Natural Language Processing: Text Classification

17 February 2022

(FYI) Soundex Code - Phonetic Algorithm

1. Keep the first letter (in upper case).
2. Replace these letters with hyphens: a,e,i,o,u,y,h,w.
3. Replace the other letters by numbers as follows:
    1: b,f,p,v
    2: c,g,j,k,q,s,x,z
    3: d,t
    4: l
    5: m,n
    6: r
4. Delete adjacent repeats of a number.
5. Delete the hyphens.
6. Keep the first three numbers or pad out with zeros.

extenssions → E235; extensions → E235
marshmellow → M625; marshmallow → M625
brimingham → B655; birmingham → B655
poiner → P560; pointer → P536

Python module for computing soundex and other phonetic keys of strings: https://pypi.org/project/phonetics/

(Example of correcting errors using semantic word similarities) Temporal analog retrieval from news archives

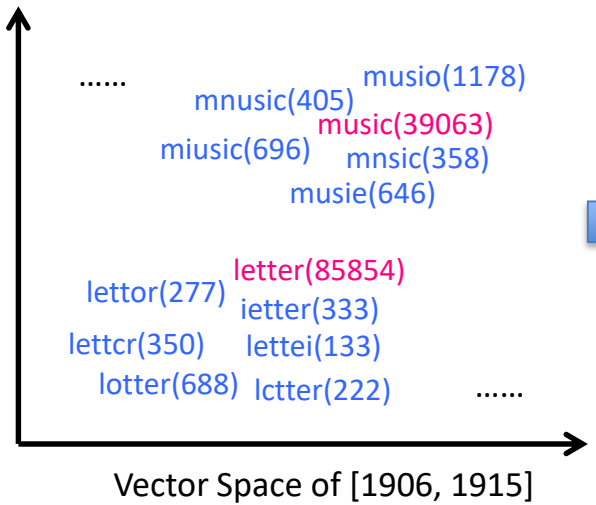
ID	q [2002,2007]	t [1987,1991]
1	Putin	Yeltsin
2	Chirac	Mitterrand
3	iPod	Walkman
4	Facebook	Usenet
5	Linux	Unix
6	spam	junk mail, autodialers, junk fax
7	spreadsheet	database, word processor
8	email	messages, letters, mail, fax
9	superman	superman, batman
10	Pixar	Tristar, Disney
11	Euro	Mark, Lira, Franc
12	Myanmar	Burma
13	Koizumi	Kaifu
14	Rogge	Samaranch
15	Serbia, Croatia, Macedonia, Montenegro, Kosovo, Slovenia, Bosnia	Yugoslavia
16	fridge	fridge, freezer, refrigerator, ice_cubes
17	NATO	NATO
18	Google	IBM, Microsoft, Matsushita, Panasonic
19	Boeing	Boeing, Airbus, Mcdonnell Douglas
20	Flash drive, USB, CDROM, DVD	floppy disc
..	...	...

Table 1. Examples of test sets where term *q* is input and term *t* is the expected temporal analog (*t* can be multiple)

Y. Zhang, A. Jatowt, S. Bhowmick and K. Tanaka: The Past is Not a Foreign Country: Detecting Semantically Similar Terms across Time, IEEE TKDE, 2793-2807 (2016)

Example of correcting errors using semantic word similarities

- OCR problem (Optical Character Problem)
  - Build **dictionary** to map wrong spellings to correct ones
    - **Input:** vector representation of all the words
    - **Output:** dictionary {wrong spelling: correct spelling}

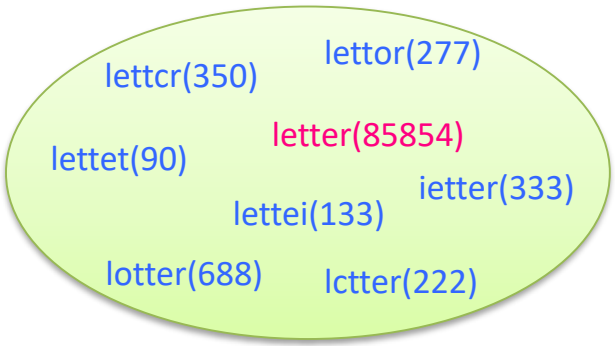


Original Spelling	Correct Form
mnusic	music
miusic	music
musie	music
.....	.....
lettcr	letter
lettor	letter
lotter	letter
.....	.....

