Text Classification

Naïve Bayes

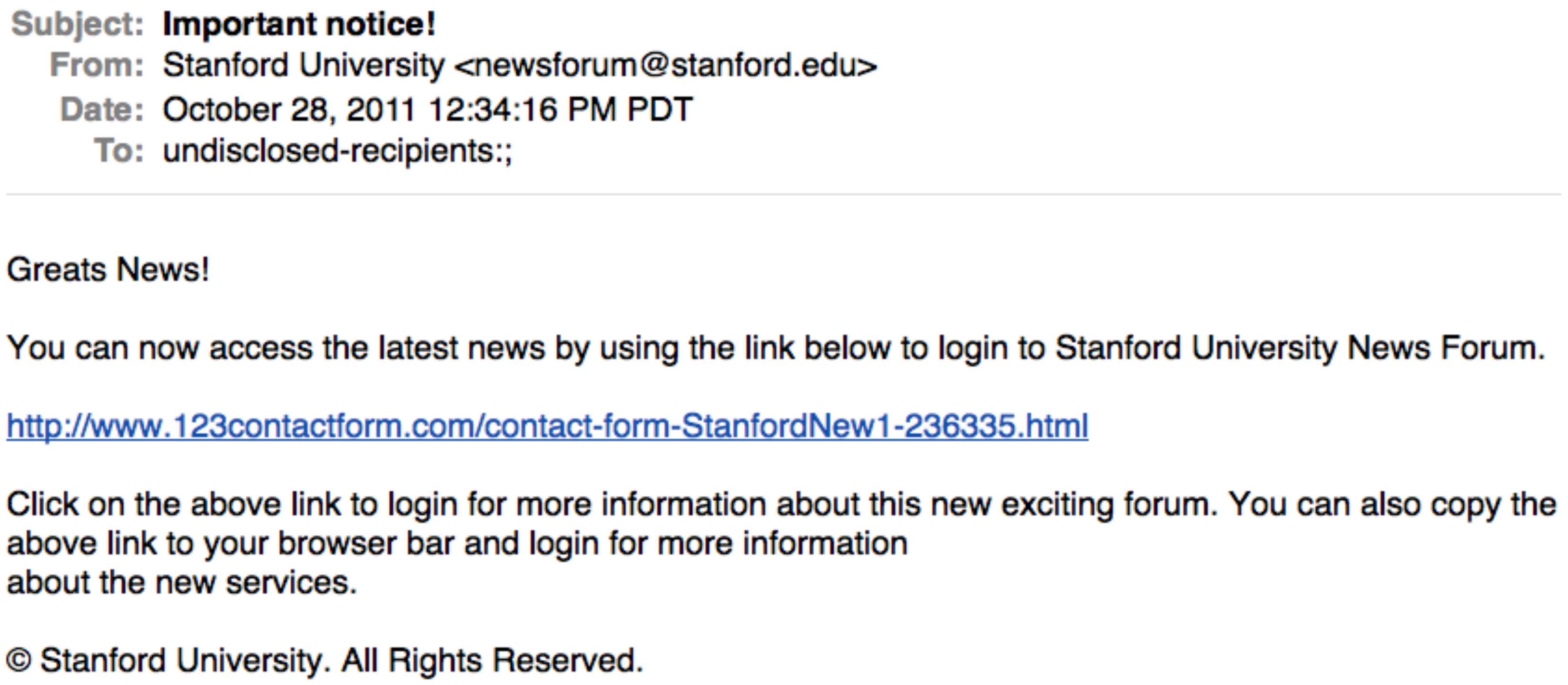
Based on *Speech and Language Processing*
(3rd ed. draft, 2017)
by Dan Jurafsky and James H. Martin

https://web.stanford.edu/~jurafsky/slp3/

Normunds Grūzītis
DatZ1131: *Ievads dabiskās valodas apstrādē*
Latvijas Universitāte

Modifikācija *Python* specseminrām

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!

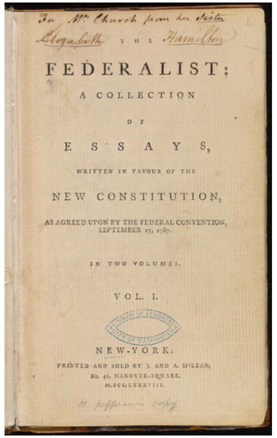
You can now access the latest news by using the link below to login to Stanford University News Forum.

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.

Who wrote which Federalist papers?


For Mrs Church from her Sister
Elizabeth THE Hamilton

FEDERALIST;

A COLLECTION

OF

E S S A Y S,

WRITTEN IN FAVOUR OF THE

NEW CONSTITUTION,

AS AGREED UPON BY THE FEDERAL CONVENTION, SEPTEMBER 17, 1787.

IN TWO VOLUMES.

VOL. I.

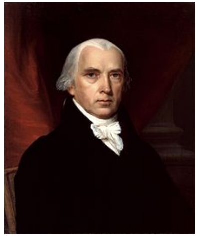
LIBRARY OF CONGRESS • CITY OF WASHINGTON •

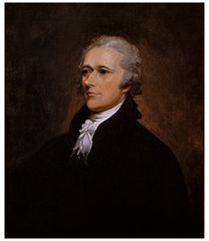
NEW-YORK:

PRINTED AND SOLD BY J. AND A. M'LEAN,
No. 41, HANOVER-SQUARE.
M,DCC,LXXXVIII.

