

Text Classification

CS4248 Natural Language Processing

Week 04

Rahul BAID, Xinyuan LU and Min-Yen KAN

4

Slides adapted from An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, Prof. Hwee Tou Ng (NUS), and Dan Jurafsky (Stanford)

Recap of Week 03

Language Models

N-grams

Perplexity: Evaluating Language Models

Unknown Words Redux

- Smoothing

- Backoff

- Interpolation

- Kneser-Ney

Announcements

Assignment #1 Due

Assignment #2 Out

Project Team Formation Announced – Due Next Week

Midterm to be offered both online and offline

Week 04 Agenda

Text Classification

Case Study: Sentiment Analysis

TF-IDF

Vector Space Model

Naïve Bayes
and a Runthrough (time permitting)

Evaluating Text Classification

Text Classification

What is Text Classification?

Text Classification

A mapping h from input data x (drawn from instance space \mathcal{X}) to a label (or labels) y from some enumerable output space \mathcal{Y}

- \mathcal{X} = set of all documents
- $\mathcal{Y} = \{\text{english, mandarin, greek, ...}\}$
- x = a single document

$$h(x) = y$$

E.g. $Lh(\mu\eta\nu\iota\nu \alpha\epsilon\iota\delta\epsilon \theta\epsilon\alpha) = \text{greek}$

Slide adapted from David Bamman (UCB)

Text Classification

Let $h(x)$ be the “true” mapping (**unknown**).

How do we find the best $\hat{h}(x)$ to approximate $h(x)$?

- Rule Based (Decision Rules)

if x has characters in
unicode point range 0370-03FF:

$\hat{h}(x) = \text{greek}$

- Supervised learning
 - Given training data in the form of $\langle x, y \rangle$ pairs, learn $\hat{h}(x)$

Slide adapted from David Bamman (UCB)

Subject Classification



MeSH Subject Category Hierarchy

Antagonists and Inhibitors

Blood Supply

Chemistry

Drug Therapy















Embryology

Epidemiology

...

Slide adapted from Dan Jaraafsky (Stanford)

Spam Detection

	Inbox	
	Starred	
	Chats	
	Sent	
	Drafts	4
	All Mail	
	Spam	128
	Trash	
	Categories	
	Social	26
	Updates	809
	Forums	705
	Promotions	445
	Mail Merge	

REPLY 01-02-2021

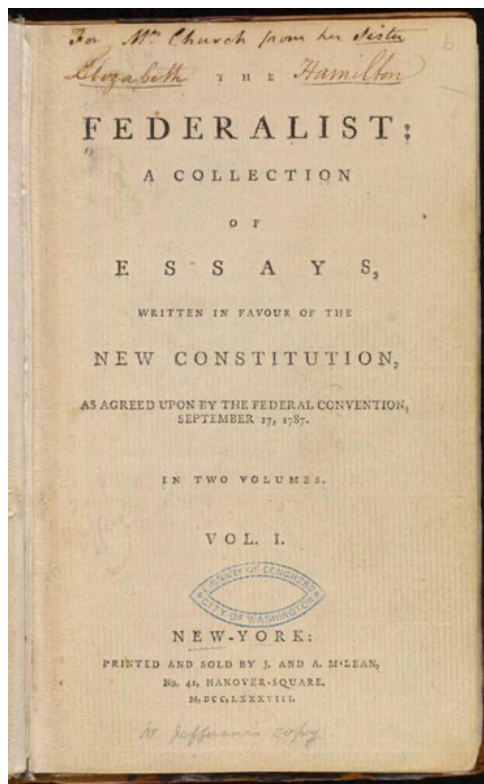
I'm Ivan Maslyae, the Vice President General Counsel and member of LUKOIL Management Committee. I have a file that has something to do with a member of your family. Get back for more details

⏪ Reply all

⏪ Reply

➡ Forward

Authorship Attribution



Who wrote which Federalist papers?

1787-8: anonymous essays try to convince New York to ratify U.S Constitution: Jay, Madison, Hamilton.

Authorship of 12 of the letters was in dispute.

1963: solved by Mosteller and Wallace using Bayesian methods

Slide adapted from Dan Jurafsky (Stanford)

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...

Slide adapted from Dan Jurafsky (Stanford)

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...

Slide adapted from Dan Jurafsky (Stanford)

The Text Classification Problem

Given a document $\mathbf{x} = (w_1, w_2, \dots, w_{|x|}) \in \mathcal{V}^*$

Handwritten notes:
 - Arrow from \mathbf{x} to "vector of different values"
 - Arrow from \mathcal{V}^* to "vocabulary"

predict a label $y \in \mathcal{Y}$

Handwritten note: "positive ? negative?"

where \mathcal{Y} is an enumerated, fixed set of classes.

N.B.: Our SLP3 textbook uses d for x and c for y . We'll use both interchangeably.

Slide adapted from Dan Jurafsky (Stanford)



Sample Text Classification Tasks

task	x	y
language ID	text	{english, mandarin, greek, ...}
spam classification	email	{spam, not spam}
authorship attribution	text	{jk rowling, james joyce, ...}
genre classification	novel	{detective, romance, gothic, ...}
sentiment analysis	text	{positive, negative, neutral, mixed}

and many more...

Slide adapted from Dan Jurafsky (Stanford)

Case Study: Sentiment Analysis

Sample Text Classification Problem

Adapted from David Bamman (UCB)

Sentiment Analysis

Product ratings

★★★★★ Five Stars

I needed the book for my natural language processing class. needless to say, I learnt a lot.
Published on November 27, 2014 by Kamran

★★★★☆ Encyclopedic Treatment of NLP

Daniel Jurafsky and James Martin have assembled an Incredible mass of Information about natural language processing. Foundations of Statistical Natural Language Processing [Read more](#)
Published on April 25, 2012 by John M. Ford

Political opinion mining


Steve Hofstetter @SteveHofstetter · 3h

BREAKING: According to the Washington Post, **Donald Trump** has instructed his staff not to pay Rudy Giuliani's legal fees.

