

**LECTURE 7**

# **Course Review**

---

# Topics to Review

## Fundamental Annotation Concepts

- MATTER
- MAMA
- Task Definition

## Annotation model & specification

- Generality vs Specificity
- XML vs JSON

## Annotation Guidelines

- Specification vs Guidelines
- Annotation Environment

## Inter-annotator agreement

- Cohen's Kappa

## Machine learning experiment

- Data segregation
- Target Function

## Modeling in Machine Learning

- Information gain & Entropy
- Naive Bayes
- Decision Trees

# **Fundamental Annotation Concepts**

(Chapter 1 & 2)

MATTER

MAMA

# Layers of linguistic description


- **Syntax** - how words and phrases are combined into phrases and sentences, respectively;
- **Semantics** - meanings of phrases/sentences and relations over these meanings;
- **Morphology** - smallest units of language that has meaning or function (aka morphemes);
- **Phonology** - sound patterns of a particular language;
- **Phonetics** - sounds of human speech, and how they are made and perceived;

Aminata was feeling "pretty frustrated" as she made her way home from her law studies class. The 19-year-old had been excited to vote for the first time in the French presidential election and had been glued to it on social media, but her candidate, the hard-left Jean-Luc Mélenchon ...

# Layers of linguistic description (II)

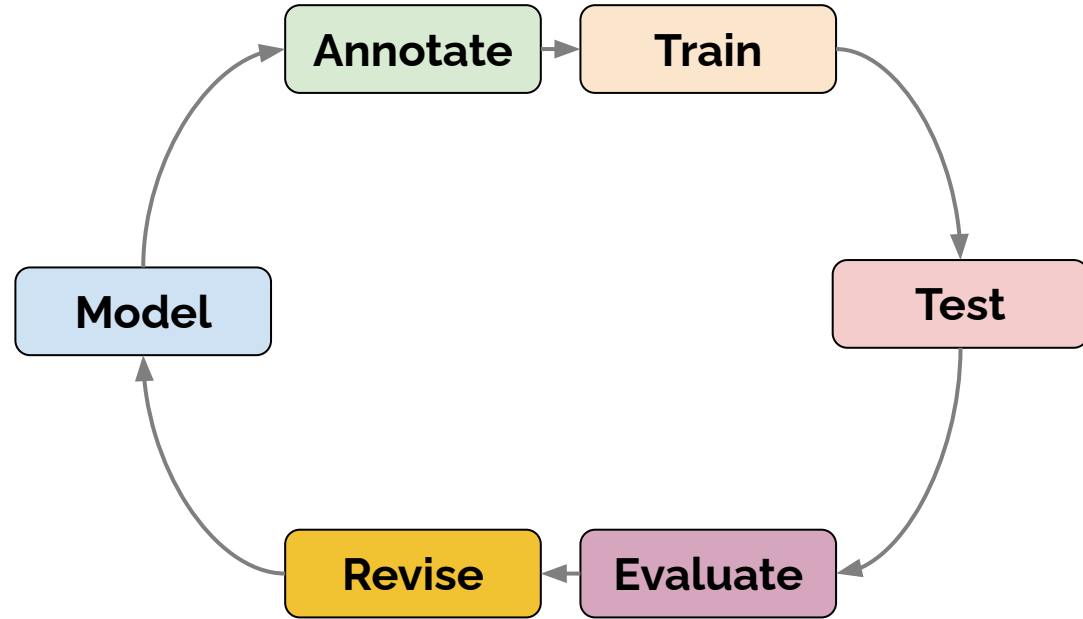
- **Lexicon** - words and phrases used in a language (vocabulary);
- **Discourse analysis** - exchanges of information and the flow of information across sentence boundaries;
- **Pragmatics** - how the context of text affects the meaning of an expression and how to recover a hidden/presupposed meaning;
- **Text structure analysis** - how narratives and other textual styles are constructed to make larger textual compositions;

Leave me now,  
and take the  
biggest part of me.



