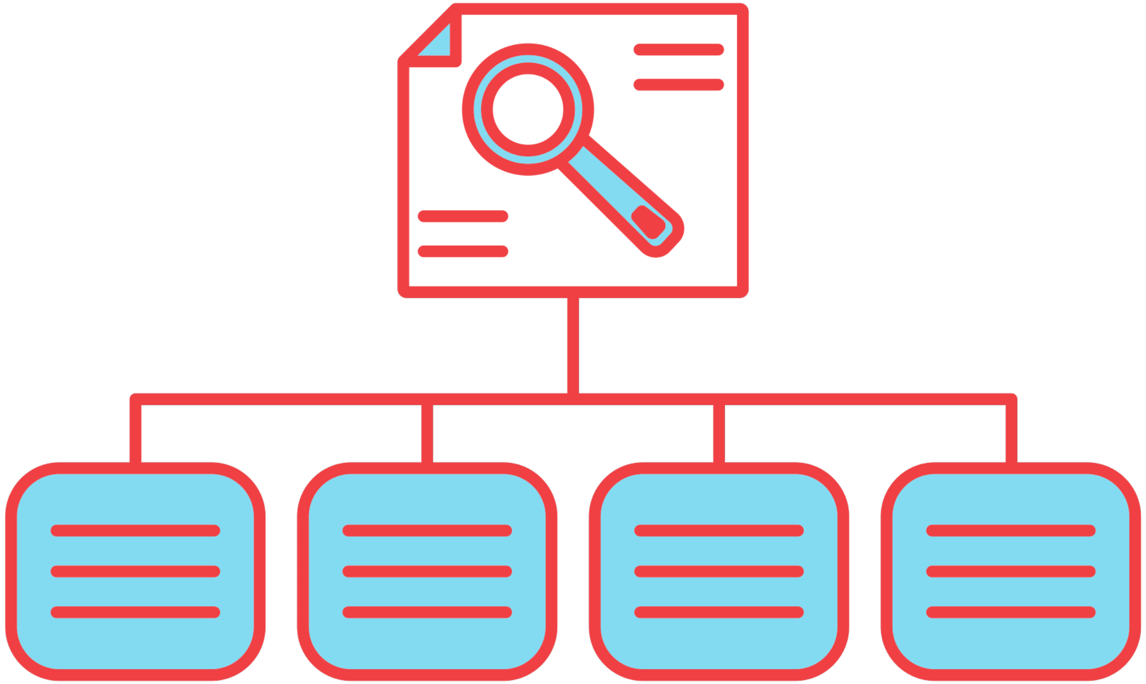


Naïve Bayes and Text Classification

Natalie Parde

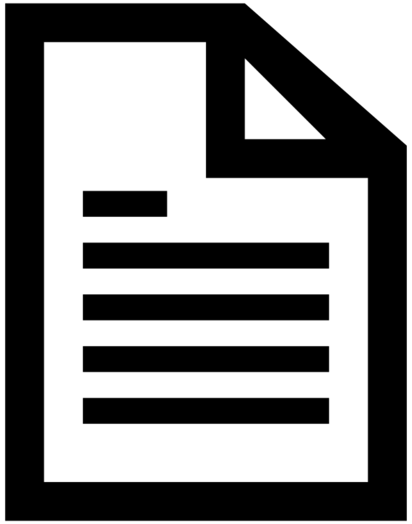
UIC CS 421

What is text classification?

The process of deciding the **category** of an **instance**

Instance: A document, sentence, word, image, transcript, or other individual language sample

Fundamental to many NLP tasks



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

"What can be the meaning of that emphatic exclamation?" cried he. "Do you consider the forms of introduction, and the stress that is laid on them, as nonsense? I cannot quite agree with you _there_. What say you, Mary? For you are a young lady of deep reflection, I know, and read great books and make extracts."

Mary wished to say something sensible, but knew not how.

"While Mary is adjusting her ideas," he continued, "let us return to Mr. Bingley."

"I am sick of Mr. Bingley," cried his wife.

"The world is full of obvious things which nobody by any chance ever observes. Where do you think that I have been?"

"A fixture also."

"On the contrary, I have been to Devonshire."

"In spirit?"

Voltaire

Sir Arthur Conan Doyle

Jane Austen

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

"What can be the meaning of that emphatic exclamation?" cried he. "Do you consider the forms of introduction, and the stress that is laid on them, as nonsense? I cannot quite agree with you _there_. What say you, Mary? For you are a young lady of deep reflection, I know, and read great books and make extracts."

Mary wished to say something sensible, but knew not how.

"While Mary is adjusting her ideas," he continued, "let us retu

"I am sick of Mr. Bingley," cried his wif

The model takes two inputs: the tokenized interview, and the corresponding POS tag list. Word embeddings for the interview text tokens are computed using pre-trained 300 dimensional GloVe embeddings trained on the Wikipedia 2014 and Gigaword 5 dataset (Pennington et al., 2014). The POS tag for each word is represented as a one-hot encoded vector. The word embeddings and POS vectors are input to two different CNNs utilizing the same architecture, and the output of the two CNNs is then flattened and given as input to a bidirectional LSTM with an attention mechanism.

Fiction

Academic

Classification can be used to make decisions about segments of text in addition to full documents.

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

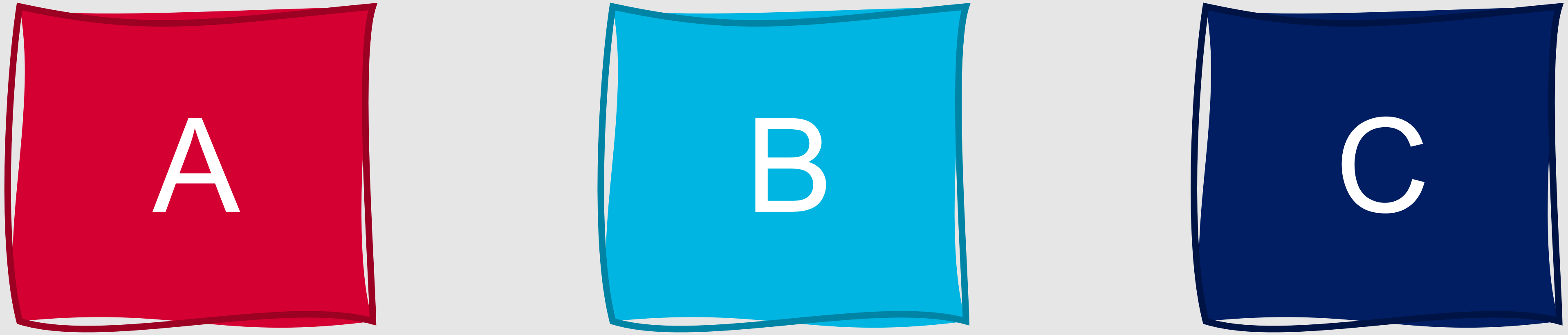
- Is this the last character in a word?