I'll write more when I'm done laughing.

899
3.8K
20.4K


Dr. Jack Brown @DrGJackBrown · 1h

FACT: **Donald Trump** has seen his last Wednesday Night sunset that he'll ever see from the White House.

64
178
1.6K

Sentiment Analysis

Movie Reviews

Movie: Soul
 Year: 2020

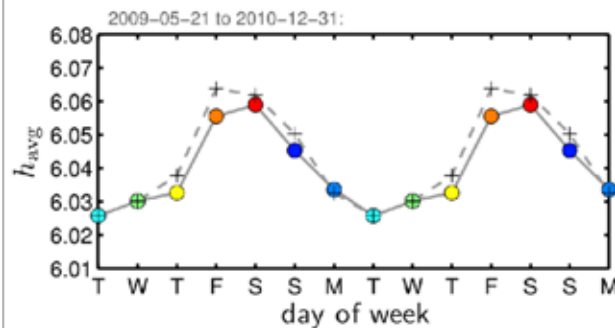
This inventive tale stars Jamie Foxx as a jazz musician caught in a world that human souls pass through on their way into and out of life. [Full review](#)

 A.O. Scott
 The NYTimes

Soul strives to help us remember that life itself is a blessing, even when it doesn't go as we planned. [Full review](#)

Paul Asay
 Plugged In

Sentiment as Tone



Dodds et al. (2011), "Temporal patterns of happiness and information in a global social network: Hedonomics and Twitter" (PLoS One)

Sentiment Analysis

The sentiment expressed in a text refers to the author's subjective or emotional attitude towards the central topic of the text.

Sentiment analysis is a classic application of text classification, and is typically approached with a BoW classifier.



Sentiment Analysis

Some linguistic phenomena require going beyond the bag-of-words:

This is not the worst thing that can happen.

I didn't like this movie.

How should we deal with **negation**?

- prepend the prefix **NOT** to every word after a token of logical negation until the next punctuation mark.

"I didn't NOT_like NOT_this NOT_movie."

Sentiment Analysis

Sometimes words are a good indicator of sentiment
(*love, amazing, hate, terrible*); but ...

Many times it requires deep world and contextual knowledge:

- *It would be nice if you acted like you understood.*
- *Valentine's Day is being marketed as a Date Movie. I think it's more of a First-Date Movie. If your date likes it, do not date that person again. And if you like it, there may not be a second date. - Robert Ebert*

Adapted from David Bamman (UCB)

Sentiment Analysis

Lack of training data?

- Use **external dictionaries**:
E.g.: General Inquirer, MPQA, LIWC, AFINN, etc.

pos	neg
unlimited	lag
prudent	contortions
superb	fright
closeness	lonely
impeccably	tenuously
fast-paced	plebeian
treat	mortification

Adapted from David Bamman (UCB)

$tf-idf$

How does a word contribute meaning to a document?

Adapted from CS3245

Word–Document Count Matrices

Store the number of occurrences of a term in a document:

- Each document is a **count vector** in \mathcal{N}^V : a column below

documents

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

Adapted from CS3245

$$tf_d(w) = d_count(w)$$

Term frequency tf

- The term frequency $tf_{w,d}$ of word w in document d is defined as the number of times that w occurs in d .
- We want to use tf when computing a representation of a document. But how?
- Raw term frequency is not what we want:
 - Relevance does not increase proportionally with term frequency.
 - A document with 10 occurrences of the term is more relevant than a document with 1 occurrence. But not 10 times more relevant.

Note: frequency = count

Adapted from CS3245

Solution: log frequency weighting

The log frequency weight of word w in d is

$$w_{t,d} = \begin{cases} 1 + \log_{10} \text{tf}_{t,d}, & \text{if } \text{tf}_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

e.g. $0 \rightarrow 0$, $1 \rightarrow 1$, $2 \rightarrow 1.3$, $10 \rightarrow 2$, $1000 \rightarrow 4$, etc. \dagger .

Adapted from CS3245

Document frequency

Rare terms are more informative than frequent terms

- Recall stop words
- Consider a term in the query that is rare in the collection (e.g., *arachnocentric*)

A document containing this term is very likely to be relevant to the query *arachnocentric*

- We want a high weight for rare terms like *arachnocentric*.

Document frequency, continued

Frequent words are less informative than **rare words**

Consider a word that is frequent in the collection (e.g., *high*, *increase*, *line*). How important is that word in characterizing it (\mathcal{R})?

Document frequency, continued

Frequent words are less informative than **rare words**

Consider a word that is frequent in the collection (e.g., *high*, *increase*, *line*). How important is that word in characterizing it (\mathcal{R})?

A document containing such a word is more important to characterize it a document that doesn't have it.

For such words , we want high positive weights for them **but lower than the weights for rare words.**

We will use document frequency (df) to capture this.

idf weight

the more often it appears, less discriminative.

df_w is the document frequency of w : the number of documents that contain w

- df_w is an inverse measure of the informativeness of w
 - $df_w \leq N$
- total number of documents in corpus*

We define the idf (**inverse document frequency**) of w by

$$idf_w = \log (N/df_w)$$

- We use $\log(N/df_w)$ instead of N/df_w to “dampen” the effect of idf.

Adapted from CS3245

Example: suppose $N = 1$ million

term	df_t	idf_t
calpurnia	1	6
animal	100	4
sunday	1,000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

$$idf_w = \log_{10}(N/df_w)$$

There is one idf value for each term w in a collection.

