



ECS763

Natural Language Processing

Unit 2: Text Classification 1: Probabilistic Models

Lecturer: Julian Hough

School of Electronic Engineering and Computer Science

OUTLINE

- 1) A probabilistic approach to NLP
- 2) Text classification
- 3) Generative classifier: Naïve Bayes
- 4) Evaluating text classifiers

OUTLINE

- 1) A probabilistic approach to NLP
- 2) Text classification
- 3) Generative classifier: Naïve Bayes
- 4) Evaluating text classifiers

Why Probability?



- In building a chatbot, what were the problems with getting reliable ‘intent’ recognition? What were the problems with a dictionary-based approach to sentiment analysis?
- You can define an input->output mapping like a database look-up....
- But what if the user doesn’t say what you manually defined in the input examples?
- It ‘breaks’ with unseen input- we need something more **robust** which accepts things **close** to what is in its database, but not exact string matches and can make a **good guess** of which intent the user’s contribution refers to.

Why Probability?



- e.g. ‘Can you tell me how I enter the uni’s computer system from home?’ **probably means** *#HowToRemoteLogin* even if ‘uni’ isn’t in the dictionary and ‘remote login’ isn’t explicitly mentioned. It’s the most **likely** intent according to its model.

‘Can you tell me how I enter the uni’s computer system from home?’

#HowToRemoteLogin

#HowToPrint

#HowToUseEmail

...



What is a Probability?

- The measure of the **likelihood** that an **event will occur** or that a given **proposition is the case**. A value between 0 (impossible) and 1 (certain). E.g.:
 - The probability that *the earth is flat*: **close to 0**
 - The probability that $1 + 1 = 3$: **0**
 - The probability that *the sun will set today*: **close to 1**
 - The probability that $1 + 1 = 2$: **1**
 - The probability that *6 is thrown in a fair dice*: **one in six, i.e. $\frac{1}{6}$**



What is a Probability?

- Probabilities of an event outcome/proposition are assigned with a **probability function** mapping the outcome to a real number e.g. for some outcome A being the case.

$$p(A) \in [0,1]$$

$$p(A = \textit{True}) \in [0,1]$$

$$p(67\textit{cm} < A \leq 68\textit{cm}) \in [0,1] \quad (\text{continuous probability function})$$

- **Complex outcome** probabilities with more than one event/state of affairs in question can be characterised in terms of **ANDs** and **ORs** over single outcomes:
 - probability that it will rain *and* be warm?
 - probability that it will rain *or* snow?



What is a Probability?

- **AND:** The probability of two 6's being thrown in a fair dice one after each other, i.e. the probability of one independent throw being a 6 and another independent throw being 6, just **multiply** the probabilities:

$$p(6 \text{ thrown}) \times p(6 \text{ thrown}) = \frac{1}{6} \times \frac{1}{6} = \frac{1}{36}$$

(conjunctive probability of independent events)

- **OR:** The probability of either a 6 or a 3 being thrown in a fair dice for a given throw, just **add** the probabilities :

$$p(6 \text{ thrown}) + p(3 \text{ thrown}) = \frac{1}{6} + \frac{1}{6} = \frac{1}{3}$$

(disjunctive probability of mutually exclusive events)



What is a Probability?

- Also what **GIVEN** we know that one event has happened, **THEN** want to know the probability another event has happened, i.e. **conditional probability**.
- e.g. Given I know I have thrown an *even numbered* dice ($\{2,4,6\}$), then what's the probability of me having thrown a 6?
 - Originally when there were 6 possible outcomes $\{1,2,3,4,5,6\}$, the probability was $\frac{1}{6}$
 - Now, given the new condition, this has changed, as there are only 3 possible outcomes $\{2,4,6\}$ we need to be concerned with- so the likelihood changes to $\frac{1}{3}$
 - We narrow the denominator from all events in the event space to a subset of that space given the information we have.



What is a Probability?

- However, we're not just concerned with independent dice throws or mutually exclusive outcomes. For potentially **dependent events**, you can't just multiply for conjunction and add for disjunction- there are **general** rules of **probability calculus (Kolmogorov, 1950)**.

- *A and B* can also be calculated in terms of *A given B* and *B given A*, so we have **the product rule**:

$$p(A \wedge B) = p(A | B) \times p(B) = p(B | A) \times p(A)$$

- For *A or B*, you have to factor out the probability of *A and B*, to avoid 'double counting' that probability mass, so in general we have **the sum rule**:

$$p(A \vee B) = p(A) + p(B) - p(A \wedge B)$$



What is a Probability?

- You can formulate conditional probability (i.e. A is the case, given B is the case) in terms of the probability of A and B being the case over the probability B is the case:

$$p(A | B) = \frac{p(A \wedge B)}{p(B)}$$

- Using the product rule for the numerator, conditional probability can be formulated in terms of **Bayes rule**:

$$p(A | B) = \frac{p(B | A)p(A)}{p(B)}$$

- This is one of the **most important equations in probability theory** (and NLP)
- It allows estimation of $p(A|B)$, using $p(B|A)$, $p(A)$ and $p(B)$ without necessarily having full access to the full joint distribution $p(A,B)$, which is often very large.



What is a Probability?

- **Bayes rule:**

$$p(A | B) = \frac{p(B | A)p(A)}{p(B)}$$



What is a Probability?

- We can think of this in terms of **set theory** where a possible outcome type (**event**) can be seen as a **set**.
- The probability of an outcome, e.g. *throws 6*, is a function of the **cardinality** (size) of the set of all instances with that outcome and the number of *all* events in the **event space** Ω , ensuring any probability is between 0 and 1.

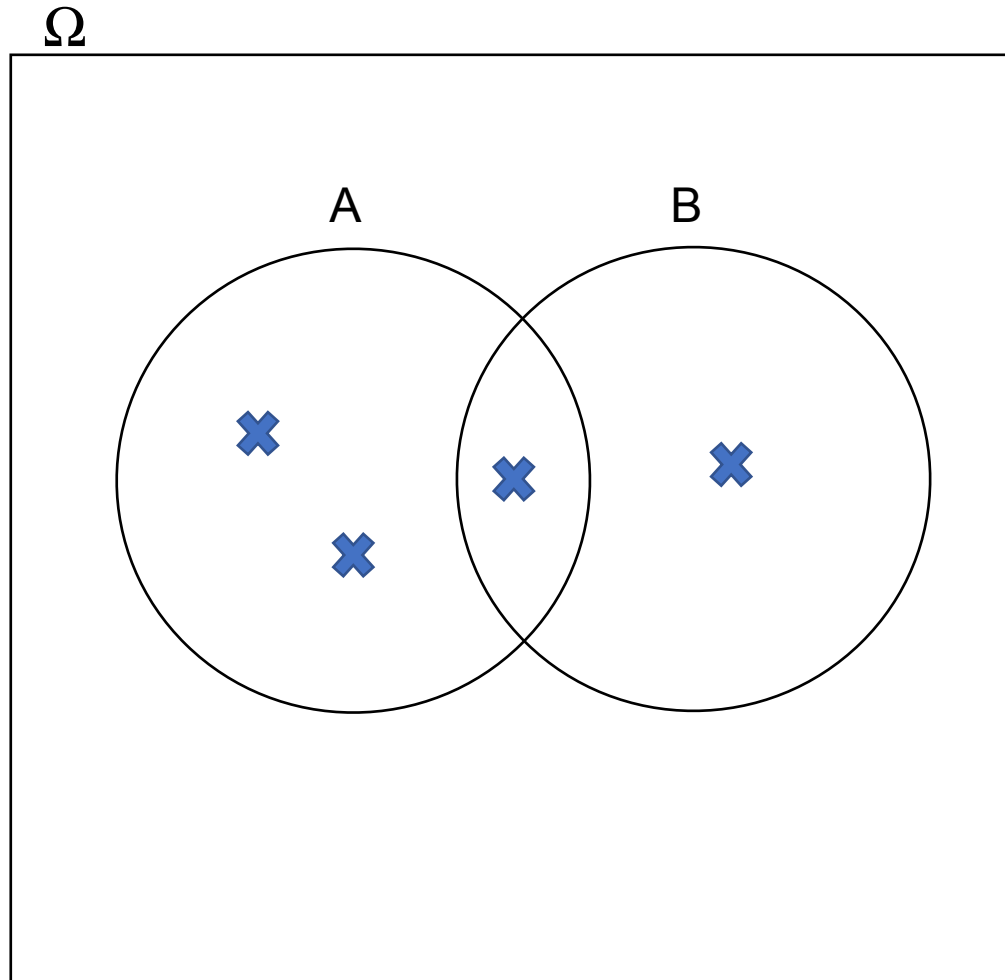
$$p(\text{throws6}) = \frac{|\text{throws6}|}{|\Omega|} \quad i.e. \text{ for any event } A, \quad p(A) = \frac{|A|}{|\Omega|}$$

