CS 463
NATURAL LANGUAGE PROCESSING

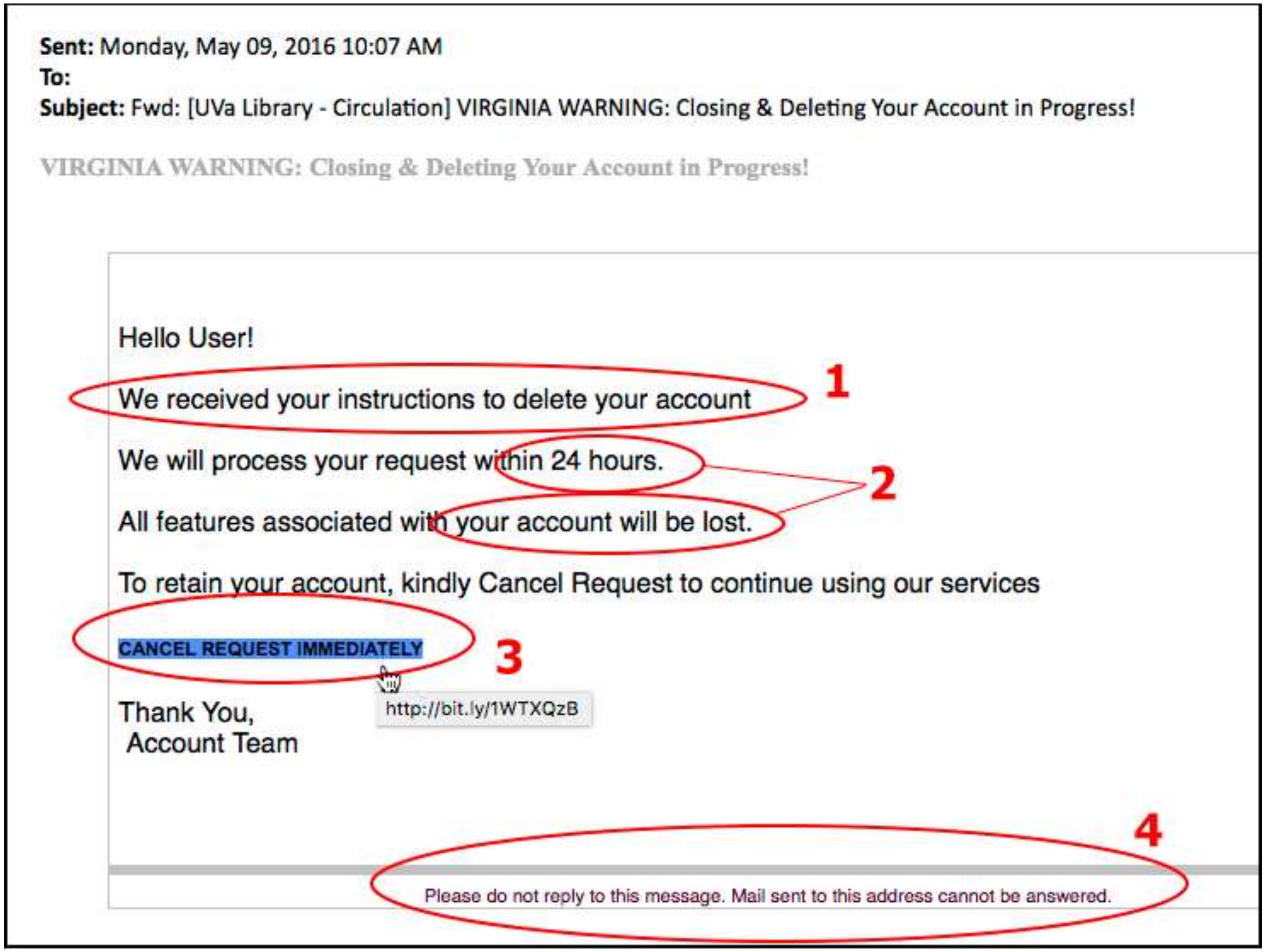
Dr. Saleh Haridy

2022-2023

Week 3

Text Classification and Naive Bayes Classifier

Is this a spam email?



Sent: Monday, May 09, 2016 10:07 AM
To:
Subject: Fwd: [UVa Library - Circulation] VIRGINIA WARNING: Closing & Deleting Your Account in Progress!

VIRGINIA WARNING: Closing & Deleting Your Account in Progress!

Hello User!

We received your instructions to delete your account 1

We will process your request within 24 hours.

