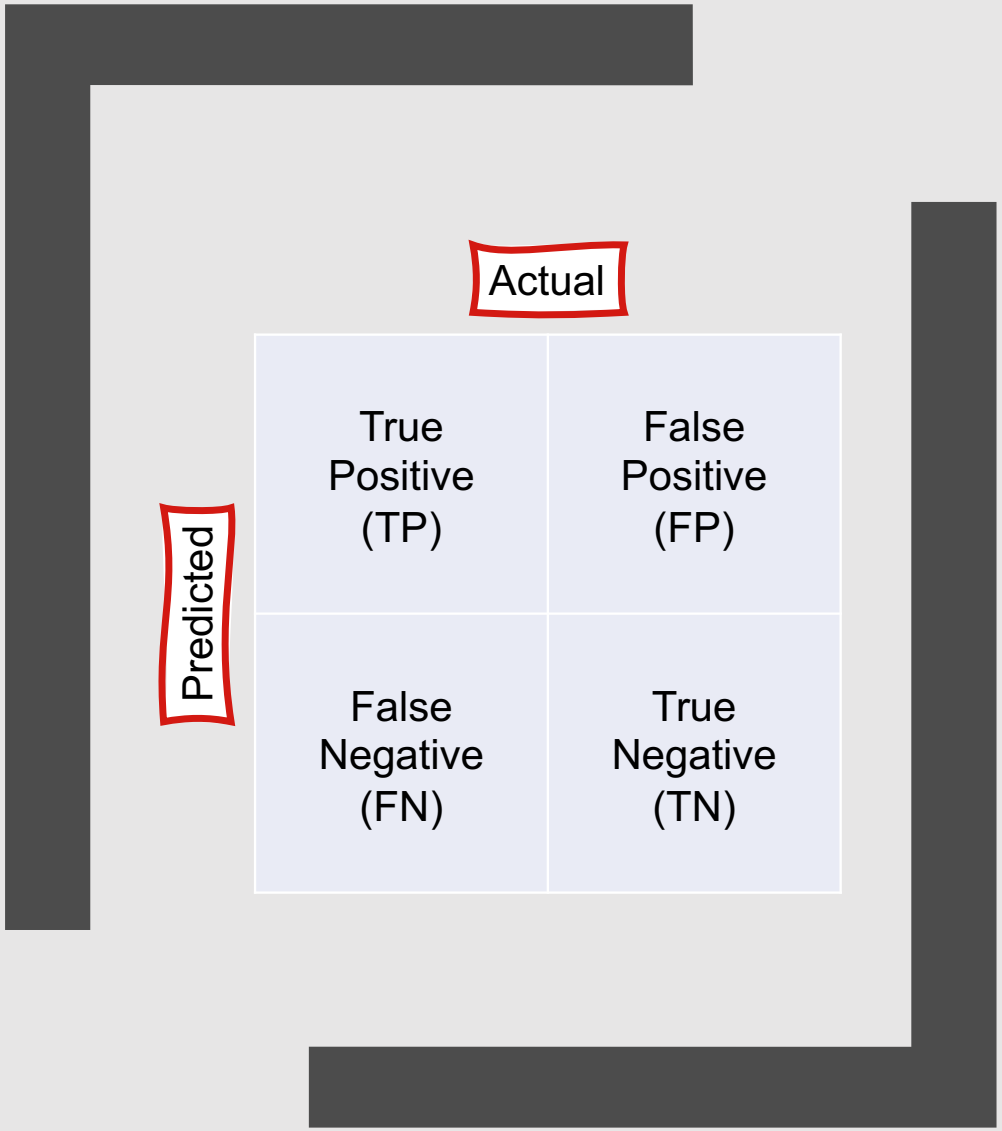
We can compute a variety of metrics using contingency tables.

Precision

Recall

F-Measure

Accuracy



Accuracy

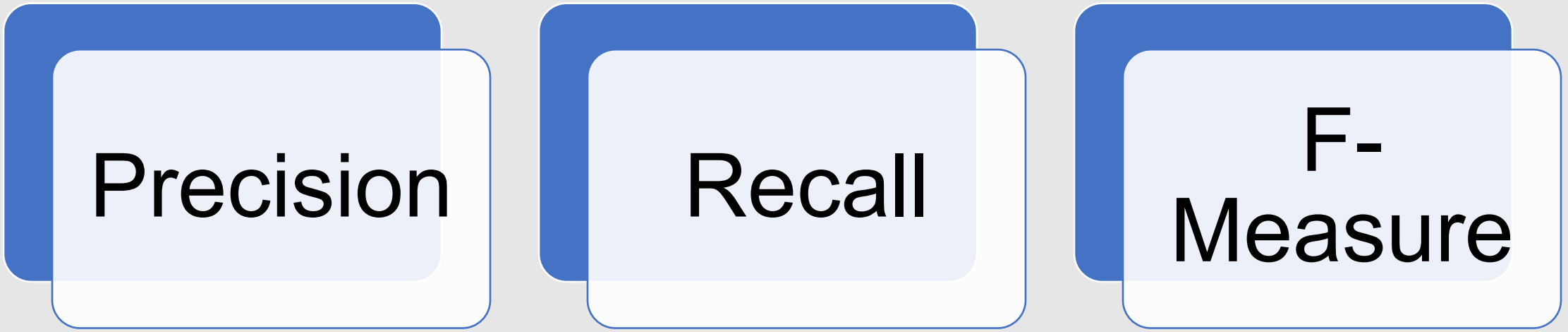
- **Accuracy:** The percentage of all observations that the system labels correctly
- Accuracy = $\frac{tp+tn}{tp+fp+tn+fn}$

Why not just use accuracy and be done with it?

