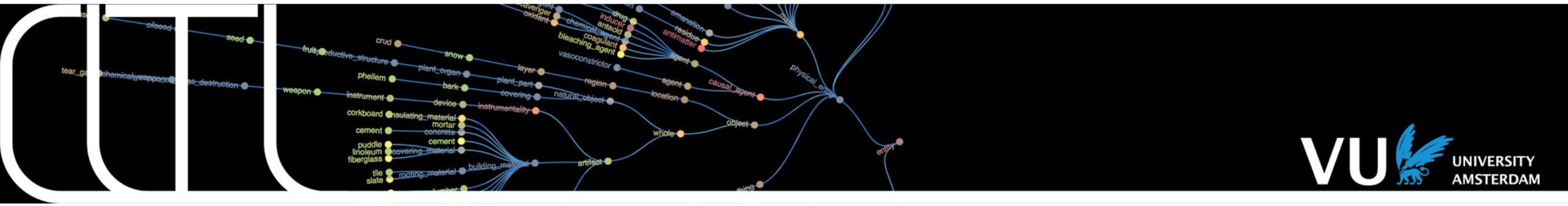
Text Mining CBS

Lecture 2: Machine Learning for NLP

Piek Vossen

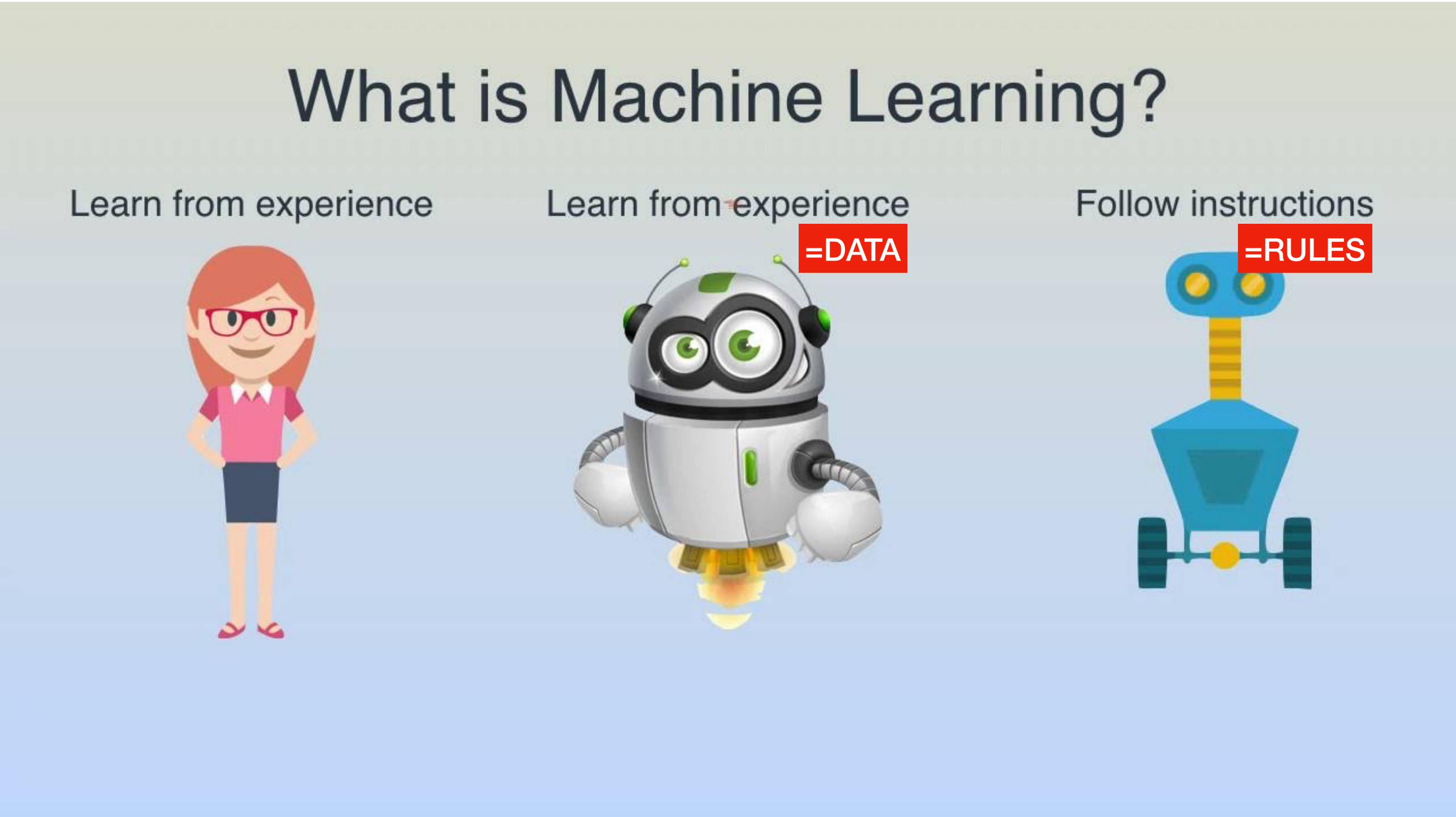


VU UNIVERSITY AMSTERDAM

Overview

- Part I: Humans against machines
- Part II: Supervised machine learning
- Part III: Deep learning
- Part IV: Evaluation

Humans against Machines



NLP Approaches

- **Rule-based:** tell the machine exactly what to do under specific circumstances

- **Machine learning**: the machine learns to associate patterns with interpretations
    - Supervised machine learning: *the machine uses labelled examples to learn*
    - Unsupervised machine learning: *the machine identifies patterns without using labelled examples*

- **Hybrid** = a combination of (un)supervised machine learning and rule-based approaches

Rules and lexicons!

- **lexicon** that lists words with properties, e.g. sentiment scores for positive (good, excellent) or negative sentiment (bad, worthless):
  - ***Ambiguity*** (***low*** *prices* versus ***low*** *ceilings*) analyse context
  - **Other issues: negation** "not friendly but clear", **intensifiers** "very friendly"

- **Examples of lexical and rule-based analysis:**
  - Sentiment: VADER (for tweets); https://github.com/cjhutto/vaderSentiment
  - Emotions: NRC (emotions expressed by words): https://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm
  - Psychological features: LIWC http://liwc.wpengine.com

VADER lexicon

Valence Aware Dictionary and sEntiment Reasoner

TOKEN, MEAN-SENTIMENT-RATING, STANDARD DEVIATION, and RAW-HUMAN-SENTIMENT-RATINGS [10 raters -4, 0, +4]

```
arrogance   -2.4    0.66332 [-3, -2, -2, -3, -1, -2, -3, -3, -2, -3]
arrogances  -1.9 0.53852[-1, -2, -3, -2, -2, -2, -2, -1, -2, -2]
arrogant-2.2 0.6[-2, -2, -2, -3, -2, -3, -3, -2, -1, -2]
arrogantly  -1.8 1.4[-3, -2, 2, -3, -1, -3, -2, -2, -2, -2]
ashamed -2.1 1.3[-3, -3, -3, -2, -2, 1, -2, -4, -1, -2]
ashamedly   -1.7    0.64031 [-2, -2, -2, -1, -2, -3, -1, -2, -1, -1]
ass -2.5 1.43178[-4, -1, -2, -1, -3, 0, -2, -4, -4, -4]
```

😀 grinning face

😁 beaming face with smiling eyes

😂 face with tears of joy

🤣 rolling on the floor laughing

😃 grinning face with big eyes

😄 grinning face with smiling eyes

😅 grinning face with sweat

😆 grinning squinting face

😉 winking face

Available as module in NLTK

NRC emotion lexicon annotated by the crowd

https://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm

- beating anger 1
- beating anticipation 0
- beating disgust 0
- beating fear 1
- beating joy 0
- beating negative 1
- beating positive 0
- beating sadness 1
- beating surprise 0
- beating trust 0

- betrayal anger 1
- betrayal anticipation 0
- betrayal disgust 1
- betrayal fear 0
- betrayal joy 0
- betrayal negative 1
- betrayal positive 0
- betrayal sadness 1
- betrayal surprise 0
- betrayal trust 0

LIWC (Linguistic Inquiry and Word Count)

- Analyses text using lexicons for ***90***(!) dimensions such as:
  - positive & negative emotions
  - social words (e.g. family related vocabulary)
  - cognitieve processes (insight, causation, certainty)
  - authority
- Analyses text properties such as punctuation, sentence length, closed class words, etc.

http://liwc.wpengine.com/results/

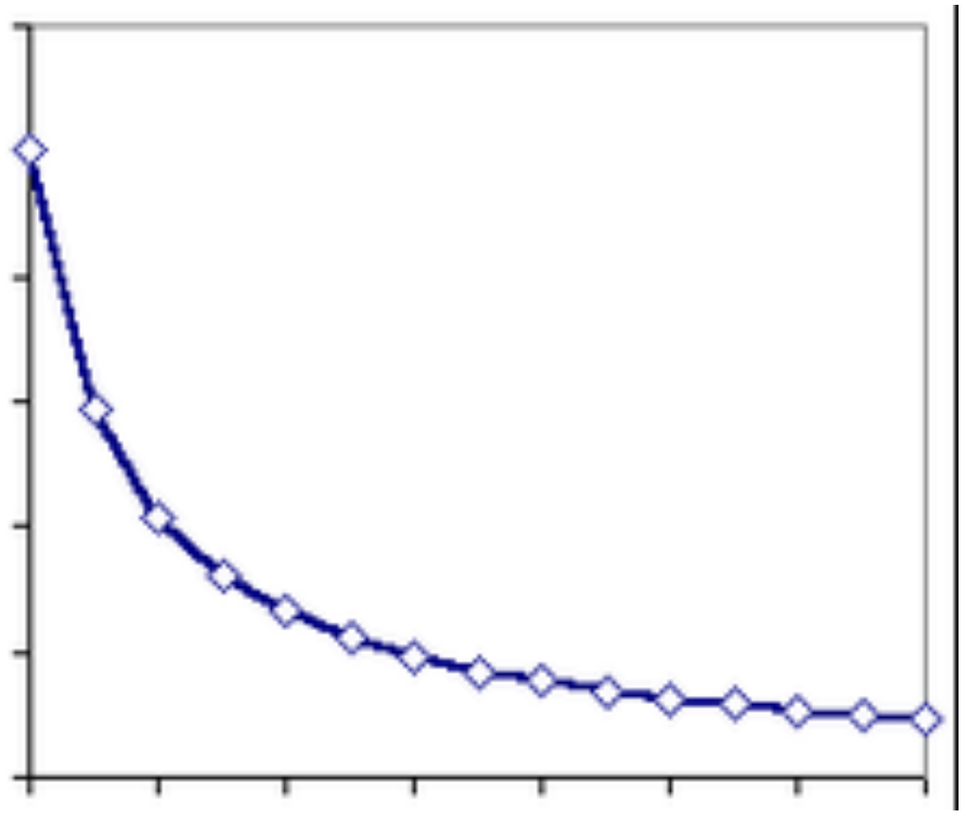
At least 26 people were killed in Sunday's church shooting in Sutherland Springs, Texas, Gov. Greg Abbott said at a press conference. About 20 others were wounded, said Freeman Martin, a regional director with the Texas Department of Public Safety, with victims ranging in age from 5 to 72 years old. Among the dead is the 14-year-old daughter of the First Baptist Church's pastor, Frank Pomeroy, according to his wife, Sherri Pomeroy, the girl's mother. The couple were traveling out of state when the shooting occurred. Authorities have not said what may have motivated the suspected shooter, who was later found dead in his vehicle. The shooting has devastated the small Texas town east of San Antonio, described as a place where "everybody knows everybody."

TRADITIONAL LIWC DIMENSION	YOUR DATA	AVERAGE FOR
I-WORDS (I, ME, MY)	0.0	
SOCIAL WORDS	11.0	
POSITIVE EMOTIONS	0.8	
NEGATIVE EMOTIONS	2.4	
COGNITIVE PROCESSES	7.1	
SUMMARY VARIABLES		
ANALYTIC	97.8	
CLOUT	80.7	
AUTHENTICITY	56.9	
EMOTIONAL TONE	7.5	

Formulating precise rules

- Easy to get some results quickly but impossible to cover all ways of expressing information

- Law of diminishing returns:
  - $Effort(X)^N$
  - N = 1 → 50% recall
  - N = 2 → 60%
  - N = 3 → 65%
  - N = 4 → 68%



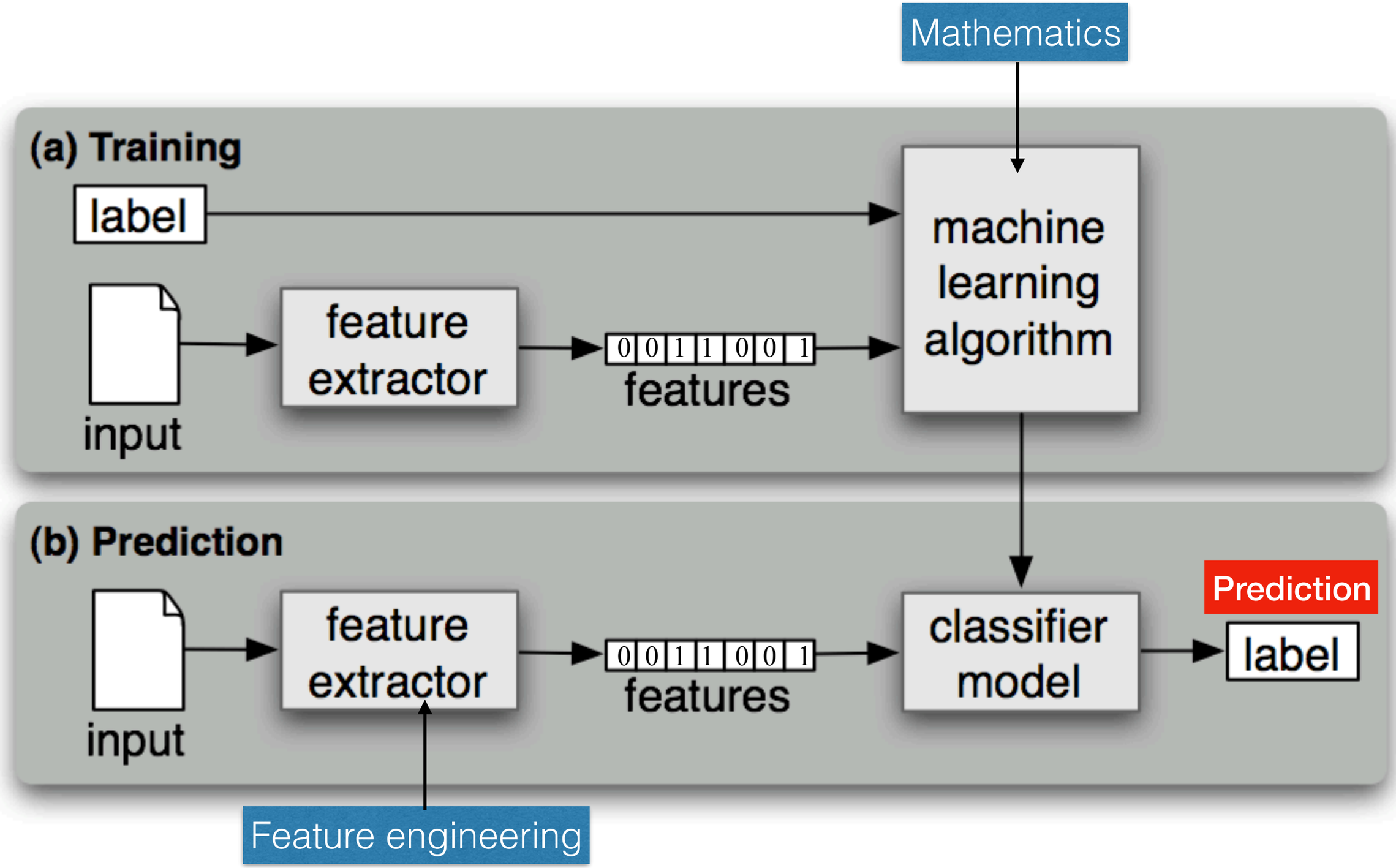
Why machine learning?

- Hand-crafted Models (e.g. grammars) failed empirical test on real data —> variation and dynamics of language is far bigger than realised

- Rules became too complex and we need too many,
  - making it impossible to maintain systems
  - making it psychologically unrealistic: what child learns these rules?

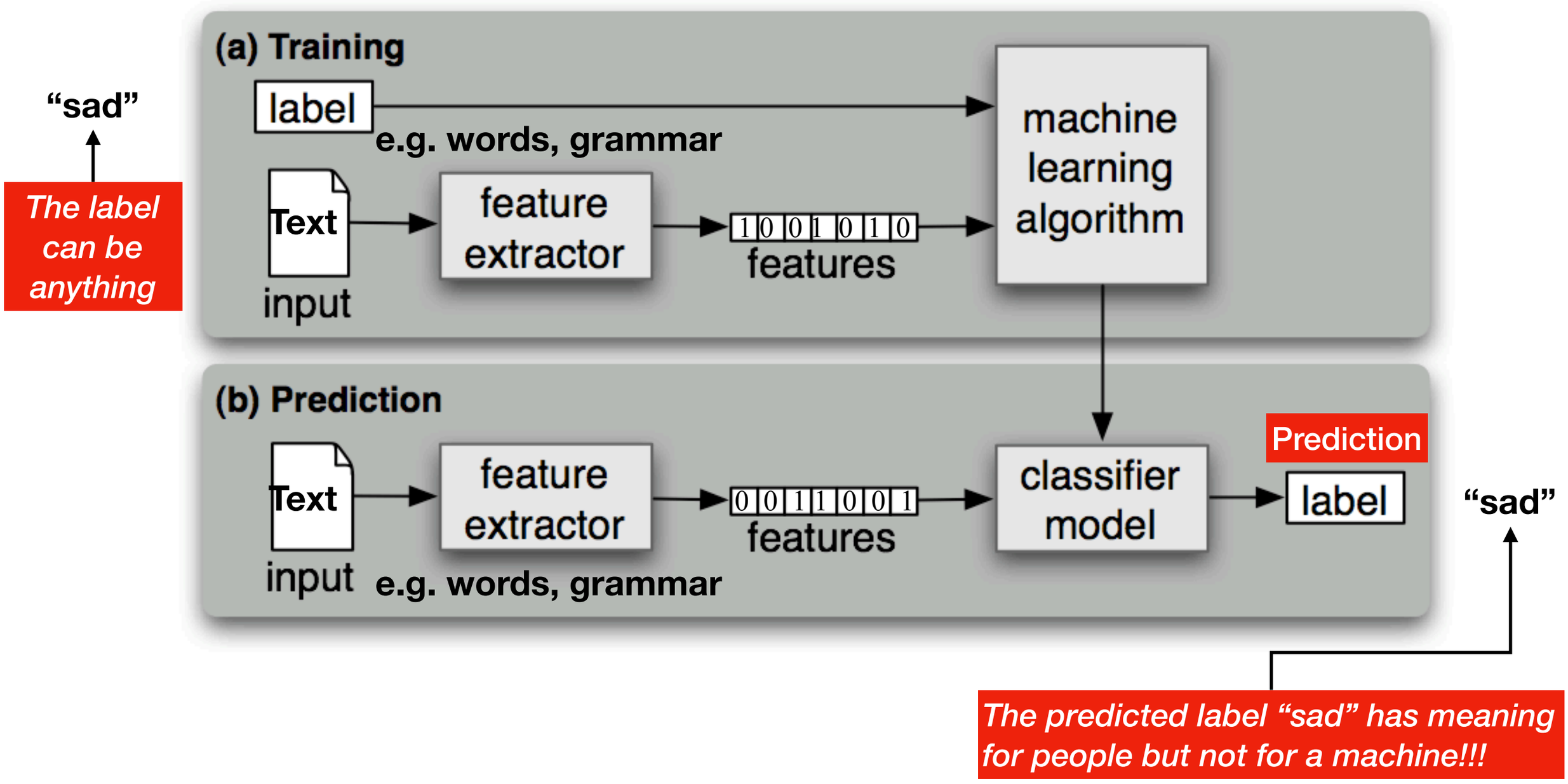
- Machine learning appears to work better than any set of the rules we can invent

Supervised Machine learning



a lot of

Machine Learning needs data!



"sad"

The label can be anything

"sad"

The predicted label "sad" has meaning for people but not for a machine!!!

