

Naïve Bayes, Text Classification, and Evaluation Metrics

Natalie Parde, Ph.D.

Department of Computer
Science

University of Illinois at
Chicago

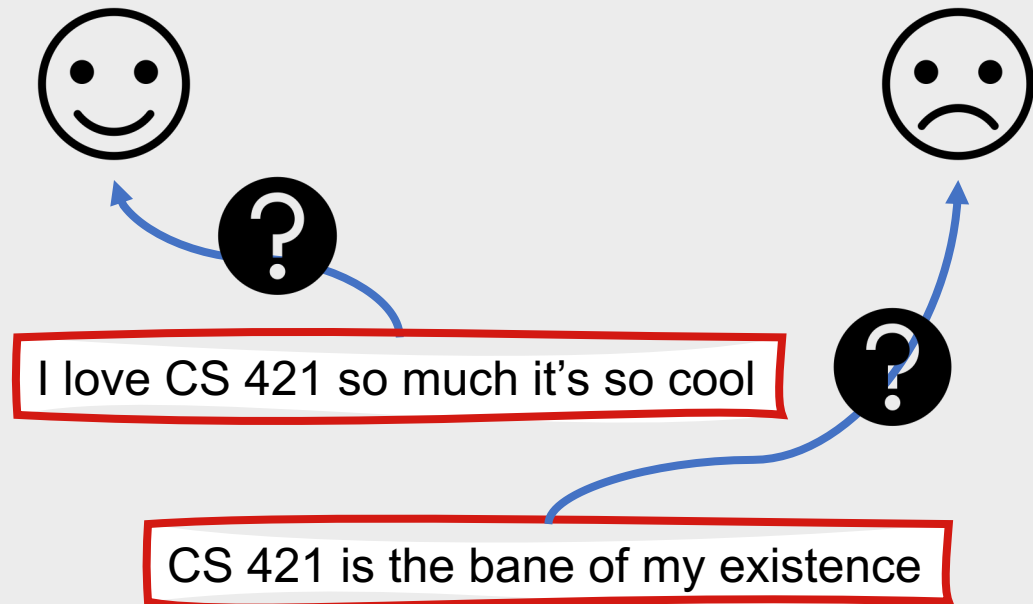
CS 421: Natural Language
Processing

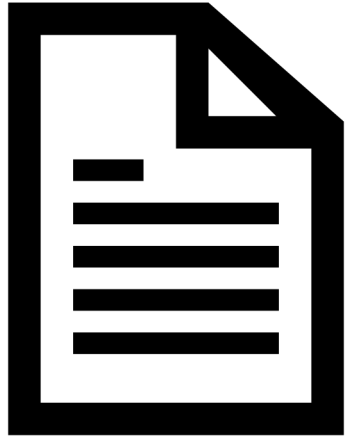
Fall 2019

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

“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 return to Mr. Bingley.”

“I am sick of Mr. Bingley,” cried his wife.

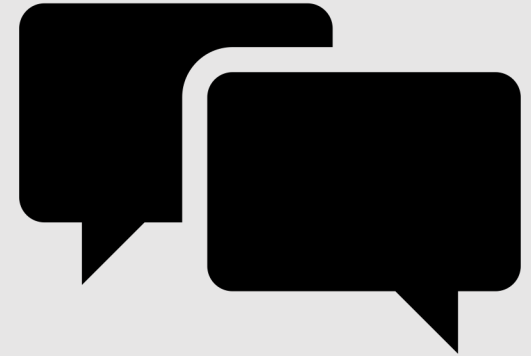
The model takes two inputs: the tokenized interview, and the corresponding POS tag list. Word embeddings for the interview text tokens are computed using pre-trained 300 dimensional GloVe embeddings trained on the Wikipedia 2014 and Gigaword 5 dataset (Pennington et al., 2014). The POS tag for each word 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