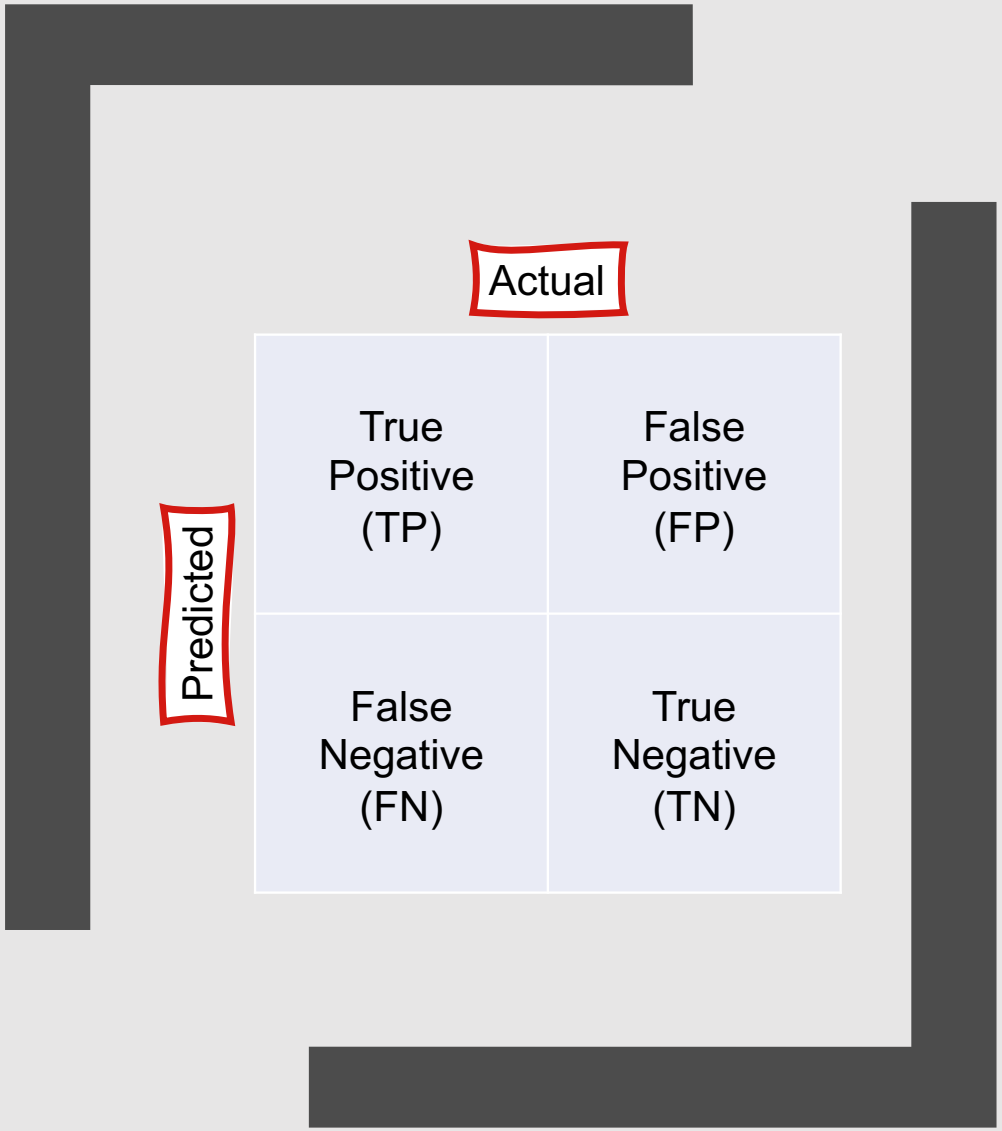
- This metric can be problematic when dealing with unbalanced datasets!
  - Imagine that we have 999,900 non-sarcastic sentences, and 100 sarcastic sentences
  - Our classifier might decide to just predict “non-sarcastic” every time to maximize its expected accuracy
    - 999900/1000000 = 99.99% accuracy
  - However, such a classifier would be useless …it would never tell us when a sentence *is* sarcastic

Thus, accuracy is a poor metric when the goal is to discover members of a less-frequent class.

- Doing so is a very common situation
  - Detecting medical issues
  - Detecting papers dealing with a certain topic
  - Detecting spam

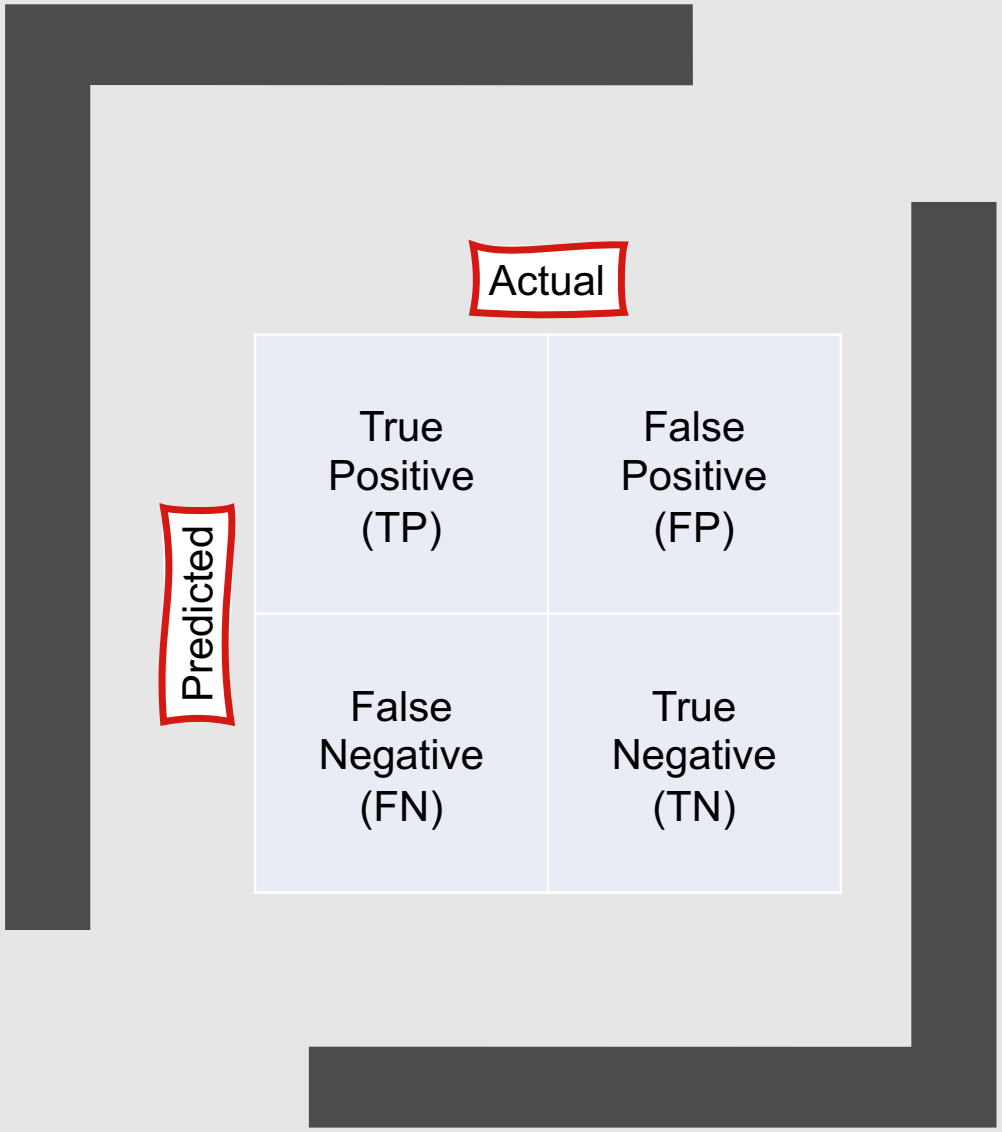


What are some alternatives that can focus on specific classes?



Precision

- **Precision:** Of the instances that the system predicted to be positive, what percentage actually are?
- Precision = $\frac{tp}{tp+fp}$



Recall

- **Recall:** Of the instances that actually are positive, what percentage did the system predict to be?
- Recall = $\frac{tp}{tp+fn}$

Precision and recall both emphasize a specific class of interest.