Mr. Jefferson's copy

- 1787-8: anonymous essays try to convince New York to ratify U.S Constitution (by Jay, Madison, Hamilton)
- Authorship of 12 of the letters in dispute
- 1963: solved by Mosteller and Wallace using **Bayesian** methods


James Madison


Alexander Hamilton

Male or female author?

1. *By 1925 present-day Vietnam was divided into three parts under French colonial rule. The southern region embracing Saigon and the Mekong delta was the colony of Cochin-China; the central area with its imperial capital at Hue was the protectorate of Annam…*
2. *Clara never failed to be astonished by the extraordinary felicity of her own name. She found it hard to trust herself to the mercy of fate, which had managed over the years to convert her greatest shame into one of her greatest assets…*

S. Argamon, M. Koppel, J. Fine, A. R. Shimoni, 2003. "Gender, Genre, and Writing Style in Formal Written Texts," Text, volume 23, number 3, pp. 321–346

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

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

- Assigning subject/topic categories
- Spam detection
- Authorship identification
- Age/gender identification
- Language Identification
- Sentiment analysis
- Intent classification (virtual assistants)
- …

Text Classification

- Text-level classification
  - vs. sentence/phrase-level classification
  - vs. character-level classification

- Segmentation and classification of language units within a text unit
  - e.g. Named Entity Recognition

Demo: `langid.py`

- https://github.com/saffsd/langid.py

```
>>> import langid
>>> langid.classify(“this is a test”)
('en', -54.41310358047485)
>>> langid.classify(“šis ir tests”)
('lt', -18.75309062004894)
>>> langid.set_languages(['en','lv','ru'])
>>> langid.classify(“šis ir tests”)
('lv', -33.77597045898437 5)
```

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: Hand-coded rules

- Rules based on combinations of words or other features
  - Spam: black-list-address OR (“dollars” AND “have been selected”)
- Precision can be high (vs. recall)
  - If rules carefully refined by expert
- But it is very difficult to scale up

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 $\gamma : d \rightarrow c$
    - a probabilistic classifier: $P(c|d)$

Classification Methods: Supervised Machine Learning

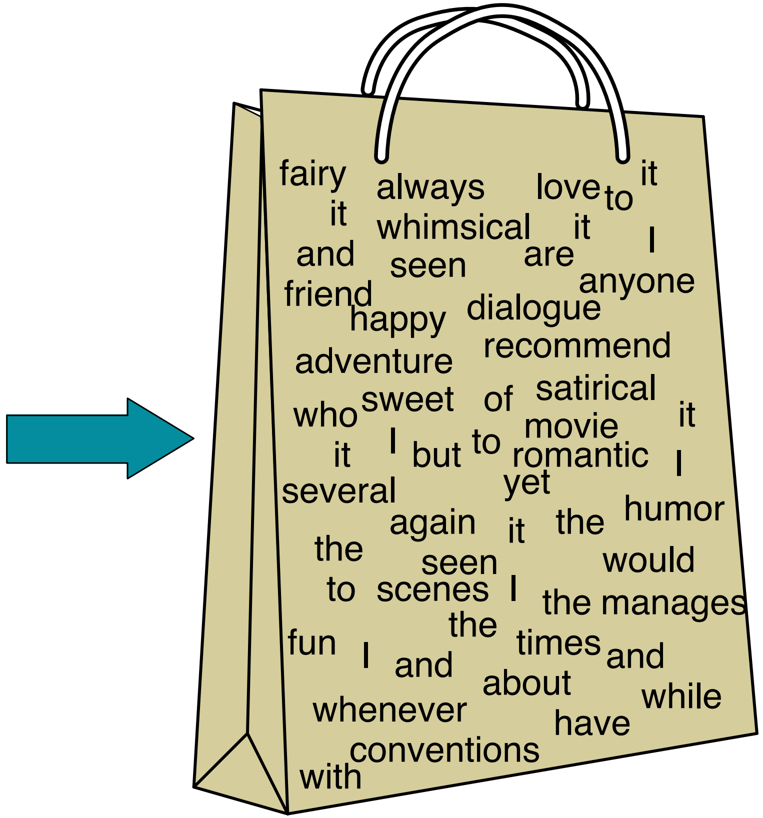
- Any kind of classifier
  - **Naïve Bayes**
  - Logistic regression
  - Support-vector machines
  - k-Nearest Neighbors
  - Perceptron
  - Neural networks

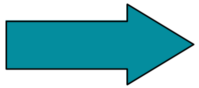
Naïve Bayes Intuition

- Simple classification method
  - Based on the Bayes rule
  - With a naïve assumption of feature independence
- Relies on very simple <u>representation</u> 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

Thomas Bayes
(1701-1761)

- For a document *d* and a class *c*

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

Naïve Bayes Classifier

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

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

Naïve Bayes Classifier

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

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

Document d represented as features x1..xn

Naïve Bayes Classifier

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

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

Naïve 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.