*valentines day*

# MATTER cycle



**Model** the phenomenon

**Annotate** with the specification

**Train** algorithms

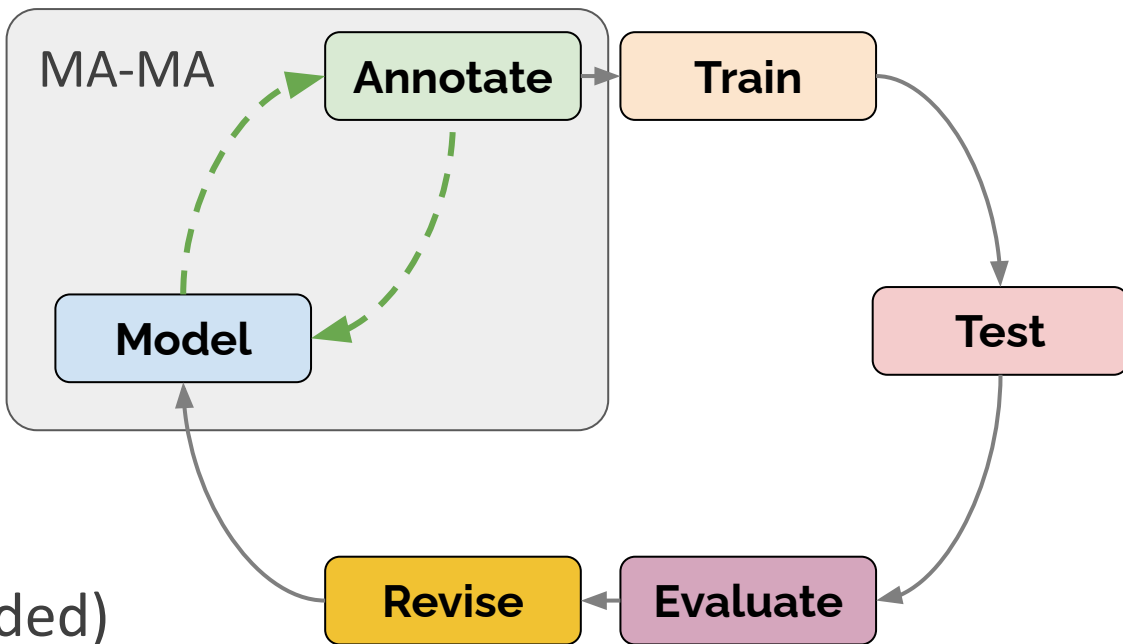
**Test** them on unseen data

**Evaluate** the results

**Revise** the model and algorithms

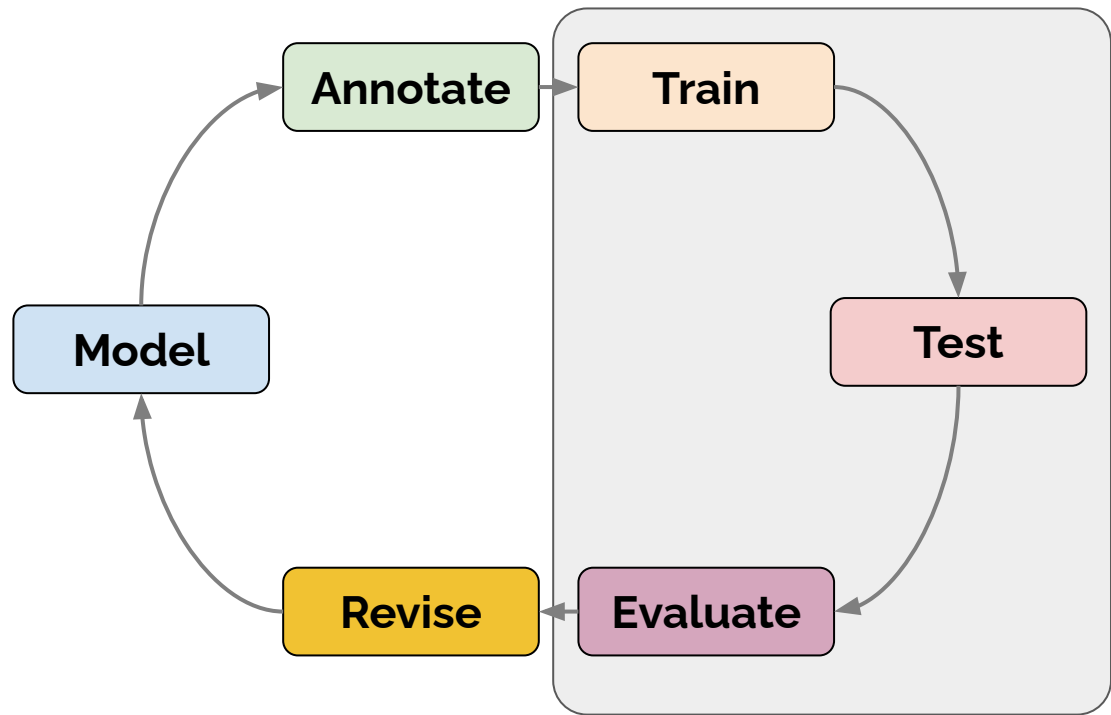
# MATTER: Annotate with the specification

- Design specification
  - Consuming tags
  - Non-consuming tags
- Write guidelines
  - Span of the tag
- Annotate
- Evaluate & revise (if needed)
  - Understanding or just agreement?
- Adjudication & gold standard



# MATTER:

Train, test, evaluate





# **Annotation model & specification**

(Chapter 3)

Generality vs Specificity

XML vs JSON



# JSON vs XML

---





# XML & JSON & JSON Lines

## Why XML

- Limitless Customisation
- Inline annotations are more interpretable
- Typeless
- Supports namespaces
- More secure

## Why JSON

- Easy (familiar with minimal dependencies)
- Fast
- Free
- No mapping

## Why JSON lines

- Appendable
- Adaptable



# Document-level annotation

- Assign genres to movies:

- *Action*
- ...
- *Thriller*



```
{  
  "title": "The Last of the Mohicans",  
  "year": "1992",  
  "cast": [  
    "Michael Douglas",  
    "Catherine O'Hara",  
    "John Heard"  
  ],  
  "genres": [  
    "Action",  
    "Adventure",  
    "Comedy"  
  ]  
}
```



Annotation Guidelines (Chapter 6)

Specification vs Guidelines

Annotation Environment

# The Specification (spec)

- The model is captured by a specification, or spec. But what does a spec look like?
- You have the goals for your annotation project. Where do you start? How do you turn a goal into a model?
- What form should your model take? Are there standardized ways to structure the phenomena?
- How do you take someone else's standard and use it to create a specification?
- What do you do if there are no existing specifications, definitions, or standards for the kinds of phenomena you are trying to identify and model?
- How do you determine when a feature in your description is an element in the spec versus an attribute on an element?

The spec is the concrete representation of your model. So, whereas the model is an abstract idea of what information you want your annotation to capture, and the interpretation of that information, the spec turns those abstract ideas into tags and attributes that will be applied to your corpus.

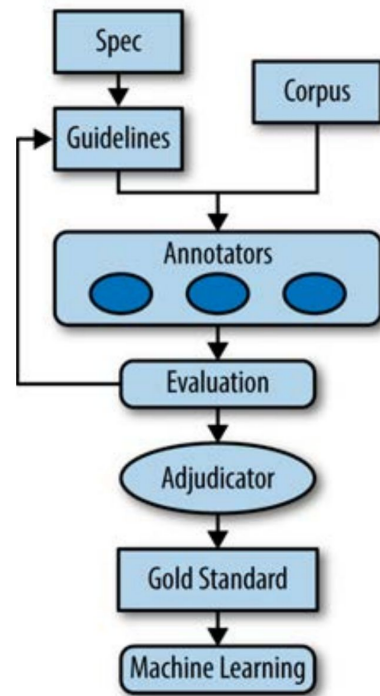
(Pustejovsky and Stubbs 2012, 67)

# Annotation Specification

Why use a specification?

1. Scale
2. Generality

(Pustejovsky and Stubbs 2012, 106)



## **Inter-annotator agreement**

(Chapter 6)

Cohen's Kappa

# Accuracy as IAA

Where do you see more (trustworthy) agreement?

		B	
		positive	negative
A	positive	4	1
	negative	2	43

1

		B	
		positive	negative
A	positive	25	1
	negative	2	22

21

2

IAA measures:  $\kappa$

Kappa

Accuracy and F1 score don't take into account expected chance agreements that are likely to occur when people annotate texts

The measures taking expected chance agreement into account:

- Cohen's  $\kappa$ : two annotators annotating each subject with a category
- Fleiss'  $\kappa$ : each subject was annotated  $n$  times with a category

