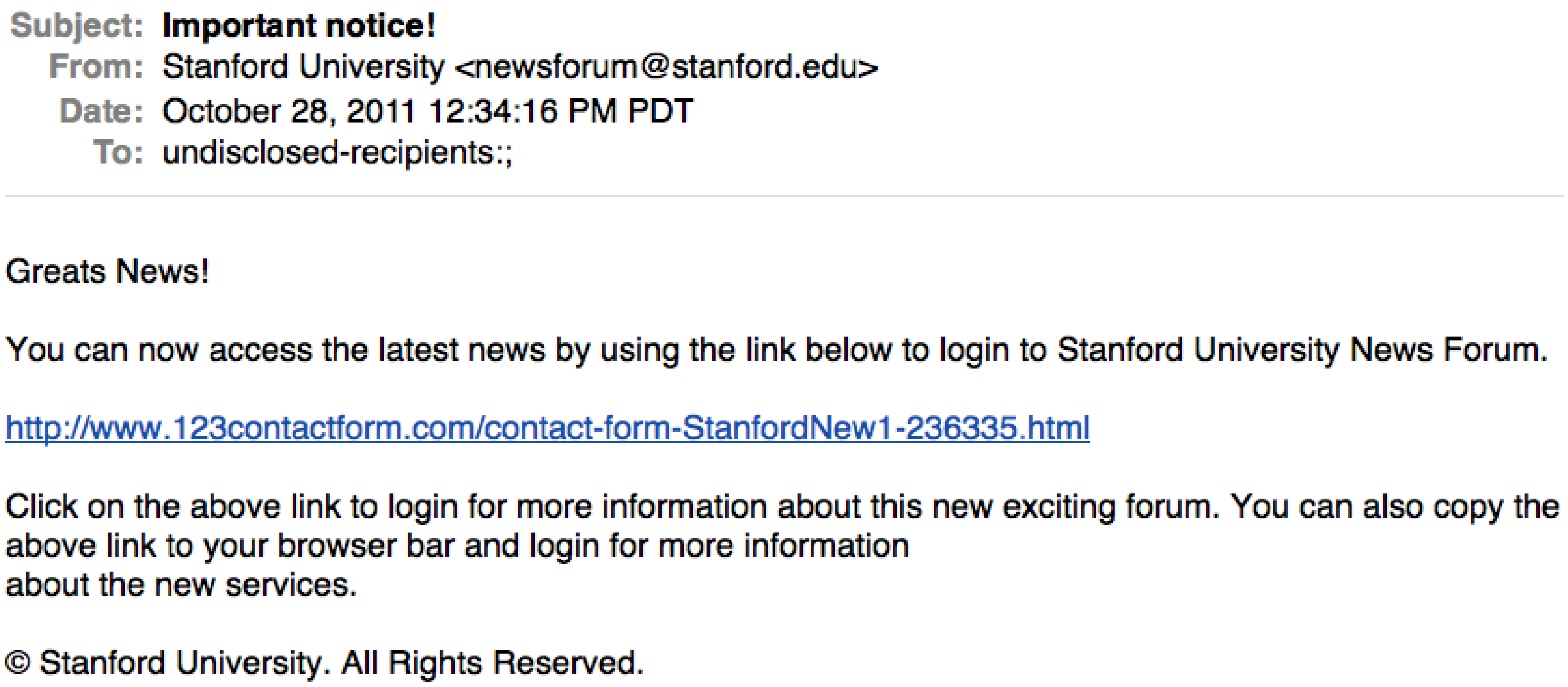
Text Classification

SLP3 Ch 2.1, 4, 5; INLP Ch 2, 4.4

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.

Positive or negative review?

- + *...zany characters and richly applied satire, and some great plot twists*
- – *It was pathetic. 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...*

What is the subject of this medical article?

