Web Science: Document Filtering

(Part 1 - Intro to Classifiers)

CS 432/532

Old Dominion University

Permission has been granted to use these slides from Frank McCown, Michael L. Nelson, Alexander Nwala, Michele C. Weigle

Main references:

Ch 6 from *Programming Collective Intelligence* by Toby Segaran

(abbreviated as PCI)

Ch 9 from *Search Engines: Information Retrieval in Practice* by Croft, Metzler, and Strohman

(*abbreviated as SEIRP)*

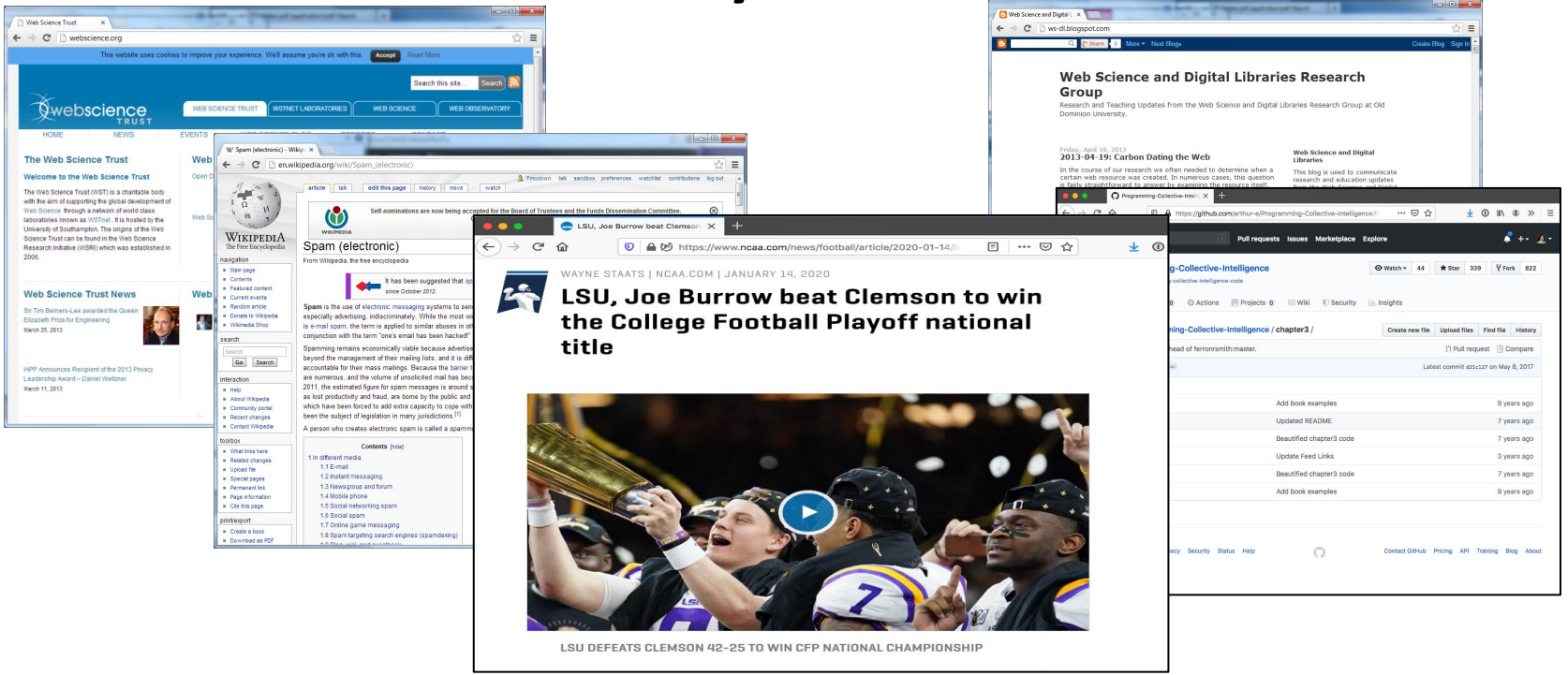
Clustering vs. Classification

- Clustering
  - grouping related items together
  - clusters may not correspond to a meaningful concept
  - uses unsupervised learning techniques

- Classification
  - task of automatically applying predefined labels to data
  - uses supervised learning techniques

