Enhancing NLP with Deep Neural Networks

github.com/adarsh0806/ml_workshop_udacity

Outline

Classic NLP

- Text Processing
- Feature Extraction
- Topic Modeling
- **Lab: Topic modeling using LDA**

Deep NLP

- Neural Networks
- Recurrent Neural Networks
- Word Embeddings
- **Lab: Sentiment Analysis using RNNs**

Introduction to NLP

Introduction to NLP

- Communication and Cognition
- Structured Languages
- Unstructured Text
- Applications and Challenges

Communication and Cognition

Language is…

- a medium of communication
- a vehicle for thinking and reasoning

Structured Languages

- Natural language lacks precisely defined structure

Structured Languages

- Mathematics:

$$y = 2x + 5$$

Structured Languages

- Formal Logic:

$$\text{Parent}(x, y) \wedge \text{Parent}(x, z) \rightarrow \text{Sibling}(y, z)$$

Structured Languages

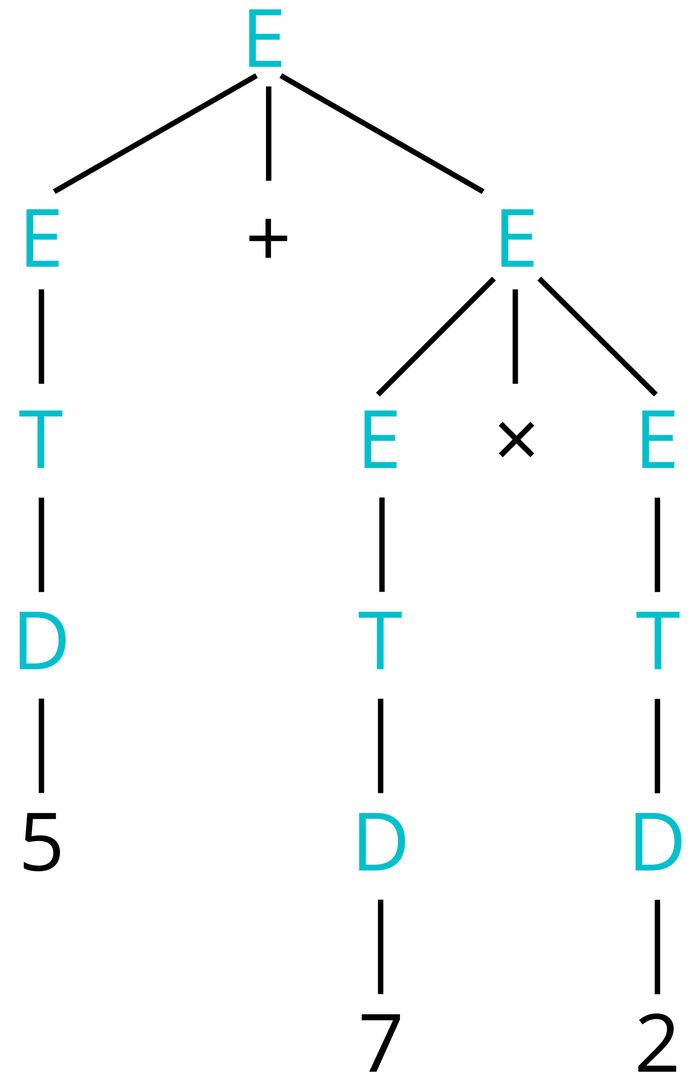
- SQL:

```
SELECT name, email
  FROM users
  WHERE name LIKE 'A%';
```

Grammar

- Arithmetic (single digit):

$$E \rightarrow E{+}E \mid E{-}E \mid E{\times}E \mid E{\div}E \mid (E) \mid D$$
$$D \rightarrow 0 \mid 1 \mid 2 \mid \ldots \mid 9$$



Grammar

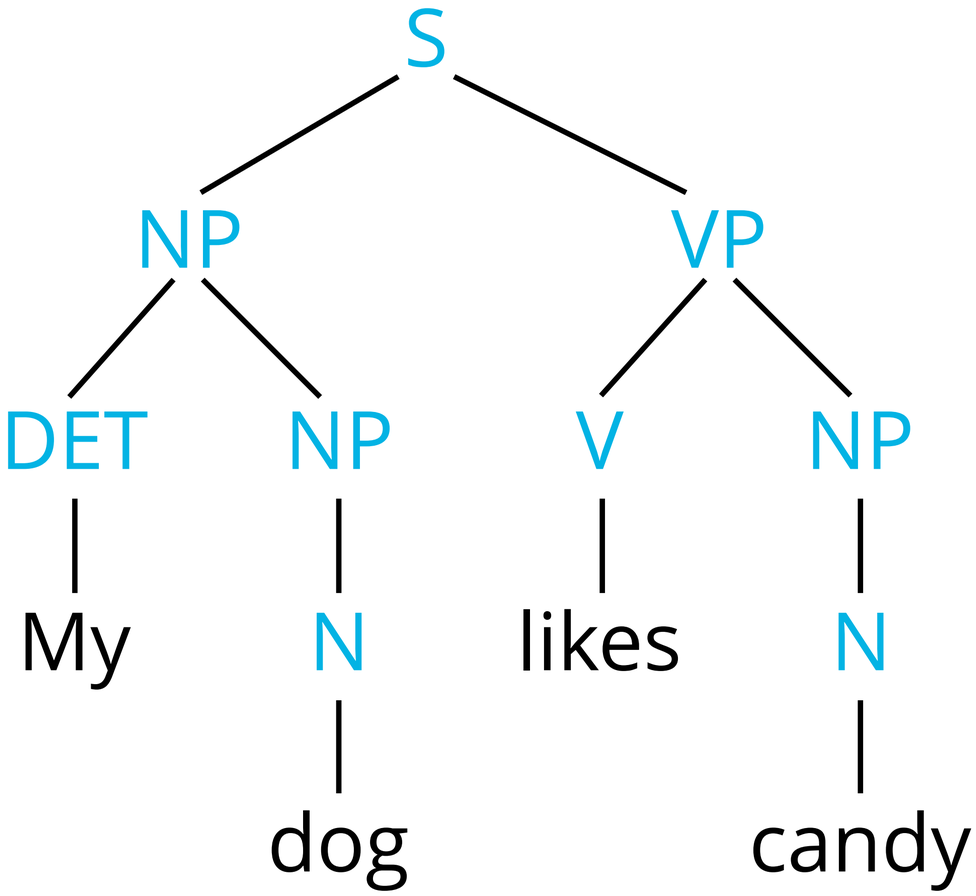
- English sentences (limited):

  S → NP VP

  NP → N | DET NP | ADJ NP

  VP → V | V NP

  …



noun

"Because he was so small, Stuart was often hard to find around the house."

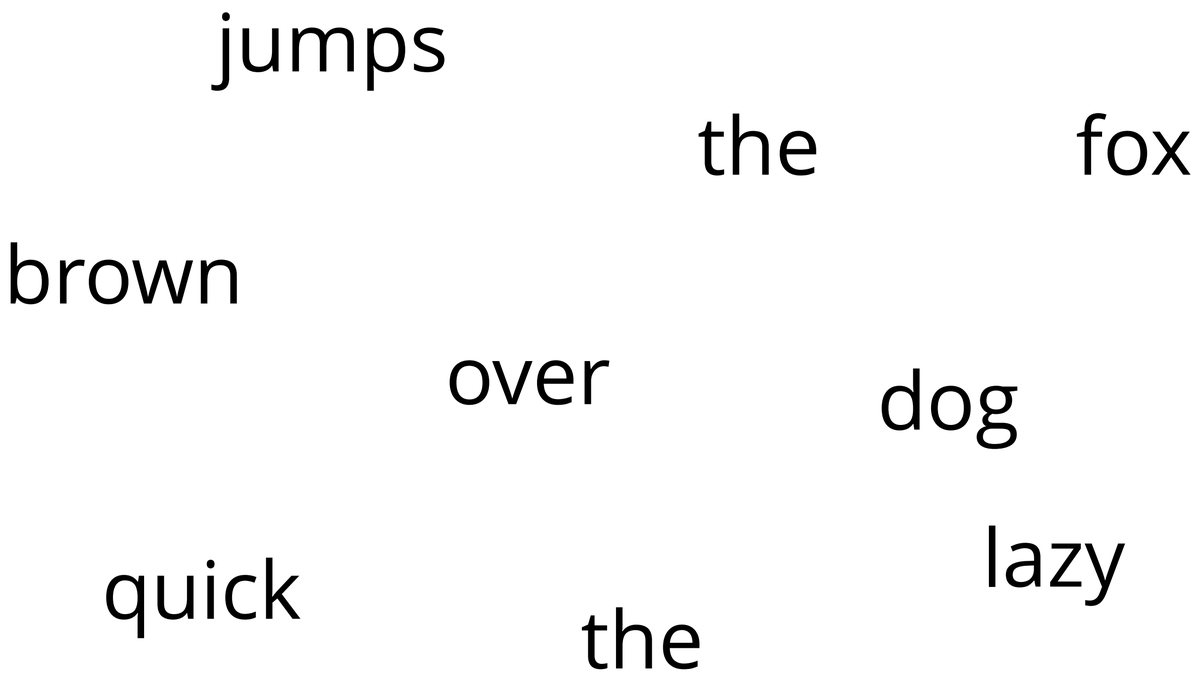
verb

– *Stuart Little*, E.B. White

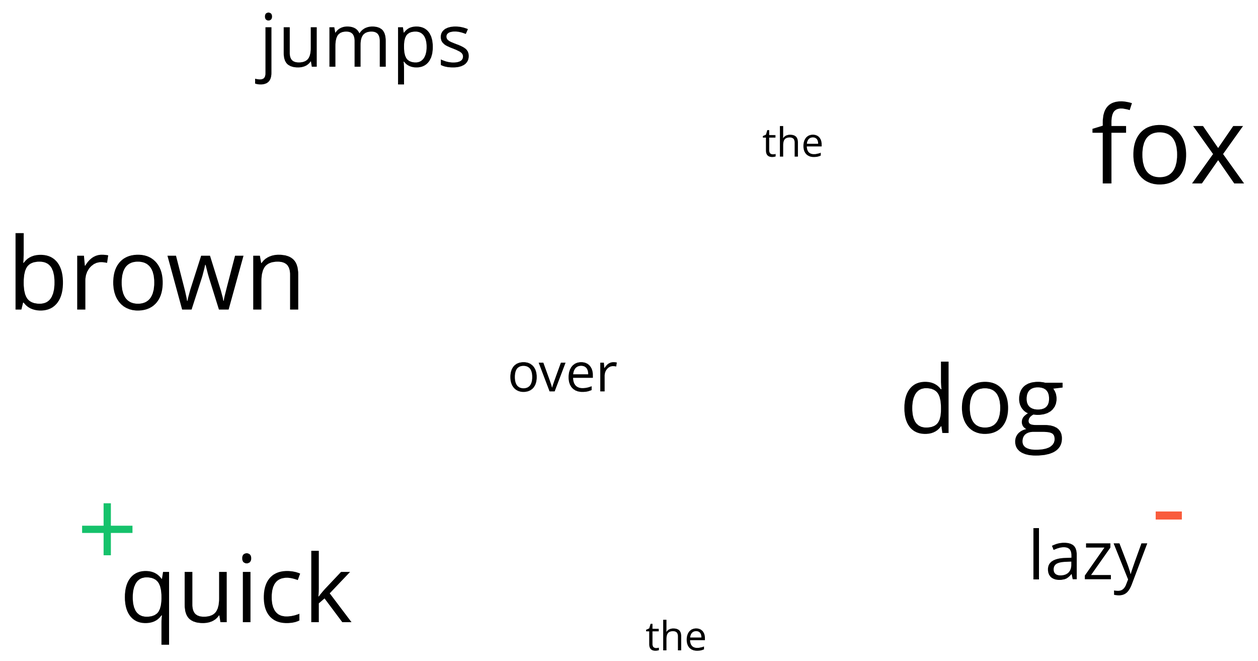
Unstructured Text

the quick brown fox jumps over the lazy dog

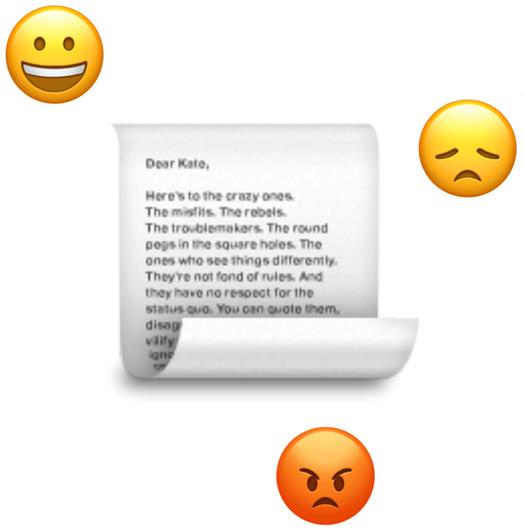
Unstructured Text

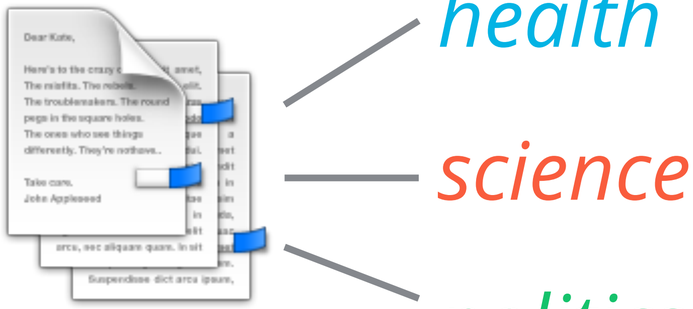


Unstructured Text



Applications





health

science

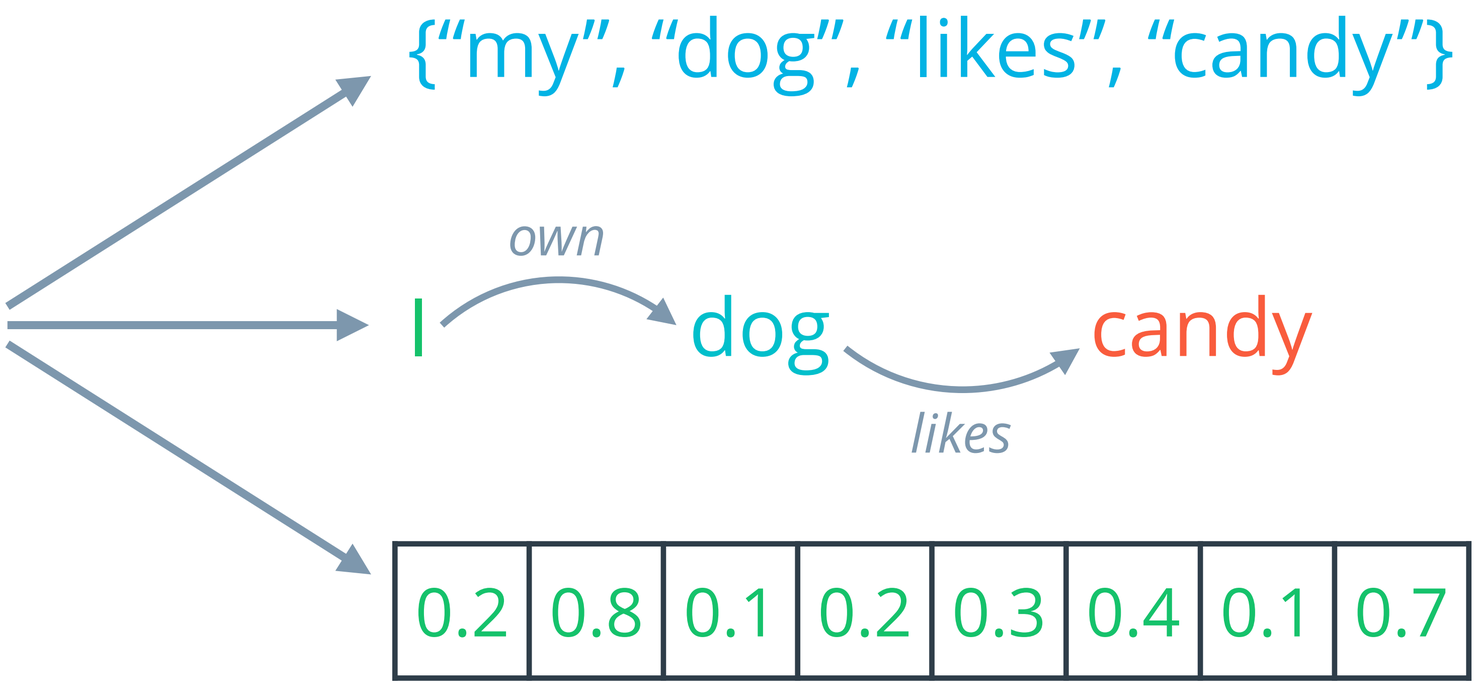
politics

what time is it?

¿que hora es?

Challenges: Representation

My dog likes candy.



Challenges: Temporal Sequence

I want to buy a gallon of milk

water

petrol

Challenges: Context

The old Welshman came home toward daylight, spattered with candle-grease, smeared with clay, and almost worn out. He found Huck still in the bed that had been provided for him, and delirious with fever. The physicians were all at the cave, so the Widow Douglas came and took charge of the patient.

—*The Adventures of Tom Sawyer*, Mark Twain

“Mary went back home. …”

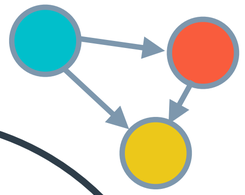
Process

{<“mary”, “go”, “home”>, … }

Transform

{<0.4, 0.8, 0.3, 0.1, 0.7>, …}

Analyze



Predict

Present

Classic NLP: Text Processing

Text Processing

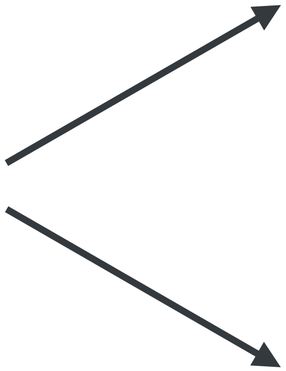
- Tokenization
- Stop Word Removal
- Stemming and Lemmatization

Tokenization

“Jack and Jill went up the hill” → <“jack”, “and”, “jill”,
“went”, “up”, “the”, “hill”>

Tokenization

“No, she didn’t do it.”



<“no,”, “she”, “didn”,
“’”, “t”, “do” “it”, “.”>

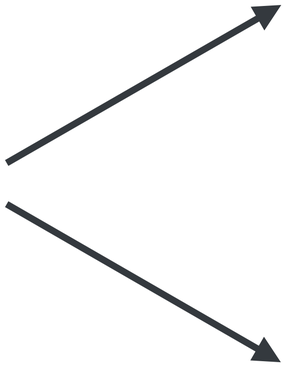
<“no”, “she”, “didnt”,
“do”, “it”>

?

Tokenization

Big money behind big special effects tends to suggest a big story. Nope, not here. Instead this huge edifice is like one of those over huge luxury condos that're empty in every American town, pretending as if there's a local economy huge enough to support such.

—Rotten Tomatoes



<"big", "money", "behind", "big", "special", "effects", "tends", "to", "suggest", "big", "story", "nope", "not", "here", "instead", "this", "huge", "edifice", "is", "like", "one", "of", "those", "over", "huge", "luxury", "condos", "that", "re", "empty", "in", "every", "american", "town", "pretending", "as", "if", "there", "local", "economy", "huge", "enough", "to", "support", "such">

<"big", "money", "behind", "big", "special", "effects", "tends", "to", "suggest", "big", "story">

<"nope", "not", "here">

<"instead", "this", "huge", "edifice", "is", "like", "one", "of", "those", "over", "huge", "luxury", "condos", "that", "re", "empty", "in", "every", "american", "town", "pretending", "as", "if", "there", "local", "economy", "huge", "enough", "to", "support", "such">

?

Stop Word Removal

wristwatch invented 1904 Louis Cartier.