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
- Statistical methods
  - Deep learning methods can be viewed as a subset of statistical methods that leverage implicitly-learned features

Rule-Based Classification Methods

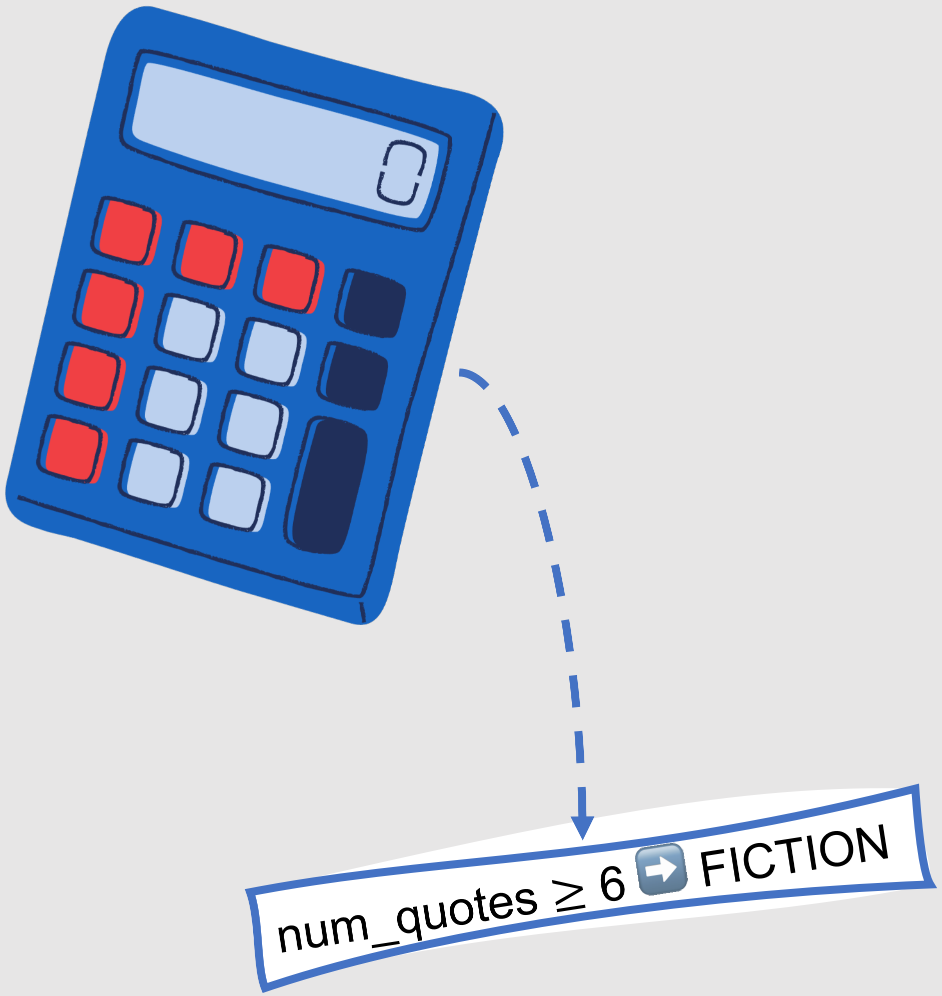
- **Manually create a set of rules** based on expected differences among features from different classes
- Use that information to classify test data

If text contains “love” ➡️ POSITIVE

If text contains quotation marks ➡️ FICTION

Statistical Classification Methods

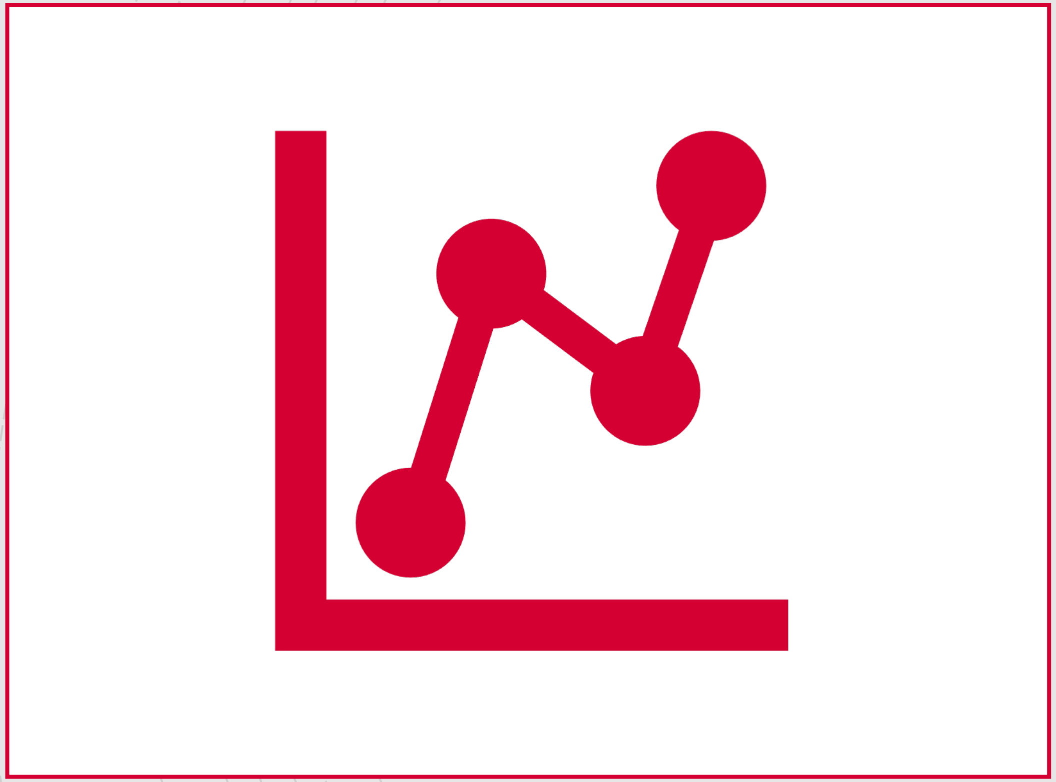
- Automatically **learn which features best distinguish different classes** from one another based on a collection of training data
- Use that information to classify test data



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**
- If data is really limited, rule-based methods will probably work better

Language is dynamic.



- This is one of the reasons why statistical methods have advantages over rule-based techniques
- Word uses can change over time, and so can data
  - He *ghosted* me
  - *Covid-19*
- With rule-based methods, we have to write new rules to accommodate changes in language
  - We also might miss some changes!
- Statistical methods can be automatically retrained when new data is available

Types of Statistical Classification Techniques

Supervised learning: Statistical classification *with* a labeled training set

Unsupervised learning: Statistical classification *without* a labeled training set

Supervised Machine Learning

- 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
  - Assess performance by comparing predicted outputs with the correct outputs that we know from a labeled test set

More formally….

