CAMPUS
DE EXCELENCIA
INTERNACIONAL

Text Mining 4
Text Classification

Madrid Summer School 2014
Advanced Statistics and Data Mining

Florian Leitner
florian.leitner@upm.es

Incentive and Applications

- Assign one or more “labels” to a collection of “texts”.

- Spam filtering
- Marketing and politics (“**Opinion/Sentiment Analysis**”)
- Grouping similar items (e.g., “**Recommendation Engines**”)
- Ranking/searching for [topic- or query-specific] documents

- Today’s topics:
  - Document Similarity, Text Classification, Sentiment Analysis

Document Similarity

- Similarity Measures
  - Cosine Similarity
  - Correlation Coefficients
- Word Vector Normalization
  - TF-IDF
- Latent Semantic Indexing
  - Dimensionality Reduction/Clustering

Information Retrieval (IR)

REMINDER

Text Vectorization: Inverted Index

Text 1: He that not wills to the end neither wills to the means.

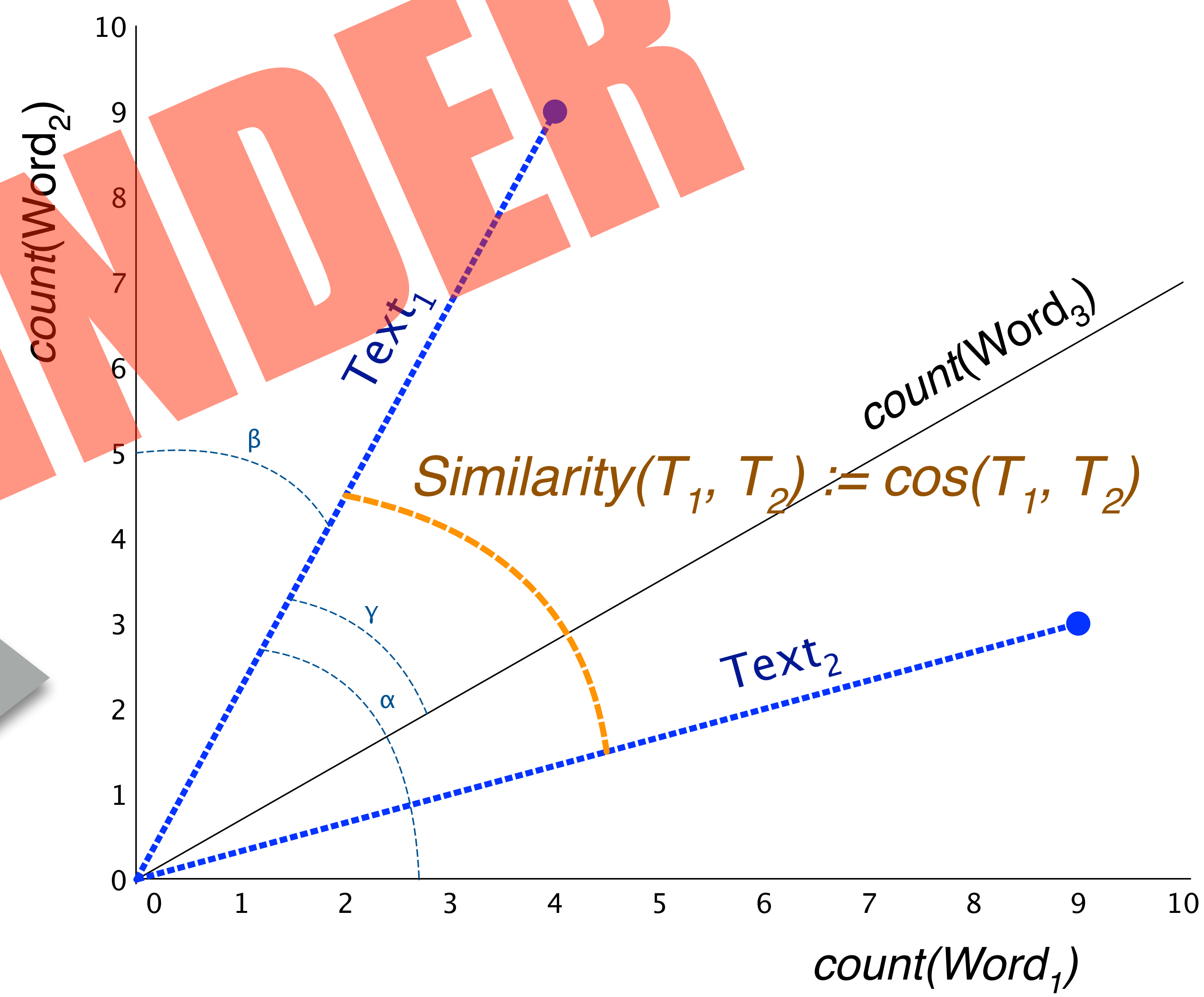
Text 2: If the mountain will not go to Moses, then Moses must go to the mountain.

Terms	Doc 1	Doc 2
end	1	0
go	0	2
he	1	0
if	0	1
means	1	0
Moses	0	2
mountain	0	2
must	0	1
not	1	1
that	1	0
the	2	2
then	0	1
to	2	2
will	2	1

document vector

term/word vector

Comparing Word Vectors: Cosine Similarity



Cosine Similarity

- Define a similarity score between document vectors (and/or query vectors in Information Retrieval)
- **Euclidian** vector **distance** is **length dependent**
- **Cosine** [angle] between two vectors **is not**

$$sim(\vec{x},\vec{y}) = cos(\vec{x},\vec{y}) = \frac{\vec{x}\cdot\vec{y}}{|\vec{x}||\vec{y}|} = \frac{\sum x_i y_i}{\sqrt{\sum x_i^2}\sqrt{\sum y_i^2}}$$

can be dropped if using unit ("length-normalized") vectors
a.k.a. "cosine normalization"

Alternative Similarity Coefficients

- **Spearman's rank correlation coefficient** ρ (r[ho])
  - Ranking is done by term frequency (**TF**; count)
  - Critique: sensitive to ranking differences that are likely to occur with high-frequency words (e.g., "the", "a", …) ➡ use the **log** of the term count, rounded to two significant digits
    - NB that this is not relevant when only short documents (e.g. titles) with low TF counts are compared
- **Pearson's chi-square test** χ^2
  - Directly on the TFs (counts) - Intuition:
    Are the TFs "random samples" from the same base distribution?
  - Usually, χ^2 should be preferred over ρ (Kilgarriff & Rose, 1998)
    - NB that both measures have no inherent normalization of document size
      - preprocessing might be necessary!