Y. Zhang, A. Jatowt, S. Bhowmick and K. Tanaka: The Past is Not a Foreign Country: Detecting Semantically Similar Terms across Time, IEEE TKDE, 2793-2807 (2016)

Example of correcting errors using semantic word similarities

- Assumptions for Improving OCR Problem:
  (1) Wrongly spelled term has similar meaning with its correctly spelled term;
  (2) The correct term is more dominant (or frequent) compared to its wrongly spelled ones;
  (3) Wrongly spelled term has one edit-distance from its correct term.



- Example results of temporal analog finding:
  - Without Error Correction:
    - car [2004,2009] → [1906,1015] vehicle, tricycle, mnotor, rmotor, car, eycles
  - With Error Correction:
    - car [2004,2009] → [1906,1915] vehicle, tricycle, motor, car, cycles

Y. Zhang, A. Jatowt, S. Bhowmick and K. Tanaka: The Past is Not a Foreign Country: Detecting Semantically Similar Terms across Time, IEEE TKDE, 2793-2807 (2016)

Classification

The Task of Text Classification

Classification

- Classification lies at the heart of both human and machine intelligence
- Deciding what letter, word, or image has been presented to our senses, recognizing faces or voices, sorting mail, assigning grades to homeworks;
- These are all examples of assigning a category to an input

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!

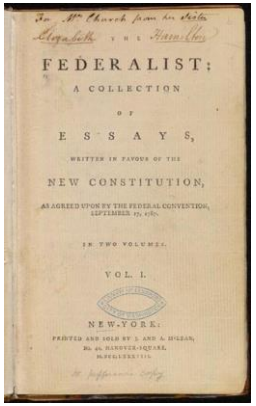
You can now access the latest news by using the link below to login to Stanford University News Forum.

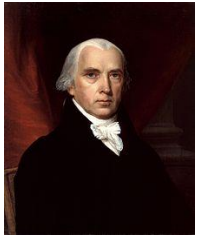
http://www.123contactform.com/contact-form-StanfordNew1-236335.html

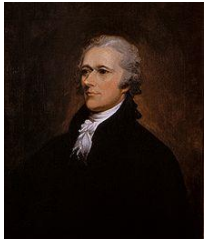
Click on the above link to login for more information about this new exciting forum. You can also copy the above link to your browser bar and login for more information
about the new services.

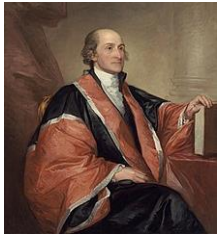
Who wrote which Federalist papers?

- 1787-8: anonymous essays try to convince New York citizens 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


FEDERALIST:
A COLLECTION
OF
ESSAYS,
WRITTEN IN FAVOUR OF THE
NEW CONSTITUTION,
AS AGREED UPON BY THE FEDERAL CONVENTION,
SEPTEMBER 17, 1787.
IN TWO VOLUMES.
VOL. I.
NEW-YORK:


James Madison


Alexander Hamilton


John Jay

What is the subject of this medical article?

Brain and Cognition

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

MEDLINE Article

?

MeSH Subject Category Hierarchy

What is the subject of this medical article?

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

MeSH Subject Category Hierarchy

Positive or negative movie 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...*

Positive or negative movie 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...*

Summary: Text Classification Usage

- Spam detection
- Authorship identification
- Language identification
- Assigning subject categories, topics, or genres
- Sentiment analysis
- …

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 “you have been selected”)
- Accuracy can be high
  - If rules carefully refined by expert
- Building and maintaining these rules is expensive

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
  - k-Nearest Neighbors
  - Neural networks
  - …

Text Classification & Naive Bayes

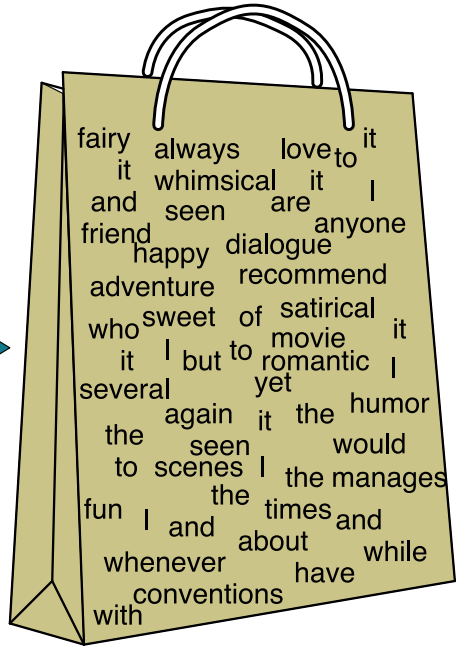
Naive Bayes

Naive Bayes Intuition

- Simple (“naive”) classification method based on Bayes rule
- Relies on very simple representation of document
  - **Bag of words**