- Positive class can be whichever class you're interested in
  - **Sarcastic** or Non-Sarcastic
  - Positive or **Negative**
- Thus, in our problematic example case, precision and recall for the positive (sarcastic) case would both be 0
  - Precision = 0/(0+0) = 0
  - Recall = 0/(0+100) = 0

	Actual	
Predicted	TP: 0	FP: 0
	FN: 100	TN: 999,900

Which is more useful: Precision or recall?

- Depends on the task!
- If it's more important to maximize the chances that all predicted true values really are true, at the expense of predicting some of the true values as false, focus on precision
- If it's more important to maximize the chances that all true values are predicted to be true, at the expense of predicting some false values to be true as well, focus on recall

What if both are important?

- **F-measure** combines aspects of both **precision** and **recall** by computing their weighted harmonic mean
  - $F_\beta = \frac{(\beta^2+1)PR}{\beta^2 P+R}$
- The β parameter weights the importance of precision and recall, depending on the needs of the application
  - β > 1 means that recall is more important
  - β < 1 means that precision is more important
  - β = 1 means that the two are equally important

F-Measure

- Most commonly, researchers set β = 1 to weight precision and recall equally
- In this case, the metric is generally referred to as F_1
  - $F_1 = \frac{(1^2+1)PR}{1^2P+R} = \frac{2PR}{P+R}$
- Although F-measure combines both precision and recall, it tends to be conservative; thus, the lower of the two numbers will factor more heavily into the final score

Example: Precision, Recall, and F_1

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my train broke down in the middle of a tunnel.	Sarcastic	
I am absolutely thrilled to announce that my new paper on F-measure was just published!	Not Sarcastic	
Oh no I am soooo sad that my 8 a.m. class is cancelled tomorrow.	Sarcastic	
Oh yay I spilled coffee on myself right before the faculty meeting!!!	Sarcastic	
Oh yay the CS office has a fancy new coffee machine!!!	Not Sarcastic	
How do I get from UIC to Willis Tower?	Not Sarcastic	
I would love nothing better than to make 200 more slides for class tonight.	Sarcastic	

Example: Precision, Recall, and F_1

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my train broke down in the middle of a tunnel.	Sarcastic	Not Sarcastic
I am absolutely thrilled to announce that my new paper on F-measure was just published!	Not Sarcastic	Not Sarcastic
Oh no I am soooo sad that my 8 a.m. class is cancelled tomorrow.	Sarcastic	Sarcastic
Oh yay I spilled coffee on myself right before the faculty meeting!!!	Sarcastic	Not Sarcastic
Oh yay the CS office has a fancy new coffee machine!!!	Not Sarcastic	Sarcastic
How do I get from UIC to Willis Tower?	Not Sarcastic	Not Sarcastic
I would love nothing better than to make 200 more slides for class tonight.	Sarcastic	Not Sarcastic

Example: Precision, Recall, and F_1

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my train broke down in the middle of a tunnel.	Sarcastic	Not Sarcastic
I am absolutely thrilled to announce that my new paper on F-measure was just published!	Not Sarcastic	Not Sarcastic
Oh no I am soooo sad that my 8 a.m. class is cancelled tomorrow.	Sarcastic	Sarcastic
Oh yay I spilled coffee on myself right before the faculty meeting!!!	Sarcastic	Not Sarcastic
Oh yay the CS office has a fancy new coffee machine!!!	Not Sarcastic	Sarcastic
How do I get from UIC to Willis Tower?	Not Sarcastic	Not Sarcastic
I would love nothing better than to make 200 more slides for class tonight.	Sarcastic	Not Sarcastic

	Actual	
Predicted	TP: ?	FP: ?
	FN: ?	TN: ?

Positive Class: Sarcastic

Example: Precision, Recall, and F_1

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my train broke down in the middle of a tunnel.	Sarcastic	Not Sarcastic
I am absolutely thrilled to announce that my new paper on F-measure was just published!	Not Sarcastic	Not Sarcastic
Oh no I am soooo sad that my 8 a.m. class is cancelled tomorrow.	Sarcastic	Sarcastic
Oh yay I spilled coffee on myself right before the faculty meeting!!!	Sarcastic	Not Sarcastic
Oh yay the CS office has a fancy new coffee machine!!!	Not Sarcastic	Sarcastic
How do I get from UIC to Willis Tower?	Not Sarcastic	Not Sarcastic
I would love nothing better than to make 200 more slides for class tonight.	Sarcastic	Not Sarcastic

	Actual	
Predicted	TP: 1	FP: ?
	FN: ?	TN: ?

Positive Class: Sarcastic

Example: Precision, Recall, and F_1

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my train broke down in the middle of a tunnel.	Sarcastic	Not Sarcastic
I am absolutely thrilled to announce that my new paper on F-measure was just published!	Not Sarcastic	Not Sarcastic
Oh no I am soooo sad that my 8 a.m. class is cancelled tomorrow.	Sarcastic	Sarcastic
Oh yay I spilled coffee on myself right before the faculty meeting!!!	Sarcastic	Not Sarcastic
Oh yay the CS office has a fancy new coffee machine!!!	Not Sarcastic	Sarcastic
How do I get from UIC to Willis Tower?	Not Sarcastic	Not Sarcastic
I would love nothing better than to make 200 more slides for class tonight.	Sarcastic	Not Sarcastic

	Actual	
Predicted	TP: 1	FP: 1
	FN: ?	TN: ?

Positive Class: Sarcastic

Example: Precision, Recall, and F_1

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my train broke down in the middle of a tunnel.	Sarcastic	Not Sarcastic
I am absolutely thrilled to announce that my new paper on F-measure was just published!	Not Sarcastic	Not Sarcastic
Oh no I am soooo sad that my 8 a.m. class is cancelled tomorrow.	Sarcastic	Sarcastic
Oh yay I spilled coffee on myself right before the faculty meeting!!!	Sarcastic	Not Sarcastic
Oh yay the CS office has a fancy new coffee machine!!!	Not Sarcastic	Sarcastic
How do I get from UIC to Willis Tower?	Not Sarcastic	Not Sarcastic
I would love nothing better than to make 200 more slides for class tonight.	Sarcastic	Not Sarcastic

	Actual	
Predicted	TP: 1	FP: 1
	FN: 3	TN: ?