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

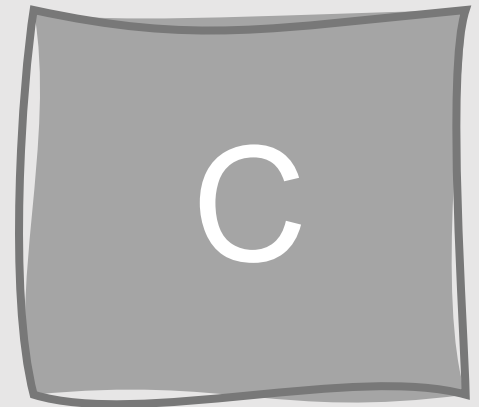
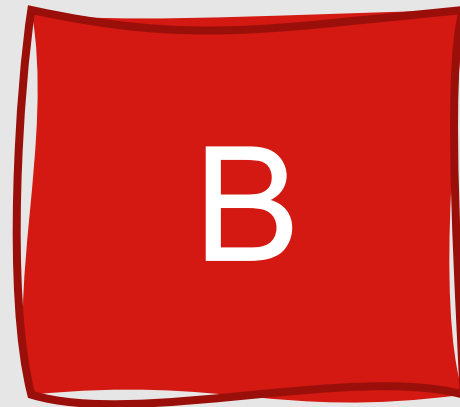
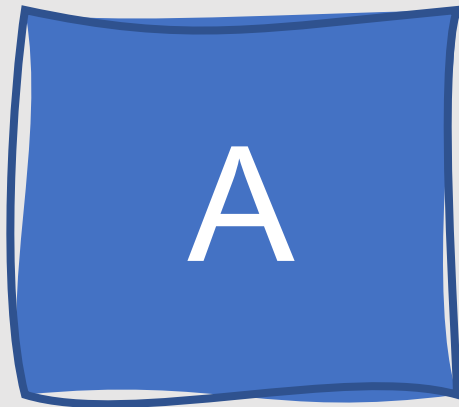
- Is the next word “the”?

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

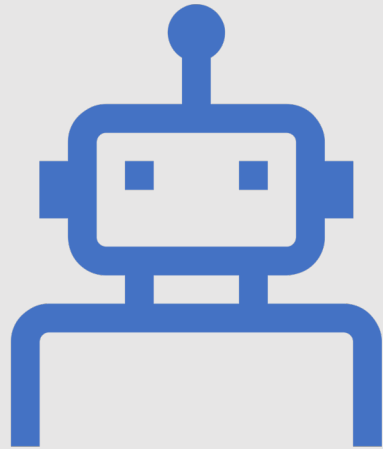
Formal Definition of Supervised Machine Learning

- Take an input x and a fixed set of output classes $Y = \{y_1, y_2, \dots, y_M\}$
- 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: $\{(d_1, c_1), \dots, (d_N, c_N)\}$
- 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
 - $n=2 \rightarrow$ 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
 - $c' = \operatorname{argmax}_{c \in C} P(c|d)$

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

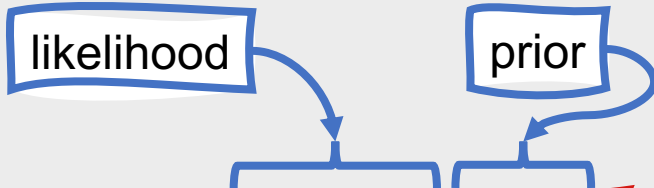
- If we take Bayes' rule:
 - $P(x|y) = \frac{P(y|x)P(x)}{P(y)}$
- And substitute it into our previous equation:
 - $c' = \operatorname{argmax}_{c \in C} P(c|d)$
- We get the following:
 - $c' = \operatorname{argmax}_{c \in C} P(c|d)$
 $= \operatorname{argmax}_{c \in C} \frac{P(d|c)P(c)}{P(d)}$

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

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 $P(d|c)$



The diagram shows two blue-outlined boxes at the top, labeled 'likelihood' and 'prior'. Arrows from both boxes point down to a red-bordered box containing the equation $c' = \operatorname{argmax}_{c \in C} P(d|c)P(c)$. The 'likelihood' box has a straight arrow, while the 'prior' box has a curved arrow.