- We can use the analogues of conjunction (and) and disjunction (or)- set **intersection** \cap and set **union** \cup :

$$p(A \wedge B) = \frac{|A \cap B|}{|\Omega|} \quad p(A \vee B) = \frac{|A \cup B|}{|\Omega|}$$

- And Bayes rule can be formulated as: $p(A|B) = \frac{|A \cap B|}{|B|}$

What is a Probability?



$$p(A) = \frac{|A|}{|\Omega|} = \frac{3}{4}$$

$$p(B) = \frac{|B|}{|\Omega|} = \frac{2}{4}$$

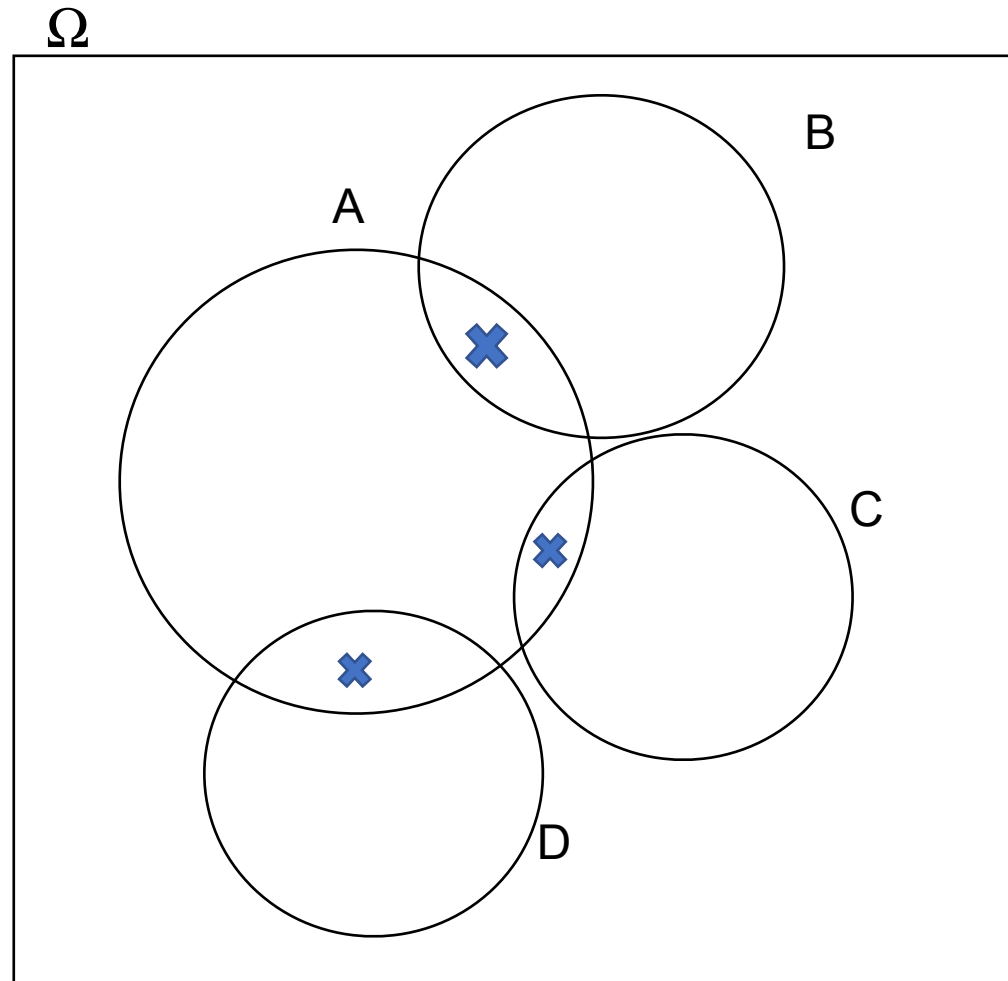
$$p(A \wedge B) = \frac{|A \cap B|}{|\Omega|} = \frac{1}{4}$$

$$p(A \vee B) = \frac{|A \cup B|}{|\Omega|} = \frac{4}{4} = 1$$

$$p(A \mid B) = \frac{|A \cap B|}{|B|} = \frac{1}{2}$$

$$p(B \mid A) = \frac{|A \cap B|}{|A|} = \frac{1}{3}$$

What is a Discrete Probability Distribution?

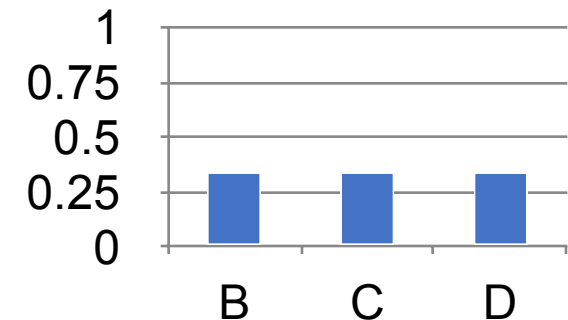


$$p(X = ? \mid A)$$

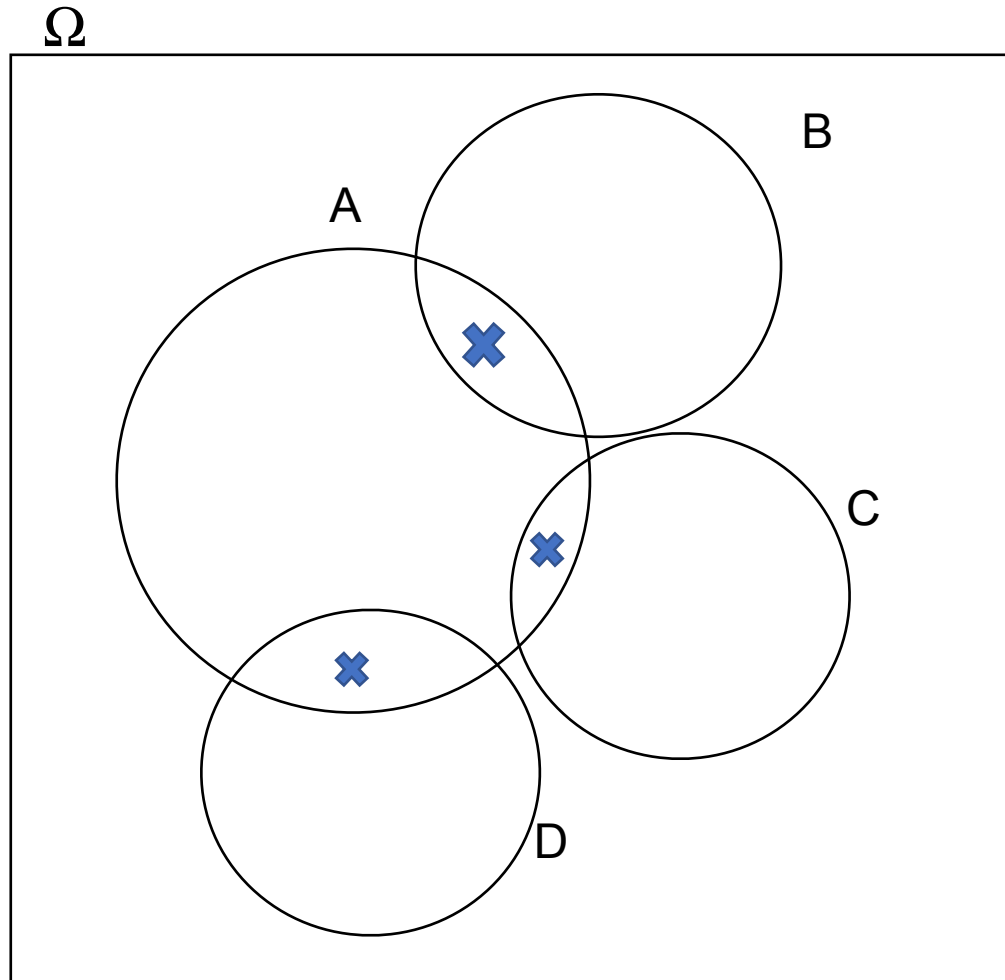
$$p(B \mid A) = \frac{|A \cap B|}{|A|} = \frac{1}{3}$$

$$p(C \mid A) = \frac{|A \cap C|}{|A|} = \frac{1}{3}$$

$$p(D \mid A) = \frac{|A \cap D|}{|A|} = \frac{1}{3}$$



What is a Discrete Probability Distribution?