- Take an input *x* from a set of inputs $x \in X$
- Consider a fixed set of output classes $y \in Y$, where $Y = \{y_1, y_2, \ldots, y_M\}$
  - 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 $c \in C$ (equivalently, learning to predict the correct class $y \in Y$ for an input $x \in X$)

Unsupervised Machine Learning

- Input instances are not associated with known labels
- Goal: Automatically discover relationships between instances and group them together accordingly
  - We can still assess performance by comparing predictions with known outputs

Types of Supervised Classification Models

- Naïve Bayes
- Logistic regression
- Support vector machine
- K-nearest neighbors
- Multilayer perceptrons (neural networks)
- …and many more!

These classification models can be further subdivided into groups.

- **Generative classifiers** build models of how classes could generate input data
  - Given an observation, they return the class most likely to have generated it
- **Discriminative classifiers** learn which features from the input are most useful to discriminate between different possible classes
  - Given an observation, they return the best match based on these weighted features

Generative and discriminative classifiers can both be probabilistic.

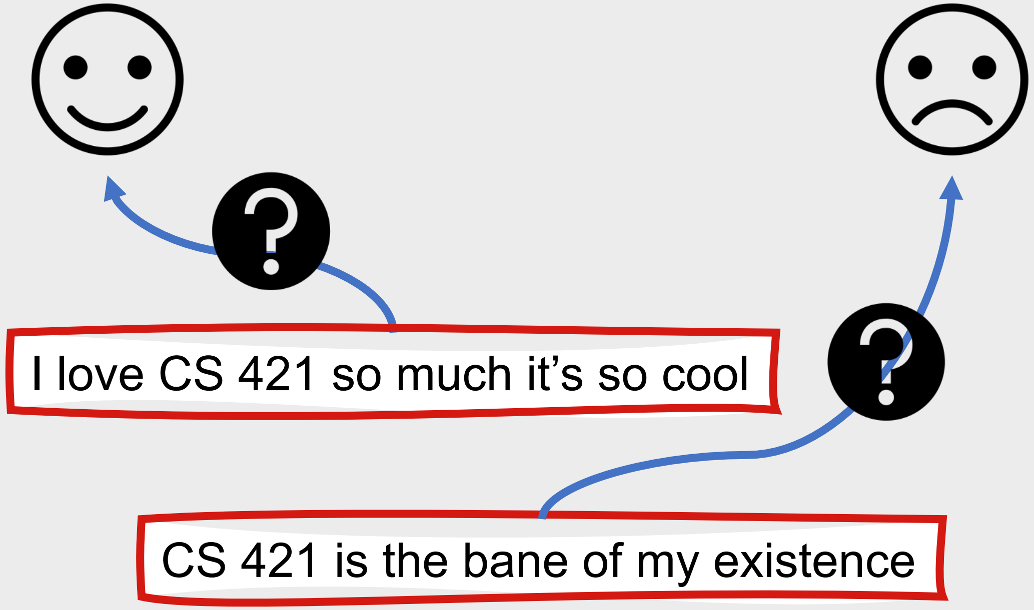
- **Probabilistic classifier:** Makes its decisions based on probability distributions across all available classes
  - Provides a probability of a given data instance belonging in each class
- Useful for downstream decision making, particularly when combining information from multiple probabilistic classifiers

One example of a probabilistic, generative classifier?

Naïve Bayes

What is Naïve Bayes?

- A **probabilistic classifier** that learns to **predict labels** for new documents



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?

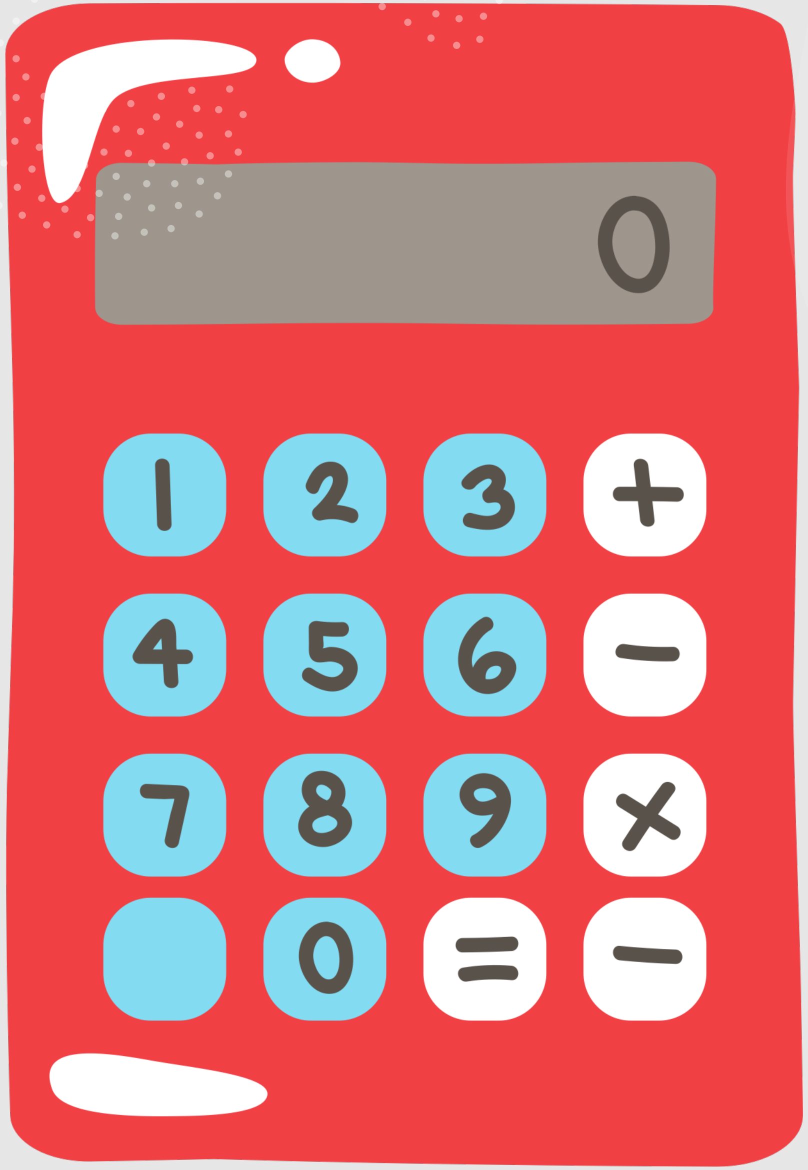
- Naïve Bayes classifiers make a naïve assumption about how features interact with one another: quite simply, they assume that they don't
- They instead **assume that all features are independent from one another**
- Is this really the case?
  - No---as already seen with language models, words are dependent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

How does naïve Bayes work?