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

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 $\{f_1, f_2, \dots, f_n\}$, where each feature f corresponds to the frequency of one of the words in the vocabulary
- This means that:

$$c' = \operatorname{argmax}_{c \in C} P(d|c)P(c) = \operatorname{argmax}_{c \in C} \underbrace{P(f_1, f_2, \dots, f_n|c)}_{\text{likelihood}} \underbrace{P(c)}_{\text{prior}}$$

likelihood

prior

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, \dots, f_n encode only a word's identity, 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:
 - $P(f_1, f_2, \dots, f_n | c) = P(f_1 | c) * P(f_2 | c) * \dots * P(f_n | c)$

This brings us to
our final equation.

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

$$= \operatorname{argmax}_{c \in \mathcal{C}} P(f_1, f_2, \dots, f_n|c) P(c)$$

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

How do we apply our Naïve Bayes classifier to text?

- Simply walk through each word in the document and compute its probability
 - $T \leftarrow$ all word (token) positions in a document
 - $c' = \operatorname{argmax}_{c \in C} P(c) \prod_{i \in T} P(w_i | c)$
- 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:
 - $c' = \operatorname{argmax}_{c \in C} \log P(c) + \sum_{i \in T} \log P(w_i | c)$

Linear Classifiers

- When we perform these computations in log space, we end up predicting a class as a linear function of the input features
 - $c' = \operatorname{argmax}_{c \in C} \log P(c) + \sum_{i \in T} P(w_i | c)$
- 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:
 - $P(w_i|c)' = \frac{\text{count}(w_i, c)}{\sum_{w \in V} \text{count}(w, c)}$
 - 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
rivalled	1
Natalie	1
notorious	1
Halloween	1

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

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

How do we fix this issue?

- Simplest solution: Laplace (add-one) smoothing
- $P(w_i|c)' = \frac{\text{count}(w_i,c)+1}{\sum_{w \in V} (\text{count}(w,c)+1)} = \frac{\text{count}(w_i,c)+1}{(\sum_{w \in V} \text{count}(w,c)) + |V|}$
- 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  $c \in C$ : # Calculate  $P(c)$ 
     $N_{\text{doc}} \leftarrow |D|$ 
     $N_c \leftarrow$  number of  $d \in D$  from class  $c$ 
     $\text{logprior}[c] \leftarrow \log(N_c / N_{\text{doc}})$ 
     $V \leftarrow$  vocabulary of  $D$ 
     $\text{superdoc}[c] \leftarrow$   $d \in D$  from class  $c$ 
    for each word  $w$  in  $V$ :
         $\text{count}(w, c) \leftarrow \text{superdoc}[c].\text{count}(w)$ 
         $\text{loglikelihood}[w, c] \leftarrow \frac{\text{count}(w, c) + 1}{(\sum_{w \in V} (\text{count}(w, c)) + |V|)}$ 
return  $\text{logprior}, \text{loglikelihood}, V$ 
```

Test Naïve Bayes