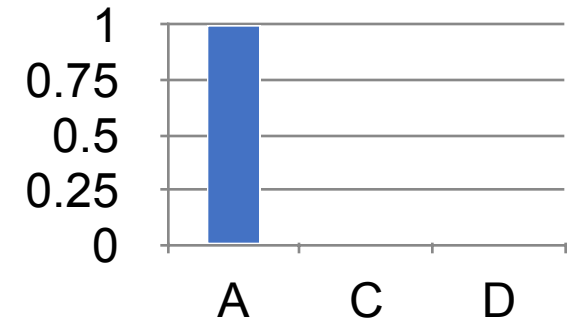


$$p(X = ? | B)$$

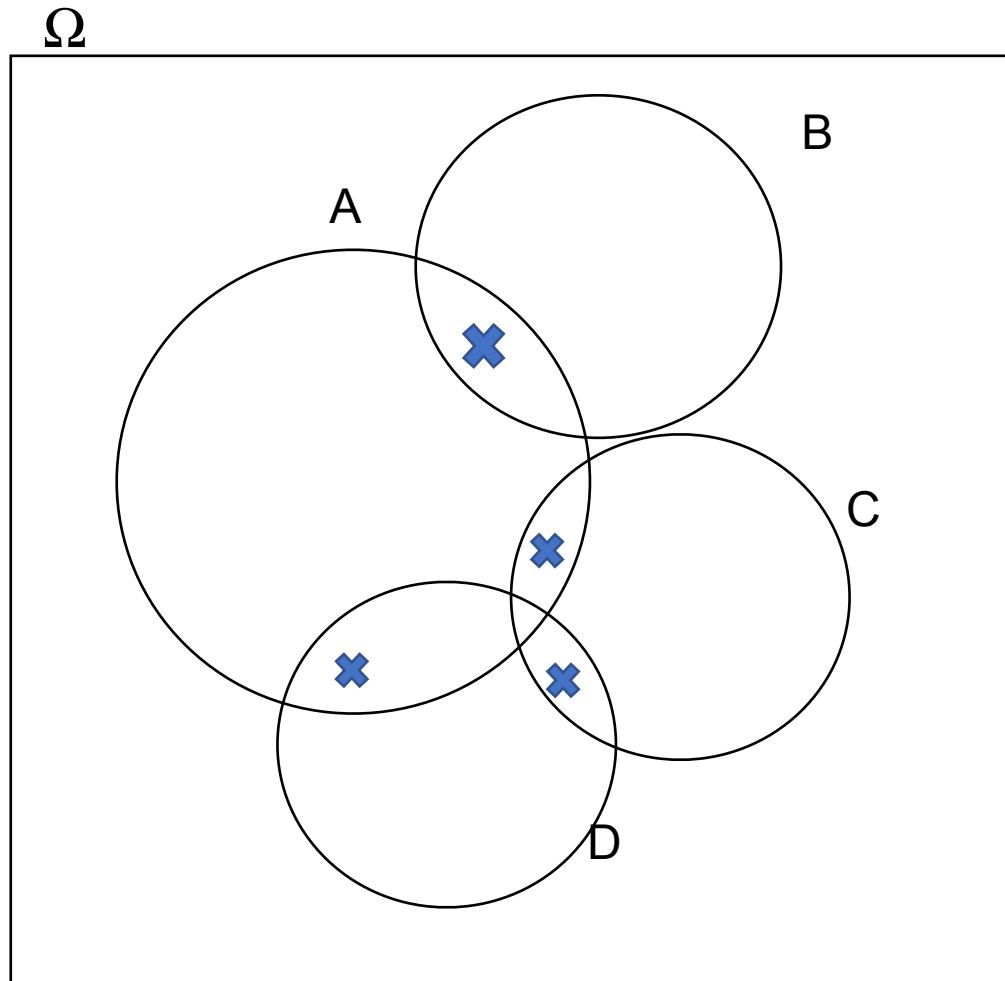
$$p(A | B) = \frac{|B \cap A|}{|B|} = \frac{1}{1} = 1$$

$$p(C | B) = \frac{|B \cap C|}{|B|} = \frac{0}{1} = 0$$

$$p(D | B) = \frac{|B \cap D|}{|B|} = \frac{0}{1} = 0$$



What is a Discrete Probability Distribution?

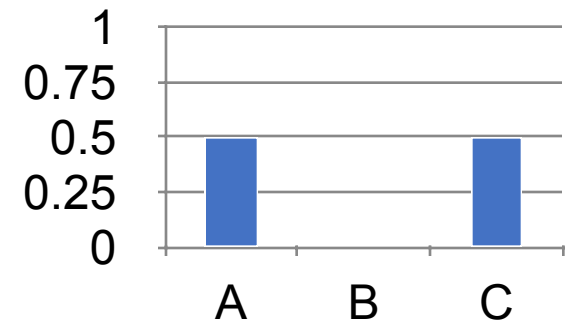


$$p(X = ? \mid D)$$

$$p(A \mid D) = \frac{|D \cap A|}{|D|} = \frac{1}{2}$$

$$p(B \mid D) = \frac{|D \cap B|}{|D|} = \frac{0}{2} = 0$$

$$p(C \mid D) = \frac{|D \cap C|}{|D|} = \frac{1}{2}$$

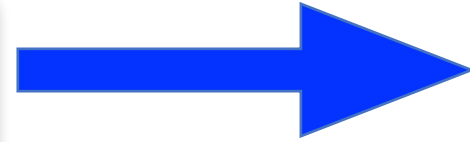


OUTLINE

- 1) A probabilistic approach to NLP
- 2) Text classification**
- 3) Generative classifier: Naïve Bayes
- 4) Evaluating text classifiers

Text Classification

i love @justinbieber #sarcasm



Positive

or

Negative?

Text Classification

From: "" <takworld@hotmail.com>

Subject: real estate is the only way... gem oalvgkay

Anyone can buy real estate with no money down

Stop paying rent TODAY !

There is no need to spend hundreds or even thousands for similar courses

I am 22 years old and I have already purchased 6 properties using the methods outlined in this truly INCREDIBLE ebook.

Change your life NOW !

=====

Click Below to order:

<http://www.wholesaledaily.com/sales/nmd.htm>

=====



Spam

or

not Spam?

Classification and categorisation

- Given:
 - A description of an **instance**, $x \in X$, where X is the *instance space*.
 - A fixed set of **categories (or classes)**:
$$C = \{c_1, c_2, \dots, c_n\}$$
- Determine:
 - The category of x : $c(x) \in C$, where $c(x)$ is a ***categorization function*** whose domain is X and whose range is C .
 - We want to know how to build categorization functions (“classifiers”).

Text Classification: definition

- *Input:*
 - a document x
 - Issue: how to represent text documents.
 - a fixed set of classes $C = \{c_1, c_2, \dots, c_J\}$
- *Output:* a predicted class $c \in C$
- *Classifier:* a function $f: x \rightarrow c$
- *Examples:* spam filtering, sentiment analysis
 - $C = \{\text{spam}, \text{not_spam}\}$
 - $C = \{\text{positive}, \text{negative}\}$
 - $C = \{\text{angry}, \text{happy}, \text{sad}, \text{excited}, \text{confused} \dots \}$

Applications of Text Classification

- Classification is the core method in applications including:
 - Document categorisation (e.g., divide news articles in articles about sport, finance, etc)
 - Sentiment analysis
 - Hate speech detection
 - Language/dialect identification
 - Deception detection
 - Stylometry
 - Spam filtering
 - And pretty much everywhere else in NLP!
- Also at the sub-document level

Supervised text classification

- An application of **supervised machine learning**
- *Input:*
 - a document x
 - a fixed set of classes $C = \{c_1, c_2, \dots, c_J\}$
 - a training set of m hand-labeled documents $(x_1, c_1), \dots, (x_m, c_m)$
- *Output:* a learned classifier $f: x \rightarrow c$

Supervised text classification

- An application of **supervised machine learning**
- *Input*:
 - a document x
 - a fixed set of classes $C = \{c_1, c_2, \dots, c_J\}$
 - a training set of m hand-labeled documents $(x_1, c_1), \dots, (x_m, c_m)$

Features of documents used for simple classification: the presence or non-presence, or counts, of WORDS in the document

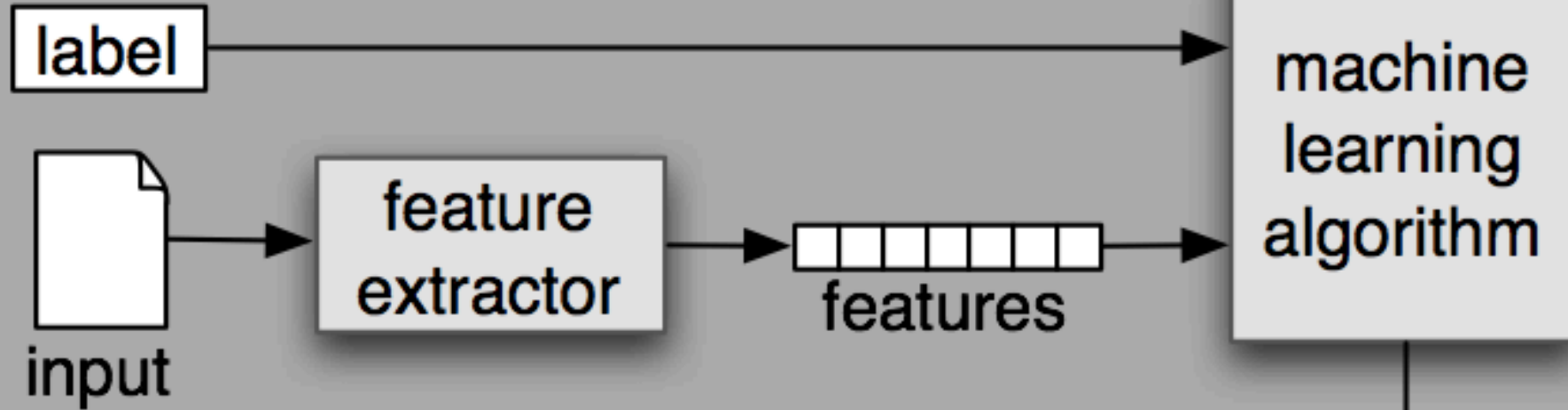
- *Output*: a learned classifier $f: x \rightarrow c$

