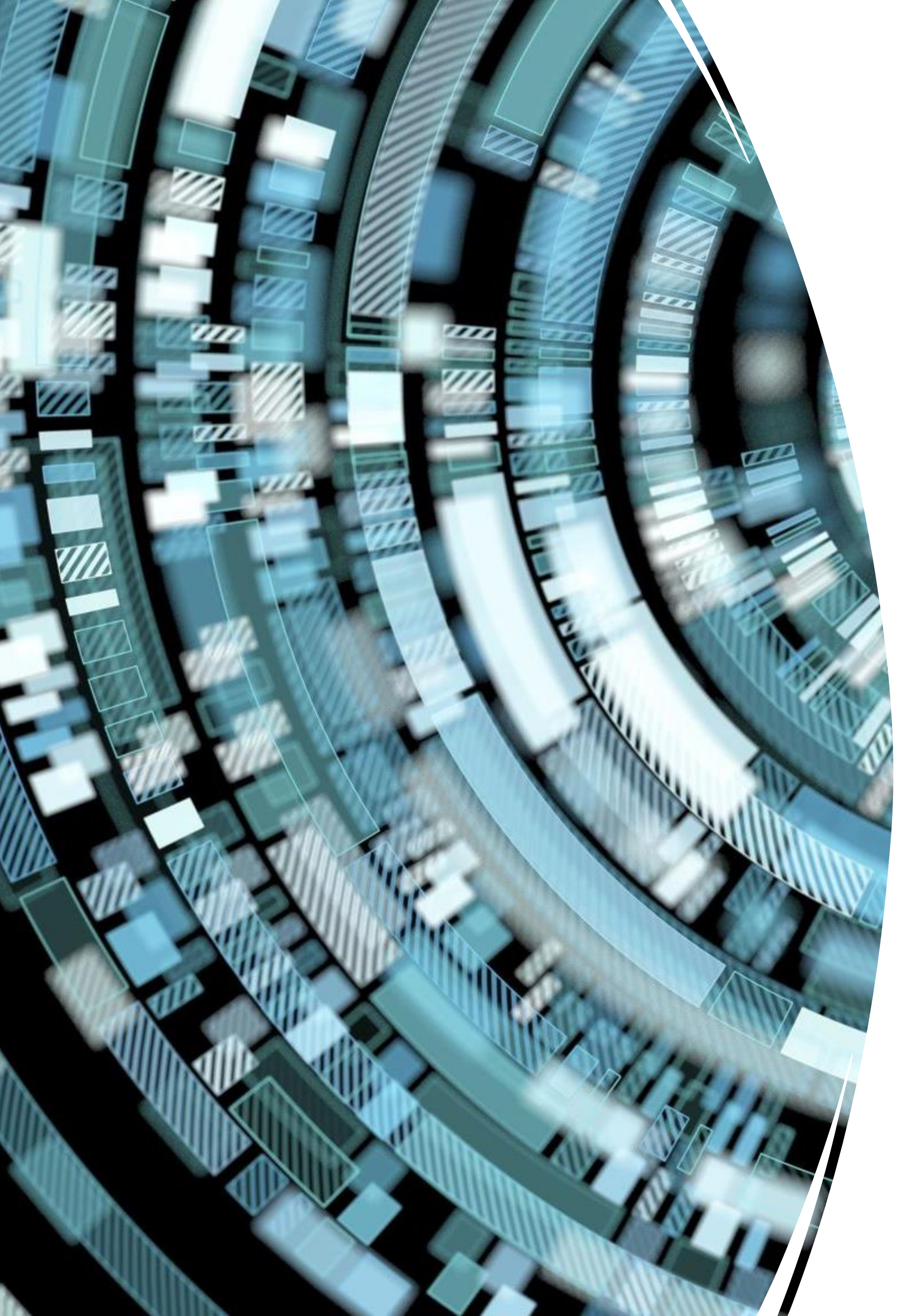
