Text Classification and Naive Bayes

The Task of Text Classification

Is this spam?

Good morning Dan,

Please familiarize yourself with the attached file.
Reply here if you have any questions.

Thank you.

John and Mike,

Appreciate your flexibility this week, as the team navigates the sensitivities surrounding some of the project work taking place at the sites. Please tentatively plan for mobilization on 05/16/2022, in order to begin the final stages of the upgrade.

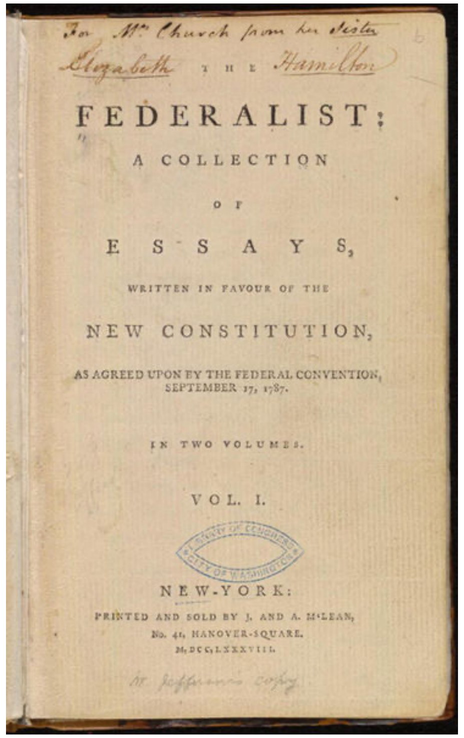
I will follow-up tomorrow with a confirmation if all indications are we will be given the "all-clear" before EOB Wednesday/SOB Thursday.

Appreciate your support.

Regards,

Judy Sewell
Project Manager

Who wrote which *Federalist Papers*?

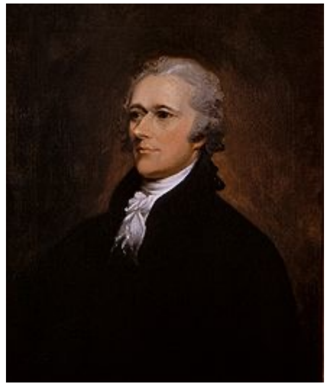

For Mrs Church from her Sister
Elizabeth THE Hamilton
FEDERALIST:
A COLLECTION
OF
ESSAYS,
WRITTEN IN FAVOUR OF THE
NEW CONSTITUTION,
AS AGREED UPON BY THE FEDERAL CONVENTION,
SEPTEMBER 17, 1787.
IN TWO VOLUMES.
VOL. I.
NEW-YORK:
PRINTED AND SOLD BY J. AND A. M'LEAN,
No. 41, HANOVER-SQUARE.
M,DCC,LXXXVIII.

1787-8: essays anonymously written by:

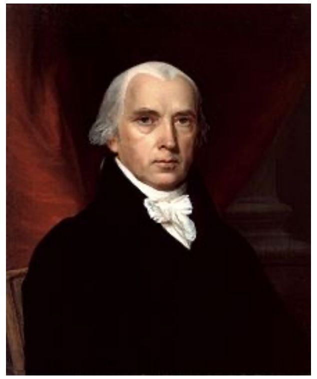
Alexander Hamilton, James Madison, and John Jay

to convince New York to ratify U.S Constitution

Authorship of 12 of the letters unclear between:



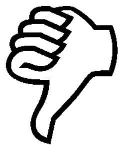
Alexander Hamilton



James Madison

1963: solved by Mosteller and Wallace using Bayesian methods

Positive or negative movie review?

-  unbelievably disappointing
-  Full of zany characters and richly applied satire, and some great plot twists
-  this is the greatest screwball comedy ever filmed
-  It was pathetic. The worst part about it was the boxing scenes.

What is the subject of this article?

MEDLINE Article


Brain and Cognition

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

?

MeSH Subject Category Hierarchy

Antogonists and Inhibitors

Blood Supply

Chemistry

Drug Therapy

Embryology

Epidemiology

…

Text Classification

Assigning subject categories, topics, or genres

Spam detection

Authorship identification (who wrote this?)