Supervised machine learning systems

- **Training corpus:** Require human annotation of text with word senses (training corpus)

- **Tag set:** different from task to task, e.g. sentiment, part-of-speech tagging, syntactic parsing, entity detection, word-sense-disambiguation

- **Feature selection:** preprocess the training text by extracting a set of features for each text and probability values

- **Classifier:** match features from training set to feature of target

NLP Datasets

- **Scientific competitions**: MUC (Message Understanding Conference, TREC (information retrieval), CLEF (crosslingual retrieval), CoNNL (natural language learning), Senseval (word sense detection), Semeval (semantic tasks), ACE (Automatic Content Extraction)

- Machine learning community: **Kaggle**

  - https://www.kaggle.com/rtatman/sentiment-lexicons-for-81-languages

- Some **blogs**:

  - http://hatespeechdata.com

  - https://github.com/niderhoff/nlp-datasets

  - https://lionbridge.ai/datasets/the-best-25-datasets-for-natural-language-processing/

  - https://blog.cambridgespark.com/50-free-machine-learning-datasets-natural-language-processing-d88fb9c5c8da

CoNLL editions

2018 - Brussels, Belgium
2017 - Vancouver, Canada
2016 - Berlin, Germany
2015 - Bejing, China
2014 - Baltimore, MD, USA
2013 - Sofia, Bulgaria
2012 - Jeju Island, Korea
2011 - Portland, OR, USA
2010 - Uppsala, Sweden
2009 - Boulder, CO, USA
2008 - Manchester, UK
2007 - Prague, Czech Republic
2006 - New York City, NY, USA
2005 - Ann Arbor, MI, USA
2004 - Boston, MA, USA
2003 - Edmonton, Canada
2002 - Taipei, Taiwan
2001 - Toulouse, France
2000 - Lisbon, Portugal
1999 - Bergen, Norway
1998 - Sidney, Australia
1997 - Madrid, Spain

“inline” Annotation

(4) a. Perhaps without realizing it, Mr. Taffner simultaneously has put his finger on the problem and an ideal solution: “Capital City” should have been a comedy, a worthy sequel to the screwball British “Carry On” movies of the 1960s.

b. Perhaps without realizing it, Mr. <ENAMEX TYPE=“PERSON”>Taffner</ENAMEX> simultaneously has put his finger on the problem and an ideal solution: <ENAMEX TYPE=“WORK_OF_ART” >“Capital City”</ENAMEX> should have been a comedy, a worthy sequel to the screwball <ENAMEX TYPE=“NORP”>British</ENAMEX> “<ENAMEX TYPE=“WORK_OF_ART” >Carry On</ENAMEX>” movies of <ENAMEX TYPE=“DATE”>the 1960s</ENAMEX>.

```
( (S (NP-SBJ (NNP Bartok))
     (VP (VBZ describes)
                 (NP (NP (DT the) (NN form))
             (PP (IN of)
                       (NP (DT the) (JJ first) (NN movement))))
             (PP-CLR (IN as)
                      (NP (NP (ADJP (ADVP (" ") (RBR more)
                            (CC or) (RBR less)) (JJ regular))
                                 (NN sonata) (NN form)))))
             (, .)
```

CONLL: Computational Natural Language Learning
https://www.conll.org

Every token on a separate line followed by TAB separated columns with annotations (TSV)

ID	FORM	LEMMA	UPOSTAG	XPOSTAG	FEATS	HEAD	DEPREL	DEPS	NER
# newdoc url = http://www.poweredbyosteons.org/2012/01/brief-history-of-bioarchaeological.html									
# newdoc s3 = s3://aws-publicdatasets/common-crawl/crawl-data/CC-MAIN-2016-07/segments...									
...									
# sent_id = http://www.poweredbyosteons.org/2012/01/brief-history-of-bioarchaeological.html#60									
# text = The American Museum of Natural History was established in New York in 1869.									
0	The	the	DT	DT	_	2	det	2:det	O
1	American	American	NNP	NNP	_	2	nn	2:nn	B-Organization
2	Museum	Museum	NNP	NNP	_	7	nsubjpass	7:nsubjpass	I-Organization
3	of	of	IN	IN	_	2	prep	_	I-Organization
4	Natural	Natural	NNP	NNP	_	5	nn	5:nn	I-Organization
5	History	History	NNP	NNP	_	3	pobj	2:prep_of	I-Organization
6	was	be	VBD	VBD	_	7	auxpass	7:auxpass	O
7	established	establish	VBN	VBN	_	7	ROOT	7:ROOT	O
8	in	in	IN	IN	_	7	prep	_	O
9	New	New	NNP	NNP	_	10	nn	10:nn	B-Location
10	York	York	NNP	NNP	_	8	pobj	7:prep_in	I-Location
11	in	in	IN	IN	_	7	prep	_	O
12	1869	1869	CD	CD	_	11	pobj	7:prep_in	O
13	.	.	.	.	_	7	punct	7:punct	O
...									

https://www.clips.uantwerpen.be/conll2003/ner/

IO(B) style
I = insight
O = outside
B = beginning

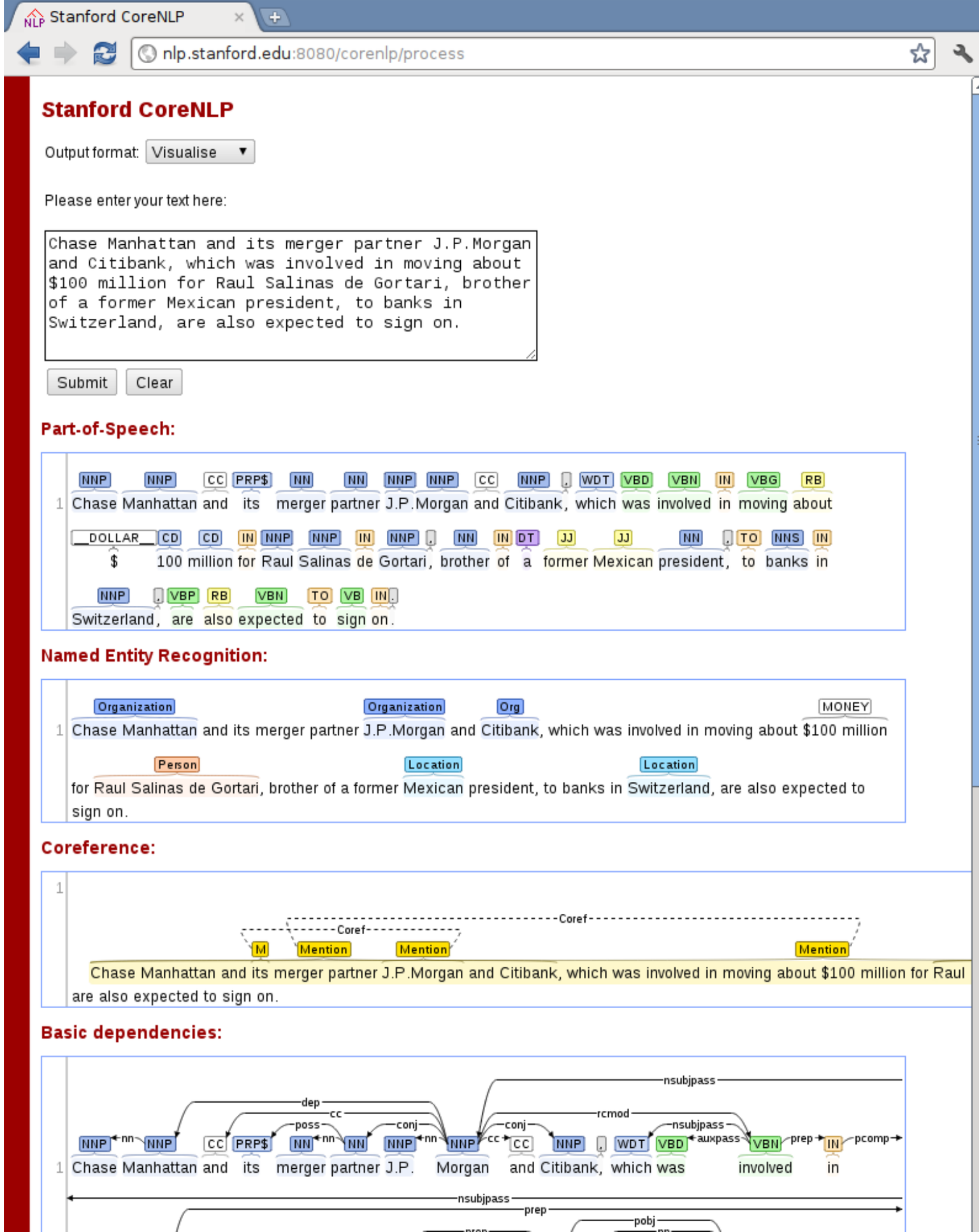
Creating the training data
Data annotation procedure

---

1. Collect texts: e.g. tweets, news, blogs, books
2. Define an **annotation scheme** or **code book**:
    - tag set (e.g. PoS labels, emotions, entity types)
    - the unit of the annotation: word, phrase, sentence, paragraph, document
    - criteria to apply a tag to a piece of text
3. Train **human annotators** to use the annotation scheme or create a **crowd task**
4. Provide an **annotation tool** that loads texts and allow the annotator to assign tags
5. **Store** the annotations with the text, e.g. in XML or TAB separated
6. Determine the **Inter-Annotator-Agreement** (IAA) by analysing texts annotated by at least one annotator
7. Fix disagreements (**adjucation**): if IAA is too low (<60 Kappa) the task is considered too difficult
8. IAA is considered the **upper ceiling of NLP;** can machines do better than humans?

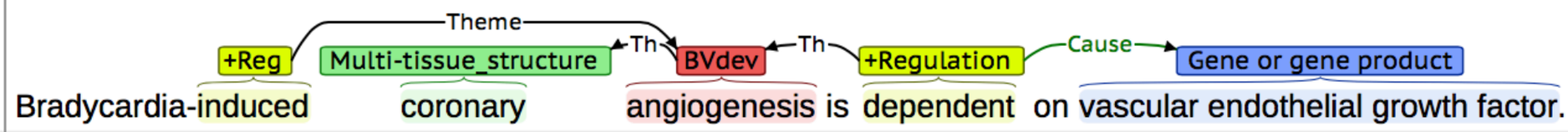
BRAT: Annotation environment

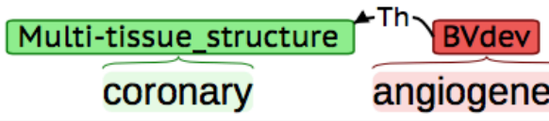
http://brat.nlplab.org/examples.html#corpus-examples-brat

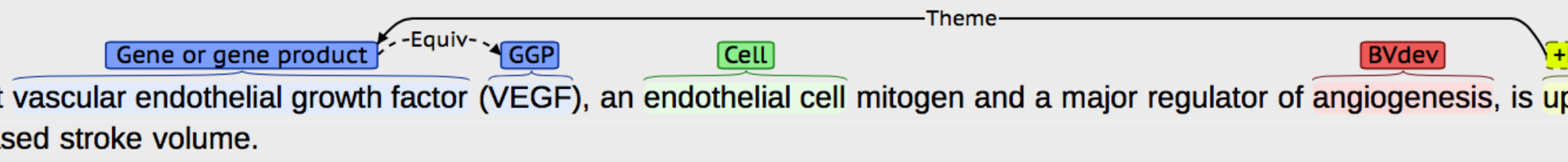


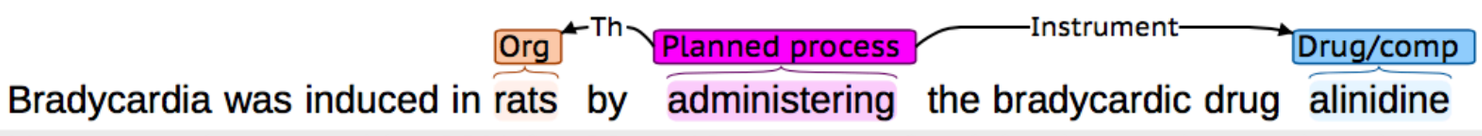
Multi-Level Event Extraction (MLEE) corpus

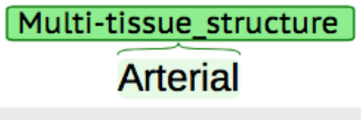
http://brat.nlplab.org/examples.html#corpus-examples-brat

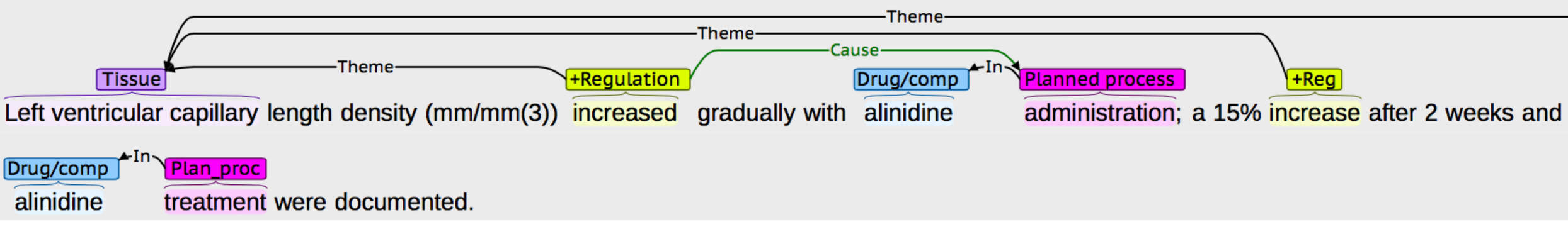


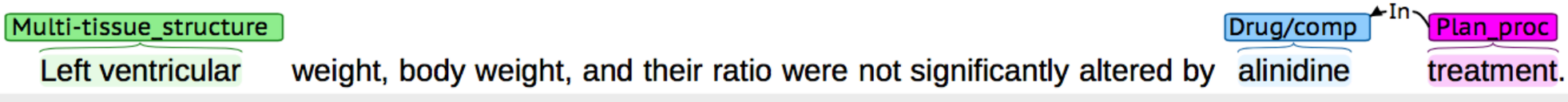


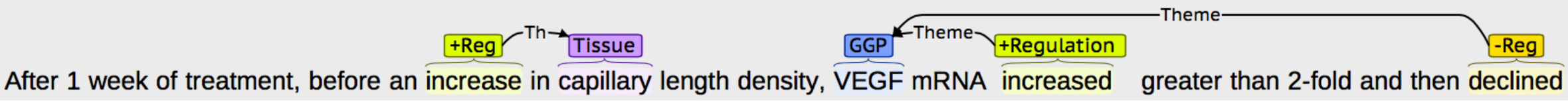












Gold, silver & bronze data

- **Gold data** is created by human annotators, e.g. IOB annotations for entities

- **Silver data** is proposed by a machine and validated by people, IOB entity annotations corrected by people

- **Bronze data** is generated by a machine and not validated, e.g. lemmas, PoS tags, dependencies annotations in addition to the IOB annotations

Feature representation for text

- Any property of the text, including annotations, except for the (***gold***) label to be predicted
- Text length, average sentence length, word length …
- Case (word shape), punctuation, …
- The (surrounding) words….
- Lexical word properties: part of speech, meaning, sentiment
- Sentence properties: syntax, semantic roles
- Document properties: topic, genre, publication date
- ***All features are mapped to vector representations!***

Words as features
Bag of words (BoW) model of text

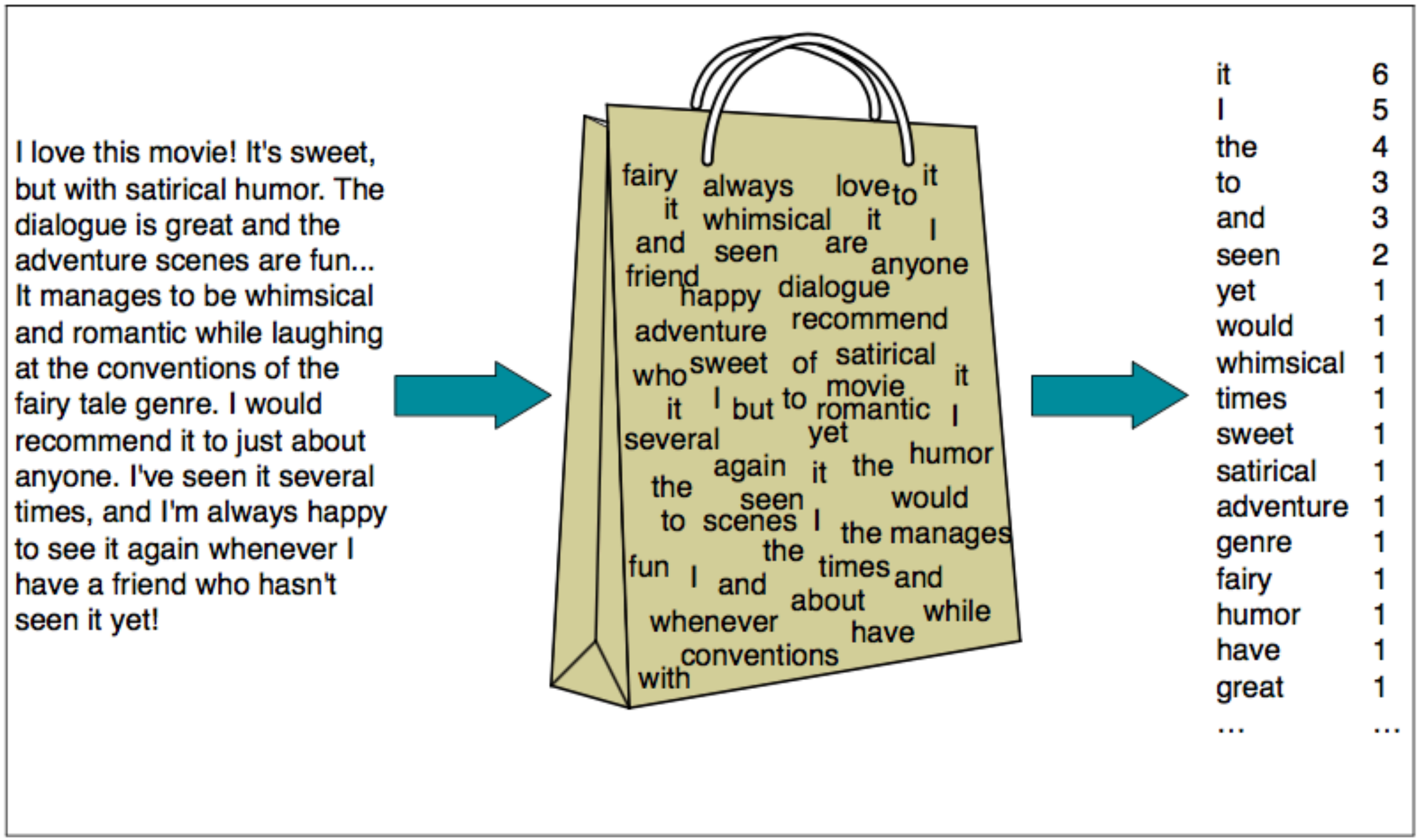


Figure 6.1 Intuition of the multinomial naive Bayes classifier applied to a movie review. The position of the words is ignored (the *bag of words* assumption) and we make use of the frequency of each word.

Word to Document index

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	5	117	0	0

Figure 15.2 The term-document matrix for four words in four Shakespeare plays. The red boxes show that each document is represented as a column vector of length four.

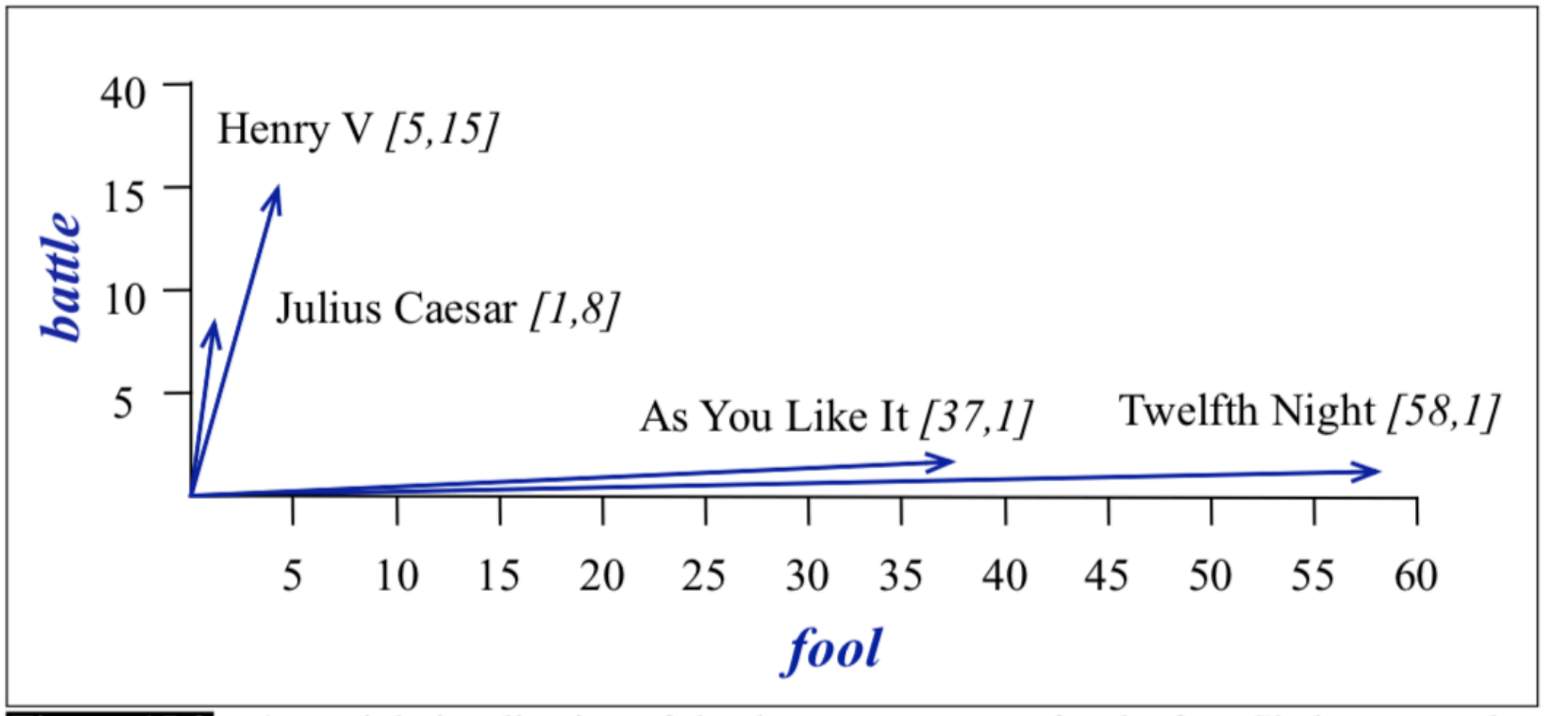


Figure 15.3 A spatial visualization of the document vectors for the four Shakespeare play documents, showing just two of the dimensions, corresponding to the words *battle* and *fool*. The comedies have high values for the *fool* dimension and low values for the *battle* dimension.

Not all words are equal
Information value (Spark-Jones 1972)

- Observations:
  - Words that occur in all texts, e.g. ***a***, ***the***, ***case***, ***of***, ***is*** have a low information value, despite their frequency
  - Words that occur once in one text are too idiosyncratic
  - Good words are typical for a topic or domain: shared to some extent but not too often
- IV (**t**|**d**) = TF*IDF = Term Frequency * Inverse Document Frequency

$$IV(\mathbf{t}|\mathbf{d}) = \frac{\text{Frequency of term } \boldsymbol{t} \text{ in document } \boldsymbol{d}}{\text{Total number of documents in which term } \boldsymbol{t} \text{ occurs}}$$

More than words, less than words
N-Grams

- Sequences of 1, 2, 3, 4, 5 etc. words:
  - the, room, was, tidy (unigram, 1-gram)
  - the_room, room_was, was_tidy (bi-gram, 2-gram)
  - the_room_was, room_was_tidy (tri-gram, 3-gram)
- Character n-grams:
  - [t, th, he, e_, _r, ro, oo, om, m_, _w, wa, as, s_, _t, ti, id, dy, y]
- ***What is an advantage of character n-grams?***

Word combinations
Pointwise Mutual Information

sum columns: 0, … 3, 7, 2, 5, 2 — total nr. of tokens = 19

	aardvark	...	computer	data	pinch	result	sugar	...	sum rows
apricot	0	...	0	0	1	0	1		1
pineapple	0	...	0	0	1	0	1		2
digital	0	...	2	1	0	1	0		4
information	0	...	1	6	0	4	0		11

Figure 15.4 Co-occurrence vectors for four words, computed from the Brown corpus, showing only six of the dimensions (hand-picked for pedagogical purposes). The vector for the word *digital* is outlined in red. Note that a real vector would have vastly more dimensions and thus be much sparser.