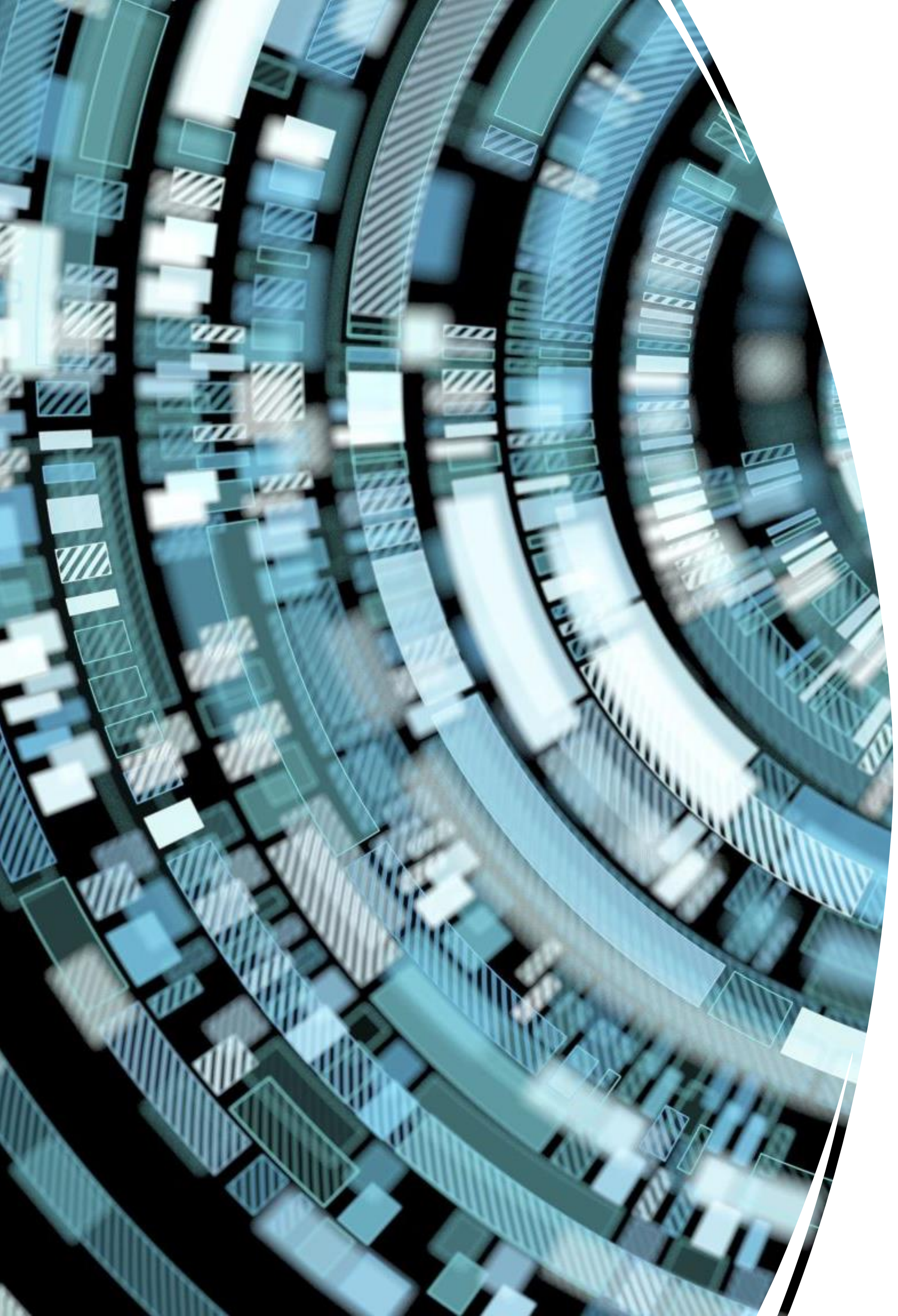
NLP - ML Classification