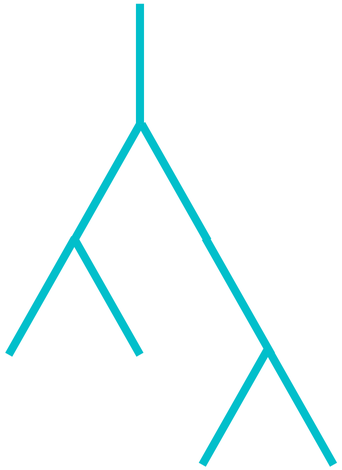
Stemming

branch~~ing~~

branch~~ed~~

branch~~es~~

branch

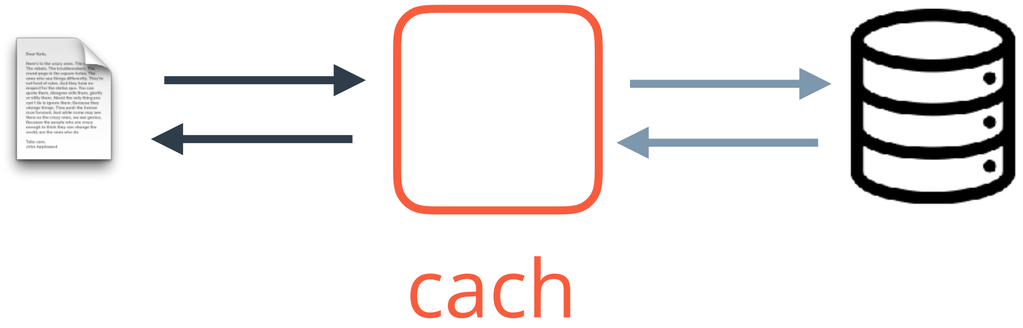


Stemming

caching

cached

caches

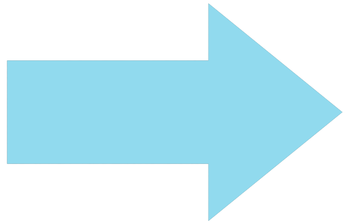


Lemmatization

is

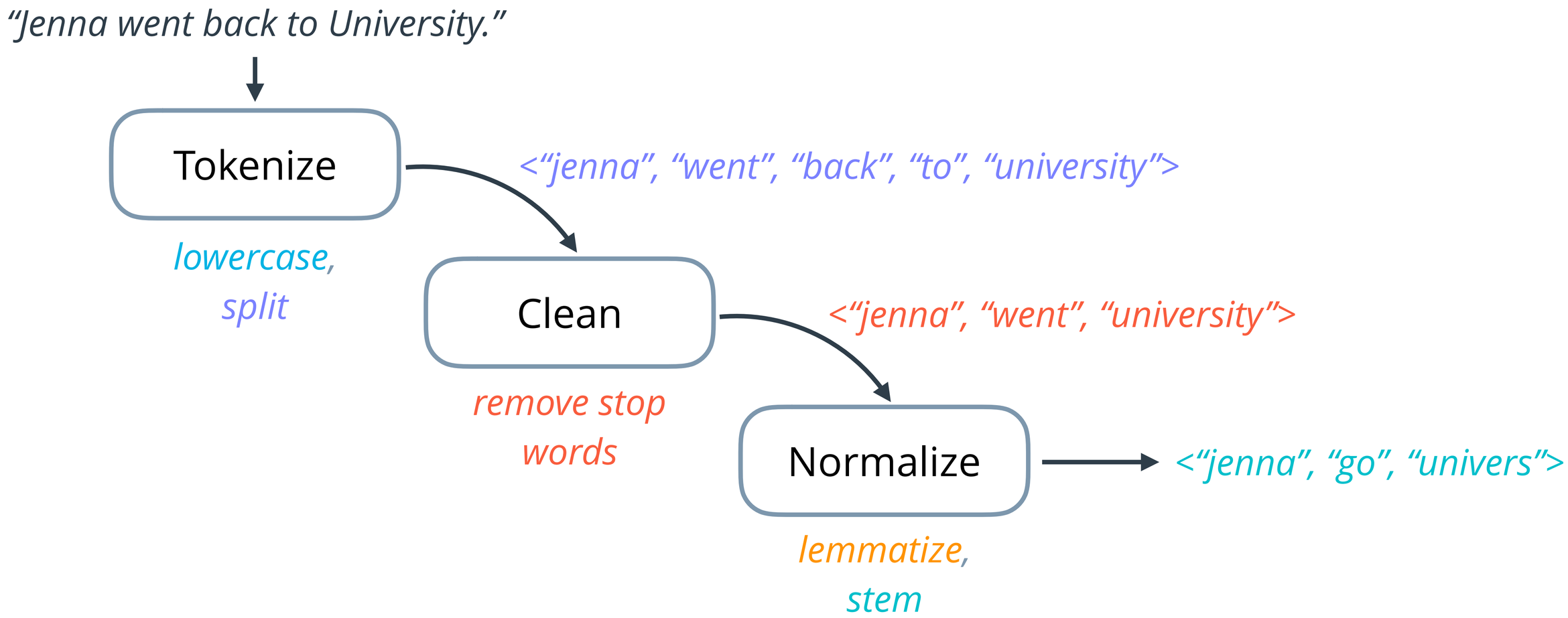
was

were



be

Text Processing Summary

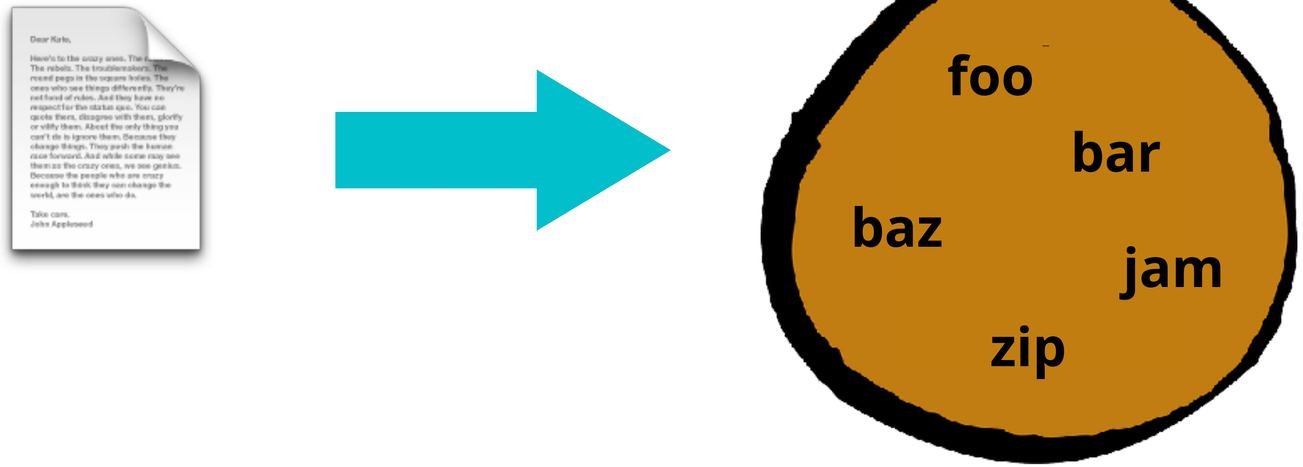


Classic NLP: Feature Extraction

Feature Extraction

- Bag of Words Representation
- Document-Term Matrix
- Term Frequency-Inverse Document Frequency (TF-IDF)

Bag of Words



Bag of Words

“Little House on the Prairie”	→	{“littl”, “hous”, “prairi”}
“Mary had a Little Lamb”	→	{“mari”, “littl”, “lamb”}
“The Silence of the Lambs”	→	{“silenc”, “lamb”}
“Twinkle Twinkle Little Star”	→	{“twinkl”, “littl”, “star”} ?

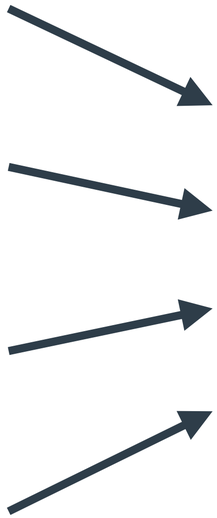
Bag of Words

“Little House on the Prairie”

“Mary had a Little Lamb”

“The Silence of the Lambs”

“Twinkle Twinkle Little Star”



littl hous prairi mari
lamb silenc twinkl star

corpus (D)

vocabulary (V)

Bag of Words

	littl	hous	prairi	mari	lamb	silenc	twinkl	star
“Little House on the Prairie”								
“Mary had a Little Lamb”								
“The Silence of the Lambs”								
“Twinkle Twinkle Little Star”								

Bag of Words - Term Matrix

term frequency

	littl	hous	prairi	mari	lamb	silenc	twinkl	star
“Little House on the Prairie”	1	1	1	0	0	0	0	0
“Mary had a Little Lamb”	1	0	0	1	1	0	0	0
“The Silence of the Lambs”	0	0	0	0	1	1	0	0
“Twinkle Twinkle Little Star”	1	0	0	0	0	0	2	1

Document Similarity

	littl	hous	prairi	mari	lamb	silenc	twinkl	star
a “Little House on the Prairie”	1	1	1	0	0	0	0	0
b “Mary had a Little Lamb”	1	0	0	1	1	0	0	0

$$\mathbf{a}\cdot\mathbf{b} = \sum a_0b_0 + a_1b_1 + \ldots + a_nb_n = 1 + 0 + 0 + 0 + 0 + 0 + 0 + 0$$

dot product

Document Similarity

	littl	hous	prairi	mari	lamb	silenc	twinkl	star
a “Little House on the Prairie”	1	1	1	0	0	0	0	0
b “Mary had a Little Lamb”	1	0	0	1	1	0	0	0

$$\mathbf{a}\cdot\mathbf{b} = \sum a_0b_0 + a_1b_1 + \ldots + a_nb_n = 1$$ *dot product*

$$\cos(\theta) = \frac{\mathbf{a}\cdot\mathbf{b}}{\|\mathbf{a}\|\cdot\|\mathbf{b}\|} = \frac{1}{\sqrt{3}\times\sqrt{3}} = \frac{1}{3}$$ *cosine similarity*

Term Specificity

	littl	hous	prairi	mari	lamb	silenc	twinkl	star
“Little House on the Prairie”	1/3	1/1	1/1	0/1	0/2	0/1	0/1	0/1
“Mary had a Little Lamb”	1/3	0/1	0/1	1/1	1/2	0/1	0/1	0/1
“The Silence of the Lambs”	0/3	0/1	0/1	0/1	1/2	1/1	0/1	0/1
“Twinkle Twinkle Little Star”	1/3	0/1	0/1	0/1	0/2	0/1	2/1	1/1
document frequency ——	3	1	1	1	2	1	1	1

Term Specificity

	littl	hous	prairi	mari	lamb	silenc	twinkl	star
“Little House on the Prairie”	1/3	1	1	0	0	0	0	0
“Mary had a Little Lamb”	1/3	0	0	1	1/2	0	0	0
“The Silence of the Lambs”	0	0	0	0	1/2	1	0	0
“Twinkle Twinkle Little Star”	1/3	0	0	0	0	0	2	1