All features associated with your account will be lost. 2

To retain your account, kindly Cancel Request to continue using our services

CANCEL REQUEST IMMEDIATELY 3

http://bit.ly/1WTXQzB

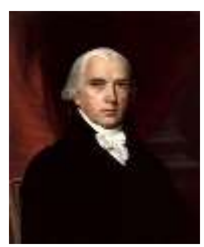
Thank You,
Account Team

4

Please do not reply to this message. Mail sent to this address cannot be answered.

Who wrote this document?

- 1787-8: 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?

Subject Category Hierarchy

MEDLINE Article


Brain and Cognition

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

?

- Antogonists and Inhibitors
- Blood Supply
- Chemistry
- Drug Therapy
- Embryology
- Epidemiology
- …

Positive or negative movie review?

+ *...zany characters and **richly** applied satire, and some **great** plot twists*

_ *It was **sad**. The **worst** part about it was the boxing scenes...*

+ *...**awesome** caramel sauce and sweet toasty almonds. I **love** this place!*

_ *...**awful** pizza and **ridiculously** overpriced...*

Application of Text Classification

- Sentiment analysis
- Spam detection
- Age/gender classification
- Authorship identification
- Language Identification
- Assigning subject categories, topics, or genres
- …

Why sentiment analysis?

- *Movie*: is this review positive or negative?
- *Products*: what do people think about the new iPhone?
- *Politics*: what do people think about this candidate or issue?
- *Prediction*: predict market trends from sentiment

Text Classification: definition

- *Input*:
  - a document d
  - a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$
- *Output*: a predicted class $c \in C$

Classification Methods: rule-based method

- Rules based on combinations of words or other features
  - spam: black-list-address OR ("dollars" AND "you have been selected")
- Accuracy can be high
  - If rules carefully refined by expert
- But building and maintaining these rules is expensive and time consuming

Classification Methods: 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

- Any kind of classifier
  - Naïve Bayes
  - Logistic regression
  - Neural networks
  - Support-vector machine (SVM)
  - k-Nearest Neighbors (KNN)
  - …

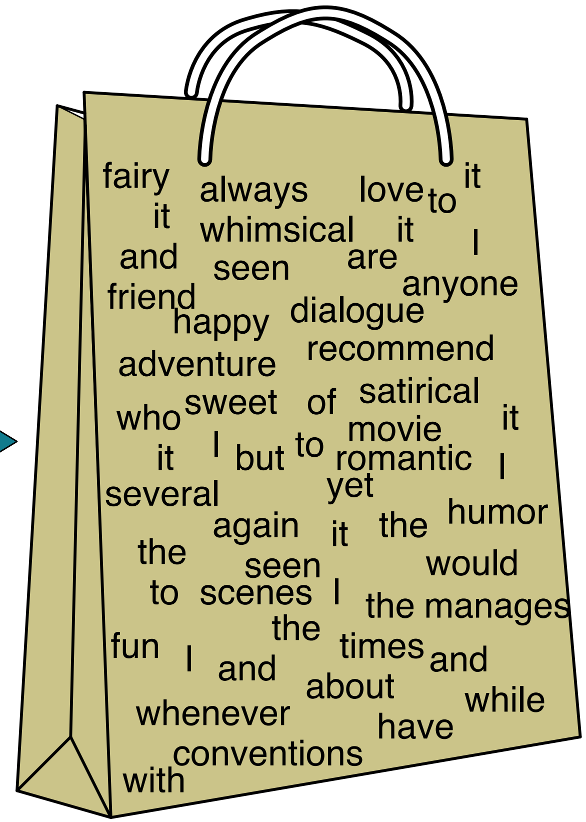
Text Classification and Naive Bayes

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

f(

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*

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

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

Drop P(d) because it has same value for all classes

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•|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

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

Compute the probability for each class
Find the class which has maximum probability

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

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**

Learning the Multinomial Naive Bayes Model

Sec.13.3

- First attempt: maximum likelihood estimates
  - simply use the frequencies in the data

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

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

Count number of times word wi occurs in class j

total number of words in class 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**?

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

- 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

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

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

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

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:

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

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

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

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

4. Scoring the test set:

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

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

- Number of words in -ve class= 14
- Number of words in +ve class= 9
- Total number of words=20

Optimizing for sentiment analysis

For tasks like sentiment, word **occurrence** seems to be more important than word **frequency** **(how many times occured)**.

- 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

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:

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

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

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		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!