Liad Magen



Machine Learning

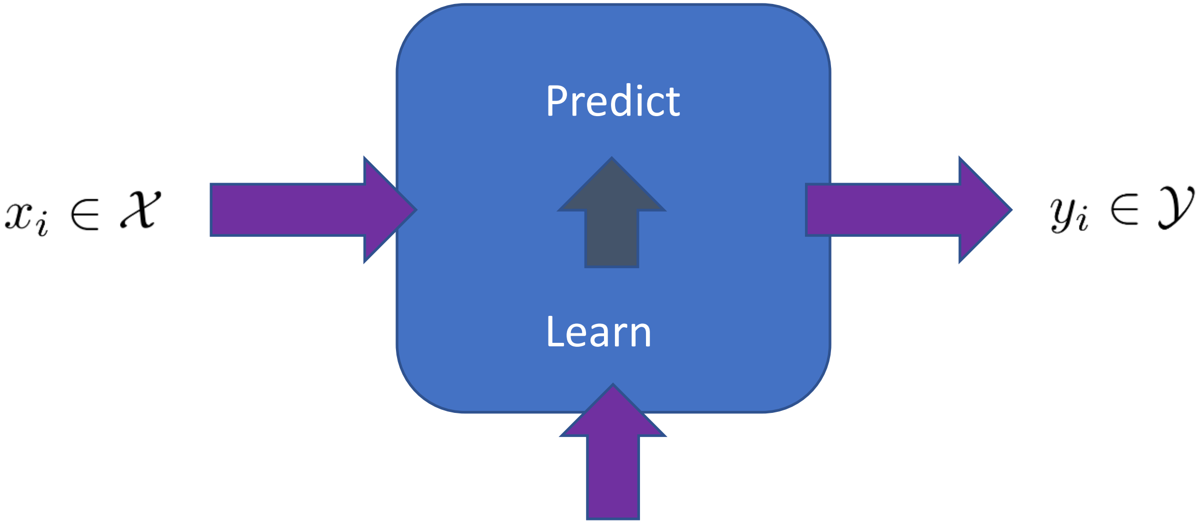
- "Learn from data"
  - Supervised
  - Unsupervised
  - Semi-Supervised
  - Few-shots / Weakly Supervised



Machine Learning

- "Learn from data"
  - Supervised
    Labeled examples
  - Unsupervised
    No Labeled Examples
  - Semi-Supervised
    Labeled Examples from non-labeled Examples
  - Few-shots / Weakly Supervised
    Few Labeled Examples

Classification with ML



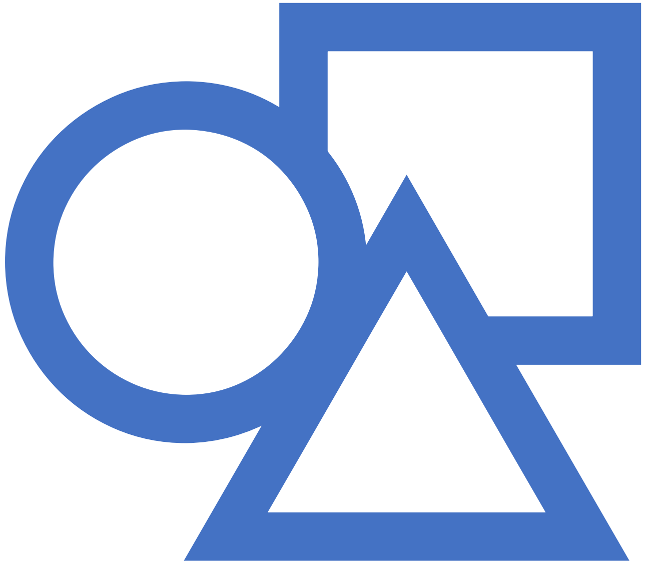
Data

$$\{(x_i, y_i) | x_i \in \mathcal{X}, y_i \in \mathcal{Y}\}_{i=1}^{|D|}$$

Classification with ML

- We are given data samples: $x_1, x_2, \ldots, x_n \qquad x_i \in \mathcal{X}$
- And their corresponding labels $y_1, y_2, \ldots, y_n \qquad y_i \in \mathcal{Y}$
- We train a function f: $f : x \in \mathcal{X} \rightarrow y \in \mathcal{Y}$
- The data-point x is represented by 'features': $f : \phi(x) \in \mathbb{R}^m \rightarrow y \in \mathcal{Y}$