Language Identification (is this Portuguese?)

Sentiment analysis

…

Text Classification: definition

Input:

- a document d
- a fixed set of classes $C = \{c_1, c_2,\ldots, c_J\}$

Output: a predicted class $c \in C$

Basic Classification Method: Hand-coded rules

Rules based on combinations of words or other features

- spam: black-list-address OR (“dollars” AND “have been selected”)

Accuracy can be high

- In very specific domains
- If rules are carefully refined by experts

But:

- building and maintaining rules is expensive
- they are too literal and specific: "high-precision, low-recall"

Classification Method: Supervised Machine Learning

Input:

- a document d
- a fixed set of classes $C = \{c_1, c_2,\ldots, c_J\}$
- A training set of m hand-labeled documents $(d_1,c_1),\ldots,(d_m,c_m)$

Output:

- a learned classifier $\gamma{:}d \rightarrow c$

Classification Methods: Supervised Machine Learning

Many kinds of classifiers!

- Naïve Bayes (this lecture)
- Logistic regression
- Neural networks
- *k*-nearest neighbors
- …

We can also use pretrained large language models!

- Fine-tuned as classifiers
- Prompted to give a classification

The Naive Bayes Classifier

Naive Bayes Intuition

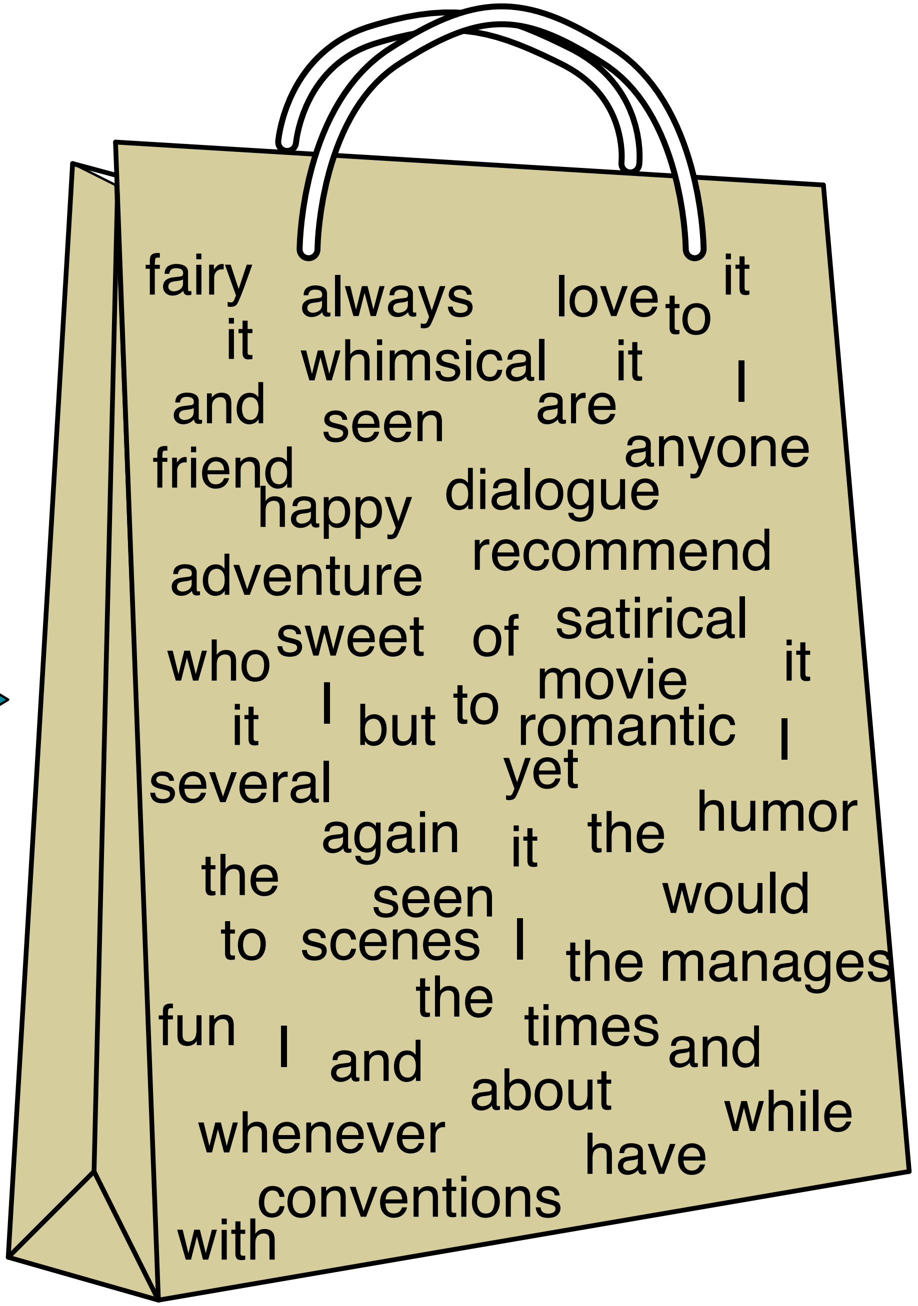
Simple ("naive") classification method based on Bayes rule

