

If the cookie had candy, then very few bites would have no candy.

$$\Pr(\text{no-candy bite} \mid \text{candy cookie}) = \frac{1}{3}$$

The probability of a no-candy bite, given a candy cookie, is 1/3.



If the cookie had no candy, then every bite would have no candy.

$$\Pr(\text{no-candy bite} \mid \text{no-candy cookie}) = 1$$

The probability of a no-candy bite, given a no-candy cookie, is 1.



So the no-candy bite probably came from a no-candy cookie!

CS 2731 / ISSP 2230

Introduction to Natural Language Processing

Session 4: Naive Bayes

Michael Miller Yoder

January 22, 2024

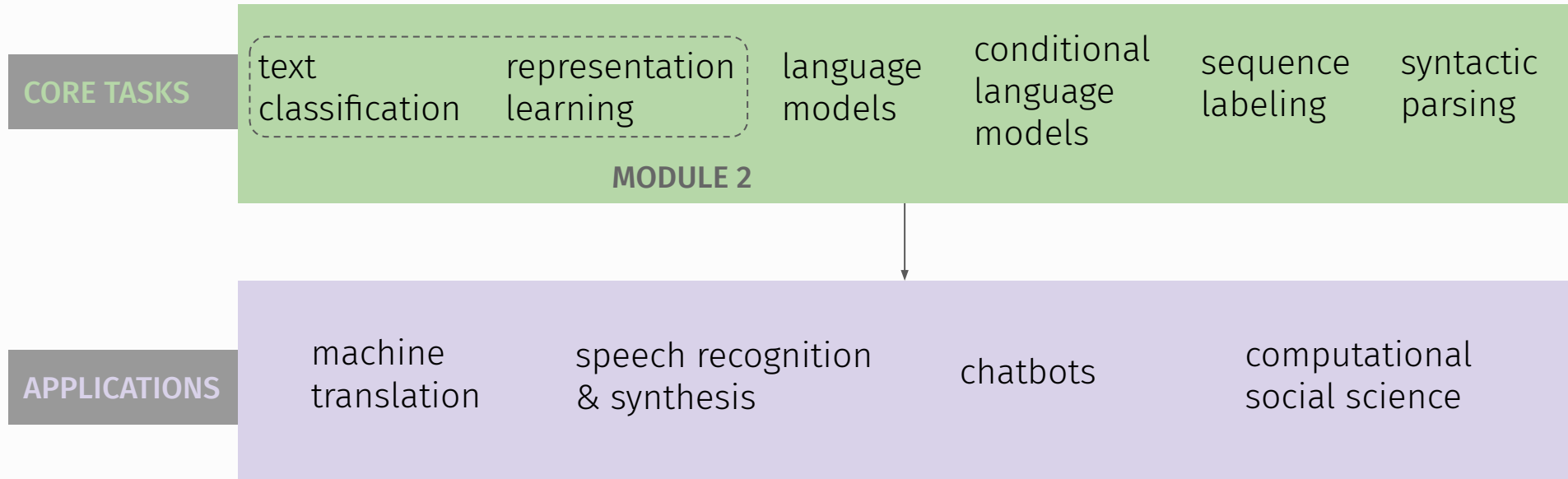
Overview: Naive Bayes

- Course logistics
- Text classification tasks
- Naive Bayes: input, formulation, training

Course logistics

- Homework 1
 - Programming problems, write report based on results
 - **Due next Thu Feb 1 at 11:59pm**
- Course project matching
 - We will email you tomorrow
 - Please plan meeting with groups to discuss project ideas
 - Project area and contribution form will be due Feb 8
 - Not released yet
- First discussion post **due Wed at 1pm**
 - Will be released on Canvas tomorrow
 - This replaces the reading quiz for Wed

Core tasks and applications of NLP



Text classification tasks

Text classification

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

ROMANCE

Pride and Prejudice

DIALOG



Is this spam?

Subject: Important notice!

From: Stanford University <newsforum@stanford.edu>

Date: October 28, 2011 12:34:16 PM PDT

To: undisclosed-recipients;;

Greats News!