Positive Class: Sarcastic

Example: Precision, Recall, and F_1

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my train broke down in the middle of a tunnel.	Sarcastic	Not Sarcastic
I am absolutely thrilled to announce that my new paper on F-measure was just published!	Not Sarcastic	Not Sarcastic
Oh no I am soooo sad that my 8 a.m. class is cancelled tomorrow.	Sarcastic	Sarcastic
Oh yay I spilled coffee on myself right before the faculty meeting!!!	Sarcastic	Not Sarcastic
Oh yay the CS office has a fancy new coffee machine!!!	Not Sarcastic	Sarcastic
How do I get from UIC to Willis Tower?	Not Sarcastic	Not Sarcastic
I would love nothing better than to make 200 more slides for class tonight.	Sarcastic	Not Sarcastic

	Actual	
Predicted	TP: 1	FP: 1
	FN: 3	TN: 2

Positive Class: Sarcastic

Example: Precision, Recall, and F_1

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my train broke down in the middle of a tunnel.	Sarcastic	Not Sarcastic
I am absolutely thrilled to announce that my new paper on F-measure was just published!	Not Sarcastic	Not Sarcastic
Oh no I am soooo sad that my 8 a.m. class is cancelled tomorrow.	Sarcastic	Sarcastic
Oh yay I spilled coffee on myself right before the faculty meeting!!!	Sarcastic	Not Sarcastic
Oh yay the CS office has a fancy new coffee machine!!!	Not Sarcastic	Sarcastic
How do I get from UIC to Willis Tower?	Not Sarcastic	Not Sarcastic
I would love nothing better than to make 200 more slides for class tonight.	Sarcastic	Not Sarcastic

Positive Class: Sarcastic

	Actual	
Predicted	TP: 1	FP: 1
	FN: 3	TN: 2

$$\text{Precision} = \frac{tp}{tp+fp} = \frac{1}{1+1} = 0.5$$

Example: Precision, Recall, and F_1

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my train broke down in the middle of a tunnel.	Sarcastic	Not Sarcastic
I am absolutely thrilled to announce that my new paper on F-measure was just published!	Not Sarcastic	Not Sarcastic
Oh no I am soooo sad that my 8 a.m. class is cancelled tomorrow.	Sarcastic	Sarcastic
Oh yay I spilled coffee on myself right before the faculty meeting!!!	Sarcastic	Not Sarcastic
Oh yay the CS office has a fancy new coffee machine!!!	Not Sarcastic	Sarcastic
How do I get from UIC to Willis Tower?	Not Sarcastic	Not Sarcastic
I would love nothing better than to make 200 more slides for class tonight.	Sarcastic	Not Sarcastic

Predicted \ Actual		
	TP: 1	FP: 1
	FN: 3	TN: 2

Positive Class: Sarcastic

$$\text{Precision} = \frac{tp}{tp+fp} = \frac{1}{1+1} = 0.5$$

$$\text{Recall} = \frac{tp}{tp+fn} = \frac{1}{1+3} = 0.25$$

Example: Precision, Recall, and F_1

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my train broke down in the middle of a tunnel.	Sarcastic	Not Sarcastic
I am absolutely thrilled to announce that my new paper on F-measure was just published!	Not Sarcastic	Not Sarcastic
Oh no I am soooo sad that my 8 a.m. class is cancelled tomorrow.	Sarcastic	Sarcastic
Oh yay I spilled coffee on myself right before the faculty meeting!!!	Sarcastic	Not Sarcastic
Oh yay the CS office has a fancy new coffee machine!!!	Not Sarcastic	Sarcastic
How do I get from UIC to Willis Tower?	Not Sarcastic	Not Sarcastic
I would love nothing better than to make 200 more slides for class tonight.	Sarcastic	Not Sarcastic

	Actual	
Predicted	TP: 1	FP: 1
	FN: 3	TN: 2

Positive Class: Sarcastic

Precision = 0.5

Recall = 0.25

$$F_1 = \frac{(1^2+1)PR}{1^2P+R} = \frac{2PR}{P+R} = \frac{2*0.5*0.25}{0.5+0.25} = 0.333$$

Example: Same, but what if the positive class is Not Sarcastic?

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my train broke down in the middle of a tunnel.	Sarcastic	Not Sarcastic
I am absolutely thrilled to announce that my new paper on F-measure was just published!	Not Sarcastic	Not Sarcastic
Oh no I am soooo sad that my 8 a.m. class is cancelled tomorrow.	Sarcastic	Sarcastic
Oh yay I spilled coffee on myself right before the faculty meeting!!!	Sarcastic	Not Sarcastic
Oh yay the CS office has a fancy new coffee machine!!!	Not Sarcastic	Sarcastic
How do I get from UIC to Willis Tower?	Not Sarcastic	Not Sarcastic
I would love nothing better than to make 200 more slides for class tonight.	Sarcastic	Not Sarcastic

	Actual	
Predicted	TP: ?	FP: ?
	FN: ?	TN: ?

Positive Class: Not Sarcastic

Precision = ?

Recall = ?

$$F_1 = \frac{(1^2+1)PR}{1^2P+R} = \frac{2PR}{P+R} = ?$$

Example: Same, but what if the positive class is Not Sarcastic?

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my train broke down in the middle of a tunnel.	Sarcastic	Not Sarcastic
I am absolutely thrilled to announce that my new paper on F-measure was just published!	Not Sarcastic	Not Sarcastic
Oh no I am soooo sad that my 8 a.m. class is cancelled tomorrow.	Sarcastic	Sarcastic
Oh yay I spilled coffee on myself right before the faculty meeting!!!	Sarcastic	Not Sarcastic
Oh yay the CS office has a fancy new coffee machine!!!	Not Sarcastic	Sarcastic
How do I get from UIC to Willis Tower?	Not Sarcastic	Not Sarcastic
I would love nothing better than to make 200 more slides for class tonight.	Sarcastic	Not Sarcastic

	Actual	
Predicted	TP: 2	FP: 3
	FN: 1	TN: 1

Positive Class: Not Sarcastic

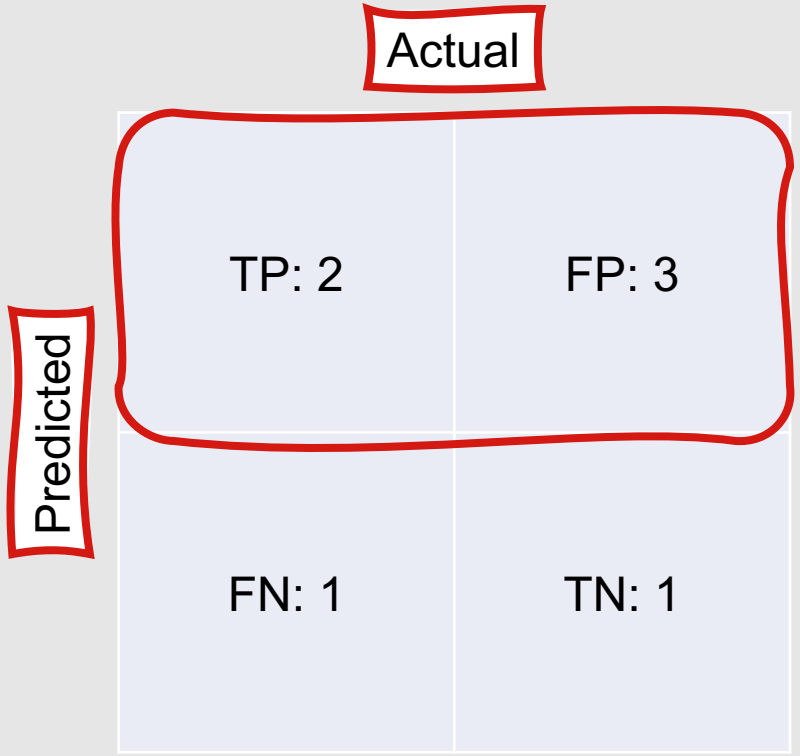
Precision = ?