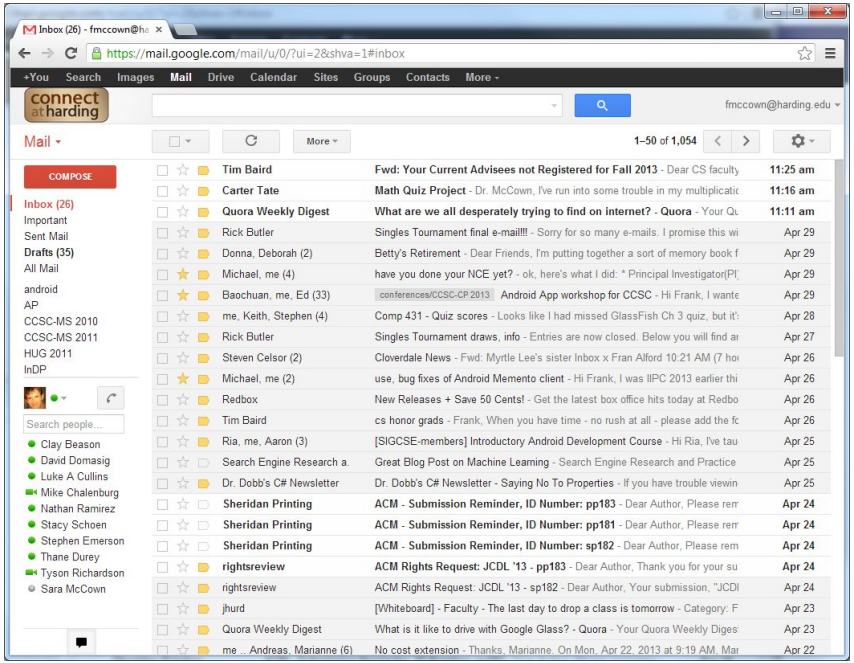
SEIRP, Ch 9

Can we classify these documents?



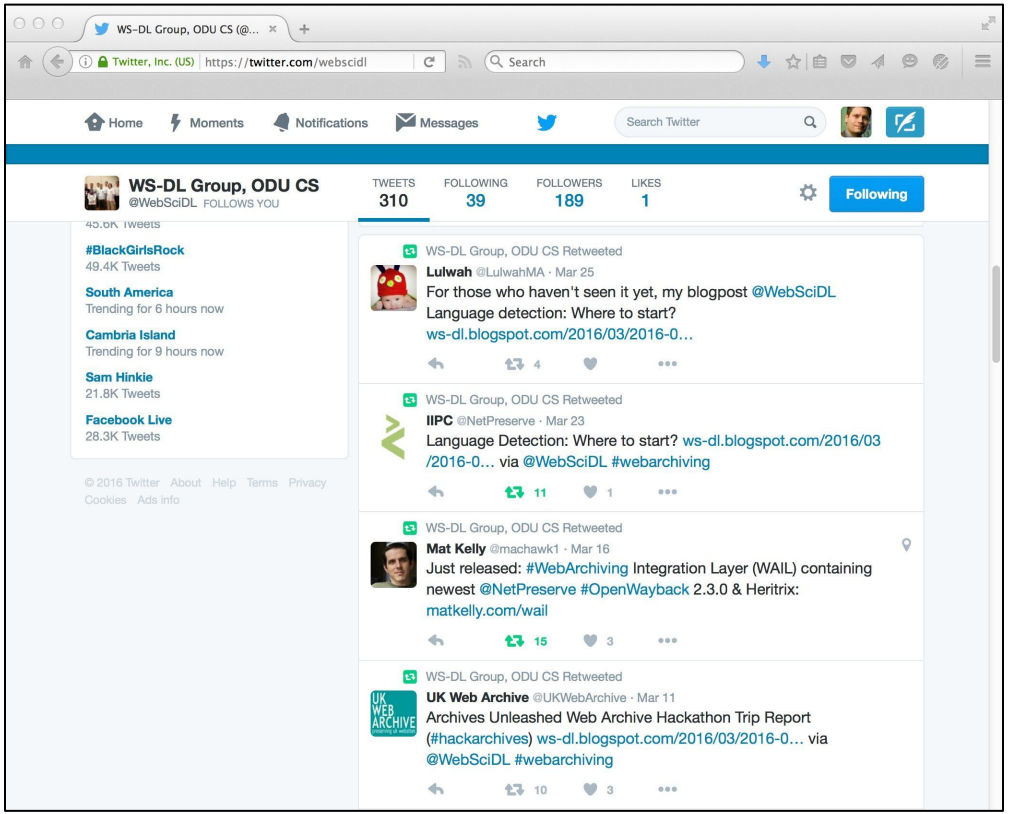
Science, Leisure, Programming, etc.

Can we classify these documents?