Adapted from CS3245

Collection vs. Document frequency

The collection frequency of w is the number of occurrences of w in the collection, counting multiple occurrences.

Example:

happens in specific context. →

+ a lot in one document. ↑

+1 per document. ↑

Word	Collection frequency	Document frequency
insurance	10440	3997
try	10422	8760

Which word is a better term to characterize a document (and should get a higher weight)?

tf-idf weighting

The *tf-idf* weight of a term is the product of its *tf* weight and its *idf* weight.

$$weight_{w,d} = (1 + \log tf_{w,d}) \times \log(N/df_w)$$

Best known weighting scheme NLP/IR

- Note: the “-” in *tf-idf* is a hyphen, not a minus sign!
- Alternative names: $tf \cdot idf$, $tf \times idf$

Increases with the number of occurrences within a document

Increases with the rarity of the word in the collection

Adapted from CS3245

Count \rightarrow Weight Matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

Each document is now represented by a real-valued **vector** of $tf - idf$ weights $\in \mathcal{R}^{|V|}$

Adapted from CS3245

Vector Space Model

Documents as vectors

So we have a $|V|$ -dimensional vector space

Words are axes of the space

Documents are **points** or **vectors** in this space

High-dimensional: tens of thousands of dimensions;
each dictionary term is a dimension

These are very **sparse** vectors – most entries are zero.

We'll see vector representations for words (vs. sentences) again later in word embeddings

Slide Credits: Dan Jurafsky (Stanford)

want to know two documents belong to the same class.

document v document w .

1st Try: Dot product

The dot product between two vectors is a scalar:

$$\text{dot product}(\mathbf{v}, \mathbf{w}) = \mathbf{v} \cdot \mathbf{w} = \sum_{i=1}^N v_i w_i = v_1 w_1 + v_2 w_2 + \dots + v_N w_N$$

for each dimension (works)

The dot product tends to be high when the two vectors have large values in the same dimensions

Dot product can be a similarity metric between vectors

Nice!

Problem with raw dot product

Dot product favors long vectors.

Dot product is higher if a vector is longer (has higher values in many dimension).

Vector length:

$$|\mathbf{v}| = \sqrt{\sum_{i=1}^N v_i^2}$$

Frequent words (*of*, *the*, *you*) have long vectors (since they occur many times with other words).

So the dot product overly favors frequent words. **Not good.**

Better: cosine for computing word similarity

if angle = 0, cosine = max value = 1.

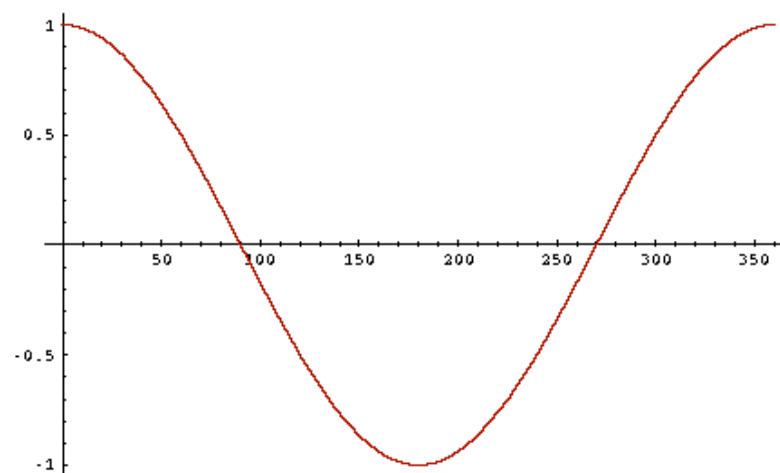
$$\text{cosine}(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}| |\vec{w}|} = \frac{\sum_{i=1}^N v_i w_i}{\sqrt{\sum_{i=1}^N v_i^2} \sqrt{\sum_{i=1}^N w_i^2}}$$

Cosine as a similarity metric

−1: vectors point in opposite directions

+1: vectors point in same directions

0: vectors are orthogonal



But since raw frequency values are non-negative, the cosine for term-term matrix vectors ranges from 0–1

Slide Credits: Dan Jurafsky (Stanford)

Cosine examples

$$\cos(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}| |\vec{w}|} = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}| |\vec{w}|} = \frac{\sum_{i=1}^N v_i w_i}{\sqrt{\sum_{i=1}^N v_i^2} \sqrt{\sum_{i=1}^N w_i^2}}$$

$$\cos(\text{cherry}, \text{information}) =$$

$$\frac{442 * 5 + 8 * 3982 + 2 * 3325}{\sqrt{442^2 + 8^2 + 2^2} \sqrt{5^2 + 3982^2 + 3325^2}} = .017$$