Relies on very simple representation of document

- **Bag of words**

The Bag of Words 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!



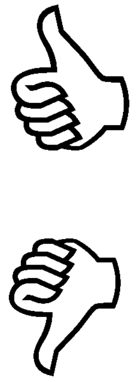
it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
…	…

The bag of words representation

$\gamma($

seen	2
sweet	1
whimsical	1
recommend	1
happy	1
...	...

$)=c$



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 Classifier (I)

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

MAP is "maximum a posteriori" = most likely class

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

Bayes Rule

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

Dropping the denominator

Naive Bayes Classifier (II)

"Likelihood" "Prior"

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

$$= \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

Document d represented as features x1..xn

Naïve Bayes Classifier (IV)

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

$O(|X|^n \bullet |C|)$ parameters

How often does this class occur?

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

We can just count the relative frequencies in a corpus

Multinomial Naive Bayes Independence Assumptions

$$P(x_1, x_2, \ldots, x_n \mid c)$$

Bag of Words assumption: Assume position doesn't matter

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

$$P(x_1, \ldots, x_n \mid c) = P(x_1 \mid c) \bullet P(x_2 \mid c) \bullet P(x_3 \mid c) \bullet \ldots \bullet P(x_n \mid c)$$

Multinomial Naive Bayes Classifier

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c_j) \prod_{x \in X} P(x \mid c)$$

Applying Multinomial Naive Bayes Classifiers to Text Classification

positions $\leftarrow$ all word positions in test document

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}}\, P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

Problems with multiplying lots of probs

There's a problem with this:

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

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

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

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**

The Naive Bayes Classifier

Naive Bayes: Learning

Learning the Multinomial Naive Bayes Model

First attempt: maximum likelihood estimates

- simply use the frequencies in the data

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

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

Parameter estimation

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

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

Create mega-document for topic *j* by concatenating all docs in this topic

- Use frequency of *w* in mega-document

Problem with Maximum Likelihood

What if we have seen no training documents with the word ***fantastic*** and classified in the topic **positive** (***thumbs-up)***?

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

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

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

Laplace (add-1) smoothing for Naïve Bayes

$$\hat{P}(w_i \mid c) = \frac{count(w_i, c) + 1}{\sum_{w \in V} \left(count(w, c) + 1\right)}$$

$$= \frac{count(w_i, c) + 1}{\left(\sum_{w \in V} count(w, c)\right) + |V|}$$

Multinomial Naïve Bayes: Learning

- From training corpus, extract *Vocabulary*

Calculate $P(c_j)$ terms

- For each c_j in C do

  $docs_j \leftarrow$ all docs with class $= c_j$

$$P(c_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$$

- Calculate $P(w_k \mid c_j)$ terms
  - $Text_j \leftarrow$ single doc containing all $docs_j$
  - For each word w_k in *Vocabulary*

    $n_k \leftarrow$ # of occurrences of w_k in $Text_j$

$$P(w_k \mid c_j) \leftarrow \frac{n_k + \alpha}{n + \alpha |Vocabulary|}$$

Unknown words

What about unknown words

- that appear in our test data
- but not in our training data or vocabulary?