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

Naïve Bayes Classifier

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

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

Applying a Naive Bayes Classifier

$\text{positions} \leftarrow$ all word positions in test document

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

Learning the Naïve Bayes Model

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

$$\hat{P}(c_j) = \frac{doccount(C = c_j)}{N_{doc}}$$

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

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 a mega-document for topic *j* by concatenating all docs in this topic
  - Use frequency of *w* in the mega-document

Problem with Maximum Likelihood

- What if we have seen no training documents with the word ***fantastic*** 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

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

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

Underflow Prevention: log space

- Multiplying lots of probabilities can result in floating-point underflow
- Since log(*xy*) = log(*x*) + log(*y*)
  - Better to sum logs of probabilities instead of multiplying probabilities
- Class with highest un-normalized log probability score is still most probable

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

- Model is now just max of sum of weights

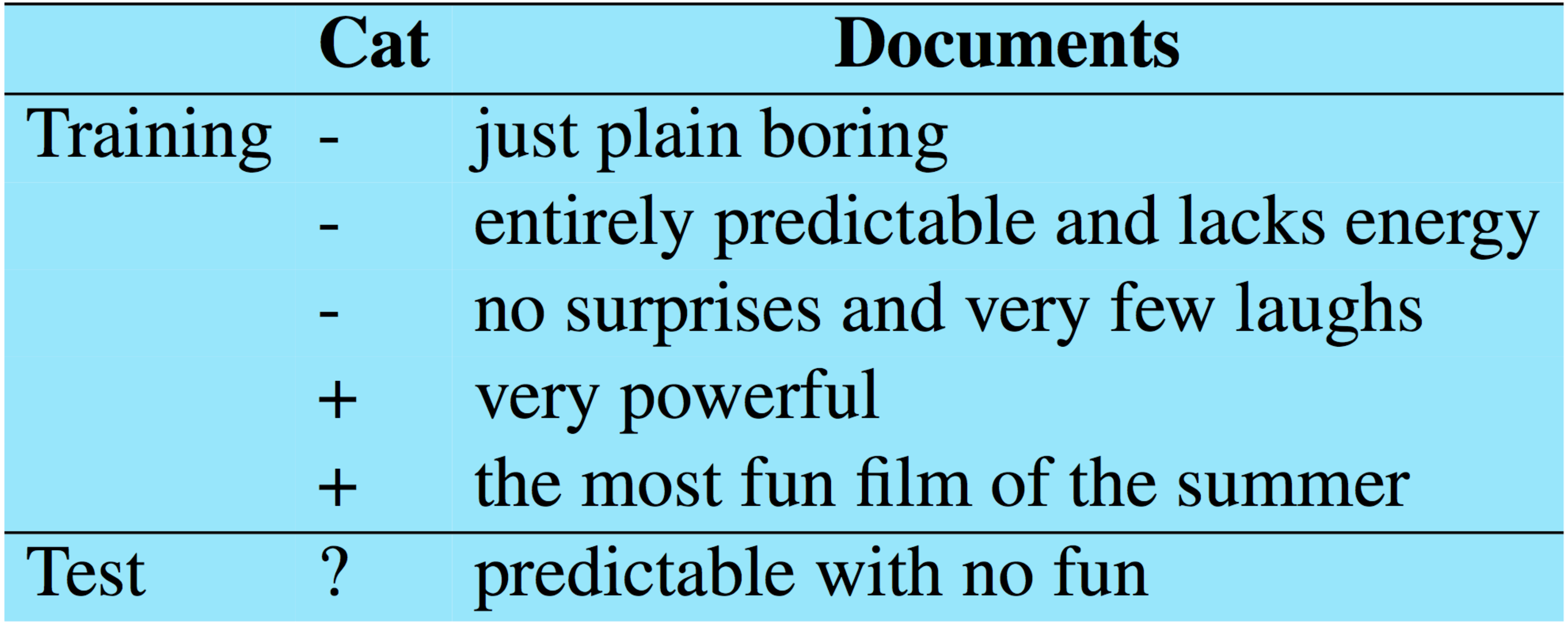
Naïve Bayes: Learning

```
function TRAIN NAIVE BAYES(D, C) returns log $P(c)$ and log $P(w|c)$

for each class $c \in C$                # Calculate $P(c)$ terms
    $N_{doc}$ = number of documents in D
    $N_c$ = number of documents from D in class c
    $logprior[c] \leftarrow \log \frac{N_c}{N_{doc}}$
    $V \leftarrow$ vocabulary of D
    $bigdoc[c] \leftarrow$ append(d) for d $\in$ D with class $c$
    for each word $w$ in V                # Calculate $P(w|c)$ terms
        $count(w,c) \leftarrow$ # of occurrences of $w$ in $bigdoc[c]$
        $loglikelihood[w,c] \leftarrow \log \frac{count(w,c) + 1}{\sum_{w' \text{ in } V} (count(w',c) + 1)}$
return $logprior$, $loglikelihood$, $V$
```

Naïve Bayes: Testing

```
function TEST NAIVE BAYES(testdoc, logprior, loglikelihood, C, V) returns best c

for each class $c \in C$
    $sum[c] \leftarrow logprior[c]$
    for each position $i$ in testdoc
        $word \leftarrow testdoc[i]$
        if $word \in V$
            $sum[c] \leftarrow sum[c] + loglikelihood[word,c]$
return $\text{argmax}_c\ sum[c]$
```