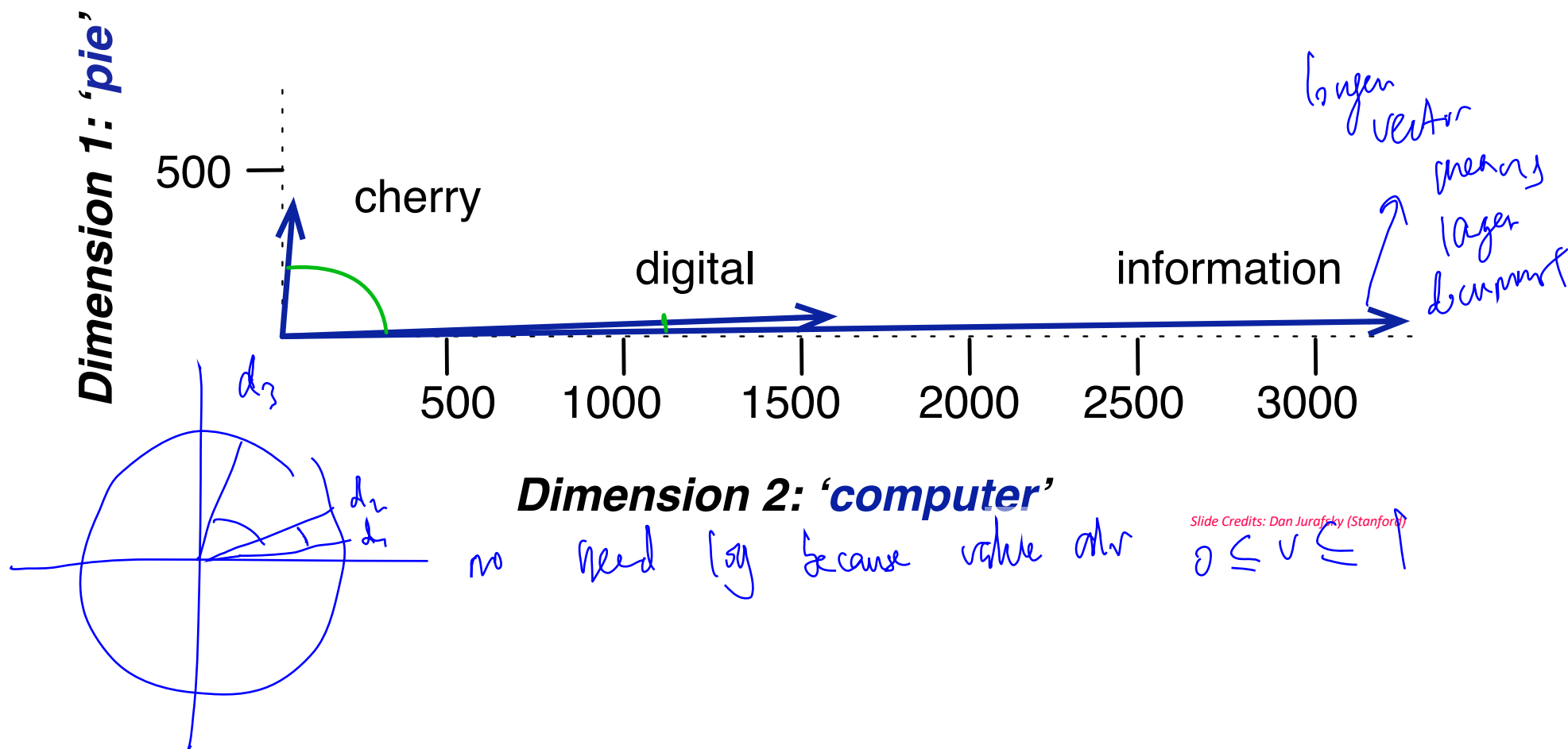
$$\cos(\text{digital}, \text{information}) =$$

$$\frac{5 * 5 + 1683 * 3982 + 1670 * 3325}{\sqrt{5^2 + 1683^2 + 1670^2} \sqrt{5^2 + 3982^2 + 3325^2}} = .996$$

document that has these words.

	pie	data	computer
cherry	442	8	2
digital	5	1683	1670
information	5	3982	3325

Visualizing cosines (well, angles)



Naïve Bayes

Back to Reverend Thomas Bayes for Text Classification

Slide Credits: CS3245 IR and Dan Jurafsky (Stanford)

Naïve Bayes Intuition

Simple (“naïve”) classification method based on Bayes rule.

Relies on a very simple representation of document:

The **Bag of Words** (BoW)

Completely compatible with
the vector space model!

Slide adapted from CS324 IR and Dan Jurafsky (Stanford)

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
...	...

Slide adapted from CS324 IR and Dan Jurafsky (Stanford)

Bayes' Rule

(bold) multiple words.

$$P(y|\mathbf{w}) = \frac{P(\mathbf{w}|y)P(y)}{P(\mathbf{w})}$$

Bayes' Rule *if I knew document w is positive document, how likely*

$$P(y|w) = \frac{P(w|y)P(y)}{P(w)}$$

common words?
rare words?

Bayes' Rule

Likelihood: How probable is the data given that our document is a member of y ?

Prior: How probable is a document to be a member of class y seeing any data?

$$P(y|\mathbf{w}) = \frac{P(\mathbf{w}|y)P(y)}{P(\mathbf{w})}$$

Posterior: How probable is the instance classified as a member of class y ?

Marginal: How probable is the evidence under any class?

Slide adapted from CS3245I Machine Learning

Naïve Bayes

Likelihood: How probable is the data given that our document is a member of y ?

Prior: How probable is a document to be a member of class y seeing any data?

$$P(y|\mathbf{w}) = P(w_1|y) \times P(w_2|y) \times \cdots \times P(w_n|y) \times P(y)$$

Posterior: How probable is the instance classified as a member of class y ?

Marginal: To think about.
Where did it go?

Slide adapted from CS3245I Machine Learning

Naïve Bayes: Self-Check

$$P(y|\mathbf{w}) = P(\mathbf{w}|y) \times P(y)$$

$$P(y|\mathbf{w}) \propto P(w_1|y) \times P(w_2|y) \times \cdots \times P(w_n|y) \times P(y)$$

What two assumptions did we make?

- 1.
- 2.

Naïve Bayes: Self-Check

$$P(y|\mathbf{w}) = P(\mathbf{w}|y) \times P(y)$$

$$P(y|\mathbf{w}) \propto P(w_1|y) \times P(w_2|y) \times \cdots \times P(w_n|y) \times P(y)$$

What two assumptions did we make?