TF-IDF

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D)$$

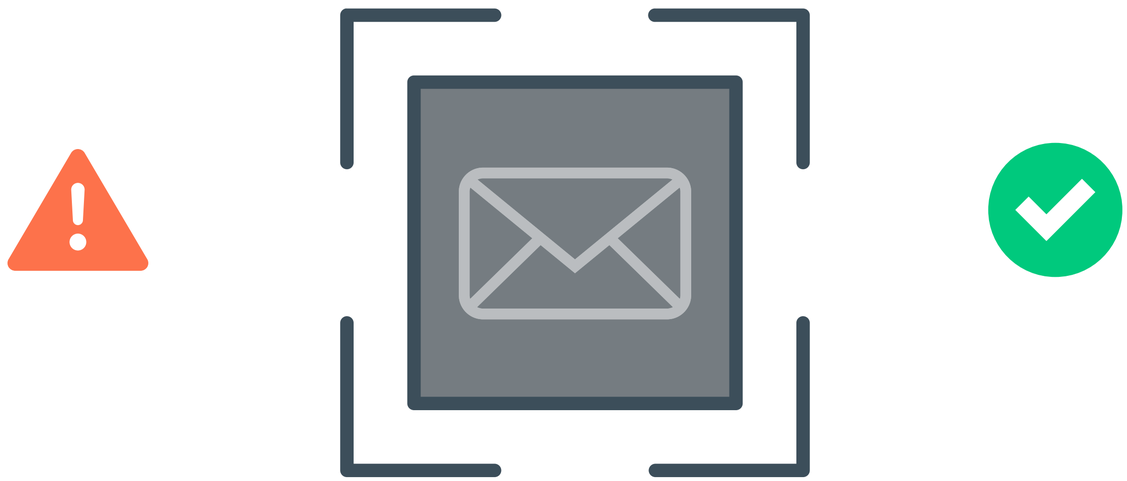
term frequency

$$\text{count}(t, d) / |d|$$

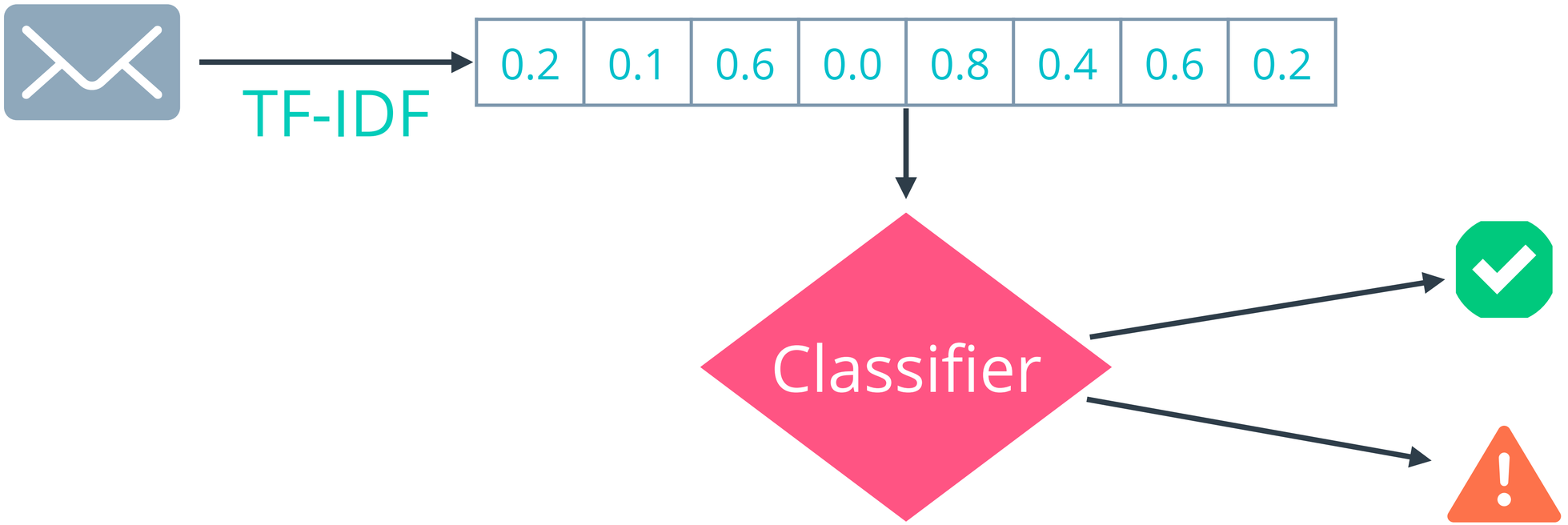
inverse document frequency

$$\log(|D| / |\{d \in D : t \in d\}|)$$

Example Task: Spam Detection



Example Task: Spam Detection

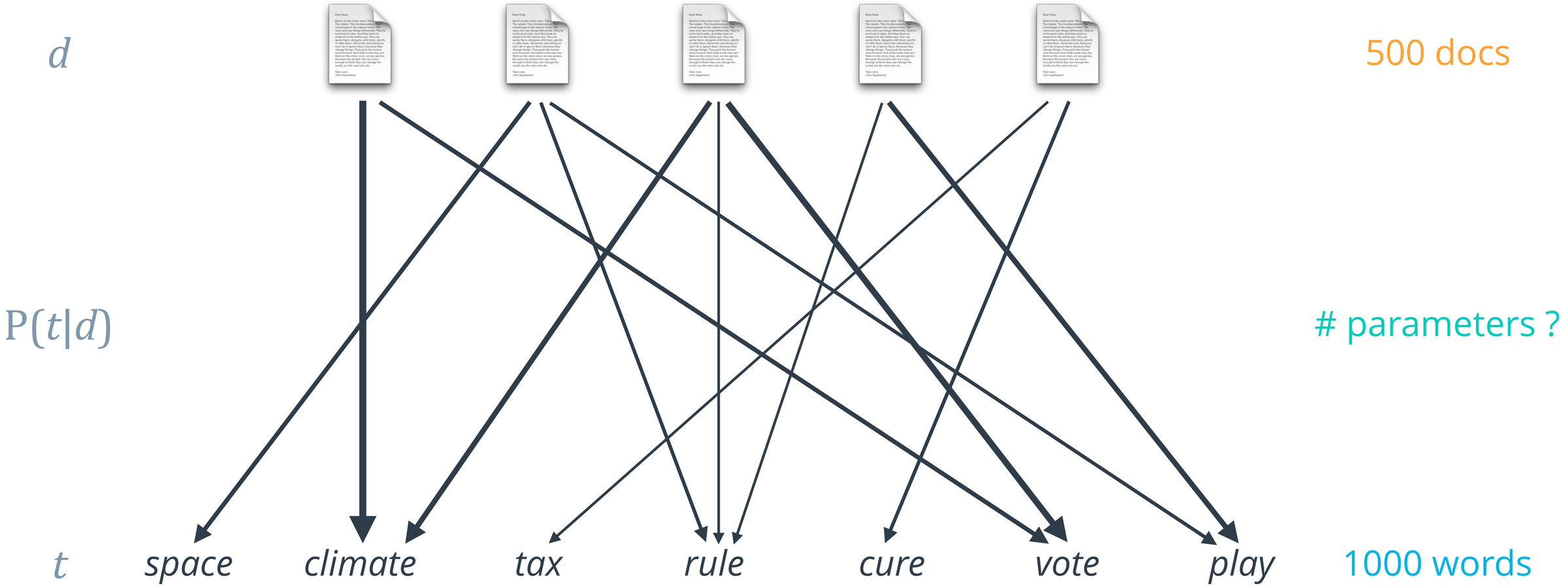


Classic NLP: Topic Modeling

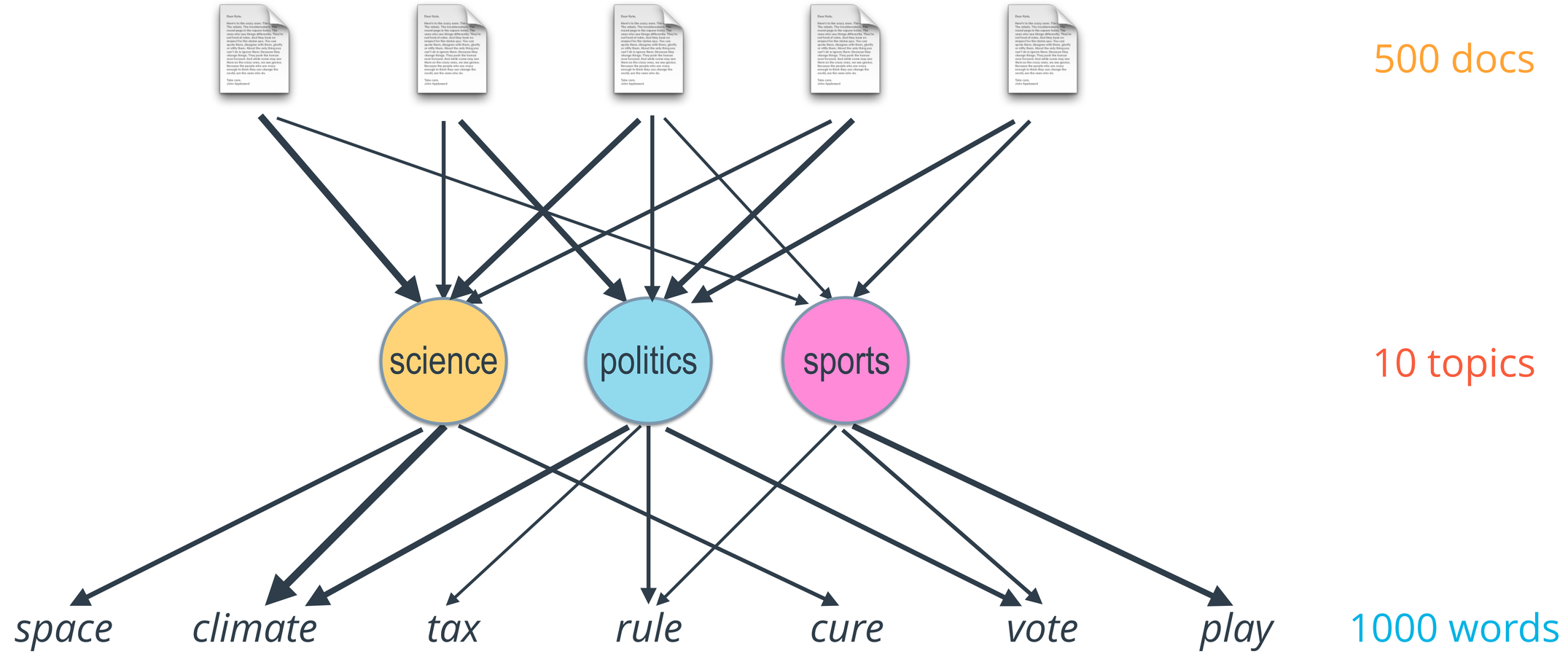
Topic Modeling

- Latent Variables
- Latent Dirichlet Allocation
- Lab: Topic Modeling using LDA

Bag of Words: Graphical Model



Latent Variables: Topics



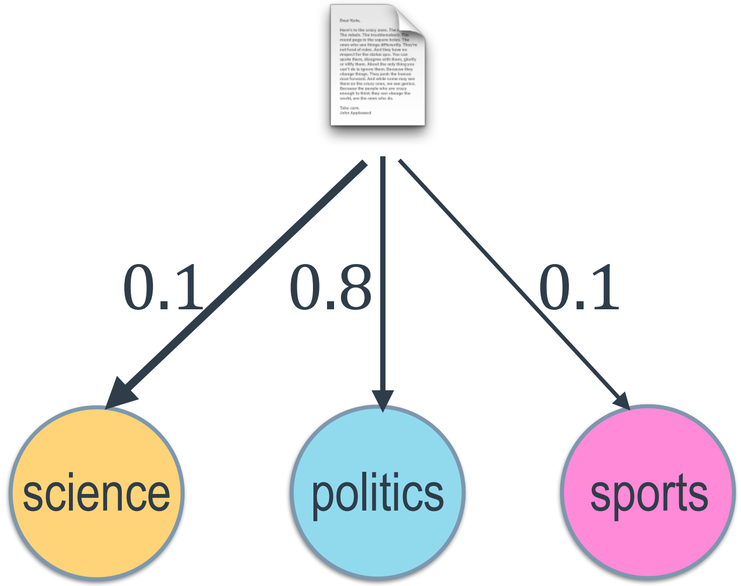
$$P(t|d) = \sum_z P(t|z)\, P(z|d)$$

Mixture Model: Document → Topic

d

$\mathrm{P}(z|d)$

z



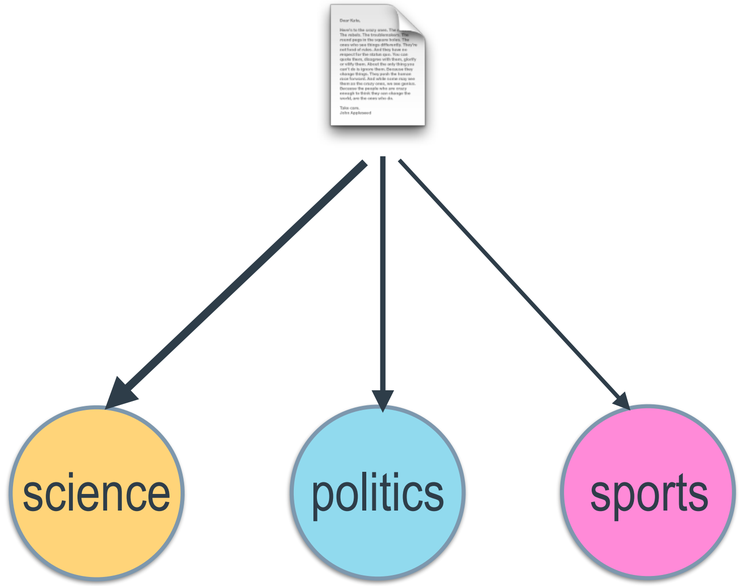
$\sum_z \mathrm{P}(z|d) = 1$

Mixture Model: Document → Topic

d

$\mathrm{P}(z|d)$

z



$\boldsymbol{\theta}$ = 0.1 | 0.8 | 0.1

$\sum_z \mathrm{P}(z|d) = 1$

multinomial distribution

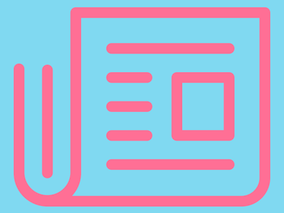
Binomial

H	T
0.5	0.5
0.7	0.3
0.4	0.6

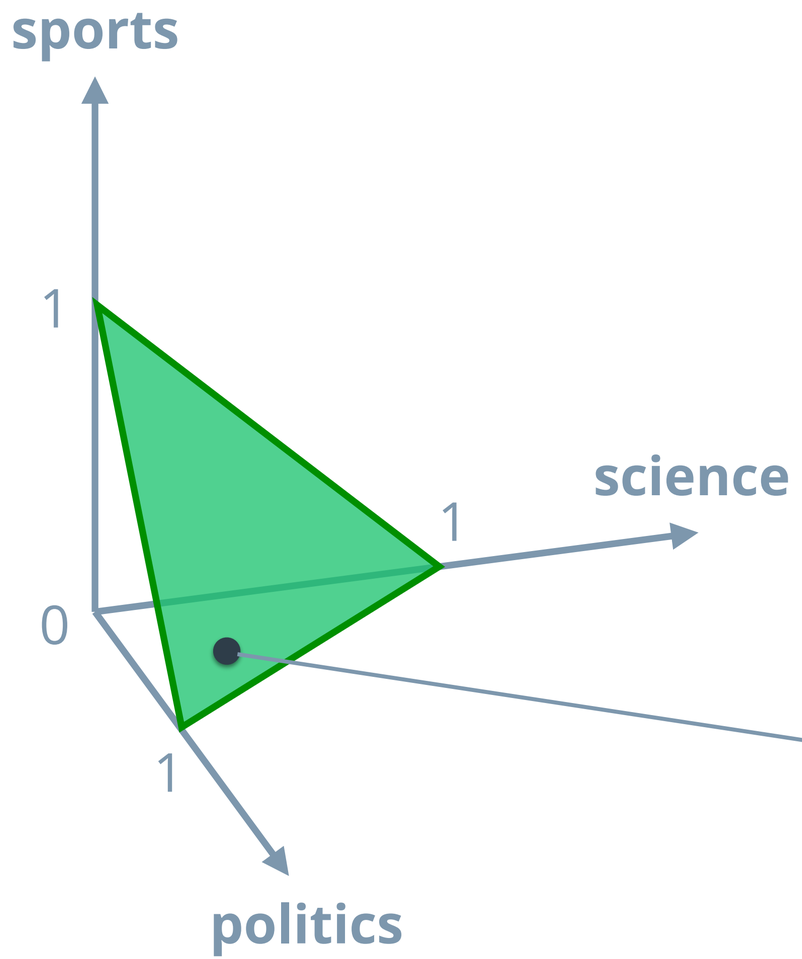
Binomial

H	T
0.5	0.5
0.7	0.3
0.4	0.6

Multinomial

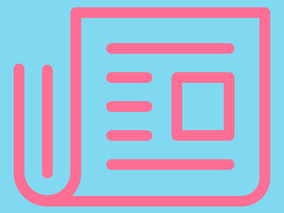


science	politics	sports
0.9	0.05	0.05
0.1	0.8	0.1
0.2	0.1	0.7

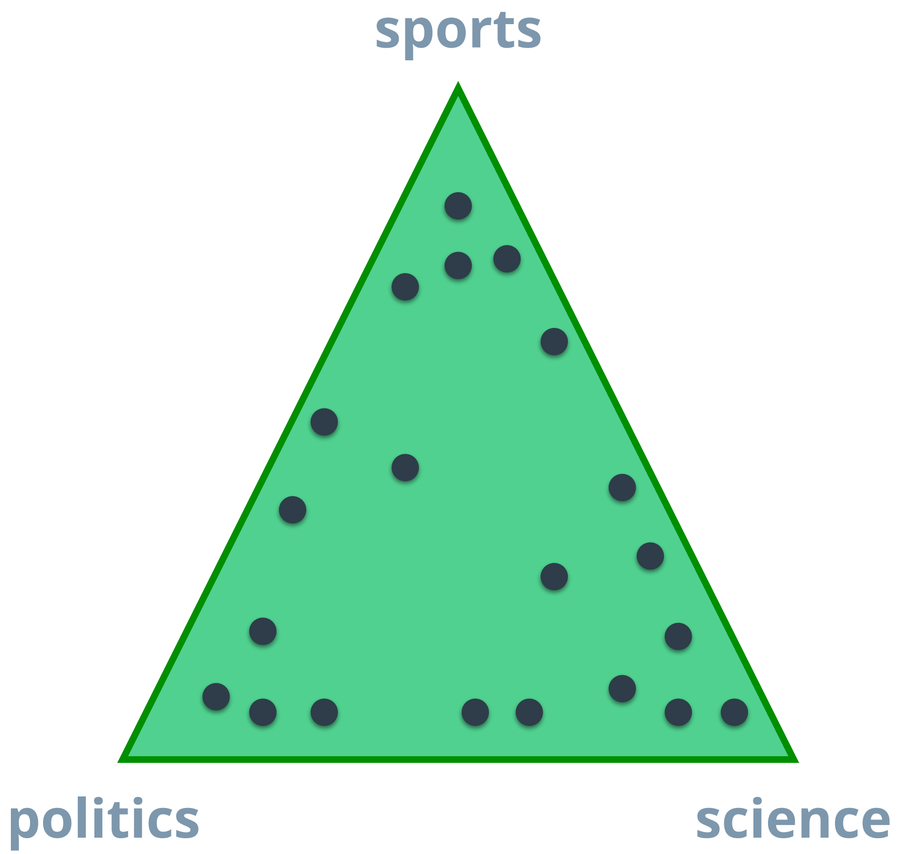


Probability Simplex

Multinomial

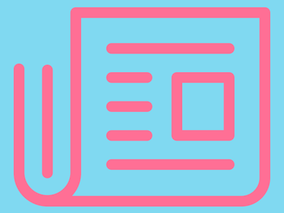


science	politics	sports
0.9	0.05	0.05