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

Example

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

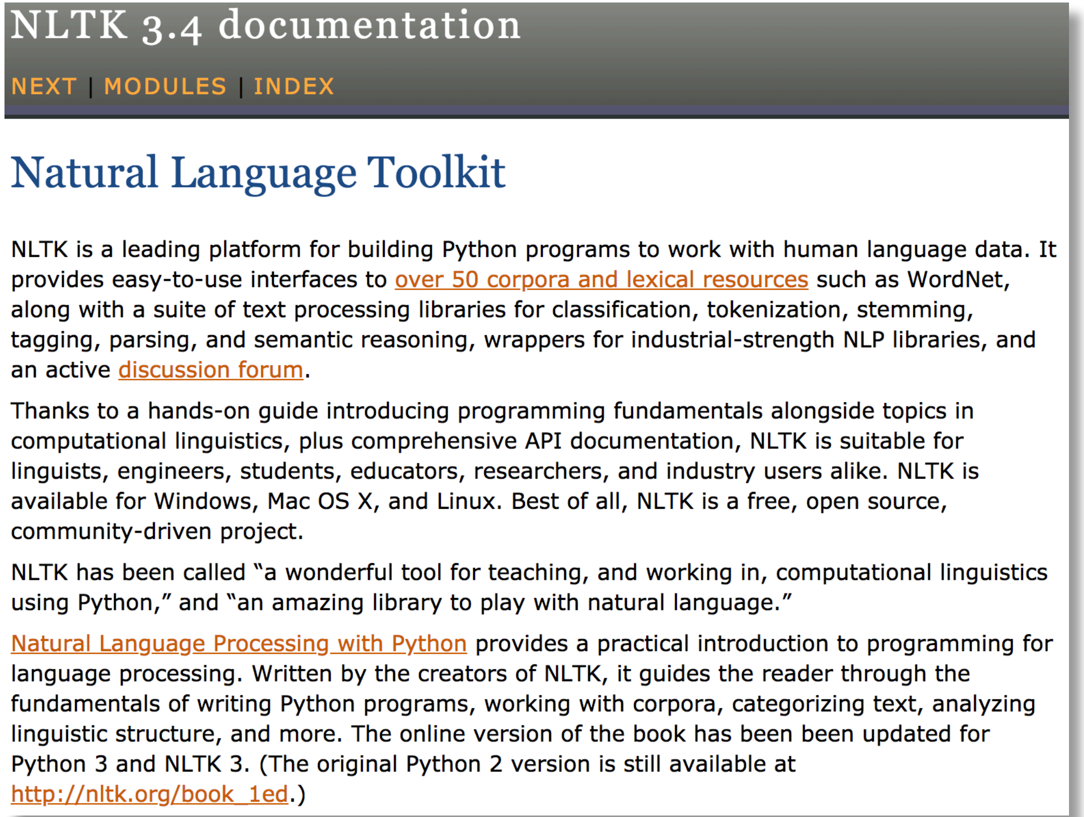
$$P(+)P(S|+) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

- P(c) = N_c / N_{doc}
  - P(+) = 2 / 5 ; P(-) = 3 / 5
- P(w_i|c) = $\frac{count(w_i, c) + 1}{(\sum_{w \in V} count(w, c)) + |V|}$
  - $P(\text{"predictable"}|-) = \frac{1+1}{14+20}$ $\quad$ $P(\text{"predictable"}|+) = \frac{0+1}{9+20}$

    $P(\text{"no"}|-) = \frac{1+1}{14+20}$ $\quad$ $P(\text{"no"}|+) = \frac{0+1}{9+20}$

    $P(\text{"fun"}|-) = \frac{0+1}{14+20}$ $\quad$ $P(\text{"fun"}|+) = \frac{1+1}{9+20}$