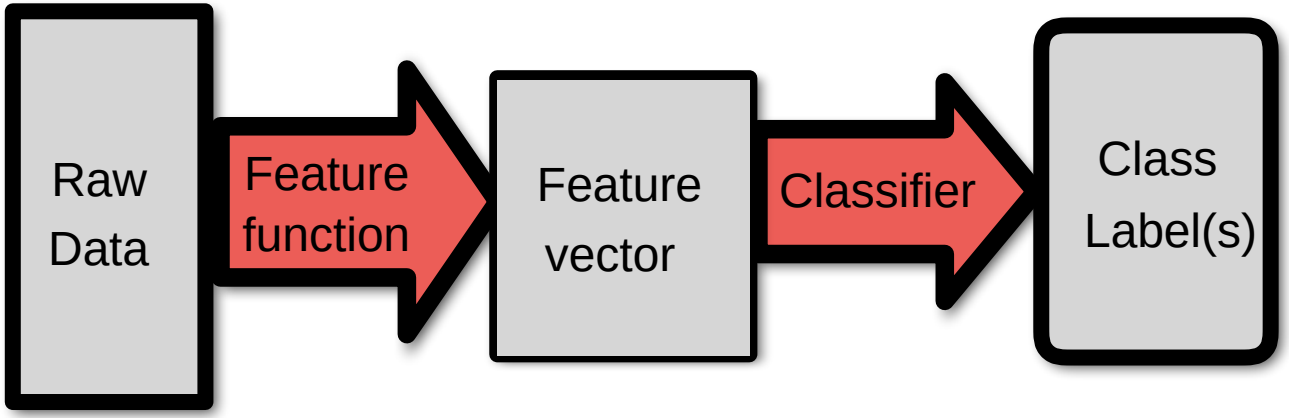
The Bag of Words Representation

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
…	…

Classification: more generally



Before we can use a classifier on data,
we should map the data to **feature vectors**

Feature Engineering

- We assume each input item is represented as a ‘feature’ vector:

  $x = (x_1 \ldots x_N)$
  - Each element in x is one feature
  - The number of elements/features N is fixed, and may be very large
  - x has to capture all the information about the item that the classifier needs
- We need to define a suitable feature function that maps raw data points to vectors
- Feature engineering (designing suitable feature functions) is very important for accurate classification

Bag of words

- Bag-of-Words representation:
  - Assume that each element x_i in (x_1….x_N) corresponds to one word type (v_i) in the vocabulary V = {v_1,…,v_N}
  - Either: $\mathbf{x_i}$**={0,1}:** Does word v_i occur (yes: 1, no: 0) in the input document?
  - Or: $\mathbf{x_i}$**={0,1,2,3..}**: How often does word v_i occur in the input document?

The bag of words representation

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

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

MAP: "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

Naive Bayes Classifier

"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 $x_1..x_n$

Naïve Bayes Classifier

$$c_{MAP} = \underset{c \hat{I} C}{\text{argmax}} P(x_1, x_2, \square \ , x_n \mid c) P(c)$$

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

Naive Bayes Independence Assumptions

- **Bag of Words assumption**: Assume position doesn't matter
- **Conditional Independence assumption**: Assume the feature probabilities $P(x_i|c_j)$ are independent given the class c

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

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 Naive Bayes Classifiers to Text Classification

positions ← all word positions in test document

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

Multiplying lots of probs

$$c_{NB} = \underset{c_j \hat{I}\ C}{\text{argmax}}\, P(c_j) \tilde{O}_{i \hat{I}\ positions} P(x_i \mid c_j)$$

- Multiplying lots of probabilities can result in floating-point underflow
- We sum logs of probabilities instead of multiplying probabilities
  log(*ab*) = log(*a*) + log(*b*)