Term Frequency times Inverse Document Frequency (TF-IDF)

- **Motivation and Background**
- The Problem
  - **Frequent terms** contribute most to a document vector's direction, but **not all** terms are **relevant** ("the", "a", …).
- The Goal
  - **Separate** important terms from frequent, but **irrelevant terms** in the collection.
- The Idea
  - Frequent **terms** appearing **in all documents** tend to be less important **versus** frequent terms in just a **few documents**. → Zipf's Law!
    - also **dampens** the effect of **topic-specific noun phrases** or an **author's bias** for a specific set of adjectives

Term Frequency times Inverse Document Frequency (TF-IDF)

- **tf.idf**(w) := tf(w) × idf(w)
  - tf: term frequency
  - idf: inverse document frequency
- $\mathbf{tf_{natural}}(w) := \mathrm{count}(w)$
  - $tf_{natural}$: total count of a term in all documents
- $\mathbf{tf_{log}}(w) := \log(\mathrm{count}(w) + 1)$
  - tf_{log}: the TF is smoothed by taking its log
- $\mathbf{idf_{natural}}(w) := N \,/\, \sum^{N} \{w_i > 0\}$
  - $idf_{natural}$: # of documents in which a term occurs
- $\mathbf{idf_{log}}(w) := \log(N \,/\, \sum^{N} \{w_i > 0\})$
  - idf_{log}: smoothed IDF by taking its log
  - where N is the **number of documents** and w_i the **count of word** w in document i

TF-IDF in Information Retrieval

- Document Vector = $\text{tf}_{\log}$ 1 → i.e. the DVs do not use any IDF weighting (simply for efficiency; QV has IDF and gets multiplied with DV values)
- Query Vector = $\text{tf}_{\log}$ $\text{idf}_{\log}$
- terms are counted on each individual document/the query
- Cosine vector length normalization for tf.idf scores:
  - Document W normalization $$\sqrt{\sum_{w \in W} tf_{log}(w)^2}$$
  - Query Q normalization $$\sqrt{\sum_{q \in Q} (tf_{log}(q) \times idf_{log}(q))^2}$$
- IDF is calculated over the indexed collection of all documents

TF-IDF Query Similarity Calculation Example

	Collection		Query Q				Document D			Similarity
Term	**df**	**idf**	**tf**	**tf**	**tf.idf**	**norm**	**tf**	**tf**	**tf.1**	**cos(Q,D)**
best	3.5E+05	1.46	1	0.30	0.44	0.21	**0**	0.00	0.00	0.00
text	2.4E+03	3.62	1	0.30	1.09	0.53	10	1.04	0.06	0.03
mining	2.8E+02	4.55	1	0.30	1.37	0.67	8	0.95	0.06	0.04
tutorial	5.5E+03	3.26	1	0.30	0.98	0.48	3	0.60	0.04	0.02
data	9.2E+05	1.04	**0**	0.00	0.00	0.00	10	1.04	0.06	0.00
…	…	…	…	…	0.00	0.00	…	16.00	…	0.00
Sums	1.0E+07		4		2.05		~355	16.11		**0.09**

$\sqrt{}$ of $\sum$ of ^{2}

3 out of hundreds of unique words match (Jaccard < 0.03)

Example idea from: Manning et al. Introduction to Information Retrieval. 2009 free PDF!

From Syntactic to Semantic Similarity

Cosine Similarity, χ^2, or Spearman's ρ all only compare tokens.

[or n-grams!]

But what if you are talking about "automobiles" and I am lazy, calling it a "car"?

We can solve this with Latent Semantic Indexing!

Latent Semantic Analysis (LSI 1/3)

- a.k.a. Latent Semantic Indexing (in Text Mining): **dimensionality reduction** for **semantic inference**
- Linear Algebra Background
  - Symmetric Diagonalization of Matrix Q: $S = Q\Lambda Q^T$

    eigenvectors

    symmetric

    eigenvalues (diagonal matrix)

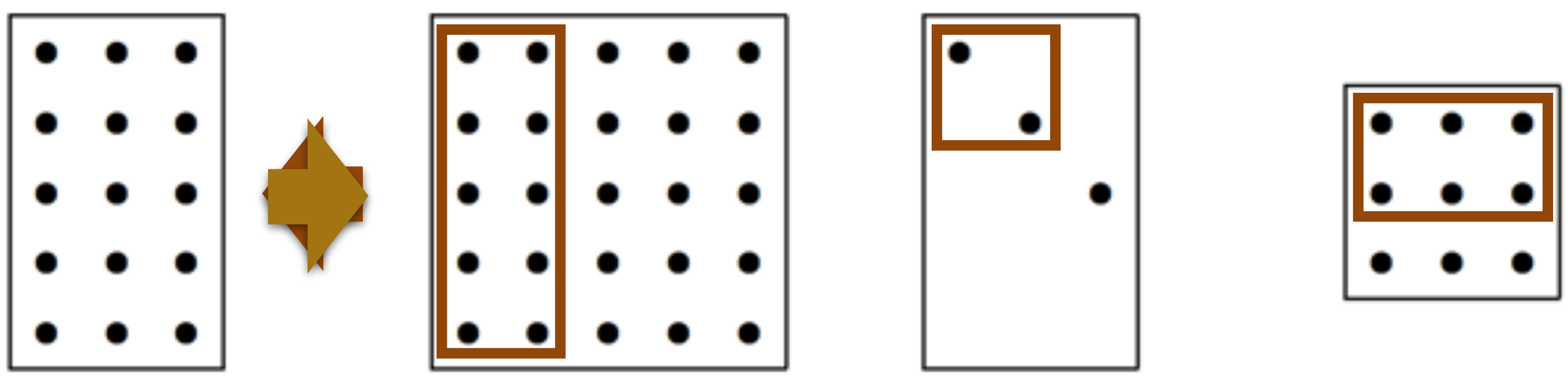
  - Singular Value Decomposition of Matrix Q: $Q = U\Sigma V^T$

    orthogonal eigenvectors (QQ^T and Q^TQ)

    singular values (dm)
- SVD in Text Mining
  - Inverted Index = Doc. Eigenvectors × Singular Values × Term Eigenvectors

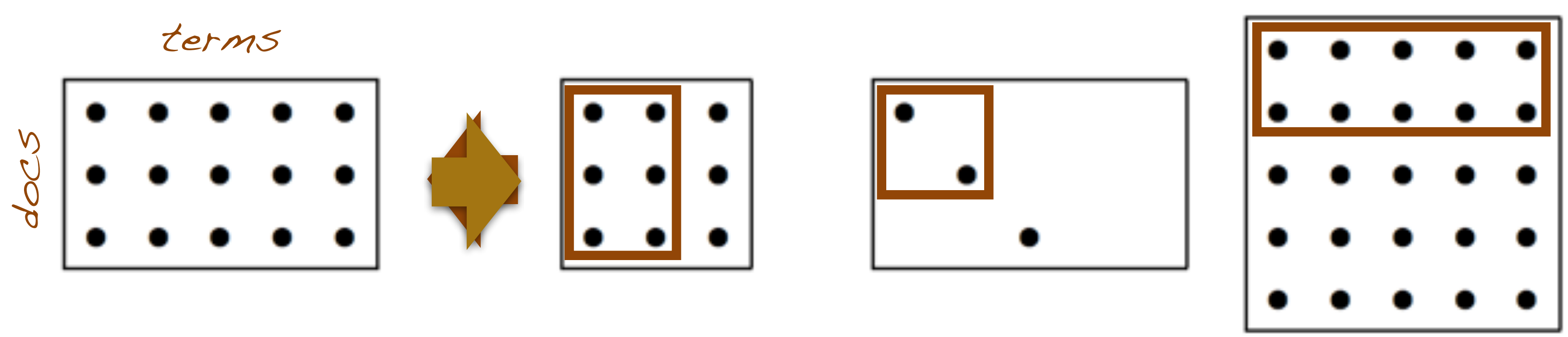
Latent Semantic Analysis (LSI 2/3)