You can now access the latest news by using the link below to login to Stanford University News Forum.

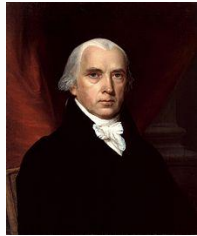
<http://www.123contactform.com/contact-form-StanfordNew1-236335.html>

Click on the above link to login for more information about this new exciting forum. You can also copy the above link to your browser bar and login for more information about the new services.

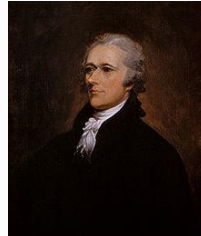
© Stanford University. All Rights Reserved.

Who wrote which Federalist papers?

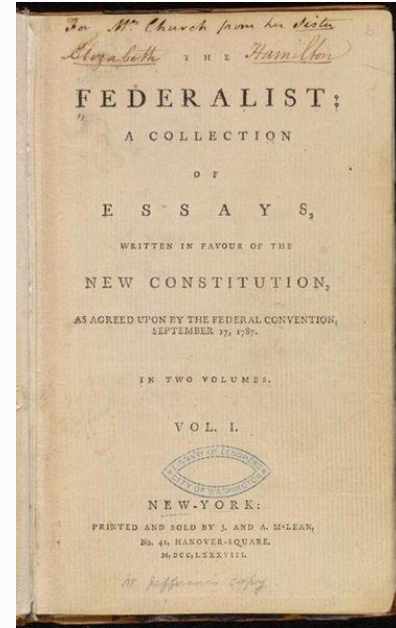
- 1787-1788: anonymous essays try to convince New York to ratify U.S Constitution: Jay, Madison, Hamilton.
- Authorship of 12 of the letters in dispute
- 1963: solved by Mosteller and Wallace using Bayesian methods



James Madison



Alexander Hamilton

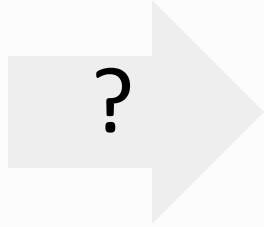


What is the subject of this medical article?

MEDLINE Article

The screenshot shows a MEDLINE article with the following details:

- Journal: *Brain*
- Title: **Syntactic frame and verb bias in aphasia: Fluency judgments of undergoer-subject sentences**
- Authors: Susanne Gahl,¹ Lisa Mann,² Chad Koverberger,³ Daniel S. Jurafsky,³ Elizabeth Eidel,⁴ Molly Rawaga,⁵ and L. Harold Amlund⁶
- Abstract (partial): The study investigates how fluent the have been argued to define "fluency" in aphasia comprehension. Syntactic frame and verb bias are investigated. The study examines the effects of the syntactic frame and verb bias on fluency judgments in aphasia. The study examines the effects of the syntactic frame and verb bias on fluency judgments in aphasia. The study examines the effects of the syntactic frame and verb bias on fluency judgments in aphasia.



MeSH Subject Category Hierarchy

Antagonists and Inhibitors

Blood Supply

Chemistry

Drug Therapy

Embryology

Epidemiology

...

Text Classification

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

Applications:

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

Notation and Setting

- We have a set of n documents (texts) $\mathbf{d}_j \in \mathcal{V}^+$, where \mathcal{V} is the vocabulary of the corpus.
 - We assume the texts are segmented already.
- We have set \mathcal{L} of labels, ℓ_j
- Human experts annotate documents with labels and give us $\{(\mathbf{d}_1, \ell_1), (\mathbf{d}_2, \ell_2), \dots, (\mathbf{d}_n, \ell_n)\}$
- We learn a *classifier* $\mathbf{classify} : \mathcal{V}^+ \rightarrow \mathcal{L}$ with this labeled training data.
- Afterwards, we use $\mathbf{classify}$ to classify new documents into their classes.

Example: Sentiment Detection

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

Naive Bayes classification

Do You Like Sportsball?