Doing everything in log space

Instead of:

$$c_{NB} = \underset{c_j \hat{I}\ C}{\text{argmax}}\, P(c_j) \tilde{O}_{i \hat{I}\ positions} P(x_i \mid c_j)$$

This:

$$c_{\text{NB}} = \underset{c_j \in C}{\text{argmax}} \left[\log P(c_j) + \sum_{i \in \text{positions}} \log P(x_i | c_j) \right]$$

Model is now maximum over sum of weights: a **linear** function of the inputs

So naive bayes is a **linear classifier**

Text Classification and Naïve Bayes

Naive Bayes: Learning

Learning the Naive 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

- Create mega-document for class *j* by concatenating all docs in this class
  - Use frequency of *w* in mega-document

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

fraction of times word w_i appears
among all words in documents of class c_j

Problem with Maximum Likelihood

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

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

- 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

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

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

Unknown words

- The problem of unknown words
  - that appear in test data but not in training data or vocab
- We can **ignore** them
  - Remove them from the test document pretending they weren't there
  - Don't include any probability for them at all
- Or we can build an unknown word model
  - But it doesn't help: knowing which class has more unknown words is not generally a useful thing to know..

Stop words

- Some systems ignore another class of words
- **Stop words:** very frequent words like *the* and *a*
  - Sort the whole vocabulary by frequency in the training, call the top 10 or 50 words the **stopword list**
  - Remove all stop words from the training and test sets as if they were never there
- But in most text classification applications, removing stop words don't help, so it's more common to use all the words in Naive Bayes

Naïve Bayes: Learning

- Calculate $P(c_j)$ terms
  - For each c_j in C do

    $docs_j \leftarrow$ all docs of class $= c_j$

$$P(c_j) \neg \frac{|docs_j|}{|\text{total \# documents}|}$$

- Extract *Vocabulary* from training set
- Calculate $P(w_k \mid c_j)$ terms
  - $Text_j \leftarrow$ single doc containing all $docs_j$
  - For each word w_k in *Vocabulary*
    - $n_k \leftarrow$ # occurrences of w_k in $Text_j$

$$P(w_k \mid c_j) \neg \frac{n_k + a}{n + a\,|Vocabulary|}$$

Naïve Bayes in Spam Filtering

- SpamAssassin Features:
  - Mentions Generic Viagra
  - Online Pharmacy
  - Mentions millions of (dollar) ((dollar) NN,NNN,NNN.NN)
  - Phrase: impress others
  - 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

https://spamassassin.apache.org/

Naïve Bayes in Language Identification

- 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 (world English, etc.)

Naive Bayes Not So Naive

- Very fast, low storage requirements
- Works 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 in case of little data
- A good dependable baseline for text classification

Text Classification and Naïve Bayes

Naïve Bayes: Relationship to Language Modeling

Naïve Bayes and Language Modeling

- Naïve bayes classifiers can use any sort of features
  - 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 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	.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

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

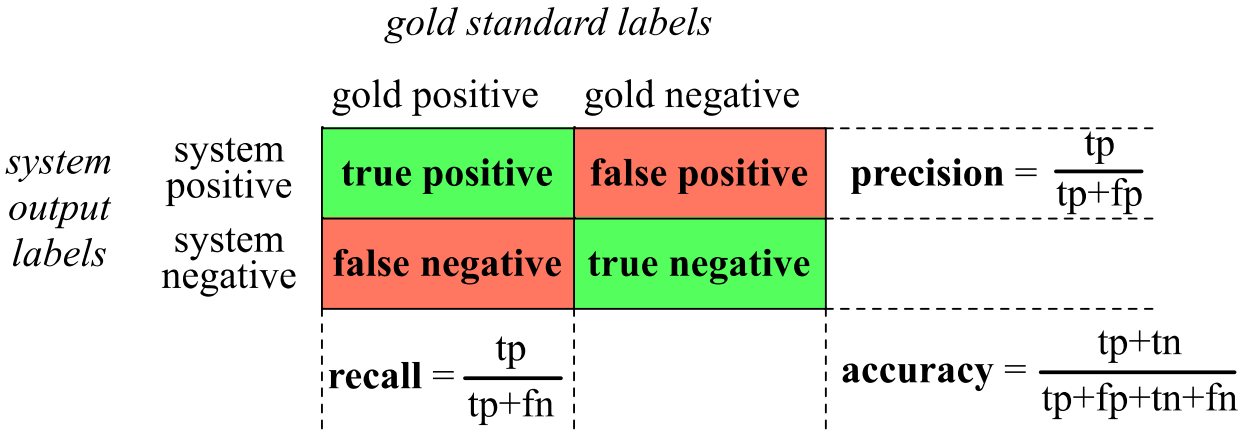
Text Classification and Naïve Bayes

Precision, Recall, and F measure

Evaluation

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

The 2-by-2 confusion matrix



Evaluation: Precision

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

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

Evaluation: Recall

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

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

Why precision and recall?