Observed  
agreement

$$\kappa \equiv \frac{A_o - A_c}{1 - A_c}$$

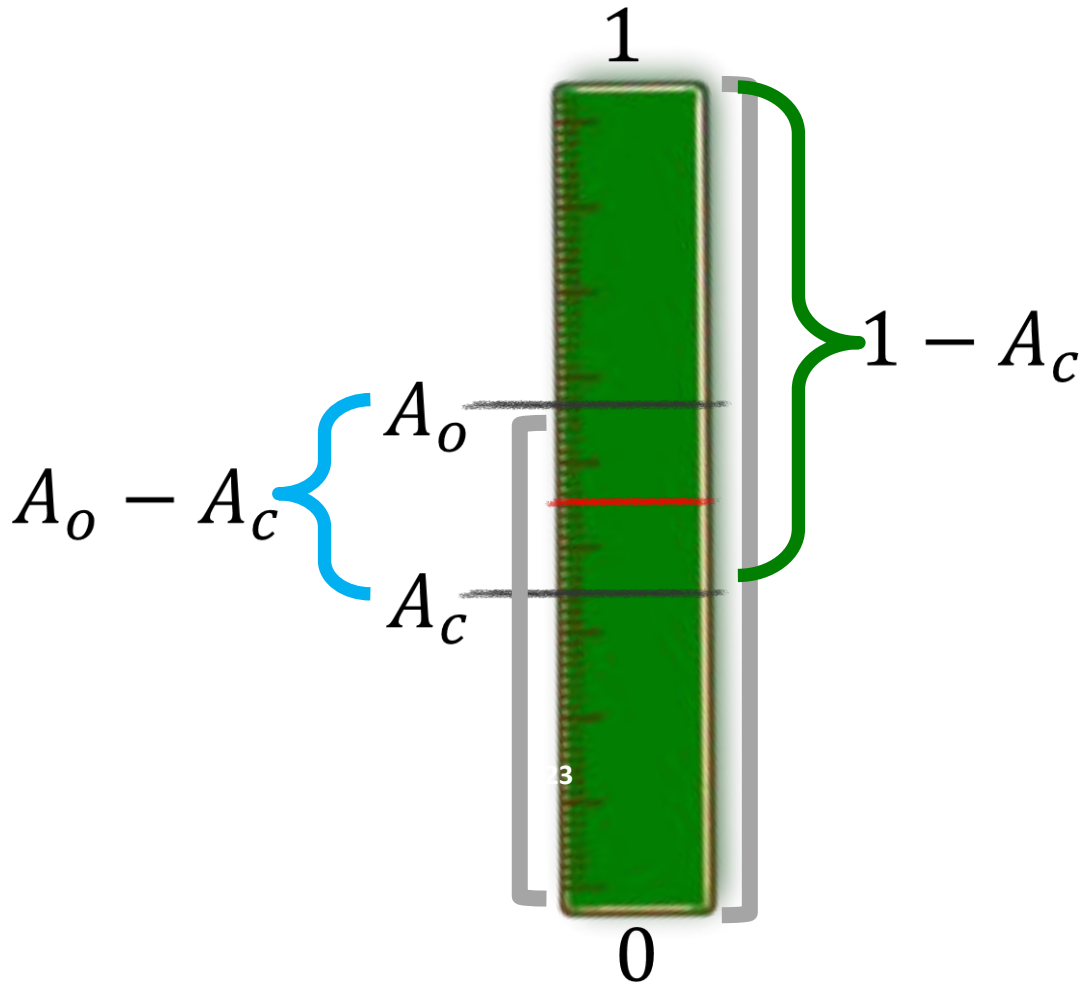
Chance  
agreement

# Idea behind $\kappa$ measures

Observed  
agreement

$$\kappa \equiv \frac{A_o - A_c}{1 - A_c}$$

Chance  
agreement

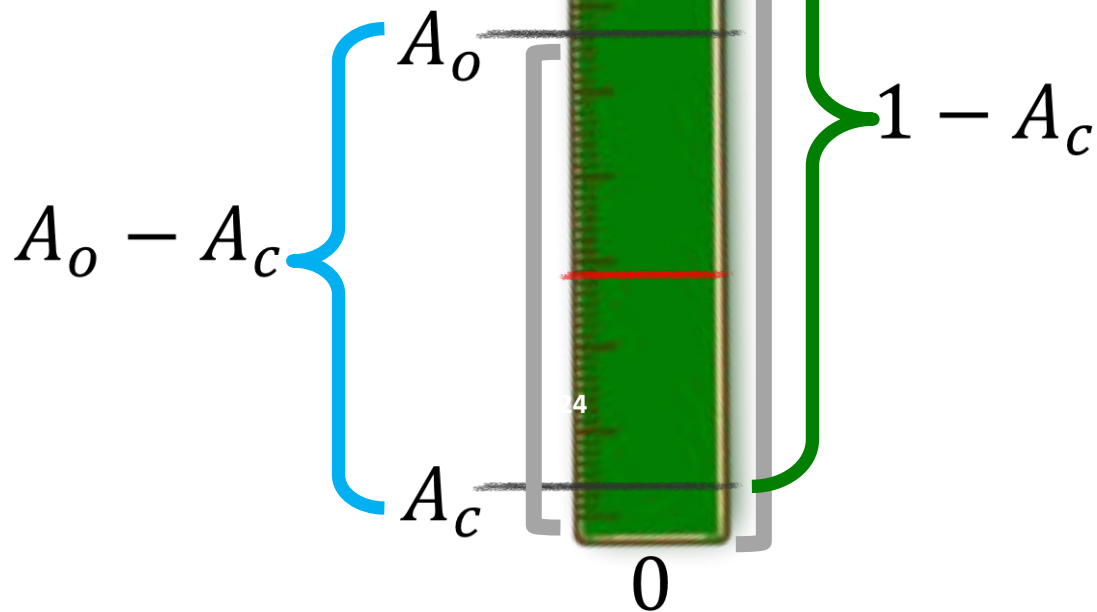


# Idea behind $\kappa$ measures

Observed  
agreement

$$\kappa \equiv \frac{A_o - A_c}{1 - A_c}$$

Chance  
agreement





# Cohen's $\kappa$

Observed  
agreement

$$\kappa \equiv \frac{A_o - A_c}{1 - A_c}$$

Chance  
agreement

## Example

250 movie reviews

Annotated by two coders:

- Ann and Bob

With three categories:

- Positive
- Neutral
- Negative



\*The errata: <https://www.oreilly.com/catalog/errata.csp?isbn=0636920020578>



Cohen's  $\kappa$ : calculate  $p_o$

Total items with  
annot. agreement

Total items

		B			
		positive	neutral	negative	
A	positive	54	28	3	
	neutral	31	18	23	
	negative	0	21	72	
					0.5760

Accuracy: ratio of agreed annotations  
and total items

Cohen's  $\kappa$ : calculate  $p_c$

Calculate distribution of categories per annotator

144		B			85/250	
	250	positive	neutral	negative		
A	positive	54	28	3	85	0.3400
	neutral	31	18	23	72	0.2880
	negative	0	21	72	93	0.3720
		85	67	98	0.5760	
		0.3400	0.2680	0.3920		

Cohen's  $\kappa$ : calculate  $p_c$

Calculate distribution of categories per annotator

Calculate chance agreement from the distributions

144		B					
	250	positive	neutral	negative			
A	positive	54	28	3	85	0.3400	0.1156
	neutral	31	18	23	72	0.2880	0.0772
	negative	0	21	72	93	0.3720	0.1458
		85	67	98	0.5760		0.3386
		0.3400	0.2680	0.3920			

0.34 x 0.34

$\Sigma$

$p_c$

Cohen's  $\kappa$ : calculate

$\kappa \equiv \frac{A_o - A_c}{1 - A_c}$							
						0.3400	0.1156
						0.2880	0.0772
						0.3720	0.1458
		85	67	98	0.5760	0.6614	0.3386
		0.3400	0.2680	0.3920	0.2374	0.3589	



# **Setup of a machine learning experiment**

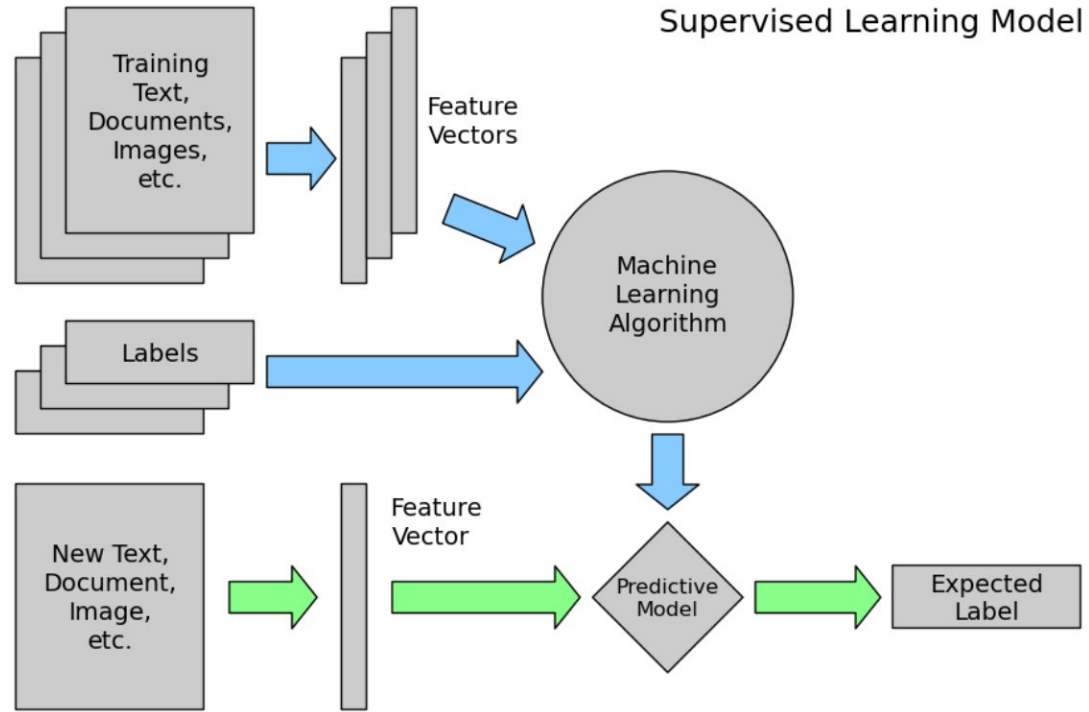
(Chapter 7)