0.1	0.8	0.1
0.2	0.1	0.7



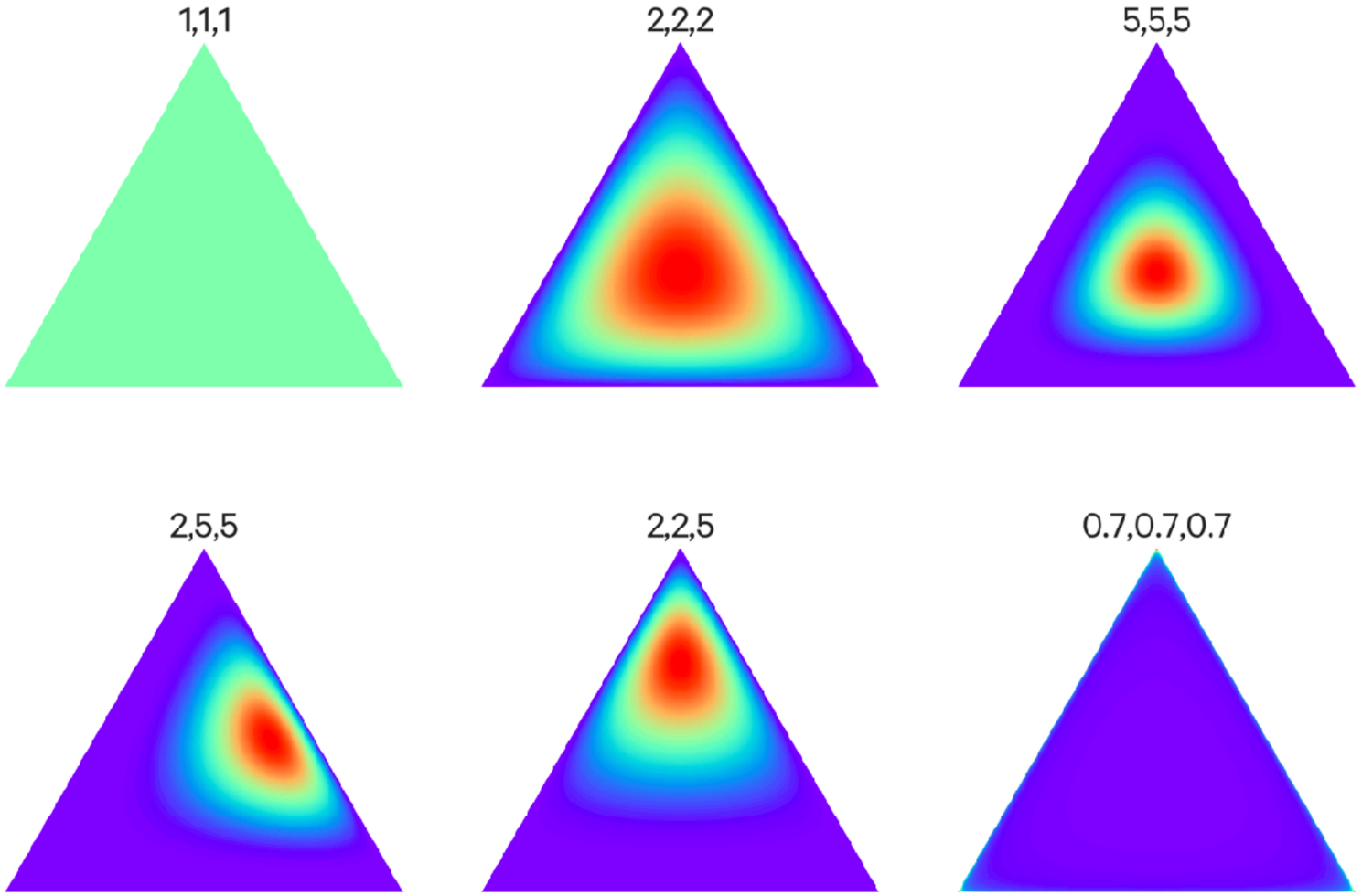
Probability Simplex

Multinomial



science	politics	sports
0.9	0.05	0.05
0.1	0.8	0.1
0.2	0.1	0.7

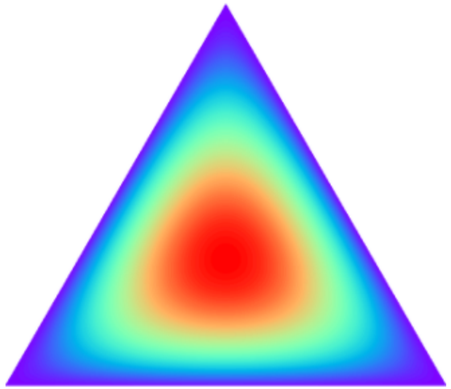
Dirichlet Distributions



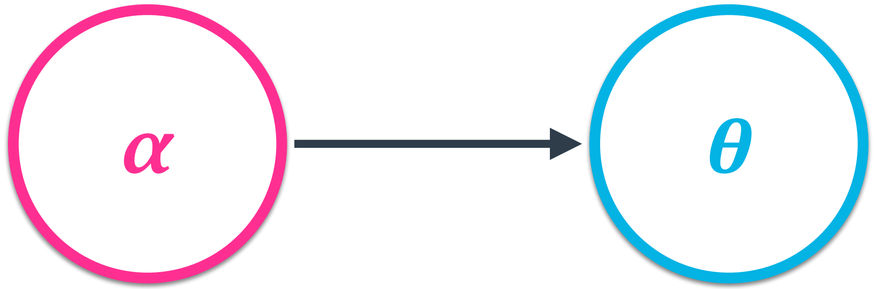
α = <a, b, c>

concentration parameters

LDA: Sample a Document



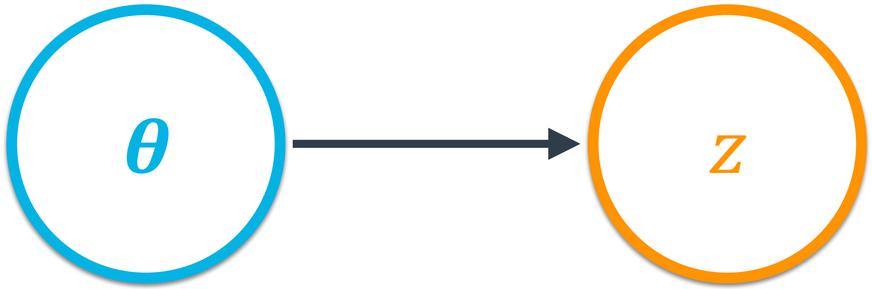
science	politics	sports
0.1	0.8	0.1



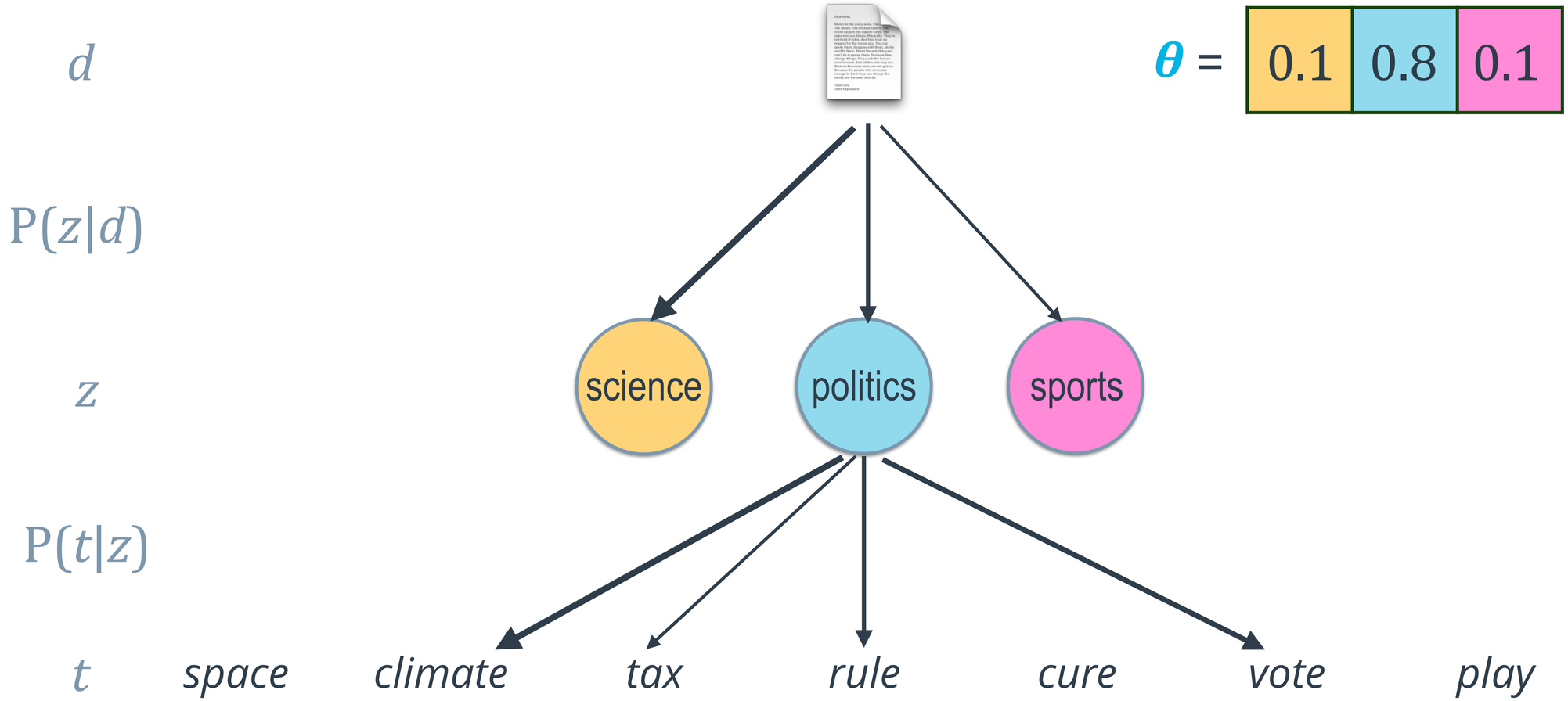
LDA: Sample a Topic

science	politics	sports
0.1	0.8	0.1

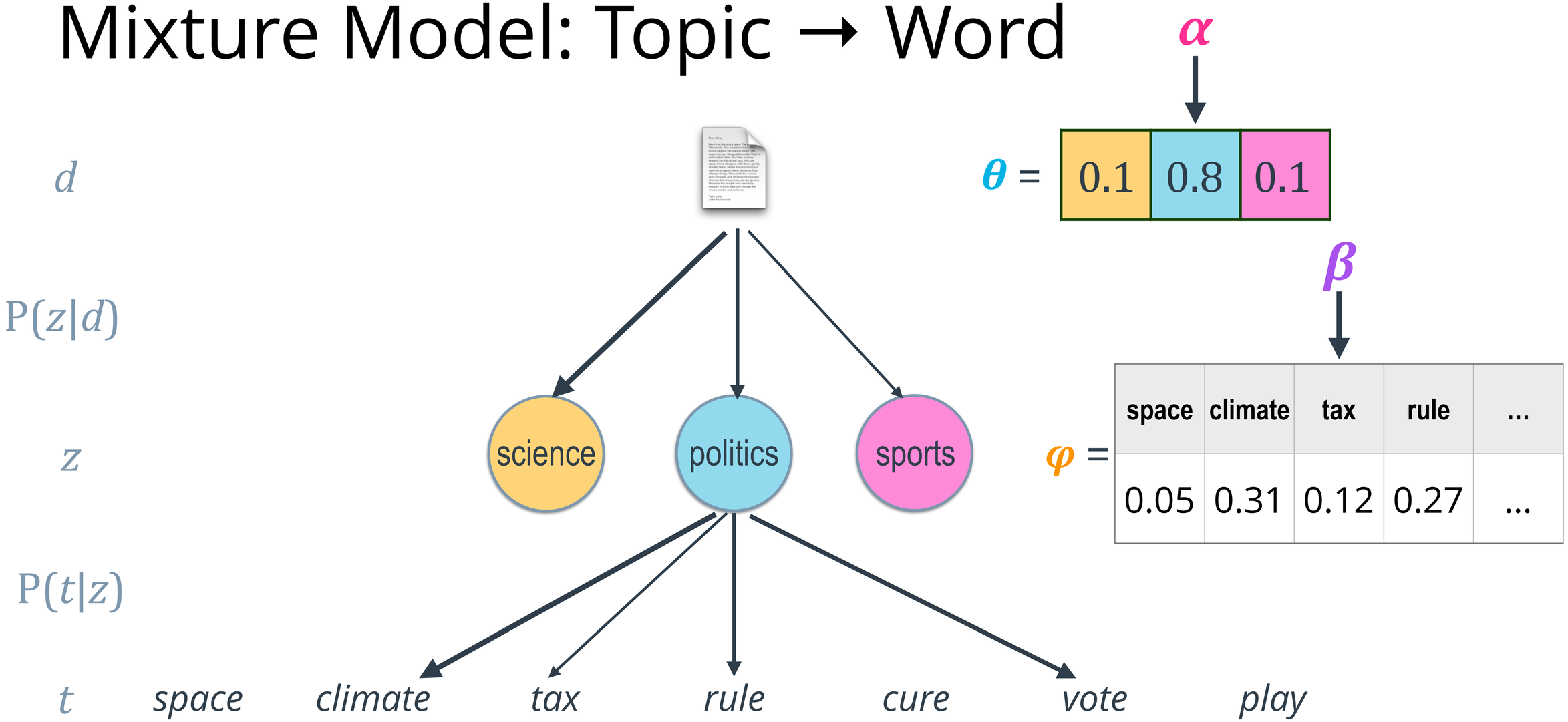
politics



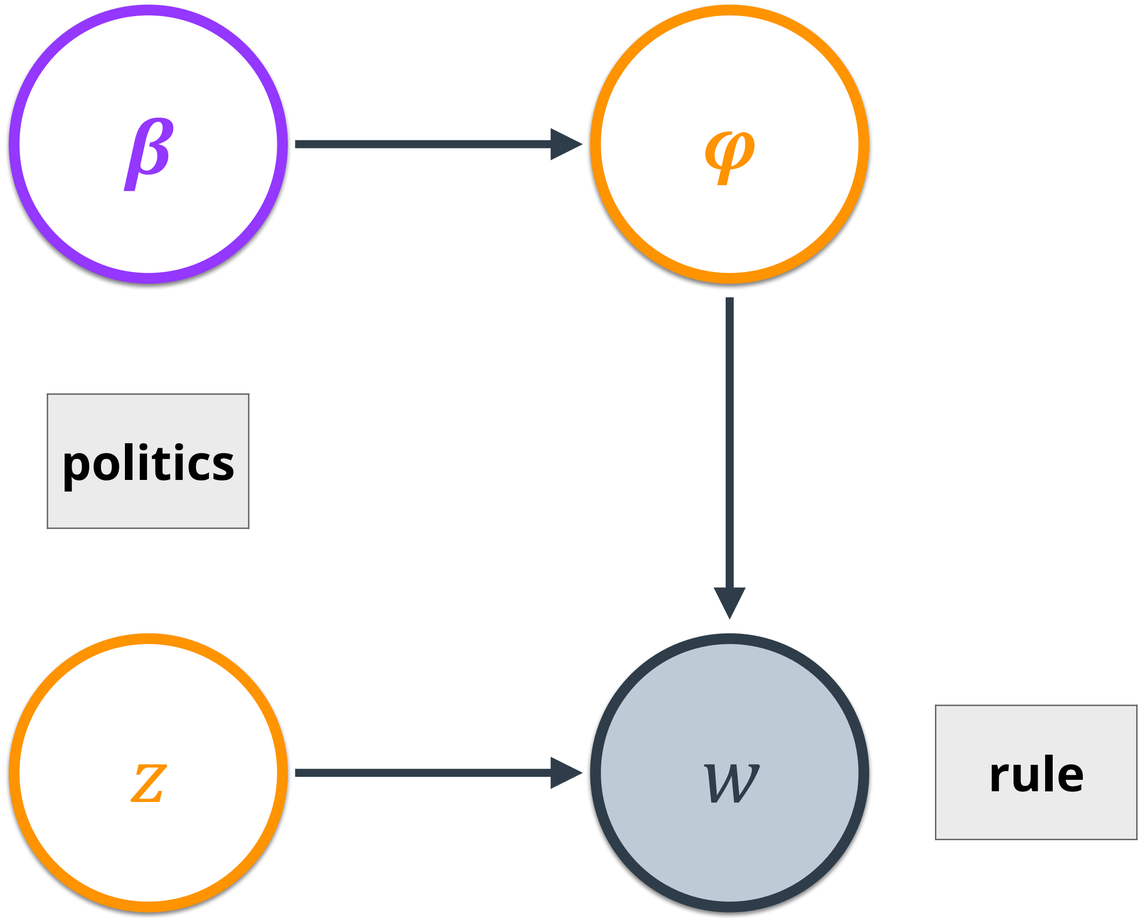
Mixture Model: Topic → Word



Mixture Model: Topic → Word

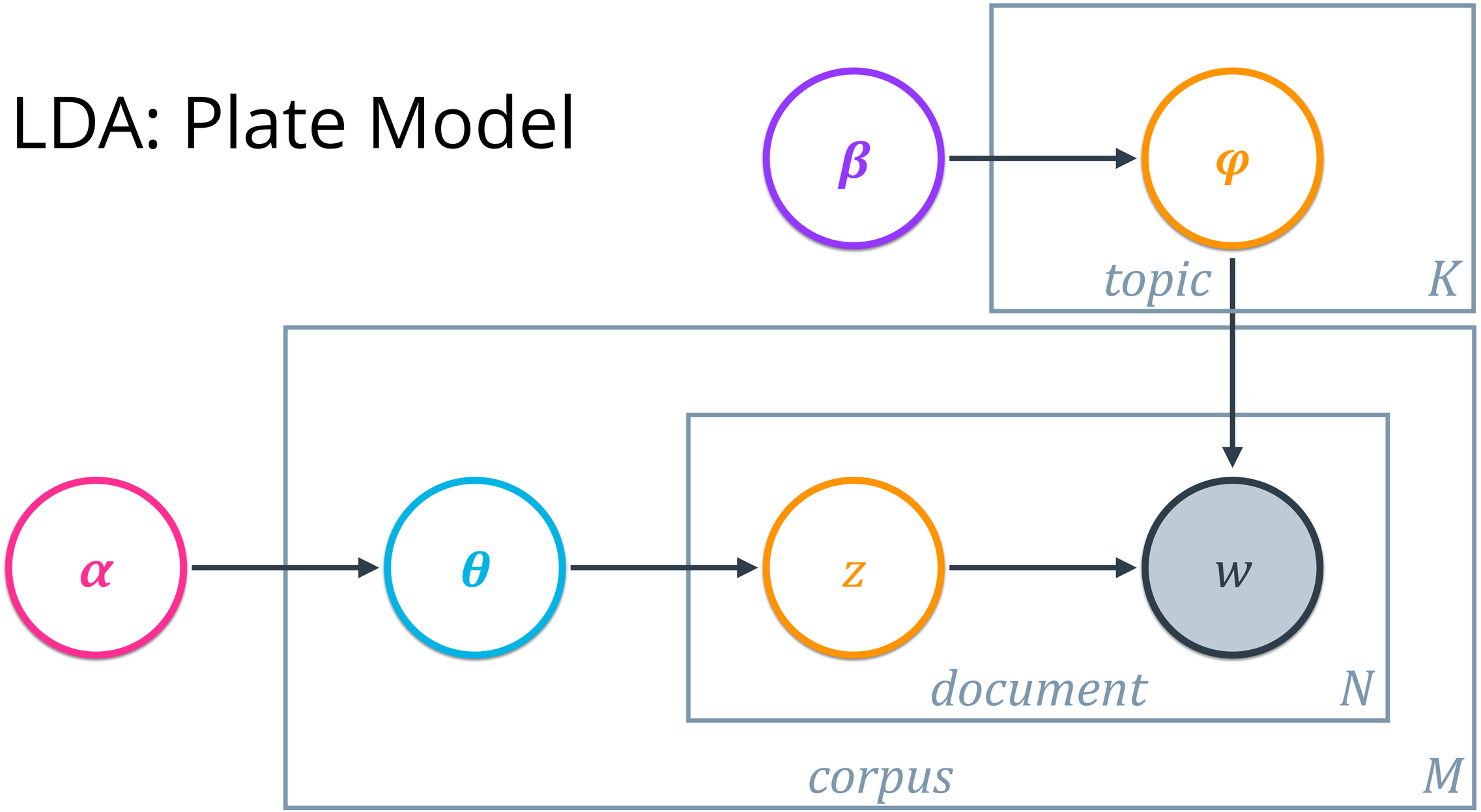


LDA: Sample a Word

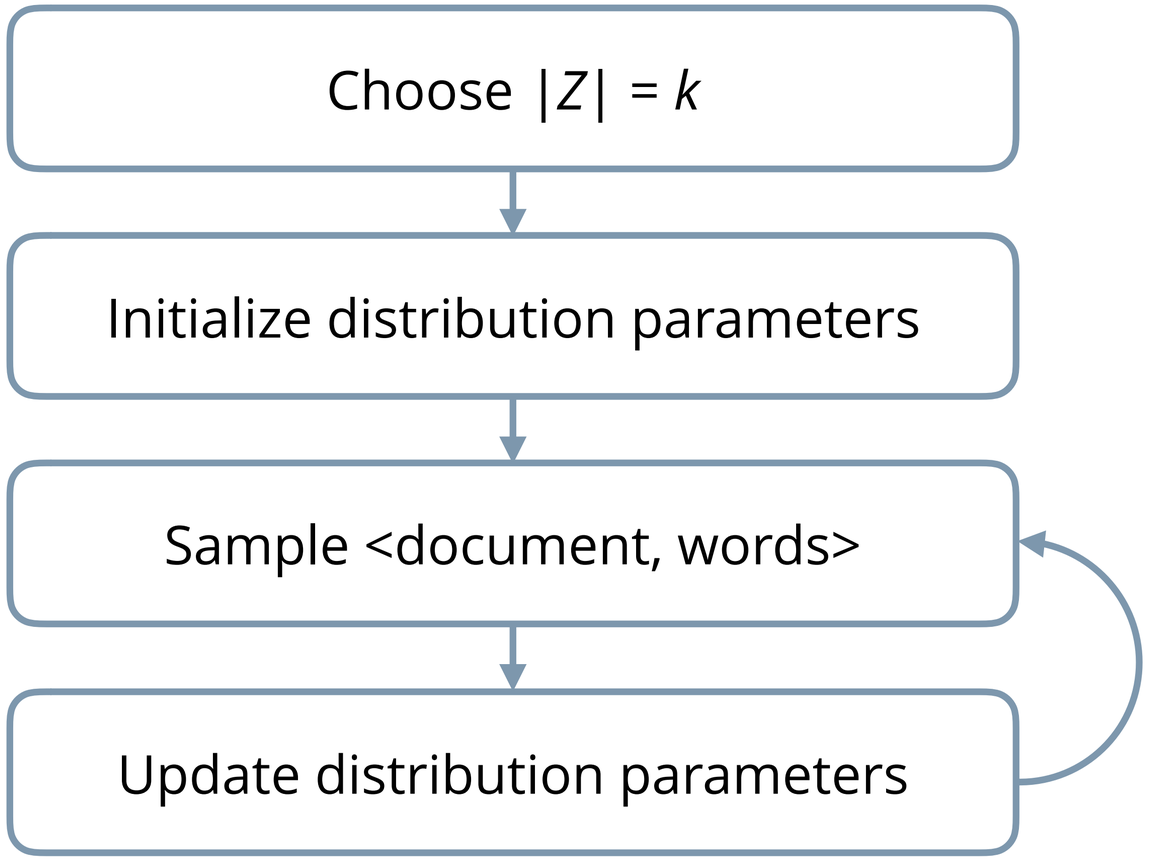


space	climate	tax	rule	...
0.05	0.31	0.12	0.27	...

LDA: Plate Model



LDA: Parameter Estimation



expectation

maximization

LDA: Use Cases

- Topic modeling, document categorization.
- Mixture of topics in a new document: $\mathrm{P}(\boldsymbol{z} \mid \boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$
- Generate collections of words with desired mixture.

LDA: Further Reading

David Blei, Andrew Ng, Michael Jordan, 2003. Latent Dirichlet Allocation, In *Journal of Machine Learning Research*, vol. 3, pp. 993-102.

Thomas Boggs, 2014. Visualizing Dirichlet Distributions with matplotlib.