$\hat{C}$ = DimRed by selecting only the largest n eigenvalues



- Inverted Index = Doc. Eigenvectors × Singular Values × Term Eigenvectors

$$C = U \Sigma V^T$$



- Image taken from: Manning et al. An Introduction to IR. 2009

Latent Semantic Analysis (LSI 3/3)

[Spearman's] rho(human, user) = -0.38
rho(human, minors) = -0.29

c1: *Human* machine *interface* for ABC *computer* applications
c2: A *survey* of *user* opinion of *computer system response time*
c3: The *EPS user interface* management *system*
c4: *System* and *human system* engineering testing of *EPS*
c5: Relation of *user* perceived *response time* to error measurement

m1: The generation of random, binary, ordered *trees*
m2: The intersection *graph* of paths in *trees*
m3: *Graph minors* IV: Widths of *trees* and well-quasi-ordering
m4: *Graph minors*: A *survey*

C

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

Ĉ

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	0.16	0.40	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
interface	0.14	0.37	0.33	0.40	0.16	-0.03	-0.07	-0.10	-0.04
computer	0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
user	0.26	0.84	0.61	0.70	0.39	0.03	0.08	0.12	0.19
system	0.45	1.23	1.05	1.27	0.56	-0.07	-0.15	-0.21	-0.05
response	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
time	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
EPS	0.22	0.55	0.51	0.63	0.24	-0.07	-0.14	-0.20	-0.11
survey	0.10	0.53	0.23	0.21	0.27	0.14	0.31	0.44	0.42
trees	-0.06	0.23	-0.14	-0.27	0.14	0.24	0.55	0.77	0.66
graph	-0.06	0.34	-0.15	-0.30	0.20	0.31	0.69	0.98	0.85
minors	-0.04	0.25	-0.10	-0.21	0.15	0.22	0.50	0.71	0.62

top 2 dim

test # dim
to use via
synonyms
or missing
words

rho(human, user) = **0.94**
rho(human, minors) = **-0.83**

From: Landauer et al. An Introduction to LSA. 1998

Principal Component vs. Latent Semantic Analysis

- **LSA** seeks for the **best linear subspace** in **Frobenius norm**, while **PCA** aims for the **best affine linear subspace**.
- **LSA** (**can**) **use** TF-IDF weighting as **preprocessing** step.
- **PCA requires the** (square) **covariance matrix** of the original matrix as its first step and therefore can only compute term-term or doc-doc similarities.
- **PCA matrices are more dense** (zeros occur only when true independence is detected).

- So far, we have seen how to establish if two documents are syntactically (kNN/LSH) and even semantically (LSI) similar.

- But how do we assign some “label” (or “class”) to a document?
  - E.g., **relevant**/not relevant; **polarity** (positive, neutral, negative); a **topic** (politics, sport, people, science, healthcare, …)
  - We could use the distances (e.g., from LSI) to **cluster** the documents
  - Instead, let’s look at **supervised methods** next.

Text Classification Approaches

efficient/fast

- **Multinomial Naïve Bayes**
- **Nearest Neighbor classification** (ASDM Course 03)
  - Reminder: **Locality Sensitivity Hashing*** (see part 3)
- **Latent Semantic Indexing*** **and/or Clustering*** (Course 03/10)
- **Maximum Entropy classification**
- **Latent Dirichlet Allocation***
- **Support Vector Machines** (ASDM Course 11)
- **Random Forests**
- (Recurrent) **Neural Networks** (ASDM Course 05)
- …

your speaker's "favorites"

high accuracy

* (optionally) unsupervised

Three Text Classifiers

- Multinomial Naïve Bayes
- Maximum Entropy (Multinomial Logistic Regression)
- Latent Dirichlet Allocation → tomorrow!

Bayes' Rule: Diachronic Interpretation

REMINDER

prior

likelihood

posterior → $$P(H|D) = \frac{P(H) \times P(D|H)}{P(D)}$$

"normalizing constant"

(law of total probability)

H - Hypothesis

D - Data

Maximum A Posterior (MAP) Estimator

- Issue: Predict the class $c \in C$ for a given document $d \in D$
- Solution: MAP, a "perfect" Bayesian estimator:

$$C_{MAP}(d) = \underset{c \in C}{argmax}\ \underbrace{P(c|d)}_{\text{the posterior}} = \underset{c \in C}{argmax}\ \frac{P(d|c)P(c)}{\underbrace{\cancel{P(d)}}_{\text{redundant}}}$$

- Problem: $d := \{w_1, ..., w_n\}$ dependent words/features W
  - exponential parametrization: one param. for each combination of W per C

Multinomial Naïve Bayes Classification

- A simplification of the MAP Estimator
  - count(w) is a discrete, **multinomial** variable (unigrams, bigrams, etc.)
  - Reduce space by making a **strong independence assumption** ("naïve")

$$C_{MAP}(d) = \underset{c \in C}{argmax}\ P(d|c)P(c) \overset{\text{independence assumption}}{\approx} \underset{c \in C}{argmax}\ P(c) \underbrace{\prod_{w \in W} P(w|c)}_{\text{“bag of words/features”}}$$

- Easy **parameter estimation**

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

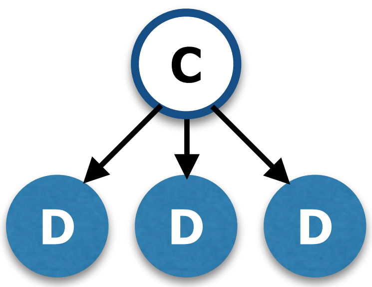
count(w_i, c): the total count of word i in all documents of class c [in our training set]

  - V is the entire **vocabulary** (collection of unique words/n-grams/...) in D
  - uses a Laplacian/Add-One Smoothing

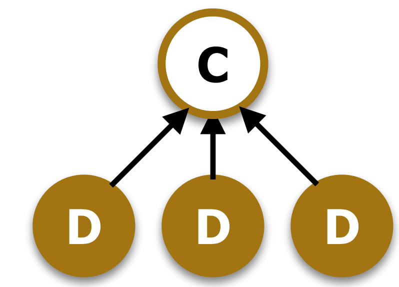
Multinomial Naïve Bayes: Practical Aspects

- Can gracefully handle **unseen words**
- Has **low space requirements**: $|V| + |C|$ floats → sum ∏ using logs!
- **Irrelevant** (=ubiquitous) **words** cancel each other out
- Opposed to SVM or Nearest Neighbor, it is very **fast**
  - Reminder: the k-shingle LSH approach to NN is fast, too.
  - But Multi-NB will probably result in lower accuracy (➡ "**baseline**").
- Each class has its own **n-gram language model**
- **Logarithmic damping** ($\log(count)$) might improve classification

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



Generative vs. Discriminative Models



- Generative models describe how the [hidden] labels "generated" the [observed] input as **joint probabilities**: P(*class*, *data*)
  - Examples: Markov Chain, Naïve Bayes, Latent Dirichlet Allocation, Hidden Markov Model, …
- Discriminative models only predict ("discriminate") the [hidden] labels **conditioned** on the [observed] input: P(*class* | *data*)
  - Examples: Logistic Regression, Support Vector Machine, Conditional Random Field, Neural Network, …
- Both can identify the most likely labels and their likelihoods
- **Only generative models**:
  - Most likely input value[s] and their likelihood[s]
  - Likelihood of input value[s] for some particular label[s]

$$P(H|D) = \frac{P(H) \times P(D|H)}{P(D)}$$

Maximum Entropy (MaxEnt) Intuition

The Principle of Maximum Entropy

go	to	1/5	1/6	??
	up	1/5	1/6	??
	outside	1/5	1/6	??
	home	1/5	1/4	??
	out	1/5	1/4	??