We **ignore** them

- Remove them from the test document!
- Pretend they weren't there!
- Don't include any probability for them at all!

Why don't we build an unknown word model?

- It doesn't help: knowing which class has more unknown words is not generally helpful!

Stop words

Some systems ignore stop words

- **Stop words:** very frequent words like *the* and *a*.
  - Sort the vocabulary by word frequency in training set
  - Call the top 10 or 50 words the **stopword list**.
  - Remove all stop words from both training and test sets
    - As if they were never there!

But removing stop words doesn't usually help

- So in practice most NB algorithms use **all** words and **don't** use stopword lists

Naive Bayes: Learning

Text Classification and Naive Bayes

Sentiment and Binary Naive Bayes

Let's do a worked sentiment example!

	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

A worked sentiment example with add-1 smoothing

	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

1. Prior from training:

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

P(-) = 3/5
P(+) = 2/5

2. Drop "with"

3. Likelihoods from training:

$$p(w_i|c) = \frac{count(w_i, c) + 1}{(\sum_{w \in V} count(w, c)) + |V|}$$

$$P(\text{“predictable”}|-) = \frac{1+1}{14+20} \qquad P(\text{“predictable”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“no”}|-) = \frac{1+1}{14+20} \qquad P(\text{“no”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“fun”}|-) = \frac{0+1}{14+20} \qquad P(\text{“fun”}|+) = \frac{1+1}{9+20}$$

4. Scoring the test set:

$$\boxed{P(-)P(S|-) = \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} = 6.1 \times 10^{-5}}$$

$$P(+)P(S|+) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

Optimizing for sentiment analysis

For tasks like sentiment, word **occurrence** seems to be more important than word **frequency**.

- The occurrence of the word *fantastic* tells us a lot
- The fact that it occurs 5 times may not tell us much more.

Binary multinominal naive bayes, or **binary NB**

- Clip our word counts at 1
- Note: this is different than Bernoulli naive bayes; see the textbook at the end of the chapter.

Binary Multinomial Naïve Bayes: Learning

- From training corpus, extract *Vocabulary*

Calculate $P(c_j)$ terms

- For each c_j in C do

  $docs_j \leftarrow$ all docs with class $= c_j$

$$P(c_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$$

- Calculate $P(w_k \mid c_j)$ terms
  - Remove duplicates in each doc:
    - For each word type w in doc_j
      - Retain only a single instance of w
  - $Text_j \leftarrow$ single doc containing all $docs_j$
  - For each word w_k in *Vocabulary*

    $n_k \leftarrow$ # of occurrences of w_k in $Text_j$

$$P(w_k \mid c_j) \leftarrow \frac{n_k + \alpha}{n + \alpha |Vocabulary|}$$

Binary Multinomial Naive Bayes on a test document *d*

First remove all duplicate words from *d*

Then compute NB using the same equation:

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(w_i | c_j)$$

Binary multinominal naive Bayes

Four original documents:

- − it was pathetic the worst part was the boxing scenes
- − no plot twists or great scenes
- \+ and satire and great plot twists
- \+ great scenes great film

Binary multinominal naive Bayes

Four original documents:

– it was pathetic the worst part was the boxing scenes
– no plot twists or great scenes
\+ and satire and great plot twists
\+ great scenes great film

	NB Counts +	NB Counts −
and	2	0
boxing	0	1
film	1	0
great	3	1
it	0	1
no	0	1
or	0	1
part	0	1
pathetic	0	1
plot	1	1
satire	1	0
scenes	1	2
the	0	2
twists	1	1
was	0	2
worst	0	1

Binary multinominal naive Bayes

Four original documents:

– it was pathetic the worst part was the boxing scenes
– no plot twists or great scenes
+ and satire and great plot twists
+ great scenes great film

After per-document binarization:

– it was pathetic the worst part boxing scenes
– no plot twists or great scenes
+ and satire great plot twists
+ great scenes film