Experimental Setup

Target Function

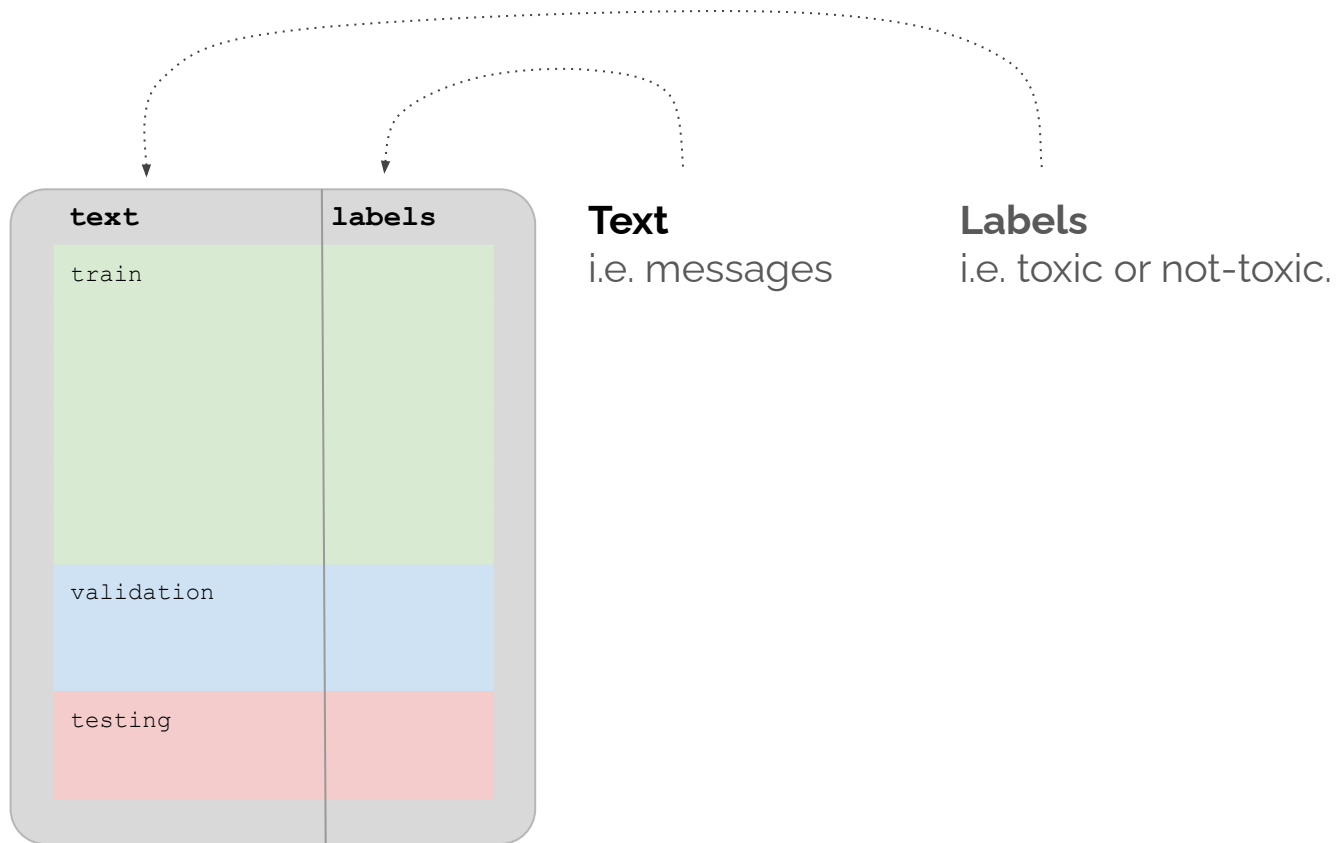
# Experimental Setup

# Supervised Learning

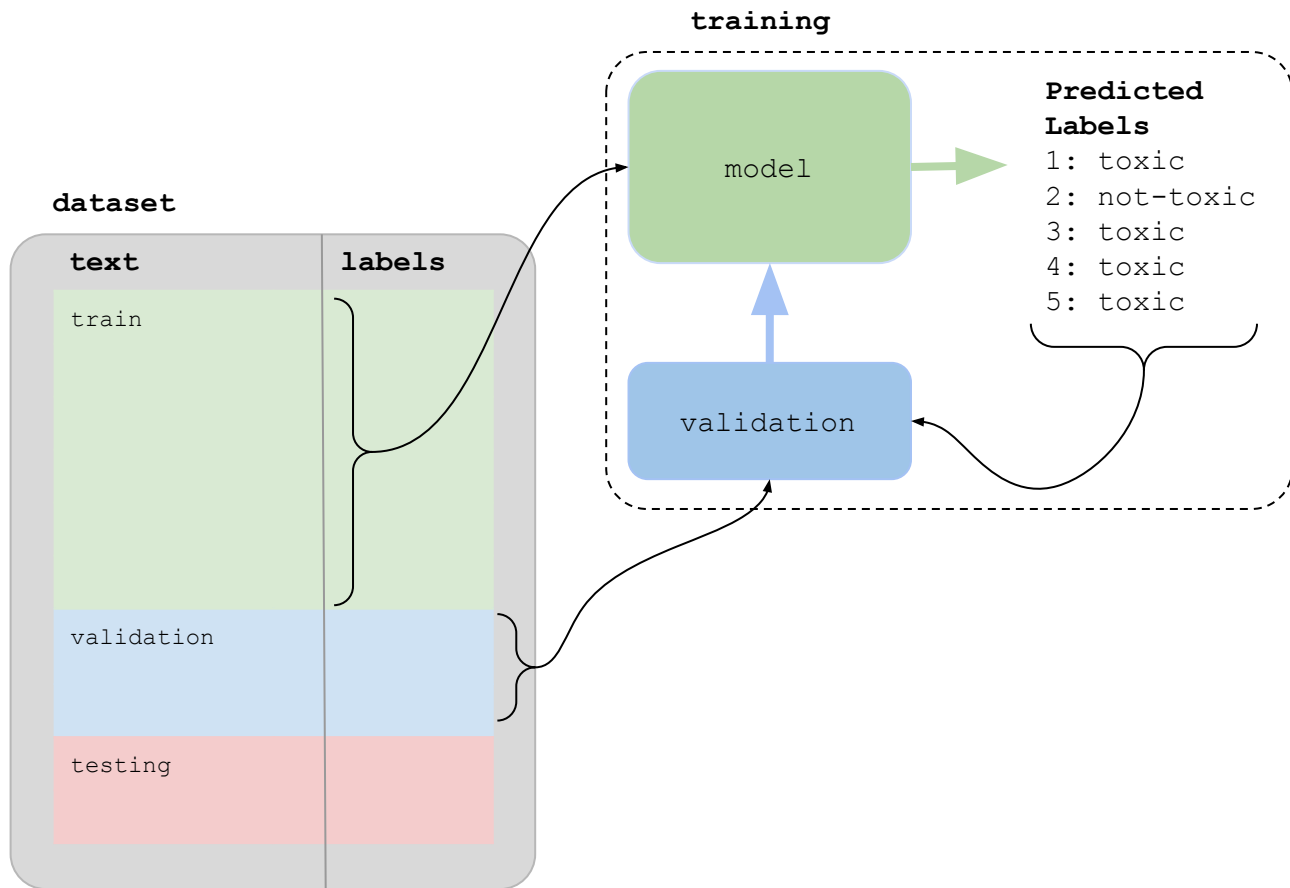




# A dataset

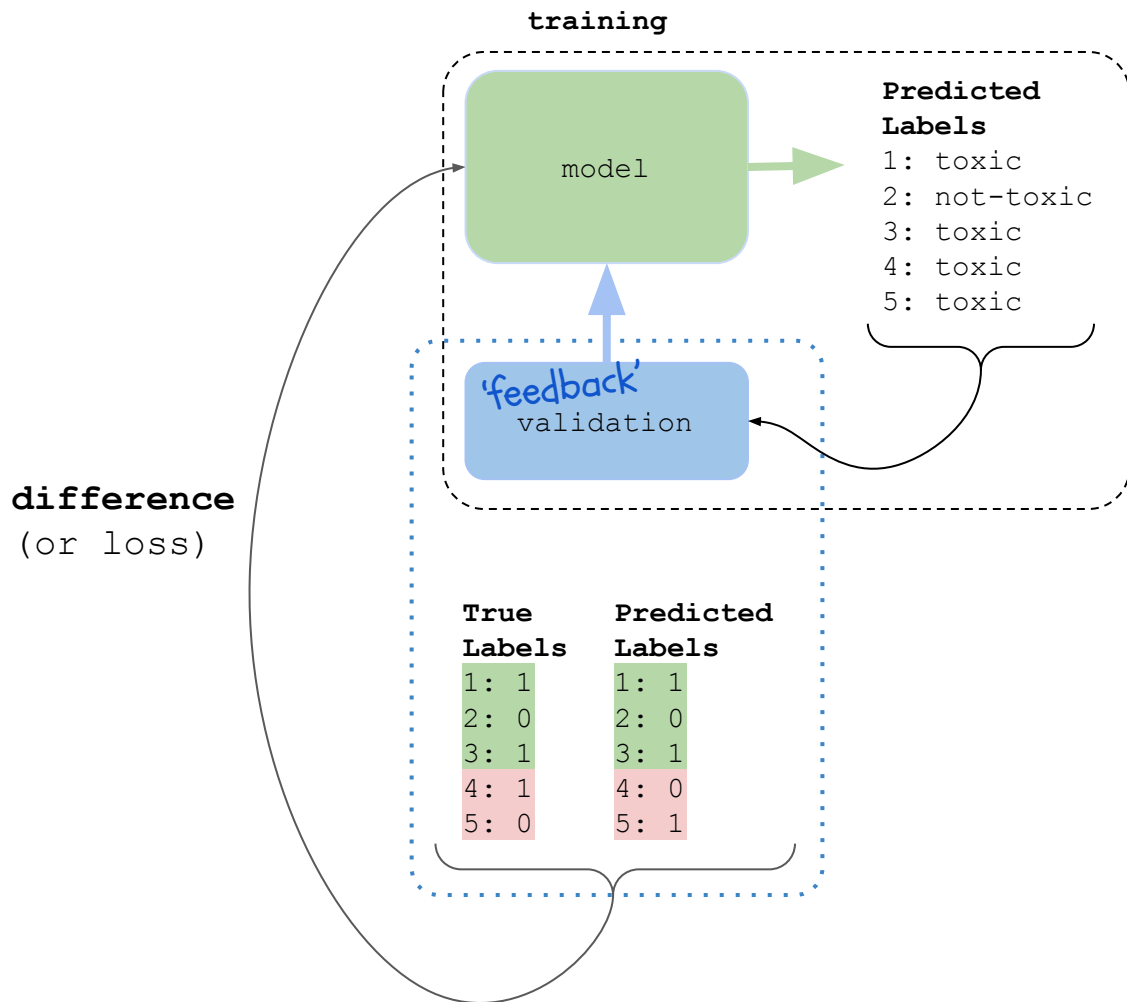