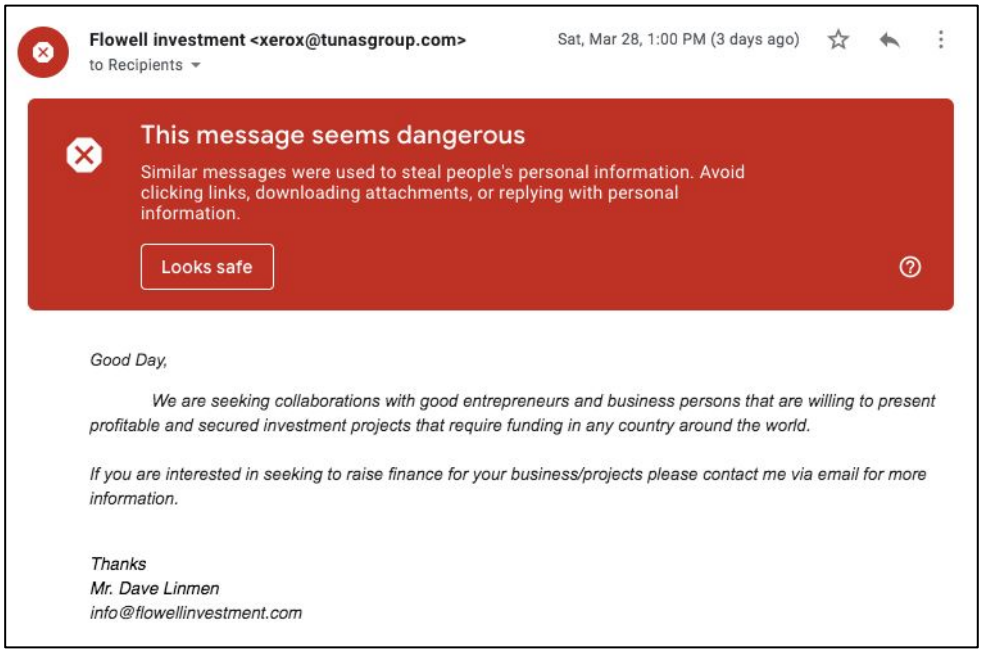


Important, Work, Personal, Spam, etc.

How about very small documents?

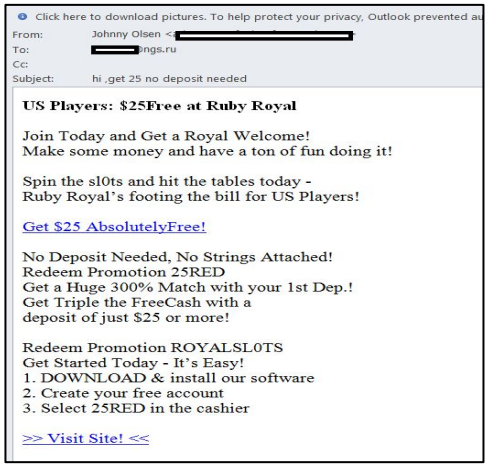


We're Going to Look at Spam Email



Rule-Based Classifiers Are Inadequate

If my email has the word "spam", is the message about...



Click here to download pictures. To help protect your privacy, Outlook prevented au

From: Johnny Olsen <[redacted]>
To: [redacted]ngs.ru
Cc:
Subject: hi ,get 25 no deposit needed

US Players: $25Free at Ruby Royal

Join Today and Get a Royal Welcome!
Make some money and have a ton of fun doing it!

Spin the sl0ts and hit the tables today -
Ruby Royal's footing the bill for US Players!

Get $25 AbsolutelyFree!

No Deposit Needed, No Strings Attached!
Redeem Promotion 25RED
Get a Huge 300% Match with your 1st Dep.!
Get Triple the FreeCash with a
deposit of just $25 or more!

Redeem Promotion ROYALSL0TS
Get Started Today - It's Easy!
1. DOWNLOAD & install our software
2. Create your free account
3. Select 25RED in the cashier

>> Visit Site! <<

Spam - Monty Python's Flying Circus
Spam (Monty Python) (Wikipedia)
Why is it called Python?

Rule-based classifiers don't consider *context*

How Do Humans Classify Things?

- We identify a number of important *features* that distinguish between possible categories.
- We *extract* these features from each item.
- We then *combine evidence* from the extracted features in some way.
- Finally, we *classify* the item using some decision mechanism based on the combined evidence.