Pointwise Mutual Information: is a combination more frequent than expected?

$$\text{PMI}(w,c) = \log_2 \frac{P(w,c)}{P(w)P(c)}$$

P = probability

$$P(\text{w=information,c=data}) = \frac{6}{19} = .316$$

$$P(\text{w=information}) = \frac{11}{19} = .579$$

$$P(\text{c=data}) = \frac{7}{19} = .368$$

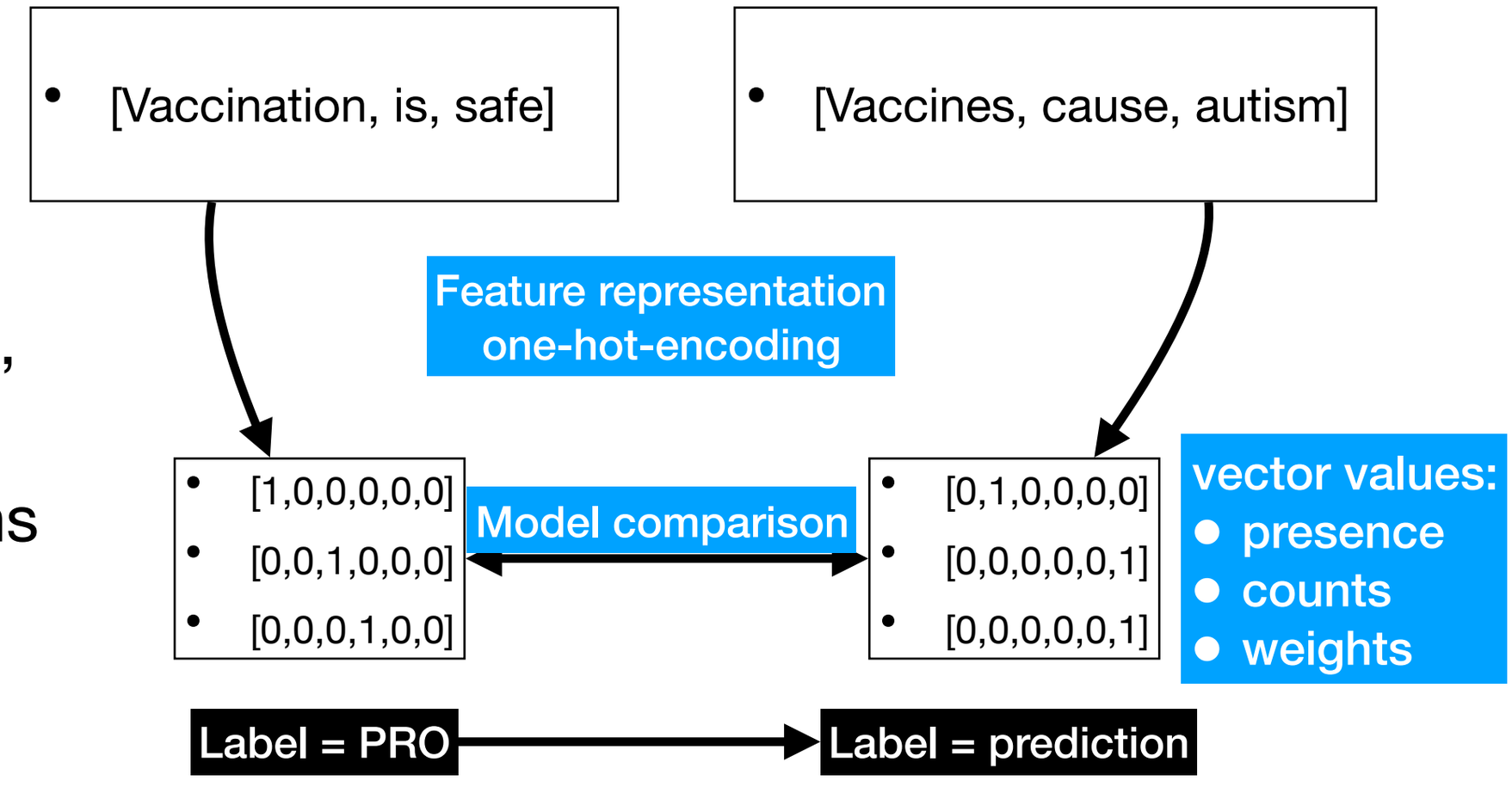
$$\text{ppmi}(\text{information,data}) = \log 2(.316/(.368 * .579)) = .568$$

Nice properties of vector representations

- Very flexible and fine-gained:
    - translate ***anything*** to a vector position: words and sentences sentences, images and pixels, word combinations, part-of-speech, word distance, etc..
        - with scalar ***values*** between 0 and 1 (association strength, signal strength, frequencies)
- Memory efficient (digits instead of words)
- Mathematical operations: similarity, analogy, multiply, concatenate, average, weigh, subtract, etc…

Representing words as vector values
One-Hot-Encoding

- **Feature space**
- vocabulary = [vaccination, vaccines, is, safe, cause, autism]
- Vector dimensions
  - [0,0,0,0,0,0]

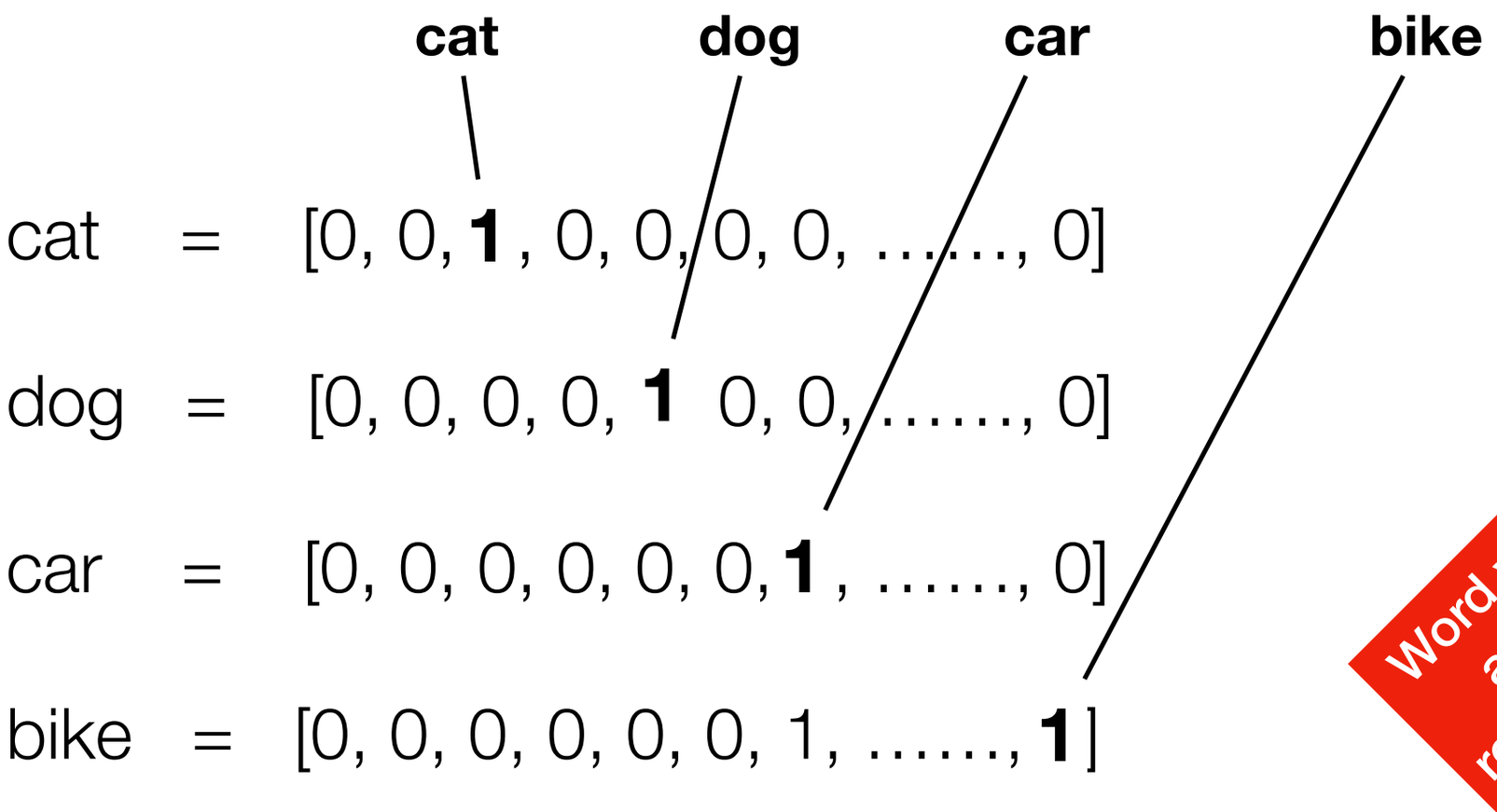


similarity of feature vectors is the normalised dot product =

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} \frac{a_i b_i}{n} = \frac{a_1 b_1 + a_2 b_2 + \cdots + a_n b_n}{n}$$

One-hot encodings to represent words as features

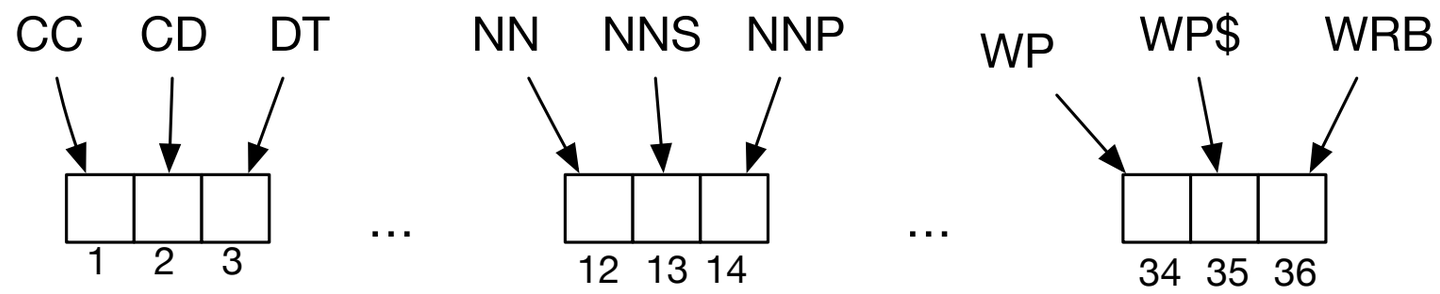
Lengths of the vocabulary is the size of the dimensions



Word vectors become large and are sparse, even if restricted to words with high information value. What about n-grams?

Part-of-Speech as one-hot-encoding

- 36 tags in the Penn Treebank: 36 dimensions (one-hot representation)



CC <1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0>

CD <0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0>

DT <0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0>

NN <0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0>

NNS <0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0>

NNP <0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0>

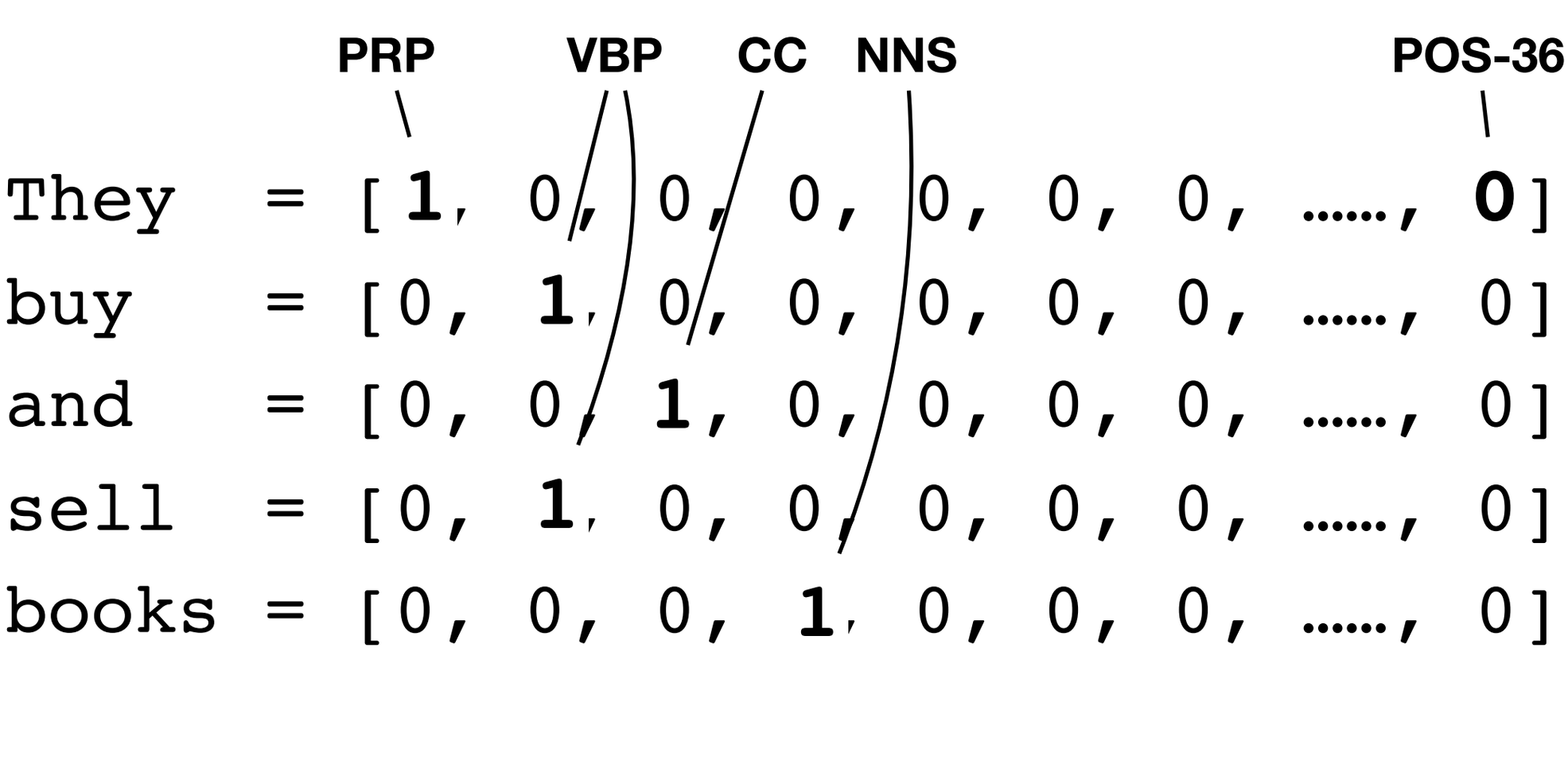
WP <0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0>

WP$ <0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0>

WRB <0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1>

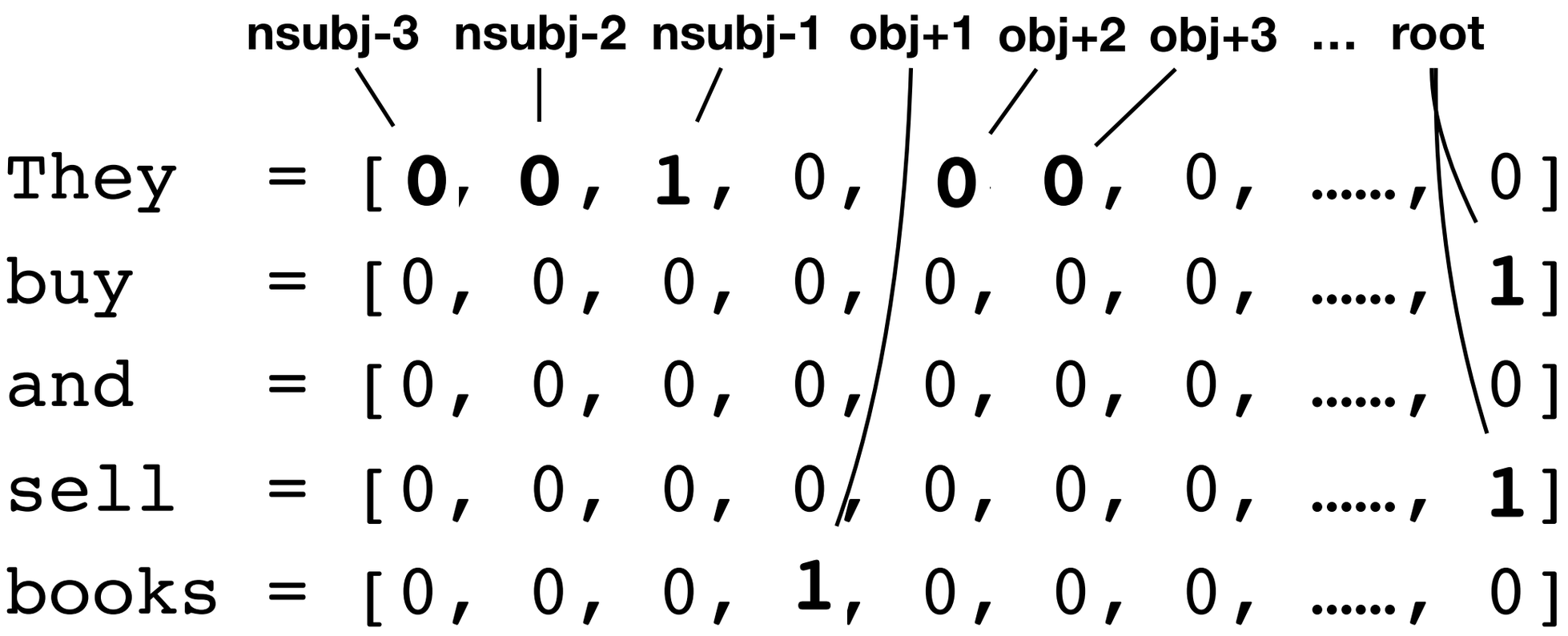
Part-of-Speech as one-hot-encoding

```
# text = They buy and sell books.
1    They   they    PRON   PRP  Case=Nom|Number=Plur              2  nsubj 2:nsubj|4:nsubj
2    buy    buy     VERB   VBP  Number=Plur|Person=3|Tense=Pres   0  root  0:root
3    and    and     CONJ   CC   _                                 4  cc    4:cc
4    sell   sell    VERB   VBP  Number=Plur|Person=3|Tense=Pres   2  conj  0:root|2:conj
5    books  book    NOUN   NNS  Number=Plur                       2  obj   2:obj|4:obj
6    .      .       PUNCT  .    _                                 2  punct 2:punct
```



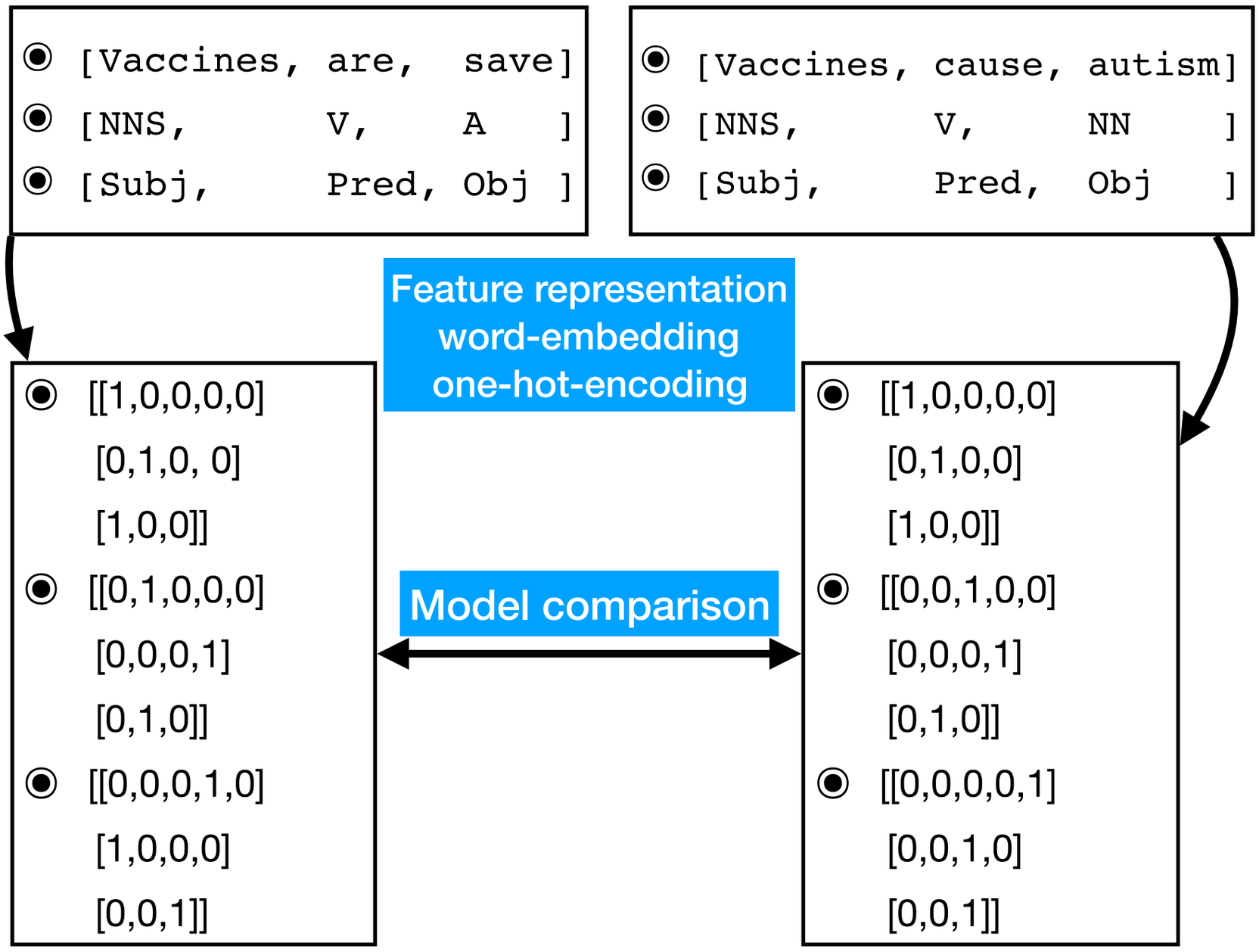
Syntactic dependencies as one-hot-encoding

```
# text = They buy and sell books.
1    They   they    PRON   PRP  Case=Nom|Number=Plur                2  nsubj 2:nsubj|4:nsubj
2    buy    buy     VERB   VBP  Number=Plur|Person=3|Tense=Pres     0  root  0:root
3    and    and     CONJ   CC   _                                   4  cc    4:cc
4    sell   sell    VERB   VBP  Number=Plur|Person=3|Tense=Pres     2  conj  0:root|2:conj
5    books  book    NOUN   NNS  Number=Plur                         2  obj   2:obj|4:obj
6    .      .       PUNCT  .    _                                   2  punct 2:punct
```

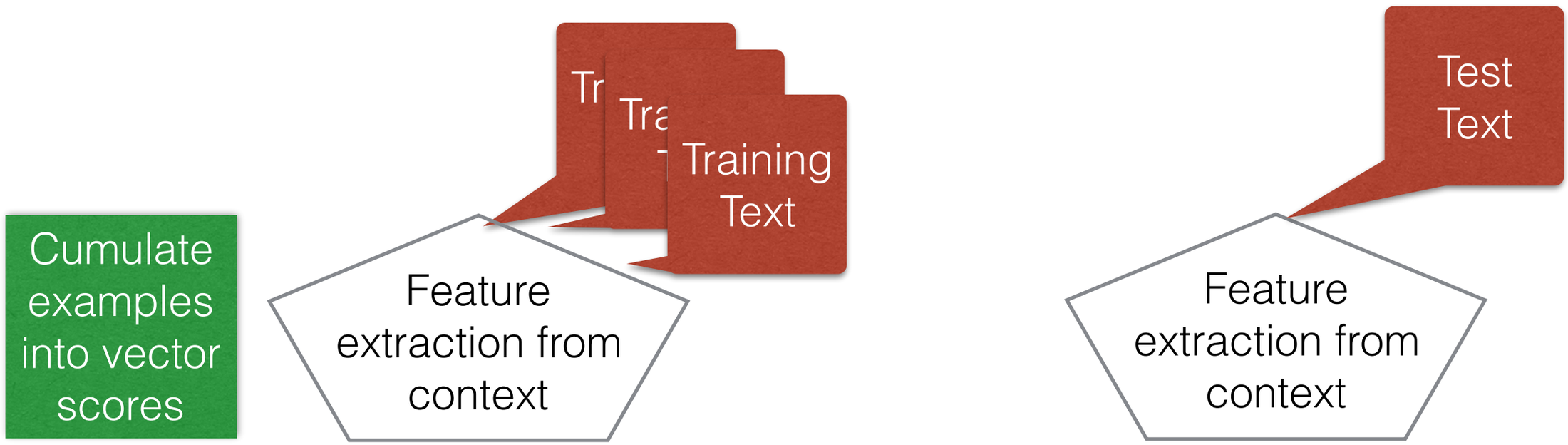


Combining one-hot-encodings

- **Feature space**
- vocabulary = [**vaccines, are, cause, save, autism**]
- PoS = [A, NNS, NN, V]
- Dependencies = [Subj, Pred, Obj]
- Vector dimensions
  - [[0,0,0,0,0] + [0,0,0,0,0] + [0,0,0]]