$w \in W$

$\sum_{w \in W} P(w \mid go) = 1$

$P(out \mid go) + P(home \mid go) = 0.5$

$P(to \mid go) + P(home \mid go) = 0.75$

Supervised MaxEnt Classification

$$p(x) = \frac{1}{1 + exp(-(\lambda_0 + \lambda_1 x))}$$

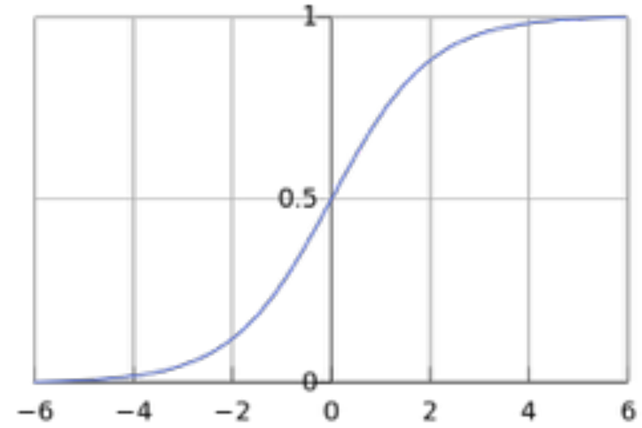
ln

$$ln \frac{p(x)}{1-p(x)} = \lambda_0 + \lambda_1 x$$

e

odds-ratio!

$$\frac{p(x)}{1-p(x)} = exp(\lambda_0 + \lambda_1 x)$$



logistic function p

Image Source: WikiMedia Commons, Qef

- a.k.a. Multinomial Logistic Regression
- **Does not assume independence between the features**
- Can **model mixtures of** binary, discrete, and real **features**
- Training data are per-feature-label probabilities: P(F, L)
  - I.e., count(f_i, l_i) $\div \sum^{N}_{i=1}$ count(f_i, l_i)
  - ➡ words → very sparse training data (zero or few examples)
- The model is commonly “learned” using gradient descent
  - Expensive if compared to Naïve Bayes, but efficient optimizers exist (**L-BFGS**)

Example Feature Functions for MaxEnt Classifiers

- Examples of indicator functions (a.k.a. **feature functions**)
  - Assume we wish to classify the general polarity (positive, negative) of product reviews:
- $f(c, w) := \{c = \text{POSITIVE} \wedge w = \text{“}\textit{great}\text{”}\}$
  - Equally, for classifying words in a text, say to detect proper names, we could create a feature:
- $f(c, w) := \{c = \text{NAME} \wedge \text{isCapitalized}(w)\}$
- Note that while we can have multiple classes, we cannot require more than one class in the whole match condition of a single indicator (feature) function.

NB: typical text mining models can have a million or more features: unigrams + bigrams + trigrams + counts + dictionary matchs + …

Maximizing Conditional Entropy

- The **conditioned** (on X) version of Shannon's **entropy** H:

NB: the chain rule

$P(x, y) = P(x)\ P(y|x)$

$$H(Y|X) = -\sum_{x \in X} P(x)\ H(Y|X = x)$$

$$= -\sum_{x \in X} P(x) \sum_{y \in Y} P(y|x)\ log_2\ P(y|x)$$

(swapped nom/denom to remove the minus)

$$= \sum_{x,y \in X,Y} P(x, y)\ log_2 \frac{P(x)}{P(x, y)}$$

- MaxEnt **training** then is about selecting the model p* that maximizes H:

$$p^* = \underset{p \in P}{argmax}\ H(P) = \underset{p \in P}{argmax}\ H(Y|X)$$

Maximum Entropy (MaxEnt 1/2)

- Some definitions:
  - The observed probability of y (the class) with x (the words) is:

$$\hat{P}(x,y) = count(x,y) \div N$$

  - An **indicator function** ("**feature**") is defined as a binary valued function that returns 1 iff class and data match the **indicated** requirements (**constraints**):

$$f(x,y) = \begin{cases} 1\ if\ \ y = c_i \wedge x = w_i \\ 0\ otherwise \end{cases}$$

real/discrete/binary features now are all the same!

  - The probability of a feature with respect to the observed distribution is:

$$\hat{P}(f_i, X, Y) = E_{\hat{P}}[f_i] = \sum \hat{P}(x,y) f_i(x,y)$$

Getting lost? Reality check:

- I have told you:
  - MaxEnt is about to maximize “conditional entropy”:
  - By multiplying binary (0/1) feature functions for observations with the joint (observation, class) probabilities, we can calculate the conditional probability of a class given its observations $H(Y|X)$
- We will still have to do:
  - Find weights for each feature [function] that lead to the best model of the [observed] class probabilities.
- And you want to know:
  - How the heck do we actually classify stuff???

Maximum Entropy (MaxEnt 2/2)

- In a **linear** model, we'd use weights ("lambdas") that identify the most relevant features of our model, i.e., we use the following MAP to select a class:

$$\underset{y \in Y}{argmax} \sum \lambda_i f_i(x, y)$$

- To do **multinomial logistic** regression, expand with a **linear combination**:

$$\underset{y \in Y}{argmax} \frac{exp(\sum \lambda_i f_i(x, y))}{\sum_{y \in Y} exp(\sum \lambda_i f_i(x, y))}$$ "exponential model"

- Next: **Estimate** the λ weights (parameters) that **maximize** the conditional **likelihood** of this logistic model (**MLE**)

Maximum Entropy (MaxEnt 2/2) [again]

- In summary, MaxEnt is about selecting the "maximal" model p*:

$$p^* = \underset{p \in P}{argmax} \; - \sum_{x \in X} p(x) \; \sum_{y \in Y} p(y|x) \; log_2 \; p(y|x)$$

select some model that maximizes the conditional entropy...