MEDLINE Article

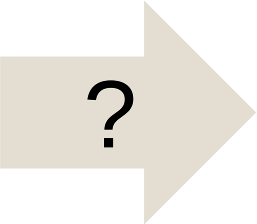
Available online at www.sciencedirect.com

Brain and Cognition

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

Susanne Gahl, Lise Menn, Gail Ramsberger, Daniel S. Jurafsky, Elizabeth Elder, Molly Rewega, and L. Holland Audrey

[illegible]



MeSH Subject Category Hierarchy

Antogonists and Inhibitors

Blood Supply

Chemistry

Drug Therapy

Embryology

Epidemiology

…

Text Classification: Definition

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

Text Classification: Methods

- Rule-based methods
  - Need experts writing rules
  - Regular expression
- Machine learning methods
  - Need annotated training data and computing resource for training
  - Generative classifiers
  - Discriminative classifiers

Regular Expressions

Regular Expressions

- A formal language for specifying text patterns
- We used char-level regular expression in word tokenization
- We may use word-level regular expression for text classification
  - Pattern for positive reviews
    - .*((very)? good|great|awesome|not bad).*
  - Pattern for negative reviews
    - .*(bad|awful|worst|ridiculously overpriced).*

Pattern	Matches
`.`	Any char
`(aaa|bbb)`	Disjunction, “aaa” or “bbb”

Regular Expressions

- Advantages
  - Interpretable
  - Easy to manipulate
  - Do not need (annotated) data and training
- Problems
  - Low coverage
    - hard even for an expert to cover all cases
  - Might be wrong
    - e.g., “not good at all”

Machine Learning Methods

Supervised Machine Learning

- Input:
  - 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 $d \to c$

Generative and Discriminative Classifiers

- Suppose we're distinguishing cat from dog images

Generative Classifier

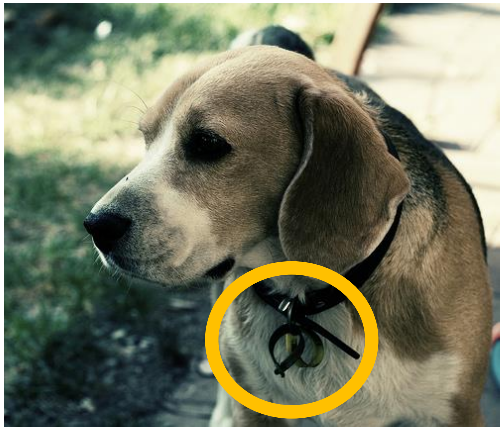
- Build a model of what's in a cat image
  - Knows about whiskers, ears, eyes…
  - Assigns a probability to any image:
    - If you ask someone to draw a cat, how likely he would draw this image?

- Also build a model for dog images

- Now given a new image:
  Run both models and see which one fits better.

Discriminative Classifier

- Just try to distinguish dogs from cats



Oh look, dogs have collars!
Let’s ignore everything else.

Generative and Discriminative Classifiers

- Finding the correct class c of a document d
- Generative classifiers

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

$$= \underset{c \in C}{\operatorname{argmax}} \overbrace{P(d|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}}$$

- Discriminative classifiers

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \overbrace{P(c|d)}^{\text{posterior}}$$

Machine Learning
-- Generative Classifiers

Generative Classifiers

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

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

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

How often does this class occur?

We can just count the relative frequencies in a corpus.

Multinomial Naïve Bayes

- Conditional Independence
  - Assume words are independent given the class c.

$$P(x_1, x_2, \ldots, x_n | c) = P_1(x_1|c) P_2(x_2|c) \cdots\ P_n(x_n|c)$$

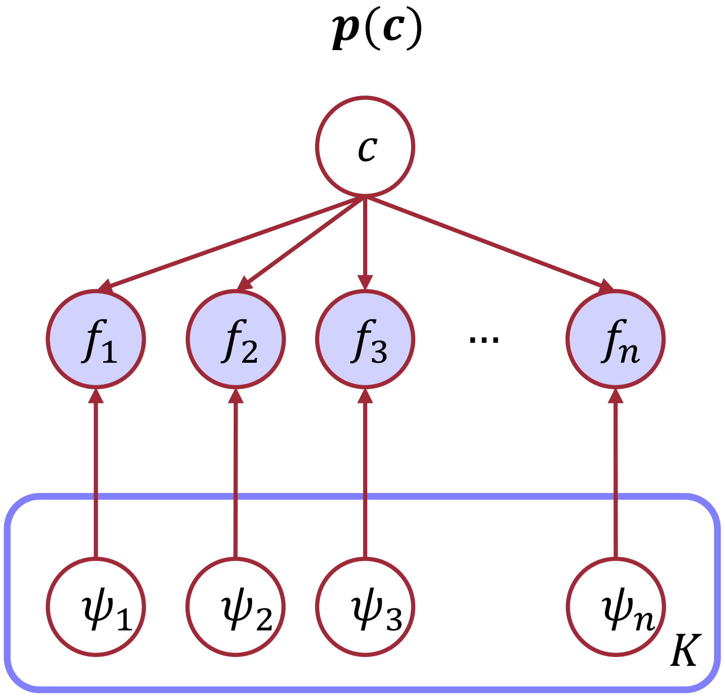
$$C_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i \in \text{positions}} P_i(x_i|c)$$