Feature models
Combine many different types of features



same vector positions and length across

LABEL [0,1,0,0,1,0,1,1,0]

LABEL [1,0,1,0,1,0,0,0,1]

[1,0,0,0,1,0,1,0,0]

feature vectors

feature vector

similarity of feature vectors is the normalised dot product =

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} \frac{a_i b_i}{n} = \frac{a_1 b_1 + a_2 b_2 + \cdots + a_n b_n}{n}$$

Problems with sparse vector encodings

- Thousands of documents have hundreds of thousands of words

- Each token in a text is represented by the combined vectors of all possible values

- Vectors become very large and sparse (hundreds of thousands of positions, most empty)

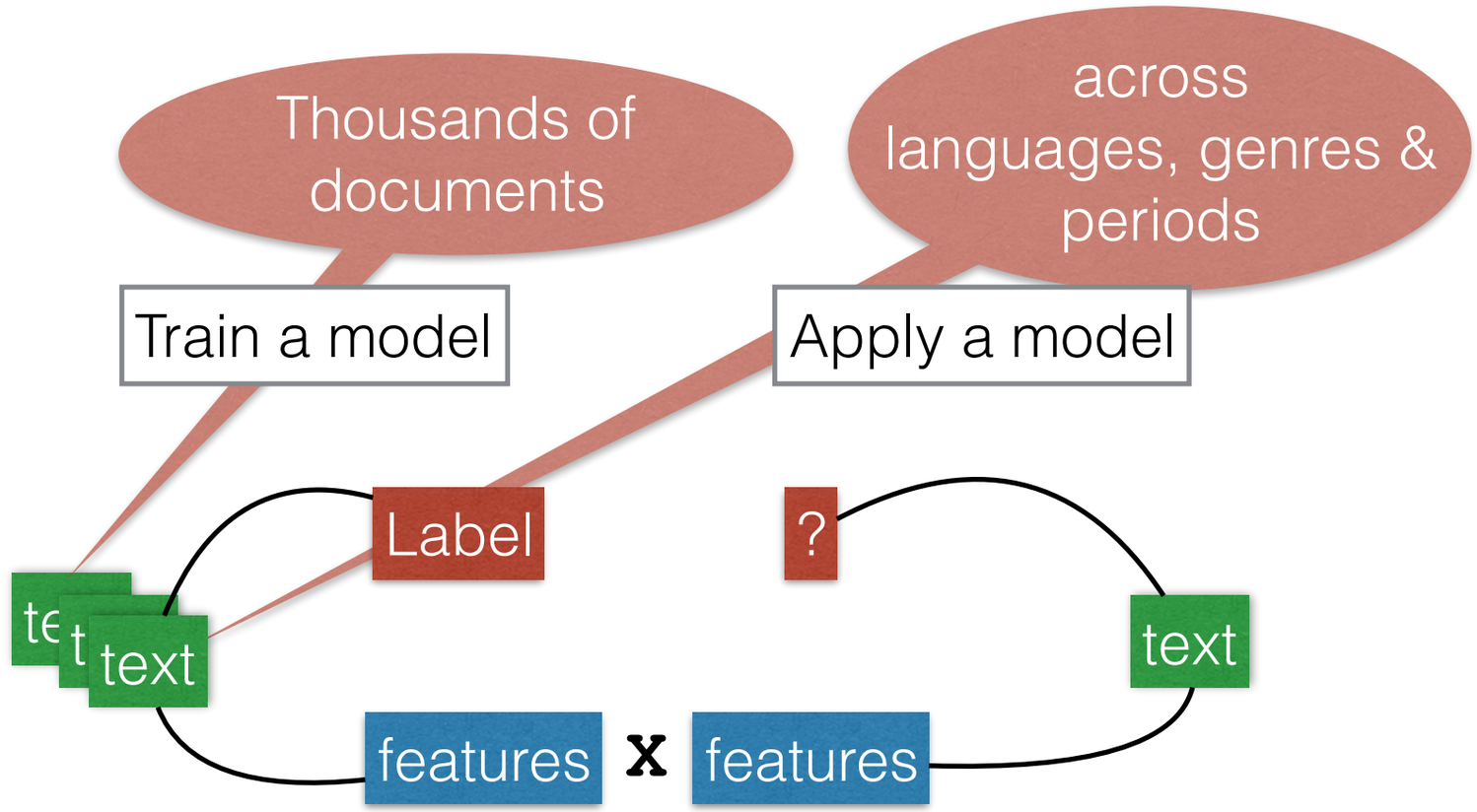
- Difficult to generalise over vector representations

- Rich and complex feature engineering is needed to compensate for lack of data

Never enough annotated data

For each module in a pipeline

- morphology
- grammar
- entities
- concepts
- hedging
- events
- time expressions
- locations
- emotions, intentions



Unsupervised learning
word embeddings
neural networks

Expand

Annotated data

Clustered data, not annotated

Supervised Learning

Unsupervised Learning

Kontagent

Tagged training data and feature engineering

Data clustering and feature engineering

Word in context or distributional model

- I'm **a bass (I** can reach the **low B**, even a **low A** in the mornings). … If you have a low **register** voice, it's also a nice **song** to **sing** with a warm **voice**, **register** wise. **The bass range** starts at a certain **C**.

- Whether you're a competitive sports **fisher** or just want to spend a fun afternoon with your family on the **lake**, **bass fishing** is a difficult pastime if you aren't properly equipped.

- In **choral music**, **the alto range** is from **G3** referring to **G** below **key** of **C** middle to **F5** which is the **F** in 2nd **octave** above the **C** middle.

- **Waxworms** are used as live-b **for trout fishing**. **Corn worms** are also excellent live-**bait when** trout **fishing**. **Nymph** of a golden **stonefly** are used as live-bait **for trout fishing**. Nymph mayfly. **salmon roe (Red caviar) Worms** are cheap and a great **bait** to use **for trout and** most types of **fish**.

- Important information **about trout fishing** on **Lake** Taupo

Harris 1954: similar contexts define similar meanings

Word - Word co-occurrence matrix

- **voice**, **register** wise. The **bass range** starts at a certain **C**.
- the **alto range** is from **G3** referring to **G** below **key** of **C**
- your **family** on the **lake**, **bass fishing** is a difficult pastime
- Important information about **trout fishing** on **Lake** Taupo

[Length of the vector is the size of defined context vocabulary]

	C	family	fishing	G	G3	lake	key	range	register	voice
bass	[1,	0,	0,	0,	0,	0,	0,	1,	1,	1]
alto	[1,	0,	0,	1,	1,	0,	1,	1,	0,	0]
bass	[0,	1,	1,	0,	0,	1,	0,	0,	0,	0]
trout	[0,	0,	1,	0,	0,	1,	0,	0,	0,	0]

Represent words by their contexts instead of one-hot-encodings:
- **we can reduce and tune dimensions by defining the context**
- **similar words get similar vector representations (generalise)**

Word - Word co-occurrence matrix

- **voice**, **register** wise. The **bass range** starts at a certain **C**.
- the **alto range** is from **G3** referring to **G** below **key** of **C**
- your **family** on the **lake**, **bass fishing** is a difficult pastime
- Important information about **trout fishing** on **Lake** Taupo

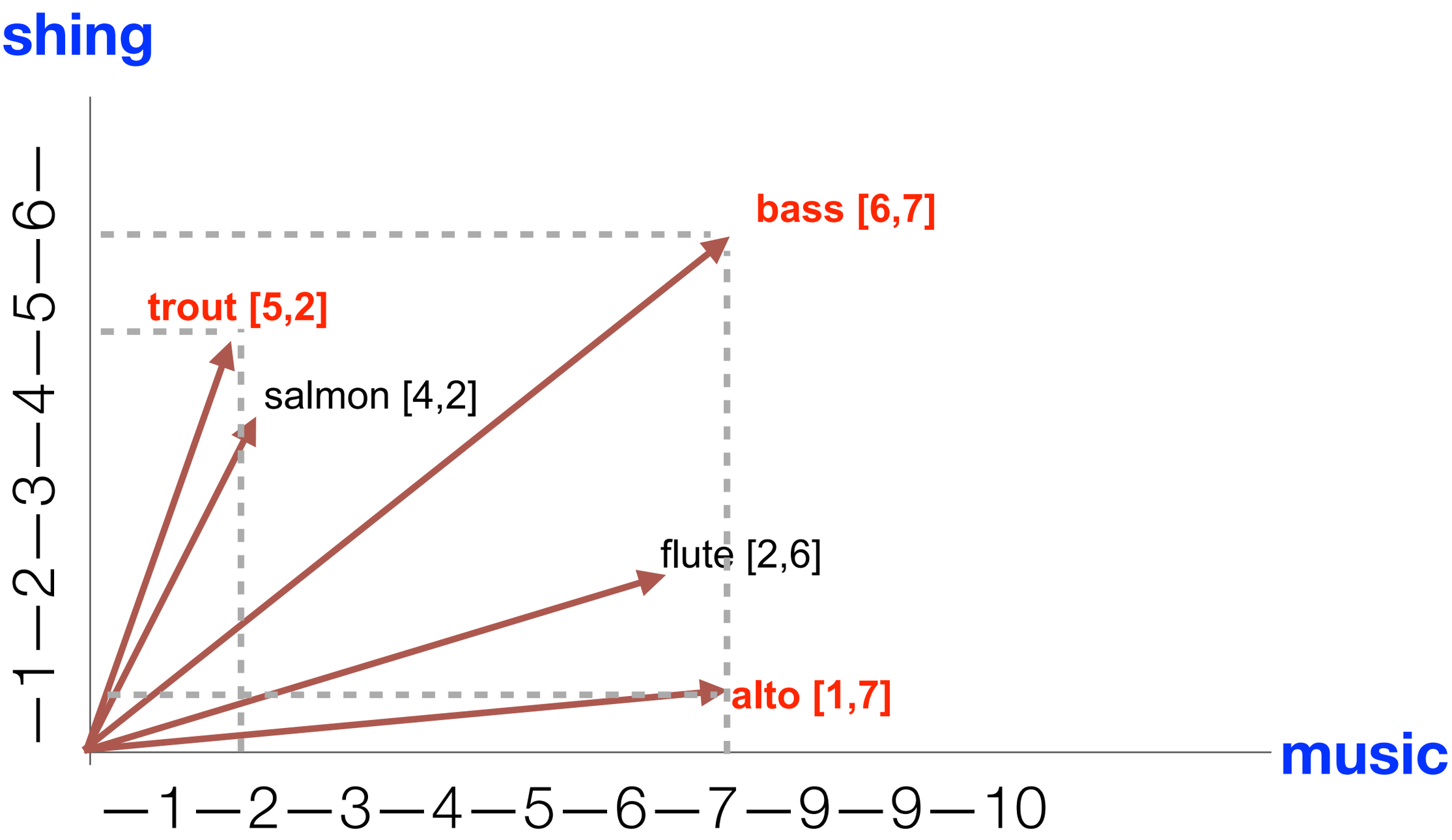
[Length of the vector is the size of defined context vocabulary]

	C	family	fishing	G	G3	lake	key	range	register	voice
bass	1	0	0	0	0	0	0	1	1	1
alto	1	0	0	1	1	0	1	1	0	0
bass	0	1	1	0	0	1	0	0	0	0
trout	0	0	1	0	0	1	0	0	0	0

similarity of word context is the normalised dot product of the vectors =

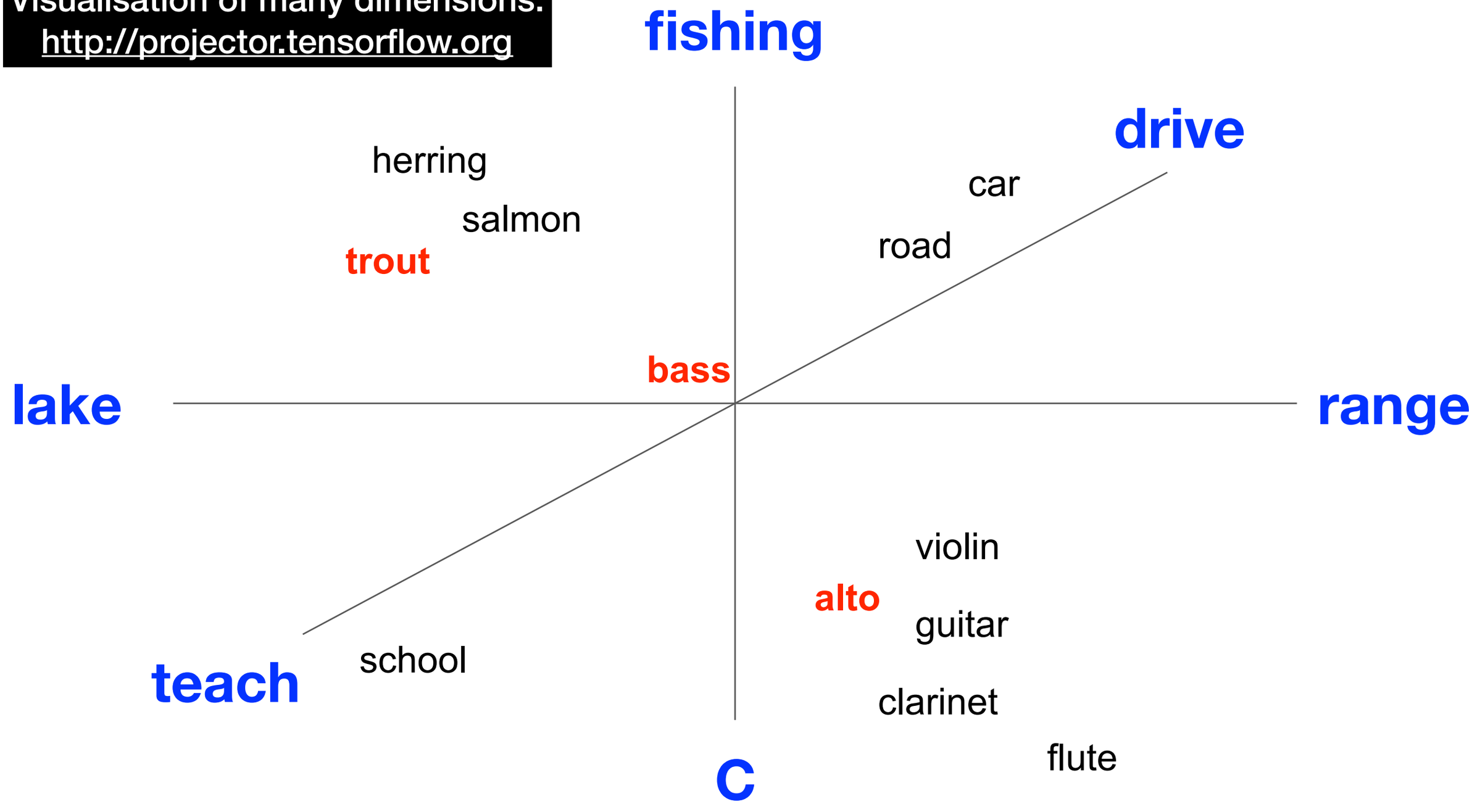
$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} \frac{a_i b_i}{n} = \frac{a_1 b_1 + a_2 b_2 + \cdots + a_n b_n}{n}$$

Simplified view in two dimensions
Word Vectors or Word embeddings

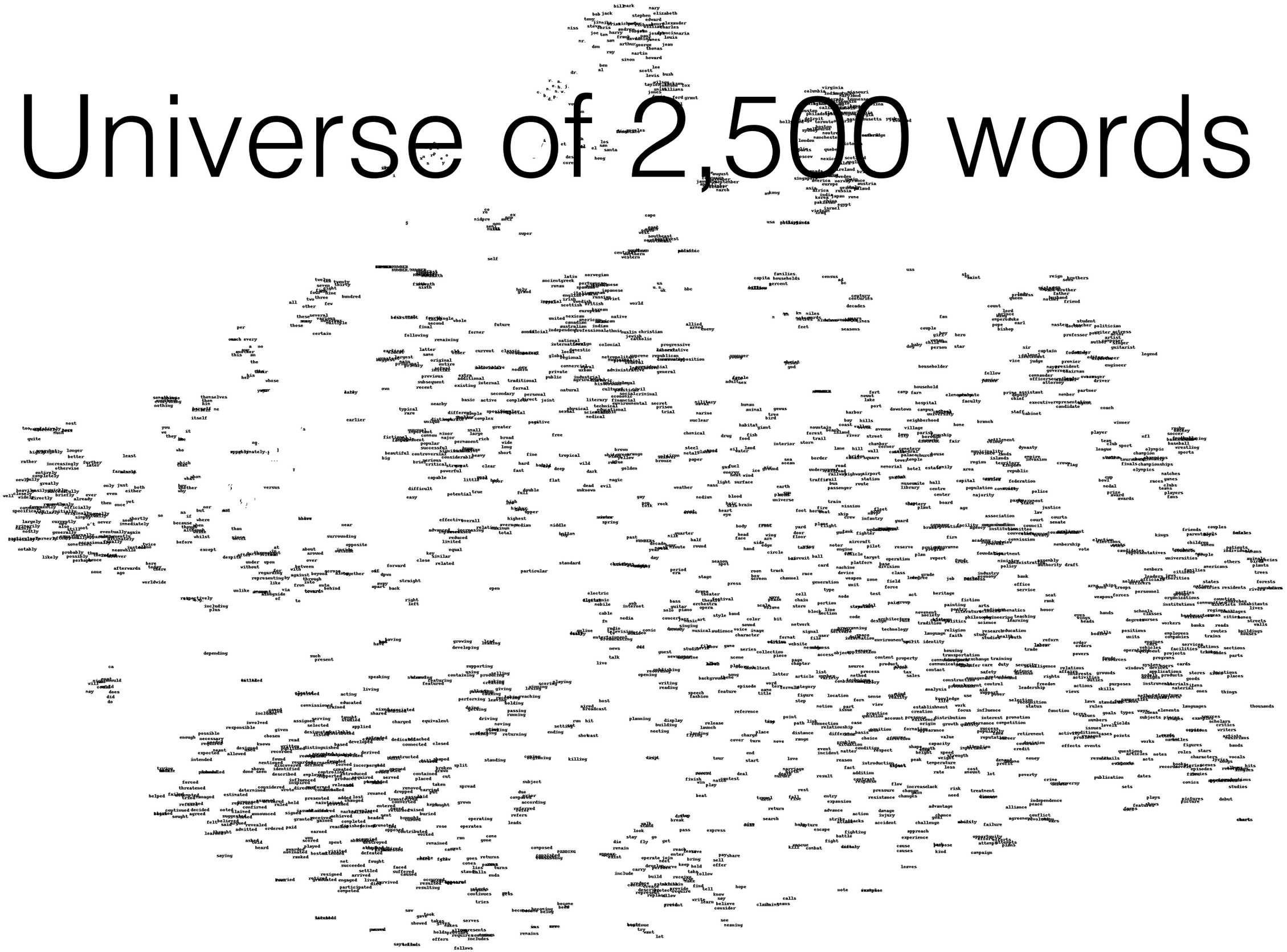


Visualisation in n-dimensions
Word Vectors or Word embeddings

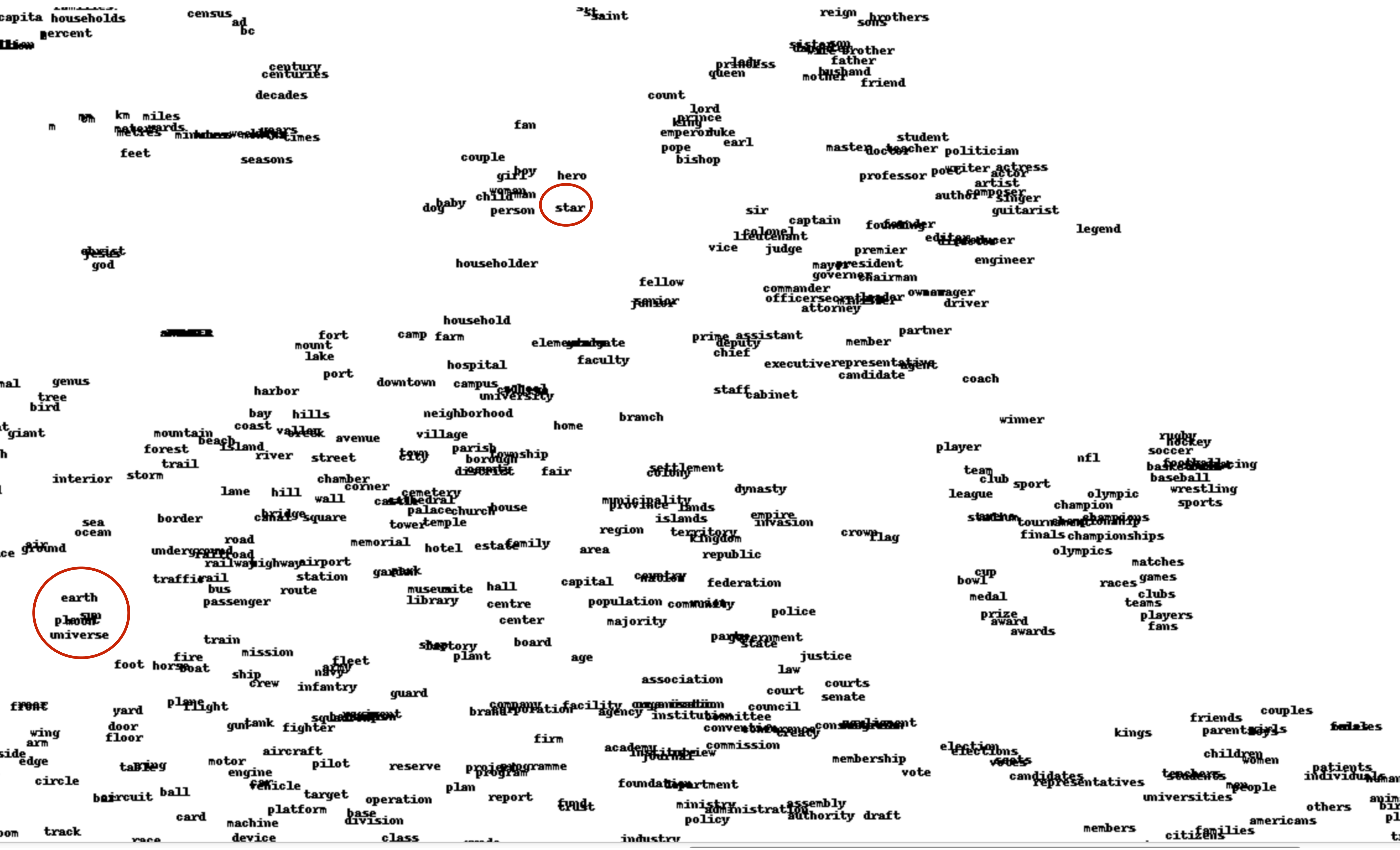
Visualisation of many dimensions:
http://projector.tensorflow.org