Supervised text classification

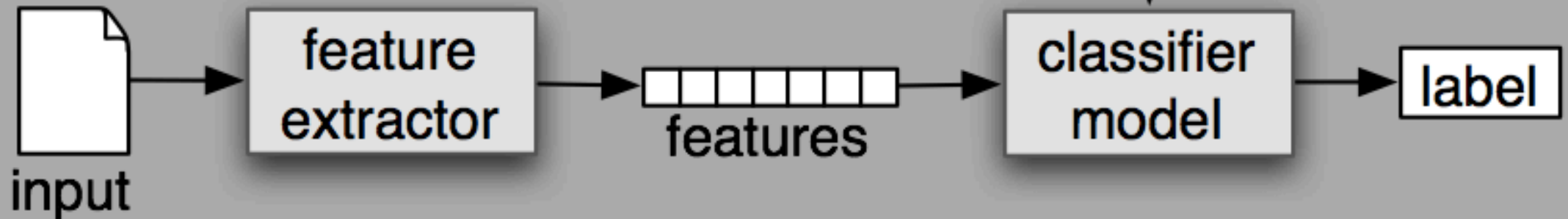
- A learned classifier $f: x \rightarrow c$ using **supervised ML methods**.
 - **Generative/probabilistic:**
 - Naive Bayes
 - Bayes nets
 - **Discriminative:**
 - Support Vector Machines (SVM)
 - Logistic Regression
 - Deep Neural Nets (more next semester NN + NLP)
 - And many more (Decision Trees, Random Forests etc.).

Supervised Machine Learning

(a) Training



(b) Prediction



Preprocessing

- How do we represent each feature instance space X ? We need to get at least the word **tokens** out of the text but can do more...
- The first step before full feature extraction is **preprocessing** the text (sometimes called **text normalisation**).
- Every NLP task using a written text requires the following steps to be applied to the raw text before extracting features:
 1. **Tokenising** (segmenting) words
 2. **Normalising** word formats
 3. **Segmenting** sentences

Tokenisation & Normalisation

- We want to split a string into words
 - Tokenisation
 - I thought "I like it!"
→ [I, thought, I, like, it]
 - First try: split on non-alphanumeric?
- And have predictable, regular forms to compress meaningless variants into one single form, reduce the number of features/ **word types**.
 - Normalisation
 - WINDOWS = Windows = windows
 - First try: convert to **lower-case**?
 - Then, try using **regexes** to match different forms?

Regular Expressions

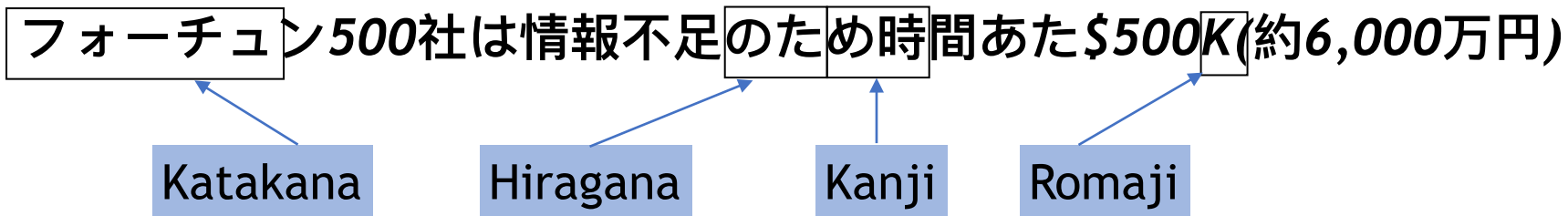
- Disjunctions `[wW]` (`windows|Windows`)
- Ranges `[a-z]` `[A-Z]` `[0-9]` `[a-zA-Z]`
- Negations `[^a]`
- Wildcards `?` `.` `*` `+`
`windows?` `window.?` `window*`
- Anchors `^` `$`
- Classes `\w` `\s` `\W` `\S`
- Find all instances of the word “the”:
`the` Misses capitalized examples
`[tT]he` Incorrectly returns other or theology
`[^a-zA-Z][tT]he[^a-zA-Z]`
`\b[tT]he\b`
- Surprisingly powerful and commonly used!

Tokenisation

- For English:
 - `split(r'\s')` or `split(r'\W')` or `split(r'\b')`
 - (or many similar options)
- How about Chinese?
 - 莎拉波娃现在居住在美国东南部的佛罗里达。
 - 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
 - Sharapova now lives in US southeastern Florida
 - Characters are generally 1 syllable and 1 morpheme.
 - Average word is 2.4 characters long. No spaces!
- Standard baseline segmentation algorithm:
 1. Start pointer at beginning of string
 2. Find longest word in dictionary matching string starting at pointer
 3. Move pointer over word, go to 2

Tokenisation

- How about Japanese?
 - multiple alphabets intermingled
 - dates/amounts in multiple formats



End-user can express query entirely in hiragana!

- How about German?
 - 'Lebensversicherungsgesellschaftsangestellter'
 - 'life insurance company employee'
 - (wait for sequence modelling lecture ...)

Stemming / Lemmatisation

- We often want base form (**stem, lemma**) rather than full word

windows → window

estimating → estimate (→ estimat)

running → run

are, is → be

- **Morphemes:**
 - The small meaningful units that make up words
 - **Stems:** The core meaning-bearing units
 - **Affixes:** Bits and pieces that adhere to stems
 - Often with grammatical functions

Porter's algorithm

The most common English stemmer

Step 1a

sses → ss	caresses → caress
ies → i	ponies → poni
ss → ss	caress → caress
s → ∅	cats → cat

Step 1b

(*v*)ing → ∅	walking → walk
	sing → sing
(*v*)ed → ∅	plastered → plaster
...	

Step 2 (for long stems)

ational → ate	relational → relate
izer → ize	digitizer → digitize
ator → ate	operator → operate

...

Step 3 (for longer stems)

al → ∅	revival → reviv
able → ∅	adjustable → adjust
ate → ∅	activate → activ

...

Why check for vowels in Step 1b?

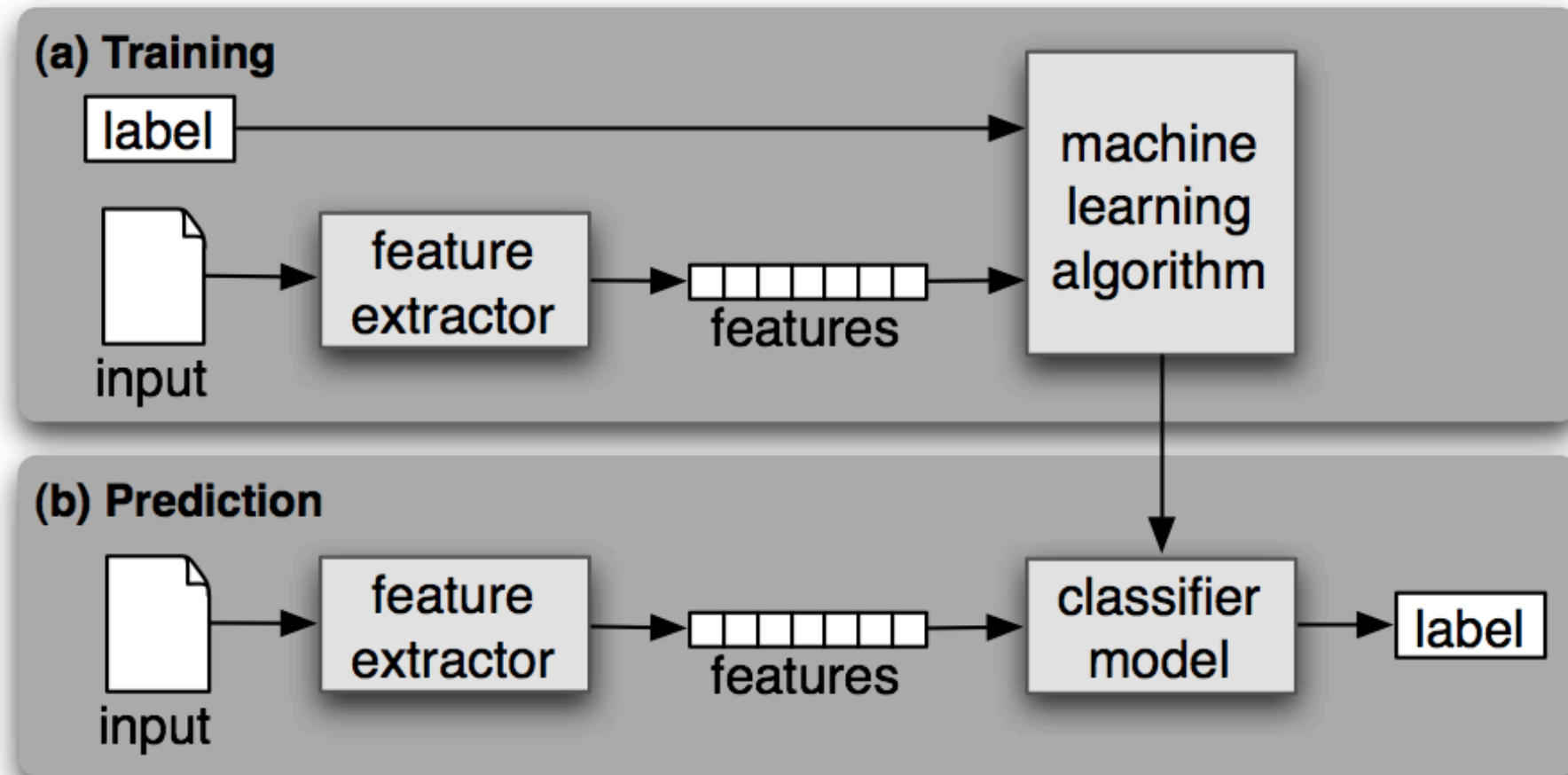
(*v*)ing → ∅	walking → walk
	sing → sing