Recall = ?

$$F_1 = \frac{(1^2+1)PR}{1^2P+R} = \frac{2PR}{P+R} = ?$$

Example: Same, but what if the positive class is Not Sarcastic?

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my train broke down in the middle of a tunnel.	Sarcastic	Not Sarcastic
I am absolutely thrilled to announce that my new paper on F-measure was just published!	Not Sarcastic	Not Sarcastic
Oh no I am soooo sad that my 8 a.m. class is cancelled tomorrow.	Sarcastic	Sarcastic
Oh yay I spilled coffee on myself right before the faculty meeting!!!	Sarcastic	Not Sarcastic
Oh yay the CS office has a fancy new coffee machine!!!	Not Sarcastic	Sarcastic
How do I get from UIC to Willis Tower?	Not Sarcastic	Not Sarcastic
I would love nothing better than to make 200 more slides for class tonight.	Sarcastic	Not Sarcastic



Positive Class: Not Sarcastic

Precision = 0.4

Recall = ?

$$F_1 = \frac{(1^2+1)PR}{1^2P+R} = \frac{2PR}{P+R} = ?$$

Example: Same, but what if the positive class is Not Sarcastic?

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my train broke down in the middle of a tunnel.	Sarcastic	Not Sarcastic
I am absolutely thrilled to announce that my new paper on F-measure was just published!	Not Sarcastic	Not Sarcastic
Oh no I am soooo sad that my 8 a.m. class is cancelled tomorrow.	Sarcastic	Sarcastic
Oh yay I spilled coffee on myself right before the faculty meeting!!!	Sarcastic	Not Sarcastic
Oh yay the CS office has a fancy new coffee machine!!!	Not Sarcastic	Sarcastic
How do I get from UIC to Willis Tower?	Not Sarcastic	Not Sarcastic
I would love nothing better than to make 200 more slides for class tonight.	Sarcastic	Not Sarcastic

	Actual	
Predicted	TP: 2	FP: 3
	FN: 1	TN: 1

Positive Class: Not Sarcastic

Precision = 0.4

Recall = 0.667

$$F_1 = \frac{(1^2+1)PR}{1^2P+R} = \frac{2PR}{P+R} = ?$$

Example: Same, but what if the positive class is Not Sarcastic?

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my train broke down in the middle of a tunnel.	Sarcastic	Not Sarcastic
I am absolutely thrilled to announce that my new paper on F-measure was just published!	Not Sarcastic	Not Sarcastic
Oh no I am soooo sad that my 8 a.m. class is cancelled tomorrow.	Sarcastic	Sarcastic
Oh yay I spilled coffee on myself right before the faculty meeting!!!	Sarcastic	Not Sarcastic
Oh yay the CS office has a fancy new coffee machine!!!	Not Sarcastic	Sarcastic
How do I get from UIC to Willis Tower?	Not Sarcastic	Not Sarcastic
I would love nothing better than to make 200 more slides for class tonight.	Sarcastic	Not Sarcastic

	Actual	
Predicted	TP: 2	FP: 3
	FN: 1	TN: 1

Positive Class: Not Sarcastic

Precision = 0.4

Recall = 0.667

$$F_1 = \frac{(1^2+1)PR}{1^2P+R} = \frac{2PR}{P+R} = \frac{2*0.4*0.667}{0.4+0.667} = 0.500$$

What if we have more than two classes?

Many NLP classification tasks have more than two classes

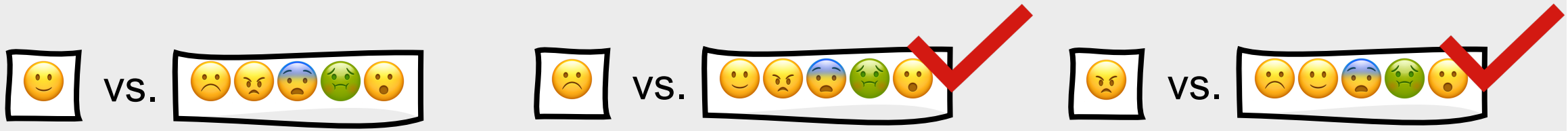
- Sentiment analysis (positive, negative, neutral)
- Part-of-speech tagging (noun, verb, adjective, etc.)
- Emotion detection (happy, sad, angry, surprised, afraid, disgusted)

Multi-class classification tasks can be divided into two groups

- Multi-label classification
- Multinomial classification

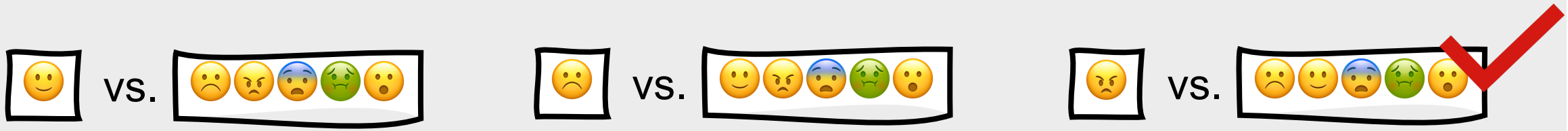
Multi-Label Classification

- Each document can be assigned more than one label
- How do we do this?
  - Build separate binary classifiers for each class
    - Positive class vs. every other class
  - Run each classifier on the test document
    - Each classifier makes its decision independently of the other classifiers, therefore allowing multiple labels to be assigned to the document