Universe of 2,500 words



Joseph Turian's map of 2500 English words produced by using t-SNE on the word feature vectors learned by Collobert & Weston, ICML 2008



capita
households
census
ad
bc
percent
saint
reign
brothers
sons
century
centuries
decades
father
husband
mother
friend
queen
count
lord
prince
king
emperor
duke
earl
pope
bishop
km
miles
feet
times
seasons
fan
couple
boy
girl
hero
child
woman
baby
dog
person
star
student
master
doctor
teacher
politician
professor
poet
writer
actress
actor
artist
author
composer
singer
guitarist
sir
captain
founder
lieutenant
vice
judge
premier
editor
engineer
legend
god
householder
mayor
president
governor
chairman
fellow
commander
officer
leader
owner
manager
driver
attorney
junior
senior
household
fort
mount
lake
port
harbor
camp
farm
hospital
downtown
campus
college
university
faculty
prime
assistant
deputy
chief
member
partner
executive
representative
agent
candidate
coach
genus
tree
bird
giant
staff
cabinet
bay
hills
coast
mountain
beach
island
valley
creek
avenue
neighborhood
village
home
branch
winner
forest
trail
river
street
town
city
parish
borough
township
district
fair
settlement
colony
player
nfl
rugby
hockey
soccer
football
basketball
baseball
interior
storm
chamber
corner
team
club
sport
league
olympic
wrestling
sports
lane
hill
wall
cemetery
cathedral
palace
church
house
dynasty
province
lands
islands
empire
invasion
champion
stadium
tournament
championships
finals
olympics
sea
ocean
border
square
tower
temple
region
territory
kingdom
crown
flag
ground
road
memorial
hotel
estate
family
area
republic
railroad
railway
highway
airport
matches
games
traffic
rail
bus
passenger
station
route
museum
library
hall
centre
center
capital
country
nation
federation
cup
bowl
races
medal
clubs
teams
earth
planet
universe
population
community
police
prize
award
awards
players
fans
majority
train
mission
board
party
government
state
justice
fire
foot
boat
ship
crew
fleet
army
navy
infantry
plant
age
association
law
court
courts
senate
guard
front
yard
plane
flight
door
floor
tank
fighter
company
corporation
facility
agency
organization
council
committee
institution
commission
convention
firm
kings
friends
couples
parents
girls
boys
children
women
wing
arm
edge
aircraft
motor
engine
pilot
reserve
program
programme
academy
journal
membership
vote
election
elections
voters
seats
candidates
representatives
teachers
students
patients
individuals
men
people
circle
vehicle
target
plan
foundation
department
universities
ball
operation
report
fund
trust
ministry
administration
assembly
authority
draft
others
americans
card
platform
base
division
policy
members
families
citizens
track
machine
device
class
industry

Word Embeddings

- Context vectors are better than one-hot-vectors but still large and need a lot of data to become effective, Baroni et al (ACL, 2014, Don’t count, predict!)

- Neural network performs the task of predicting a word or the context

- Optimize the weights in the vector to perform the task

- The weighted vector becomes the representation of the word

- The vector does NOT represent the context (!!!), rather ***it represents the task*** to predict the context (!!!)

Computational implementation, Word2Vec, Mikolov et al. 2013

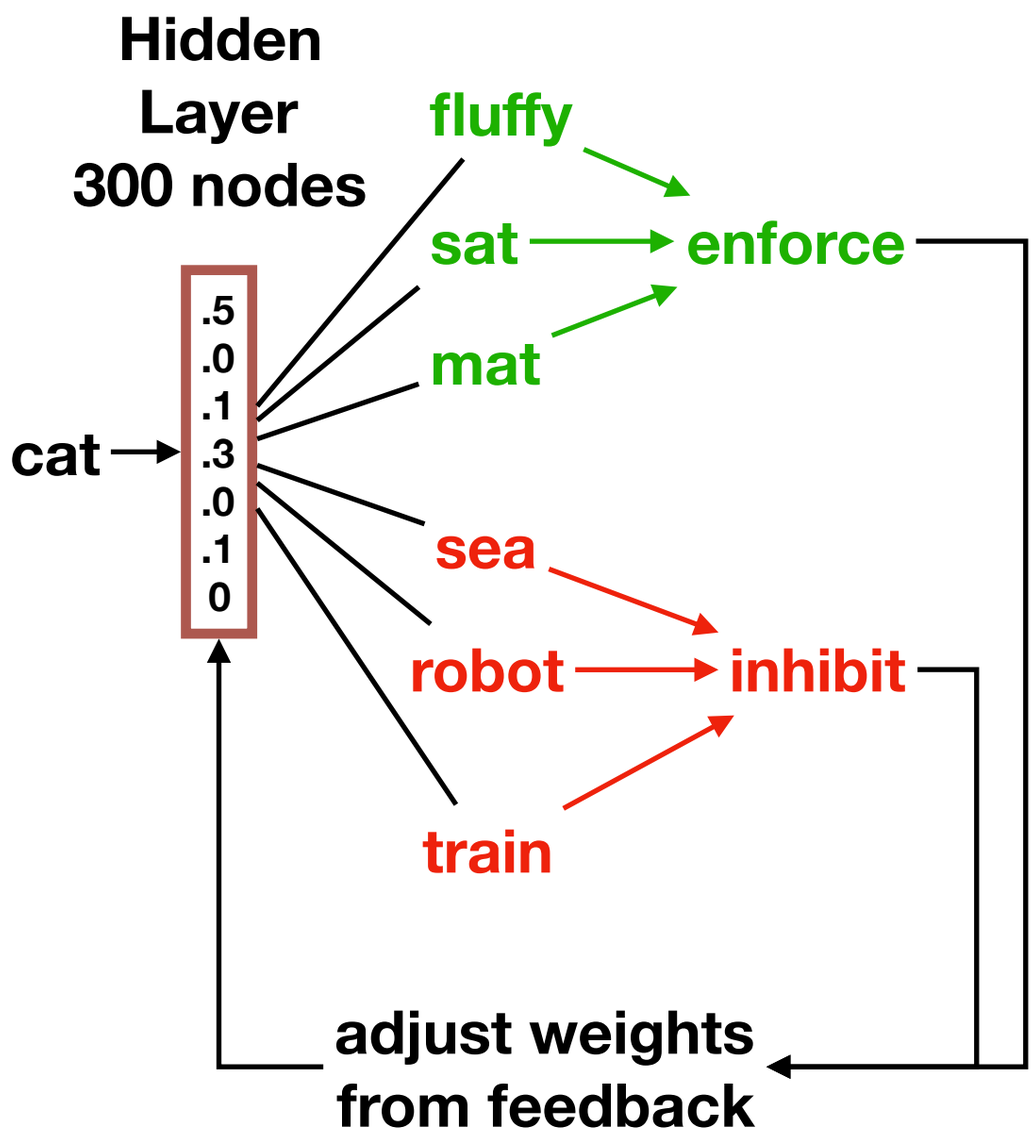
Word embeddings created by a neural network

- **break down contexts of a word into pairs (positive examples)**
- **create random pairs as negative examples, e.g. (fly, brown)**

Skip-gram Mikolov et al. 2013

- The fluffy cat sat on the mat

- The fluffy
- The fluffy cat
- fluffy cat sat
- cat sat on
- sat on the
- on the mat
- the mat



Embeddings represent families of related words as features

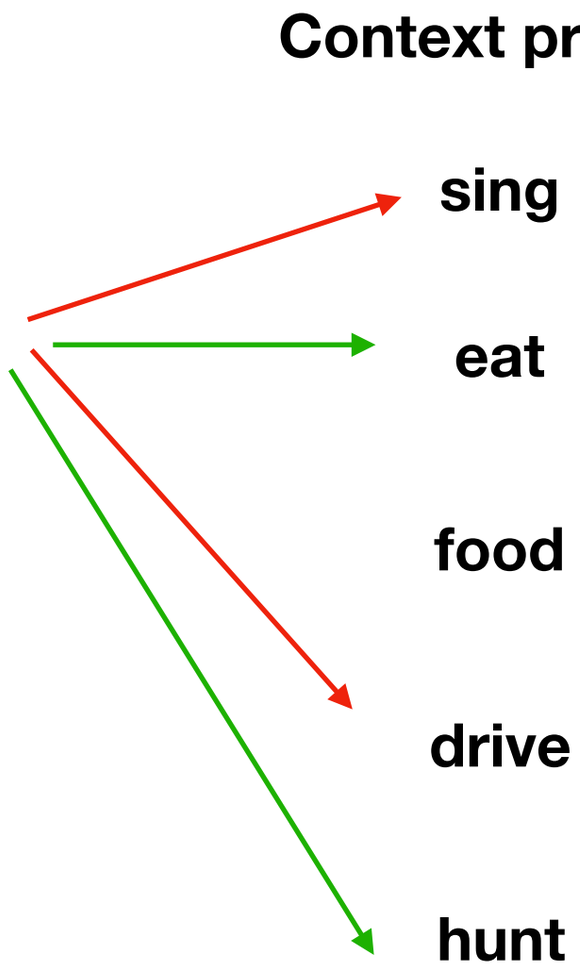
Lengths of the number of nodes in the embedding layer (300-500 node)

cat = [0.1, 0.4, 0.3, 0.9, 0.1, 0.8, 0.1]

dog = [0.1, 0.3, 0.4, 0.8, 0.1, 0.9, 0.1]

car = [0.2, 0.7, 0.3, 0.5, 0.1, 0.4, 0.1]

bike = [0.2, 0.7, 0.3, 0.5, 0.1, 0.4, 0.1]



Similarity through word-embeddings: 400 dimensions

- gensimmodel.most_similar(‘vaccination’)

[(‘immunisation', 0.7288434505462646),
 ('vaccinations', 0.6694121360778809),
 ('vaccine', 0.6497822999954224),
 ('immunization', 0.6272329092025757),
 ('vaccinated', 0.6225659251213074),
 ('vaccines', 0.6096125841140747),
 ('vaccinating', 0.5848771333694458),
 ('immunisations', 0.5839511156082153),
 ('immunised', 0.5639201402664185),
 ('vaccinate', 0.562936544418335)

vector([-0.399699, 0.06482 , 0.368686, -0.377926, -0.208074, 0.092846,
        0.263108, 0.001644, -0.131597, 0.248029, 0.193516, 0.358381,
       -0.061733, 0.3458 , 0.101475, -0.437568, 0.09023 , -0.221989,
       -0.177281, 0.186275, -0.177728, 0.03433 , 0.083448, -0.147547,
       -0.197424, 0.026185, 0.037277, 0.026462, 0.221429, 0.11221 ,
       -0.232993, 0.16042 , -0.232213, 0.295674, 0.008174, -0.127702,
       -0.042977, 0.044021, 0.015354, -0.008699, 0.050983, 0.307894,
       -0.180218, 0.174326, -0.015698, -0.046843, 0.278651, -0.115122,
        0.105387, 0.004491, 0.094357, 0.005139, 0.13199 , -0.051129,
        0.025236, 0.025753, -0.052252, -0.137729, 0.49573 , 0.098202,
        0.006868, 0.110939, 0.12603 , 0.296865, -0.376774, -0.060058,
        0.25929 , 0.357815, 0.053625, -0.095404, -0.080501, 0.233063,
        0.093019, -0.043595, 0.472759, -0.173012, 0.001466, 0.077951,
        0.222487, 0.019556, 0.237507, -0.160088, 0.117915, 0.179803,
        0.107175, 0.142339, 0.046431, 0.423276, -0.157801, 0.221353,

- gensimmodel.most_similar(‘autism’)

[(‘autistic', 0.709283709526062),
 ('adhd', 0.6003193855285645),
 ('dyslexia', 0.5945056676864624),
 ('dyspraxia', 0.5788421034812927),
 ('schizophrenia', 0.5415143966674805),
 ('high-functioning', 0.5388866662979126),
 ('aspergers', 0.5324583053588867),
 ('me/cfs', 0.5201038718223572),
 ('asd', 0.5089726448059082),
 ('neurodevelopmental', 0.5016077160835266)

vector([2.97010e-02, 1.77924e-01, -9.36180e-02, -7.38020e-02,
       -2.63494e-01, 5.39020e-02, 1.31954e-01, 1.20233e-01,
       -1.05351e-01, 1.14572e-01, 3.09344e-01, 6.28220e-02,
        2.01120e-02, -9.78420e-02, 1.20324e-01, 2.33840e-02,
        1.49410e-01, -3.08743e-01, -1.76643e-01, -4.06421e-01,
       -6.07290e-02, -2.45634e-01, 2.38010e-01, 1.11927e-01,
       -4.30489e-01, -4.17800e-02, 8.06020e-02, 1.17752e-01,
       -2.21880e-02, -9.04810e-02, -1.72199e-01, 1.33377e-01,
        5.44661e-01, 1.62586e-01, -4.16860e-02, -2.54711e-01,
        2.82401e-01, -8.09320e-02, -1.66021e-01, 1.35894e-01,
       -2.60487e-01, 1.47950e-01, 2.57920e-02, -1.48656e-01,
       -2.07999e-01, -2.04788e-01, 1.16318e-01, 2.79210e-01,
       -1.65796e-01, -4.18540e-02, 1.07020e-02, 6.09600e-03,
       -2.67482e-01, -2.57364e-01, 8.29890e-02, -4.44768e-01,
        2.52700e-02, 9.05790e-02, 2.24890e-01, 3.46180e-01,
        1.56335e-01, 3.06857e-01, 1.90560e-01, 9.49000e-03,

Similarity through word-embeddings: 400 dimensions

gensimmodel.most_similar(‘cause')

[('causes', 0.7360146045684814),
 ('causing', 0.6391870975494385),
 ('caused', 0.634495735168457),
 ('cause(s', 0.4665626883506775),
 ('suffer', 0.447427034378051 76),
 ('aggravate', 0.4438691139221 1914),
 ('palliate', 0.4382469654083252),
 ('predisposes', 0.4310530424118042),
 ('occur', 0.42683202028274536),
 ('aggravates', 0.4251204133033 7524)]

gensimmodel.most_similar(‘prevent')

[('preventing', 0.7388827800750732),
 ('avoid', 0.6655234694480896),
 ('prevents', 0.6595998406410217),
 ('prevented', 0.6542751789093018),
 ('deter', 0.5833278894424438),
 ('protect', 0.5643754601478577),
 ('discourage', 0.5477097034454346),
 ('forestall', 0.5160406827926636),
 ('stop', 0.514601409435272 2),
 ('eliminate', 0.5051982402801514)]

gensimmodel.most_similar(‘trigger’)

[('triggers', 0.6730290055274963),
 ('triggering', 0.630126297473907 5),
 ('triggered', 0.618744969367981),
 ('activates', 0.4707192182540893 6),
 ('monostable', 0.4318832755088806),
 ('microswitch', 0.4238961040973663 3),
 ('activate', 0.4231356978416443),
 ('deactivates', 0.4156171083450317 4),
 ('desensitize', 0.4147326052188873 3),
 ('full-auto', 0.412780284881591 8)]

gensimmodel.similarity(‘cause','trigger') = 0.33

gensimmodel.similarity(‘cause','prevent') = 0.37

gensimmodel.similarity(‘trigger','prevent') = 0.25

Using GenSim and GloVe to represent words

```
gensimmodel.most_similar('berlin')
```

```
[('munich', 0.665853972033691),
 ('hamburg', 0.6416077017784119),
 ('stuttgart', 0.602530002593941),
 ('germany', 0.597375750541687),
 ('nationalgalerie', 0.596231520175938),
 ('münchner', 0.594410419464113),
 ('leipzig', 0.588937163352963),
 ('charlottenburg', 0.583330333232796),
 ('bochum', 0.580147862434387),
 ('düsseldorf', 0.580057740211486)]
```

```
gensimmodel.most_similar('rome')
```

```
[('viterbo', 0.600153028964996),
 ('ravenna', 0.600064635276794),
 ('italy', 0.592195272445678),
 ('aquileia', 0.58372366428375),
 ('perugia', 0.581674814224243),
 ('civitavecchia', 0.578354597091675),
 ('anagni', 0.572831213474274),
 ('tarentum', 0.570874810218811),
 ('fiesole', 0.561999797821045),
 ('vigilius', 0.560334444046021)]
```

```
gensimmodel.get_vector('rome')
```

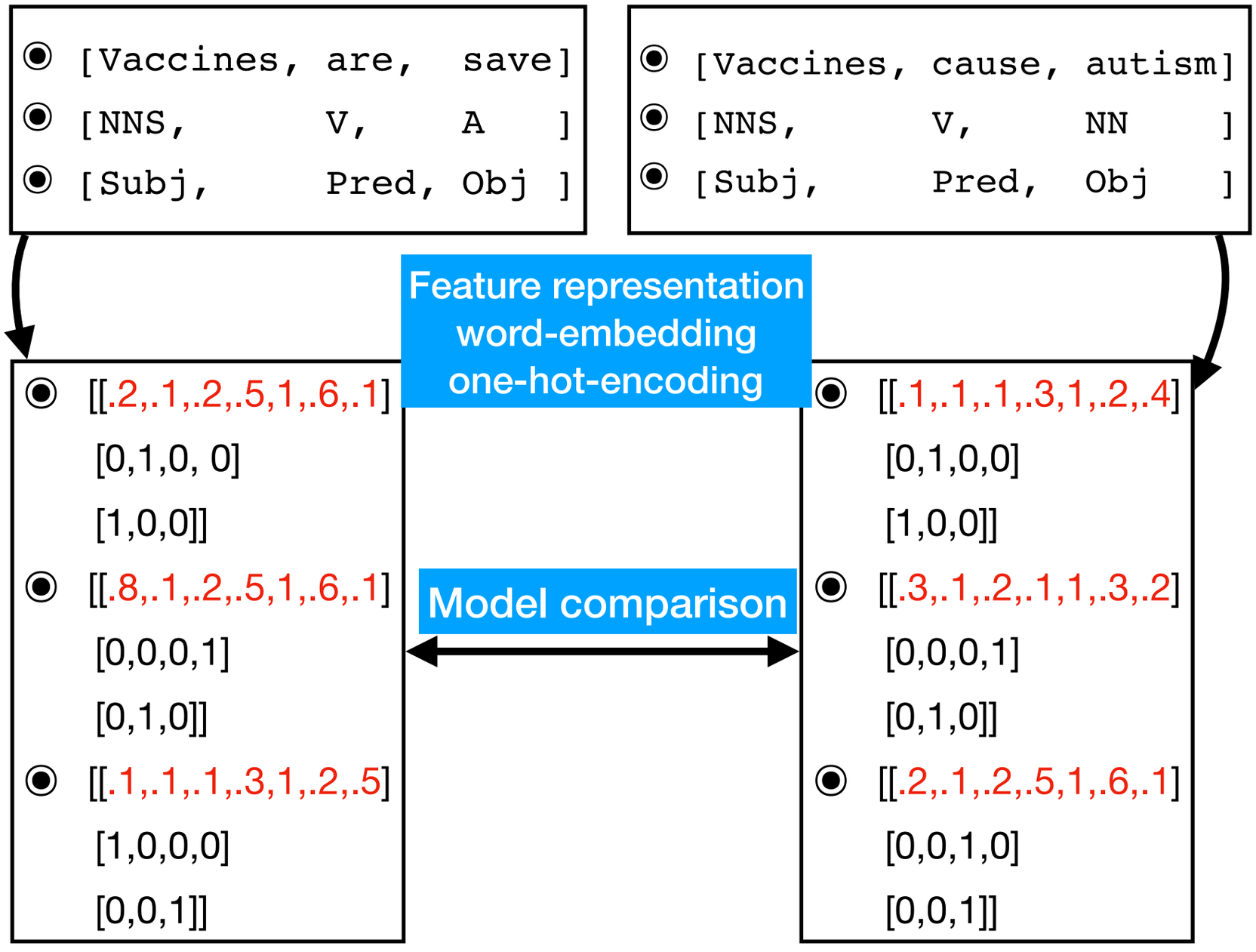
```
array([-2.79722e-01,  4.18270e-02, -1.84113e-01,
-1.74999e-01,
       3.41771e-01, -2.45552e-01, -9.80010e-02, -5.23456e-01,
       1.25658e-01,  5.60530e-02, -2.36231e-01,  1.59386e-01,
      -7.44650e-02, -1.02767e-01,  2.82295e-01,  5.69800e-02,
       1.49481e-01, -1.35064e-01, -7.17500e-02,  4.90650e-02,
      -2.56283e-01,  3.47652e-01, -6.65500e-03, -3.70058e-01,
      -1.30084e-01, -5.62090e-02, -5.44500e-03, -2.89338e-01,
      -2.74530e-01, -5.11710e-02, -9.45050e-02, -3.46780e-02,
      -3.86790e-02, -1.21387e-01,  6.13910e-02, -2.72660e-02,
      -1.51958e-01, -1.80081e-01,  9.48250e-02,  1.23013e-01,
       1.92840e-01, -2.00165e-01,  2.13930e-01,  1.46464e-01,
      -2.79190e-01, -1.09058e-01,  1.18088e-01, -2.45300e-02,
etc…. 400 dominations
```

```
gensimmodel.get_vector('rome').size = 400
```

Combining one-hot-encoding & word embedding

- **Feature space**
- vocabulary = [**word-embedding**]
- PoS = [A, NNS, NN, V]
- Dependencies = [Subj, Pred, Obj]
- Vector dimensions
  - [[300 embedding] +
    [0,0,0,0,0] +
    [0,0,0]]

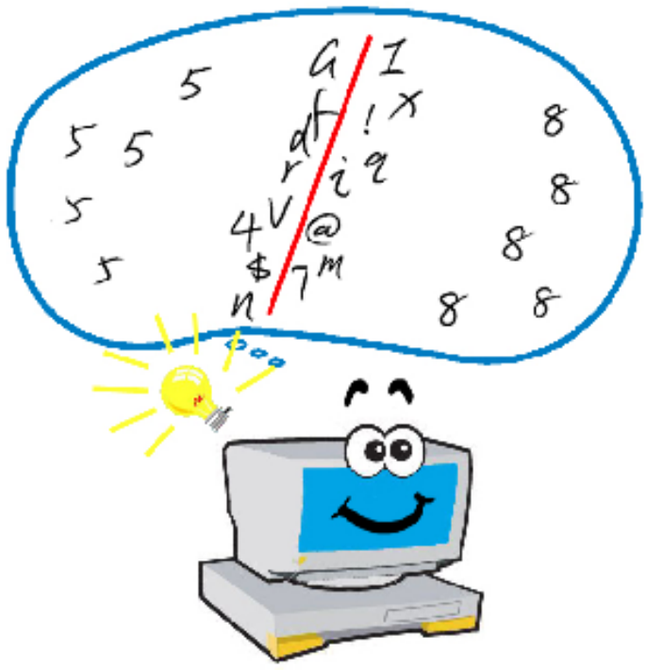
- most similar to [.2,.1,.2,.5,1,.6,.1]
- medicine(s), antibiotics, penicillin, medication, insulin, treatment, injection, prescription, drug(s), therapy