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

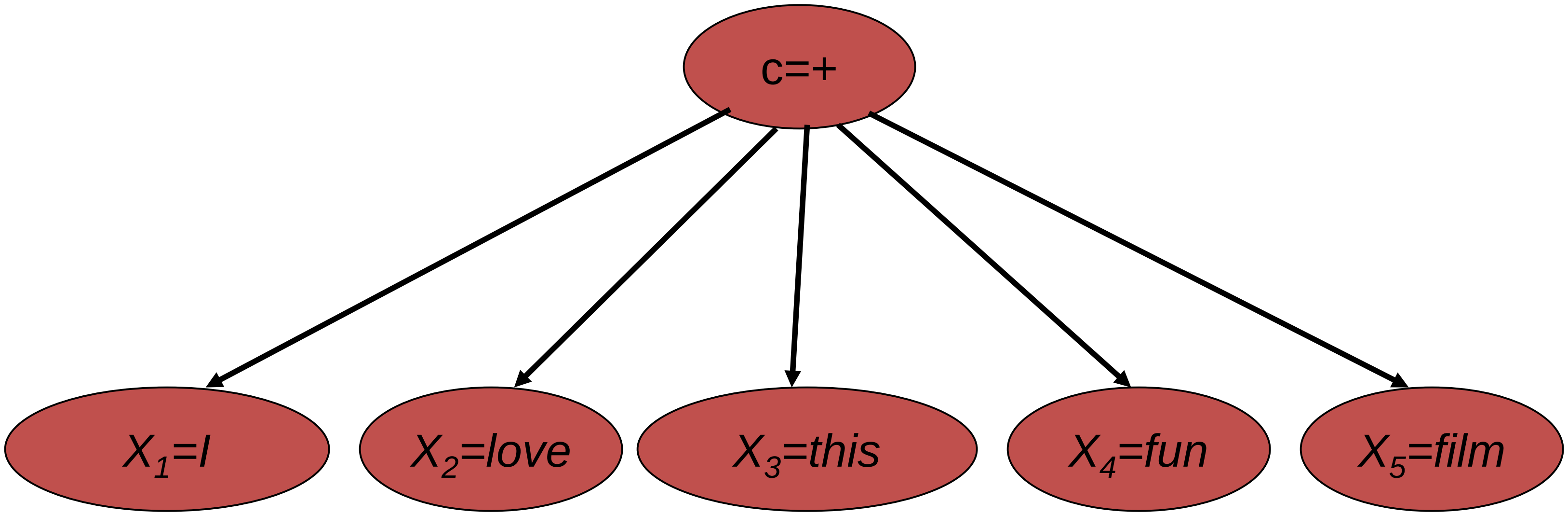
- Very Fast, low storage requirements
- Work well with very small amounts of training data
- Robust to Irrelevant Features

  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**

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
  - Naive bayes has an important similarity to language modeling.

Each class = a unigram language model

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

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

Sec.13.2.1

- Which class assigns the higher probability to a document 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

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

Evaluation

- Let's consider just binary text classification tasks
- Imagine you're the CEO of Delicious PIZZA Company
- You want to know what people are saying about your PIZZA
- So you build a "Delicious PIZZA " tweet detector
  - Positive class: tweets about Delicious PIZZA Co
  - Negative class: all other tweets

The 2-by-2 confusion matrix

		Actual Label		
		Actual Positive	Actual Negative	
system output labels	system positive	**true positive**	**false positive**	$\textbf{precision} = \frac{\text{tp}}{\text{tp+fp}}$
	system negative	**false negative**	**true negative**	
		$\textbf{recall} = \frac{\text{tp}}{\text{tp+fn}}$		$\textbf{accuracy} = \frac{\text{tp+tn}}{\text{tp+fp+tn+fn}}$

Evaluation: Accuracy

- Why don't we use **accuracy** as our metric?
- Imagine we saw 1 million tweets
  - 100 of them talked about Delicious PIZZA Co.
  - 999,900 talked about something else
- We could build a dumb classifier that just labels every tweet "not about PIZZA"
  - It would get 99.99% accuracy!!! Wow!!!!
  - But useless! Doesn't return the comments we are looking for!
  - That's why we use **precision** and **recall** instead

Evaluation: Precision

- % of items actually present in the input that were correctly identified by the system.

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

Evaluation: Recall

- % of items the system detected (i.e., items the system labeled as positive) that are in fact positive (according to the correct labels)

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

Why Precision and recall

- Our dumb PIZZA-classifier
  - Just label nothing as "about PIZZA"

Accuracy=99.99%

but

Recall = 0

- (it doesn't get any of the 100 PIZZA tweets)

Precision and recall, unlike accuracy, emphasize true positives:

- finding the things that we are supposed to be looking for.