1. Bag of Words: Position doesn't matter
2. Conditional Independence: no interaction between words. i.e., all $P(w_i|y)$ don't give any information

Slide adapted from CS3245I Machine Learning

given a document, want to find which class is most likely.

document is given always, definitely constant!

$$c_{MAP} = \operatorname{argmax}_{c \in \mathcal{C}} P(c|d)$$

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

$$= \operatorname{argmax}_{c \in \mathcal{C}} P(d|c)P(c)$$

$$= \operatorname{argmax}_{c \in \mathcal{C}} \underbrace{P(f_1, f_2, \dots, f_n|c)}_{\text{likelihood}} \underbrace{P(c)}_{\text{prior}}$$

$$= \operatorname{argmax}_{c \in \mathcal{C}} P(f_1|c)P(f_2|c)\dots P(f_n|c)P(c)$$

Independence Assumption

$$c_{NB} = \operatorname{argmax}_{c \in \mathcal{C}} P(c) \prod_{f \in \mathcal{F}} P(f|c)$$

Equation for NB classifier

how probable is this class?

Naïve Bayes Classifier

Maximum a posteriori or mostly likely class

Bayes rule

Dropping the $P(d)$ in the denominator

Document d represented as features f_1, \dots, f_n (such as word counts) BoW assumption

use argmax, all denominator same

Maximum Likelihood Estimates

In relative frequency estimation, the parameters are set to empirical frequencies. Estimating $P(w|c)$:

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

Fraction of times word w_i appears among all words of in all documents of topic c_j

- concatenate all documents into one big class c text. use frequency of w_i in this document.

Estimating $P(c)$:

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

N_{c_j} = the number of documents in our dataset with class c_j

N_{doc} = the total number of documents

Slide Credits: David Bamman (UCB)

Other Issues We've Seen Before

1. **Sparsity**: Multiplying small probabilities leads to numerical underflow.

Solution: transform to the equivalent addition of log probabilities.

$$c_{NB} = \operatorname{argmax}_{c \in \mathcal{C}} \log P(c) + \sum_{f \in \mathcal{F}} \log P(f|c)$$

2. **Out of Vocabulary**: There will be new words seen during test time.

Solution: Smoothing, Backoff and Interpolation; Subword (BPE) Counting

$$\frac{\text{count}(w_i, c_j) + \alpha}{\sum_{w \in \mathcal{V}} \text{count}(w, c_j) + |V|\alpha}$$

α = smoothing hyperparameter.
Laplace (add-1) smoothing: $\alpha = 1$

Summary: Multinomial NB Classifier

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 $bigdoc[c]$

$\loglikelihood[w, c] \leftarrow \log \frac{count(w, c) + 1}{\sum_{w' \in V} (count(w', c) + 1)}$

return \logprior , \loglikelihood , V

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 $\operatorname{argmax}_c sum[c]$

likelihood

$$P(y|x) = \frac{P(x|y) P(y)}{P(x)}$$

N.B.: Our SLP3 textbook uses d for x and c for y . We'll use both interchangeably.

Naïve Bayes Runthrough

Weather Dataset

	Outlook	Temperature	Humidity	Windy	Play Golf
1	Rainy	Hot	High	False	No
2	Rainy	Hot	High	True	No
3	Overcast	Hot	High	False	Yes
4	Sunny	Mild	High	False	Yes
5	Sunny	Cool	Normal	False	Yes
6	Sunny	Cool	Normal	True	No
7	Overcast	Cool	Normal	True	Yes
8	Rainy	Mild	High	False	No
9	Rainy	Cool	Normal	False	Yes
10	Sunny	Mild	Normal	False	Yes
11	Rainy	Mild	Normal	True	Yes
12	Overcast	Mild	High	True	Yes
13	Overcast	Hot	Normal	False	Yes
14	Sunny	Mild	High	True	No

Slide adapted from CS3245I Machine Learning

C_1 C_2

Frequency Table

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	8/14
	Overcast	4	0	4/14
	Rainy	2	3	5/14
		9/14	5/14	

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Slide adapted from CS3245I Machine Learning

Likelihood Table

C_1 C_2

		Play Golf		
		Yes	No	
Outlook	Sunny	3/9	2/5	5/14
	Overcast	4/9	0/5	4/14
	Rainy	2/9	3/5	5/14
		9/14	5/14	

$$P(\text{Yes} | \text{Sunny}) = \frac{P(\text{Sunny}) P(\text{Yes}) \times P(\text{Yes})}{P(\text{Sunny})} = \frac{\frac{3}{9} \times \frac{2}{5}}{\frac{5}{14}} = \frac{3}{5}$$

Slide adapted from CS3245I Machine Learning

Likelihood Table

$$P(x_i|y) = P(\text{Sunny}|\text{Yes}) = \frac{3}{9}$$

$$P(x_i) = P(\text{Sunny}) = \frac{5}{14}$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3/9	2/5	5/14
	Overcast	4/9	0/5	4/14
	Rainy	2/9	3/5	5/14
		9/14	5/14	

$$P(y) = P(\text{Yes}) = \frac{9}{14}$$

$$P(y|x) = P(\text{Yes}|\text{Sunny}) = \frac{3}{9} \times \frac{9}{14} \div \frac{5}{14} = \frac{3}{5}$$

Slide adapted from CS3245I Machine Learning

All of the likelihood tables

These statistics make up the model representation for NB, our h .