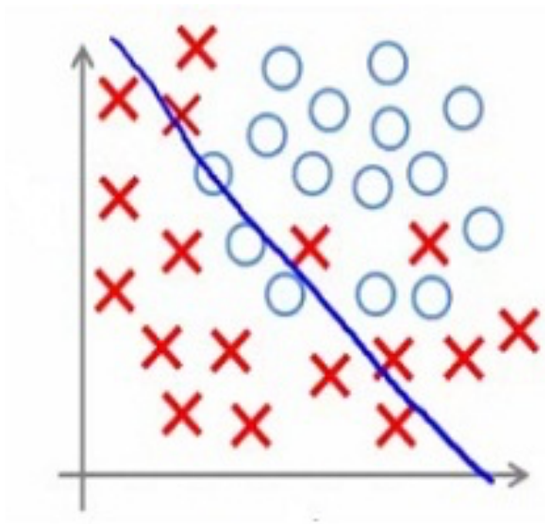


Machine learning

- Is about fitting the data
- Adjust model to avoid errors and make the right predictions
- Problems:
    - Data sparseness (underestimated variation and dynamics)
    - Overfitting: you learn specifics from the training data that should not be learned

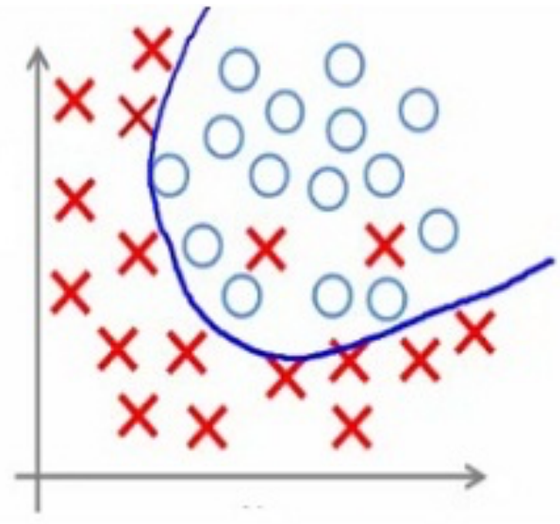
Fitting the data: right - wrong



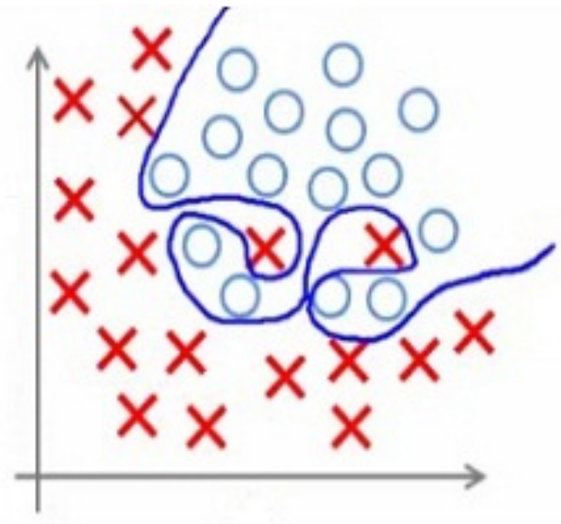


Under-fitting

(too simple to explain the variance)



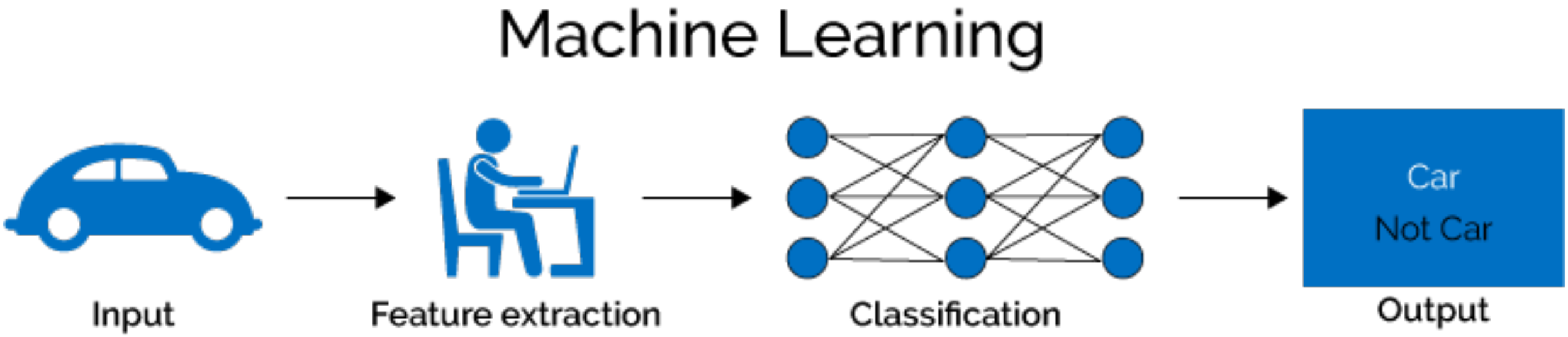
Appropriate-fitting

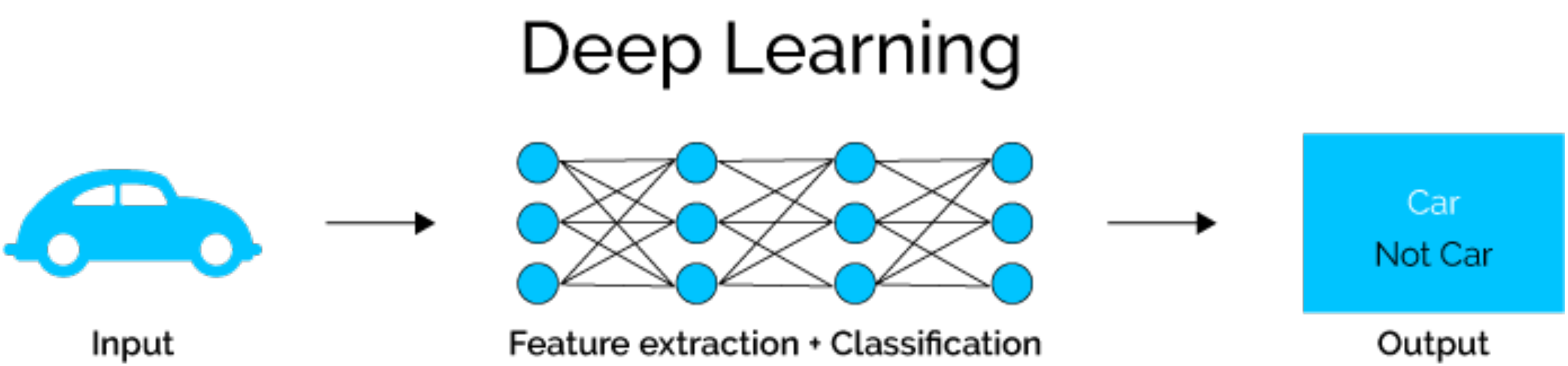


Over-fitting

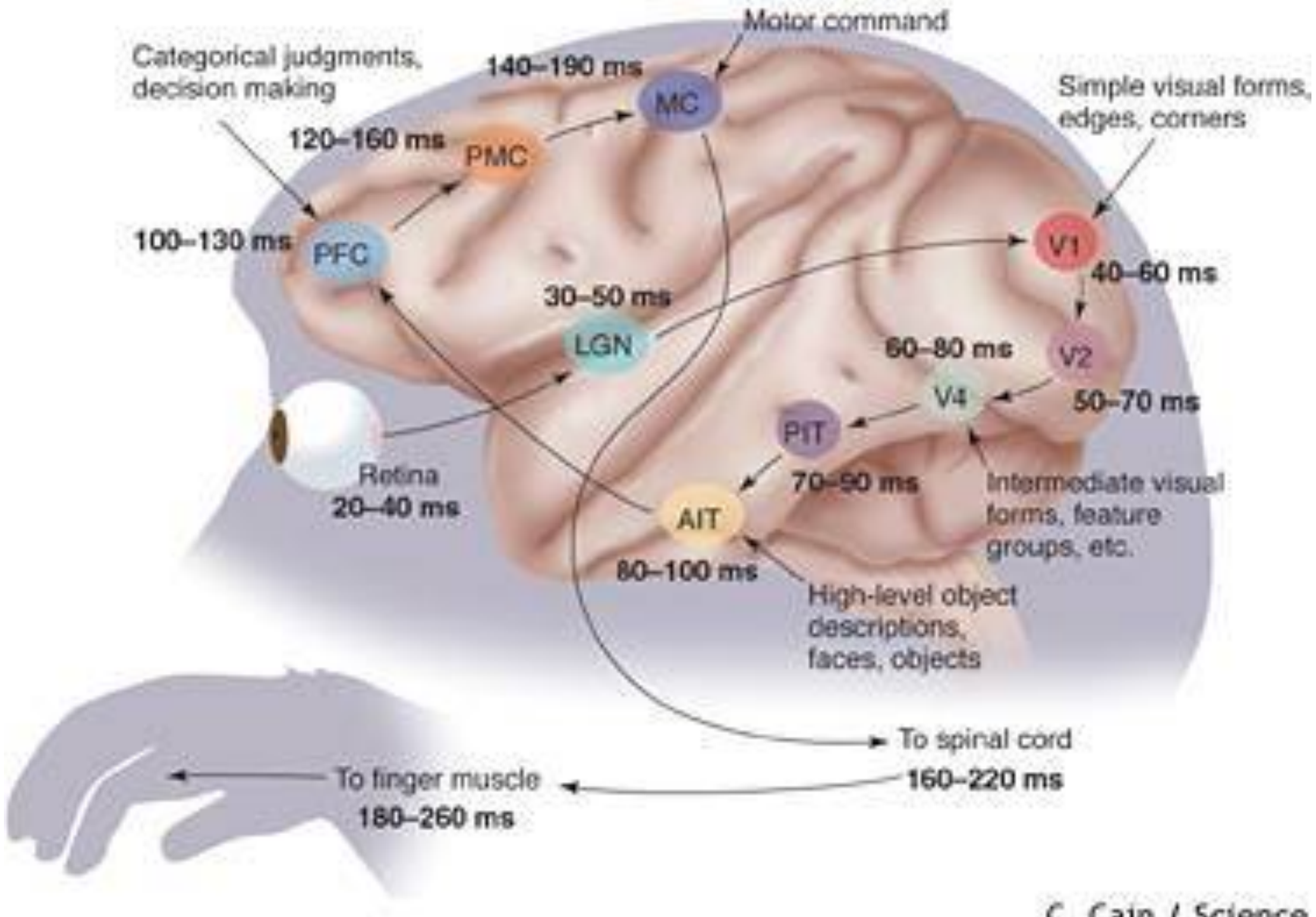
(forcefitting -- too good to be true)

Deep learning



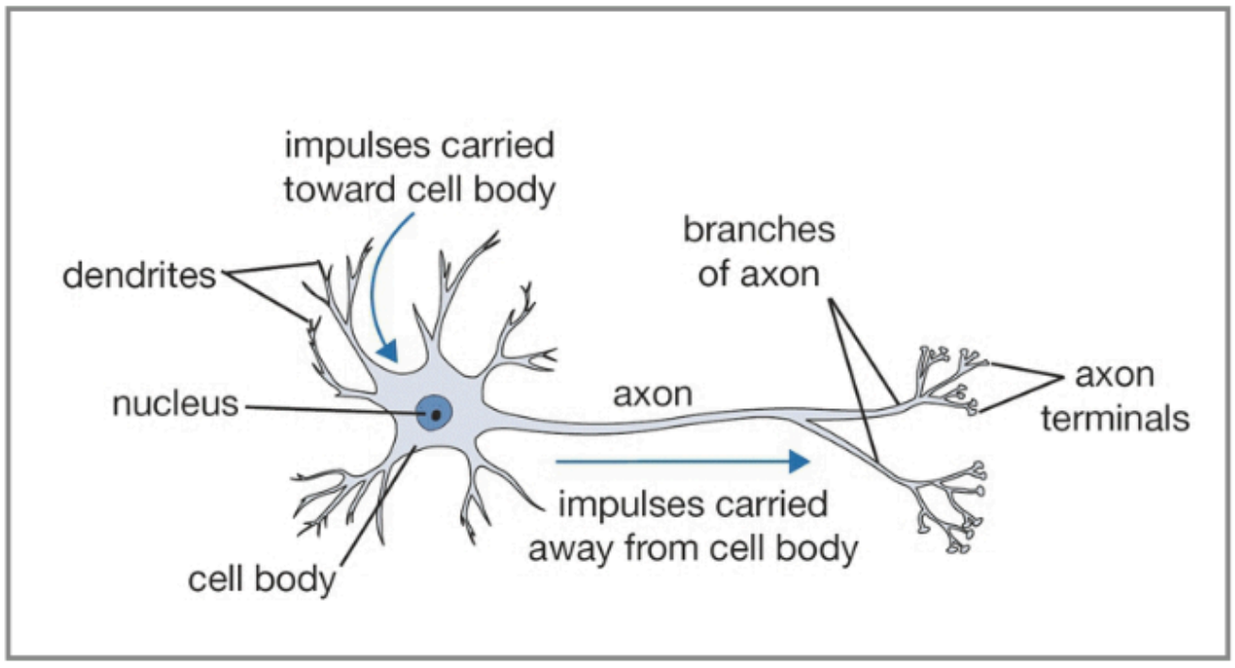


Abstraction in the brain



C. Cain / Science

Brain analogy



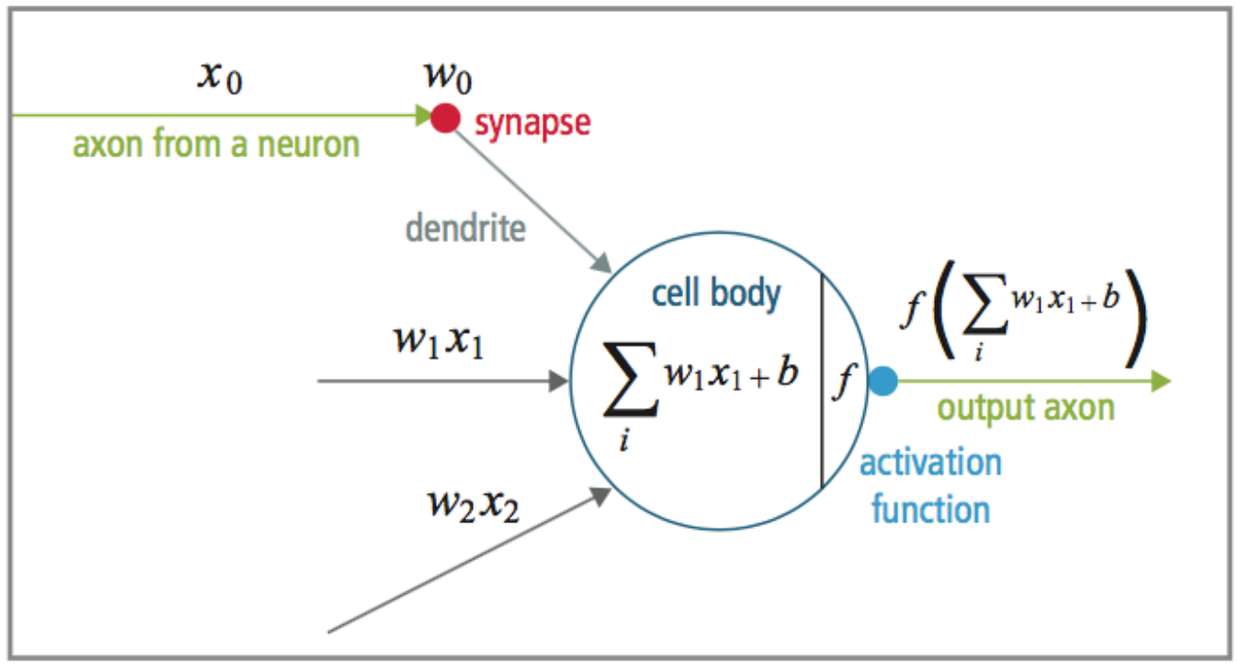
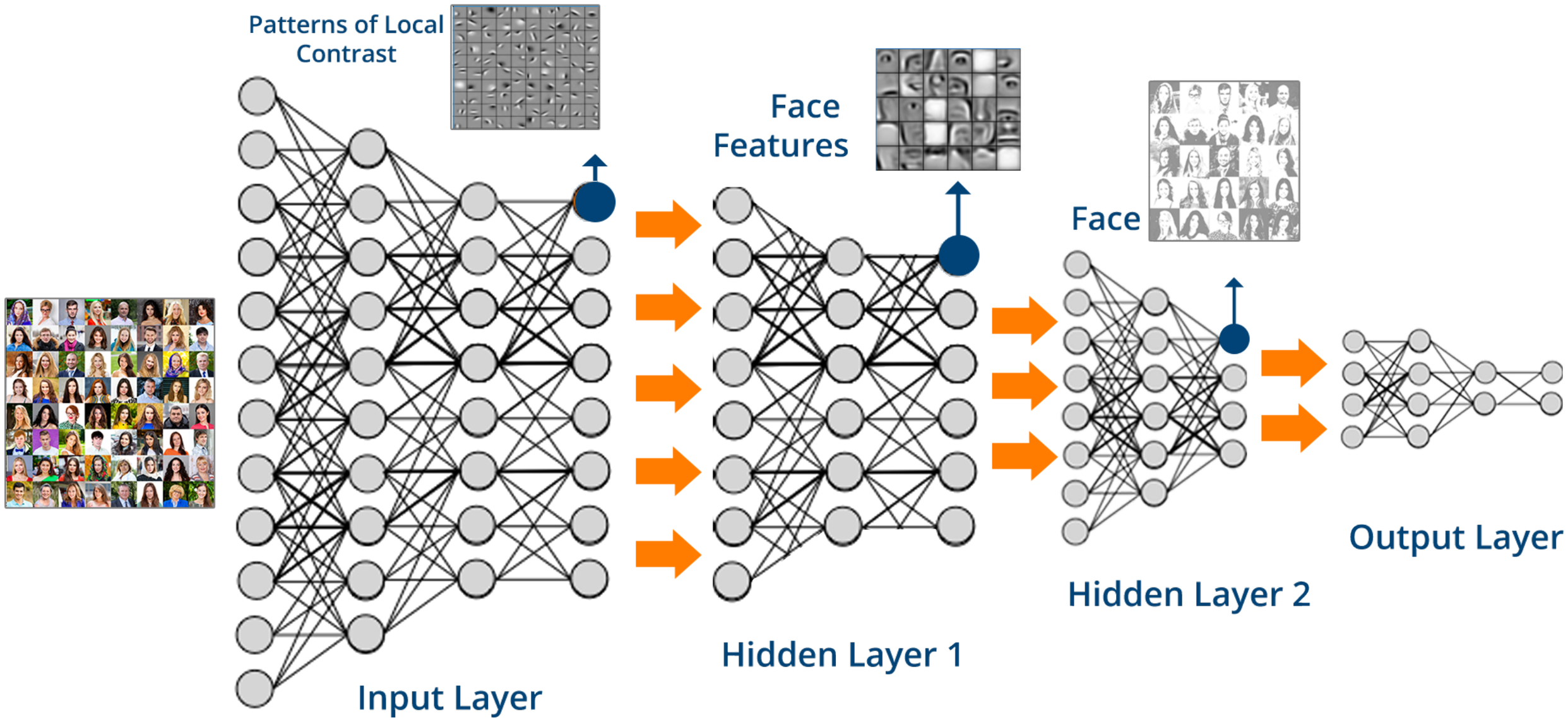


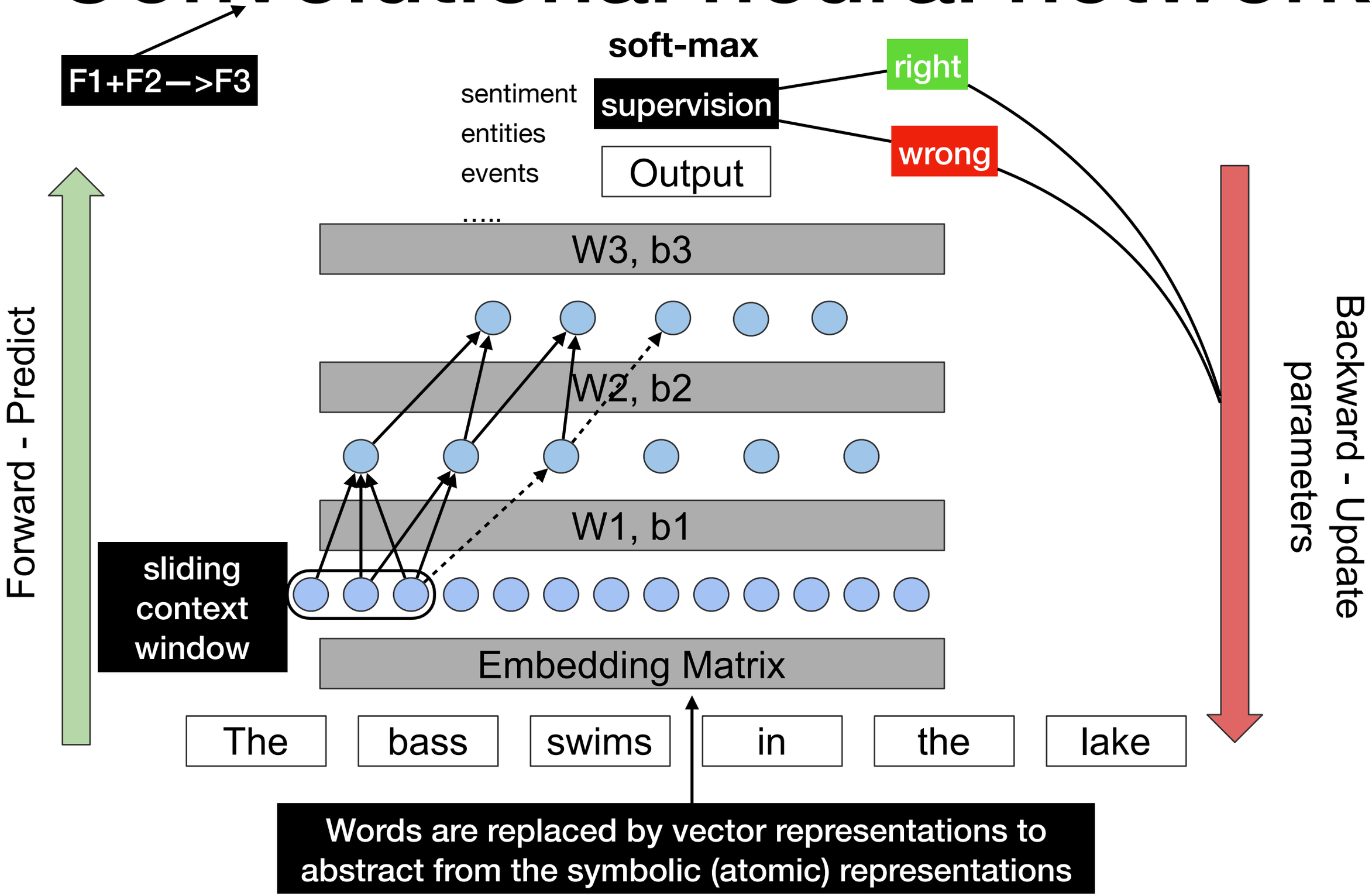
Figure 3: Illustration of a biological neuron (left) and its mathematical model (right) [2].