- That obeys the following conditional equality constraint:

$$\sum_{x \in X} P(x) \; \sum_{y \in Y} P(y|x) \; f(x,y) = \sum_{x \in X, y \in Y} P(x,y) \; f(x,y)$$

...using a conditional model that matches the (observed) joint probabilities

- Next: Using, e.g., **Langrange multipliers**, one can establish the optimal λ parameters of the model that maximize the entropy of this probability:

$$p^*(y|x) = \frac{exp(\sum \lambda_i f_i(x,y))}{\sum_{y \in Y} exp(\sum \lambda_i f_i(x,y))}$$

Newton's Method for Paramter Optimization

- Problem: find the λ parameters
  - an "**optimization problem**"
- MaxEnt surface is **concave**
  - one **single maximum**
- Using **Newton's method**
  - iterative, hill-climbing search for max.
  - the **first derivative** f' is zero at the [global] maximum (the "goal")
  - the **second derivative** f'' indicates rate of change: $\Delta\lambda_i$ (search direction)
  - takes the most direct route to the maximum *as opposed to gradient descent, which will follow a possibly curved path to the optimum*
- Using **L-BFGS**
  - a **heuristic** to simplify Newton's method *it is said to be "quasi-Newtonian"*
  - L-BFGS: **limited memory** **B**royden–**F**letcher–**G**oldfarb–**S**hanno
  - normally, the **partial second derivatives** would be stored in the **Hessian**, a matrix that **grows quadratically** with respect to the number of features
  - only uses the last few [partial] gradients to **approximate the search direction**

MaxEnt vs. Naïve Bayes

MaxEnt model (as observed)

Lights Working | Lights Broken

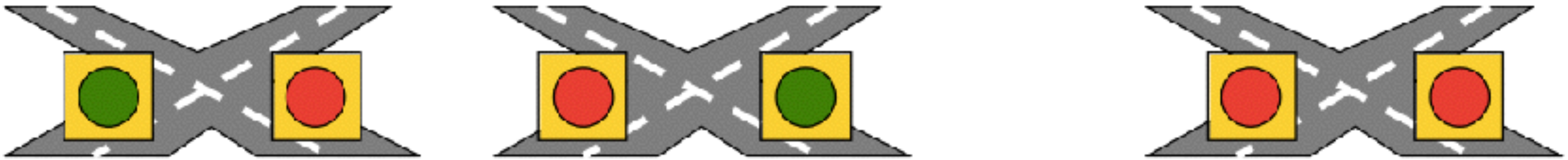


Image Source: Klein & Manning. Maxent Models, Conditional Estimation, and Optimization. ACL 2003 Tutorial

P(g,r,w) = 3/7 P(r,g,w) = 3/7 P(r,r,b) = 1/7

MaxEnt adjusts the Langrange multipliers (weights) to **model** the correct (observed) **joint probabilities**. But even MaxEnt cannot model **feature interaction**!

F	a	a	
b	1	1	$F_1 = 2/3$ $F_2 = 2/3$ 4/9 2/9
b	1	0	2/9 1/9

NB the example has dependent features: the two stoplights

Naïve Bayes model

- P(w) = 6/7
- P(r|w) = 1/2
- P(g|w) = 1/2
- P(b) = 1/7
- P(r|b) = 1
- P(g|b) = 0
- P(r,r,b) = (1/7)(1)(1) = 4/28
- P(r,g,b) = P(g,r,b) = P(g,g,b) = 0
- P(*,*,w) = (6/7)(1/2)(1/2) = 6/28

P(g,g,w) = 6/28???

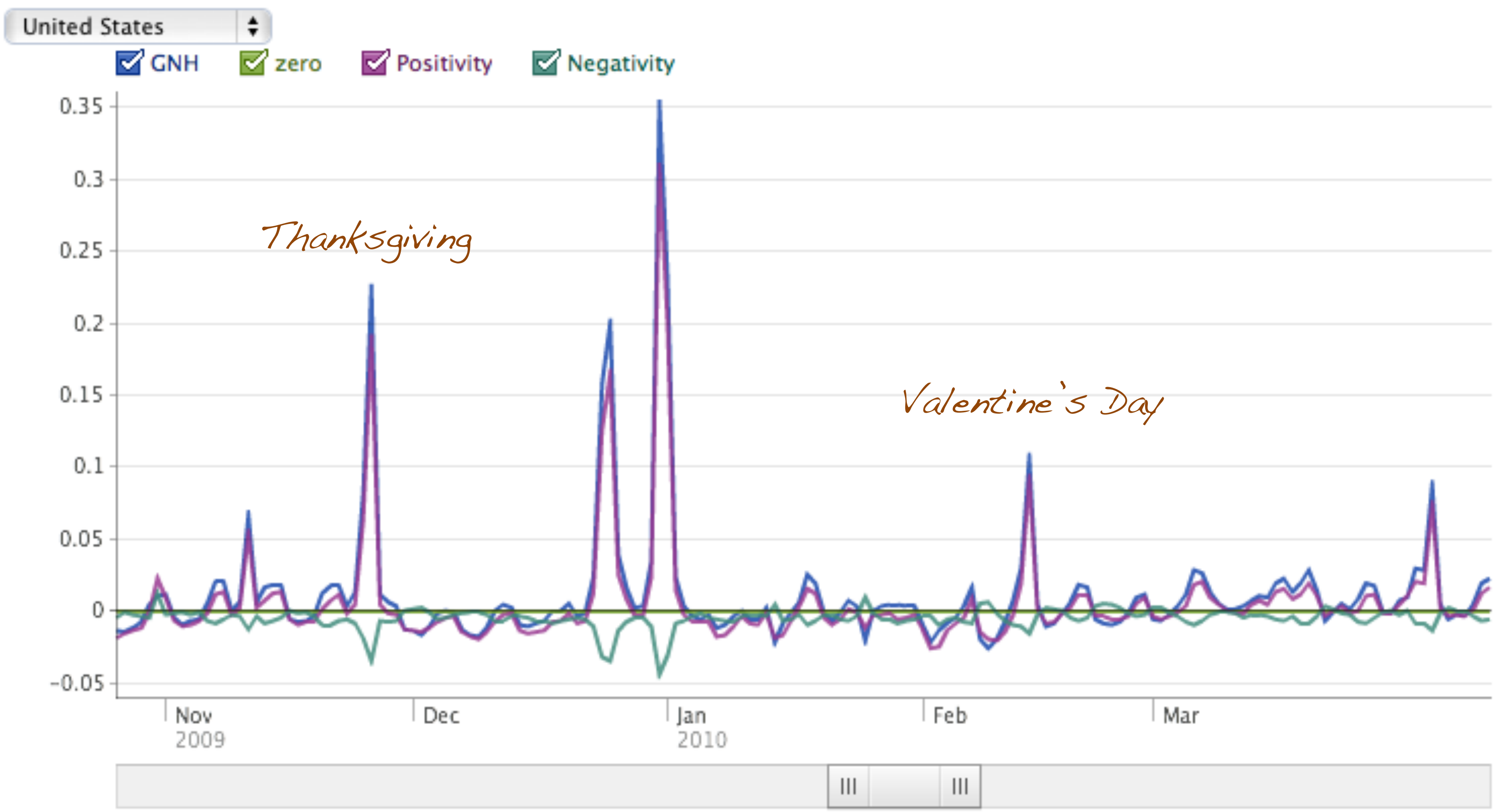
Klein & Manning. MaxEnt Models, Conditional Estimation and Optimization. ACL 2003