Stemming in other languages

- What if we're working with Turkish?
 - **Uygarlaştıramadıklarımızdanmışsınızcasına**
 - `(behaving) as if you are among those whom we could not civilize'
 - **Uygar** `civilized' + **laş** `become'
 - + **tır** `cause' + **ama** `not able'
 - + **dık** `past' + **lar** `plural'
 - + **ımız** `p1pl' + **dan** `abl'
 - + **mış** `past' + **sınız** `2pl' + **casına** `as if'
- More advanced grammar-based methods:
 - e.g. finite state transducers (see later)

Features won't necessarily be derived from 'words'

- Could be stems, lemmas, other normalised forms, plus extras...
- But let's come back to that...



OUTLINE

- 1) A probabilistic approach to NLP
- 2) Text classification
- 3) Generative classifier: Naïve Bayes**
- 4) Evaluating text classifiers

Bayesian Methods

- Learning and classification methods based on probability theory.
- **Bayes rule** plays a critical role.
- Learn a **generative model** that approximates how data is produced: model learns to generate instance data from the class/category, rather than estimate the mapping from data to class directly.
- Use **prior probability** of each category.
- Categorization produces a **posterior probability** distribution over the possible categories given a description of an item.

Product Rule and Bayes Rule

$$P(C, X) = P(C | X)P(X) = P(X | C)P(C)$$

Class

Instance data/
document

Likelihood (of data given class)

Prior probability
of class

$$P(C | X) = \frac{P(X | C)P(C)}{P(X)}$$

Evidence (probability of data)

Posterior
probability



Generative model: The classification problem is 'flipped' by using the likelihood: each class learns a distribution over, or 'generates', instance data

Maximum a posteriori Hypothesis for classification

$$c_{MAP} = \operatorname{argmax}_{c_j \in C} P(c_j | X)$$

Pick the value for c_j which gives the following equation the highest (Maximum) value

$$c_{MAP} = \operatorname{argmax}_{c_j \in C} \frac{P(X | c_j)P(c_j)}{P(X)}$$

Apply Bayes Rule to replace $P(c_j | X)$

$$c_{MAP} = \operatorname{argmax}_{c_j \in C} P(X | c_j)P(c_j)$$

Denominator (normalizing constant) not needed as we are only concerned with what the most likely class is given X relative to other classes, not its probability value

Naïve Bayes Classifier

However X , the text in the document is not treated as one variable.

Task: Classify a new instance based on a tuple of attribute values $\langle x_1, x_2, \dots, x_n \rangle$

$$c_{MAP} = \operatorname{argmax}_{c_j \in C} P(c_j | x_1, x_2, \dots, x_n)$$

$$c_{MAP} = \operatorname{argmax}_{c_j \in C} \frac{P(x_1, x_2, \dots, x_n | c_j) P(c_j)}{P(x_1, x_2, \dots, x_n)}$$

$$c_{NB} = c_{ML} = \operatorname{argmax}_{c_j \in C} P(x_1, x_2, \dots, x_n | c_j) P(c_j)$$

Naïve Bayes Classifier: first approximation

- $P(c_j)$
 - Prior can be estimated from the frequency of classes in the training examples/documents.
- $P(x_1, x_2, \dots, x_n | c_j)$
 - Likelihood: simple assumption for text: attributes are text/word positions, values are words.

$$\begin{aligned} c_{NB} &= \operatorname{argmax}_{c_j \in C} P(c_j) \prod_i P(x_i | c_j) \\ &= \operatorname{argmax}_{c_j \in C} P(c_j) P(x_1 = \text{"our"} | c_j) \dots P(x_n = \text{"text"} | c_j) \end{aligned}$$

- Computationally complex: $O(|X|^n \cdot |C|)$
- Could only be estimated if a very, very large number of training examples was available.
- Lots of words very rare in text data given Zipf's law in any case (Unit 1)- the event of them occurring at specific positions (e.g. the fifth word) in a text is even more sparse:

$$p(X_5 = \text{viagra} | C = \text{spam}) = \frac{N(X_5 = \text{viagra}, C = \text{spam})}{N(C = \text{spam})} = 0$$

Naïve Bayes Classifier

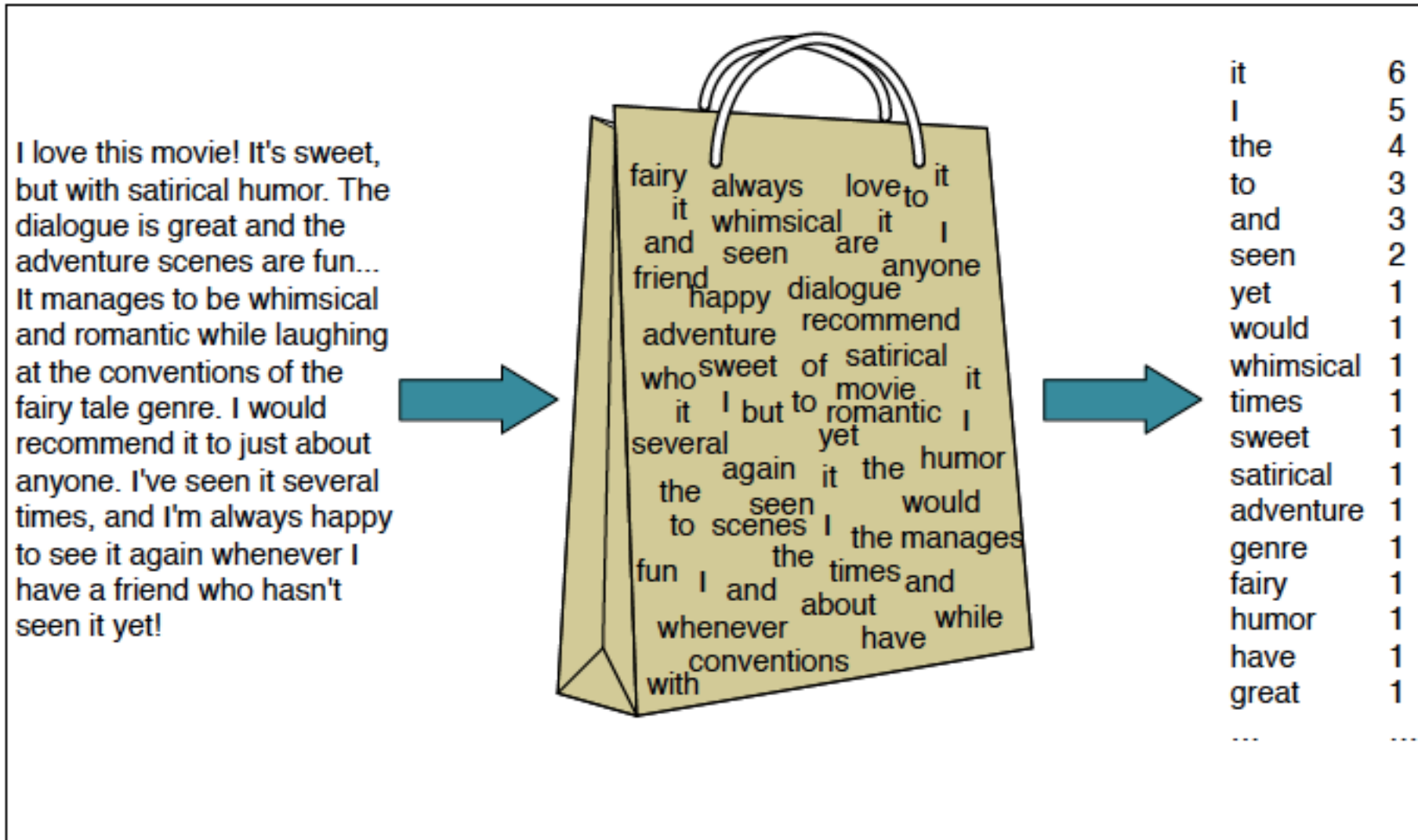
- So how can we estimate likelihood of features given class $P(x_1, x_2 \dots x_n | c)$?
- To avoid sparsity of set of attributes, **assume, naively**, that:
 - features are unordered words (a “**bag of words**”)
 - features (words appearing) are **conditionally independent**, so the probability of observing the conjunction of features is equal to the product of the individual probabilities:

$$P(x_1, \dots, x_n | c) = P(x_1 | c) \cdot P(x_2 | c) \cdot P(x_3 | c) \cdot \dots \cdot P(x_n | c)$$