- 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.

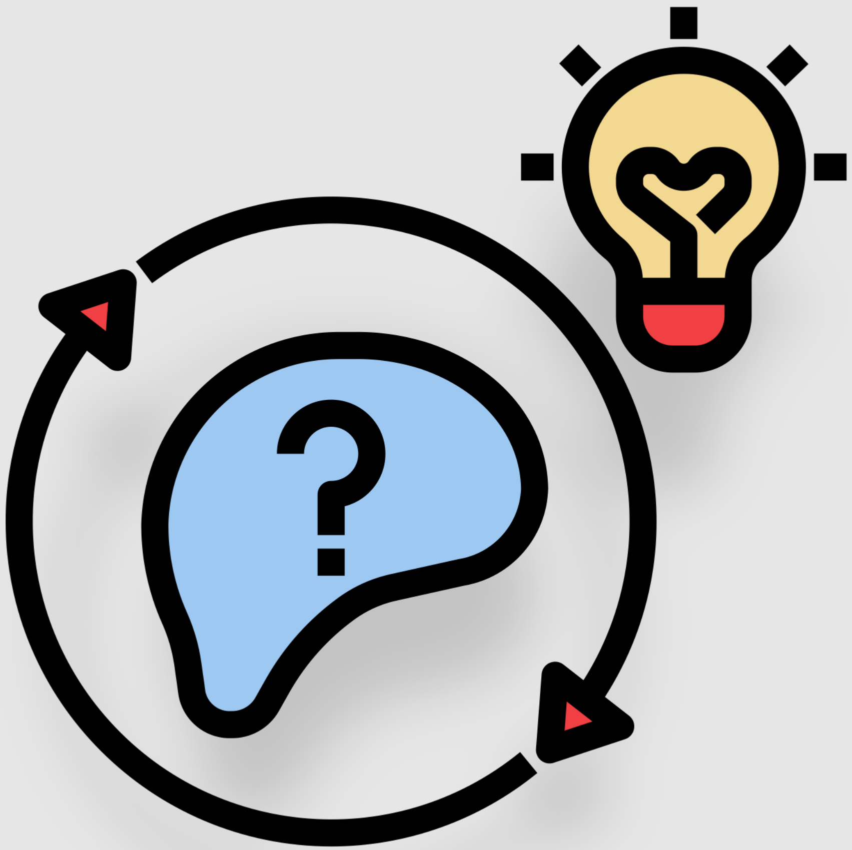
- Bayesian inference uses **Bayes' rule** to transform probabilities like those shown previously into other probabilities that are easier or more convenient to calculate
- Bayes' rule:
  - $P(y|x) = \frac{P(x|y)P(y)}{P(x)}$



Applying Bayesian inference to Naïve Bayes

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

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



We can simplify this even further….

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

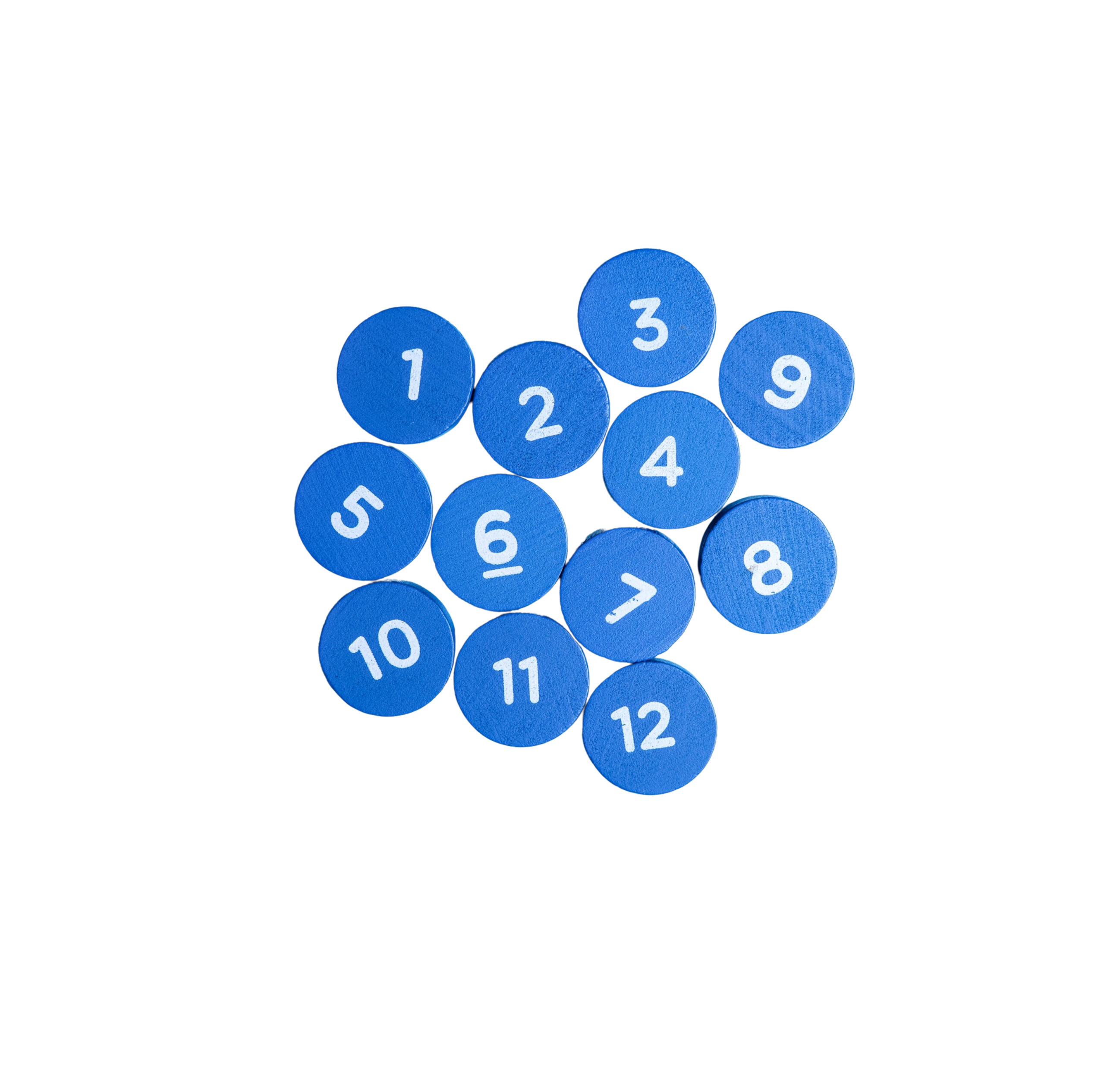
What does this mean?