SEIRP, Ch 9

Supervised Learning

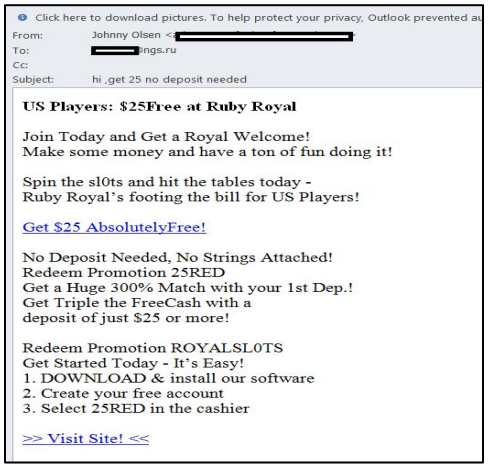
- Classification is a supervised learning problem.

- Training set
  - set of fully labeled items (items and their classification) used to build the model
- Testing set
  - set of unlabeled items given to the model for automatic classification

SEIRP, Ch 9

Classifying with Supervised Learning

We "teach" the program to learn the difference between spam as unsolicited bulk email, luncheon meat, and comedy troupes by providing examples of each *classification*


Click here to download pictures. To help protect your privacy, Outlook prevented au

From: Johnny Olsen <[redacted]>
To: [redacted]ings.ru
Cc:
Subject: hi ,get 25 no deposit needed

US Players: $25Free at Ruby Royal

Join Today and Get a Royal Welcome!
Make some money and have a ton of fun doing it!

Spin the sl0ts and hit the tables today -
Ruby Royal's footing the bill for US Players!

Get $25 AbsolutelyFree!

No Deposit Needed, No Strings Attached!
Redeem Promotion 25RED
Get a Huge 300% Match with your 1st Dep.!
Get Triple the FreeCash with a
deposit of just $25 or more!

Redeem Promotion ROYALSL0TS
Get Started Today - It's Easy!
1. DOWNLOAD & install our software
2. Create your free account
3. Select 25RED in the cashier

>> Visit Site! <<

[Spam - Monty Python's Flying Circus]
[Spam (Monty Python)] (Wikipedia)
[Why is it called Python?]

Classifiers

- Need *features* for classifying documents
- Feature is anything you can determine that is present or absent in the item
- Best features are common enough to appear frequently but not all the time (cf. stopwords)
- Words in document are a useful feature
- For spam detection, certain words like *casino* appear more often in spam than not spam

Features for Spam Classification

- Many *external features* can be used depending on type of document
  - Links pointing in? Links pointing out?
  - Recipient list? Sender's email and IP address?
- Many *internal features*
  - Use of certain words or phrases
  - Color and sizes of words
  - Document length
  - Grammar analysis
- We will focus on internal features to build a classifier

Classifying with Supervised Learning

- We use an *item's features* for classification
  - item = document
  - feature = word
  - classification = {good|bad}

```
"the quick brown fox jumps over the lazy dog", "good"
"make quick money in the online casino", "bad"
"buy pharmaceuticals now", "bad"
"the quick rabbit jumps fences", "good"
```

Web Science:
Document Filtering
(Part 2 - Classifiers and Probabilities)

CS 432/532

Old Dominion University

Permission has been granted to use these slides from Frank McCown, Michael L. Nelson, Alexander Nwala, Michele C. Weigle

Main references:

Ch 6 from *Programming Collective Intelligence* by Toby Segaran

(abbreviated as PCI)

Ch 9 from *Search Engines: Information Retrieval in Practice* by Croft, Metzler, and Strohman

(*abbreviated as SEIRP)*

From Counts to Probabilities

```
"Nobody owns the water.", "good"
"the quick brown fox jumps over the lazy dog", "good"
"make quick money in the online casino", "bad"
"buy pharmaceuticals now", "bad"
"the quick rabbit jumps fences", "good"
```

- Count the number of times a word falls into a category {good|bad}
  - "quick": {bad: 1, good: 2}
- Want the probability that a word is in a particular category
  - probability that "quick" is "good"

Events and Probabilities

- Independent: this means each event is not affected by other events
  - example: two separate coin tosses

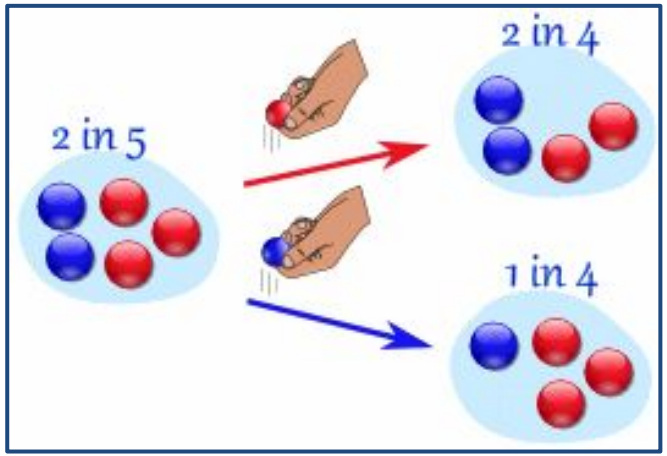
- Mutually Exclusive: this means events cannot happen at the same time