A combined measure: F

- F measure: a single number that combines P and R:

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

- We almost always use balanced F_1 (i.e., $\beta = 1$)

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

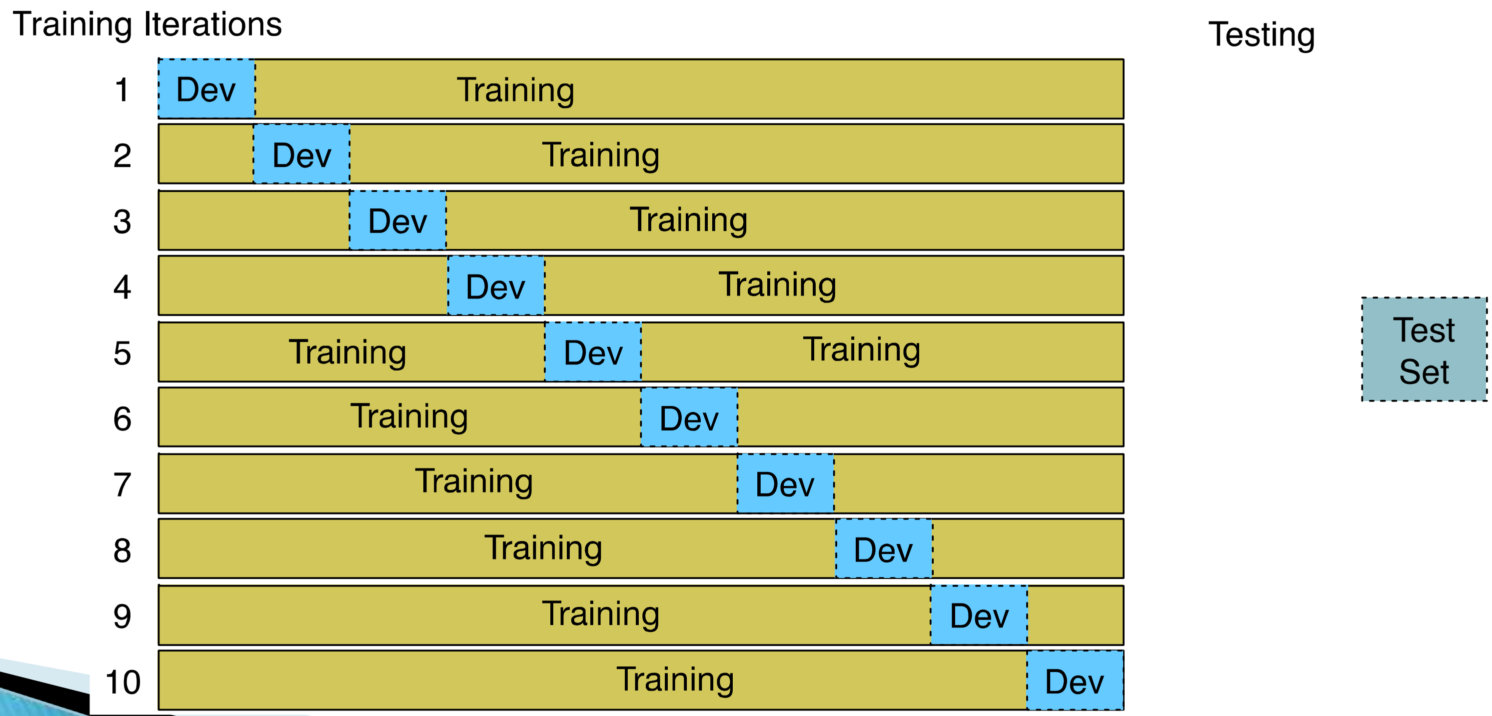
Development Test Sets ("Devsets") and Cross-validation

Training set | Development Test Set | Test Set

- Train on training set, tune on devset, report on testset
  - This avoids overfitting ('tuning to the test set')
  - More conservative estimate of performance
  - But paradox: want as much data as possible for training, and as much for dev; how to split?

Cross-validation: multiple splits

- Pool results over splits, Compute pooled dev performance



Confusion Matrix for 3-class classification

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

How to combine P/R from 3 classes to get one metric

- Macroaveraging:
  - compute the performance for each class, and then average over classes
- Microaveraging:
  - collect decisions for all classes into one confusion matrix
  - compute precision and recall from that table.

Macroaveraging and Microaveraging

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

$$\begin{array}{c}\text{microaverage}\\\text{precision}\end{array} = \frac{268}{268+99} = \mathbf{.73}$$

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