Sentiment Analysis

as an example **domain** for text classification

Cristopher Potts. Sentiment Symposium Tutorial. 2011
http://sentiment.christopherpotts.net/index.html

Facebook's "Gross National Happiness" Timeline



Source: Cristopher Potts. Sentiment tutorial. 2011

Opinion/Sentiment Analysis

- Harder than “regular” document classification
  - irony, neutral (“non-polar”) sentiment, negations (“not good”), syntax is used to express emotions (“!”), context dependent
- Confounding polarities from individual aspects (phrases)
  - e.g., a car company’s “customer service” vs. the “safety” of their cars
- Strong commercial interest in this topic
  - “Social” (commercial?) networking sites (FB, G+, …; advertisement)
  - Reviews (Amazon, Google Maps), blogs, fora, online comments, …
  - Brand reputation and political opinion analysis

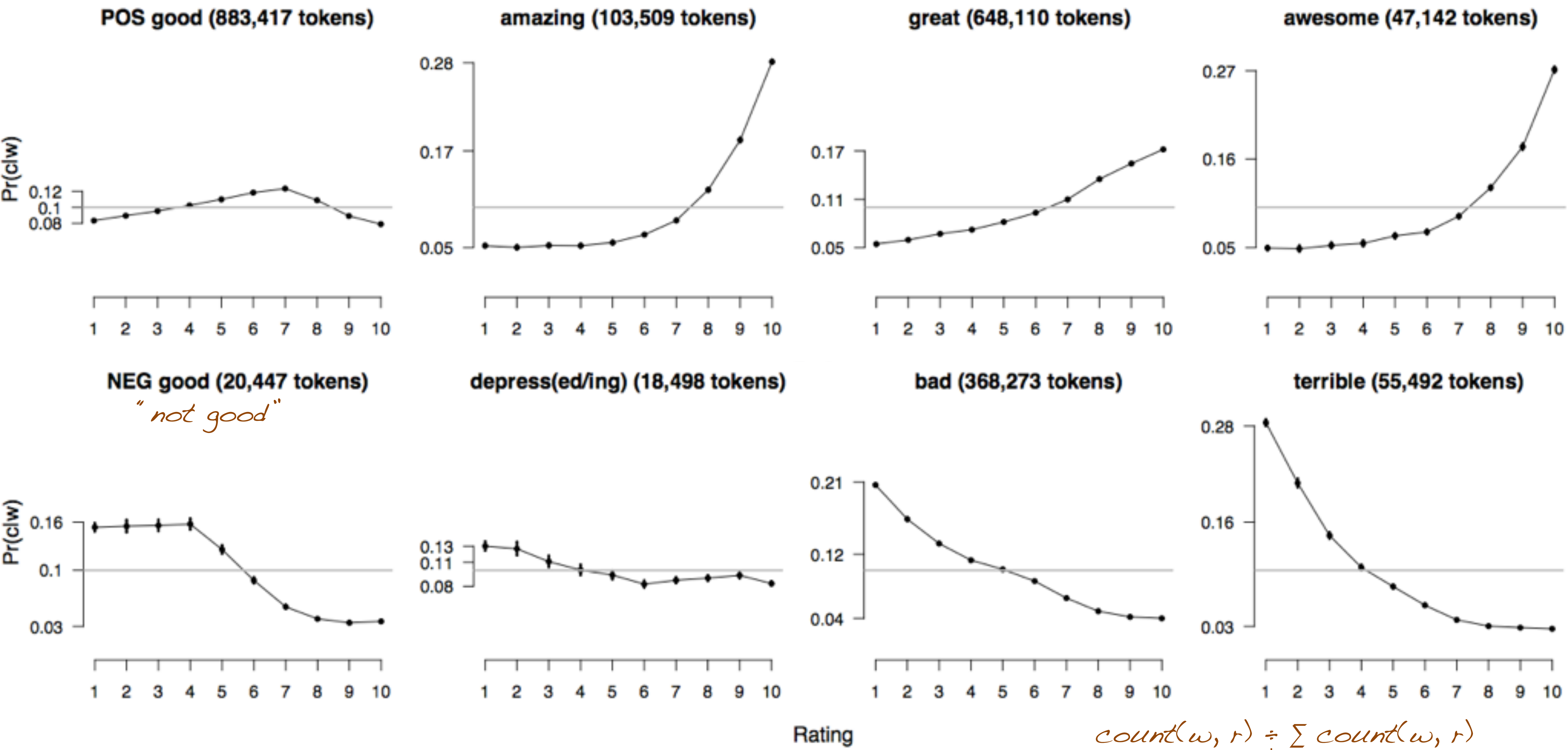
5+1 Lexical Resources for Sentiment Analysis

Cristopher Potts. Sentiment Symposium Tutorial. 2011

Disagree-ment	Opinion Lexicon	General Inquirer	SentiWordNet	LIWC
Subjectivity Lexicon	33/5402 (0.6%)	49/2867 (2%)	1127/4214 (27%)	12/363 (3%)
Opinion Lexicon		32/2411 (1%)	1004/3994 (25%)	9/403 (2%)
General Inquirer			520/2306 (23%)	1/204 (0.5%)
SentiWord Net				174/694 (25%)

MPQA Subjectivity Lexicon:	http://mpqa.cs.pitt.edu/
Liu's Opinion Lexicon:	http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html
General Inquirer:	http://www.wjh.harvard.edu/~inquirer/
SentiWordNet:	http://sentiwordnet.isti.cnr.it/
LIWC (commercial, $90):	http://www.liwc.net/
NRC Emotion Lexicon (+1):	http://www.saifmohammad.com/ (➡Publications & Data)