# Training a Machine Learning Model

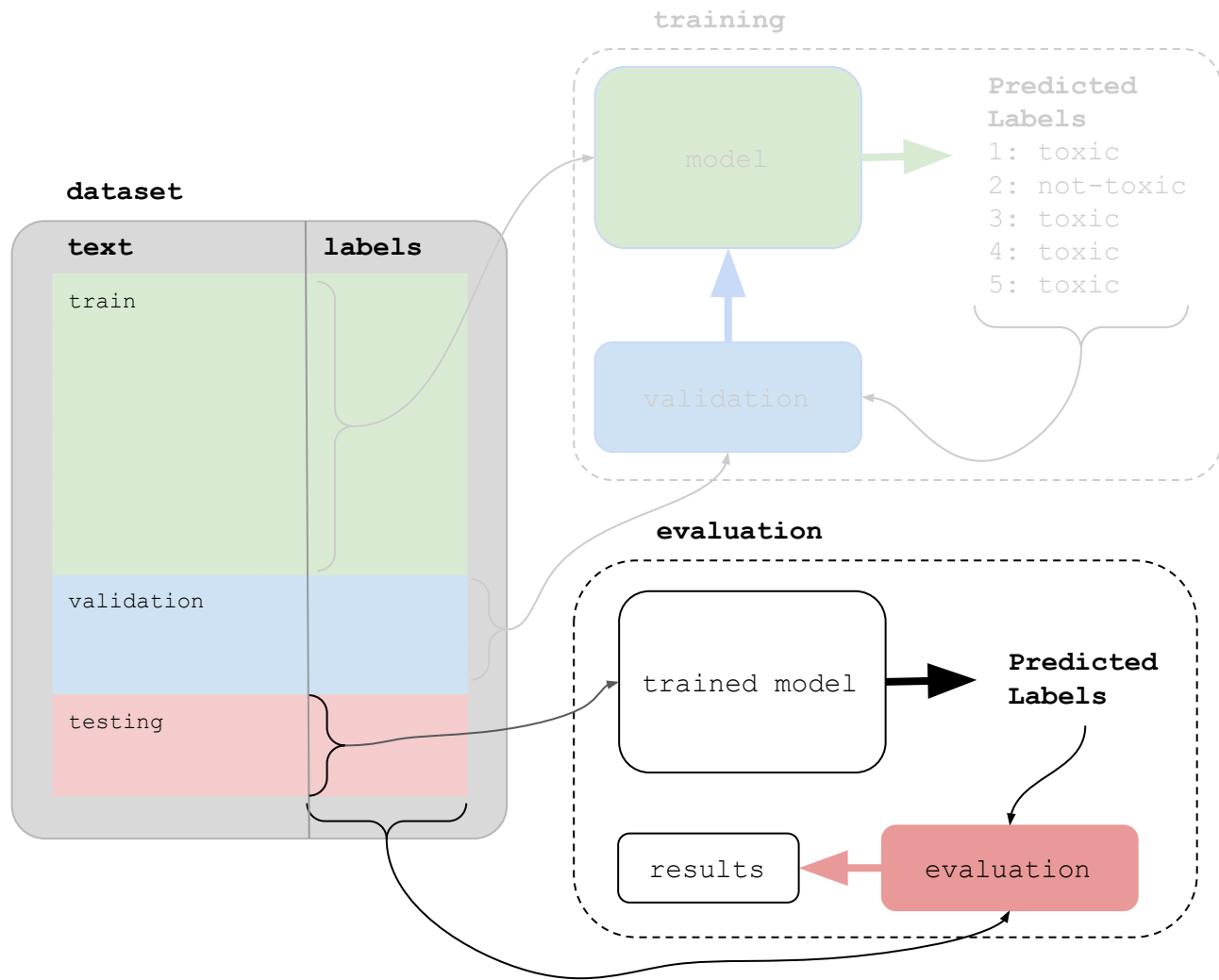


# Validation

The model improves its predictions based on the difference between the true labels and predicted labels.