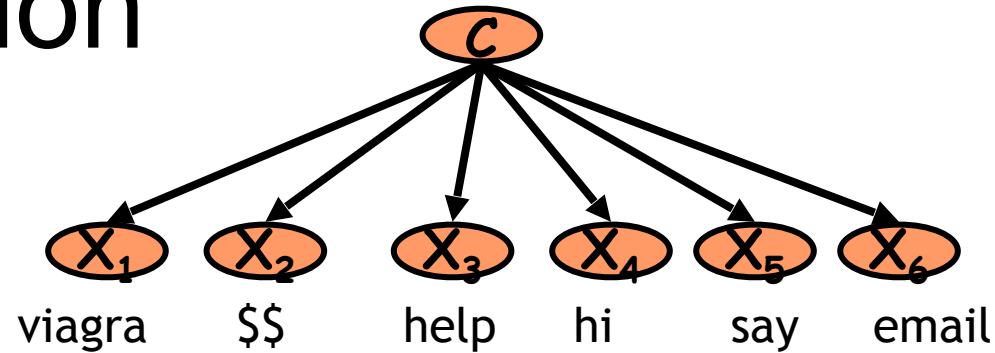
- So, the classifier can scan through the positions in a text, but each position x_i is treated as having the same parameters regardless of its position:

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

Bag of words feature counts



Learning a Naïve Bayes model: first approximation



- Common practice: Maximum Likelihood
 - simply use the frequencies in the data

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

$$\hat{P}(spam) = \frac{50}{250} = 0.2$$

Number of spam documents (points to 50)

Total number of documents (points to 250)

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

$$\hat{P}(\text{"viagra"} | spam) = \frac{count(\text{"viagra"}, spam)}{\sum_{w \in V} count(w, spam)} = \frac{24}{5000}$$

Number of times 'viagra' appears in all spam documents (points to count("viagra", spam))

Counts of words in vocab that appear in all spam documents (i.e. combined length of all spam documents) (points to denominator)

Underflow Prevention

- Multiplying lots of probabilities, which are between 0 and 1 by definition, can result in floating-point underflow when calculated by a digital computer with finite memory.
- Since $\log(xy) = \log(x) + \log(y)$, it is better to perform all computations by **summing logs of probabilities** rather than multiplying probabilities first before obtaining the log.

$$\begin{aligned} c_{NB} &= \operatorname{argmax}_{c_j \in C} P(c_j) \prod_{i \in \text{positions}} P(x_i | c_j) \\ &= \operatorname{argmax}_{c_j \in C} \log(P(c_j)) + \sum_{i \in \text{positions}} \log(P(x_i | c_j)) \end{aligned}$$

- Class with highest final un-normalized log probability score is still the most probable.

Worked example

	#DOCS	viagra	\$\$\$	help	matt	hi	and	email	nlp	Total
SPAM	50	24	30	14	7	33	50	23	0	181
NOT	200	1	0	21	150	129	200	51	32	584

Worked example

	#DOCS	viagra	\$\$\$	help	matt	hi	and	email	nlp	Total
SPAM	50	24	30	14	7	33	50	23	0	181
NOT	200	1	0	21	150	129	200	51	32	584

	P(c)	P(w c)								Total
SPAM	0.2	0.132	0.166	0.077	0.038	0.182	0.276	0.127	0.000	1
NOT	0.8	0.002	0.000	0.036	0.257	0.221	0.342	0.087	0.054	1

Worked example

	#DOCS	viagra	\$\$\$	help	matt	hi	and	email	nlp	Total
SPAM	50	24	30	14	7	33	50	23	0	181
NOT	200	1	0	21	150	129	200	51	32	584

	P(c)	P(w c)								Total
SPAM	0.2	0.132	0.166	0.077	0.038	0.182	0.276	0.127	0.000	1
NOT	0.8	0.002	0.000	0.036	0.257	0.221	0.342	0.087	0.054	1

	log	log							
SPAM	-1.61	-2.02	-1.8	-2.56	-3.25	-1.70	-1.29	-2.06	-Inf
NOT	-0.22	-6.37	-Inf	-3.33	-1.36	-1.51	-1.07	-2.44	-2.9

- Can we classify:

hi matt

hi matt want some viagra?

Problem with Max Likelihood

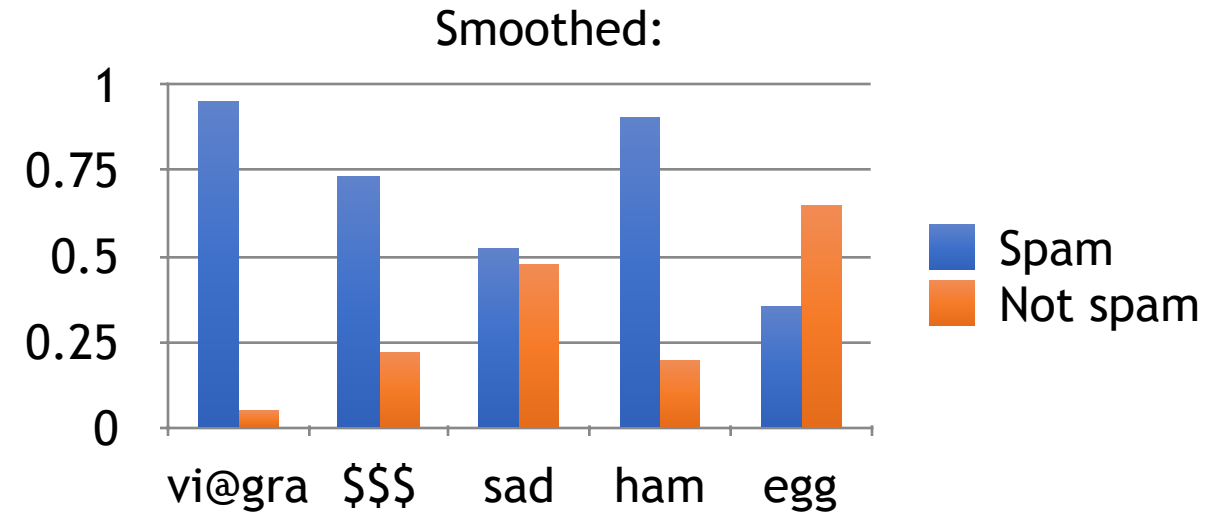
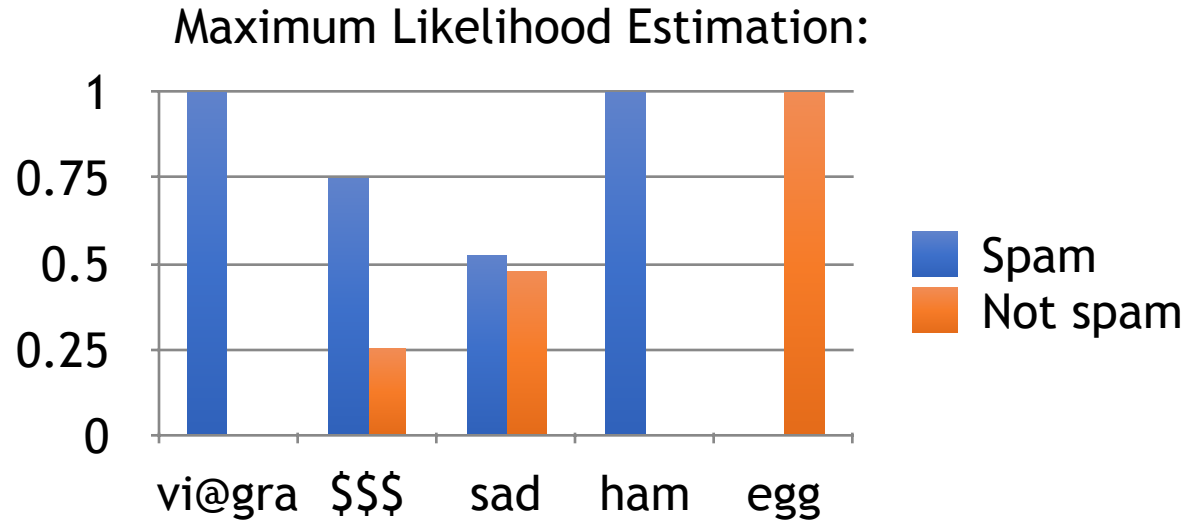
- What if we haven't seen "nlp" with $c = \text{spam}$? i.e. have no training cases leading to no co-occurrence of the word with the class?