	NB Counts +	NB Counts −
and	2	0
boxing	0	1
film	1	0
great	3	1
it	0	1
no	0	1
or	0	1
part	0	1
pathetic	0	1
plot	1	1
satire	1	0
scenes	1	2
the	0	2
twists	1	1
was	0	2
worst	0	1

Binary multinominal naive Bayes

Four original documents:

- – it was pathetic the worst part was the boxing scenes
- – no plot twists or great scenes
- + and satire and great plot twists
- + great scenes great film

After per-document binarization:

- – it was pathetic the worst part boxing scenes
- – no plot twists or great scenes
- + and satire great plot twists
- + great scenes film

	NB Counts +	NB Counts −	Binary Counts +	Binary Counts −
and	2	0	1	0
boxing	0	1	0	1
film	1	0	1	0
great	3	1	2	1
it	0	1	0	1
no	0	1	0	1
or	0	1	0	1
part	0	1	0	1
pathetic	0	1	0	1
plot	1	1	1	1
satire	1	0	1	0
scenes	1	2	1	2
the	0	2	0	1
twists	1	1	1	1
was	0	2	0	1
worst	0	1	0	1

Counts can still be 2! Binarization is within-doc!

Text Classification and Naive Bayes

Sentiment and Binary Naive Bayes

Text Classification and Naive Bayes

More on Sentiment Classification

Sentiment Classification: Dealing with Negation

I really like this movie

I really **don't** like this movie

Negation changes the meaning of "like" to negative.

Negation can also change negative to positive-ish

- **Don't** dismiss this film
- **Doesn't** let us get bored

Sentiment Classification: Dealing with Negation

Das, Sanjiv and Mike Chen. 2001. Yahoo! for Amazon: Extracting market sentiment from stock message boards. In Proceedings of the Asia Pacific Finance Association Annual Conference (APFA).
Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. EMNLP-2002, 79—86.

Simple baseline method:

Add NOT_ to every word between negation and following punctuation:

```
didn't like this movie , but I
```

↓

```
didn't NOT_like NOT_this NOT_movie but I
```

Sentiment Classification: Lexicons

Sometimes we don't have enough labeled training data

In that case, we can make use of pre-built word lists

Called **lexicons**

There are various publically available lexicons

MPQA Subjectivity Cues Lexicon

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann (2005). Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. Proc. of HLT-EMNLP-2005.

Riloff and Wiebe (2003). Learning extraction patterns for subjective expressions. EMNLP-2003.

Home page: https://mpqa.cs.pitt.edu/lexicons/subj_lexicon/

6885 words from 8221 lemmas, annotated for intensity (strong/weak)

- 2718 positive
- 4912 negative

+ : *admirable, beautiful, confident, dazzling, ecstatic, favor, glee, great*

– : *awful, bad, bias, catastrophe, cheat, deny, envious, foul, harsh, hate*

The General Inquirer

Philip J. Stone, Dexter C Dunphy, Marshall S. Smith, Daniel M. Ogilvie. 1966. The General Inquirer: A Computer Approach to Content Analysis. MIT Press

- Home page: http://www.wjh.harvard.edu/~inquirer
- List of Categories: http://www.wjh.harvard.edu/~inquirer/homecat.htm
- Spreadsheet: http://www.wjh.harvard.edu/~inquirer/inquirerbasic.xls

Categories:

- Positiv (1915 words) and Negativ (2291 words)
- Strong vs Weak, Active vs Passive, Overstated versus Understated
- Pleasure, Pain, Virtue, Vice, Motivation, Cognitive Orientation, etc

Free for Research Use

Using Lexicons in Sentiment Classification

Add a feature that gets a count whenever a word from the lexicon occurs

- E.g., a feature called "**this word occurs in the positive lexicon**" or "**this word occurs in the negative lexicon**"

Now all positive words (*good, great, beautiful, wonderful*) or negative words count for that feature.

Using 1-2 features isn't as good as using all the words.

- But when training data is sparse or not representative of the test set, dense lexicon features can help

Naive Bayes in Other tasks: Spam Filtering

SpamAssassin Features:

- Mentions millions of (dollar) ((dollar) NN,NNN,NNN.NN)
- From: starts with many numbers
- Subject is all capitals
- HTML has a low ratio of text to image area
- "One hundred percent guaranteed"
- Claims you can be removed from the list

Naive Bayes in Language ID

Determining what language a piece of text is written in.