- Our dumb pie-classifier
  - Just labels nothing as "about pie"

Accuracy=99.99%

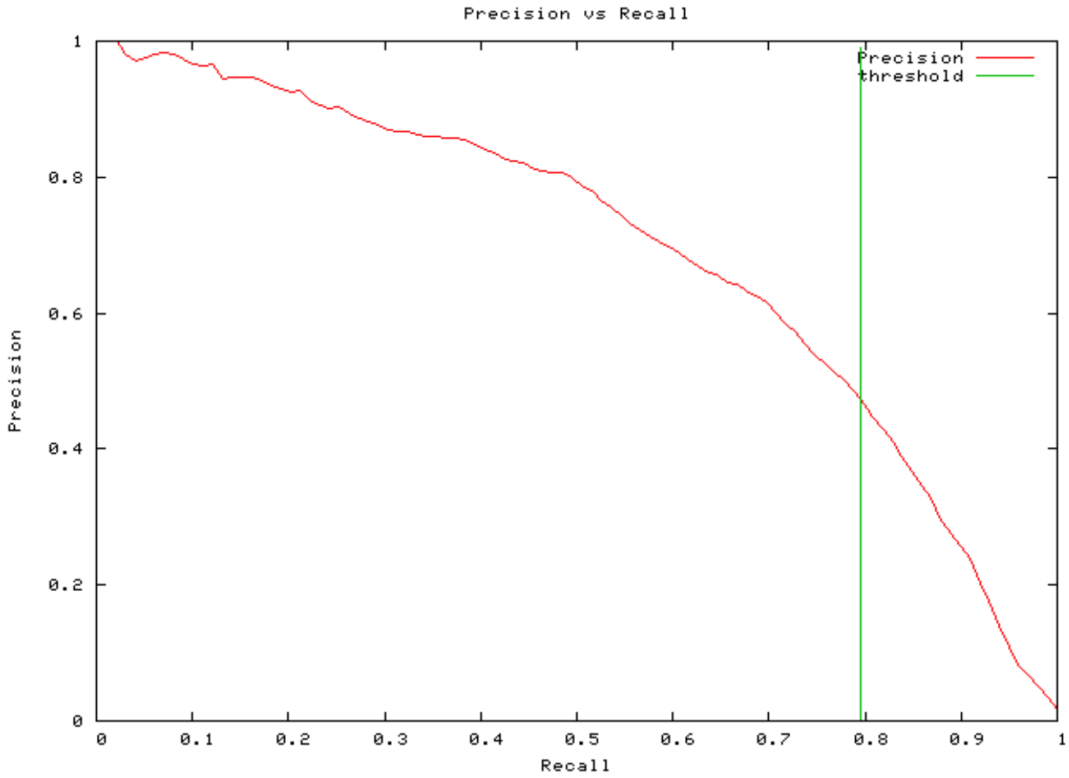
but

Recall = 0

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

Precision and recall, unlike accuracy, emphasize true positives:

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


Precision vs Recall
Precision
threshold
Precision
Recall
1
0.8
0.6
0.4
0.2
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

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., $\beta = 1$)

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

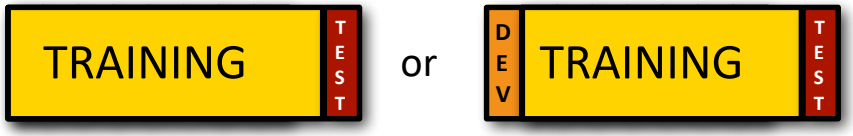
Development Test Sets ("Devsets") and Cross-validation

- Train on training set, tune on devset, report on testset
  - This avoids overfitting (‘tuning to the test set’)
  - More conservative estimate of performance
  - We want as much data as possible for training, and much for dev, as well as much for test, so how to split?