Polarity of Sentiment Keywords in IMDB



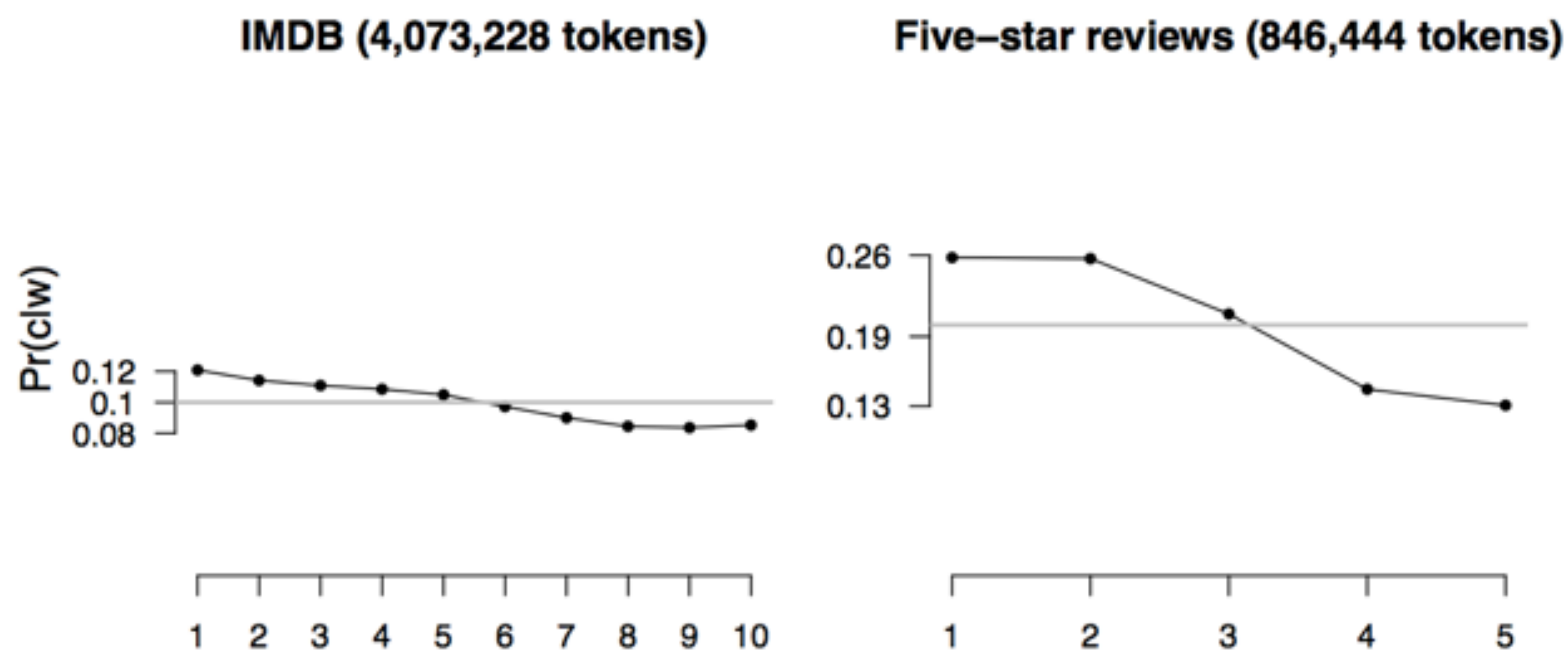
count(w, r) ÷ ∑ count(w, r)

Cristopher Potts. On the negativity of negation. 2011

Note: P(rating | word) = P(word | rating) ÷ P(word)

Negations

- Is the use of negation associated to polarity?



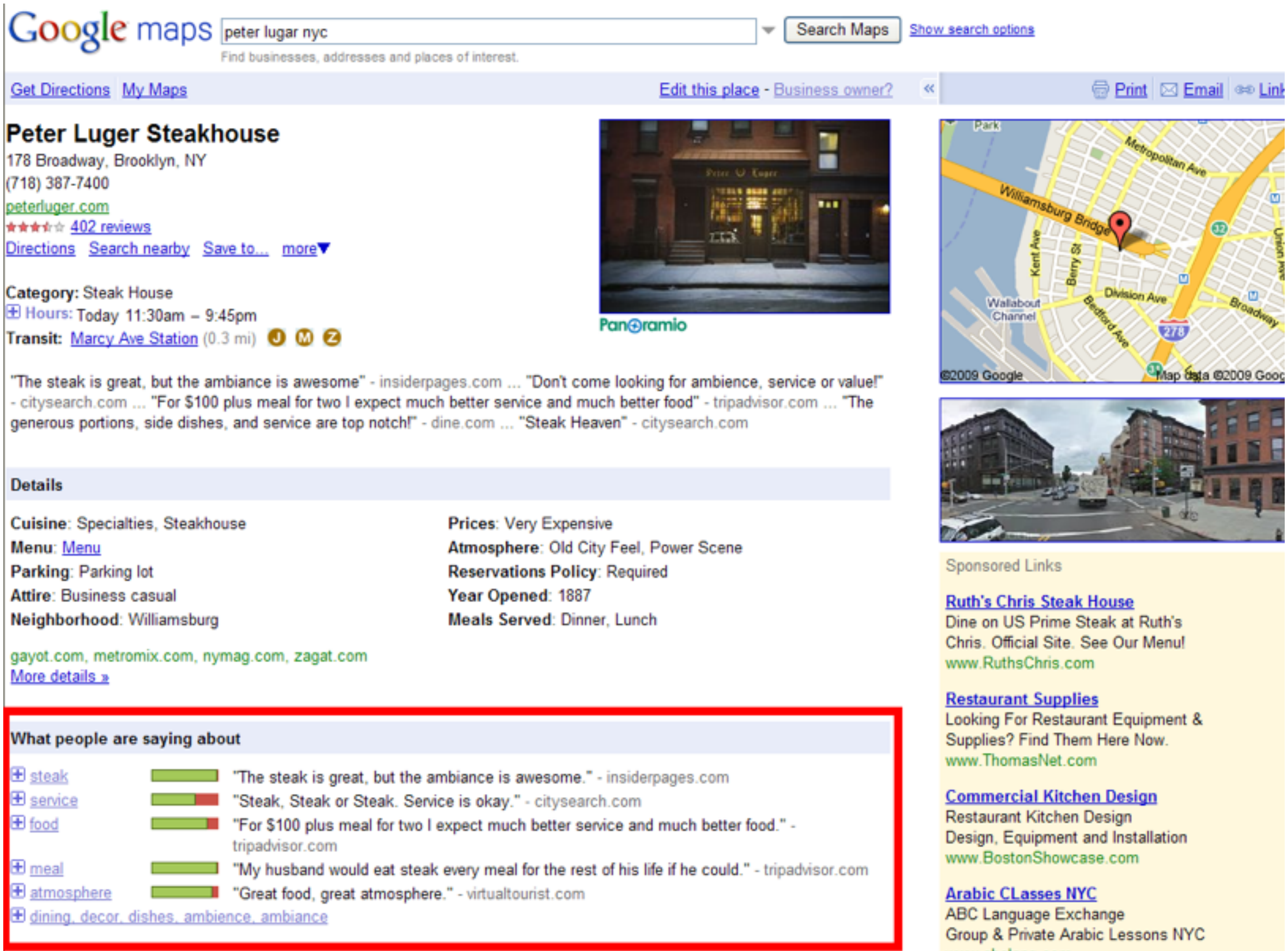
Cristopher Potts. On the negativity of negation. 2011

- **Yes**, **but** there are far more deeper issues going on…

Detecting the Sentiment of Individual Aspects

- Goal: Determine the sentiment for a particular aspect or establish their polarity.
  - An “aspect” here is a phrase or concept, like “customer service”.
  - “They have a **great**$_{+}$ customer service team, but the delivery **took ages**$_{-}$.”
- Solution: Measure the co-occurrence of the aspect with words of distinct sentiment or relative co-occurrence with words of the same polarity.
  - The “sentiment” keywords are taken from some lexical resource.