Lab: Topic Modeling using LDA

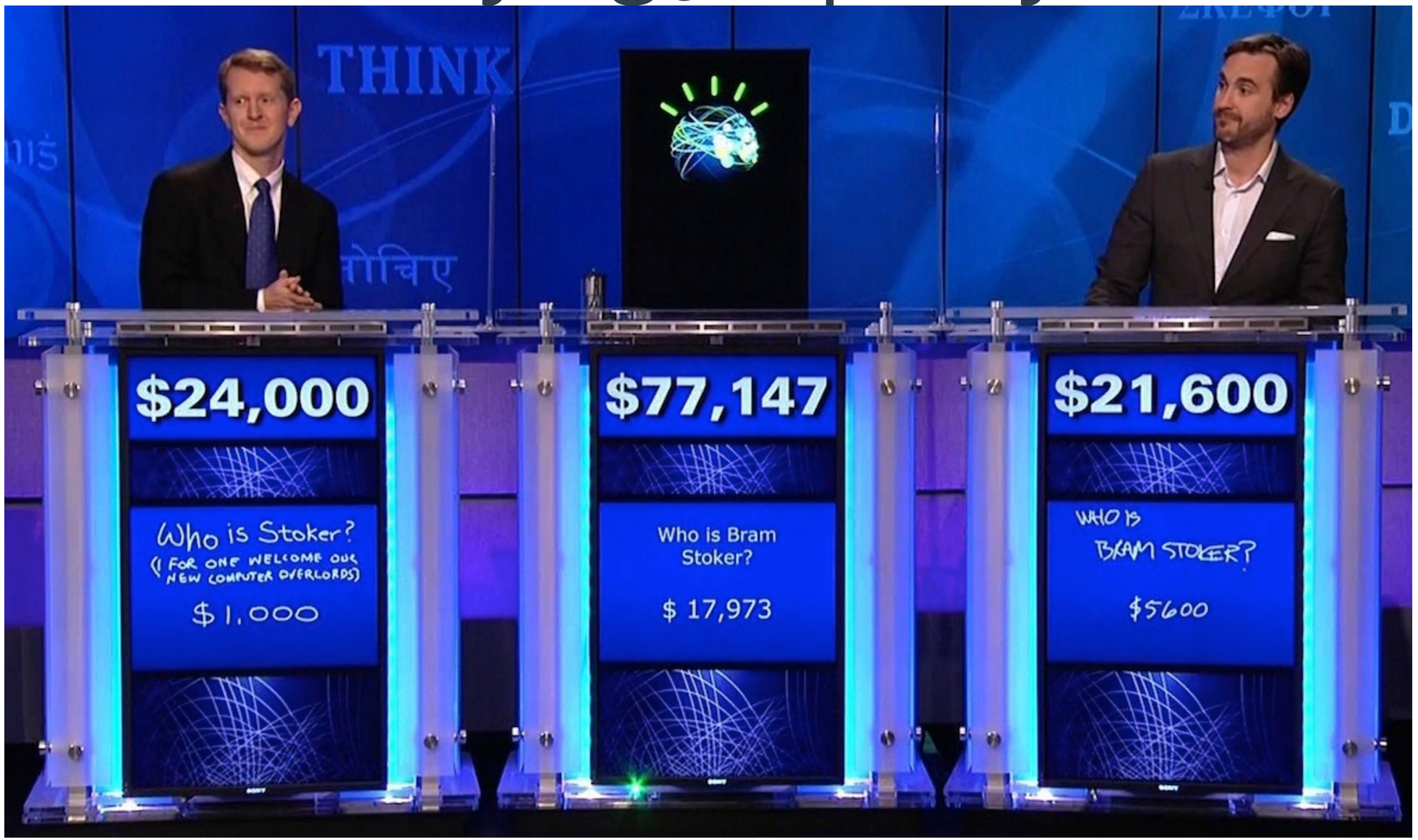
Categorize Newsgroups Data

Deep NLP: Neural Networks

Playing Go

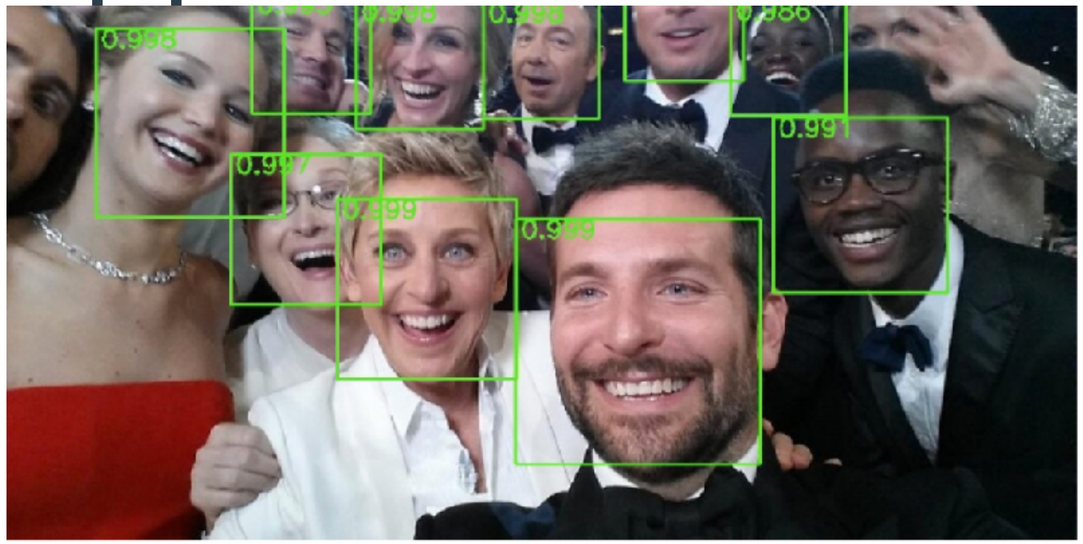

ALPHAGO
00:05:30
Google DeepMind
Challenge Match
LEE SEDOL
00:28:28

Playing Jeopardy

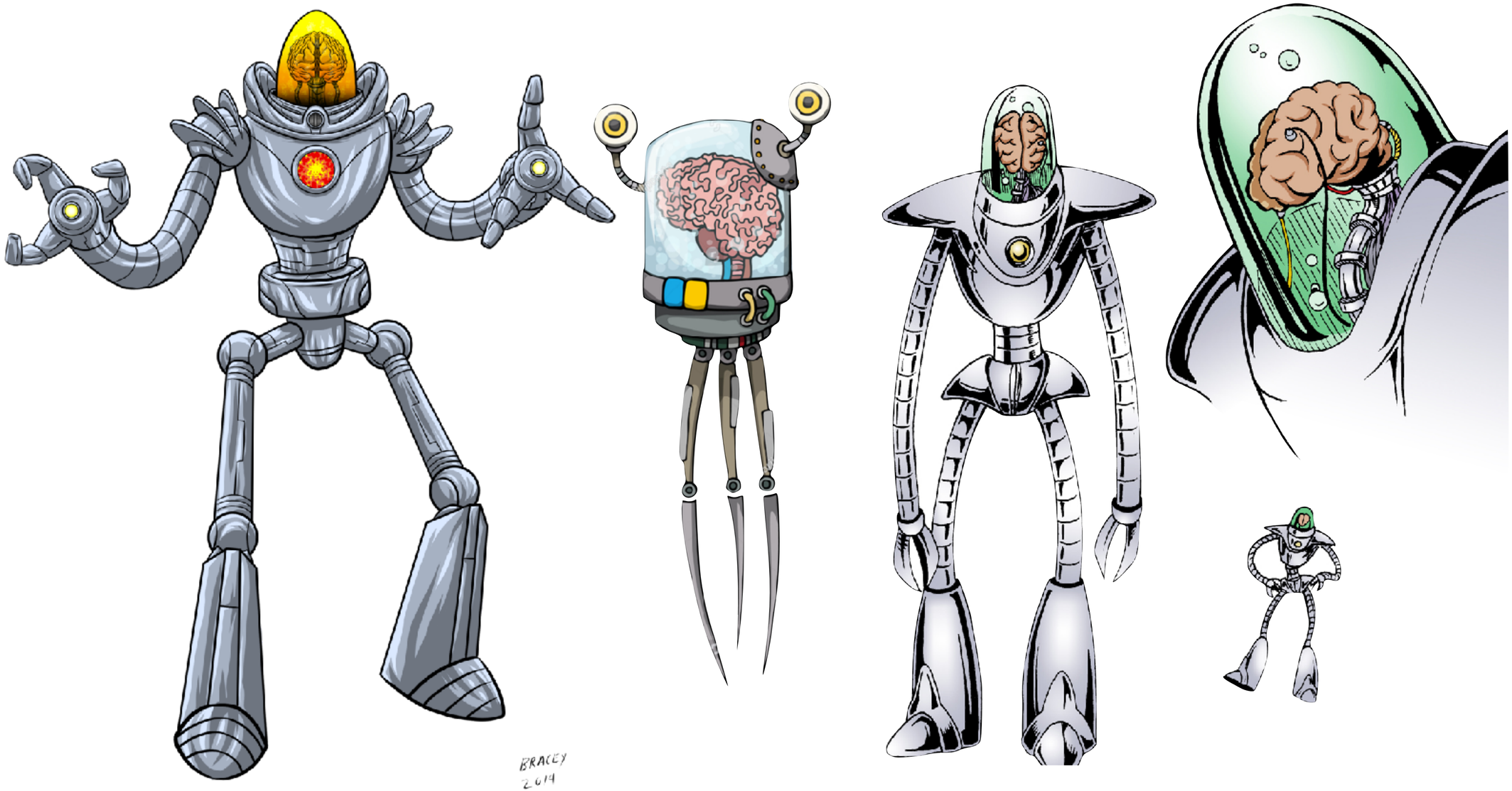


Self Driving Car

Many other applications

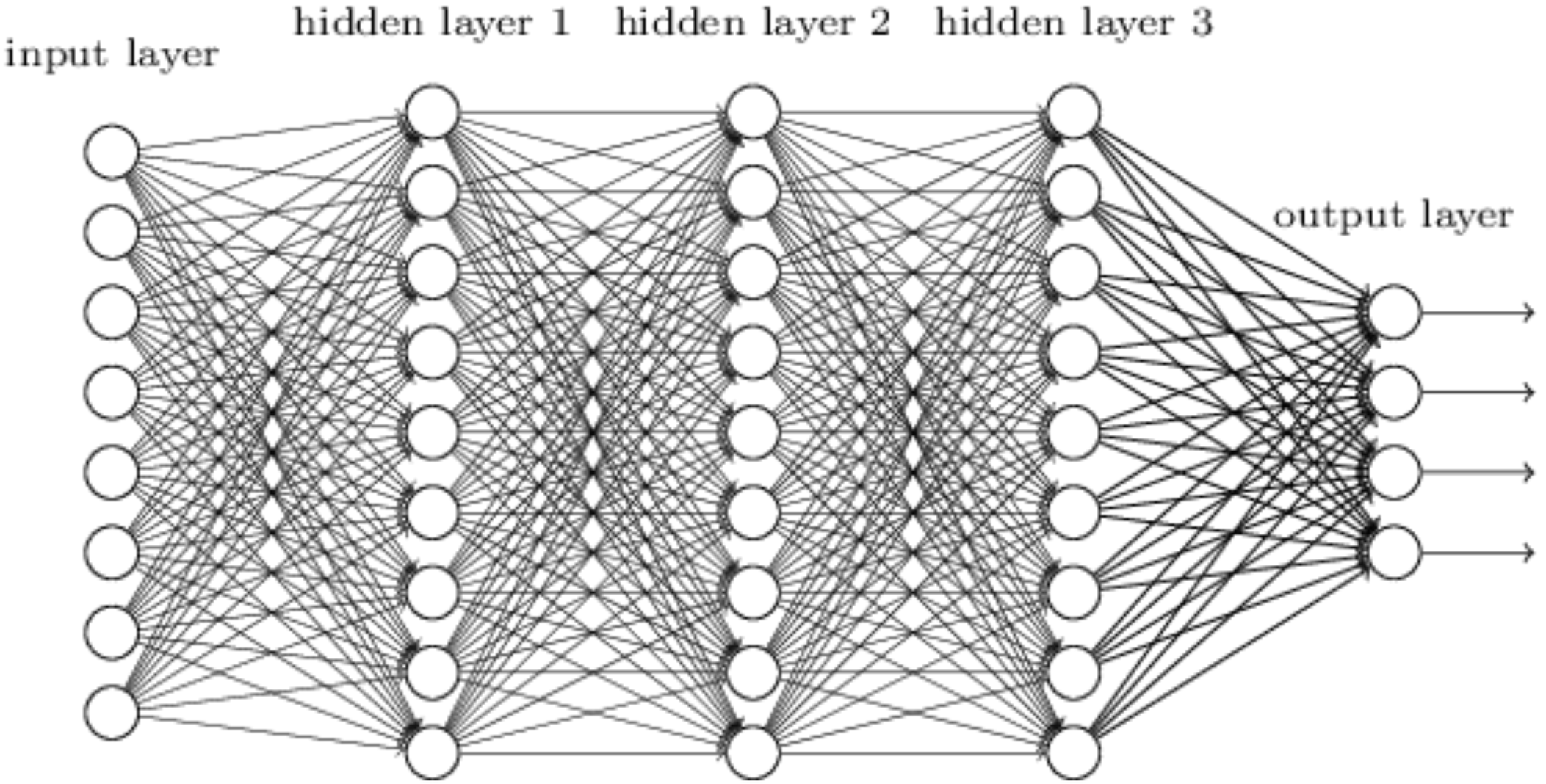


Neural Networks



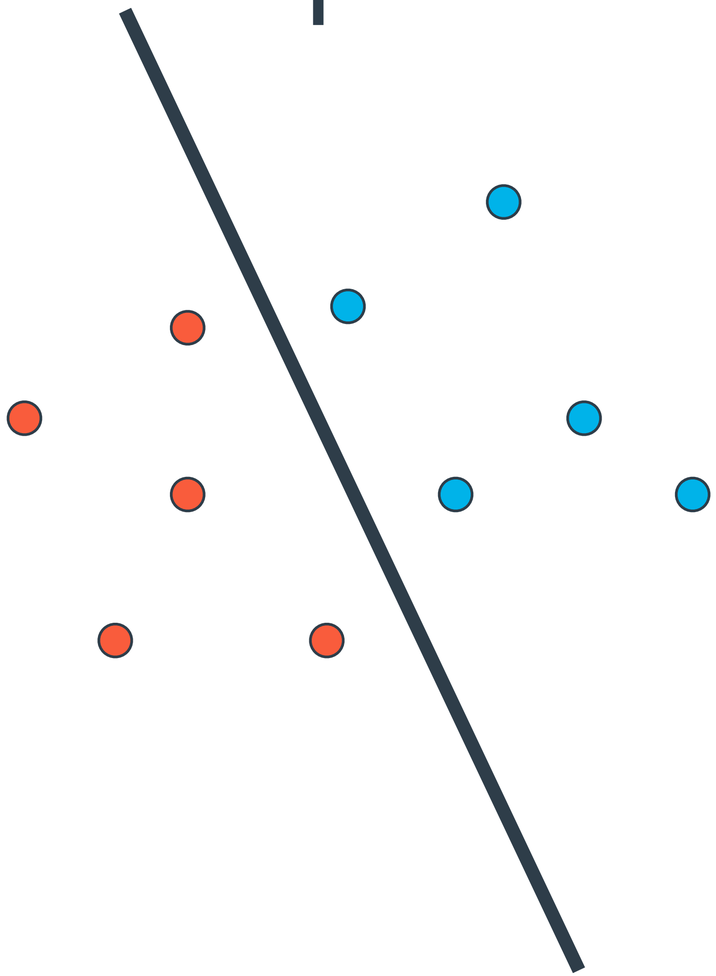
Neural Networks

Neural Networks

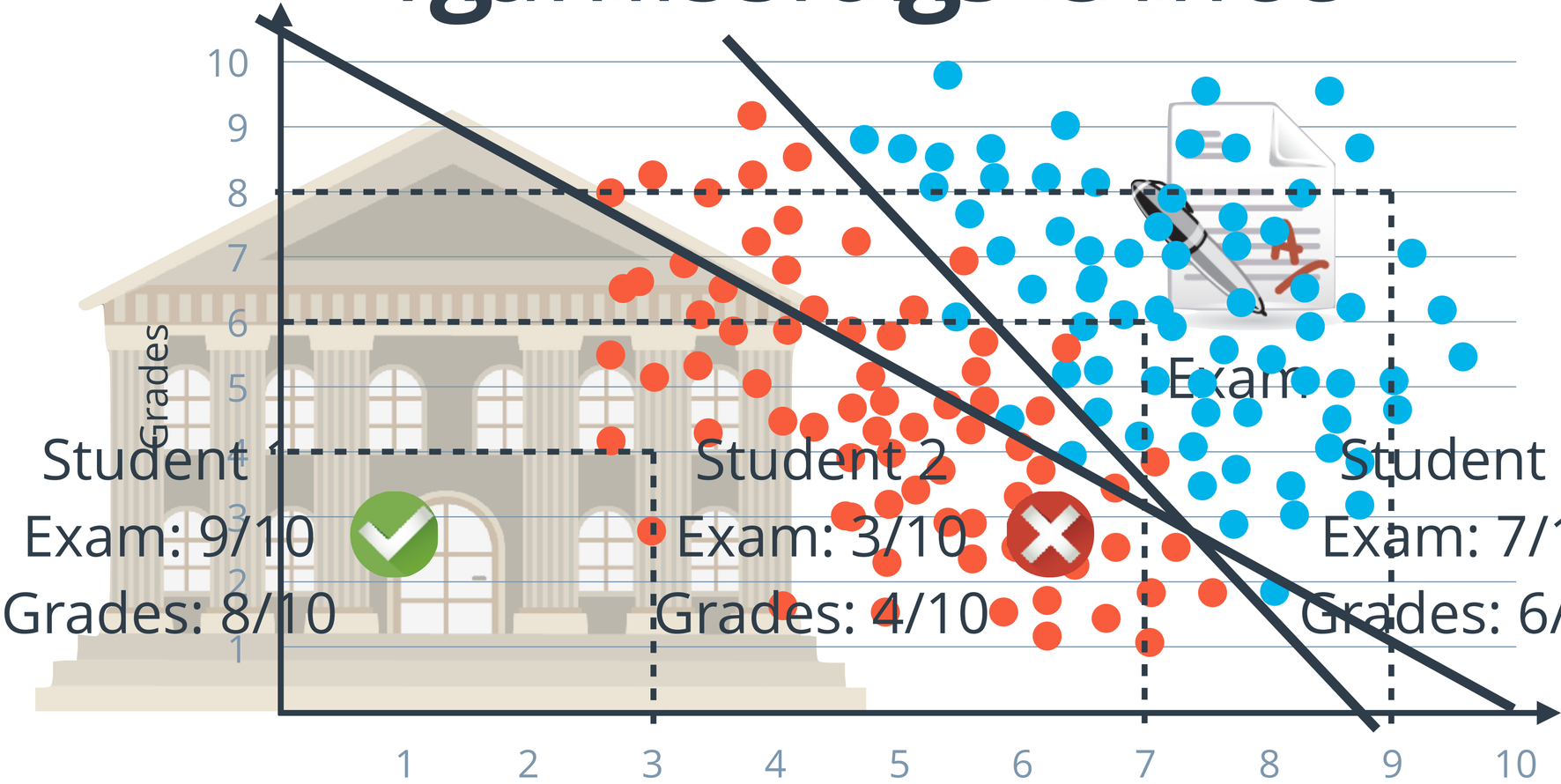


Neural Networks

Goal: Split Data



Logistic Regression



Student 1
Exam: 9/10
Grades: 8/10

Student 2
Exam: 3/10
Grades: 4/10

Student 3
Exam: 7/10
Grades: 6/10

Grades
Quiz

Does the student get accepted?

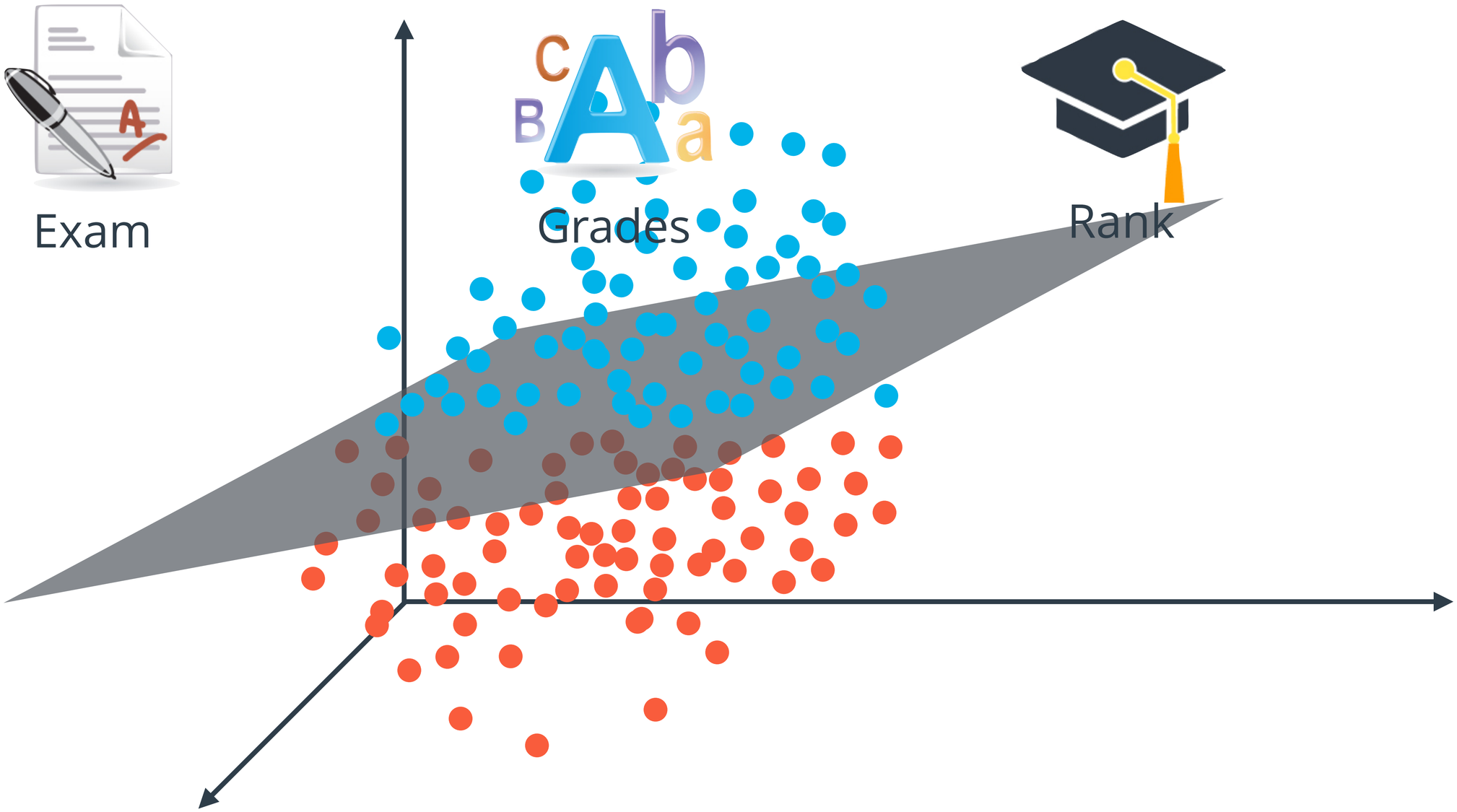
- Yes
- No

Score = Exam + Grades
Score > 10

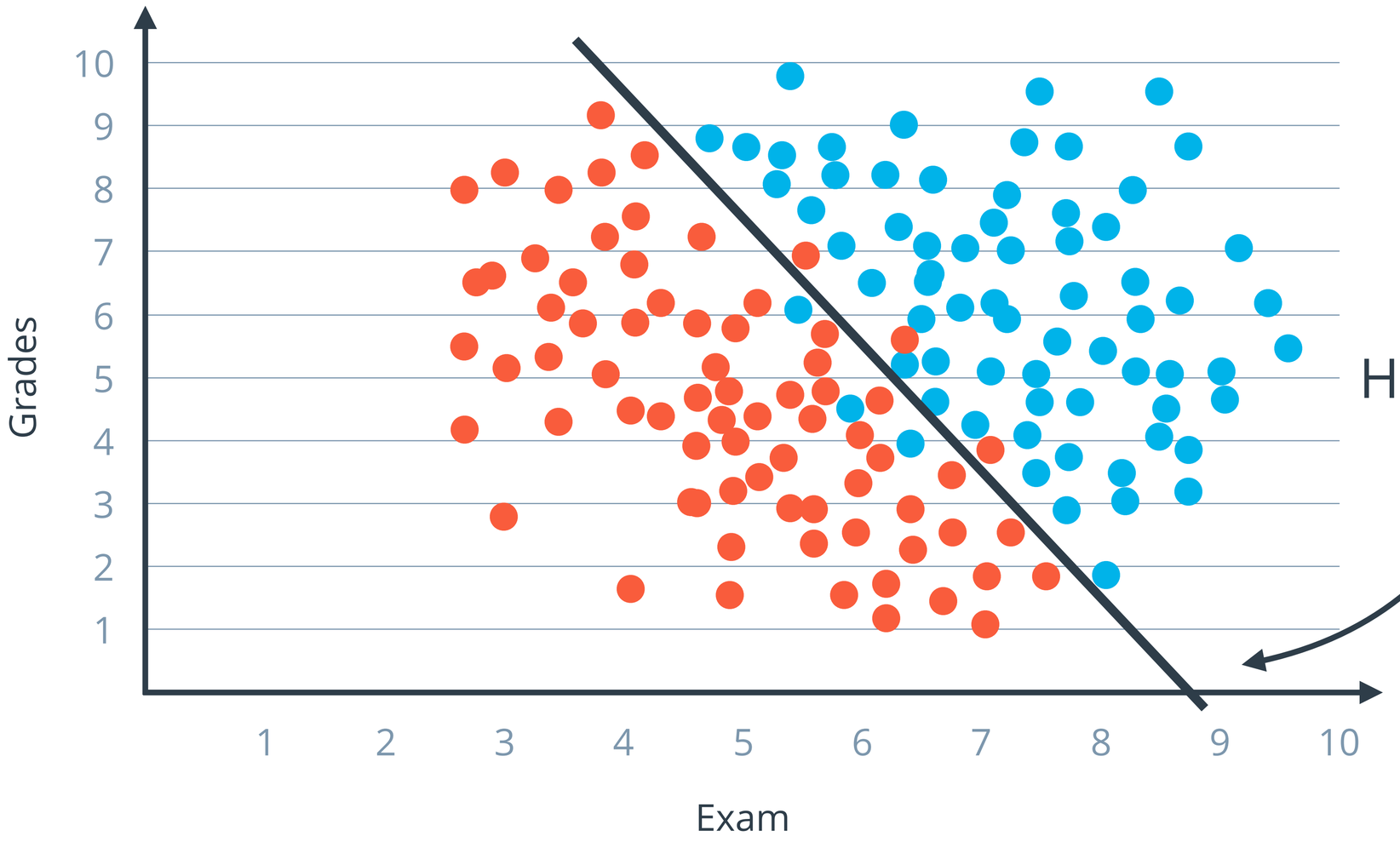
Score = Exam + 2*Grades
Score > 18

Score = Exam - Grades
Score > 5

Admissions Office

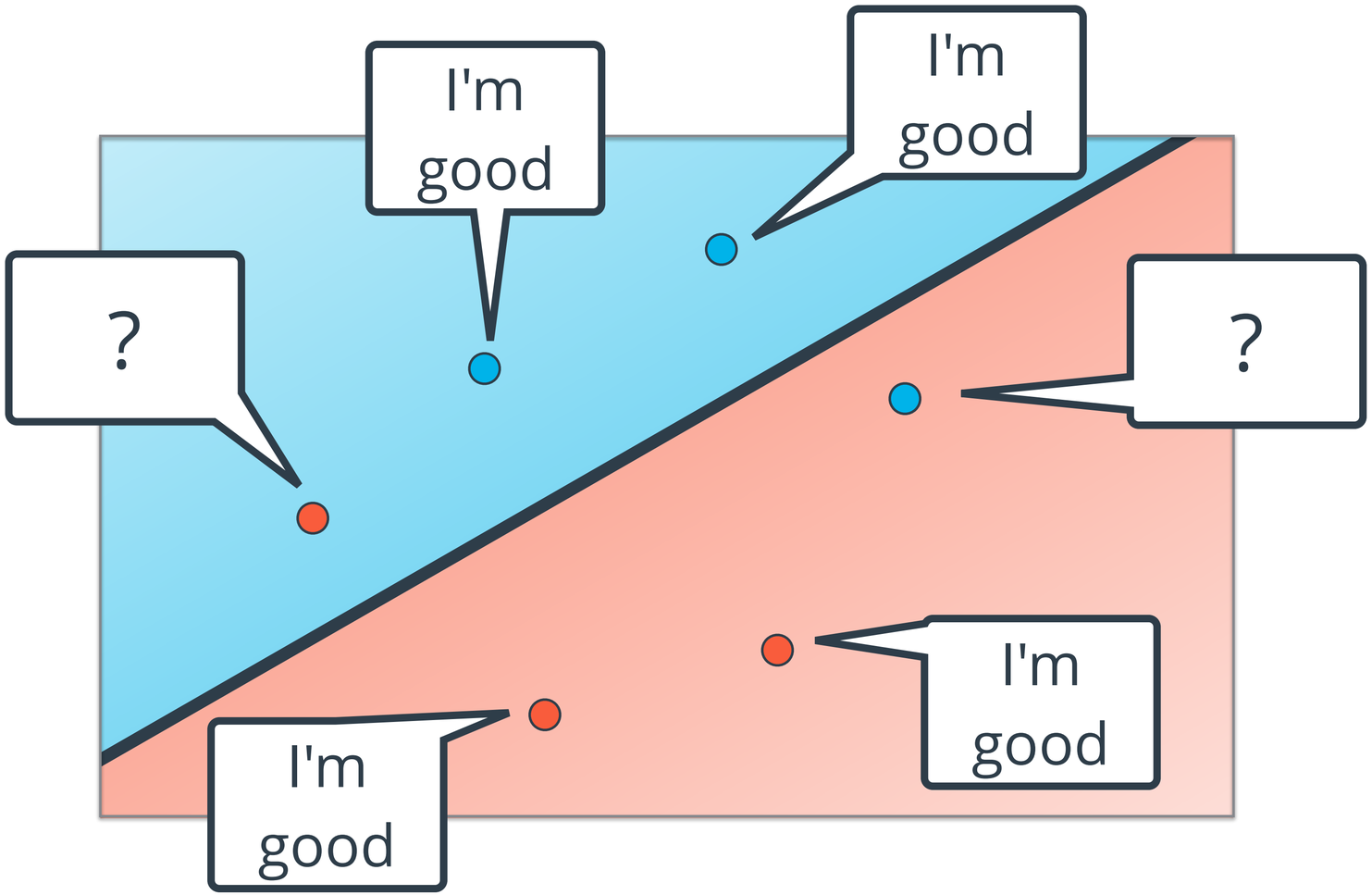


Admissions Office

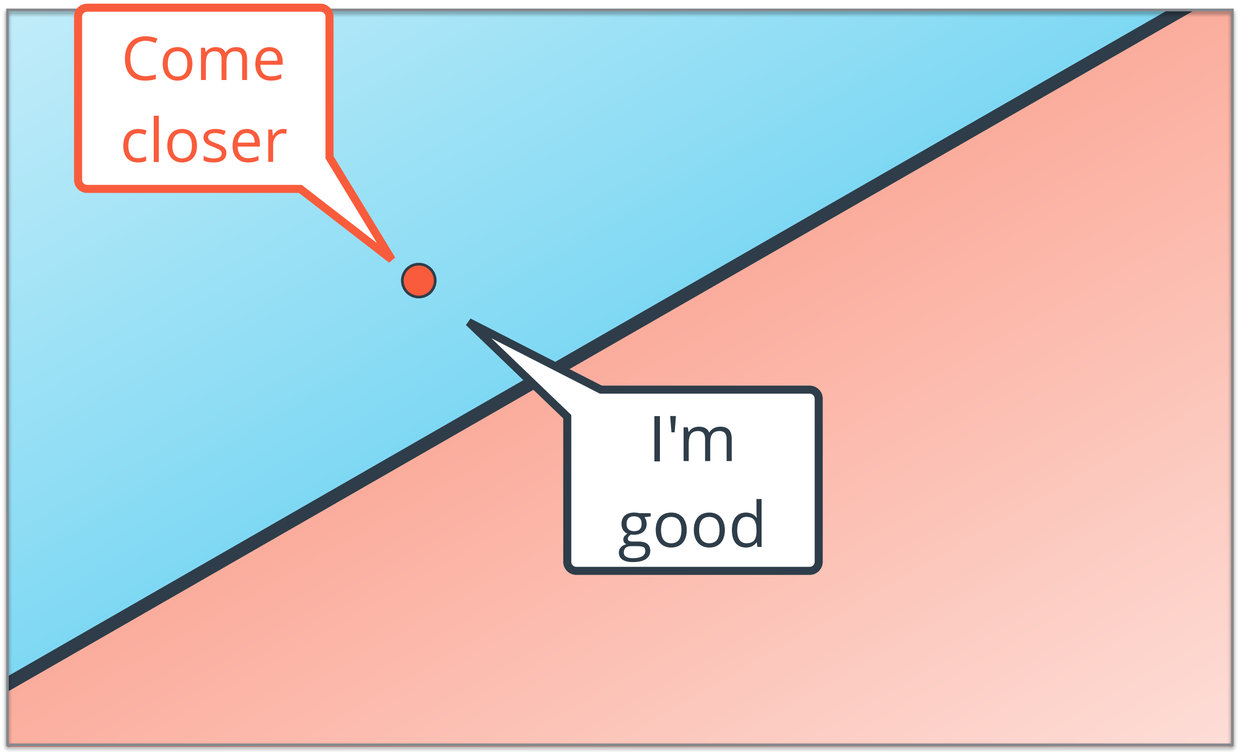


Question:
How do we find this line?