By: Andrej Karapthy 2015

Deep learning for vision



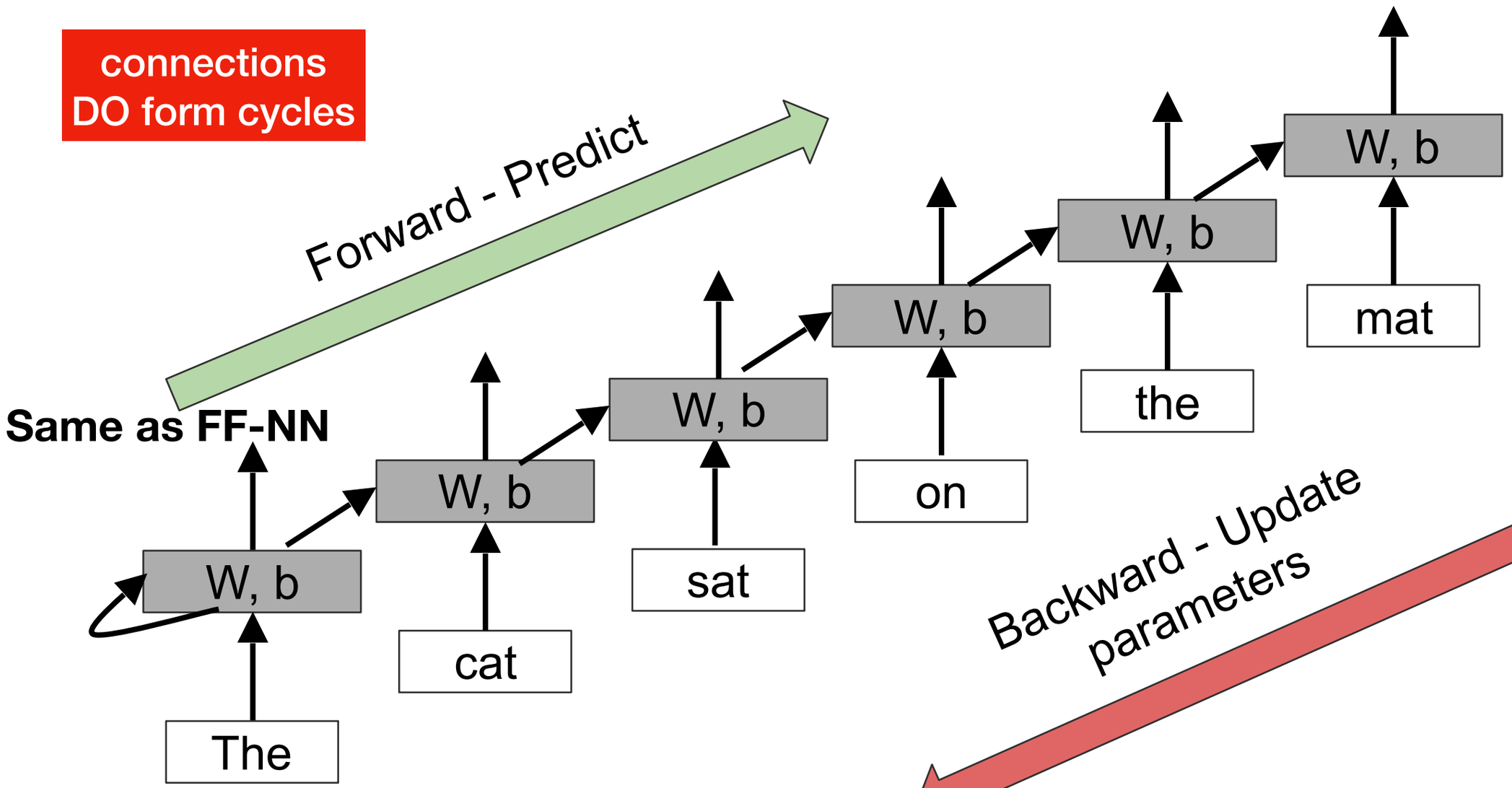
Convolutional neural network



More details

Recurrent neural network

connections
DO form cycles



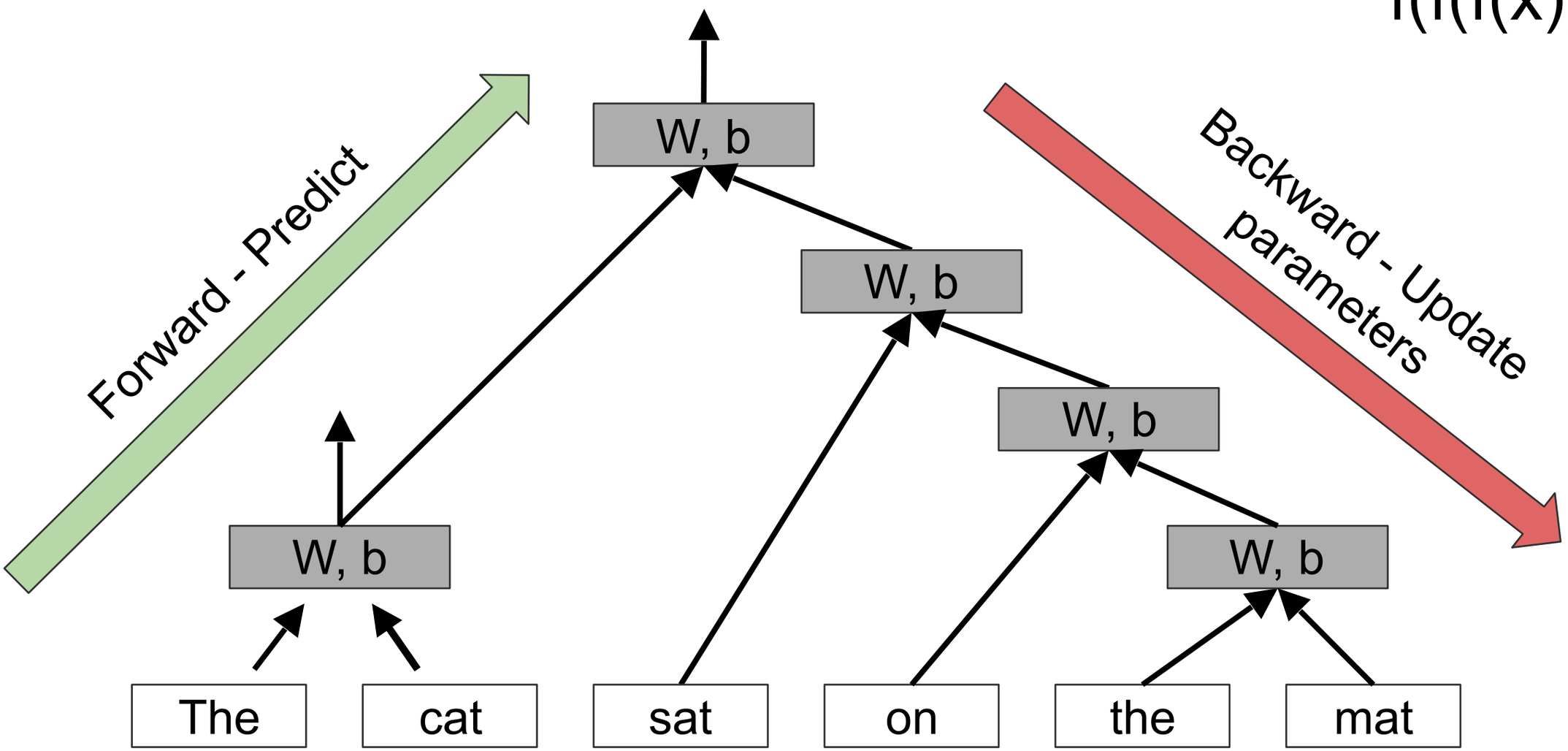
Sequential feed through text, where the previous forward (upward) result feeds into the next sequence input. Predictive value from left to right.

Phrase Classification, Machine Translation, Chat, Autocompletion

More details

Recursive neural network

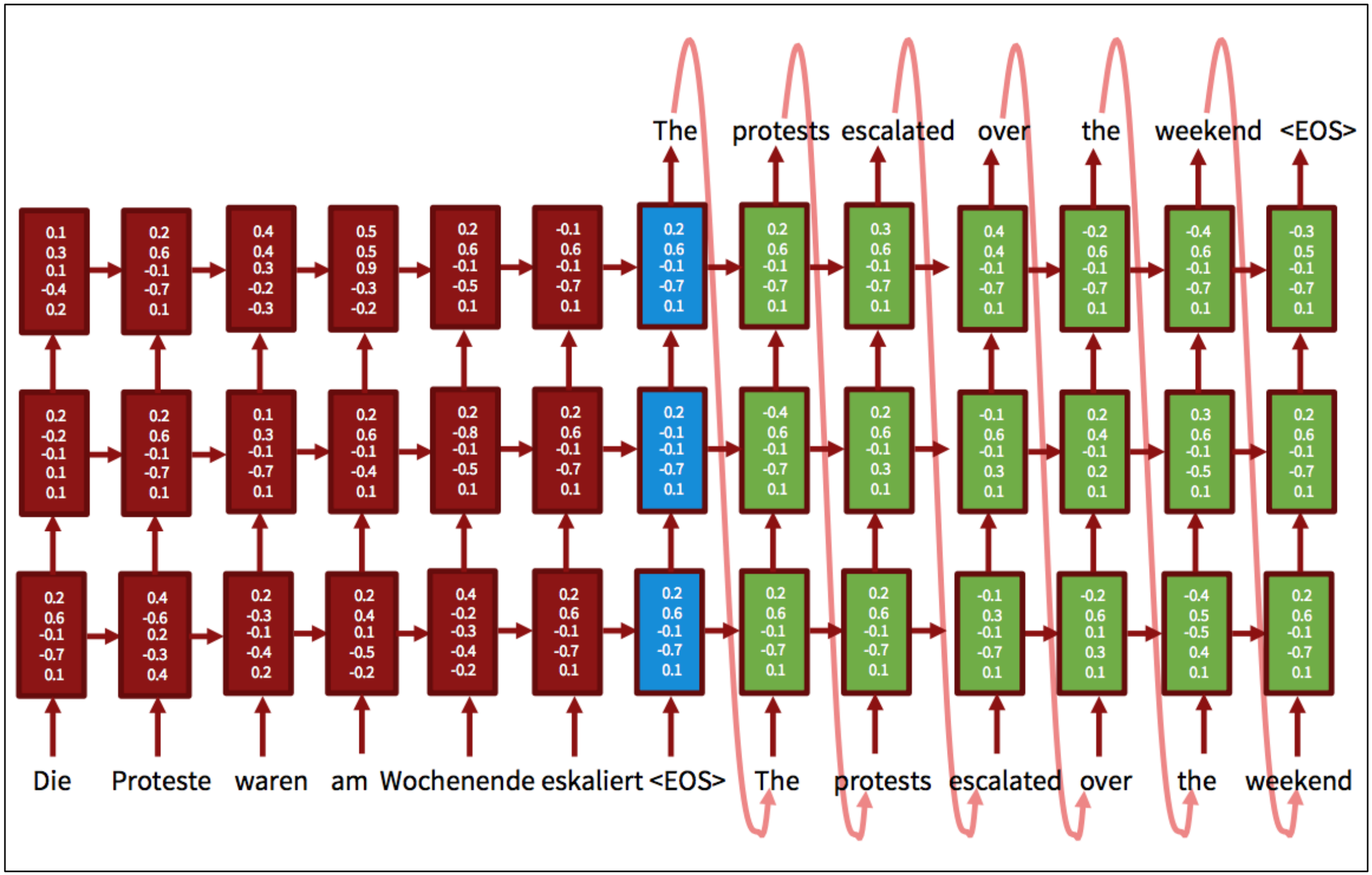
f(x), f(f(x)),
f(f(f(x))),...



Learns which n-grams are more likely and are to be combined into convolutional result but also derives a vector representation of this result for sequence prediction

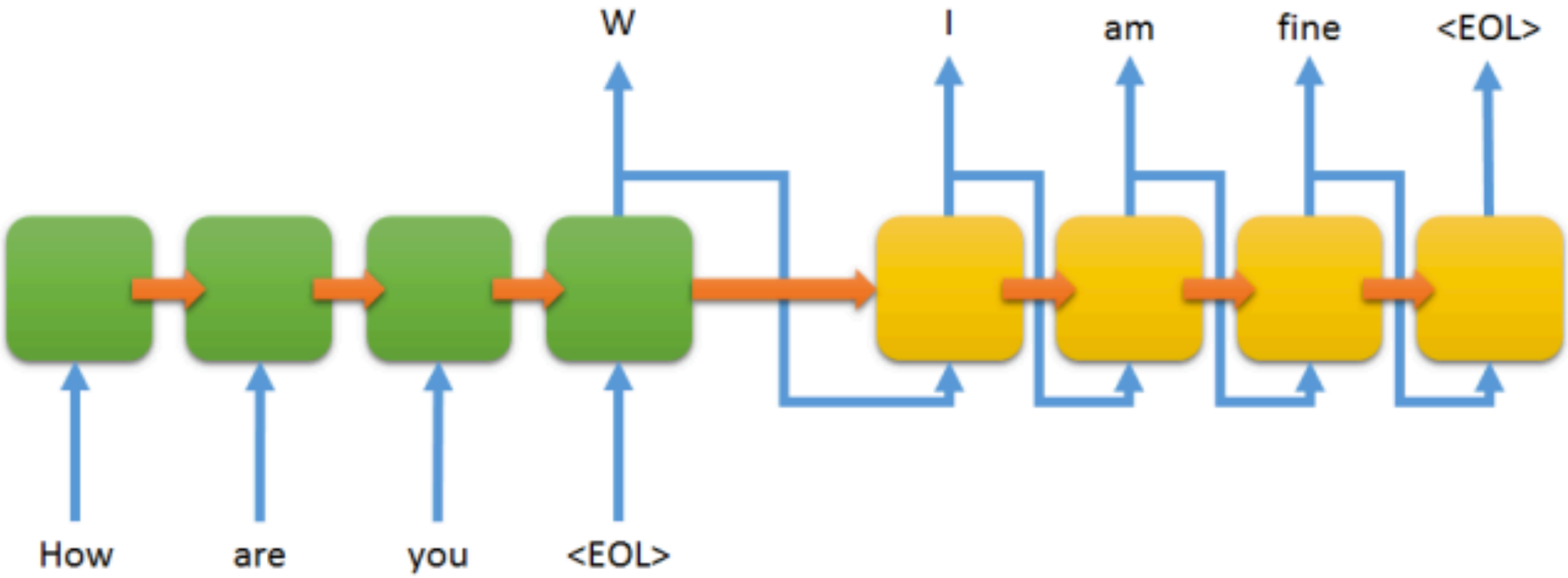
Which words group together (phrase structures) and creates a representation for the group: syntactic parsing, sentence-sentiment

Machine translation: long-short-term-memory



Google Neural MT

Long short-term memory



Encoder is a Neural Network that captures a complete sequence

Question

Source language

…and passes this as the input to the Decoder to predict a sequence

Answer

Target language

Vinyals & Le 2015

Movie subtitles
1.3 billion words

Google's conversatiemachine

- Human: who is skywalker ?
- Machine: he is a hero .
- Human: who is bill clinton ?
- Machine: he 's a billionaire .
- Human: is sky blue or black ?
- Machine: blue .
- Human: does a cat have a tail ?
- Machine: yes .
- Human: does a cat have a wing ?
- Machine: no
- Human: can a cat fly ?
- Machine: no .
- Human: how many legs does a cat have ?
- Machine: four , i think .

- **Human: how many legs does a spider have ?**
- **Machine: three , i think .**
- **Human: how many legs does a centipede have ?**
- **Machine: eight .**
- Human: what is the color of the sky ?
- Machine: blue .
- **Human: what is the color of water ?**
- **Machine: water .**
- Human: what is the color of blood ?
- Machine: it is the same as a black eye .
- Human: how much is two plus two ?
- Machine: four .
- **Human: how much is ten minus two ?**
- **Machine: seventy - two .**

Overall NLP approach

- Complex problem is broken down into a number of smaller problems

- Simple, structural problems solved first and higher-level semantic tasks are solved later, using the output of earlier modules as input:

  - pipeline architecture with dependencies across modules

- For each problem different techniques:

  - knowledge-base & rules (linguistic knowledge)

  - machine learning (supervised and unsupervised), data driven

Knowledge or data driven

Somebody needs to build the lexicon & rules

Somebody needs to annotate training data

Knowledge-Based	Machine Learning Systems
• based on hand-coded rules	• use statistics or other machine learning
• developed by NLP specialists	• developers do not need NLP expertise
• make use of human intuition	• requires large amounts of training data
• easy to understand results	• cause of errors is hard to understand
• development could be very time consuming	• development is quick and easy
• changes may require rewriting rules	• changes may require re-annotation

Table 2.1: Summary of Knowledge-Based vs Machine Learning Approaches to NLP

Lexical database or ontology

Word	Properties
Apple	Name, Company
Samsung	Name, Company
infringe	Verb, negative
its	Pronoun, possessive

Rules

If
Company (w) &
Negative (w+1) &
Event (w+1)
then
NEG

Samsung infringed patents

Data

Prices are high and service is low [neg]

Prices are low, service is high [pos]

Low ceilings and steep stairs

Features

[1,0,0,0,1,0,1] neg

[1,0,1,1,0,0,1] pos

[1,0,0,0,1,0,1] ?

Model

Weights

Types of machine learning in NLP

explicit feature representation

Decision trees (DT)	closest to rules
Hidden Markov Models (HMM)	part-of-speech sequences
Naive Bayes (NB)	bag-of-words
Maximum Entropy (ME)	class distribution
Support Vector Machine (SVM)	data delineation
K-nearest-neighbors (KNN)	distance in feature space
Conditional Random Fields (CRF)	sequential dependencies

Neural networks (NN)	hidden feature representation

How to trust data systems?

- Evaluation regime
- Train and test data
- Evaluation metrics

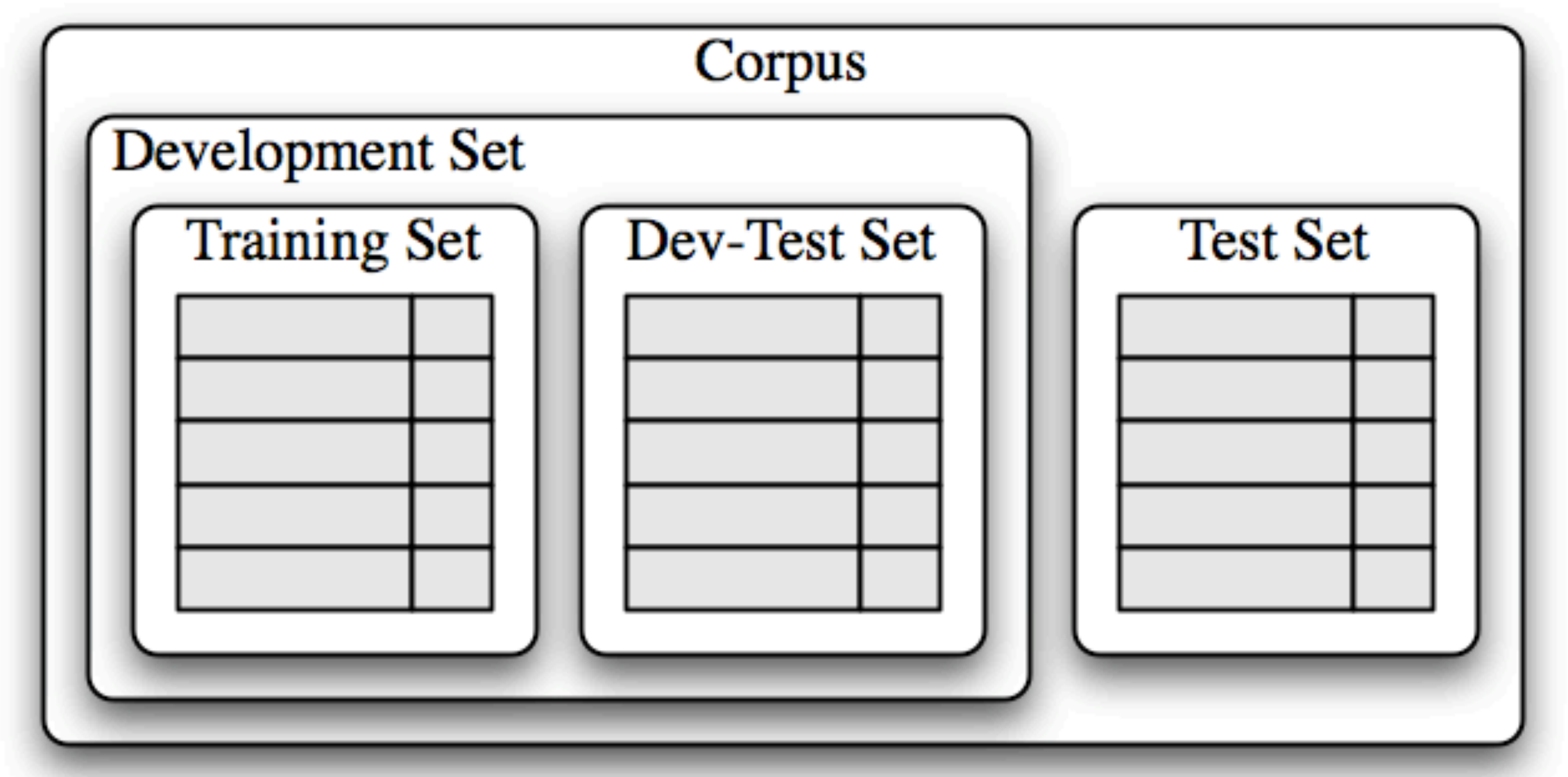
Evaluation

- Evaluation regimes:
  - Conference of Natural Language Learning (CoNLL): http://www.conll.c
  - Automatic Content Extraction (ACE): https://www.ldc.upenn.edu/collab projects/ace/annotation-tasks-and-specifications
  - SemEval competitions: https://en.wikipedia.org/wiki/SemEval
- Data sets:
  - training data (if machine learning is to be used)
  - development data
  - test data or (10-)fold cross-validation
- Precision, recall and F-measure