Slide credit: David Mortensen

Your mission: identify every
English-language sports article on the
Internet

Observation: 80% of sports articles contain the word *win*.

You assume that all articles containing *win* are likely to be sports articles

But the Good Reverend Bayes asks you:

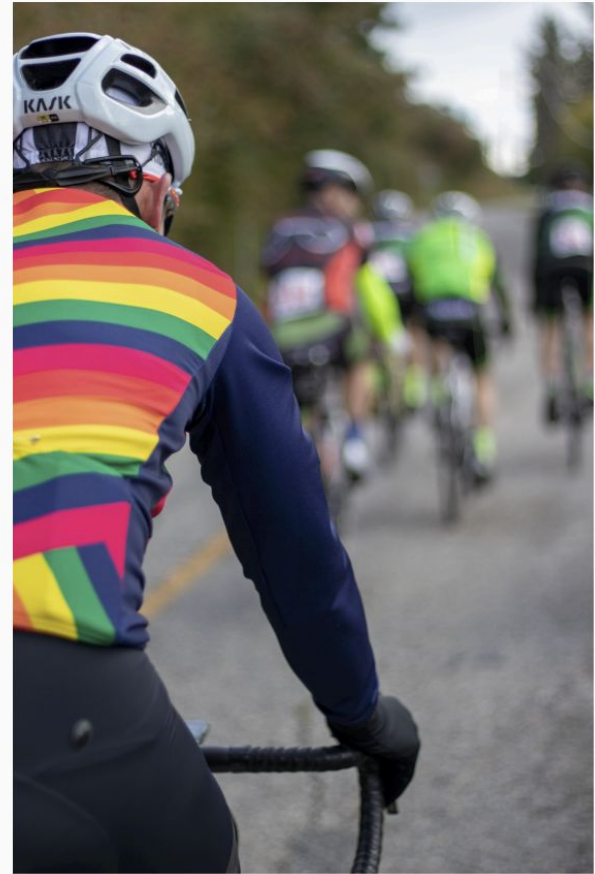
How many times does *win* occur generally?

And what percentage of articles on your
newfangled (and possibly demonic)

Internet are about sports?



Frequency of *win*: 0.0005
Frequency of **sports**: 0.01



$$\hat{P}(\text{sports}|\text{win}) = \frac{P(\text{win}|\text{sports})P(\text{sports})}{P(\text{win})}$$

$$\hat{P}(\text{sports}|\text{win}) = \frac{0.8P(\text{sports})}{P(\text{win})}$$

$$\hat{P}(\text{sports}|\text{win}) = \frac{0.8(0.01)}{P(\text{win})}$$

$$0.16 = \frac{0.8(0.01)}{0.0005}$$

In this example, the probability that a document is about **sports** given that it contains *win* is only 16%

- LIKELIHOOD is high $P(\text{win}|\text{sports})$
- PRIOR is relatively low $P(\text{sports})$
- MARGINAL LIKELIHOOD is relatively high $P(\text{win})$

Input to classification tasks: features

- A training set of movie reviews (with star ratings 1 - 5)
- A set of features for each message (considered as a bag of words)
 - For each word: Number of occurrences
 - Whether phrases such as *Excellent*, *sucks*, *blockbuster*, *biggest*, *Star Wars*, *Disney*, *Adam Sandler*, ...are in the review

Bag of words document representation

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...

Term-document matrix

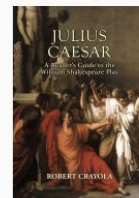
- Each cell is the count of term t in a document d ($tf_{t,d}$).
- Each document is a **count vector** in \mathbb{N}^V , a column below.