- Bag of Words assumption
  - Assume position doesn’t matter
  - All the positions share the same conditional distribution

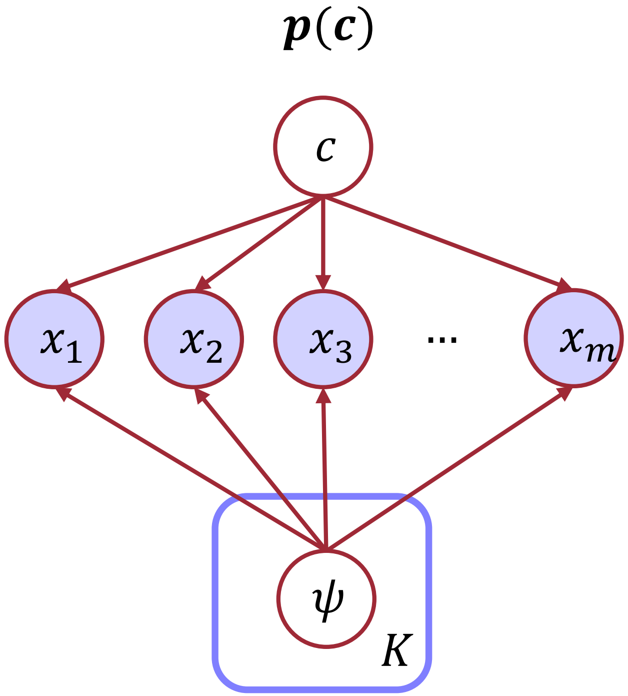
$$\forall i, j\colon\ P_i(x|c) = P_j(x|c)$$

Probabilistic graphical models for Naïve Bayes

- General form: $P(f_i|c;\psi_i)$
- Bag of Word: $P(x_i|c;\psi)$, ψ are irrelative to position.



General form



Bag of word

Learning Multinomial Naïve Bayes

- First attempt: maximum likelihood estimates
  - Simply use the frequencies in the data
  - N_{c_j} is the number of documents with class c; N_{total} is the total number of documents.

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

  - $count(w, c)$ is the times the word w appears among all words in all documents of topic c.

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

Problem with Maximum Likelihood

- What if we have seen no training documents with the word ***fantastic*** and class ***positive***?

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

- Zero conditional probability leads to zero posterior, no matter the other evidence!

$$C_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i \in \text{positions}} P(x_i|c)$$

Laplace (add-1) smoothing for Naïve Bayes

Pseudo-count

$$\hat{P}(\text{"fantastic"}|\text{positive}) = \frac{count(\text{"fantastic"},\ \text{positive}) + 1}{\sum_{w \in V}(count(w, \text{positive}) + 1)}$$

$$= \frac{count(\text{"fantastic"},\ \text{positive}) + 1}{(\sum_{w \in V} count(w, \text{positive})) + |V|}$$

Unknown words

- Words that
  - …appear in our test data
  - …but not in our training data or vocabulary
- We **ignore** them
  - Remove them from the test document and 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!

Let's do a 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 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:

$$\hat{P}(c_j) = \frac{N_{c_j}}{N_{total}} \qquad \begin{aligned} P(-) &= 3/5 \\ P(+) &= 2/5 \end{aligned}$$

3. Drop "with"

2. 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:

$$C_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i \in \text{positions}} P(x_i|c)$$

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

$$P(+)P(S|+) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.3 \times 10^{-5}$$

Machine Learning
-- Discriminative Classifiers

Generative and Discriminative Classifiers

- Finding the correct class c of a document d
- Generative classifiers

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

$$= \underset{c \in C}{\operatorname{argmax}} \ \overbrace{P(d|c)}^{\text{likelihood}} \ \overbrace{P(c)}^{\text{prior}}$$

- Discriminative classifiers

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \ \overbrace{P(c|d)}^{\text{posterior}}$$

Representing text with feature vectors

- As a sparse feature vector
  - tf-idf
  - n-gram frequencies (n-gram = n-word combination)
  - Task-specific features
- As a dense feature vector
  - Latent Semantic Analysis
  - Computed from the sequence of word embeddings of the input text.