- 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*)

$$c' = \underset{c \in C}{\operatorname{argmax}} \underbrace{P(d|c)}_{\text{likelihood}} \underbrace{P(c)}_{\text{prior}}$$

To find these probabilities….

- We need to represent our text sample using one or more numbers
- These numbers can represent different **features** of the data



Feature Representation: 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**

Usman's poem about Thanksgiving rivaled Natalie's notorious Halloween poem.

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

Bag of Words Features

- Bags of words are 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:
  - $c' = \underset{c \in C}{\operatorname{argmax}} P(d|c)P(c) = \underset{c \in C}{\operatorname{argmax}} \underbrace{P(f_1, f_2, \ldots, f_n|c)}_{\text{likelihood}} \underbrace{P(c)}_{\text{prior}}$

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.

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

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

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

How do we apply our Naïve Bayes classifier to text?

- Extract bag of words features and insert them into the equation
  - $c' = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i \in N} P(f_i|c)$
- To avoid underflow (the generation of numbers that are too tiny to be adequately represented) and increase speed, in real-world applications we usually do these computations in log space:
  - $c' = \underset{c \in C}{\operatorname{argmax}} \log P(c) + \sum_{i \in N} \log P(f_i|c)$

Generalizing from this principle, we can see how linear classifiers work.

- When we perform these computations in log space, we end up predicting a class as a **linear function** of the input features
  - $c' = \underset{c \in C}{\operatorname{argmax}} \log P(c) + \sum_{i \in T} \log P(w_i|c)$
- 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 $P(f_i|c)$?
- To compute P(*c*), we figure out what percentage of the instances in our training set are in class *c*
  - Let N_c be the number of instances in our training data with class *c*
  - Let N_{doc} be the total number of instances, or documents
  - $P(c)' = \frac{N_c}{N_{doc}}$
- To compute $P(f_i|c)$….
  - **Maximum likelihood estimates!**

For now, we're assuming that features are words from a bag of words.

- Thus, to compute P(f_i|*c*), we'll just need P(w_i|*c*)
  - Fraction of times w_i appears among all words in all documents of class *c*
- How do we do this?
  - Concatenate all instances 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*

Recall that zero probabilities are a pain.

- 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
  - $c' = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i \in T} P(w_i|c)$

How do we fix this issue?

- Smoothing!
- Simplest solution: Laplace (add-one) smoothing
  - $P(w_i|c)' = \frac{count(w_i,c)+1}{\sum_{w\in V}(count(w,c)+1)} = \frac{\boldsymbol{count(w_i,c)+1}}{\boldsymbol{\sum_{w\in V}(count(w,c))+|V|}}$

What about unknown words?

- Some words will inevitably occur in the test data despite never having occurred in the training data
- Easy solution for Naïve Bayes?
  - Ignore words that didn't exist in the training data (remove from test document + do not compute any probabilities for them)

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 Algorithm (Training)

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) # Remove log() if we're not in log space
    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(count(w_i,c)+1 / (∑_{w∈V}(count(w,c))+|V|)) # Remove log() if we're not in log space
return logprior,loglikelihood,V
```

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

Final Algorithm (Testing)

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] # Multiply instead of add if we're not in log space
return
```
$\underset{c}{\operatorname{argmax}}\, sum[c]$

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

Not Sarcastic

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

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.	?

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

- 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.

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.	?

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

- 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.

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) = $\frac{1+1}{15+21} = 0.056$
  - P("Natalie"|Not Sarcastic) = $\frac{1+1}{12+21} = 0.061$

Make sure to use smoothing!

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) = $\frac{1+1}{15+21} = 0.056$
  - P("Natalie"|Not Sarcastic) = $\frac{1+1}{12+21} = 0.061$
  - P("Usman"|Sarcastic) = $\frac{1+1}{15+21} = 0.056$
  - P("Usman"|Not Sarcastic) = $\frac{1+1}{12+21} = 0.061$

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) = $\frac{1+1}{15+21} = 0.056$
  - P("Natalie"|Not Sarcastic) = $\frac{1+1}{12+21} = 0.061$
  - P("Usman"|Sarcastic) = $\frac{1+1}{15+21} = 0.056$
  - P("Usman"|Not Sarcastic) = $\frac{1+1}{12+21} = 0.061$
  - P("soooo"|Sarcastic) = $\frac{1+1}{15+21} = 0.056$
  - P("soooo"|Not Sarcastic) = $\frac{0+1}{12+21} = 0.030$

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) = $\frac{1+1}{15+21} = 0.056$
  - P("Natalie"|Not Sarcastic) = $\frac{1+1}{12+21} = 0.061$
  - P("Usman"|Sarcastic) = $\frac{1+1}{15+21} = 0.056$
  - P("Usman"|Not Sarcastic) = $\frac{1+1}{12+21} = 0.061$
  - P("soooo"|Sarcastic) = $\frac{1+1}{15+21} = 0.056$
  - P("soooo"|Not Sarcastic) = $\frac{0+1}{12+21} = 0.030$
  - P("totally"|Sarcastic) = $\frac{1+1}{15+21} = 0.056$
  - P("totally"|Not Sarcastic) = $\frac{0+1}{12+21} = 0.030$

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.	?

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

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) = $\frac{1+1}{15+21} = 0.056$
  - P("Natalie"|Not Sarcastic) = $\frac{1+1}{12+21} = 0.061$
  - P("Usman"|Sarcastic) = $\frac{1+1}{15+21} = 0.056$
  - P("Usman"|Not Sarcastic) = $\frac{1+1}{12+21} = 0.061$
  - P("soooo"|Sarcastic) = $\frac{1+1}{15+21} = 0.056$
  - P("soooo"|Not Sarcastic) = $\frac{0+1}{12+21} = 0.030$
  - P("totally"|Sarcastic) = $\frac{1+1}{15+21} = 0.056$
  - P("totally"|Not Sarcastic) = $\frac{0+1}{12+21} = 0.030$
  - P("happy"|Sarcastic) = $\frac{0+1}{15+21} = 0.028$
  - P("happy"|Not Sarcastic) = $\frac{1+1}{12+21} = 0.061$

Example: Naïve Bayes

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

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	0.056	0.061
Usman	0.056	0.061
soooo	0.056	0.030
totally	0.056	0.030
happy	0.028	0.061

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 * 0.056 * 0.056 * 0.056 * 0.056 * 0.028 = 1.377 * 10^{-7}

Word	P(Word|Sarcastic)	P(Word|Not Sarcastic)
Natalie	0.056	0.061
Usman	0.056	0.061
soooo	0.056	0.030
totally	0.056	0.030
happy	0.028	0.061