- Dependent (aka *Conditional*): this means an event is affected by other events

Adapted from: Conditional Probability, mathisfun

Conditional Probabilities

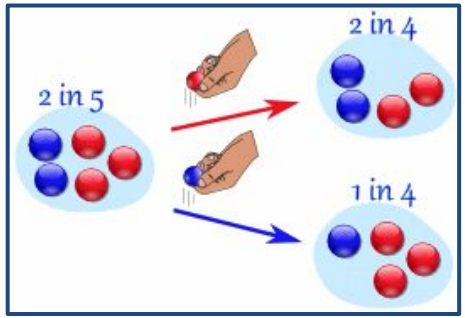
- Example: Marbles in a Bag
- What are the chances of getting a blue marble?



Adapted from: Conditional Probability, mathisfun

Conditional Probabilities

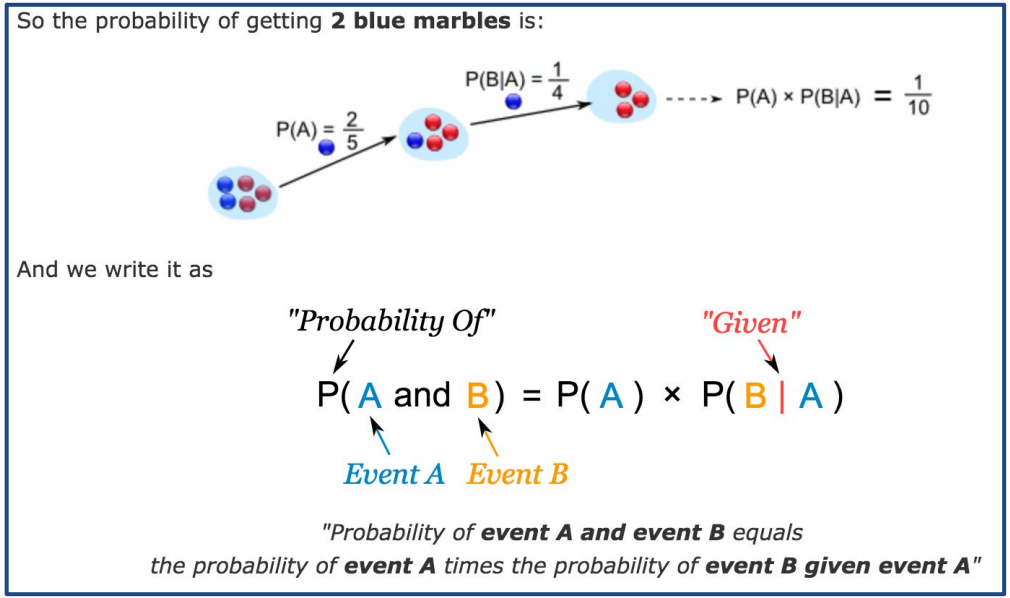
- Example: Marbles in a Bag
- What are the chances of getting a blue marble?
  - if a red marble was picked before, then the chance of picking a blue marble next is 2 in 4
  - if a blue marble before, then the chance of a blue marble next is 1 in 4



Adapted from: Conditional Probability, mathisfun

Conditional Probabilities: notation

- P(A) means "Probability of Event A"
- P(B|A) means "Event B given Event A"



P(A) = 2/5 = 0.4
P(B|A) = 1/4 = 0.25
P(A and B) = 1/10 = 0.1

Adapted from: Conditional Probability, mathisfun

From Counts to Probabilities

```
"Nobody owns the water.", "good"
"the quick brown fox jumps over the lazy dog", "good"
"make quick money in the online casino", "bad"
"buy pharmaceuticals now", "bad"
"the quick rabbit jumps fences", "good"
```

- Count the number of times a word falls into a category {good|bad}
  - "quick": {bad: 1, good: 2}
- Want the probability that a word is in a particular category
  - probability that "quick" is "good"

Pr(word|category)

```
"Nobody owns the water.", "good"
"the quick brown fox jumps over the lazy dog", "good"
"make quick money in the online casino", "bad"
"buy pharmaceuticals now", "bad"
"the quick rabbit jumps fences", "good"
```

- Pr(word|category) - for a given *category*, the probability that the *word* appears