Google's Review Summaries



Point-wise Mutual Information

- Mutual Information measures the dependence of two variables.

$$I(X;Y) = \sum_{Y} \sum_{X} P(x,y)\ log_2 \frac{P(x,y)}{P(x)P(y)}$$

- Point-wise MI: MI of two **individual** events only
  - e.g., neighboring words, phrases in a document, …
  - even a mix of two co-occurring events (e.g. a word and a phrase)

NB: define a maximum distance between w_1 and w_2

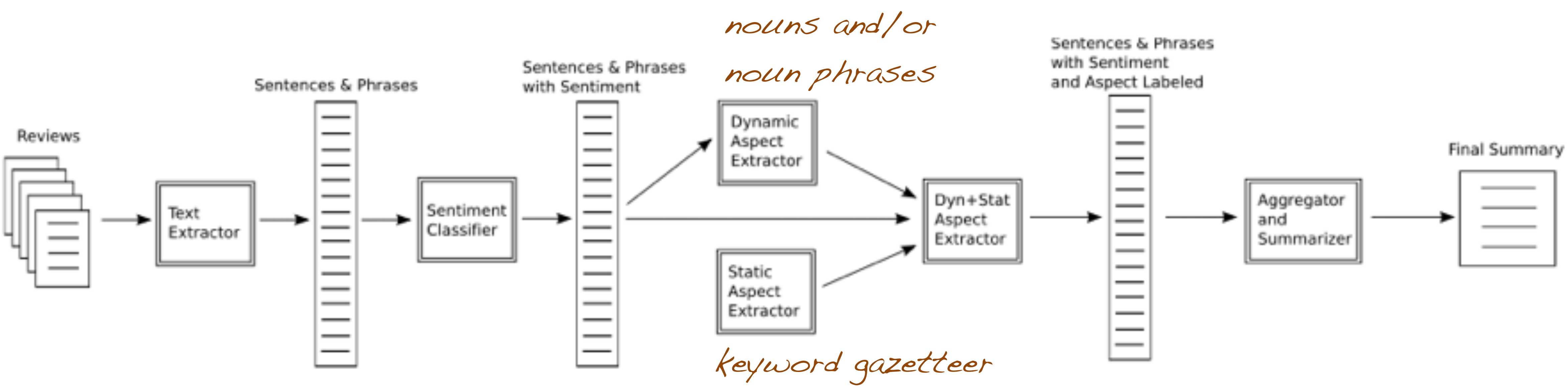
$$PMI(w_1, w_2) = log_2 \frac{P(w_1, w_2)}{P(w_1)P(w_2)}$$

w_1: a phrase/aspect
w_2: one of a set of pos./neg. sentiment words

- Can be normalized to a [-1,1] range: $\frac{PMI(w_1, w_2)}{-log_2\ P(w_1, w_2)}$
  - -1: the two words/phrases/events do not occur together; 0: the words/phrases/events are independent; +1: the words/phrases/events always co-occur

Using PMI to Detect Aspect Polarity

- **Polarity(aspect)** := PMI(*aspect*, pos-sent-kwds) - PMI(*aspect*, neg-sent-kwds)
  - Polarity > 0 = positive sentiment
  - Polarity < 0 = negative sentiment
- Google's approach:



- Blair-Goldensohn et al. Building a Sentiment Summarizer for Local Service Reviews. WWW 2008

Subjectivity Clues and Polarity Intensifiers

subjectivity clues

(4) Philip Clapp, president of the National Environment Trust, sums up well the general thrust of the reaction of environmental movements: "There is no reason at all to believe that the polluters are suddenly going to become reasonable."

unmodified

polarity intensifiers

long-distance negations

- Known as the problem of "**contextual polarity**":
  - The **evil** baron was **held in check**. ("evil"=neg. subjective expression, "held in check"=context reverses neg. subjective expression)
- Wilson et al. Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. HLT-EMNLP 2005
  - MPQA Subjectivity Lexicon: http://mpqa.cs.pitt.edu/lexicons/subj_lexicon/

More Context Issues…

- **Feelings**: "it is too bad", "I am very sorry for you", …
- **Agreement**: "I must agree to", "it is the case that", …
- **Hedging**: "a little bit", "kind/sort of", …
- ….

(not part of this course)

- Requires the use of **dependency parsing** to detect the relevant **context**

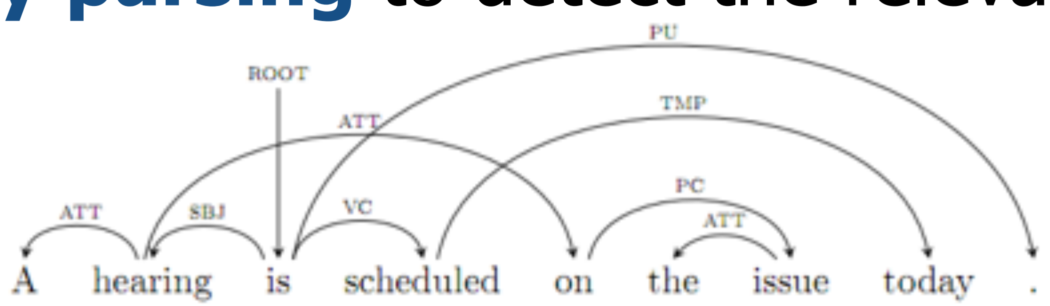


Image Source: WikiBooks, Daniele Pighin

The Take-Home Message:

- Inter-rater **human agreement** for sentiment tasks
  - often only at around 80% (Cohen's κ ~ 0.7)
  - sentiment expressions have a very high "uncertainty" (ambiguity)

Evaluation Metrics

Evaluation is all about answering questions like:

How to measure a change to an approach?

Did adding that feature improve or decrease performance?

Is the approach good at locating the relevant pieces or good at excluding the irrelevant bits?

How to compare entirely different methods?

this one is really hard...

Basic Evaluation Metrics: Accuracy, F-Measure, MCC Score

- **True/False Positive/Negative**
  - counts; TP, TN, FP, FN
- **Precision** (P)
  - correct hits [TP] ÷ all hits [TP + FP]
- **Recall** (R; **Sensitivity**, TPR)
  - correct hits [TP] ÷ true cases [TP + FN]
- **Specificity** (True Negative Rate)
  - correct misses [TN] ÷ negative cases [FP + TN]

NB: no result order:
lesson 5 (tomorrow)

- **Accuracy**
  - correct classifications [TP + TN] ÷ all cases [TP + TN + FN + FP])
  - highly **sensitive to** class **imbalance**
- **F-Measure** (F-Score)
  - the harmonic mean between P & R
    = 2 TP ÷ (2 TP + FP + FN)
    = (2 P R) ÷ (P + R)
  - does **not** require a **TN** count
- **MCC Score** (Mathew's Correlation Coefficient)
  - χ^2**-based**: (TP TN - FP FN) ÷ sqrt[(TP+FP)(TP+FN)(TN+FP)(TN+FN)]
  - **robust against** class **imbalance**

Practical: Twitter Sentiment Detection

- Implement a MaxEnt classifier to detect the sentiment of Twitter “tweets” for Apple/iPhone and Google/Android products.
- Try to improve the result of 70 % accuracy by choosing better features and implementing a better tokenization strategy.
- Experiment with making use of the sentiment clues and polarity intensifiers from the Subjectivity Lexicon.

"Romance should never begin with sentiment.
It should begin with science and end with a settlement."

Oscar Wilde, 1895