Goal: Split Data



Goal: Split Data

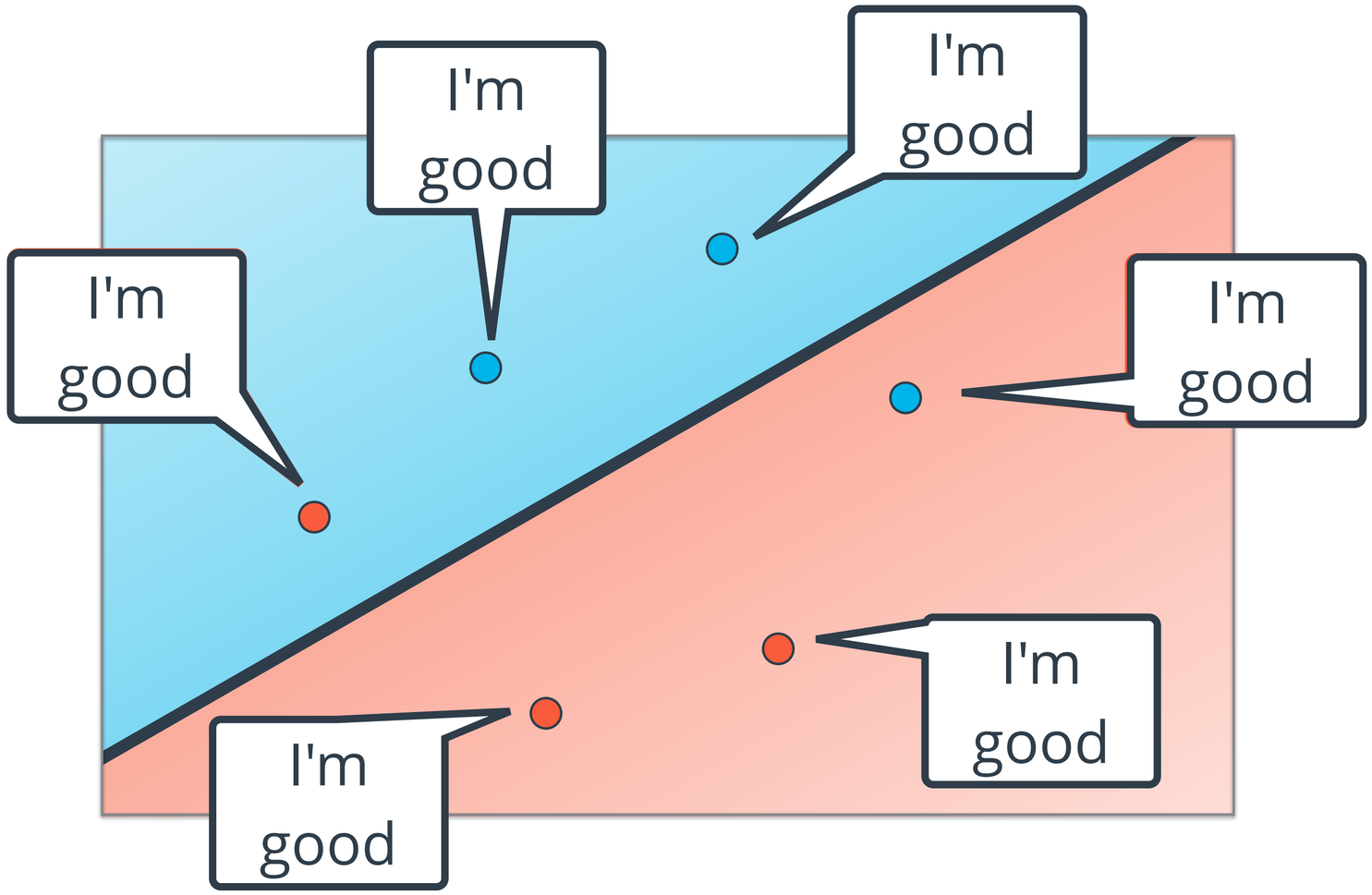


Quiz

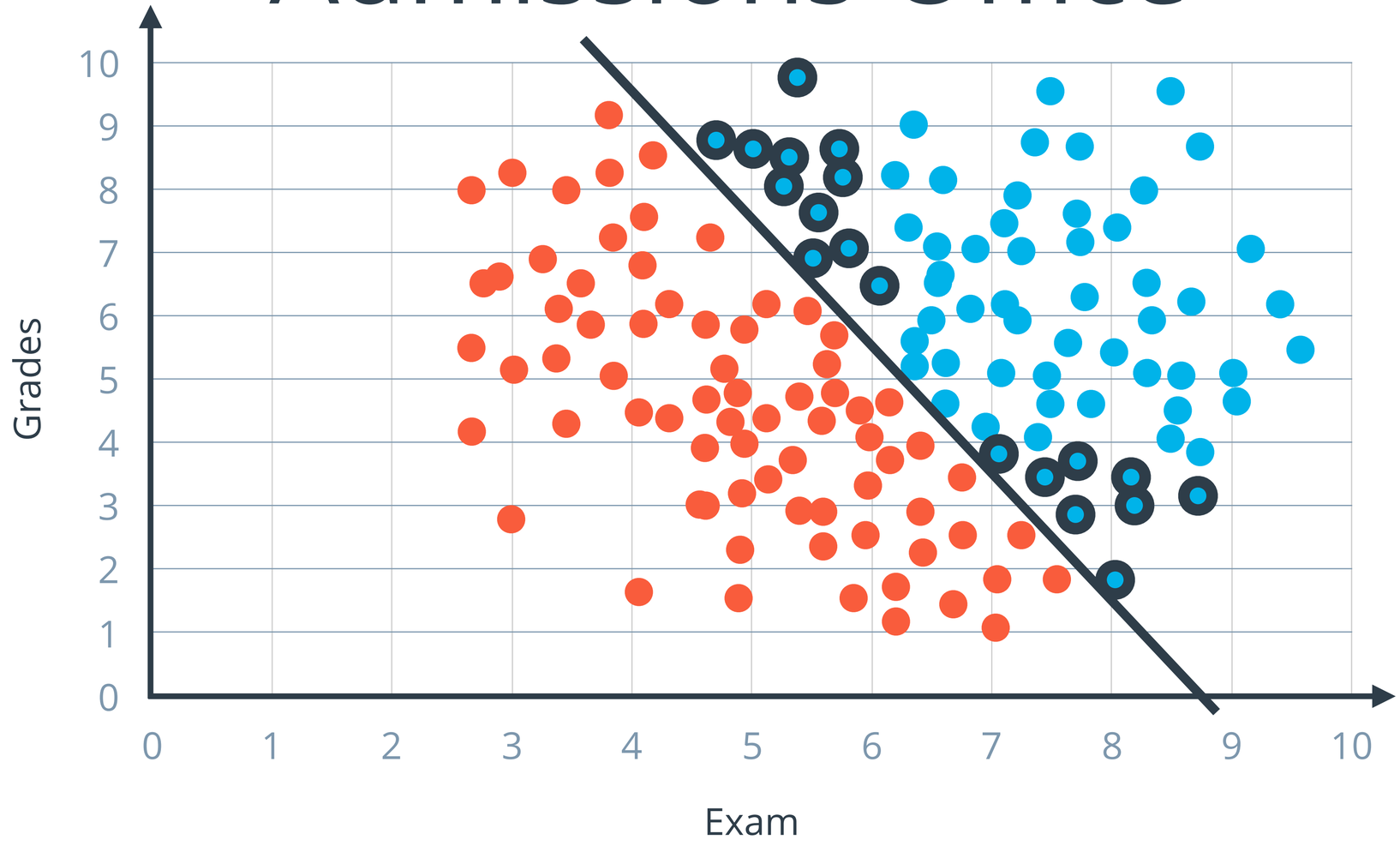
Where would the misclassified point want the line to move?