Demo: `nltk.NaiveBayesClassifier`



NLTK 3.4 documentation

NEXT | MODULES | INDEX

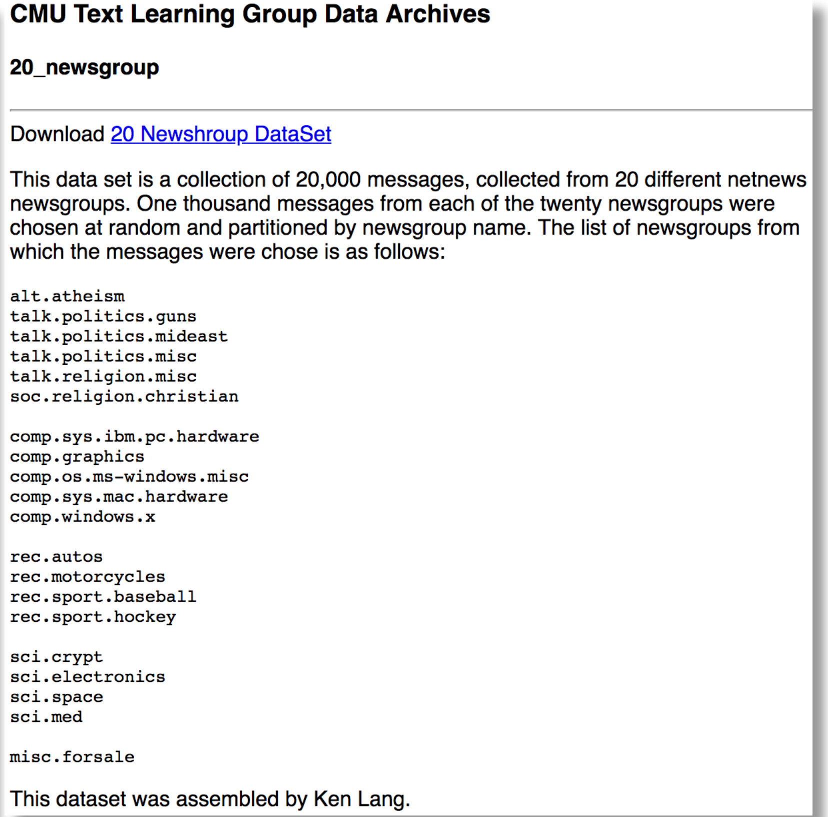
Natural Language Toolkit

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

Thanks to a hands-on guide introducing programming fundamentals alongside topics in computational linguistics, plus comprehensive API documentation, NLTK is suitable for linguists, engineers, students, educators, researchers, and industry users alike. NLTK is available for Windows, Mac OS X, and Linux. Best of all, NLTK is a free, open source, community-driven project.

NLTK has been called "a wonderful tool for teaching, and working in, computational linguistics using Python," and "an amazing library to play with natural language."

Natural Language Processing with Python provides a practical introduction to programming for language processing. Written by the creators of NLTK, it guides the reader through the fundamentals of writing Python programs, working with corpora, categorizing text, analyzing linguistic structure, and more. The online version of the book has been been updated for Python 3 and NLTK 3. (The original Python 2 version is still available at http://nltk.org/book_1ed.)



CMU Text Learning Group Data Archives

20_newsgroup

Download 20 Newshroup DataSet

This data set is a collection of 20,000 messages, collected from 20 different netnews newsgroups. One thousand messages from each of the twenty newsgroups were chosen at random and partitioned by newsgroup name. The list of newsgroups from which the messages were chose is as follows:

```
alt.atheism
talk.politics.guns
talk.politics.mideast
talk.politics.misc
talk.religion.misc
soc.religion.christian

comp.sys.ibm.pc.hardware
comp.graphics
comp.os.ms-windows.misc
comp.sys.mac.hardware
comp.windows.x

rec.autos
rec.motorcycles
rec.sport.baseball
rec.sport.hockey

sci.crypt
sci.electronics
sci.space
sci.med

misc.forsale
```

This dataset was assembled by Ken Lang.

Demo: `nltk.NaiveBayesClassifier`

- `import nltk`
- **Training**:
  - `nb = nltk.NaiveBayesClassifier.`**`train`**`(`*`vectorize`*`(TRAIN_DATA))`
  - `dmp = open("nb_classifier.pickle", "wb")`
  - `pickle.dump(nb, dmp)`
- **Using**:
  - `dmp = open("nb_classifier.pickle", "rb")`
  - `nb = pickle.load(dmp)`
  - `nb.`**`classify`**`(`*`vectorize`*`(TEXT))`
  - `nb.`**`prob_classify`**`(`*`vectorize`*`(TEXT))`

Text normalization

- Lemmatization (pros/cons)
- Named entities (overfitting)
  - amounts, dates/times, person names
  - places, organizations, products
- Rare words (the long tail → size of the vector)
- Frequent typos (data sparsity)
- Stoplist (frequent non-sense words, etc.)
- English vs. inflected languages