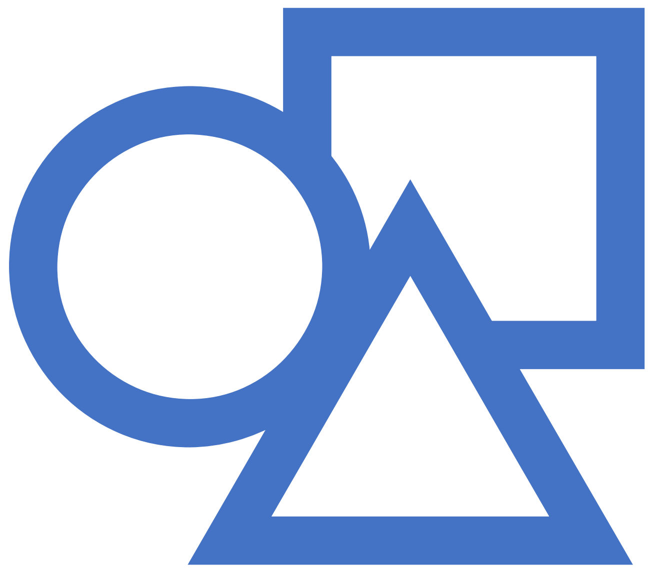
Feature Function



Feature Function

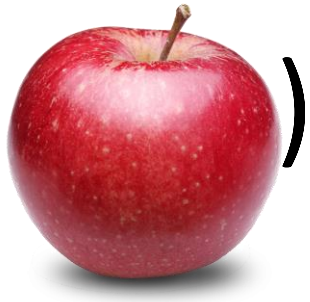
- How do we represent an object?

f()



Feature Function

- Perform **measurements** and obtain **features**

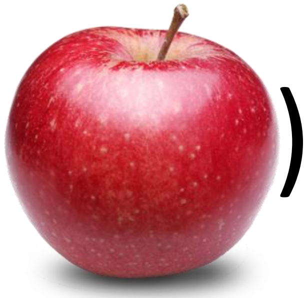
$\varphi($  $)$

= (1.3 , 34 , 8.2 , #ff0000)

(Diameter, weight, softness, color)

Feature Function

- Perform **measurements** and obtain **features**
- **Indicator** features / **1-hot** vector / **binary** features

φ ()

Buckets: (0-1, 1-2, 2-3)

= (0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0 ,0 ,0)
(Diameter, weight, softness, color)

Feature functions for text?

Your Turn:

What can we measure over text?

Types of Classification Problems

- Binary $$y \in \{-1, 1\}$$
- Multi-Class $$y \in \{1, 2, ..., k\}$$
- Multi-Label $$y \in 2^{\{1,2,\ldots,k\}}$$
- (Regression...?)

Types of classifiers

- Generative vs Discriminative
- Probabilistic vs Non-Probabilistic
- Linear vs non-Linear

$$\mathrm{P}(x, y)$$

$$\mathrm{P}(y|x)$$

$$\mathrm{score}(x, y)$$

$$f(x) = y$$

Types of classifiers

- Generative vs Discriminative
- Probabilistic vs Non-Probabilistic
- Linear vs non-Linear

$\mathrm{P}(x, y)$ generative

$\mathrm{P}(y|x)$ Discriminative

$\mathrm{score}(x, y)$ Discriminative

$f(x) = y$ Discriminative

Types of classifiers

- Generative vs Discriminative
- Probabilistic vs Non-Probabilistic
- Linear vs non-Linear

prob	$\mathrm{P}(x, y)$	generative
prob	$\mathrm{P}(y \mid x)$	Discriminative
Non-prob	$\mathrm{score}(x, y)$	Discriminative
Non-prob	$f(x) = y$	Discriminative

Popular Classifiers

- kNN (k-Nearest Neighbors)
- Decision Trees
  - Decision Forests
  - Gradient-boosted Forests
- Logistic Regression
- SVM
- Neural Networks

Scikit-learn (sklearn): a popular and good package for those activities

Test your memory - what are:

- Training set // development set // test set
- Loss function
- Overfitting
- Regularization
- Evaluation metrics

Generic NLP Solution

- Find an annotated corpus
- Split it into train/dev & test parts
- Convert it to a vector representation
  - Decide on the output type
  - Decide on the features
  - Convert each training example to a feature vector
- Train a machine learning model on the training set
- Apply your model on the test-set
- Measure the accuracy

Generic NLP Solution

- **Find an annotated corpus**
  - Difficult to create your own corpus (expensive)
  - Decide what are you classifying?
    What should the output classes be?
  - Consider: is the problem even solvable?
    Can humans do that?
    At what level of accuracy can humans do it?