Logistic regression (two classes)

- Make probabilities with Sigmoid.

$$P(y=1)=\sigma(w\cdot x+b)=\frac{1}{1+\exp\big(-(w\cdot x+b)\big)}$$

$$P(y=0)=1-\sigma(w\cdot x+b) \qquad =\sigma\big(-(w\cdot x+b)\big)$$

$$=\frac{\exp\big(-(w\cdot x+b)\big)}{1+\exp\big(-(w\cdot x+b)\big)}$$

$\sigma(-x)=1-\sigma(x)$

- Turning a probability into a classifier

$$f(x)=\begin{cases}1, & \text{if } P(y=1|x)>0.5\\ 0, & \text{otherwise}\end{cases}$$

if $w\cdot x+b>0$

If $w\cdot x+b\leq 0$

Logistic regression (>2 classes)

- Still compute the dot product between weight vector w and input vector x.
- But now we need a separate weight vector for each of the K classes.

$$p(y = c|x) = \frac{\exp(w_c \cdot x + b_c)}{\sum_{j=1}^{k} \exp\left(w_j \cdot x + b_j\right)}$$

- Softmax function
  - Turning a vector $z = [z_1, z_2, \dots, z_k]$ of k arbitrary values into probabilities.

$$\text{softmax}(z) = \left[\frac{\exp z_1}{\sum_{j=1}^{k} \exp z_j}, \frac{\exp z_2}{\sum_{j=1}^{k} \exp z_j}, \dots, \frac{\exp z_k}{\sum_{j=1}^{k} \exp z_j}\right]$$

  - Q: how is softmax related to sigmoid?

Learning in Logistic Regression

- Maximizing conditional log likelihood of the true labels
  - = minimizing cross-entropy loss
  - often with a regularization term to avoid overfitting

$$\mathcal{L} = \overbrace{-\frac{1}{N}\sum_{i \in N} \log p_\theta(y_i^* | x)}^{\mathcal{L}_{CE}} + \lambda \overbrace{R(\theta)}^{\mathcal{L}_{reg}}$$

- Optimized with stochastic gradient descent

More Discriminative Classifiers

- Support vector machines
- Neural networks
- Decision trees
- …

Evaluation

Datasets

Training set

Development Set

Test Set

- Train on training set, tune on dev set, report on test set

The 2-by-2 confusion matrix

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

Why Precision and Recall

- Why don't we just use **accuracy** as our metric?

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

- Imagine we save 1 million messages.
  - 100 of them talked about "Pie"
  - 999,900 talked about something else
- We could build a dumb classifier that just labels every message "not about pie"
  - It would get 99.99% accuracy!!! Wow!!!
  - But useless! Doesn't return the messages we are looking for!
  - Recall = 0
    - It doesn't get any of the 100 Pie messages.

A combined measure

- 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., $\beta = 1$)

$$F_1 = \frac{2PR}{P + R}$$

Confusion Matrix for 3-class classification

gold standard labels

system output labels		urgent	normal	spam
	urgent	8	10	1
	normal	5	60	50
	spam	3	30	200

$$\textbf{precision}_u = \frac{8}{8 + 10 + 1}$$

$$\textbf{precision}_n = \frac{60}{5 + 60 + 50}$$

$$\textbf{precision}_s = \frac{200}{3 + 30 + 200}$$

$$\textbf{recall}_u = \frac{8}{8+5+3}$$

$$\textbf{recall}_n = \frac{60}{10+60+30}$$

$$\textbf{recall}_s = \frac{200}{1 + 50 + 200}$$

Combine P/R from 3 classes to get one metric

- Macro-averaging:
  - Compute the performance for each class
  - Then average over classes
- Micro-averaging:
  - Aggregate statistics for all classes into one confusion matrix
  - Compute the precision and recall from that table

Macro-/Micro-averaging

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

$$\text{Precision} = .52$$

Class 3: Spam

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

$$\text{Precision} = .86$$

Pooled

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

$$\begin{matrix}\text{micro-average}\\ \text{precision}\end{matrix} = .73$$

$$\begin{matrix}\text{macro-average}\\ \text{precision}\end{matrix} = \frac{.42 + .52 + .86}{3} = .60$$

Summary

Text Classification

- Rule-based methods
  - Regular expression
- Machine learning methods
  - Generative classifiers
    - Naive Bayes
  - Discriminative classifiers
    - Logistic regression
- Evaluation
  - Precision, recall, F-measure
  - Macro-/micro-averaging