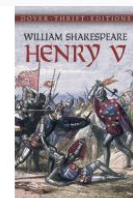
As You Like It



Twelfth Night



Julius Caesar



Henry V

<i>battle</i>	1	1	8	15
<i>soldier</i>	2	2	12	36
<i>fool</i>	37	58	1	5
<i>clown</i>	6	117	0	0

Spam Detection

- A training set of email messages (marked *Spam* or *Not-Spam*)
- A set of features for each message
 - For each word: Number of occurrences
 - Whether phrases such as “Nigerian Prince”, “email quota full”, “won ONE HUNDRED MILLION DOLLARS” are in the message
 - Whether it is from someone you know
 - Whether it is a reply to your message
 - Whether it is from your domain (e.g., `cmu.edu`)

Features in Text Classification

- Running example $\mathbf{d} =$ “The spirit is willing but the flesh is weak”
- *Feature random variables*
- For $j \in \{1, \dots, d\}$ F_j is a discrete random variable taking values in \mathcal{F}_j
- Most of the time these can be frequencies of words or n-grams (including character n-grams) in a text.
 - $f_{f\text{-spirit}} = 1, f_{f\text{-is}} = 2, f_{f\text{-the-flesh}} = 1, \dots$
- They can be boolean “exists” features.
 - $f_{e\text{-spirit}} = 1, f_{e\text{-is}} = 1, f_{e\text{-strong}} = 0, \dots$

Naive Bayes formulation

Bayes' Rule applied to documents and classes

- For a document d and a class c

$$P(c | d) = \frac{P(d | c)P(c)}{P(d)}$$

Naive Bayes classification formula

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

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

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

MAP is “maximum a posteriori” = most likely class

Bayes Rule

Dropping the denominator

Naive Bayes classification formula

"Likelihood"

"Prior"

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



$P(c, d) = P(d | c)P(c)$ by def of conditional probability/chain rule

(Back to the) Naive Bayes classification formula

"Likelihood"

"Prior"

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

$$= \operatorname{argmax}_{c \in C} P(x_1, x_2, \dots, x_n | c)P(c)$$

Document d
represented as
features $x_1 \dots x_n$

Could only be estimated if a very, very large
number of training examples was available.

How often does this
class occur? Count the
relative frequency in a
corpus

Naive Bayes independence assumptions

$$P(x_1, x_2, \dots, x_n | c)$$

Bag of words assumption: Assume word position doesn't matter

Conditional Independence: Assume the feature probabilities $P(x_i | c_j)$ are independent given the class c .

$$P(x_1, \dots, x_n | c) = P(x_1 | c) \cdot P(x_2 | c) \cdot P(x_3 | c) \cdot \dots \cdot P(x_n | c)$$

Naive Bayes classification formula

$$c_{MAP} = \operatorname{argmax}_{c \in C} P(x_1, x_2, \dots, x_n | c)P(c)$$

For all word positions (tokens) in a test document:

$$c_{NB} = \operatorname{argmax}_{c_j \in C} P(c_j) \prod_{i \in \text{positions}} P(x_i | c_j)$$

Problems with multiplying lots of probs

There's a problem with this:

$$c_{NB} = \operatorname{argmax}_{c_j \in C} P(c_j) \prod_{i \in \text{positions}} P(x_i | c_j)$$

- Multiplying lots of probabilities can result in floating-point underflow!
 - $.0006 * .0007 * .0009 * .01 * .5 * .000008...$
- Idea: Use logs, because $\log(ab) = \log(a) + \log(b)$
 - We'll sum logs of probabilities instead of multiplying probabilities!

We actually do everything in log space

Instead of this:
$$c_{NB} = \operatorname{argmax}_{c_j \in C} P(c_j) \prod_{i \in \text{positions}} P(x_i | c_j)$$

This:
$$c_{NB} = \operatorname{argmax}_{c_j \in C} \left[\log P(c_j) + \sum_{i \in \text{positions}} \log P(x_i | c_j) \right]$$

Notes:

1) Taking log doesn't change the ranking of classes!

The class with highest probability also has highest log probability!