```
for each class  $c \in C$ :
     $\text{sum}[c] \leftarrow \text{logprior}[c]$ 
    for each position  $i$  in  $\text{testdoc}$ :
         $\text{word} \leftarrow \text{testdoc}[i]$ 
        if  $\text{word} \in V$ :
             $\text{sum}[c] \leftarrow \text{sum}[c] + \text{loglikelihood}[\text{word}, c]$ 
return  $\underset{c}{\text{argmax}} \text{sum}[c]$ 
```

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?

$$• P(c)' = \frac{N_c}{N_{doc}}$$

- $P(\text{Sarcastic}) = 2/4 = 0.5$
- $P(\text{Not Sarcastic}) = 2/4 = 0.5$

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?
 - $P(c)' = \frac{N_c}{N_{doc}}$
- $P(\text{Sarcastic}) = 2/4 = 0.5$
- $P(\text{Not Sarcastic}) = 2/4 = 0.5$
- 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.

$$\begin{aligned}P(\text{Sarcastic}) &= 0.5 \\P(\text{Not Sarcastic}) &= 0.5\end{aligned}$$

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.

$$\begin{aligned}P(\text{Sarcastic}) &= 0.5 \\ P(\text{Not Sarcastic}) &= 0.5\end{aligned}$$

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.

$$\begin{aligned}P(\text{Sarcastic}) &= 0.5 \\P(\text{Not Sarcastic}) &= 0.5\end{aligned}$$

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?

$$P(w_i|c)' = \frac{\text{count}(w_i, c)}{\sum_{w \in V} \text{count}(w, c)}$$

$P(\text{Sarcastic}) = 0.5$
 $P(\text{Not Sarcastic}) = 0.5$

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?

- $$P(w_i|c)' = \frac{\text{count}(w_i,c)+1}{(\sum_{w \in V} \text{count}(w,c))+|V|}$$
- $$P(\text{"Natalie"}|\text{Sarcastic}) = \frac{1+1}{27+34} = 0.033$$
- $$P(\text{"Natalie"}|\text{Not Sarcastic}) = \frac{1+1}{21+34} = 0.036$$

$P(\text{Sarcastic}) = 0.5$
 $P(\text{Not Sarcastic}) = 0.5$

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?

- $$P(w_i|c)' = \frac{\text{count}(w_i,c)+1}{(\sum_{w \in V} \text{count}(w,c))+|V|}$$
- $$P(\text{"Natalie"}|\text{Sarcastic}) = \frac{1+1}{27+34} = 0.033$$
- $$P(\text{"Natalie"}|\text{Not Sarcastic}) = \frac{1+1}{21+34} = 0.036$$
- $$P(\text{"Usman"}|\text{Sarcastic}) = \frac{1+1}{27+34} = 0.033$$
- $$P(\text{"Usman"}|\text{Not Sarcastic}) = \frac{1+1}{21+34} = 0.036$$

$$P(\text{Sarcastic}) = 0.5$$

$$P(\text{Not Sarcastic}) = 0.5$$

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?

- $$P(w_i|c)' = \frac{\text{count}(w_i,c)+1}{(\sum_{w \in V} \text{count}(w,c))+|V|}$$
- $$P(\text{"Natalie"}|\text{Sarcastic}) = \frac{1+1}{27+34} = 0.033$$
- $$P(\text{"Natalie"}|\text{Not Sarcastic}) = \frac{1+1}{21+34} = 0.036$$
- $$P(\text{"Usman"}|\text{Sarcastic}) = \frac{1+1}{27+34} = 0.033$$
- $$P(\text{"Usman"}|\text{Not Sarcastic}) = \frac{1+1}{21+34} = 0.036$$
- $$P(\text{"soooo"}|\text{Sarcastic}) = \frac{1+1}{27+34} = 0.033$$
- $$P(\text{"soooo"}|\text{Not Sarcastic}) = \frac{0+1}{21+34} = 0.018$$

$$P(\text{Sarcastic}) = 0.5$$

$$P(\text{Not Sarcastic}) = 0.5$$

Natalie told Usman she was soooo totally happy for him.

Example: Naïve Bayes

- What are the likelihoods from the training set for the remaining words in the test instance?

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.	?

- $P(w_i|c)' = \frac{\text{count}(w_i,c)+1}{(\sum_{w \in V} \text{count}(w,c))+|V|}$
- $P(\text{"Natalie"}|\text{Sarcastic}) = \frac{1+1}{27+34} = 0.033$
- $P(\text{"Natalie"}|\text{Not Sarcastic}) = \frac{1+1}{21+34} = 0.036$
- $P(\text{"Usman"}|\text{Sarcastic}) = \frac{1+1}{27+34} = 0.033$