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 * 0.056 * 0.056 * 0.056 * 0.056 * 0.028 = 1.377 * 10^{-7}
  - P(Not Sarcastic)*P(*s*|Not Sarcastic) = 0.5 * 0.061 * 0.061 * 0.030 * 0.030 * 0.061 = 1.021 * 10^{-7}

Word	P(Word|Sarcastic)	P(Word|Not Sarcastic)
Natalie	0.056	0.061
Usman	0.056	0.061
soooo	0.056	0.030
totally	0.056	0.030
happy	0.028	0.061

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 * 0.056 * 0.056 * 0.056 * 0.056 * 0.028 = 1.377 * 10^{-7}**
  - P(Not Sarcastic)*P(*s*|Not Sarcastic) = 0.5 * 0.061 * 0.061 * 0.030 * 0.030 * 0.061 = 1.021 * 10^{-7}

Word	P(Word|Sarcastic)	P(Word|Not Sarcastic)
Natalie	0.056	0.061
Usman	0.056	0.061
soooo	0.056	0.030
totally	0.056	0.030
happy	0.028	0.061

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. 😢

There are many ways to handle negation....

- One simple strategy?
  - Add the prefix “NOT_” to every word after a token known to indicate negation (e.g., “n’t,” “not,” etc.)
  - Compute frequencies for these new "words" just like our other tokens
  - Use this enhanced vocabulary when training and testing models
- Detecting negation and its scope is a complex ongoing research challenge

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 or 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, supervised classification algorithm**
- 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
- **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

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

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

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

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	0.056	0.061
Usman	0.056	0.061
soooo	0.056	0.030
totally	0.056	0.030
happy	0.028	0.061

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	0.056	0.061
Usman	0.056	0.061
soooo	0.056	0.030
totally	0.056	0.030
happy	0.028	0.061

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	$\frac{1+1}{27+34}$ = 0.033	$\frac{1+1}{21+34}$ = 0.036
Usman	$\frac{1+1}{27+34}$ = 0.033	$\frac{1+1}{21+34}$ = 0.036
soooo	$\frac{1+1}{27+34}$ = 0.033	$\frac{0+1}{21+34}$ = 0.018
totally	$\frac{1+1}{27+34}$ = 0.033	$\frac{0+1}{21+34}$ = 0.018
happy	$\frac{0+1}{27+34}$ = 0.016	$\frac{1+1}{21+34}$ = 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	$\frac{0+1}{27+34}$	$\frac{0+1}{21+34}$

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	$\frac{0+1}{27+34}=0.016$	$\frac{0+1}{21+34}=0.018$
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	$\frac{0+1}{27+34}=0.016$	$\frac{0+1}{21+34}=0.018$

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

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 = **$1.26 * 10^{-16}$**

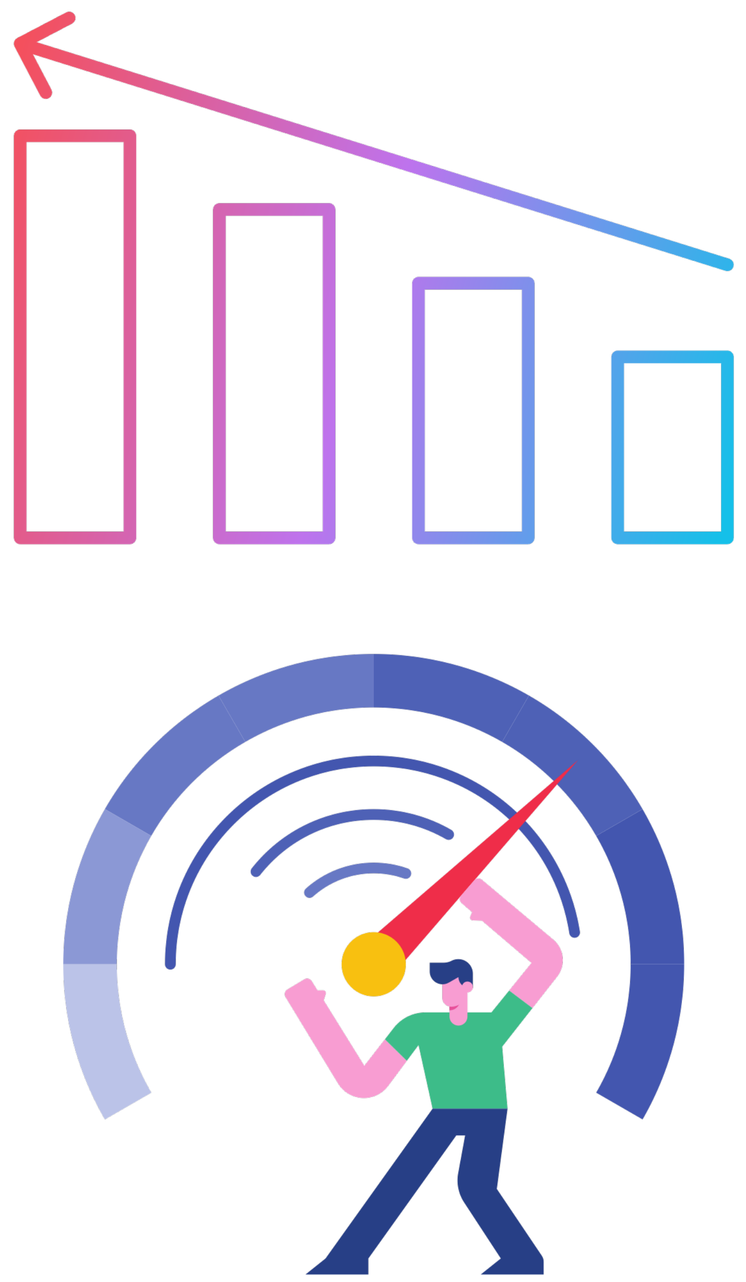
P("Natalie told Usman she was soooo totally happy for him"|Not Sarcastic) = 0.036 * 0.018 * 0.036 * 0.018 * 0.055 * 0.018 * 0.018 * 0.036 * 0.036 * 0.018 = **$1.75 * 10^{-16}$**

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**! This is a good example of how stop words can be problematic in text classification, particularly with extremely tiny datasets.

We've learned a bit about text classification now….

How can we measure the performance of our models?



Evaluating Text Classifiers

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
  - Appen: https://appen.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

Actual

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

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

Why not just use accuracy and be done with it?

- This metric can be unreliable 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.

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

Precision

Recall

F-Measure

What are some alternatives that can focus on specific classes?