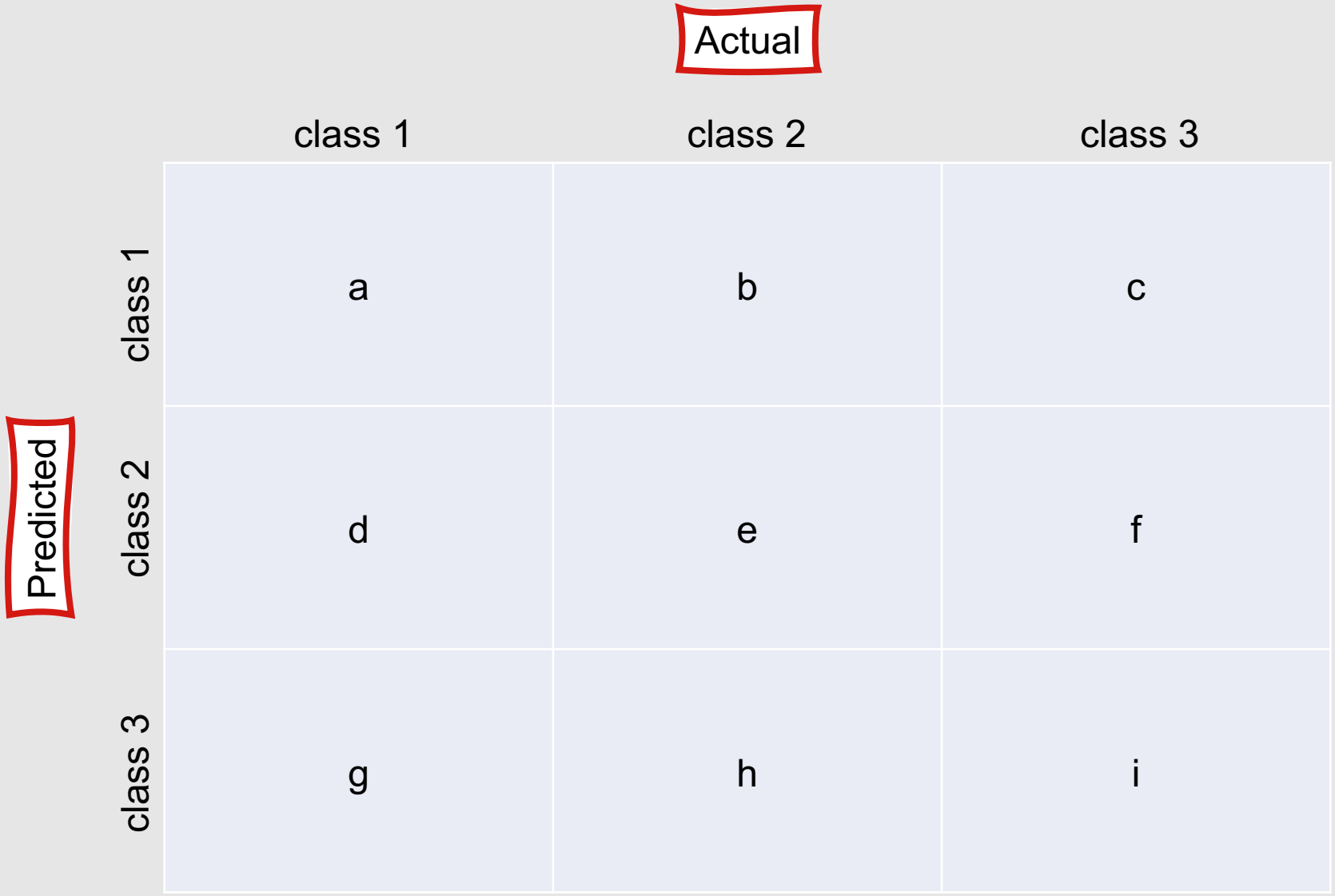


Multinomial Classification

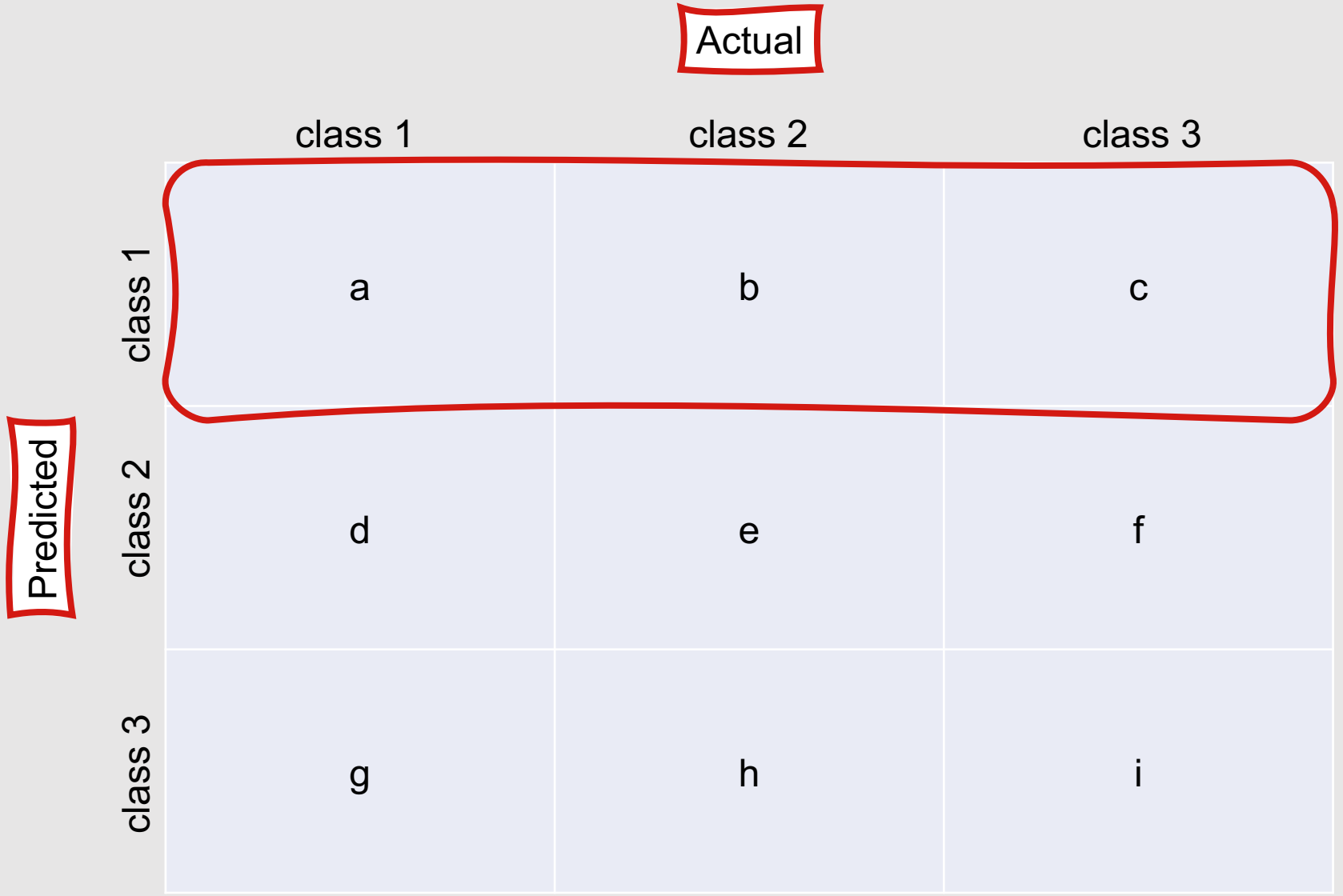
- Each document can only be assigned one label
- How do we do this?
  - Same setup:
    - Build separate binary classifiers for each class
    - Run each classifier on the test document
  - Different outcome:
    - Choose the label from the classifier with the highest score