# Evaluation



# Feature Representation

# The Inputs: Feature Representation

Can we just give  
the model text?

- We need to transform examples into something a machine can understand.
- We need to tell the machine what to look for, what the relevant aspects of the phenomenon are.

# The Inputs: Feature Representation

in other words:

- we need to turn each example into some sort of machine-readable summary of itself (choosing **relevant features**)
- → our examples must become **vectors of feature values**

**what *are* relevant features?**

## The Inputs:

- we know what we want to learn (**target class**):
  - for example:  or



## The Inputs:

- we know what we want to learn (**target class**):
  - for example: positive or negative
- we have a set of examples to learn from (**instances**)

# The Inputs:

- we know what we want to learn (**target class**):
  - for example: positive or negative
- we have a set of examples to learn from (**instances**)
- what clues might be useful to guess the class from the examples?
  - words in the text
  - types of words in the text (nouns, adjectives, adverbs, ...)
  - (time of) day
  - id of the twitter user
  - ...

clues → **features** (possible predictors)

observed occurrences → **feature values**

# Bag of Words

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...

# The Inputs: Feature Representation

a corpus

	day	improve	life	problems
"a day in the life of"	1	0	1	0
"day to day problems"	2	0	0	1
"day by day we improve"	2	1	0	0

document features

The diagram illustrates the process of converting text documents into numerical features. A corpus of three documents is shown on the left, each with its word counts for four specific terms: 'day', 'improve', 'life', and 'problems'. These counts are organized into a table on the right, where each row represents a document and each column represents a feature. The first document, 'a day in the life of', has counts of 1 for 'day', 0 for 'improve', 1 for 'life', and 0 for 'problems'. The second document, 'day to day problems', has counts of 2 for 'day', 0 for 'improve', 0 for 'life', and 1 for 'problems'. The third document, 'day by day we improve', has counts of 2 for 'day', 1 for 'improve', 0 for 'life', and 0 for 'problems'. The numerical counts are highlighted as 'document features'.

For a model to categorise text, we usually need to represent text documents numerically.

A rudimentary example of feature representation are word counts.

# Bag of Words

- Vector representation doesn't consider the ordering of words in a document
- *John is quicker than Mary and Mary is quicker than John have the same vectors*
- This is called the bag of words model.

# Term Frequency

- The term frequency  $tf_{t,d}$  of term  $t$  in document  $d$  is defined as the number of times that  $t$  occurs in  $d$ .
- We want to use  $tf$  when computing query-document match scores. But how?

# Term Frequency

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

# 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*.



# Tf-IDF

$$\text{idf}(W) = \log \frac{\#(\text{documents})}{\#(\text{documents containing word } W)}.$$

$$\text{tf-idf} = \text{tf} * \text{idf}$$

	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

# Target Function

# Target functions

## Classification

- Binary (e.g. logistic regression)
  - spam vs not-spam; sentiment analysis
- Multi-class (e.g. multinomial logistic regression)
  - natural language inference, genre detection

## Structure prediction

- Sequence labeling: POS tagging. segmentation
- Parsing: semantic & syntactic parsing

## Regression analysis:

- Scalar value (i.e., a measure)
- Linear is the simplest

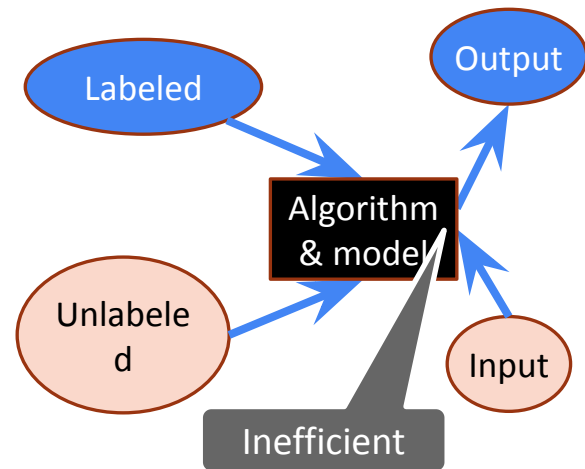
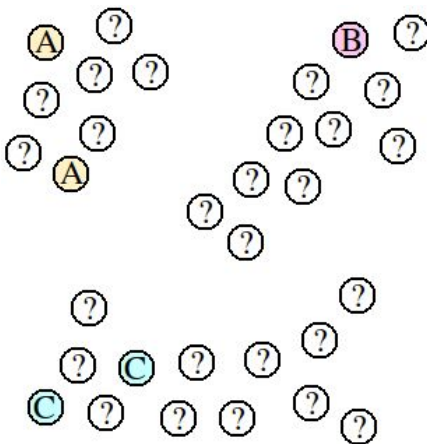
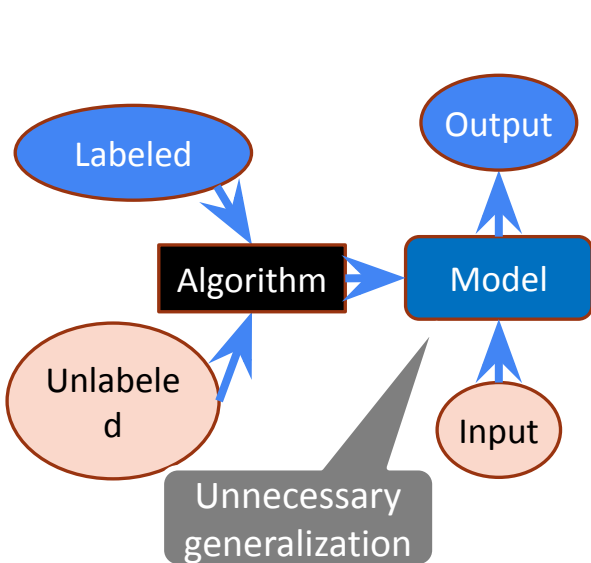


# Inductive & transductive learning



When solving a problem of interest, do not solve a more general problem as an intermediate step. Try to get the answer that you really need but not a more general one.