		Play Golf	
		Yes	No
Outlook	Sunny	3/9	2/5
	Overcast	4/9	0/5
	Rainy	2/9	3/5

		Play Golf	
		Yes	No
Temperature	Hot	2/9	2/5
	Mild	4/9	2/5
	Cool	3/9	1/5

		Play Golf	
		Yes	No
Windy	True	3/9	3/5
	False	6/9	2/5

		Play Golf	
		Yes	No
Humidity	High	3/9	4/5
	Normal	6/9	1/5

Slide adapted from CS3245I Machine Learning

Applying Smoothed NB at test time

Outlook	Temperature	Humidity	Windy	Play Golf
Overcast	Cool	High	True	?

$$\begin{aligned}
 P(\text{yes} | X) &= P(\text{Overcast} | \text{yes}) \times P(\text{Cool} | \text{yes}) \times P(\text{High} | \text{yes}) \times P(\text{True} | \text{yes}) \times P(\text{yes}) \\
 P(\text{no} | X) &= P(\text{Overcast} | \text{no}) \times P(\text{Cool} | \text{no}) \times P(\text{High} | \text{no}) \times P(\text{True} | \text{no}) \times P(\text{no})
 \end{aligned}$$

Slide adapted from CS3245 Machine Learning

Applying NB at test time

Outlook	Temperature	Humidity	Windy	Play Golf
Overcast	Cool	High	True	?

$$P(Yes|x) = P(Overcast|Yes) \times P(Cool|Yes) \times P(High|Yes) \times P(True|Yes) \times P(Yes)$$

$$= \frac{4}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14}$$

$$P(No|x) = P(Overcast|No) \times P(Cool|No) \times P(High|No) \times P(True|No) \times P(No)$$

Outlook		Play Golf	
		Yes	No
Outlook	Sunny	3/9	2/5
	Overcast	4/9	0/5
	Rainy	2/9	3/5

Temperature		Play Golf	
		Yes	No
Temperature	Hot	2/9	2/5
	Mild	4/9	2/5
	Cool	3/9	1/5

Humidity		Play Golf	
		Yes	No
Humidity	High	3/9	4/5
	Normal	6/9	1/5

Windy		Play Golf	
		Yes	No
Windy	True	3/9	3/5
	False	6/9	2/5

Slide adapted from CS3245! Machine Learning

$$P(Yes|x) = P(Overcast|Yes) \times P(Cool|Yes) \times P(High|Yes) \times P(True|Yes) \times P(Yes)$$

$$\frac{0.0010582}{0.0010582 + 0.0} = 1.0$$

$$P(No|x) = \frac{0}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} \times \frac{5}{14} = 0.0$$

$$\frac{0.0}{0.0010582 + 0.0} = 0.0$$

Slide adapted from CS3245I Machine Learning

Laplace Smoothed Frequency Table

		Play Golf	
		Yes	No
Outlook	Sunny	$3+1=4$ (4/12)	$2+1=3$ (3/8)
	Overcast	$4+1=5$ (5/12)	$0+1=1$ (1/8)
	Rainy	$2+1=3$ (3/12)	$3+1=4$ (4/8)

Priors
$P(\text{Yes}) = 9+1$ (10/14+2)
$P(\text{No}) = 5+1$ (6/14+2)

Slide adapted from CS3245I Machine Learning

Frequency Table

		Play Golf	
		Yes	No
Outlook	Sunny	3+1=4 (4/12)	2+1=3 (3/8)
	Overcast	4+1=5 (5/12)	0+1=1 (1/8)
	Rainy	2+1=3 (3/12)	3+1=4 (4/8)

Priors
$P(\text{Yes}) = 9+1 (10/16)$
$P(\text{No}) = 5+1 (6/16)$

$$P(\text{Yes}|x) = P(\text{Overcast}|\text{Yes}) \times P(\text{Cool}|\text{Yes}) \times P(\text{High}|\text{Yes}) \times P(\text{True}|\text{Yes}) \times P(\text{Yes})$$

$$P(\text{Yes}|x) = \frac{5}{12} \times \frac{4}{12} \times \frac{4}{11} \times \frac{4}{11} \times \frac{10}{16} = 0.01147$$

$$P(\text{No}|x) = P(\text{Overcast}|\text{No}) \times P(\text{Cool}|\text{No}) \times P(\text{High}|\text{No}) \times P(\text{True}|\text{No}) \times P(\text{No})$$

$$P(\text{No}|x) = \frac{1}{8} \times \frac{2}{8} \times \frac{5}{7} \times \frac{4}{7} \times \frac{6}{16} = 0.00478$$

Slide adapted from CS3245! Machine Learning

Applying NB at test time

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Cool	High	True	?

$$P(Yes|x) = P(Rainy|Yes) \times P(Cool|Yes) \times P(High|Yes) \times P(True|Yes) \times P(Yes)$$

$$= \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14}$$

$$P(No|x) = P(Rainy|No) \times P(Cool|No) \times P(High|No) \times P(True|No) \times P(No)$$

$$= \frac{3}{5} \times \frac{1}{5} \times \frac{3}{5} \times \frac{4}{8} \times \frac{5}{14}$$

Outlook		Play Golf	
		Yes	No
Sunny		3/9	2/5
Overcast		4/9	0/5
Rainy		2/9	3/5

Temperature		Play Golf	
		Yes	No
Hot		2/9	2/5
Mild		4/9	2/5
Cool		3/9	1/5

Windy		Play Golf	
		Yes	No
True		3/9	2/5
False		6/9	2/5

Humidity		Play Golf	
		Yes	No
High		3/9	4/5
Normal		6/9	1/5

Slide adapted from CS3245I Machine Learning

Applying NB at test time

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Cool	High	True	?

$$P(Yes|x) = P(Rainy|Yes) \times P(Cool|Yes) \times P(High|Yes) \times P(True|Yes) \times P(Yes)$$

$$P(Yes|x) = \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14} = 0.00529$$

$$P(No|x) = P(Rainy|No) \times P(Cool|No) \times P(High|No) \times P(True|No) \times P(No)$$

$$P(No|x) = \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} \times \frac{5}{14} = 0.02057$$