Pr(word|category)

Multiple-Bernoulli model

$$Pr(w|c) = df_{w,c} / N_c$$

$df_{w,c}$ - num docs in c containing w

N_c - num docs in c

PCI book examples use this model

Multinomial model

$$Pr(w|c) = tf_{w,c} / |c|$$

$tf_{w,c}$ - times w appears in documents in c

$|c|$ - num words in c

SEIRP, Ch 9, pp. 346-349

Pr(word|category)

```
"Nobody owns the water.", "good"
"the quick brown fox jumps over the lazy dog", "good"
"make quick money in the online casino", "bad"
"buy pharmaceuticals now", "bad"
"the quick rabbit jumps fences", "good"
```

- Multiple Bernoulli model
  - Pr("quick"|"good") = num docs in “good” with “quick” / num docs in “good” = 2 / 3 = 0.6667
- Multinomial model
  - Pr("quick"|"good") = num times "quick" appears in "good" / num words in "good" = 2 / 18 = 0.111

Problem of Data Sparseness

```
"Nobody owns the water.", "good"
"the quick brown fox jumps over the lazy dog", "good"
"make quick money in the online casino", "bad"
"buy pharmaceuticals now", "bad"
"the quick rabbit jumps fences", "good"
```

- Pr("money"|"good") = num docs in "good" with "money" / num docs in "good" = 0 / 3 = **0**

- *data sparseness* - not every possible event is observed in the training set
- Start all probabilities at some *assumed probability*

Weighted Probability

$$Pr(w|c) = ((\alpha\ ap) + (cf_w\ Pr^*(w|c))) / (\alpha + cf_w)$$

α - weight (1)
ap - assumed probability (0.5)
cf_w - times w appears in training set

$Pr^*(w|c) = df_{w,c} / N_c$ original Multiple Bernoulli probability
$df_{w,c}$ - num docs in c containing w
N_c - num docs in c

PCI, pp. 122-123 (`weightedprob()`)

Weighted Probability: Pr("money", "good")

```
"Nobody owns the water.", "good"
"the quick brown fox jumps over the lazy dog", "good"
"make quick money in the online casino", "bad"
"buy pharmaceuticals now", "bad"
"the quick rabbit jumps fences", "good"
```

$$Pr(w|c) = ((\alpha\, ap) + (cf_w\, Pr^*(w|c)) / (\alpha + cf_w)$$

Pr^*("money"|"good") = 0

Pr("money", "good") =

(1 * 0.5) + (1 * 0) / (1 + 1) = 0.5 / 2 = **0.25**

Web Science: Document Filtering

(Part 3 - Bayesian Classifiers)

CS 432/532

Old Dominion University

Permission has been granted to use these slides from Frank McCown, Michael L. Nelson, Alexander Nwala, Michele C. Weigle

Main references:

Ch 6 from *Programming Collective Intelligence* by Toby Segaran

(abbreviated as PCI)

Ch 9 from *Search Engines: Information Retrieval in Practice* by Croft, Metzler, and Strohman

(*abbreviated as SEIRP)*

Naïve Bayesian Classifier

- Move from terms to documents:
  - $Pr(document) = Pr(term_1) * Pr(term_2) * \ldots * Pr(term_n)$
- *Naïve* because we assume all terms occur independently
  - we know this is as simplifying assumption; it is naïve to think all terms have equal probability for completing:

    "New York _____"

    "International Business _____"
- *Bayesian* because we use Bayes' Theorem to invert the conditional probabilities

Probability of Whole Document

- Naïve Bayesian classifier determines probability of entire document being given a classification
  - Pr(Category | Document)
- Assume:
  - Pr(*python*|*bad*) = 0.2
  - Pr(*casino*|*bad*) = 0.8
- So Pr(*python* & *casino*|*bad*) = 0.2 * 0.8 = 0.16
- This is Pr(Document|Category)
- How do we calculate Pr(Category|Document)?

Bayes' Rule

$$Pr(c|d) = Pr(d|c)\ Pr(c) / Pr(d)$$

$Pr(c) = \mathrm{N}_c / \mathrm{N}$

N_c - num docs in c

N - total docs

$Pr(d)$ - we'll ignore this for now...

SEIRP, Ch 9, pp. 342-344

Bayes' Rule Applied to Documents and Classes

That is, Naïve Bayes classifies a document d as follows:

$$\begin{aligned}\text{Class}(d) &= \arg\max_{c \in \mathcal{C}} P(c|d) \\ &= \arg\max_{c \in \mathcal{C}} \frac{P(d|c)P(c)}{\sum_{c \in C} P(d|c)P(c)}\end{aligned}$$

where $\arg\max_{c \in \mathcal{C}} P(c|d)$ means "return the class c, out of the set of all possible classes $\mathcal{C}$, that maximizes $P(c|d)$." This is a mathematical way of saying that we are trying to find the most likely class c given the document d.