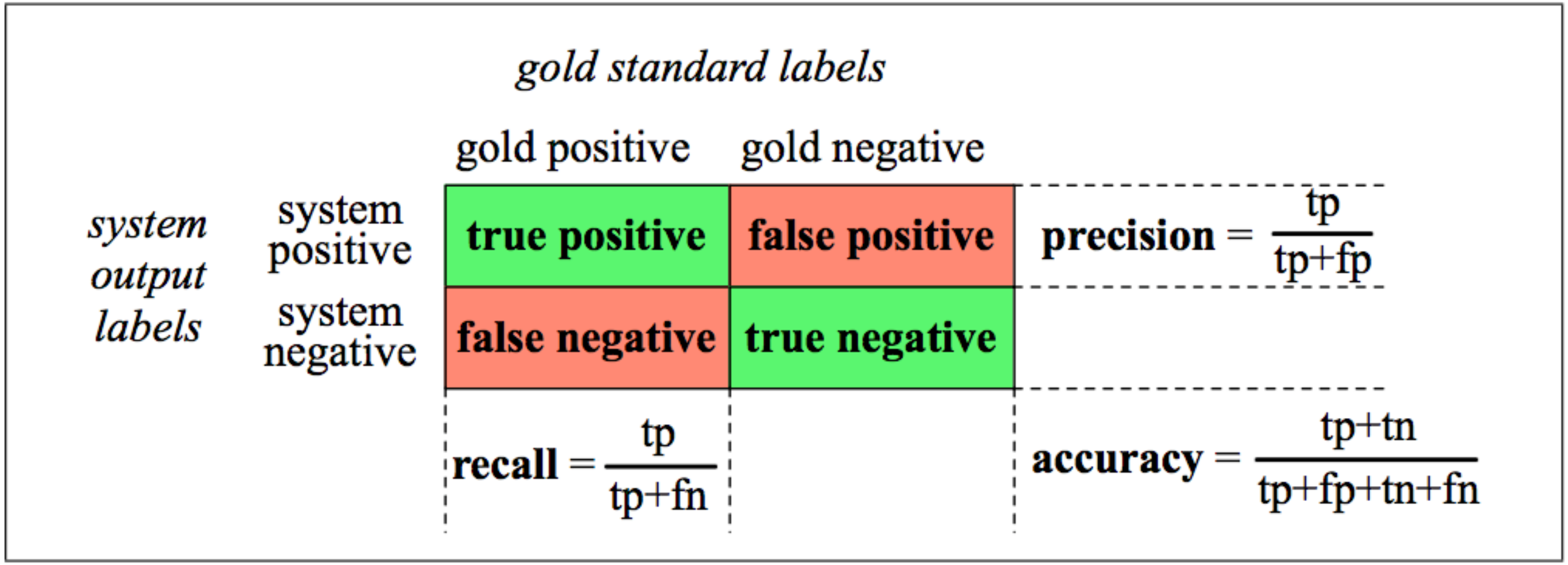
Text Classification and Naïve Bayes

Evaluation

Datasets

- Split the annotated examples split into:
  - Training dataset (e.g. 70%)
  - Validation or development dataset (e.g. 15%); optionally
  - Test dataset (e.g. 15%)
- vs. k-fold cross-validation
  - k folds (groups) of approximately equal size
  - Each sample is given the opportunity to be used in the hold out set 1 time and used to train the model k-1 times

The 2-by-2 contingency table



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 guesses that are correct
Recall: % of all correct cases that are guessed

But beware of accuracy!

Accuracy

- Asks what % of all the observations our system labeled correctly
- Does not work well when the classes are **unbalanced**
  - When the goal is to discover something that is rare – unbalanced in frequency (a very common situation)
- Example:
  - Dataset: 1M tweets of which 10K are positive cases (e.g. specific emotion)
  - The most frequent class approach: P(-) = 99% vs. P(+) = 1%
- Therefore we turn to the **trade-off** between precision and recall

A combined F-measure

- A combined measure that assesses the P/R trade-off is F-measure (weighted harmonic mean):

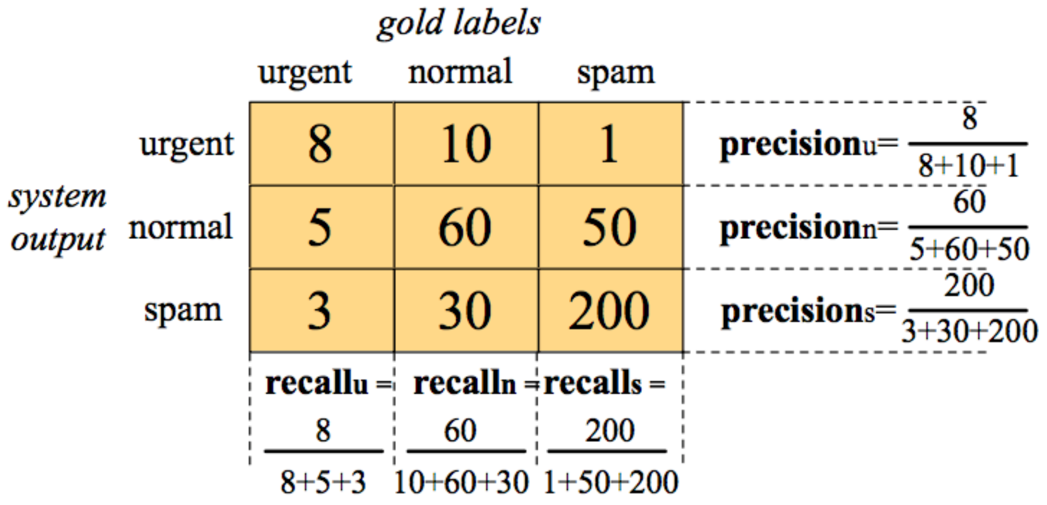
$$F_\beta = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

- The β parameter differentially weights the importance of R and P
  - Values of $\beta > 1$ favor R, while values of $\beta < 1$ favor P
  - When $\beta = 1$, P and R are equally balanced
- The balanced F_1 measure is usually used:

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

Confusion matrix

- For each pair of classes $<c_1,c_2>$ how many documents from c_1 were incorrectly assigned to c_2?
  - e.g. 3-way one-of email categorization