Features based on character n-grams do very well

Important to train on lots of varieties of each language

(e.g., American English varieties like African-American English, or English varieties around the world like Indian English)

Summary: Naive Bayes is Not So Naive

Very Fast, low storage requirements

Work well with very small amounts of training data

Robust to Irrelevant Features

Irrelevant Features cancel each other without affecting results

Very good in domains with many equally important features

Decision Trees suffer from *fragmentation* in such cases – especially if little data

Optimal if the independence assumptions hold: If assumed independence is correct, then it is the Bayes Optimal Classifier for problem

A good dependable baseline for text classification

- **But we will see other classifiers that give better accuracy**

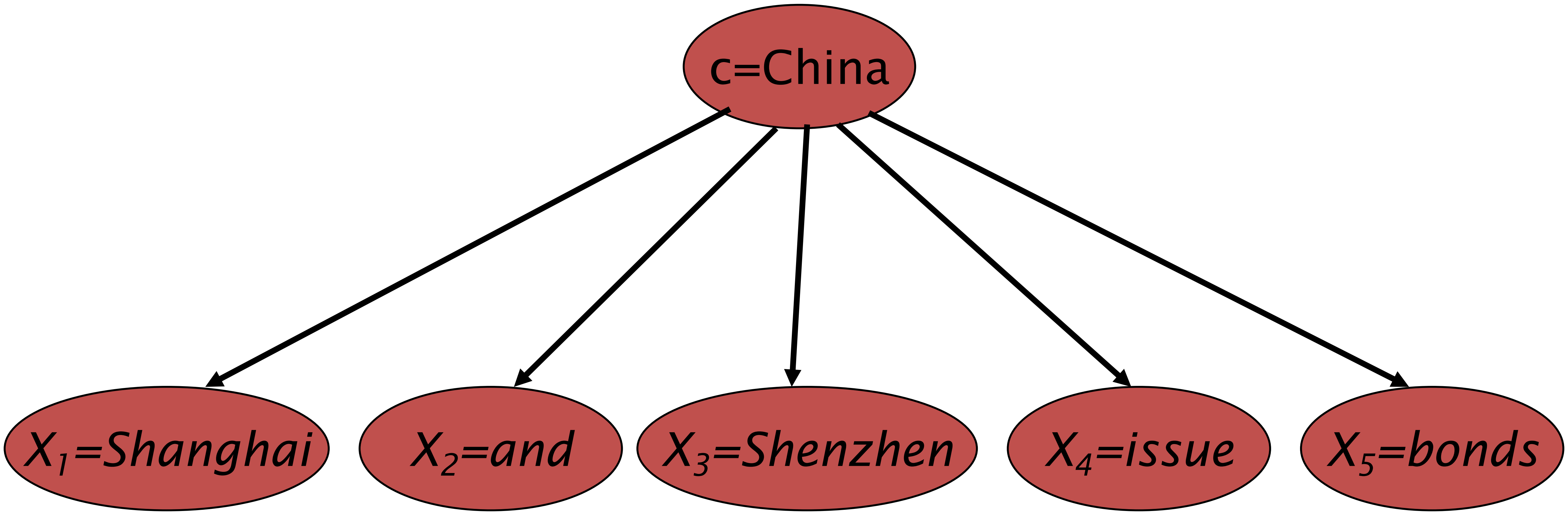
Slide from Chris Manning

More on Sentiment Classification

Text Classification and Naïve Bayes

Naïve Bayes: Relationship to Language Modeling

Generative Model for Multinomial Naïve Bayes



Naïve Bayes and Language Modeling

- Naïve bayes classifiers can use any sort of feature
  - URL, email address, dictionaries, network features
- But if, as in the previous slides
  - We use **only** word features
  - we use **all** of the words in the text (not a subset)
- Then
  - Naïve bayes has an important similarity to language modeling.