Evaluating Classifiers

Evaluation setup:

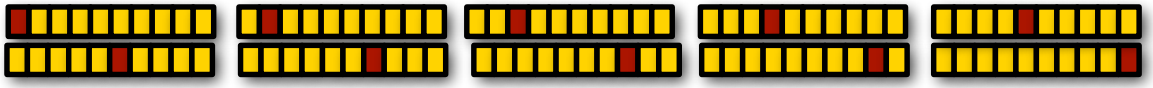
Split data into separate **training,** (**dev**elopment) and **test** sets.



Better setup: **n-fold cross validation**:

Split data into *n* sets of equal size

Run *n* experiments, using set *i* to test and remainder to train



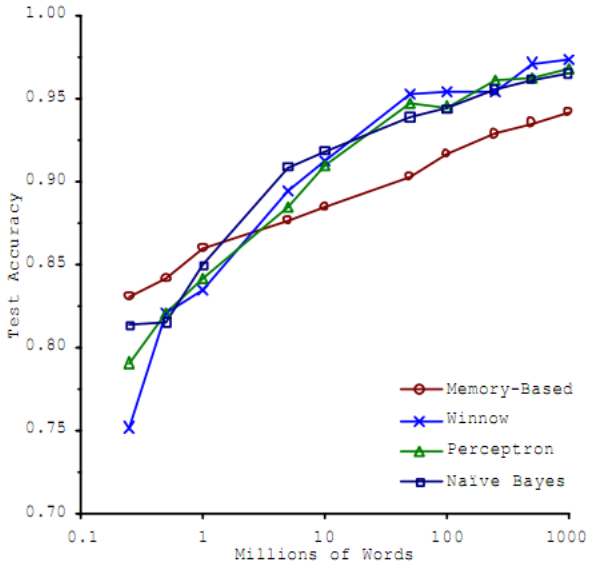
This gives average, maximal and minimal measures

When **comparing two classifiers**:

Use the **same** test and training data with the same classes

Accuracy as a function of data size..

- With enough data
  - Classifier algorithm may not matter..

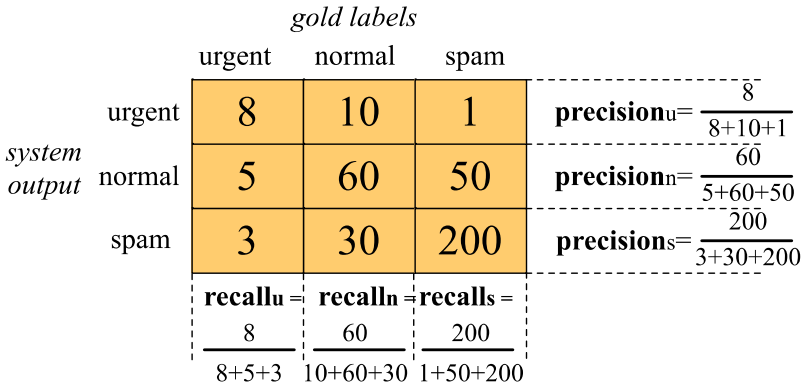


Michele Banko and Eric Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In Proceedings of the 39th Annual Meeting on Association for Computational Linguistics (ACL '01). Association for Computational Linguistics, USA, 26–33.

Text Classification and Naive Bayes

Evaluation with more than two classes

Confusion Matrix for 3-class classification



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

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

Macro-averaging and Micro-averaging Examples

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

$$\text{microaverage precision} = \frac{268}{268+99} = \mathbf{.73}$$

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

Sentiment Analysis

Naive Bayes

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

	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

Prior from training:

P(-) = 3/5

P(+) = 2/5

Dropping "with"

Likelihoods from training:

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

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

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

Scoring the test set:

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

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

Optimizing for sentiment analysis

For tasks like sentiment assessment, word **occurrence** is more important than word **frequency**

- The occurrence of the word *fantastic* tells us a lot
- The fact that it occurs 5 times may not tell us much more..

Binary naive bayes, or **binary NB**

- Clip the word counts at 1

Binary Naive Bayes on a test document *d*

- First remove all duplicate words from *d*
- Then compute NB using the same equation:

$$c_{NB} = \underset{c_j \hat{\imath} C}{\text{argmax}} P(c_j) \tilde{\text{O}}_{i \hat{\imath} \ positions} P(w_i \mid c_j)$$

Binary 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 because binarization is within documents