- $P(\text{"Usman"}|\text{Not Sarcastic}) = \frac{1+1}{21+34} = 0.036$
- $P(\text{"soooo"}|\text{Sarcastic}) = \frac{1+1}{27+34} = 0.033$
- $P(\text{"soooo"}|\text{Not Sarcastic}) = \frac{0+1}{21+34} = 0.018$
- $P(\text{"totally"}|\text{Sarcastic}) = \frac{1+1}{27+34} = 0.033$
- $P(\text{"totally"}|\text{Not Sarcastic}) = \frac{0+1}{21+34} = 0.018$

$P(\text{Sarcastic}) = 0.5$
 $P(\text{Not Sarcastic}) = 0.5$

Natalie told Usman she was soooo totally happy for him.

Example: Naïve Bayes

- What are the likelihoods from the training set for the remaining words in the test instance?

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.	?

- $P(w_i|c)' = \frac{\text{count}(w_i,c)+1}{(\sum_{w \in V} \text{count}(w,c))+|V|}$
- $P(\text{"Natalie"}|\text{Sarcastic}) = \frac{1+1}{27+34} = 0.033$
- $P(\text{"Natalie"}|\text{Not Sarcastic}) = \frac{1+1}{21+34} = 0.036$
- $P(\text{"Usman"}|\text{Sarcastic}) = \frac{1+1}{27+34} = 0.033$
- $P(\text{"Usman"}|\text{Not Sarcastic}) = \frac{1+1}{21+34} = 0.036$
- $P(\text{"soooo"}|\text{Sarcastic}) = \frac{1+1}{27+34} = 0.033$
- $P(\text{"soooo"}|\text{Not Sarcastic}) = \frac{0+1}{21+34} = 0.018$
- $P(\text{"totally"}|\text{Sarcastic}) = \frac{1+1}{27+34} = 0.033$
- $P(\text{"totally"}|\text{Not Sarcastic}) = \frac{0+1}{21+34} = 0.018$
- $P(\text{"happy"}|\text{Sarcastic}) = \frac{0+1}{27+34} = 0.016$
- $P(\text{"happy"}|\text{Not Sarcastic}) = \frac{1+1}{21+34} = 0.036$

$P(\text{Sarcastic}) = 0.5$
 $P(\text{Not Sarcastic}) = 0.5$

Natalie told Usman she was soooo totally happy for him.

Example: Naïve Bayes

- Given all of this information, how should we classify the test sentence?

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

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	0.033	0.036
Usman	0.033	0.036
soooo	0.033	0.018
totally	0.033	0.018
happy	0.016	0.036

$P(\text{Sarcastic}) = 0.5$
 $P(\text{Not Sarcastic}) = 0.5$

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 ?

- $c' = \operatorname{argmax}_{c \in \mathcal{C}} P(c) \prod_{i \in T} P(w_i | c)$
- $P(\text{Sarcastic}) * P(s | \text{Sarcastic}) = 0.5 * 0.033 * 0.033 * 0.033 * 0.033 * 0.016 = 9.487 * 10^{-9}$

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

$P(\text{Sarcastic}) = 0.5$
 $P(\text{Not Sarcastic}) = 0.5$

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 ?

- $c' = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i \in T} P(w_i | c)$
- $P(\text{Sarcastic}) * P(s | \text{Sarcastic}) = 0.5 * 0.033 * 0.033 * 0.033 * 0.033 * 0.016 = 9.487 * 10^{-9}$
- $P(\text{Not Sarcastic}) * P(s | \text{Not Sarcastic}) = 0.5 * 0.036 * 0.036 * 0.018 * 0.018 * 0.036 = 7.558 * 10^{-9}$

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

$P(\text{Sarcastic}) = 0.5$
 $P(\text{Not Sarcastic}) = 0.5$

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 ?

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

- $$P(\text{Sarcastic}) * P(s | \text{Sarcastic}) = 0.5 * 0.033 * 0.033 * 0.033 * 0.033 * 0.016 = 9.487 * 10^{-9}$$

- $$P(\text{Not Sarcastic}) * P(s | \text{Not Sarcastic}) = 0.5 * 0.036 * 0.036 * 0.018 * 0.018 * 0.036 = 7.558 * 10^{-9}$$

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

$$P(\text{Sarcastic}) = 0.5$$

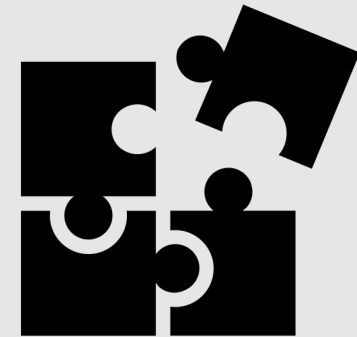
$$P(\text{Not Sarcastic}) = 0.5$$

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