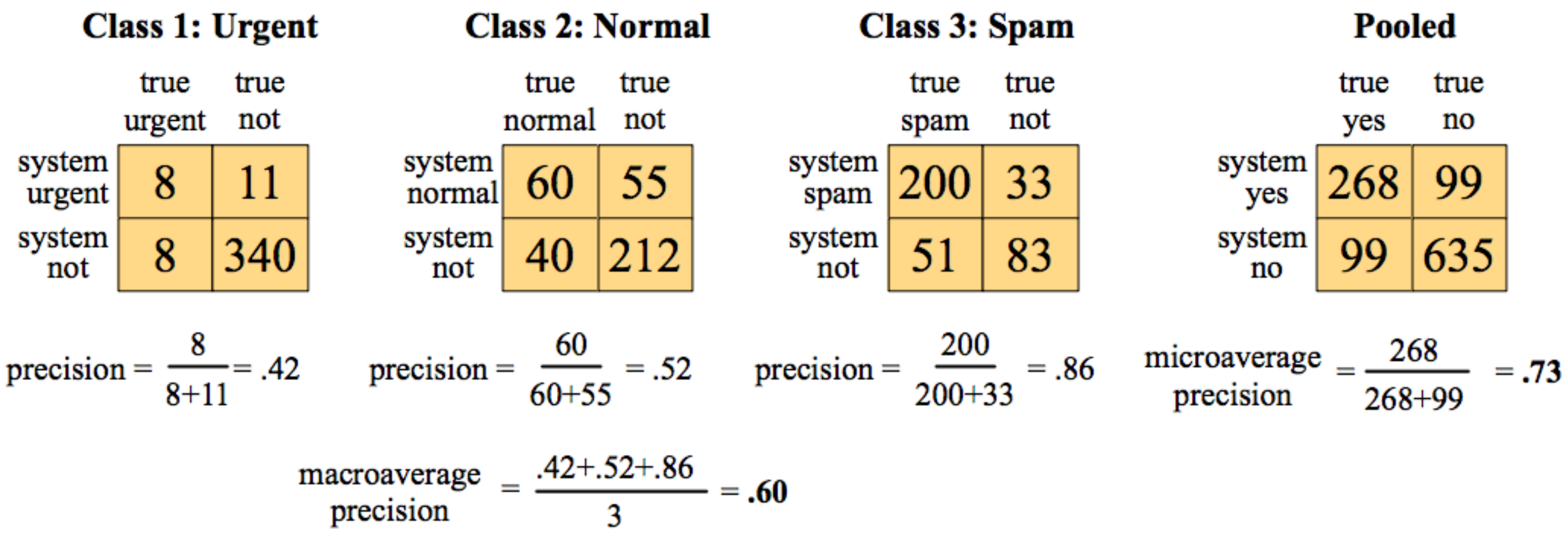


		gold labels				
		urgent	normal	spam		
system output	urgent	8	10	1	**precision**u= $\frac{8}{8+10+1}$	0.42
	normal	5	60	50	**precision**n= $\frac{60}{5+60+50}$	0.52
	spam	3	30	200	**precision**s= $\frac{200}{3+30+200}$	0.86
		recallu = $\frac{8}{8+5+3}$	**recall**n = $\frac{60}{10+60+30}$	**recall**s = $\frac{200}{1+50+200}$		
		0.50	0.60	0.80		

Micro- vs. Macro-Averaging

- If we have more than one class, how do we combine multiple performance measures into one quantity?
- **Macroaveraging**: compute performance for each class, then average
- **Microaveraging**: collect decisions for all classes, compute contingency table, evaluate

Micro- vs. Macro-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

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

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

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

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

- Microaveraged score is dominated by the more frequent classes
- Macroaverage better reflects the statistics of the smaller classes, and so is more appropriate when performance on all the classes is equally important

Demo: scikit-learn

- `from sklearn.model_selection import` **`Kfold`**
- `from sklearn.metrics import` **`classification_report`**
- `from sklearn.metrics import` **`confusion_matrix`**

- `kfold =` **`KFold`**`(n_splits=k, shuffle=True)`
- `kfold.`**`split`**`(DATA_SET)`

- **`classification_report`**`(GOLD_ANSWERS, SILVER_ANSWERS)`
- **`confusion_matrix`**`(GOLD_ANSWERS, SILVER_ANSWERS)`

Text Classification and Naïve Bayes

Multinominal vs. multivalue classification

More Than Two Classes: Sets of binary classifiers

- Dealing with any-of or multivalue classification
  - A document can belong to 0, 1, or >1 classes
- For each class $c \in C$
  - Build a (binary) classifier γ_c to distinguish c from all other classes $c' \in C$
- Given test doc d,
  - Evaluate it for membership in each class using each γ_c
  - d belongs to any class for which γ_c returns true

More Than Two Classes: Sets of binary classifiers

- One-of or multinomial classification
  - Classes are mutually exclusive: each document in exactly one class
- For each class $c \in C$
  - Build a classifier γ_c to distinguish c from all other classes $c' \in C$
- Given test doc d,
  - Evaluate it for membership in each class using each γ_c
  - d belongs to the one class with maximum score

Evaluation: Classic Reuters-21578 Data Set

- Most (over)used data set, 21,578 docs (each 90 types, 200 tokens)
- 9603 training, 3299 test articles (ModApte/Lewis split)
- 118 categories
  - An article can be in more than one category
  - Learn 118 binary category distinctions
- Average document (with at least one category) has 1.24 classes
- Only about 10 out of 118 categories are large

Common categories (#train, #test)

- Earn (2877, 1087)
- Acquisitions (1650, 179)
- Money-fx (538, 179)
- Grain (433, 149)
- Crude (389, 189)
- Trade (369,119)
- Interest (347, 131)
- Ship (197, 89)
- Wheat (212, 71)
- Corn (182, 56)