Multi-Class Contingency Matrix



Multi-Class Precision



$$\text{Precision} = \frac{a}{a+b+c}$$

Multi-Class Recall

$$\text{Precision} = \frac{a}{a+b+c}$$

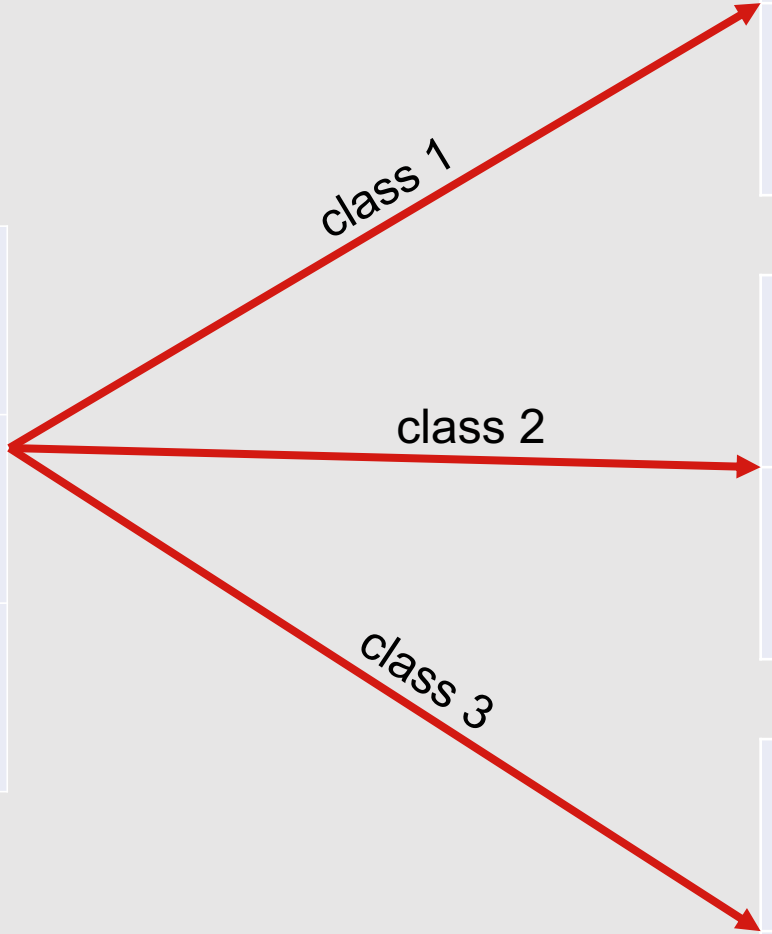
$$\text{Recall} = \frac{a}{a+d+g}$$

Macroaveraging and Microaveraging

- We can check the system's **overall performance** in multi-class classification settings by combining all of the precision values (or all of the recall values) in two ways:
  - **Macroaveraging**
  - **Microaveraging**
- **Macroaveraging:** Compute the performance for each class, and then average over all classes
- **Microaveraging:** Collect decisions for all classes into a single contingency table, and compute precision and recall from that table

Macroaveraging

Predicted \ Actual	class 1	class 2	class 3
class 1	a	b	c
class 2	d	e	f
class 3	g	h	i



TP: a	FP: b+c
FN: d+g	TN: e+f+h+i

TP: e	FP: d+f
FN: b+h	TN: a+c+g+i

TP: i	FP: g+h
FN: c+f	TN: a+b+d+e

Macroaveraging

Predicted \ Actual	class 1	class 2	class 3
class 1	a	b	c
class 2	d	e	f
class 3	g	h	i

$$\text{Precision}_{\text{Class1}} = \frac{tp}{tp+fp}$$

TP: a	FP: b+c
FN: d+g	TN: e+f+h+i

class 1

$$\text{Precision}_{\text{Class2}} = \frac{tp}{tp+fp}$$

TP: e	FP: d+f
FN: b+h	TN: a+c+g+i

class 2

class 3

$$\text{Precision}_{\text{Class3}} = \frac{tp}{tp+fp}$$

TP: i	FP: g+h
FN: c+f	TN: a+b+d+e

Macroaveraging

Predicted \ Actual	class 1	class 2	class 3
class 1	a	b	c
class 2	d	e	f
class 3	g	h	i

$$\text{Precision}_{\text{Class1}} = \frac{tp}{tp+fp}$$

TP: a	FP: b+c
FN: d+g	TN: e+f+h+i

class 1

$$\text{Precision}_{\text{Class2}} = \frac{tp}{tp+fp}$$

TP: e	FP: d+f
FN: b+h	TN: a+c+g+i

class 2

class 3

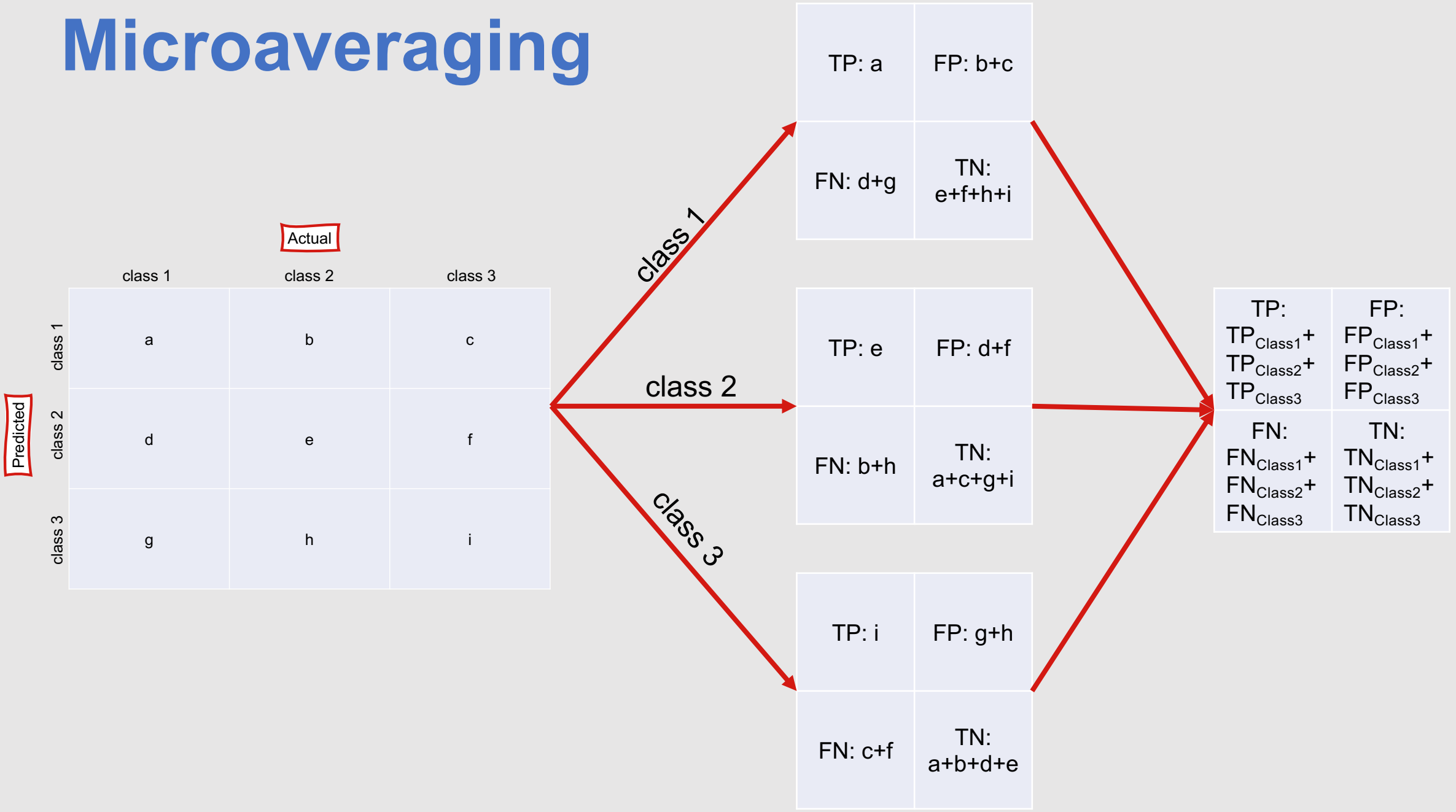
$$\text{Precision}_{\text{Class3}} = \frac{tp}{tp+fp}$$