2) It's a linear model:

Just a max of a sum of weights: a **linear** function of the inputs

So Naive Bayes is a **linear classifier**

Types of Naive Bayes

How to calculate $P(x_i|c_j)$ depends on what type of feature x_i and classifier

- **Gaussian Naïve Bayes** Considers a feature vector of continuous variables
- **Multinomial Naïve Bayes** Considers a feature vector representing frequencies
- **Bernoulli Naïve Bayes** Considers a vector of binary features

For sentiment analysis, language ID, and other document classification tasks, **Multinomial Naïve Bayes** is generally best

Multinomial Naive Bayes Model: Training

First attempt: maximum likelihood estimates

- Simply use the frequencies in the data (c=class, w=word)

$$\hat{P}(c_j) = \frac{N_{c_j}}{N_{total}}$$

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

fraction of times word w_i appears among all words in documents of topic c_j

Problem with Maximum Likelihood (Zeros)

What if we have seen no training documents with the word *fantastic* and classified in the topic positive?

$$\hat{P}(\text{"fantastic"} \mid \text{positive}) = \frac{\text{count}(\text{"fantastic"}, \text{positive})}{\sum_{w \in V} \text{count}(w, \text{positive})} = 0$$

Zero probabilities cannot be conditioned away, no matter the other evidence!

$$c_{MAP} = \operatorname{argmax}_c \hat{P}(c) \prod_i \hat{P}(x_i \mid c)$$

Laplace (add 1) smoothing for Naïve Bayes

$$\begin{aligned}\hat{P}(w_i | c) &= \frac{\text{count}(w_i, c) + 1}{\sum_{w \in V} (\text{count}(w, c) + 1)} \\ &= \frac{\text{count}(w_i, c) + 1}{\left(\sum_{w \in V} \text{count}(w, c) \right) + |V|}\end{aligned}$$

Smoothing Example

- Two words found in test set: **good** and **wonderful**
- There are 100 word types found with the label “positive” (+)
- In documents labeled +: *good*, 5 times; *wonderful*, 0 times
- Vocabulary (V): 1000 types

$$\hat{P}(\text{good}|+) = \frac{\text{count}(\text{good}, +) + 1}{(\sum_{w \in V} \text{count}(w, +)) + |V|}$$
$$\frac{5 + 1}{100 + 1000}$$

$$\hat{P}(\text{wonderful}|+) = \frac{\text{count}(\text{wonderful}, +) + 1}{(\sum_{w \in V} \text{count}(w, +)) + |V|}$$
$$\frac{0 + 1}{100 + 1000}$$

A Blueprint for NB

To classify a document d :

$$\text{classify}(d) = \operatorname{argmax}_{c \in C} \left(\log p(c) + \sum_{i=1}^{|d|} \log p(w_i | c) \right)$$

Training:

$\mathcal{D} \leftarrow$ all documents

for each $c \in C$ **do**

$\mathcal{D}_c \leftarrow$ docs with class c

$p(c) \leftarrow \frac{|\mathcal{D}_c|}{|\mathcal{D}|}$

Training:

$T_c \leftarrow$ concatenate(\mathcal{D}_c)

for each $w_i \in V$ **do**

$n_i \leftarrow$ count(w_i, T_c)

$p(w_i | c) \leftarrow \frac{n_i + \alpha}{|T_c| + \alpha |V|}$

At inference time, compute smoothed probabilities for words that were not in the training set.

An Example

	ℓ	d
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

- $|V| = 20$
- Add 1 Laplace smoothing

$$\pi_- = p(-) = \frac{3}{5}$$
$$\pi_+ = p(+) = \frac{2}{5}$$

$$N_- = 13$$

$$N_+ = 9$$

$$p(\text{"predictable"} \mid -) =$$

$$p(\text{"predictable"} \mid +) =$$

$$p(\text{"no"} \mid -) =$$

$$p(\text{"no"} \mid +) =$$

$$p(\text{"fun"} \mid -) =$$

$$p(\text{"fun"} \mid +) =$$

$$p(+)\pi_+ =$$

$$p(-)\pi_- =$$

Other Optimizations for Sentiment Analysis

- Ignore unknown words in the test.
- Ignore **stop words** like *the, a, with*, etc.
 - Remove most frequent 10-100 words from the training and test documents.
- Count vs existence of words: Binarized features.
- Negation Handling `didnt like this movie , but I` → `didnt NOT_like NOT_this NOT_movie , but I`

Questions?

Homework 1 due **next Thu, Feb 1**