SEIRP, Ch 9, p. 342

Bayes' Rule Applied to Documents and Classes

Dan Jurafsky

Naïve 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

Adapted from Text Classification and Naïve Bayes (stanford.edu), (archived copy)

Bayes' Rule Applied to Documents and Classes

Dan Jurafsky

Naïve Bayes Classifier (II)

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

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

Document d represented as features x1..xn

Adapted from Text Classification and Naïve Bayes (stanford.edu), (archived copy)

Bayes' Rule Applied to Documents and Classes

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

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

Adapted from Text Classification and Naïve Bayes (stanford.edu), (archived copy)

Bayes' Rule Applied to Documents and Classes

That is, Naïve Bayes classifies a document d as follows:

$$\text{Class}(d) = \arg\max_{c \in \mathcal{C}} P(c|d)$$

$$= \arg\max_{c \in \mathcal{C}} \frac{P(d|c)P(c)}{\sum_{c \in C} P(d|c)P(c)}$$

$$P(d|c) = \prod_{i=1}^{n} P(w_i|c)$$

$$P(c) = \frac{N_c}{N}$$

N_c - num docs in c

N - total docs

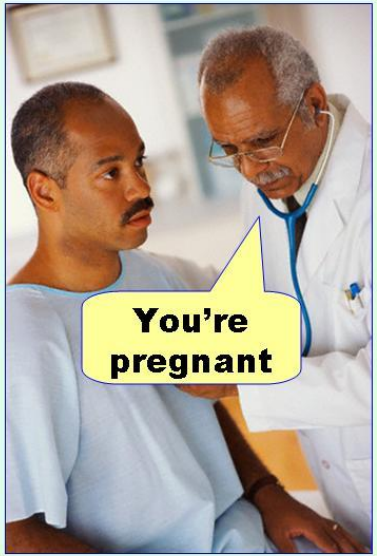
SEIRP, Ch 9, p. 344

Choosing Spam or Not Spam

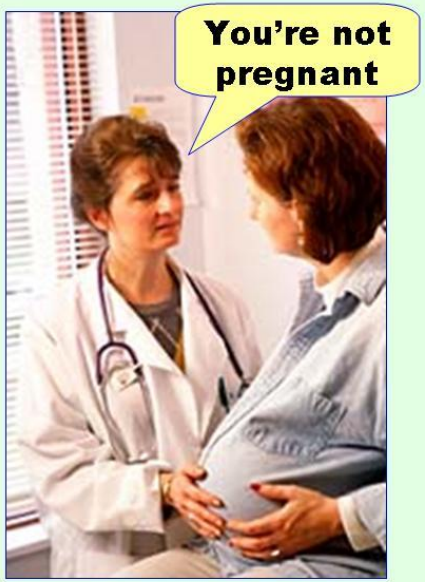
- Should we always choose the category with the highest probability?
  - what if the difference is really small?
- Sometimes the classifier should admit that it doesn't know the best category
- Better to have spam in inbox (*false negative*) than important messages filtered to spam (*false positive*)

False Positives vs. False Negatives

Type I error
(false positive)



Type II error
(false negative)

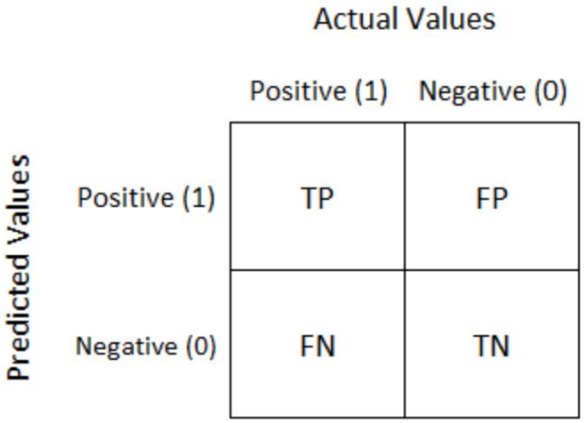


From: I always get confused about Type I and II errors. Can you show me something to help me remember the difference?