Each class = a unigram language model

- Assigning each word: P(word | c)
- Assigning each sentence: $P(s|c)=\Pi\ P(\text{word}|c)$

Class *pos*

0.1	I
0.1	love
0.01	this
0.05	fun
0.1	film

I	love	this	fun	film
0.1	0.1	.05	0.01	0.1

P(s | pos) = 0.0000005

Naïve Bayes as a Language Model

- Which class assigns the higher probability to s?

Model pos	
0.1	I
0.1	love
0.01	this
0.05	fun
0.1	film

Model neg	
0.2	I
0.001	love
0.01	this
0.005	fun
0.1	film

I	love	this	fun	film
0.1	0.1	0.01	0.05	0.1
0.2	0.001	0.01	0.005	0.1

$P(s|pos) > P(s|neg)$

Text Classification and Naïve Bayes

Naïve Bayes: Relationship to Language Modeling

Precision, Recall, and F1

Evaluating Classifiers: How well does our classifier work?

Let's first address binary classifiers:

- Is this email spam?

  spam (+) or not spam (-)

- Is this post about Delicious Pie Company?

  about Del. Pie Co (+) or not about Del. Pie Co(-)

We'll need to know

1. What did our classifier say about each email or post?
2. What should our classifier have said, i.e., the correct answer, usually as defined by humans ("gold label")

First step in evaluation: The confusion matrix

		gold standard labels	
		gold positive	gold negative
system output labels	system positive	**true positive**	**false positive**
	system negative	**false negative**	**true negative**

Accuracy on the confusion matrix

gold standard labels

system output labels		gold positive	gold negative
	system positive	**true positive**	**false positive**
	system negative	**false negative**	**true negative**

$$\textbf{accuracy} = \frac{\text{tp+tn}}{\text{tp+fp+tn+fn}}$$

Why don't we use accuracy?

Accuracy doesn't work well when we're dealing with uncommon or imbalanced classes

Suppose we look at 1,000,000 social media posts to find Delicious Pie-lovers (or haters)

- 100 of them talk about our pie
- 999,900 are posts about something unrelated

Imagine the following simple classifier

Every post is "not about pie"

Accuracy re: pie posts

100 posts are about pie; 999,900 aren't

			gold standard labels	
			gold positive	gold negative
system output labels	system positive		**true positive**	**false positive**
	system negative		**false negative**	**true negative**

$$\textbf{accuracy} = \frac{\text{tp+tn}}{\text{tp+fp+tn+fn}}$$

Why don't we use accuracy?

Accuracy of our "nothing is pie" classifier

- 999,900 true negatives and 100 false negatives
- Accuracy is 999,900/1,000,000 = 99.99%!
- But useless at finding pie-lovers (or haters)!!
- Which was our goal!

Accuracy doesn't work well for unbalanced classes

- Most tweets are not about pie!

Instead of accuracy we use precision and recall

gold standard labels

		gold positive	gold negative	
system output labels	system positive	**true positive**	**false positive**	**precision** $= \frac{tp}{tp+fp}$
	system negative	**false negative**	**true negative**	
		recall $= \frac{tp}{tp+fn}$		**accuracy** $= \frac{tp+tn}{tp+fp+tn+fn}$

Precision: % of selected items that are correct

Recall: % of correct items that are selected

Precision/Recall aren't fooled by the"just call everything negative" classifier!

Stupid classifier: Just say no: every tweet is "not about pie"

- 100 tweets talk about pie,   999,900 tweets don't
- Accuracy = 999,900/1,000,000 = 99.99%

But the Recall and Precision for this classifier are terrible:

$$\textbf{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

$$\textbf{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

A combined measure: F1

F1 is a combination of precision and recall.

$$\mathrm{F}_1 = \frac{2PR}{P+R}$$

F1 is a special case of the general "F-measure"

F-measure is the (weighted) harmonic mean of precision and recall

$$\mathrm{HarmonicMean}(\mathrm{a}_1, \mathrm{a}_2, \mathrm{a}_3, \mathrm{a}_4, \ldots, \mathrm{a}_\mathrm{n}) = \frac{\mathrm{n}}{\frac{1}{\mathrm{a}_1} + \frac{1}{\mathrm{a}_2} + \frac{1}{\mathrm{a}_3} + \ldots + \frac{1}{\mathrm{a}_\mathrm{n}}}$$

$$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha)\frac{1}{R}} \quad \text{or} \left(\text{with } \beta^2 = \frac{1-\alpha}{\alpha}\right) \quad F = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

F1 is a special case of F-measure with β=1, α=½

Suppose we have more than 2 classes?