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

- Zero probabilities cannot be conditioned away, no matter the other evidence!
- Also give infinite negative values for log probs.

Solution: smoothing

- Need to reserve some probability mass for unobserved events: **smoothing**




- E.g. **Laplace (add-1) smoothing** for Naïve Bayes; ‘pretend’ we’ve seen each word co-occur with each class once more than observed in training:

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

Size of the vocabulary

Worked example

Add 1 to all counts.
Now no 0
counts.



	#DOCS	viagra	\$\$\$	help	matt	hi	and	email	nlp	Total
SPAM	50	25	31	15	8	34	51	24	1	189
NOT	200	2	1	22	151	130	201	52	33	592

Worked example

	#DOCS	viagra	\$\$\$	help	matt	hi	and	email	nlp	Total
SPAM	50	25	31	15	8	34	51	24	1	189
NOT	200	2	1	22	151	130	201	52	33	592

	P(c)	P(w c)								Total
SPAM	0.2	0.132	0.164	0.079	0.042	0.180	0.269	0.127	0.005	1
NOT	0.8	0.003	0.002	0.037	0.255	0.220	0.340	0.088	0.056	1

Now no 0 probs.
Some higher non-0
probs slightly lower
(discounted); some
previously low probs
now higher

Worked example

	#DOCS	viagra	\$\$\$	help	matt	hi	and	email	nlp	Total
SPAM	50	25	31	15	8	34	51	24	1	189
NOT	200	2	1	22	151	130	201	52	33	592

	P(c)	P(w c)								Total
SPAM	0.2	0.132	0.164	0.079	0.042	0.180	0.269	0.127	0.005	1
NOT	0.8	0.003	0.002	0.037	0.255	0.220	0.340	0.088	0.056	1

	log	log							
SPAM	-1.61	-2.02	-1.8	-2.53	-3.16	-1.71	-1.31	-2.06	-5.24
NOT	-0.22	-5.69	-6.38	-3.29	-1.37	-1.52	-1.08	-2.43	-2.89

Now no infinite
negative values

- Now, can we classify:

hi matt

hi matt want some viagra?

Worked example

	$\log(P(c))$	viagra	\$\$\$	help	matt	hi	and	email	nlp
SPAM	-1.61	-2.02	-1.8	-2.53	-3.16	-1.71	-1.31	-2.06	-5.24
NOT	-0.22	-5.69	-6.38	-3.29	-1.37	-1.52	-1.08	-2.43	-2.89

- Can we classify:

hi matt

$$C_{NB} = \operatorname{argmax}_{c_j \in C} \log(P(c_j)) + \sum_{i \in \text{positions}} \log(P(x_i | c_j))$$

	$\log(P(c_j))$	$\sum_{i \in \text{positions}} \log(P(x_i c_j))$	$\log(P(c_j)) + \sum_{i \in \text{positions}} \log(P(x_i c_j))$
SPAM	-1.61	-1.71 + -3.16	-6.49
NOT	-0.22	-1.52 + -1.37	-3.11

hi matt want some viagra?

	$\log(P(c_j))$	$\sum_{i \in \text{positions}} \log(P(x_i c_j))$	$\log(P(c_j)) + \sum_{i \in \text{positions}} \log(P(x_i c_j))$
SPAM	-1.61	-1.71 + -3.16 + -2.02	-8.51
NOT	-0.22	-1.52 + -1.37 + -5.69	-8.80

Text Classification Algorithms: Training

- **From** training corpus, **extract** *Vocabulary*
- **Calculate** required $P(c_j)$ and $P(x_k | c_j)$ terms:

- **For each** c_j in C **do**

$docs_j \leftarrow$ subset of documents for which the target class is c_j

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

$Text_j \leftarrow$ single document containing all $docs_j$

For each word x_k in *Vocabulary* **do**:

$n_k \leftarrow$ number of occurrences of x_k in $Text_j$

$$P(x_k | c_j) \leftarrow \frac{n_k + 1}{n + |Vocabulary|}$$

Text Classification Algorithms: Classifying

- positions \leftarrow all word positions in current document which contain tokens found in *Vocabulary*
- **Return** c_{NB} , where

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

A note about Naïve Bayes posterior probabilities

- Classification results of Naïve Bayes (the class with maximum posterior probability) are usually fairly **accurate classifiers** on test data.
- However, due to the inadequacy of the conditional independence assumption, the actual posterior-**probability numerical estimates are not very accurate.**
 - Output probabilities are generally very close to 0 or 1.
- Also, Laplace add-1 smoothing is not the most effective smoothing. **Add-k (Lidstone)** smoothing and others (more details in Unit on Language Modelling). Can make quite a big difference in performance.

OUTLINE

- 1) A probabilistic approach to NLP
- 2) Text classification
- 3) Generative classifier: Naïve Bayes
- 4) Evaluating text classifiers

Classifier 1

i love @justinbieber

Ground truth
Label

Positive

Prediction by classifier

Positive



i love @justinbieber #sarcasm

Negative

Positive



So very grateful for what
BTS has done for me. I've
made some amazing friends
because of them, also.
Saranghae.

Positive

Positive



@BTS are so over-rated!

Negative

Negative



RAW ACCURACY = 3/4

Classifier 2

i love @justinbieber

Ground truth
Label

Positive

Prediction by classifier

Positive



i love @justinbieber #sarcasm

Negative

Negative



So very grateful for what
BTS has done for me. I've
made some amazing friends
because of them, also.
Saranghae.

Positive

Negative



@BTS are so over-rated!

Negative

Negative



RAW ACCURACY = 3/4

Measuring Performance

- Classification accuracy most simply put seems to be what % of messages were classified correctly?
- **Is this what we care about?** (Two classifiers could have the same overall accuracy but behave differently)

	Overall accuracy	Accuracy on spam	Accuracy on ham
System 1	95%	99.99%	90%
System 2	95%	90%	99.99%

- **Which system do you prefer ('safe' or not over-filtering)?**

Precision and recall

- **Precision:** % of predictions by classifier of class C that are correct predictions
- **Recall:** % of ground truth items of class C that are predicted correctly by classifier

Confusion matrix

		Gold/ground-truth label	
		Ground truth label class C	Ground truth label not class C
System's output	Class C predicted	tp	fp
	Class C not predicted	fn	tn

$$P = \text{tp}/(\text{tp}+\text{fp})$$

$$R = \text{tp}/(\text{tp}+\text{fn})$$

tp: true positive
fp: false positive
fn: false negative

Precision and recall

- **E.g.** a test set of 100 examples, balanced dataset (50 C, 50 not C):

Confusion matrix

		Gold/ground-truth label	
		Ground truth label class C	Ground truth label not class C
System's output	Class C predicted	24	6
	Class C not predicted	26	44

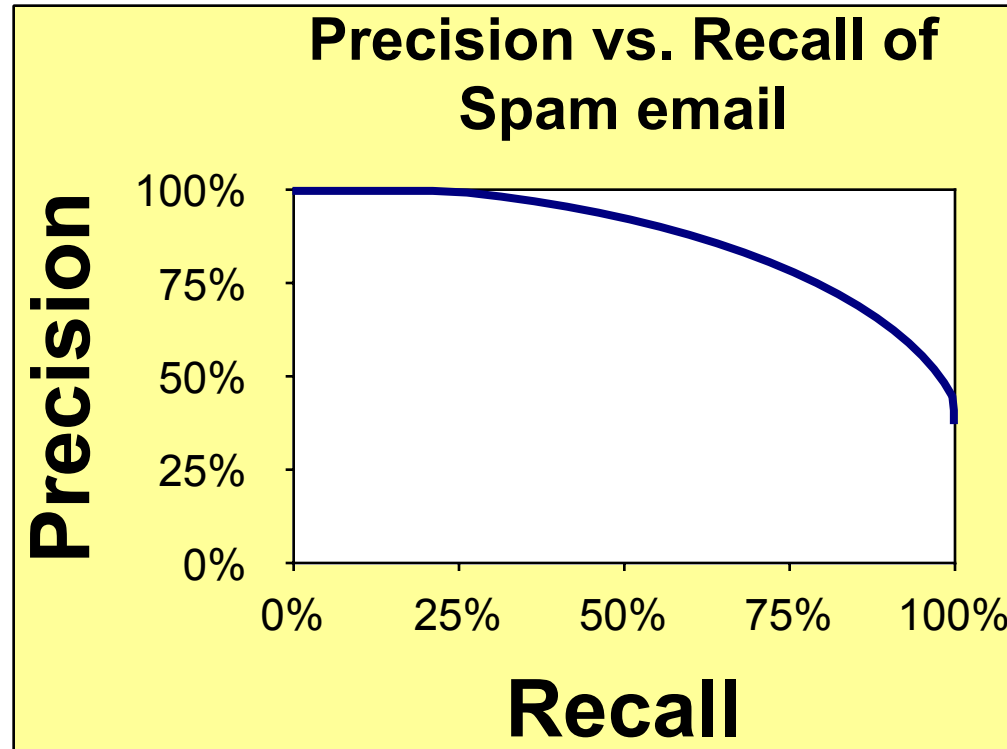
$$P = tp/(tp+fp)$$

$$24/(24+6) = 4/5 = 0.8$$

$$R = tp/(tp+fn)$$

$$24/(24+26) = 0.48$$

Precision and recall



- **Precision** =
$$\frac{\text{\#spam correctly predicted}}{\text{\#spam predicted}}$$
- **Recall** =
$$\frac{\text{\#spam correctly predicted}}{\text{\#actual spam messages}}$$