TP: i	FP: g+h
FN: c+f	TN: a+b+d+e

Macroaveraged Precision

$$= \frac{Precision_{Class1} + Precision_{Class2} + Precision_{Class3}}{3}$$

Microaveraging



What's better: Microaveraging or macroaveraging?

- Depends on the scenario!
- Microaverages tend to be dominated by more frequent classes, since the counts are all pooled together
- Macroaverages tend to be more evenly distributed across classes
- Thus, if performance on all classes is equally important, macroaveraging is probably better; if performance on the most frequent class is more important, microaveraging is probably better

Training, Validation, and Test Sets

Text corpora should generally be divided into three separate subsets (sometimes called **splits** or **folds**):

- **Training:** Used to train the classification model
- **Validation:** Used to check performance while developing the classification model
- **Test:** Used to check performance only after model development is finished

The percentage of data in each fold can vary

- In many cases, researchers like to reserve 75% or more of their corpus for training, and split the remaining data between validation and test

Why is a validation set necessary?

It helps avoid overfitting

- **Overfitting:** Artificially boosting performance on the test set by tweaking training parameters such that they are particularly well-suited for the test set itself

Why is overfitting problematic?

- Models that have been overfit to the test data tend to perform poorly on other (non-test) samples in the same domain
- This means that the finished model cannot generalize easily to real-world scenarios, where the entire test set is not known in advance

What if the entire dataset is pretty small?

In cases where the entire dataset is small, it may be undesirable to reserve an entire fold of data for validation

- Smaller training set (less data from which to learn)
- Smaller test set (less data on which to evaluate)

In these cases, a reasonable alternative is cross-validation

- Randomly split the dataset into *k* folds
- Train on *k*-1 folds and test on the other fold
- Repeat with a different combination of *k*-1 folds and other fold
- Overall, repeat *k* times
- Average the performance across all *k* training/test runs

Cross-Validation

Most commonly, *k*=10 in cross-validation

- Referred to as **10-fold cross-validation**

With really small datasets, *k* may need to be smaller

One problem with cross-validation?

- To avoid overfitting, we can't look at any of the data because it's technically *all* test data!

To avoid this issue, we can:

- Create a fixed training set and test set
- Perform *k*-fold cross-validation on the training set (where it's fine to look at the data) while developing the model
- Evaluate the model on the test set as usual, training on the entire training set

Statistical Significance Testing

- We've trained and evaluated our classification model …how do we know it's better (or worse) than other alternate models?
- We can't necessarily say that Model A is better than Model B purely because its precision/recall/F_1/accuracy is higher!
  - Model A might be performing better than Model B just due to chance
- To confirm our suspicions that Model A really is better, we need to perform **statistical significance testing** to reject the **null hypothesis** that Model A is better than Model B just due to chance

Null Hypothesis

Given observation: Model A performs x% better than Model B

Null Hypothesis: If we had many test sets of the same size as ours, and measured Model A's and Model B's performance on all of them, then on average Model A might accidentally perform x% better than Model B

P-Value

In statistical significance testing, the probability that we'll see equally big performance differences by chance is referred to as the ***p*-value**

If the *p*-value is sufficiently low (generally 0.05 or 0.01), then we can **reject the null hypothesis**

If we reject the null hypothesis, that means that we have identified a **statistically significant difference** between the performance of Model A and Model B

How do we determine our *p*-value?

- Traditional statistical tests (e.g., paired t-test) are often not valid in NLP because they expect data to be **normally distributed**
- Instead, the standard approach to computing *p*-values in NLP is to use **non-parametric tests**:
  - Bootstrap test
  - Approximate randomization

Bootstrap Test

- Repeatedly draws many small samples from the test set, with replacement
- Assumes each sample is representative of the overall population
- For each sample, checks to see how well Model A and Model B perform on it
- Keeps a running total of the number of samples for which the difference between Model A's and Model B's performance is more than twice as much as the difference between Model A's and Model B's performance in the overall test set
- Divides the final total by the total number of samples checked to determine the *p*-value

Formal Algorithm: Bootstrap Test

```
Calculate δ(x)  # Performance difference between Models A and B
for i = 1 to b do:  # b = number of samples
      for j = 1 to n do:  # n = size of bootstrap sample
            Randomly select a test instance and add it to the
            bootstrap sample
      Calculate δ(x*(i))  # Performance difference between Models A
                          # and B for the bootstrap sample x*(i)
for each x*(i):
      s = s+1 if δ(x*(i)) > 2δ(x)
p(x) = s/b
```

Summary: Text Classification and Evaluation Metrics

- Classification model performance is determined by comparing the model's predictions to a set of **gold standard labels**
- The similarities and differences between predicted and actual labels can be summarized in a **contingency table** containing **true positives**, **false positives**, **true negatives**, and **false negatives**
- Four common metrics can be computed from values in this table
  - **Precision:** Of the observations predicted to be true, how many actually are?
  - **Recall:** Of the observations that are true, how many were predicted to be?
  - **F-Measure:** What is the harmonic mean between precision and recall?
  - **Accuracy:** What percentage of observations did the model label correctly?
- Multi-class classification can be **multi-label classification** (an instance can have multiple labels) or **multinomial classification** (an instance has one distinct label)
- To check overall performance in multi-class settings, performance metrics can be **macroaveraged** or **microaveraged**
- Model performance can be evaluated using a **test set** or **cross-validation**
- To ensure that model performance really is better than alternate approaches, **statistical significance testing** should be performed