Classification of Language Data

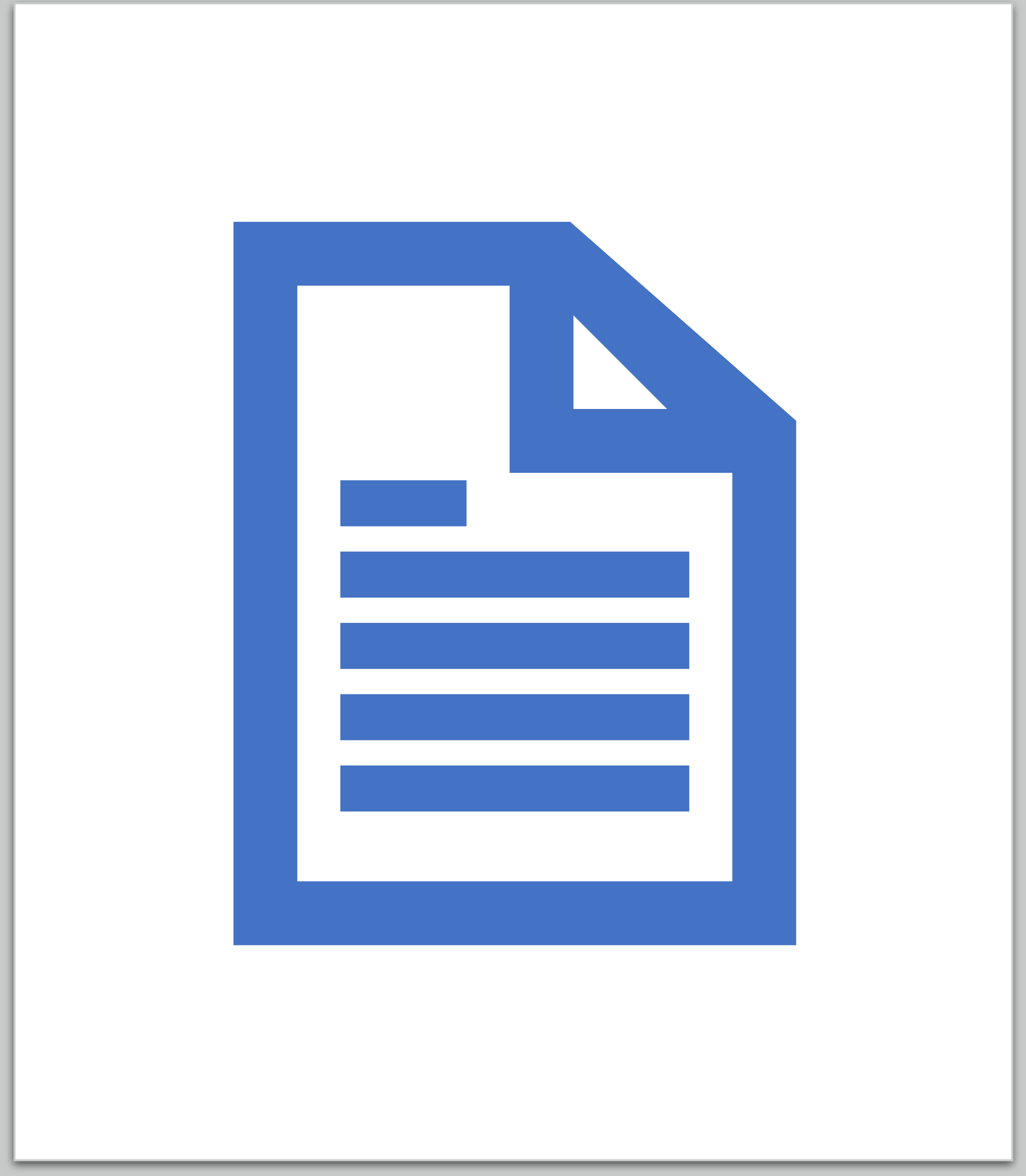
What are some things we would like to classify?

and to what categories?

Let's examine it by the basic units of processing.

What do we process?

- A Collection of Documents
  - By date or domain
- Document
- Section
  - Subsection
- Paragraph
- Sentence
  - Phrases
- Word
  - Characters / Morphemes



Document Labeling examples?