Trade off precision vs. recall by setting threshold

Measure the curve on annotated development data (or test data)

Choose a threshold where user is comfortable

A combined measure: F

- A combined measure that assesses the P/R tradeoff is F measure (weighted HARMONIC MEAN)
- General F measure formula:

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

- Most used version: F1 measure
 - i.e., with $\beta = 1$ (that is, $\alpha = 1/2$):

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

Why not simple (arithmetic) mean?

- Consider the case of a (binary) classification problem, and a system that classifies every item as 1
 - That system would have $R = 1$, but $P = 0$
 - So the mean of R and P would be .5 even though this is the worst possible system
 - But that system would have an $F1$ of 0 (check)
- It is always possible to inflate R at the expense of P and vice-versa.
- Balancing the weight of R and P maximally for the best F-score is therefore vital to indicating good performance.

Splitting dataset

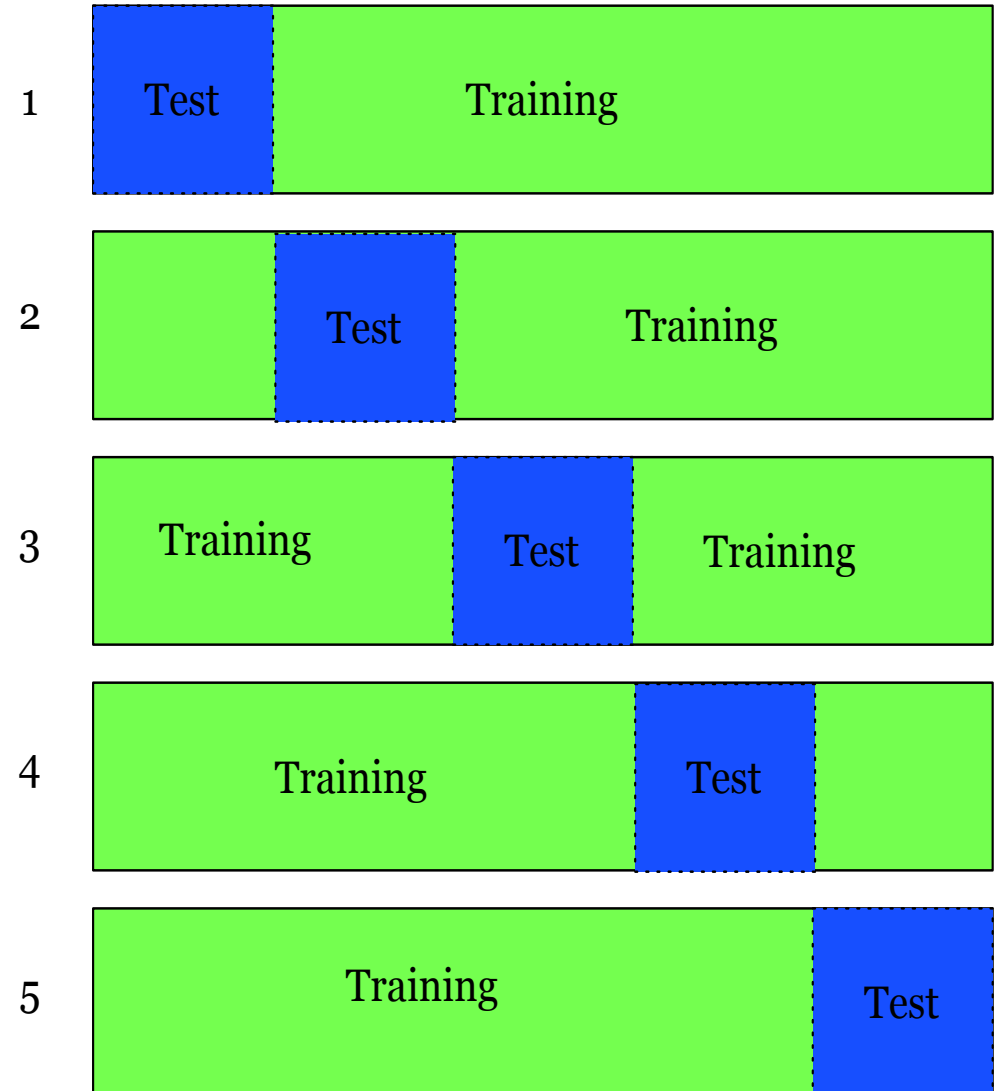


- **Training set**
 - (labelled however you want/appropriate for the task).
 - **Don't test on this! Train/fit only!**
- **Held-out set**
 - Labelled similarly to the main training set, but not used for training- like a practice for the real test, to allow hyper parameters to be experimented with.
 - **Iteratively develop based on results on this set (Do not eyeball/optimize on the test set!)**
- **Unseen test set**
 - (should be independently labelled).
 - **Do not eye-ball of fine-tune on the test data!**- this test run should be done very occasionally after development (or in a competition, only once!)
 - More conservative estimate of performance than on heldout data normally.

Cross-Validation

- **Cross-validation** over multiple splits
- You can repeat the train/heldout split multiple times by **iterating** over the training data.
- For **k-fold cross-validation**, break up data into k folds.
 - (Equal positive and negative inside each fold, or random?)
- For each fold
 - Choose the fold as a temporary test set.
 - Train on k-1 folds, compute performance on the test fold.
- Report **average (macro) performance** of the k runs (and micro too over all instances if useful).
- Better handles **sampling errors/biases** from different datasets.

Iteration



Evaluation and development tips

- What is/are the class(es) of interest for the application? Not all will be equally important.
- Don't just look at the **micro-averaged accuracy scores** across all classes, look at the **macro/class averages** and **individual class performances** too.
- Look at how the classifier is being confused, through a **confusion matrix**.

		<i>gold labels</i>			
		urgent	normal	spam	
<i>system output</i>	urgent	8	10	1	$\text{precision}_u = \frac{8}{8+10+1}$
	normal	5	60	50	$\text{precision}_n = \frac{60}{5+60+50}$
	spam	3	30	200	$\text{precision}_s = \frac{200}{3+30+200}$
		$\text{recall}_u = \frac{8}{8+5+3}$	$\text{recall}_n = \frac{60}{10+60+30}$	$\text{recall}_s = \frac{200}{1+50+200}$	

Evaluation and development tips

- Which classes are doing better and worse? Which ones are being confused for which?
- Remember one class's FN is another class's FP and vice-versa. 'Low precision' or 'low recall' in general can be less meaningful than talking about those measures on **specific classes** (particularly if the target class is rare/sparse).
- Are some errors/confusions more important to work on than others?

Evaluation and development tips

- **Qualitative analysis:** print out and **look at the data and the error types!**
- This will give you solutions and ideas.
- e.g. **Preprocessing improvement:** a word like 'Escort' could be a car or could be something spammy- if you've gotten rid of capitalization, you might need to add it back in. Capitalisation indicates 'spamminess' e.g. "DEAR SIR:"
- e.g. **Feature selection/augmentation.** E.g. 'not' changes the meanings of words. Simple method: Add NOT_ to every word between the negation word and following punctuation:

didn't like this movie , but I

becomes:

didn't NOT_like NOT_this NOT_movie but I

SVMs vs other approaches:

Classic Reuters-21578 Data Set

- Most (over)used data set.
- 21578 documents:
 - 9603 training, 3299 test articles
- 118 categories.
 - An article can be in more than one category.
 - Learn 118 binary category distinctions.
- Average document: about 90 types, 200 tokens.
- Quite sparse problem: only about 10 out of 118 categories are large.

Common categories
(#train, #test)

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

READING

- Manning and Schuetze (1998)
 - Chapter 2 Mathematical Foundations
- Jurafsky and Martin (2021, online)
 - Chapter 2 [Regular Expressions, Text Normalization, Edit Distance](#)
 - Chapter 4 [Naive Bayes and Sentiment Classification](#)