Precision

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

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

Recall

Actual

Predicted

True Positive (TP)	False Positive (FP)
False Negative (FN)	True Negative (TN)

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

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
  - $\beta > 1$ means that recall is more important
  - $\beta < 1$ means that precision is more important
  - $\beta = 1$ 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
  - $F_1 = \frac{(1^2+1)PR}{1^2 P+R} = \frac{2PR}{P+R}$
- 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 smoke alarm broke.	Sarcastic	
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	
Oh yay more things to grade!!!	Sarcastic	
Oh yay my new subscription box arrived!!!	Not Sarcastic	
Where is the closest coffee shop?	Not Sarcastic	
I just love large group meetings.	Sarcastic	

Example: Precision, Recall, and F_1

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large group meetings.	Sarcastic	Not Sarcastic

Example: Precision, Recall, and F_1

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large group meetings.	Sarcastic	Not Sarcastic

Positive Class: Sarcastic

Actual

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

Example: Precision, Recall, and F_1

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large group meetings.	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 smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large group meetings.	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 smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large group meetings.	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 smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large group meetings.	Sarcastic	Not Sarcastic

Positive Class: Sarcastic

Actual

Predicted

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

Example: Precision, Recall, and F_1

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large group meetings.	Sarcastic	Not Sarcastic

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

Positive Class: Sarcastic

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

Example: Precision, Recall, and F_1

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large group meetings.	Sarcastic	Not Sarcastic

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

Positive Class: Sarcastic

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

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

Example: Precision, Recall, and F_1

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large group meetings.	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 smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large group meetings.	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 smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large group meetings.	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 smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large group meetings.	Sarcastic	Not Sarcastic

Actual

Predicted

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

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 smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large group meetings.	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 smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large group meetings.	Sarcastic	Not Sarcastic

Actual

Predicted

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

Positive Class: Not Sarcastic

Precision = 0.4

Recall = 0.667

$$F_1 = \frac{(1^2+1)PR}{1^2P+R} = \frac{2PR}{P+R} = \frac{2*0.4*0.667}{0.4+0.667} = 0.50009$$

What if we have more than two classes?

Many NLP classification tasks have more than two classes

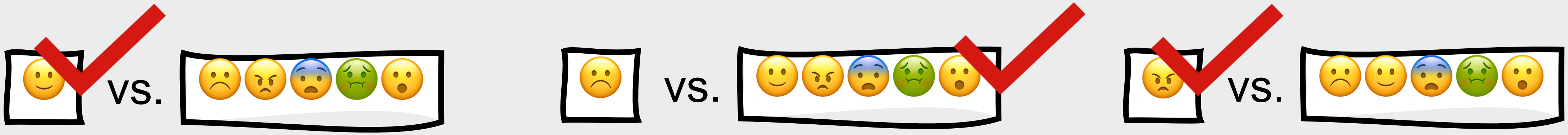
- Sentiment analysis (positive, negative, neutral)
- Part-of-speech tagging (noun, verb, adjective, etc.)
- Emotion detection (happy, sad, angry, surprised, afraid, disgusted)