Tuning for False Negatives

- Use a *threshold* for each category
- Only classify as spam if probability difference is *threshold* times higher for spam than for not spam
- Can have a different threshold for each category.
  Examples:
  - "Spam" threshold = 3
  - "Not Spam" threshold = 1
- If threshold not met, classify as 'unknown'

Confusion Matrix

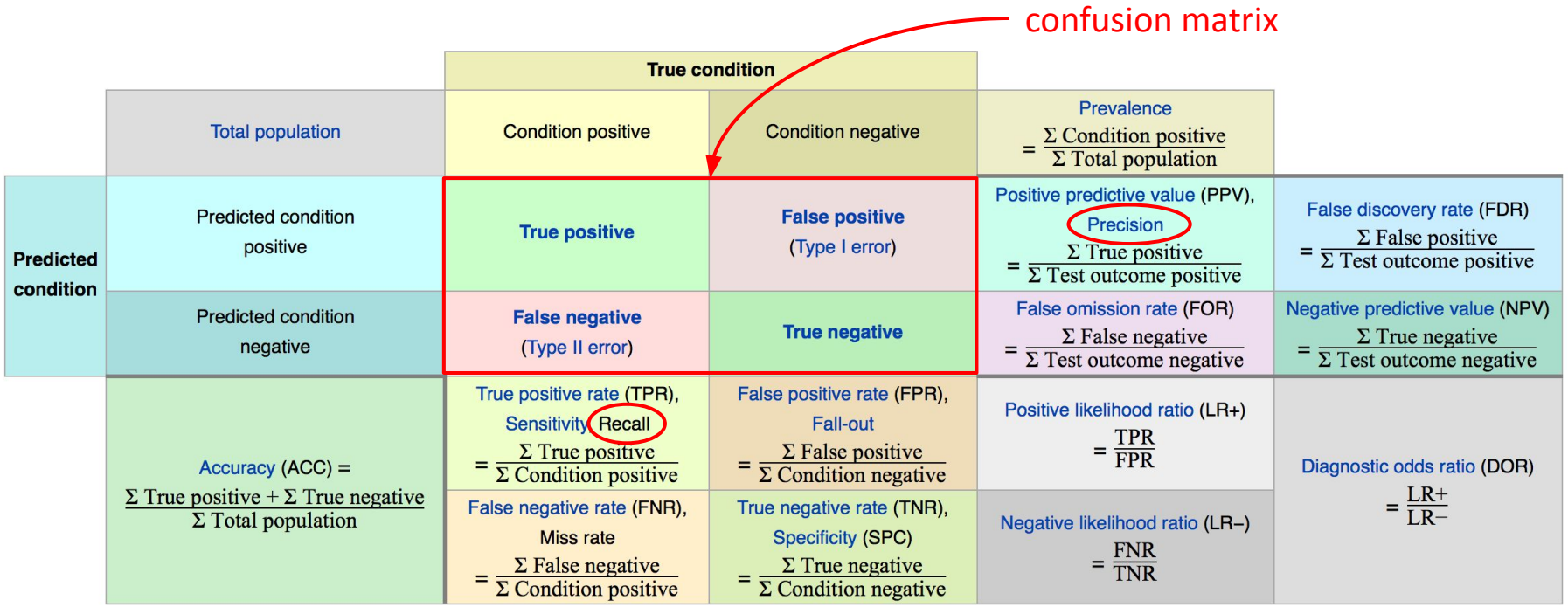


		Actual class	
		Cat	Dog
Predicted class	Cat	5	2
	Dog	3	3

- Demonstrates performance of a prediction/classification algorithm
- Actual (known) values are on the columns, Predicted (algorithm) values are on the rows
- Cells are populated by numbers of true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN)

References: Confusion matrix (Wikipedia),
Understanding Confusion Matrix. When we get the data, after data… | by Sarang Narkhede

Stats Derived from Confusion Matrix



		True condition			
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**	**False positive** (Type I error)	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Test outcome positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Test outcome positive}}$
	Predicted condition negative	**False negative** (Type II error)	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Test outcome negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Test outcome negative}}$
	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	True positive rate (TPR), Sensitivity, Recall = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$
		False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	True negative rate (TNR), Specificity (SPC) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$	

Reference: Precision and recall (Wikipedia)

How Do Humans Classify Things?

- We identify a number of important *features* that distinguish between possible categories.
- We *extract* these features from each item.
- We then *combine evidence* from the extracted features in some way.
- Finally, we *classify* the item using some decision mechanism based on the combined evidence.

SEIRP, Ch 9

Objectives

- Differentiate between clustering and classification.
- Explain the general steps in classification.
- Explain the purpose of training sets and testing sets in supervised learning.
- Explain how Bayes' Rule is used in a Naive Bayes Classifier.
- Train and test a Naive Bayes Classifier and draw a confusion matrix for the classification results.