Document Labeling examples:

- Language classification
- Topic classification
- Author classification
- Sentiment classification
- Interestingness level
- Relevance level (recommendation systems)
- Spam Detection

Binary?
multi-class?
multi-label?

Classification
or Ranking
(score)?

- Language classification
- Topic classification
- Author classification
- Sentiment classification
- Interestingness level
- Relevance level (recommendation systems)
- Spam Detection

Sentence Labeling examples

- Mostly same as document classification – but shorter text = harder task!
- What tasks are relevant for *sentences* but not for *documents*?
- Sometimes a document problem is a sentence problem. When?

Types vs Tokens

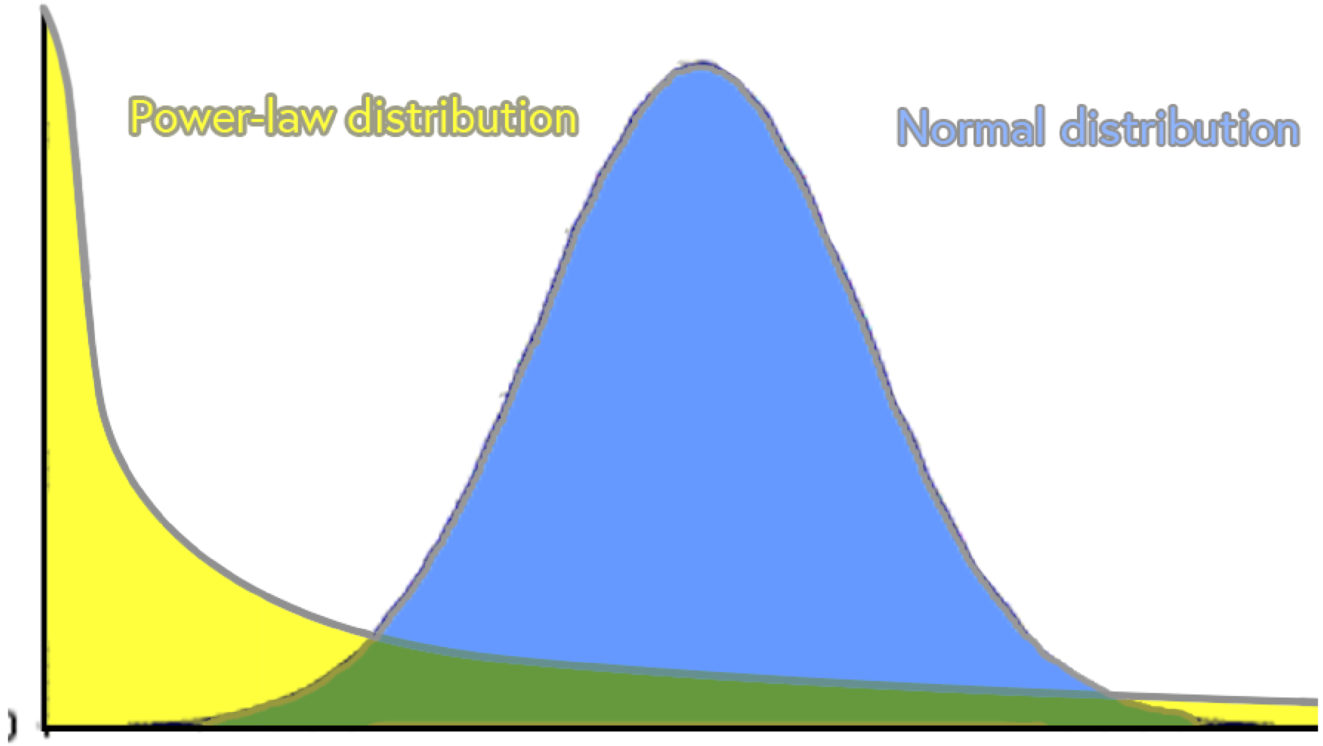
"To be or not to be"