Classification Paradigms

- Multi-label classification
- Multinomial classification

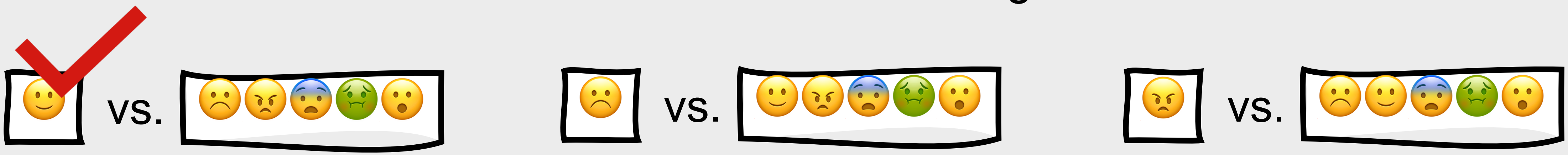
Multi-Label Classification

- 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

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

Multi-Class Precision

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

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

Multi-Class Recall

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

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

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

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

class 1

TP: a	FP: b+c
FN: d+g	

class 2

TP: e	FP: d+f
FN: b+h	

class 3

TP: i	FP: g+h
FN: c+f	

Macroaveraging

Actual

Predicted

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

class 1

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

TP: a	FP: b+c
FN: d+g	

class 2

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

TP: e	FP: d+f
FN: b+h	

class 3

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

TP: i	FP: g+h
FN: c+f	

Macroaveraging

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

Macroaveraged Precision

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

class 1

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

TP: a	FP: b+c
FN: d+g	

class 2

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

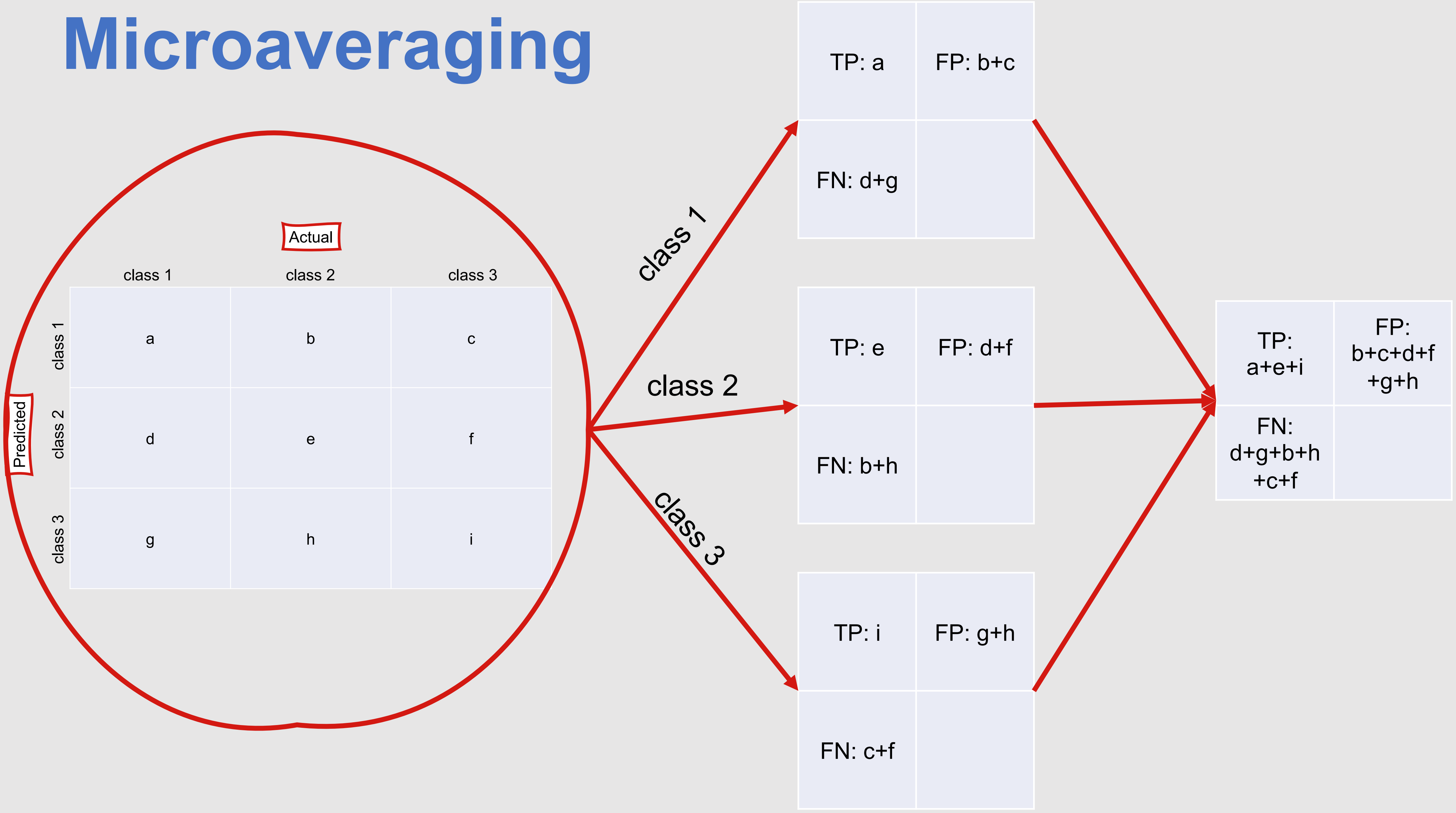
TP: e	FP: d+f
FN: b+h	

class 3

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

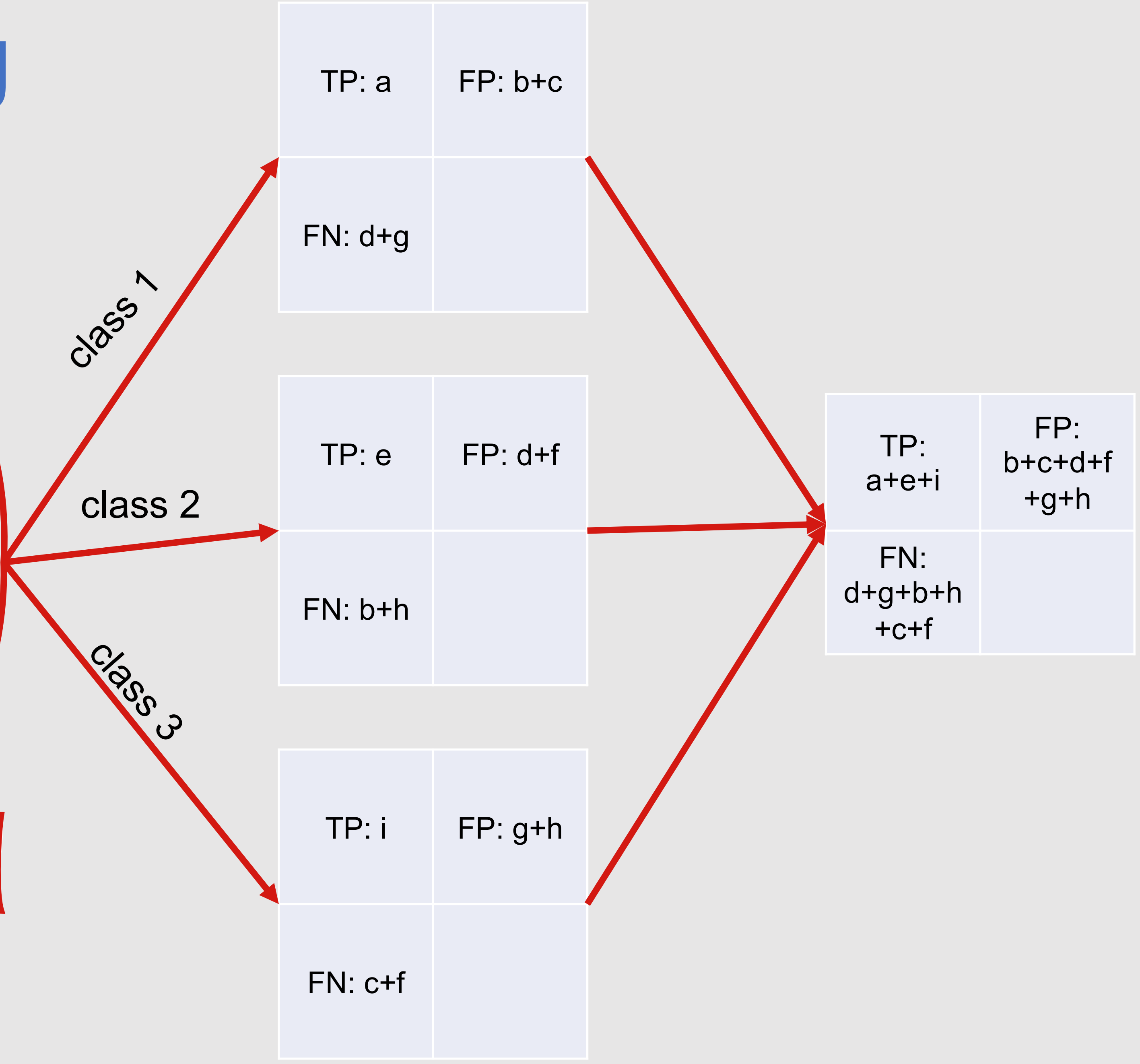
TP: i	FP: g+h
FN: c+f	

Microaveraging



Microaveraging

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



$$\text{Precision} = \frac{\text{tp}}{\text{tp+fp}} = \frac{a+e+i}{a+e+i+b+c+d+f+g+h}$$

$$\text{Recall} = \frac{\text{tp}}{\text{tp+fn}} = \frac{a+e+i}{a+e+i+d+g+b+h+c+f}$$

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 parameters such that they are particularly well-suited for the test data
- Why is overfitting bad?
  - Models that have been overfit tend to perform poorly on unseen samples in the same domain
  - This means that they 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:** This is due to chance, rather than some meaningful reason
    - 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

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?

- There are a variety of ways to determine p-values
- We select methods based on several factors
  - Distribution of our data
  - Number of samples in our dataset
- Most NLP tasks do not involve data from a known distribution
- Because of this, it's common to use **non-parametric tests** to determine statistical significance:
  - Bootstrap test

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
```

Interested in learning more about statistical significance testing in NLP?

- Paper: https://aclanthology.org/P18-1128.pdf
- Book: https://www.morganclaypool.com/doi/10.2200/S00994ED1V01Y202002HLT045

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** or **multinomial classification**
- 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