- (●) Closer
- ( ) Farther

Perceptron Algorithm



Admissions Office

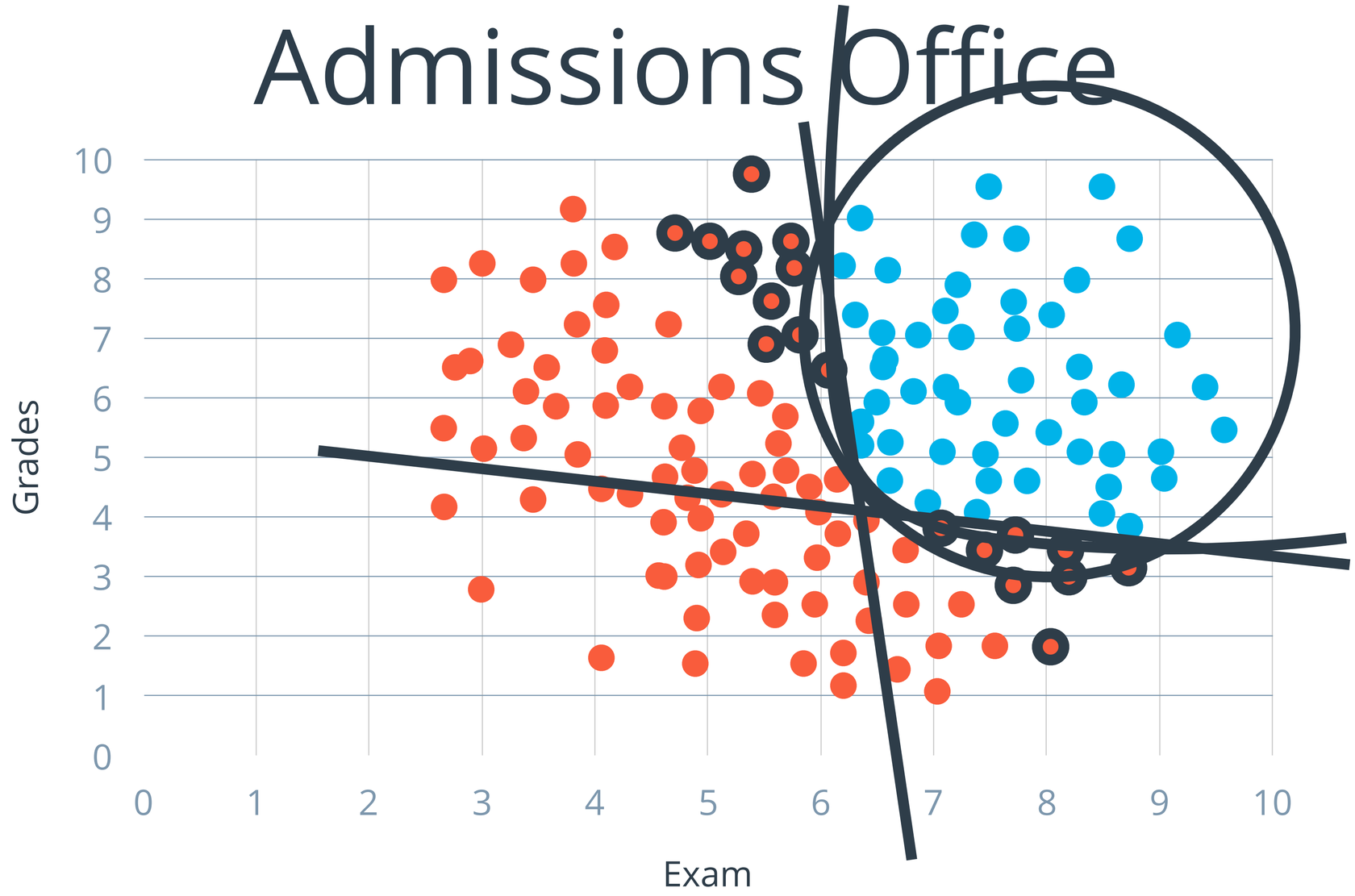


Student 4

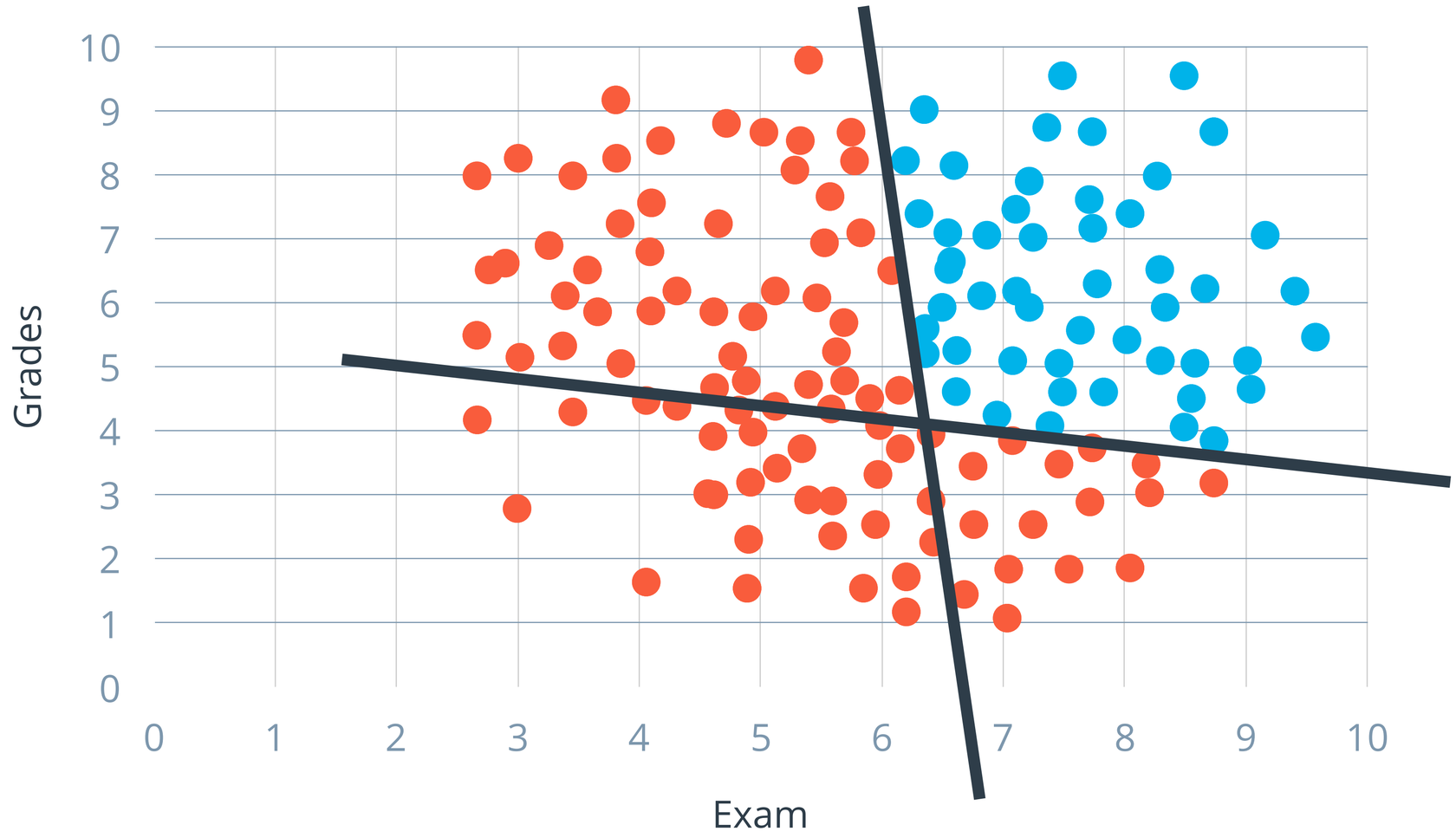
Exam: 10

Grades: 0

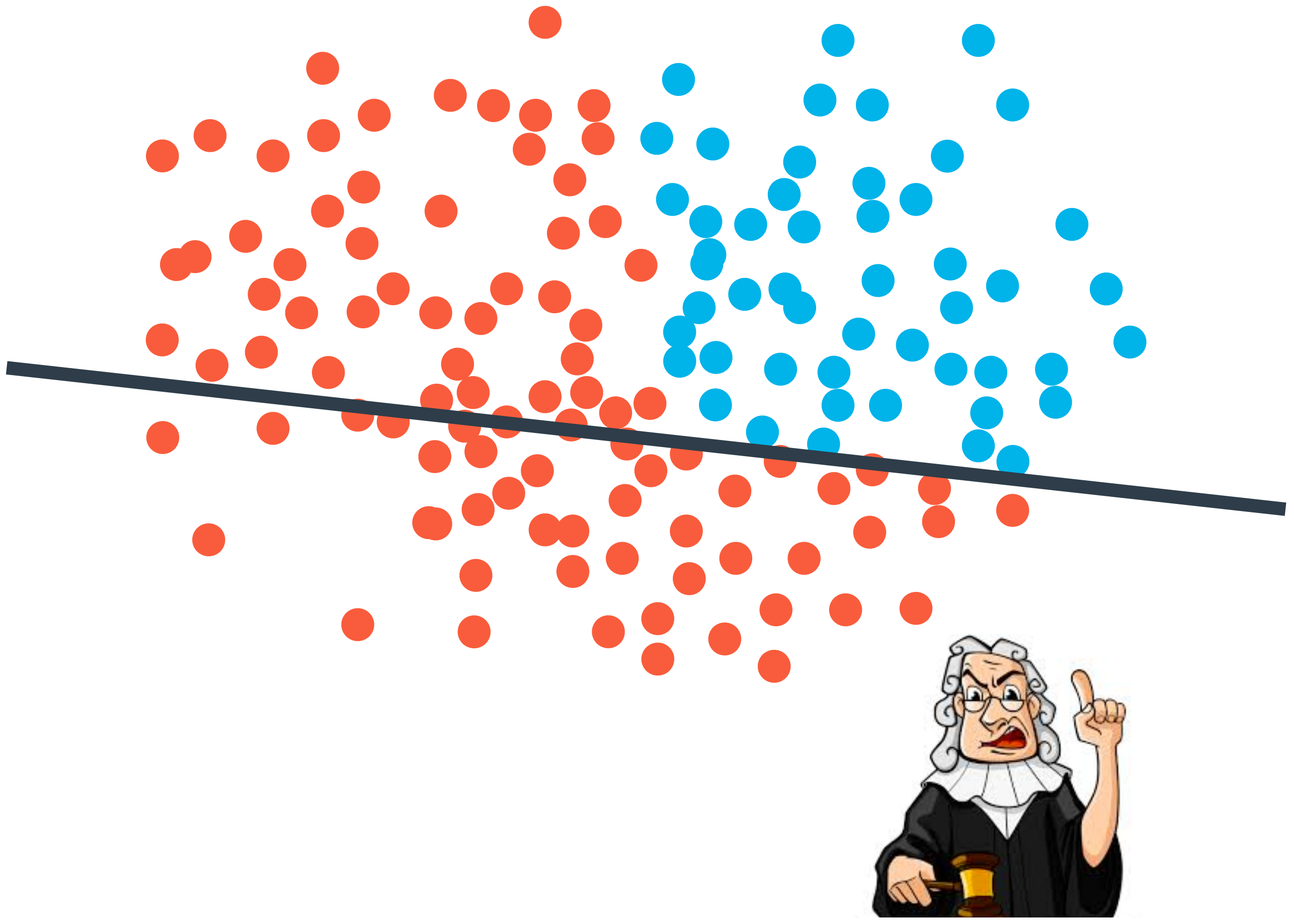
Admissions Office



Admissions Office

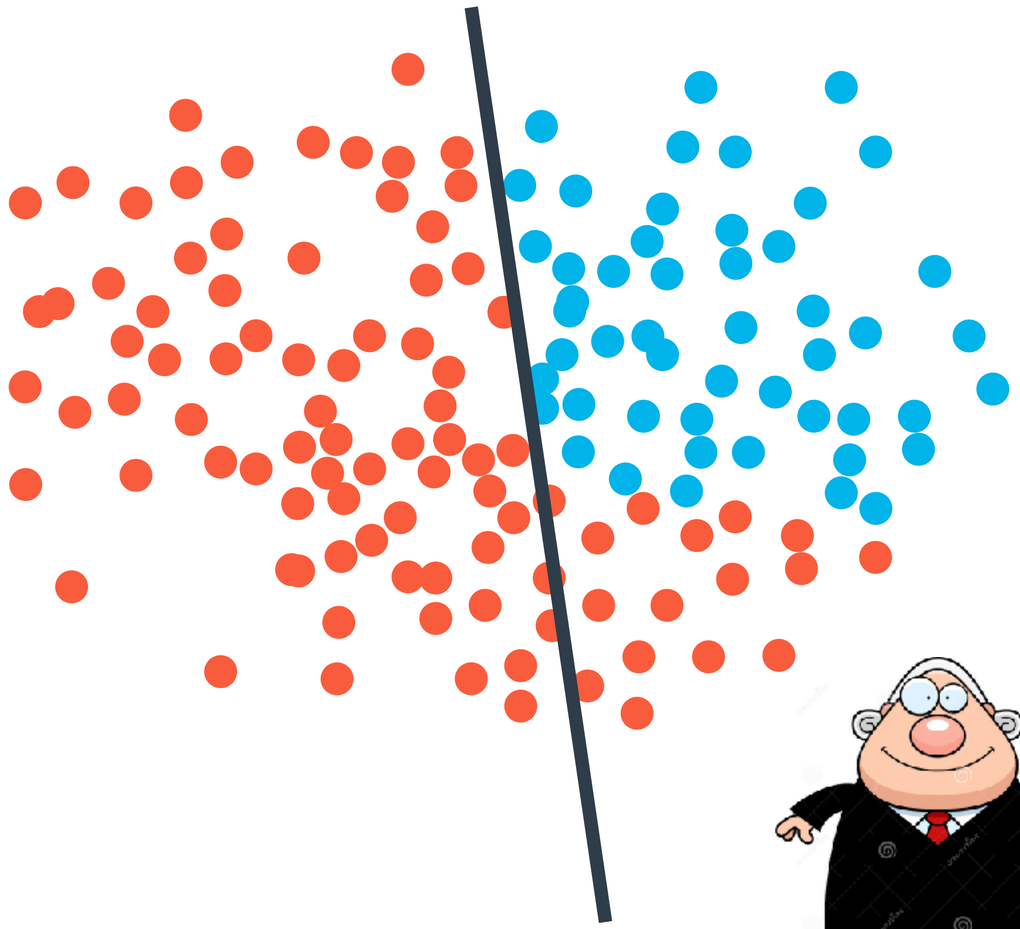


Admissions Office



Judge 1

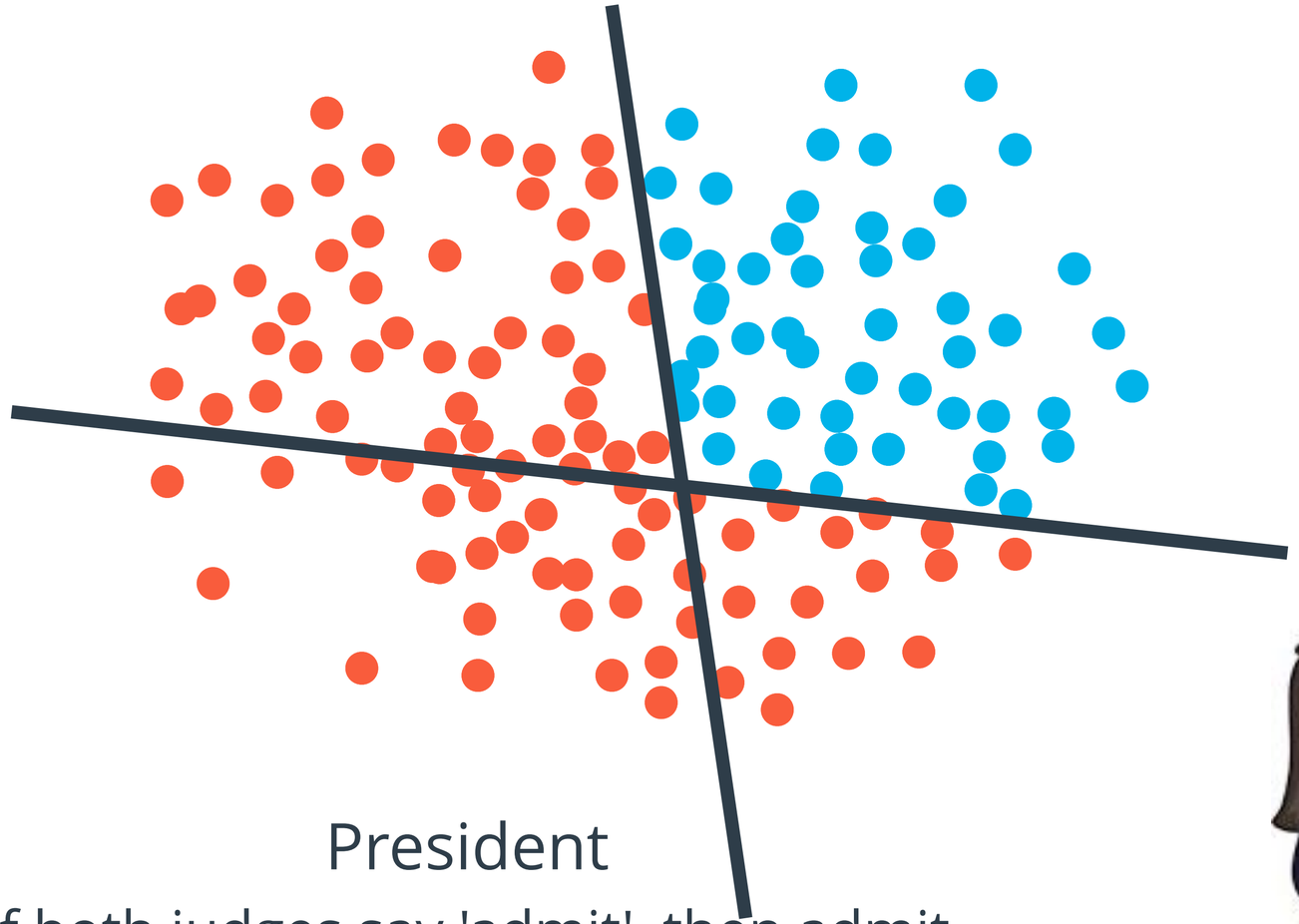
Score = 1*Exam + 8*Grades



Judge 2

Score = 7*Exam + 2*Grades

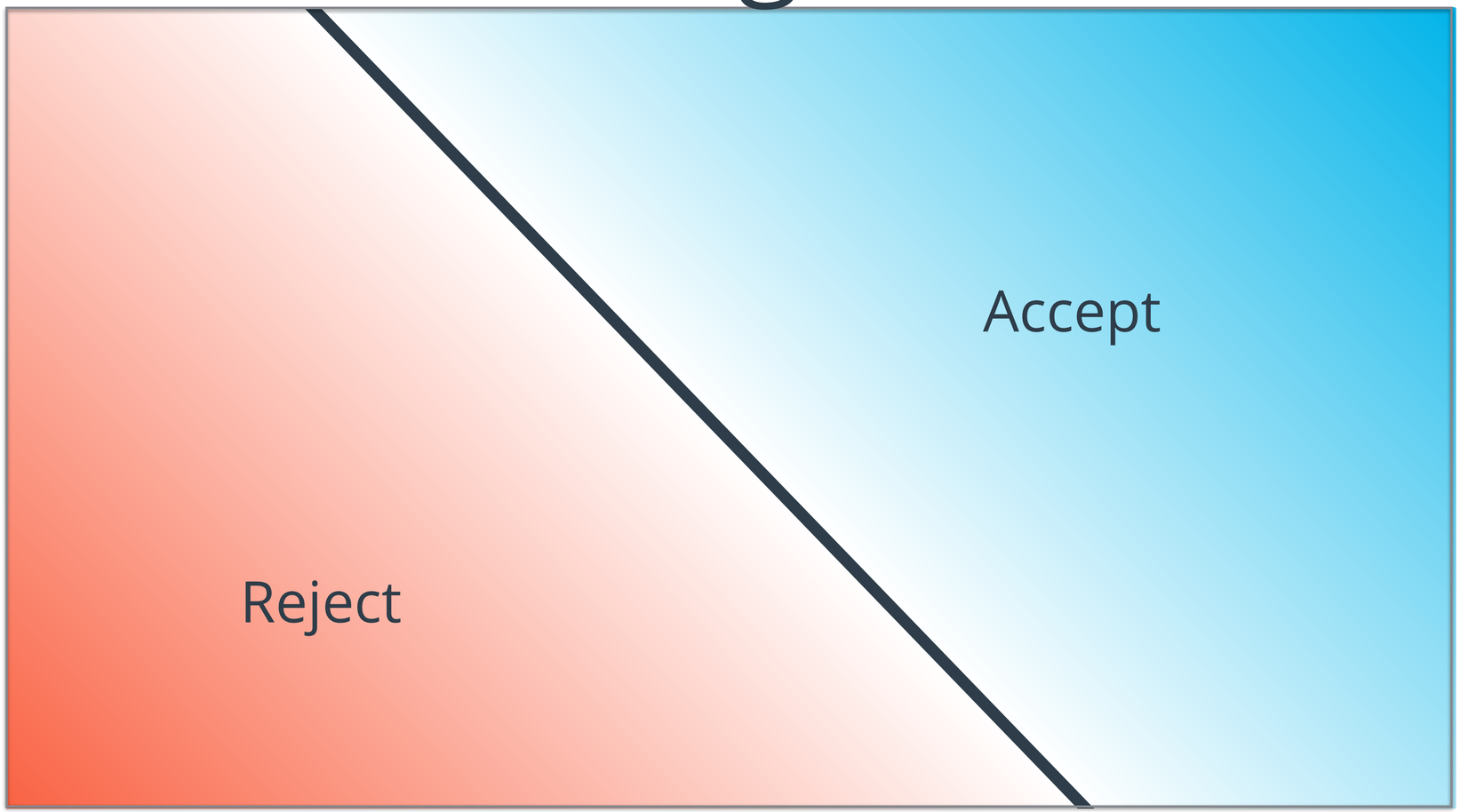
Admissions Office



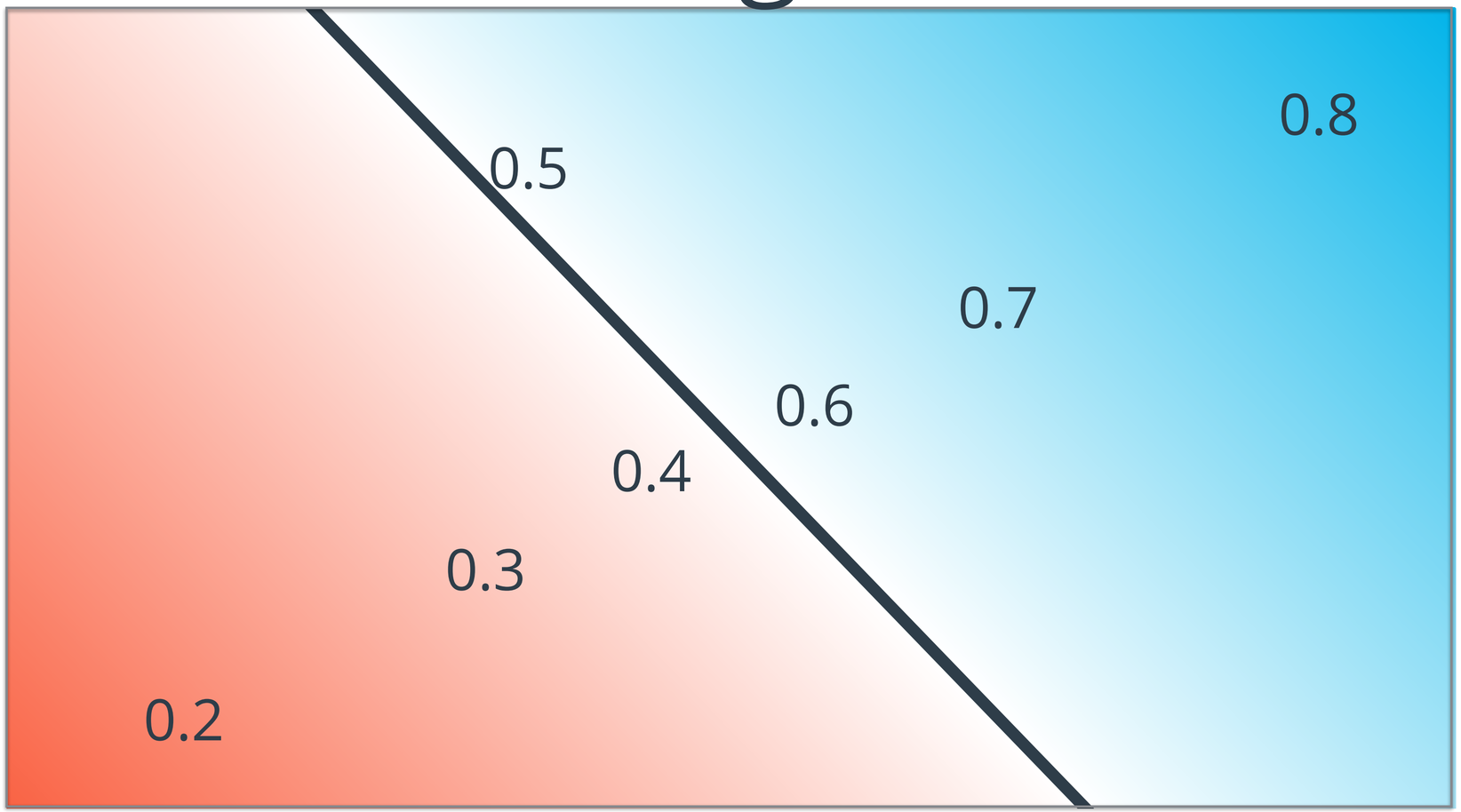
President

If both judges say 'admit', then admit

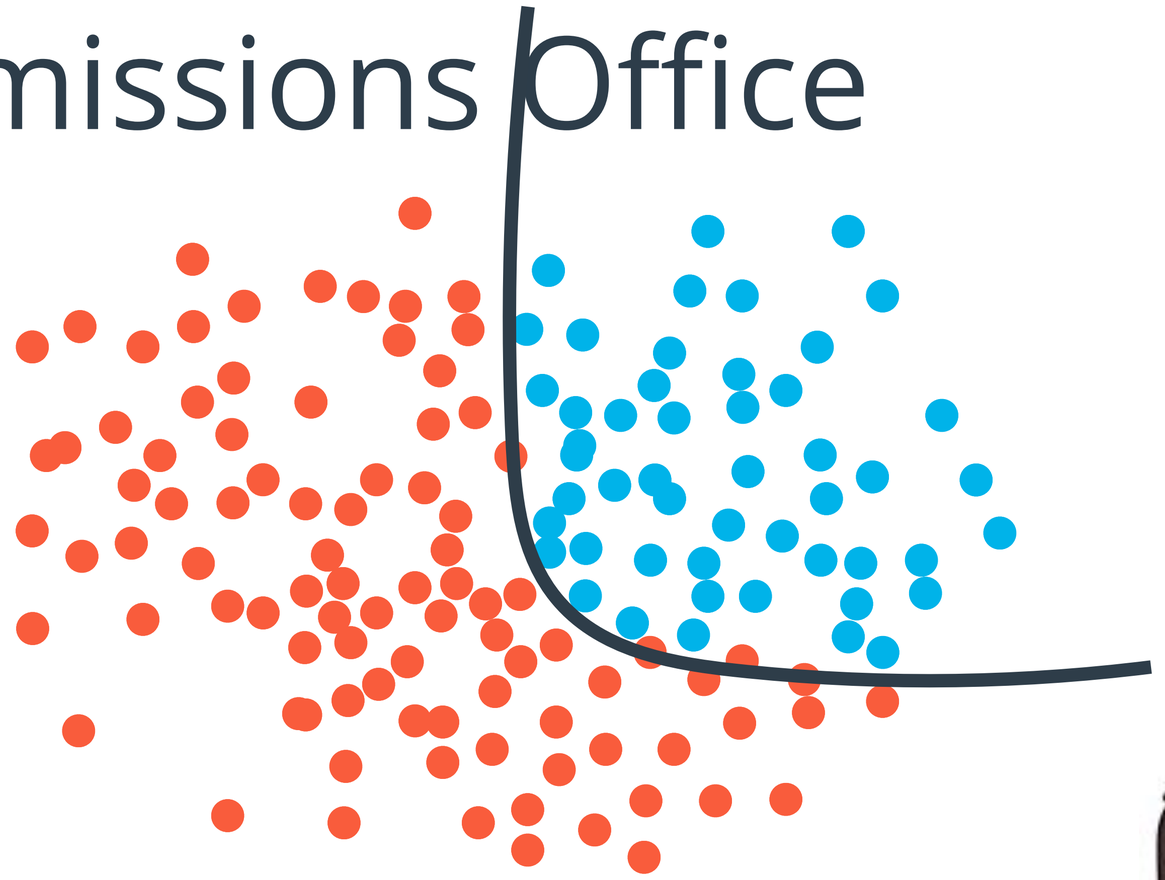
Normalizing the Score



Normalizing the Score



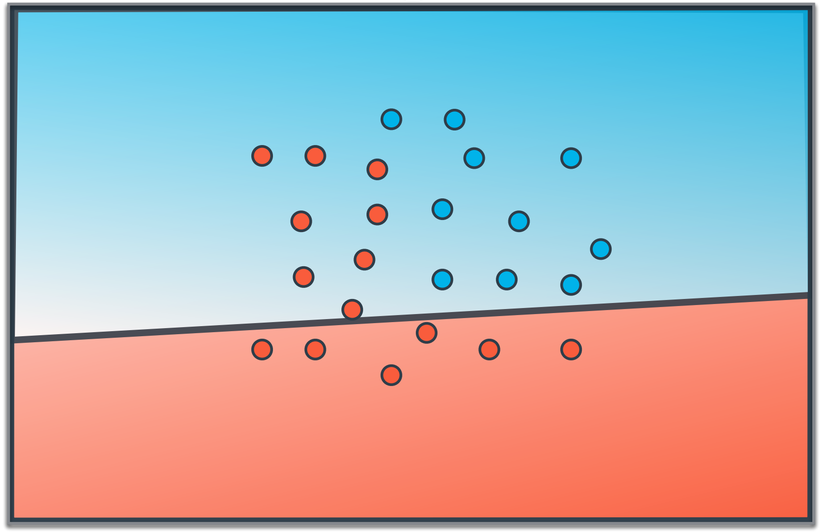
Admissions Office



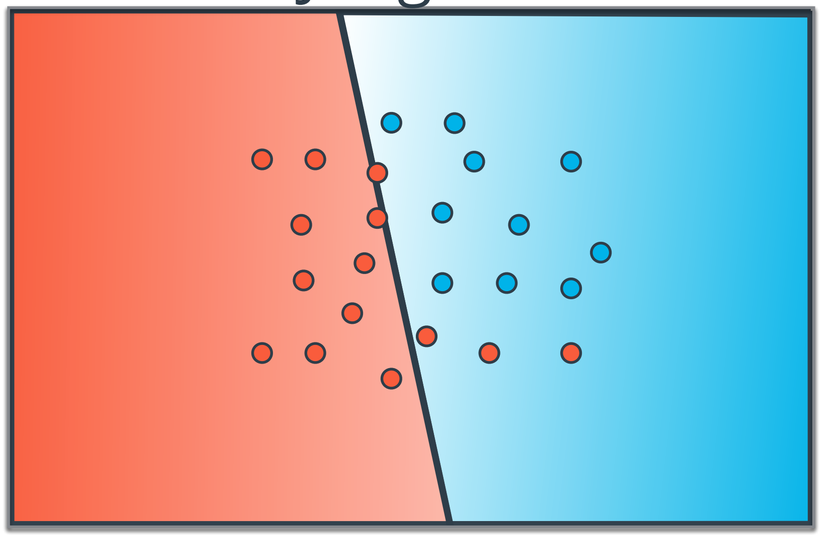
President

Score = 2*(Judge 1 Score) + 3*(Judge 2 Score)