- 6 Tokens
- 4 Types

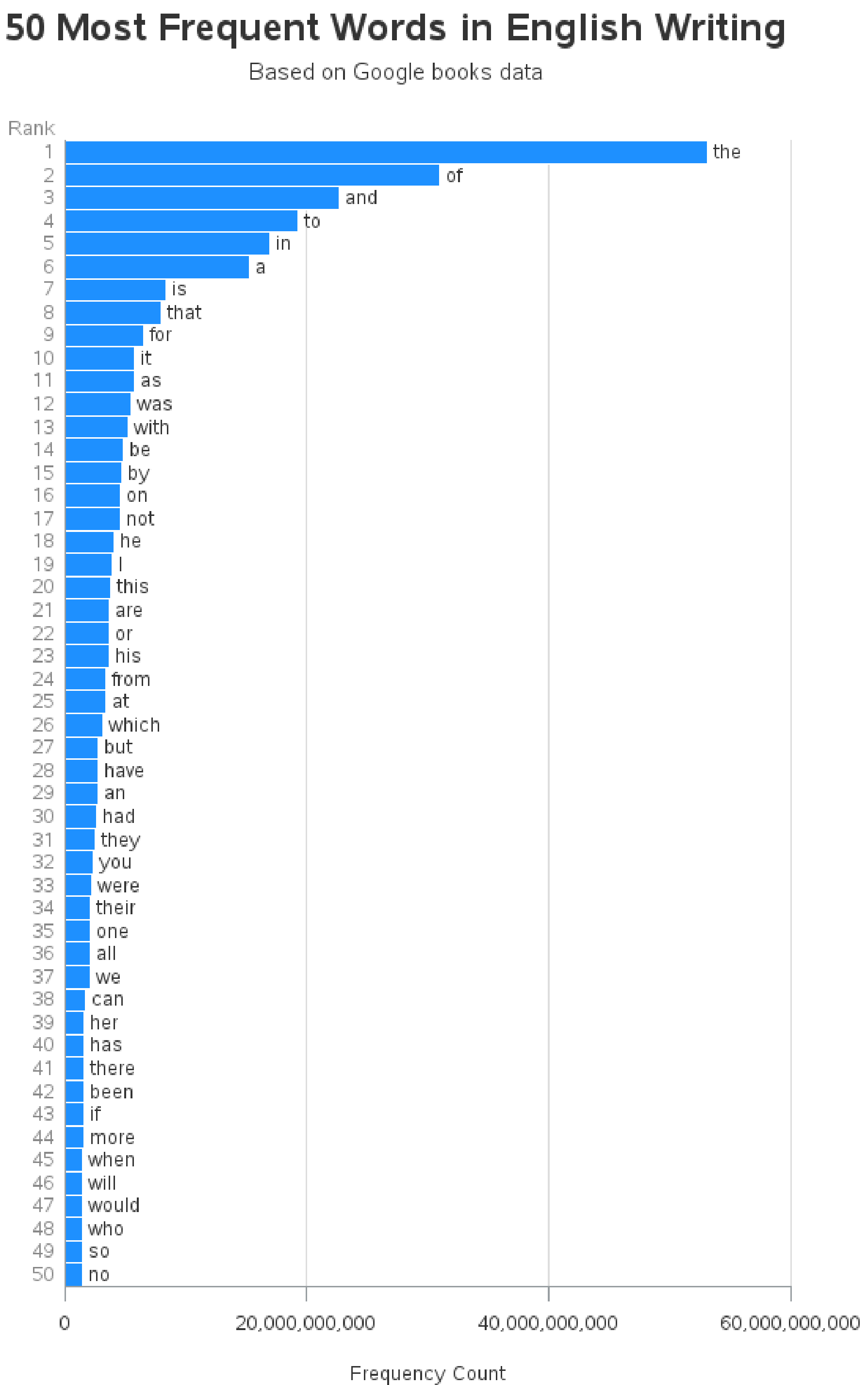
Word level Labeling

- Classification of **tokens** vs **types**:
  - Labeling **types**:
    - Nouns vs. Adjectives
    - Happy vs. Sad
      positive vs negative words
  - Labeling **Tokens**:
    - Sentence boundary detection
    - Spelling mistakes ("then" instead of "than")

Language data properties



- Zipf Law
- Word frequencies follow a "power-law distribution":
  - **Long tail**
  - Most events rarely occur



Zipf Law

frequency of a word is inversely proportional to its rank in the frequency table

$$n(r) \propto \frac{1}{r^z} \qquad z \approx 1$$

Zipf Law

In a 43M words text, there are:

- 316,710 unique words (types)
- 144,999 words occur only once
- 42,525 words occur 2 times
- 21,618 words occur 3 times
- 13,306 words occur 4 times
- 9,488 words occur 5 times
- 26,024 words appear >50 times

Zipf Law

No matter how large the training corpus is:

- It's likely to contain previously unseen word forms
- There will be many previously unseen word-*pairs*
- There will be even more previously unseen word-*triplets*
- There will be even more previously unseen *sentences*

Summary pause – 1m

SUMMARIZE IN FEW SENTENCES YOUR MAIN TAKE-AWAYS.