Reuters Text Categorization data set (Reuters-21578) document

```
<REUTERS TOPICS="YES" LEWISSPLIT="TRAIN" CGISPLIT="TRAINING-SET" OLDID="12981"
NEWID="798">

<DATE> 2-MAR-1987 16:51:43.42</DATE>

<TOPICS><D>livestock</D><D>hog</D></TOPICS>

<TITLE>AMERICAN PORK CONGRESS KICKS OFF TOMORROW</TITLE>

<DATELINE>    CHICAGO, March 2 - </DATELINE><BODY>The American Pork Congress kicks off tomorrow,
March 3, in Indianapolis with 160 of the nations pork producers from 44 member states determining industry positions
on a number of issues, according to the National Pork Producers Council, NPPC.

    Delegates to the three day Congress will be considering 26 resolutions concerning various issues, including the future
direction of farm policy and the tax law as it applies to the agriculture sector. The delegates will also debate whether to
endorse concepts of a national PRV (pseudorabies virus) control and eradication program, the NPPC said.

    A large trade show, in conjunction with the congress, will feature the latest in technology in all areas of the industry,
the NPPC added. Reuter

&#3;</BODY></TEXT></REUTERS>
```

Text Classification and Naïve Bayes

Text Classification: Practical Issues

The Real World

- Gee, I’m building a text classifier for real, now!
- What should I do?

No training data? Manually written rules

If (wheat or grain) and not (whole or bread) then

    Categorize as grain

- Need careful crafting
  - Human tuning on development data
  - Time-consuming: 2 days per class

Very little data?

- Use Naïve Bayes
  - Naïve Bayes is a "high-bias" algorithm (Ng and Jordan 2002 NIPS)
- Get more labeled data
  - Find clever ways to get humans to label data for you
- Try semi-supervised training methods:
  - Bootstrapping, EM over unlabeled documents, …

A reasonable amount of data?

- Perfect for all the clever classifiers
  - SVM
  - Regularized Logistic Regression
- You can even use user-interpretable decision trees
  - Users like to hack
  - Management likes quick fixes

A huge amount of data?

- Can achieve high accuracy!
- At a cost:
  - SVMs (train time) or kNN (test time) can be too slow
  - Regularized logistic regression can be somewhat better
- So Naïve Bayes can come back into its own again!

How to tweak performance

- Domain-specific features and weights: *very* important in real performance
- Sometimes need to collapse terms:
  - Part numbers, chemical formulas, …
  - But stemming generally doesn't help
- Upweighting: Counting a word as if it occurred twice:
  - title words (Cohen & Singer 1996)
  - first sentence of each paragraph (Murata, 1999)
  - In sentences that contain title words (Ko *et al*, 2002)

P.S. 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)=Π 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	0.01	0.05	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)

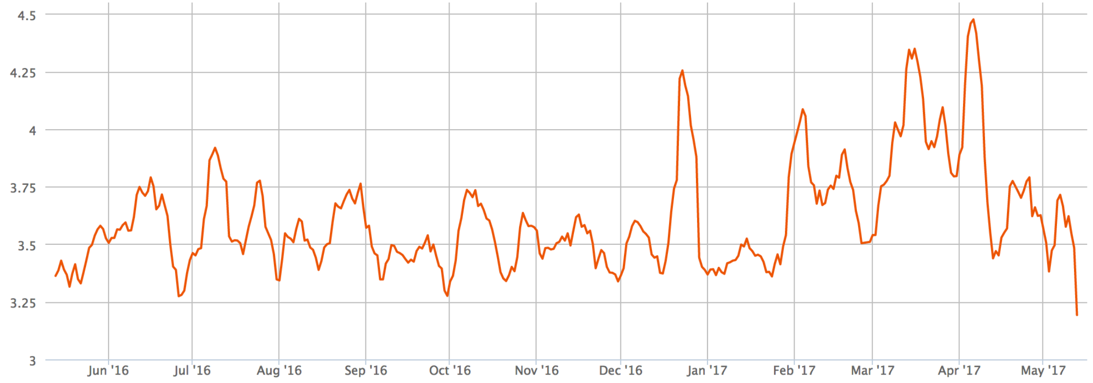
Optimizing for sentiment analysis

- Whether a word occurs or not seems to matter more than its frequency
  - It often improves performance to clip the word counts in each document at 1
    - For each document we remove all duplicate words <u>before</u> concatenating them into the single big document
  - = **Binary** multinominal naive Bayes
- An simple but quite efficient baseline commonly used in sentiment classification to deal with <u>negation</u>:
  - Add prefix *NOT* to every word after a logical negation until a punctuation mark

```
didnt like this movie , but I
didnt NOT_like NOT_this NOT_movie , but I
```

Optimizing for sentiment analysis

- We might have insufficient labeled training data to train an accurate naive Bayes classifier using all words in the training set
- We can instead derive the positive and negative word features from **sentiment lexicons**, e.g. LIWC (Pennebaker et al., 2007)
- See also: http://barometrs.korpuss.lv



Summary: Naive Bayes is Not So Naive

- Very **fast**, **low storage** requirements
- 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 there are other classifiers that give better accuracy