		Play Golf	
		Yes	No
Outlook	Sunny	3/9	2/5
	Overcast	4/9	0/5
	Rainy	2/9	3/5

		Play Golf	
		Yes	No
Temperature	Hot	2/9	2/5
	Mild	4/9	2/5
	Cool	3/9	1/5

		Play Golf	
		Yes	No
Windy	True	3/9	3/5
	False	6/9	2/5

		Play Golf	
		Yes	No
Humidity	High	3/9	4/5
	Normal	6/9	1/5

Slide adapted from CS3245I Machine Learning

$$\begin{array}{r}
 \frac{2}{9} \quad \frac{3}{5} \\
 \hline
 \frac{2}{9} \quad \frac{1}{5} \\
 \hline
 \frac{3}{9} \quad \frac{4}{5}
 \end{array}$$

Applying NB at test time

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Cool	High	True	?

$$P(Yes|x) = \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14} = 0.00529$$

$$P(No|x) = \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} \times \frac{5}{14} = 0.02057$$

		Play Golf	
		Yes	No
Outlook	Sunny	3/9	2/5
	Overcast	4/9	0/5
	Rainy	2/9	3/5

		Play Golf	
		Yes	No
Temperature	Hot	2/9	2/5
	Mild	4/9	2/5
	Cool	3/9	1/5

		Play Golf	
		Yes	No
Windy	True	3/9	3/5
	False	6/9	2/5

		Play Golf	
		Yes	No
Humidity	High	3/9	4/5
	Normal	6/9	1/5

Slide adapted from CS3245I Machine Learning

Applying NB at test time

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Cool	High	True	?

$$P(Yes|x) = \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14} = 0.00529$$

$$\frac{0.00529}{0.02057 + 0.00529} = 0.2$$

$$P(No|x) = \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} \times \frac{5}{14} = 0.02057$$

$$\frac{0.02057}{0.02057 + 0.00529} = 0.8$$

		Play Golf	
		Yes	No
Outlook	Sunny	3/9	2/5
	Overcast	4/9	0/5
	Rainy	2/9	3/5

		Play Golf	
		Yes	No
Temperature	Hot	2/9	2/5
	Mild	4/9	2/5
	Cool	3/9	1/5

		Play Golf	
		Yes	No
Windy	True	3/9	3/5
	False	6/9	2/5

		Play Golf	
		Yes	No
Humidity	High	3/9	4/5
	Normal	6/9	1/5

Slide adapted from CS3245I Machine Learning

Applying NB at test time

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Cool	High	True	No

$$P(Yes|x) = \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14} = 0.00529$$

$$\frac{0.00529}{0.02057 + 0.00529} = 0.2$$

$$P(No|x) = \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} \times \frac{5}{14} = 0.02057$$

$$\frac{0.02057}{0.02057 + 0.00529} = 0.8$$

max

		Play Golf	
		Yes	No
Outlook	Sunny	3/9	2/5
	Overcast	4/9	0/5
	Rainy	2/9	3/5

		Play Golf	
		Yes	No
Temperature	Hot	2/9	2/5
	Mild	4/9	2/5
	Cool	3/9	1/5

		Play Golf	
		Yes	No
Windy	True	3/9	3/5
	False	6/9	2/5

		Play Golf	
		Yes	No
Humidity	High	3/9	4/5
	Normal	6/9	1/5

Slide adapted from CS3245I Machine Learning

Naïve Bayes Summary

Naïve Bayes is a LM!

Let's relate to last week's lecture on Language Models, which assigns a probability to a sentence.

$$P(\textit{its, water, is, so, transparent, that})$$

Naïve Bayes then:

Naïve Bayes is a LM!

Let's relate to last week's lecture on Language Models, which assigns a probability to a sentence.

$$P(\textit{its, water, is, so, transparent, that})$$

Naïve Bayes then:

- Makes a non-contextual decision (unigram model)
- Treats each class like a separate language model*

Naïve Bayes Summary

Naïve Bayes is a class-specific language model.

Good baseline: Robust, fast to train, and has and low storage requirements.

NB makes strong assumptions that features are:

- conditionally independent of each other
- order doesn't matter

Evaluating Classifications

Going beyond Accuracy

Accuracy

Our basic metric is accuracy: how often is our classifier right?

$$acc(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^N [y^{(i)} = \hat{y}]$$

if $acc(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^N [1] = 1$,
 very accurate:
 either 1 or 0.

Slides adapted from Diyi Yang (GaTech) and David Bamman (UCB)

Beyond Right and Wrong

↑ system says false, but true.
For any set of labels, there are 2 ways to be wrong:

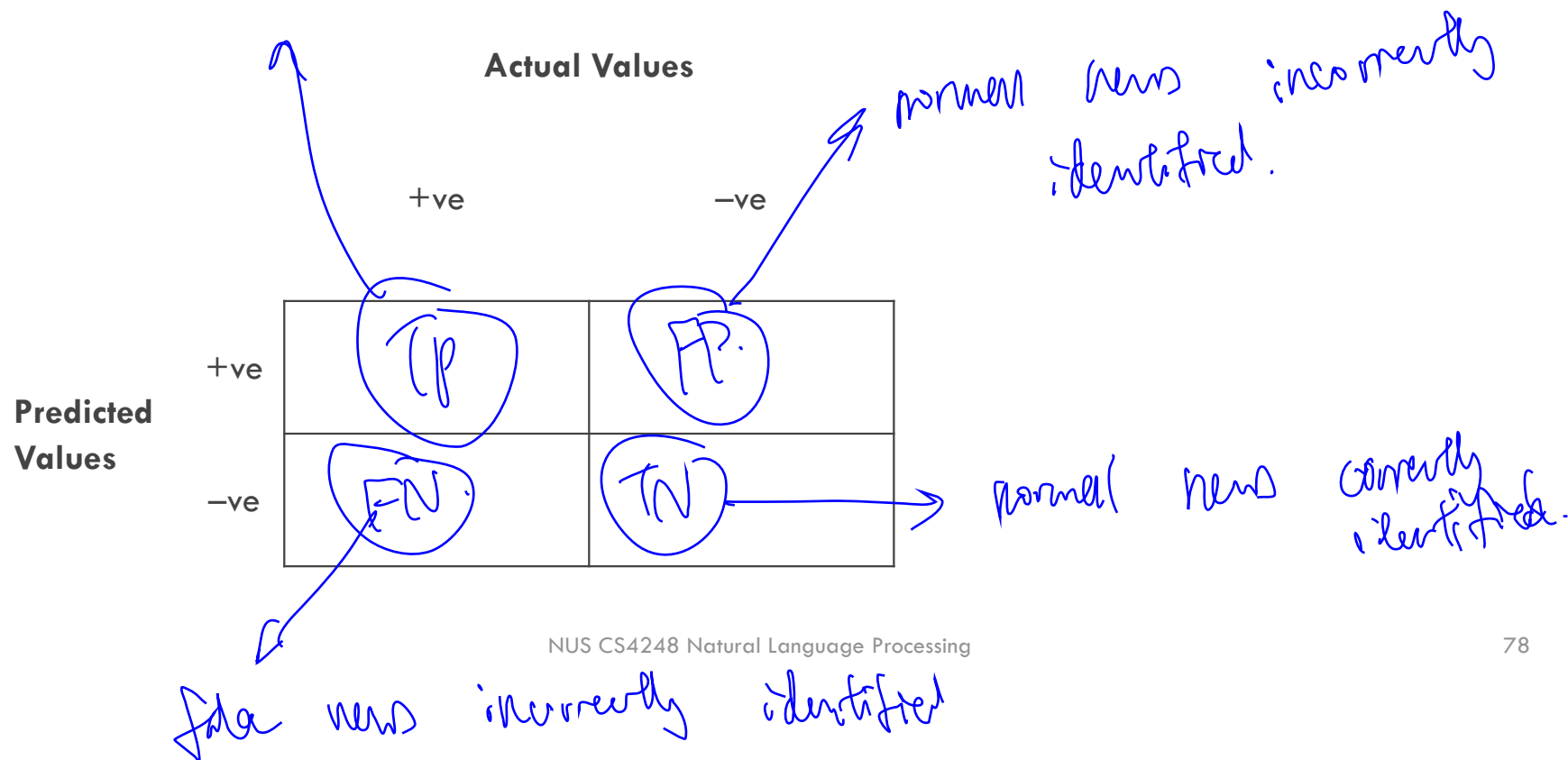
- **False positive (FP)**: the system incorrectly predicts the label
- **False negative (FN)**: the system incorrectly fails to predict the label. → system says true, but false news.

Similarly, there are 2 ways to be right:

- **True positive (TP)**: the system correctly predicts the label. ✓ fake news, indeed fake news
- **True negative (TN)**: the system correctly predicts the label does not apply to it. — not fake, indeed not fake

The Binary Confusion Matrix

False news correctly identified



Recall

(need to cover everything)

Recall is the fraction of positive instances which were correctly classified:

		Actual Values	
		+ve	-ve
Predicted Values	+ve	TP	FP
	-ve	FN	TN

Handwritten blue bracket on the left side of the table, spanning the TP and FN rows, indicating the total number of positive instances.

$$r = \frac{TP}{TP + FN}$$

Question: What is Accuracy?

$$(TP + TN) / (TP + TN + FP + FN)$$

Slides adapted from Diyi Yang (GaTech)

Precision (web search)

Precision is the fraction of **positive** predictions that were correct:

Actual Values			
		+ve	-ve
Predicted Values	+ve	TP	FP
	-ve	FN.	TN

$$p = \frac{TP}{TP + FP}$$

How do you generalize this to k classes?

Slides adapted from Diyi Yang (GaTech)

Precision/Recall

You can get high recall for a class (but low precision) by calling all documents as that class!

high FP, no FN.

Cover
everything

Recall is a non-decreasing function of the number of docs classified to that class.

In most systems, precision decreases when the # of documents assigned to a class increases; i.e., when recall increases

This is not a theorem, but a result with strong empirical confirmation

A combined measure: F

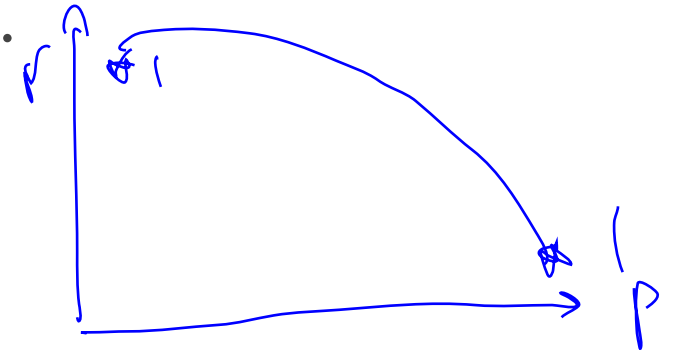
We can combine precision and recall into a single measure to rule them all (actually multiple ways to do this).

$$\frac{2}{\frac{1}{p} + \frac{1}{r}}$$

$$= \frac{2}{\frac{p+r}{pr}}$$

$$= \frac{2pr}{p+r}$$

$$F = \frac{2 \cdot p \cdot r}{p + r}$$



Use a harmonic mean, not an arithmetic mean. **Why?**

any one of them 0, $F=0$.

mean vs mean F_1 ?