CoNLL editions

2018 - Brussels, Belgium
2017 - Vancouver, Canada
2016 - Berlin, Germany
2015 - Bejing, China
2014 - Baltimore, MD, USA
2013 - Sofia, Bulgaria
2012 - Jeju Island, Korea
2011 - Portland, OR, USA
2010 - Uppsala, Sweden
2009 - Boulder, CO, USA
2008 - Manchester, UK
2007 - Prague, Czech Republic
2006 - New York City, NY, USA
2005 - Ann Arbor, MI, USA
2004 - Boston, MA, USA
2003 - Edmonton, Canada
2002 - Taipei, Taiwan
2001 - Toulouse, France
2000 - Lisbon, Portugal
1999 - Bergen, Norway
1998 - Sidney, Australia
1997 - Madrid, Spain

Evaluation framework



Folded cross validation

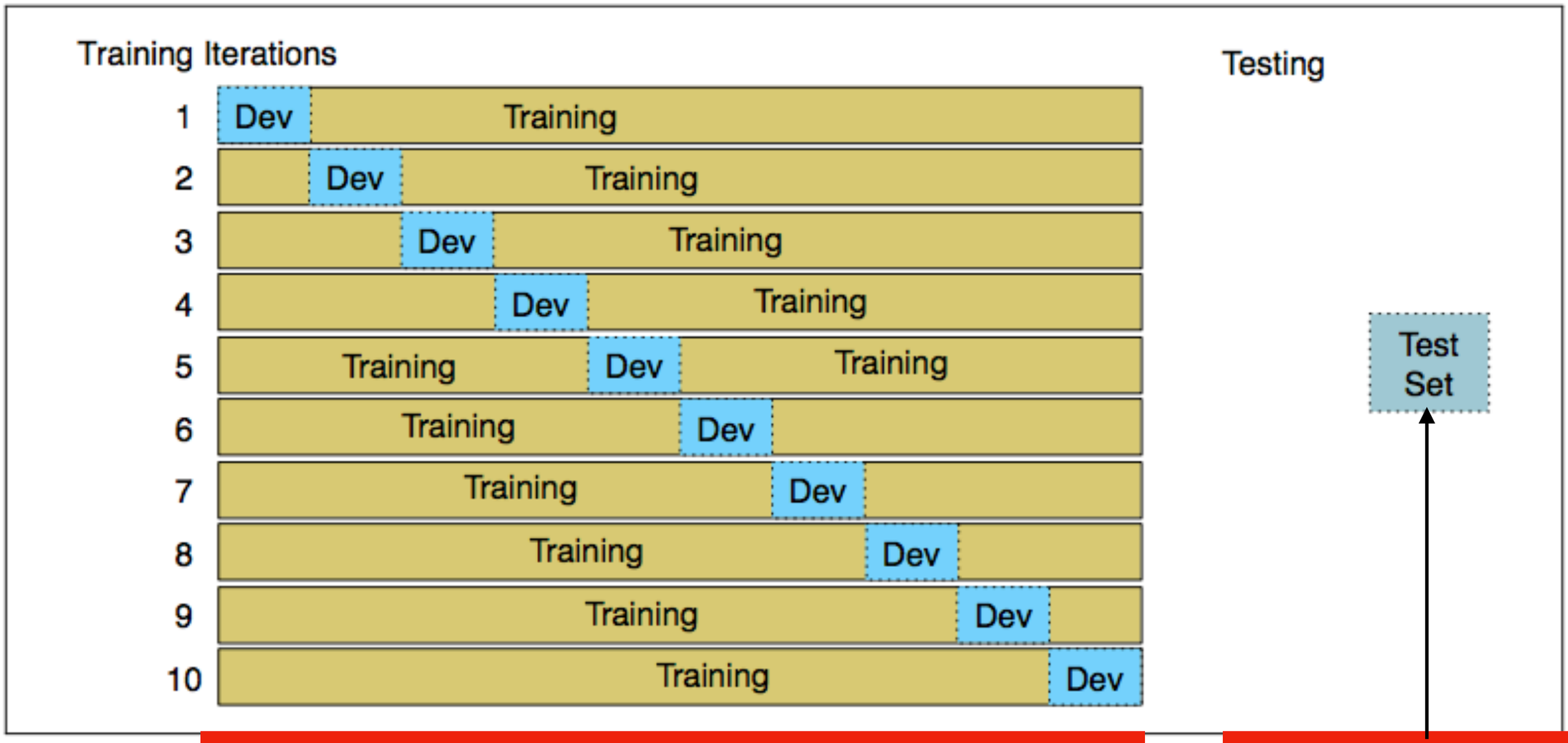


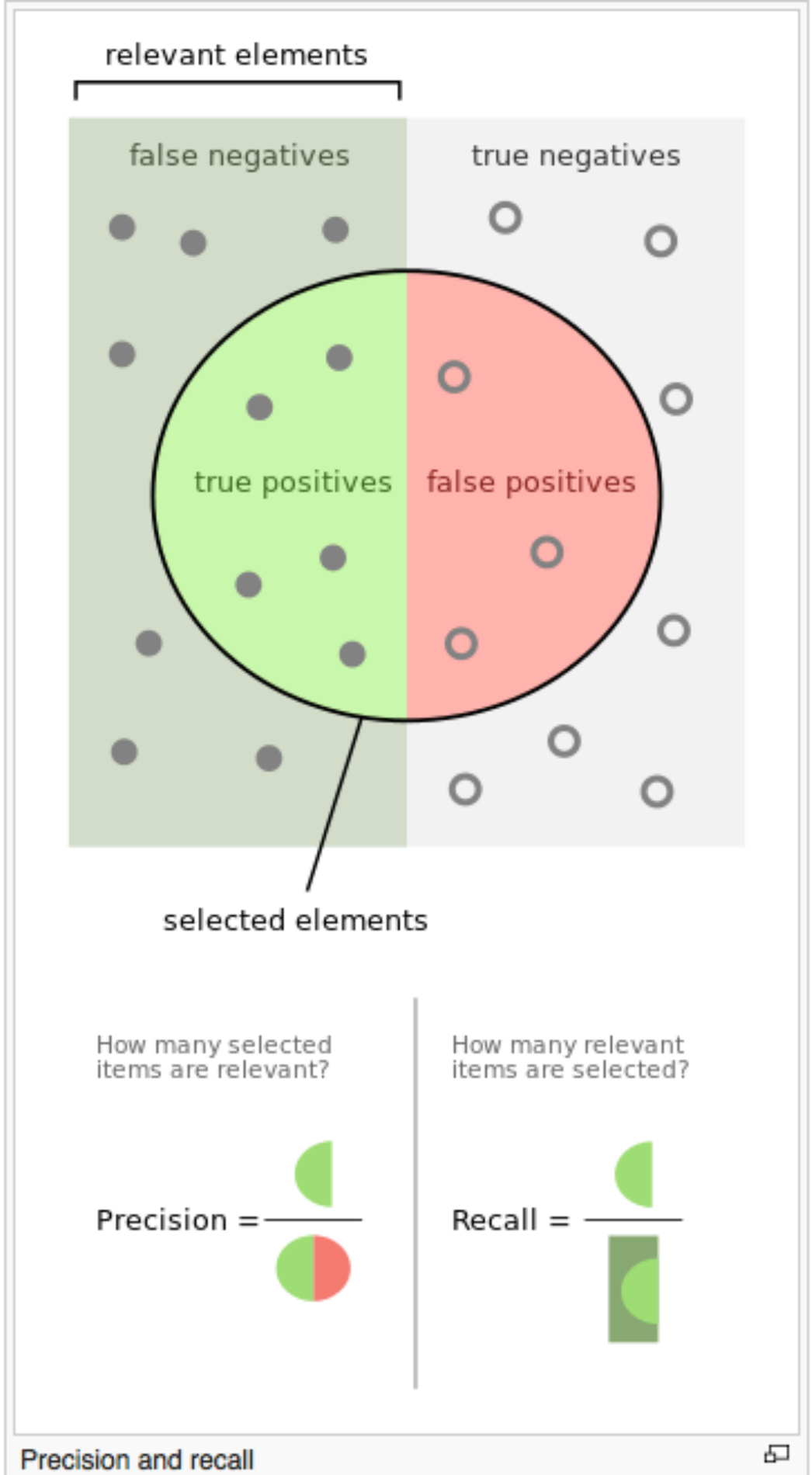
Figure 6.7 10-fold crossvalidation

Precision, Recall, F1-measure

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

$$F_1 = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}.$$



Precision and recall

Contingency table/confusion matrix

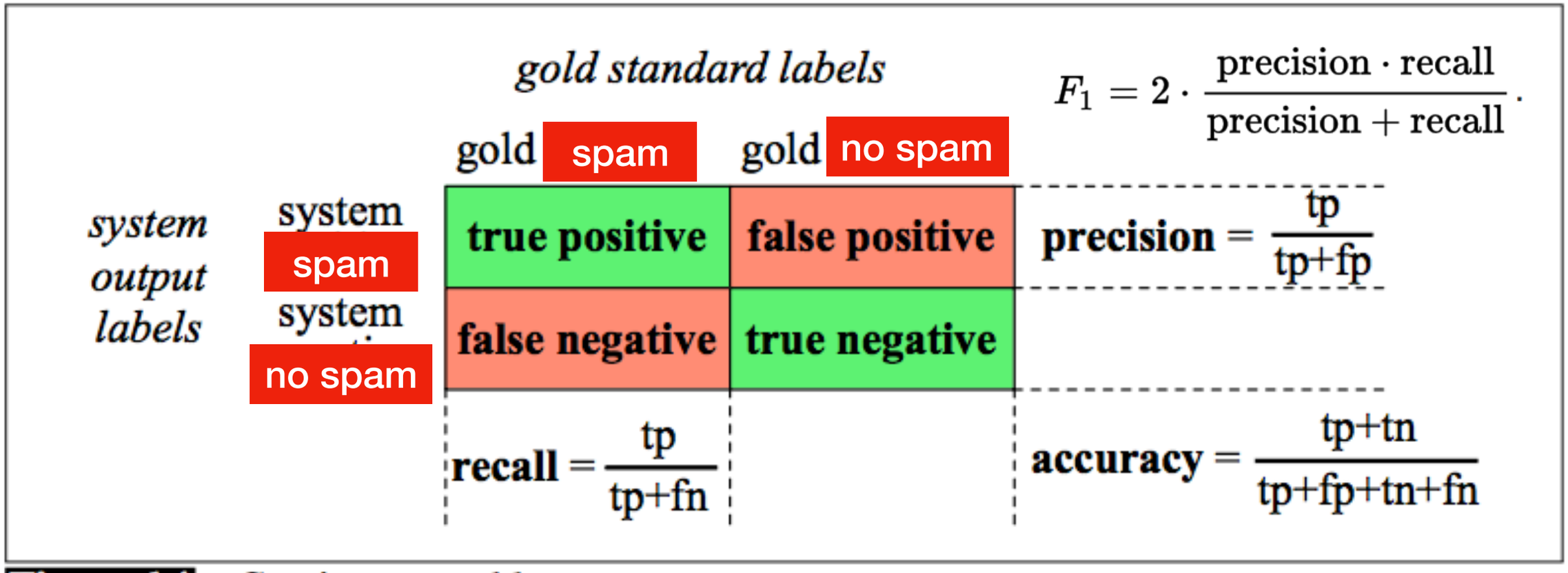
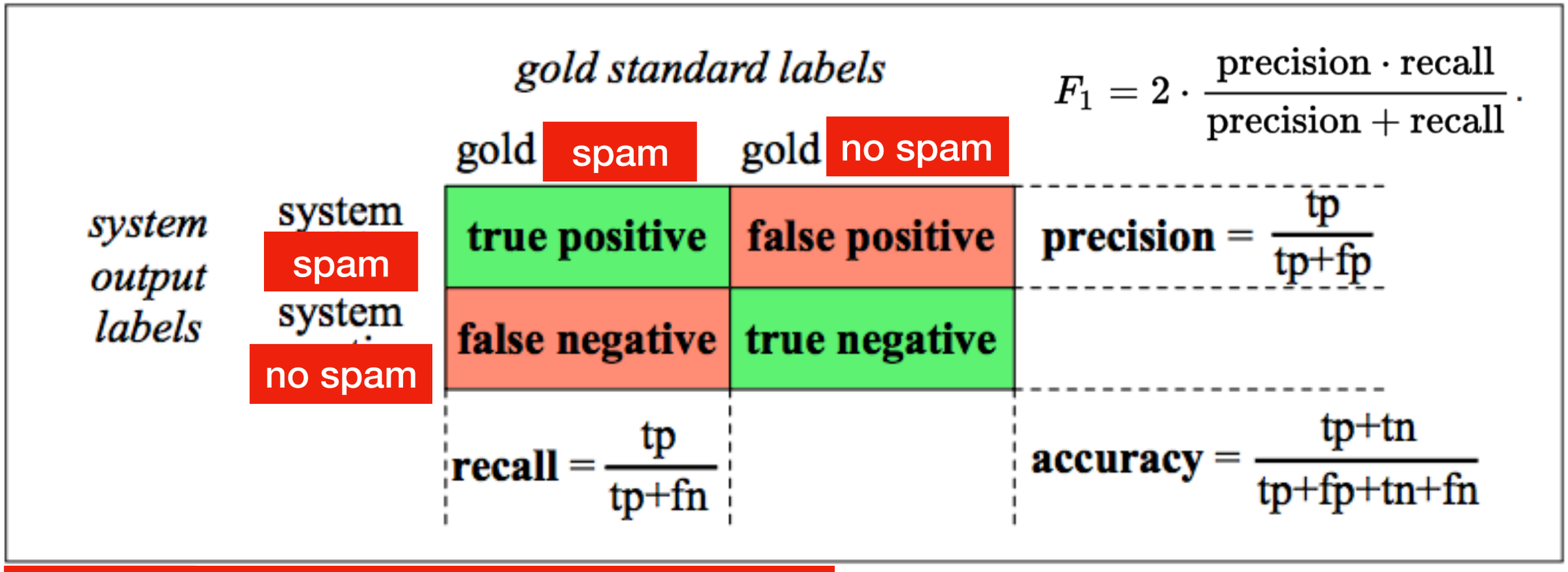


Figure 6.4 Contingency table

Binary classification

accuracy, precision, recall, harmonic mean (F1)

Contingency table/confusion matrix



Gold data: 100 mails, 20 spam mails
System: detects 10 spam mails, 5 correct

$$\frac{5\text{ TP} + 75\text{ TN}}{5\text{ TP} + 5\text{ FP} + 75\text{ TN} + 15\text{ FN}} = 0.8 \text{ accuracy}$$

Accuracy only works for balanced data!!

$$\frac{5\text{ TP}}{5\text{ TP} + 5\text{ FP}} = 0.5 \text{ precision}$$

$$\frac{5\text{ TP}}{5\text{ TP} + 15\text{ FN}} = 0.25 \text{ recall}$$

$$\frac{2 \times 0.5 \times 0.25}{0.5 + 0.25} = 0.33 \text{ F1}$$

Multiclass: spam, urgent, normal

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

Figure 6.5 Confusion matrix for a three-class categorization task, showing for each pair of classes (c_1, c_2), how many documents from c_1 were (in)correctly assigned to c_2

- Build separate binary classifiers for each class using positive cases for *c* and all cases for *not-c (!c)*
  - **any-of**: run all classifiers and allow multiple results
  - **one-of** or multinomial: run all classifiers and take the best

Based on: Jurafsky & Martin: https://web.stanford.edu/~jurafsky/slp3/

Micro & Macro averaging

Class 1: Urgent

	true urgent	true not
system urgent	8	11
system not	8	340

$$\text{precision} = \frac{8}{8+11} = .42$$

Class 2: Normal

	true normal	true not
system normal	60	55
system not	40	212

$$\text{precision} = \frac{60}{60+55} = .52$$

Class 3: Spam

	true spam	true not
system spam	200	33
system not	51	83

$$\text{precision} = \frac{200}{200+33} = .86$$

Pooled

	true yes	true no
system yes	268	99
system no	99	635

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

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

Figure 6.6 Separate contingency tables for the 3 classes from the previous figure, showing the pooled contingency table and the microaveraged and macroaveraged precision.

- macro: if performance is balanced across classes
- micro: if nr. of cases is balanced across classes

Based on: Jurafsky & Martin: https://web.stanford.edu/~jurafsky/slp3/

Contingency table for sentiment

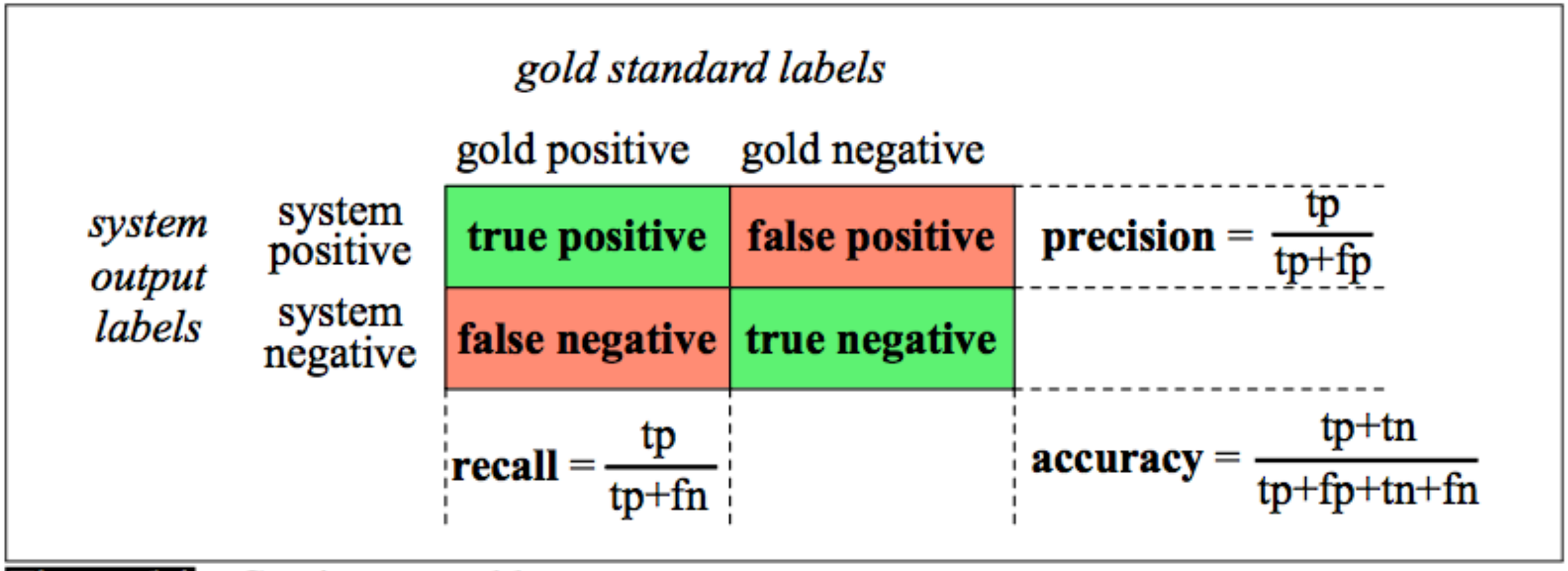


Figure 6.4 Contingency table

Gold data: 100 reviews, 40 positive, 30 negative

Contingency table for sentiment

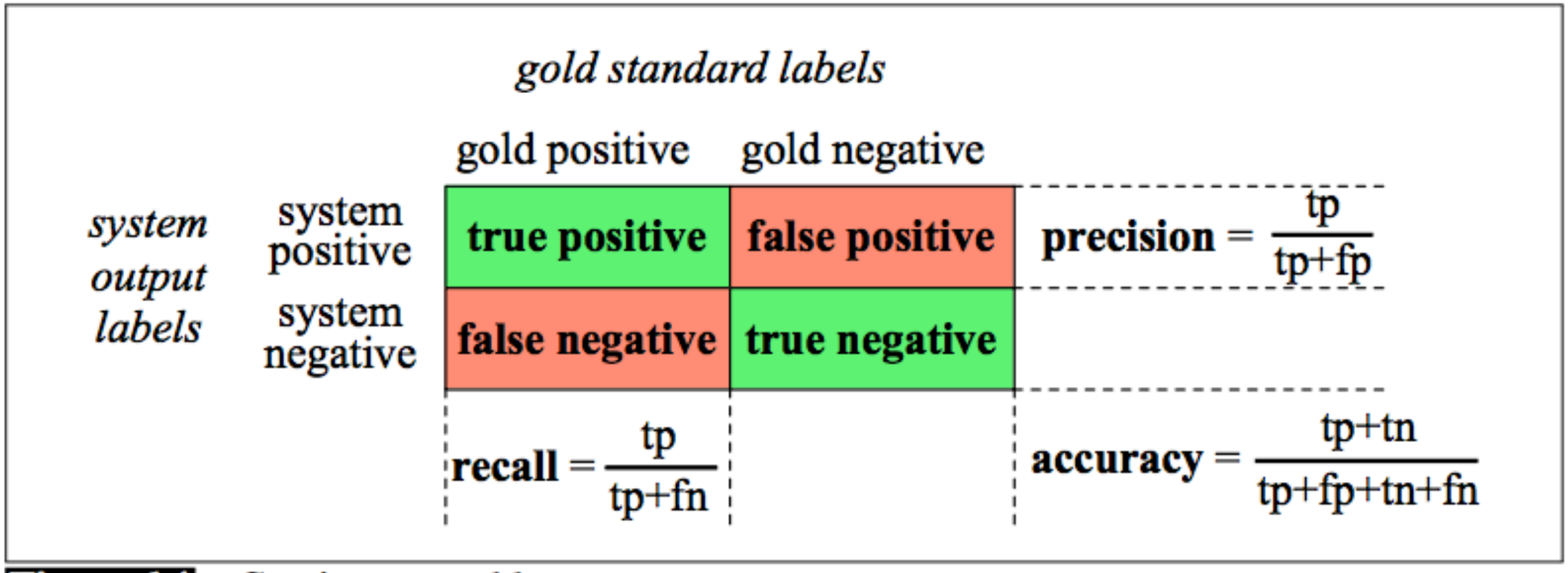


Figure 6.4 Contingency table

Gold data: 100 reviews, 40 positive, 30 negative

positive: 35 TP + 10 FP

$$\frac{35\text{ TP}}{35\text{ TP} + 10\text{ FP}} = 0.78 \text{ precision}$$

$$\frac{35\text{ TP}}{35\text{ TP} + 5\text{ FN}} = 0.87 \text{ recall}$$

negative: 20TP + 5FP

$$\frac{20\text{ TP}}{20\text{ TP} + 5\text{ FP}} = 0.8 \text{ precision}$$

$$\frac{20\text{ TP}}{20\text{ TP} + 10\text{ FN}} = 0.66 \text{ recall}$$

What matters precision or recall depends on the application

- High precision, low recall:
  - ***Fully automated system*** that takes decisions (self-driving car)
  - Find precisely what you are looking for and ignore the rest: one perfect example is enough to act upon
  - Reduce human work
- High recall, low precision:
  - **No data will slip through**, but a human checks the decision
  - Find everything on a topic and filter afterwards
  - Spot all potential risk, an alert (spam filters, buienradar (shower radar))
  - Requires human work
- Common strategy: first maximise the recall and next improve the precision