**Vladimir Vapnik**



# **Modeling in Machine Learning**

(Chapter 7)

Information gain & Entropy

Naive Bayes

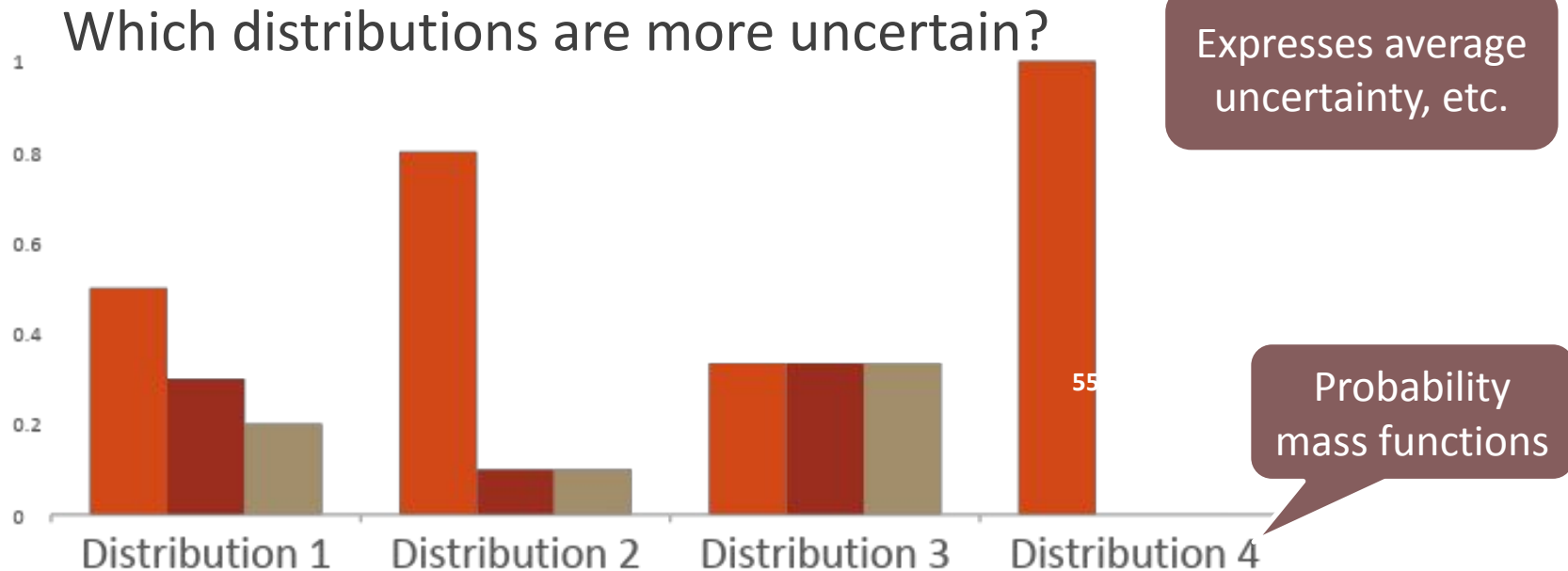
Decision Trees

# Understanding entropy

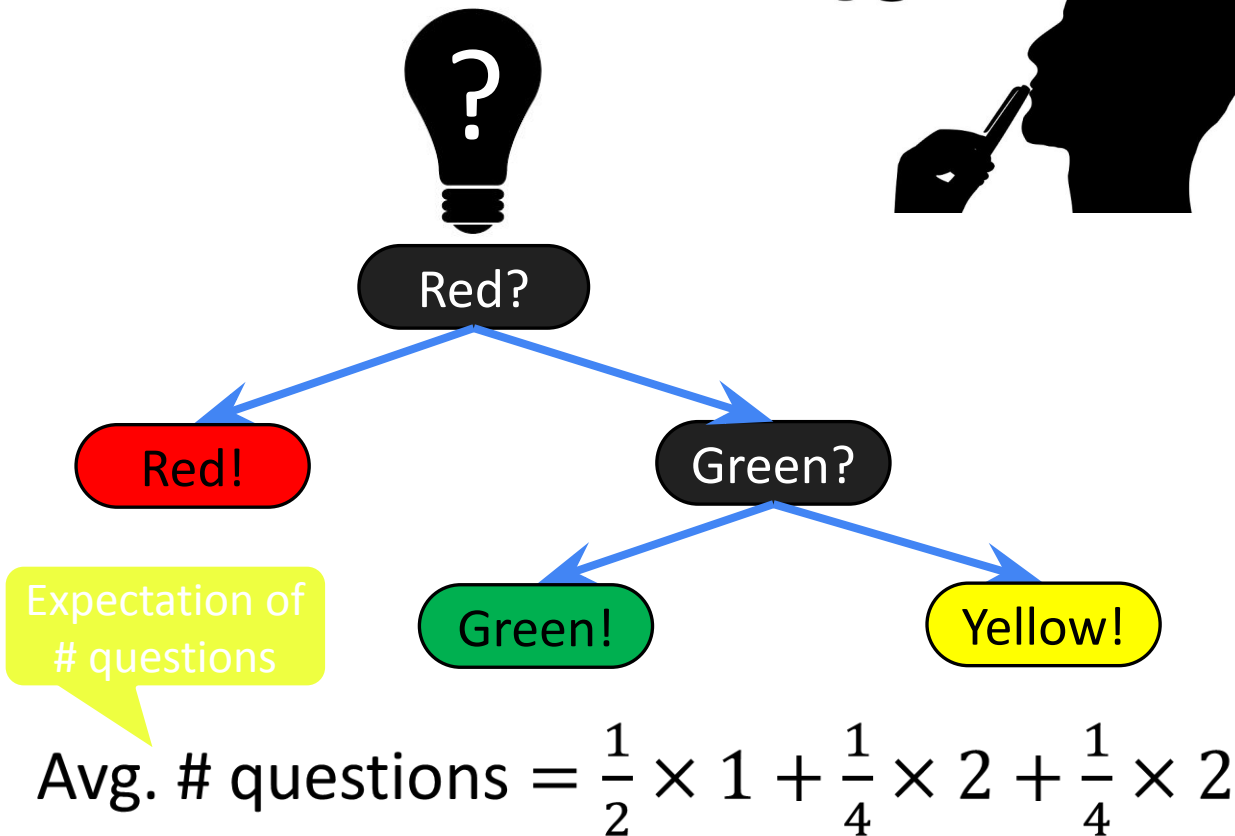
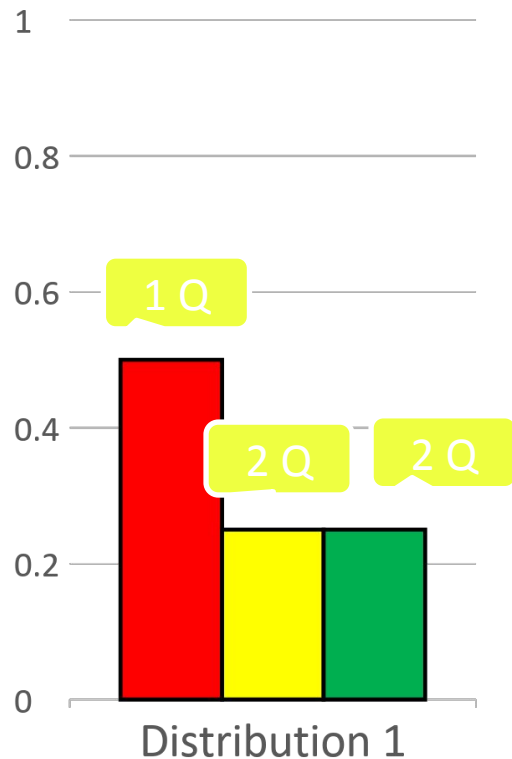
---

# Entropy

The measure of uncertainty, chaos, mess, and diversity



# Entropy: intuition





# Entropy: formula

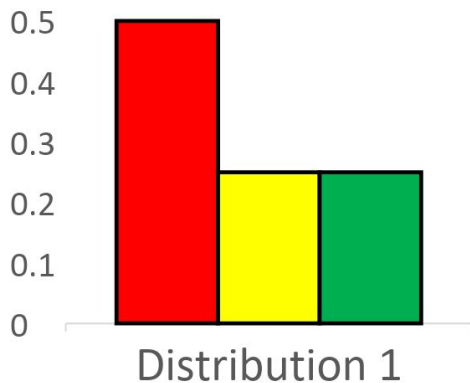
The entropy of a discrete random variable  $X$ :

$$H(X) = - \sum_{x \in V(X)} p(x) \log p(x) = \sum_{x \in V(X)} p(x) \log \frac{1}{p(x)}$$