Lots of text classification tasks have more than two classes.

- Sentiment analysis (positive, negative, neutral) , named entities (person, location, organization)

We can define precision and recall for multiple classes like this 3-way email task:

system output \ *gold labels*	urgent	normal	spam	
urgent	8	10	1	$\textbf{precision}_u = \frac{8}{8+10+1}$
normal	5	60	50	$\textbf{precision}_n = \frac{60}{5+60+50}$
spam	3	30	200	$\textbf{precision}_s = \frac{200}{3+30+200}$
	$\textbf{recall}_u = \frac{8}{8+5+3}$	$\textbf{recall}_n = \frac{60}{10+60+30}$	$\textbf{recall}_s = \frac{200}{1+50+200}$	

How to combine P/R values for different classes: Microaveraging vs Macroaveraging

Class 1: Urgent

	true urgent	true not
system urgent	8	11
system not	8	340

$$\text{precision} = \frac{8}{8+11} = .42$$

Class 2: Normal

	true normal	true not
system normal	60	55
system not	40	212

$$\text{precision} = \frac{60}{60+55} = .52$$

Class 3: Spam

	true spam	true not
system spam	200	33
system not	51	83

$$\text{precision} = \frac{200}{200+33} = .86$$

Pooled

	true yes	true no
system yes	268	99
system no	99	635

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

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

Text Classification and Naive Bayes

Precision, Recall, and F1

Text Classification and Naive Bayes

Avoiding Harms in Classification

Harms of classification

Classifiers, like any NLP algorithm, can cause harms

This is true for any classifier, whether Naive Bayes or other algorithms

Representational Harms

- Harms caused by a system that demeans a social group
  - Such as by perpetuating negative stereotypes about them.
- Kiritchenko and Mohammad 2018 study
  - Examined 200 sentiment analysis systems on pairs of sentences
  - Identical except for names:
    - common African American (Shaniqua) or European American (Stephanie).
    - Like "I talked to Shaniqua yesterday" vs "I talked to Stephanie yesterday"
- Result: systems assigned lower sentiment and more negative emotion to sentences with African American names
- Downstream harm:
  - Perpetuates stereotypes about African Americans
  - African Americans treated differently by NLP tools like sentiment (widely used in marketing research, mental health studies, etc.)

Harms of Censorship

- **Toxicity detection** is the text classification task of detecting hate speech, abuse, harassment, or other kinds of toxic language.
  - Widely used in online content moderation
- Toxicity classifiers incorrectly flag non-toxic sentences that simply mention minority identities (like the words "blind" or "gay")
  - women (Park et al., 2018),
  - disabled people (Hutchinson et al., 2020)
  - gay people (Dixon et al., 2018; Oliva et al., 2021)
- Downstream harms:
  - Censorship of speech by disabled people and other groups
  - Speech by these groups becomes less visible online
  - Writers might be nudged by these algorithms to avoid these words making people less likely to write about themselves or these groups.

Performance Disparities

1. Text classifiers perform worse on many **languages** of the world due to lack of data or labels
2. Text classifiers perform worse on **varieties** of even high-resource languages like English
   - Example task: language identification, a first step in NLP pipeline ("Is this post in English or not?")
   - English language detection performance worse for writers who are African American (Blodgett and O'Connor 2017) or from India (Jurgens et al., 2017)

Harms in text classification

- **Causes:**
  - Issues in the data; NLP systems amplify biases in training data
  - Problems in the labels
  - Problems in the algorithms (like what the model is trained to optimize)
- **Prevalence**: The same problems occur throughout NLP (including large language models)
- **Solutions**: There are no general mitigations or solutions
  - But harm mitigation is an active area of research
  - And there are standard benchmarks and tools that we can use for measuring some of the harms

Text Classification and Naive Bayes

Avoiding Harms in Classification