IT’S OK, I’LL WAIT ;)

Properties of Language Data

Sparse

- Zipf Law
- Variability

Symbolic

- Abstract symbol to meaning mapping
- Variability
- Ambiguity

Many levels of granularity:

- Document, paragraph, sentence, word, character

How would these affect a classifier over text data?

Representing Text as Features

Representing text as features

- **Indicator features** over **events in the data**
  (counts)   words, characters, n-grams, lemmas, stems

Pre-processing

- **Tokenization**
  punctuations could be considered as tokens, too (up to us to decide).

  To be , or not to be ???

  New-York

Representing text as Features

"to be or not to be"

Bag of Words (BoW) - word counts:

{ "be": 2, "to": 2, "not": 1, "or": 1, "something": 0, "else": 0 }

(0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, … ,1, 0, 1, 0, …. ,0)
('a', 'the', 'there', 'to', …, 'be', 'if', 'an' … , 'or', 'as', 'not'…, 'zebra')

Representing text as Features

"to be or not to be"

Bag of words (BoW) -word counts:

{ "be": 2, "to": 2, "not": 1, "or": 1, "something": 0, "else": 0 }

Reweighting:
TF/IDF or Pointwise Mutual Information – PMI
{ "be": 1.2, "to": 0.1, "not": 0.9, "or": 0.3 }

TF / IDF + PMI

{ "be": 2, "to": 2, "not": 1, "or": 1 }

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

$$PMI(a,b) = \log\left(\frac{P(a,b)}{P(\mathrm{a})\mathrm{P}(\mathrm{b})}\right)$$

N-grams

["to", "be", "or", "not", "to", "be",

"to be", "be or", "or not", "not to", "to be",

"to be or", "be or not", "or not to", "not to be"]

When are n-grams useful?

What are potential problems with n-grams?

If I have bigrams, do I still need unigrams?

Why not use 4-grams of 5-grams?

N-gram example

- The soup was not too bad
- "Not too bad" as a 3-gram has a special meaning

Dictionary-based features (preprepared lists)

- If we have word-lists, might as well use them
  I.e., lists of cities/countries.
- How many times a "negative word" appears?
- How many times a name of a city in Austria occurs?

Pre-processing: stemming / Lemmatizing

- Lemma: the "*dictionary word*" of a word

create, created, creating, creator, creativity:
create, create, create, creator, creativity

- Stem: a "*base form*", based on heuristics

create, created, creating, creator, creativity: **creat**

Pre-processing: stemming / Lemmatizing

- Lemma: the "dictionary word" of a word

create, created, creating, creator, creativity
create, create, create, creator, creativity

- Stem: a "base form", based on heuristics

create, created, creating, creator, creativity
creat creat creat creat creat

Q: When are stemming/lemmatizing helping?
Which one is better?
What are the downsides of each?

Was it a lot?

Take a minute to write down the 10 most important things you've just learned.

Document Classification

- Bag-of-words (+ Linear Classifier)

  often surprisingly effective (especially for longer documents).

- Re-weighting (TF/IDF, or PMI) often helps.

- Lemmatization/stemming sometimes helps.

- Dictionaries can be useful, if available.

Beyond bag-of-words

- **Indicator features** over **events in the data**

counts **words, chars, n-grams, lemmas, stems...**

Recall the sentence-boundary detection problem.

What kind of classification is this?

What could be possible features for it?

Beyond Classification

Classification is very common, but it is not always enough.

When it is not, we do something more complex (combining classifiers, structured output, etc).

The big questions

Is the problem even solvable?
Can humans do it?
At what level of accuracy can humans do it?
At what level of agreement can humans do it?

- How do I represent this as an ML problem? and which ML problem?
- How do instances look like?
- What are good features?
- Where / how do I get data?
- How do I evaluate?