Values of the random variable

Probability mass function

Base = 2  
(serves as a scale)



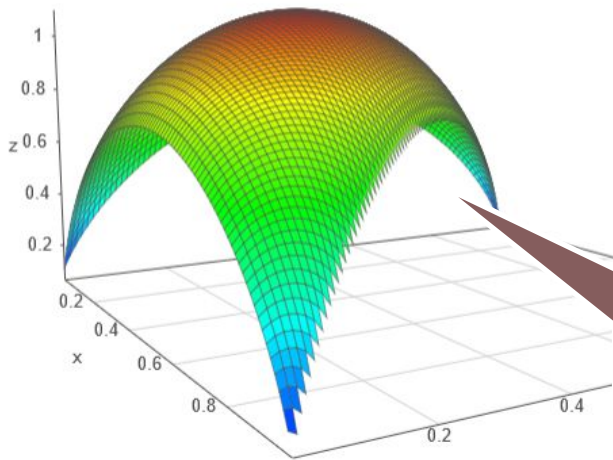
$H(X)$   $\equiv$

Entropy doesn't depend on the values of the random variable

$$\text{Avg. \# questions} = \frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{4} \times 2$$

# Entropy: two outcomes

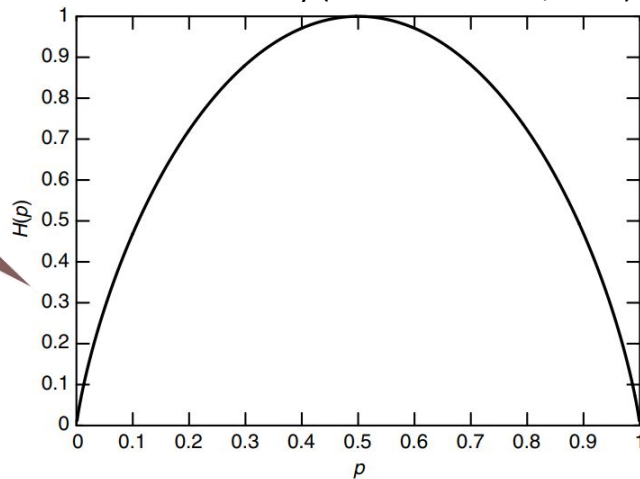
The entropy of a coin:  $X = \begin{cases} 1 & \text{with probability } p, \\ 0 & \text{with probability } 1 - p. \end{cases}$



$$0 \times \log 0 \stackrel{\text{def}}{=} 0$$

Entropy for a  
three-valued random  
variable

Elements of Information theory (Cover & Thomas, 2006)



# Naive Bayes

- simple classification method based on **Bayes rule**
- relies on a simple representation of documents: **bag of words**

```
x love xxxxxxxxxxxxxxxxxxxx sweet
xxxxxxxx satirical xxxxxxxxxxxx
xxxxxxxxxxxx great xxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxx fun xxxx
xxxxxxxxxxxxxxxx whimsical xxxx
romantic xxxx laughing
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx recommend xxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xx several xxxxxxxxxxxxxxxxxxxx
xxxxx happy xxxxxxxx again
xxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

great	2
love	2
recommend	1
laugh	1
happy	1
...	...

---

# Conditional probability

- conditional probability of an event: a probability obtained with the **additional information** that **some other event** has already occurred
    - new information is used to revise the probability of the initial event
  - **prior vs posterior probability**
    - prior: probability obtained “as things stand”, before any additional information is acquired
    - posterior: probability value which has been revised by using additional information
-

---

# An Example

in a corpus: happy tweets = 45% ; sad tweets = 55%

- I select one instance, how probable is it to be happy?
  - The tweet contains the word “cheerful”.
    - “cheerful” occurs in 70% of happy tweets, and in 25% of sad tweets.
    - Does the probability now change?
  - which one is prior and which one posterior?
-

---

# An Example

in a corpus: happy tweets = 45% ; sad tweets = 55%

- I select one instance, how probable is it to be happy?
  - The tweet contains the word “cheerful”.
    - “cheerful” occurs in 70% of happy tweets, and in 25% of sad tweets.
    - Does the probability now change?
  - which one is prior and which one posterior?
-

---

# An Example

in a corpus: happy tweets = 45% ; sad tweets = 55%

- I select one instance, how probable is it to be happy?
  - The tweet contains the word “cheerful”.
    - “cheerful” occurs in 70% of happy tweets, and in 25% of sad tweets.
    - Does the probability now change?
  - which one is prior and which one posterior?
-

---

# Bayes Theorem

$$p(c_j|i) = \frac{p(i|c_j)p(c_j)}{[p(c_j)p(i|c_j)] + [p(\neg c_j)p(i|\neg c_j)]}$$

$c_j$  : a given class (happy)

$i$  : a given instance (“I always feel cheerful on Friday evening”)

---



---

# Bayes Theorem

$$p(c_j|i) = \frac{p(i|c_j)p(c_j)}{[p(c_j)p(i|c_j)] + [p(\neg c_j)p(i|\neg c_j)]}$$

$p(c_j|i)$  = probability of instance  $i$  being in class  $c_j$

how likely is it this given tweet from the corpus is happy?

---

---

# Bayes Theorem

$$p(c_j|i) = \frac{p(i|c_j)p(c_j)}{[p(c_j)p(i|c_j)] + [p(\neg c_j)p(i|\neg c_j)]}$$

$p(i|\neg c_j)$  = pr of generating instance  $i$  given  $\neg c_j$  (sad) given  $\neg c_j$   
(sad) how likely is it to get  $i$ ?

*FALSE POSITIVE: 25%*

---

---

# Bayes Theorem

$$p(c_j|i) = \frac{p(i|c_j)p(c_j)}{[p(c_j)p(i|c_j)] + [p(\neg c_j)p(i|\neg c_j)]}$$

$p(i|c_j)$  = (likelihood function) = pr of generating instance  $i$  given class  $c_j$  (happy) given class  $c_j$  (happy) how likely is it to get  $i$ ?

TRUE POSITIVE: 70%

---

---

# Working Example

p “cheerful” in of happy tweets

p Happy

$$p(c_j|i) = \frac{0.70 \cdot 0.45}{[0.45 \cdot 0.70] + [0.55 \cdot 0.25]}$$

p Sad

p “cheerful” in of sad tweets

- $c_j$ : happy
- $i$ : “I always feel cheerful on Friday evening”

- $p(i|c_j)$  = given class  $c_j$  (happy) how likely is it to get  $i$ ?
- $p(i|\neg c_j)$  = given class  $\neg c_j$  (sad) how likely is it to get  $i$ ?

---

# A Classifier

- How do we make a classifier out of this?

$$p(c_j|i) = \frac{p(i|c_j)p(c_j)}{p(i)}$$

---

# A Naive Bayes Classifier

- How do we make a classifier out of this?
- We add a decision rule:  
maximum a posteriori (map)

$$p(c_j|i) = \frac{p(i|c_j)p(c_j)}{p(i)}$$

$$c_{map} = \arg \max_{c \in C} \frac{p(i|c) \cdot p(c)}{p(i)}$$

---