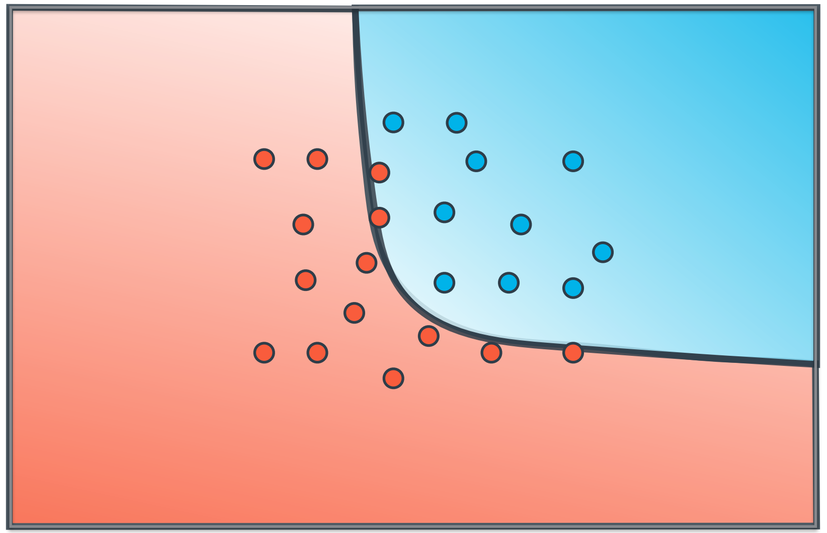
Judge 1



Judge 2



President



Neural Network

Exam

Grades

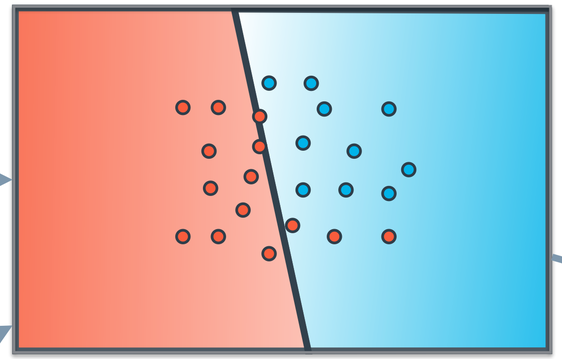
1

7

8

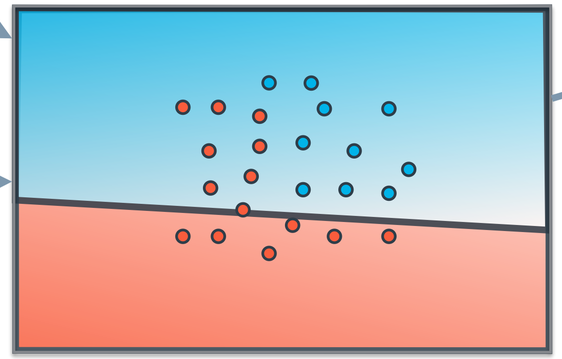
2

Judge 1



Score = **1***Exam + **8***Grades

Judge 2

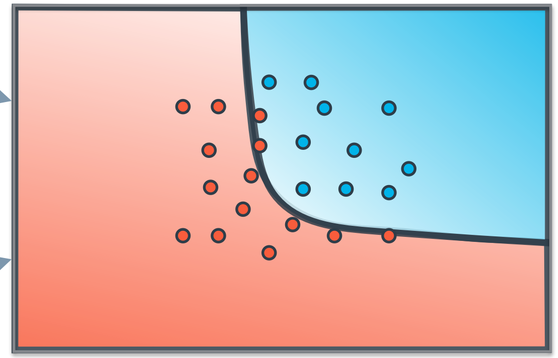


Score = **7***Exam + **2***Grades

2

3

President



Score = **2***(Judge 1 score)
+ **3***(Judge 2 score)

Neural Network

Judge 1

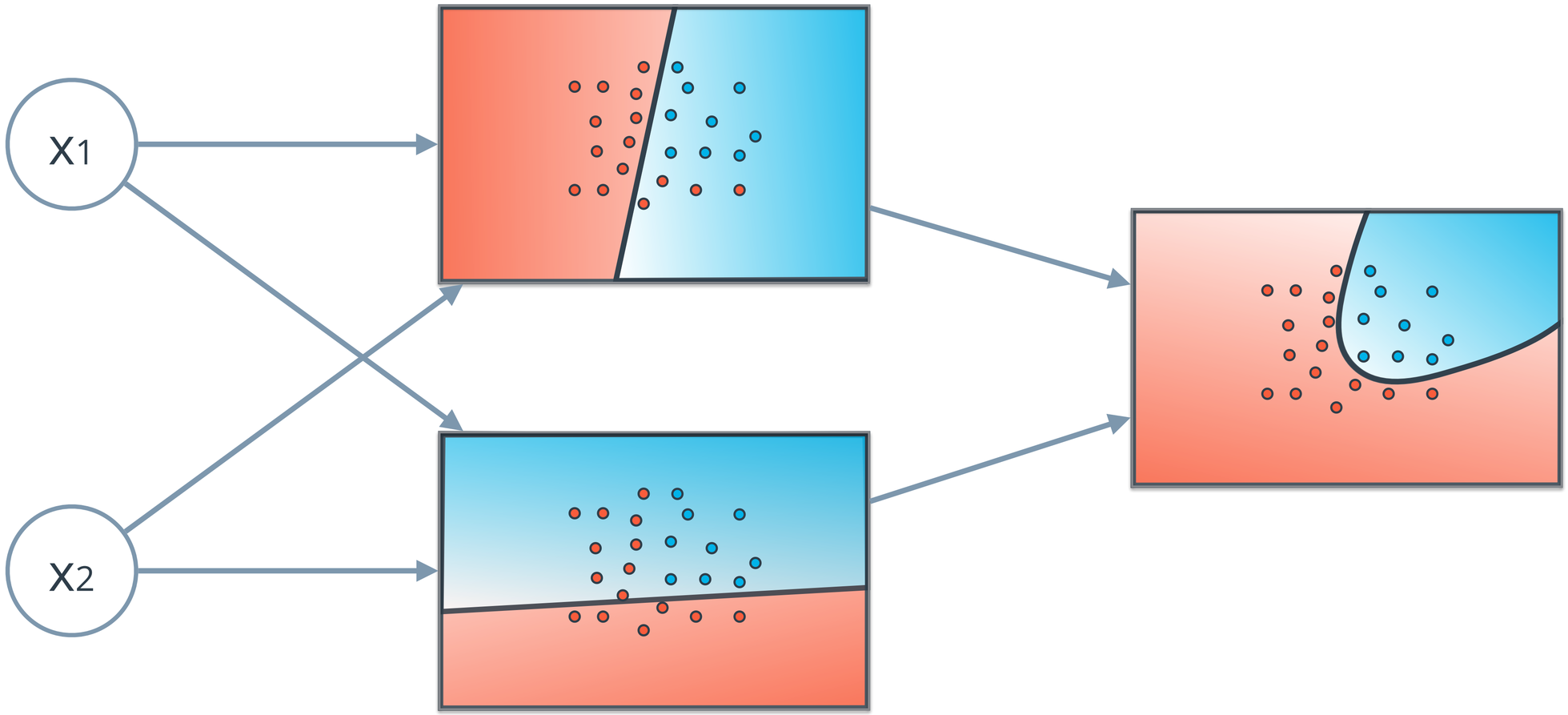
Exam

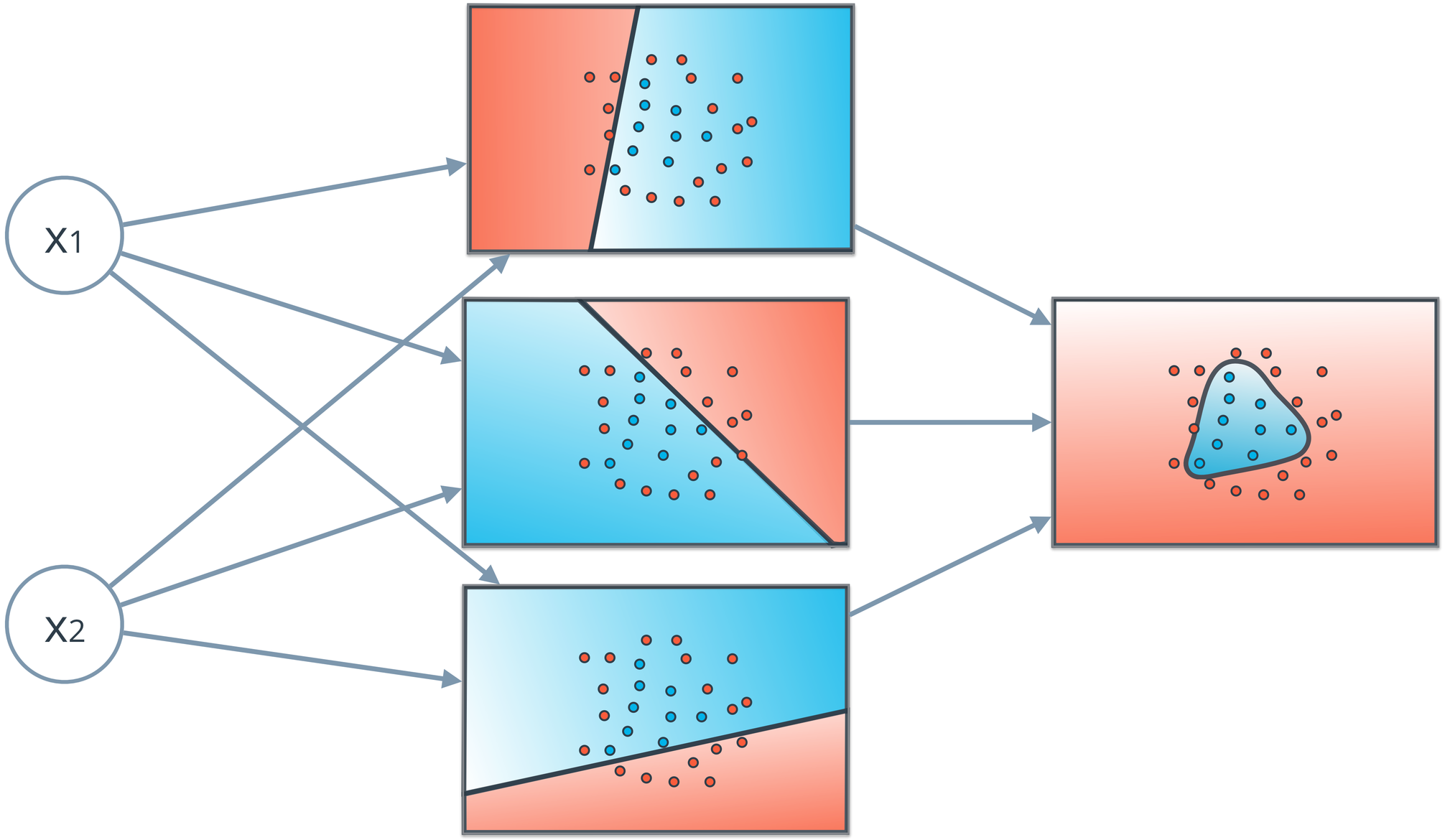
President

Judge 2

Grades

Neural Network

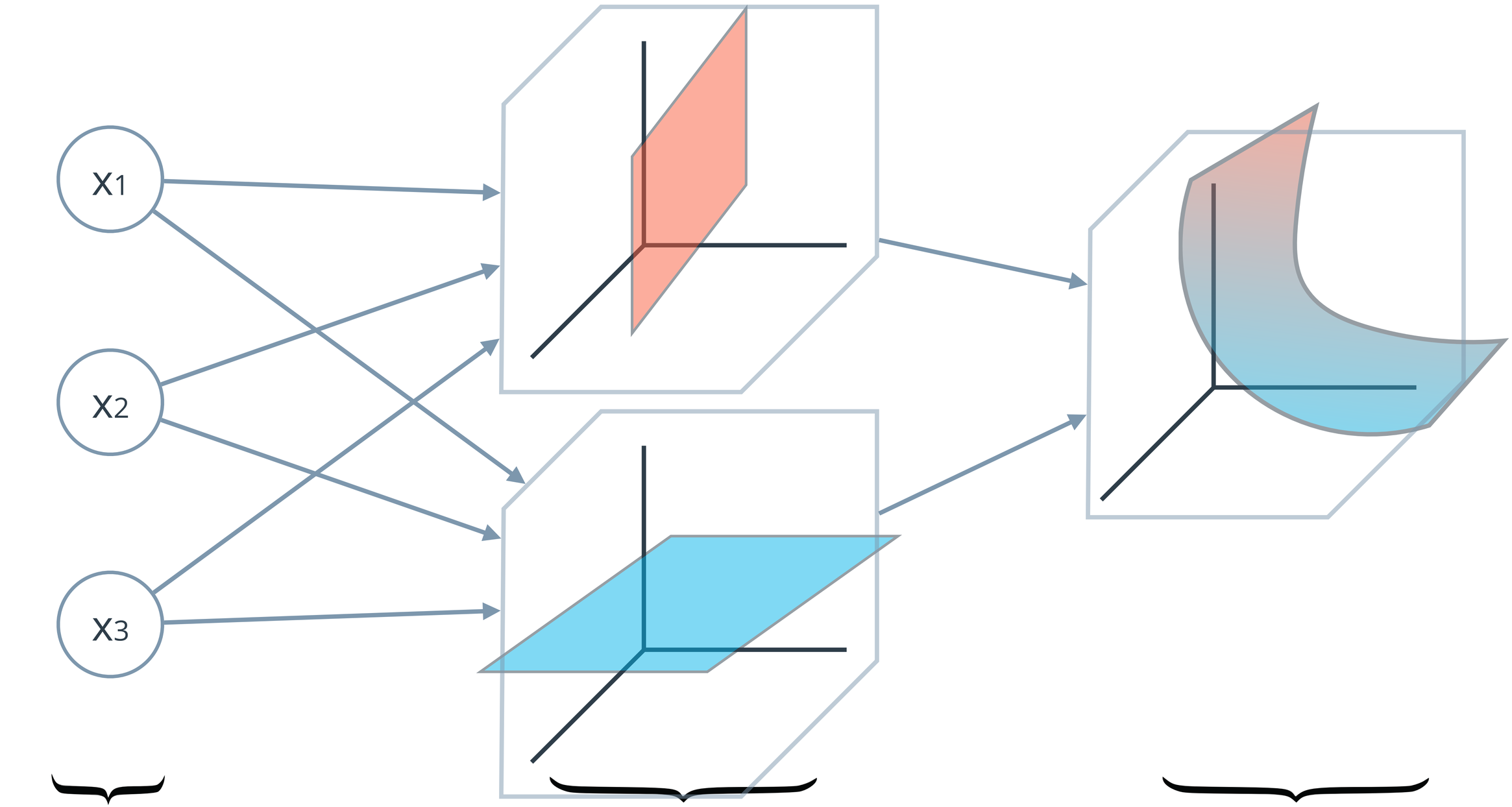


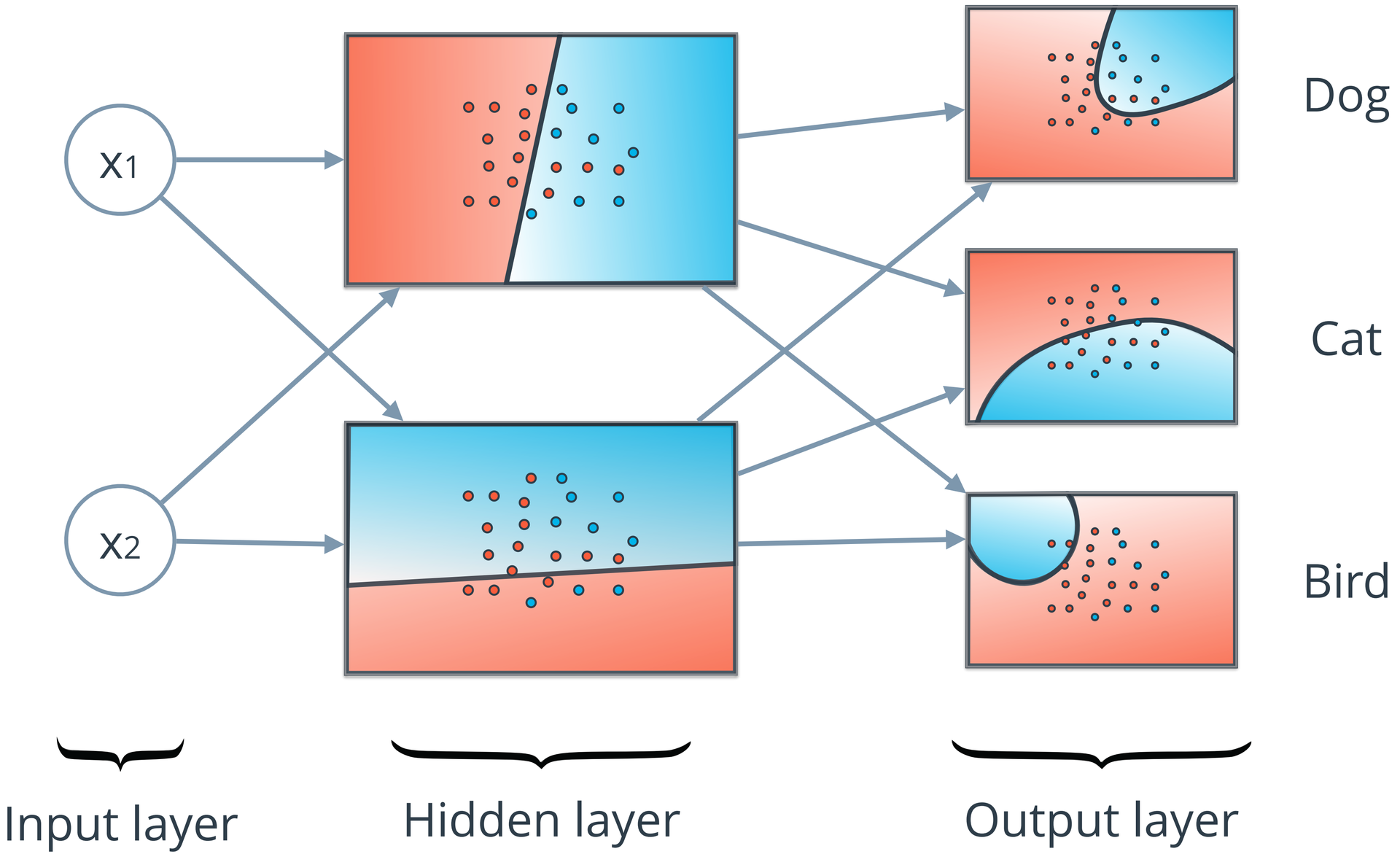


Input layer

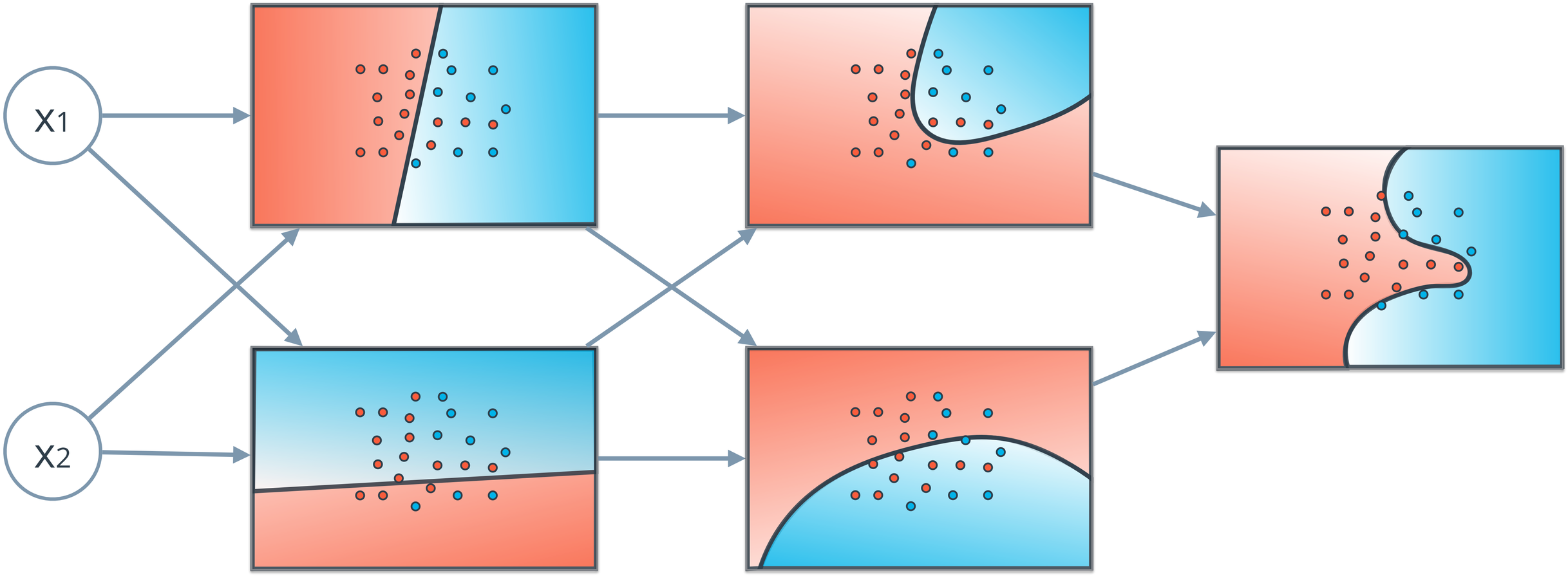
Hidden layer

Output layer

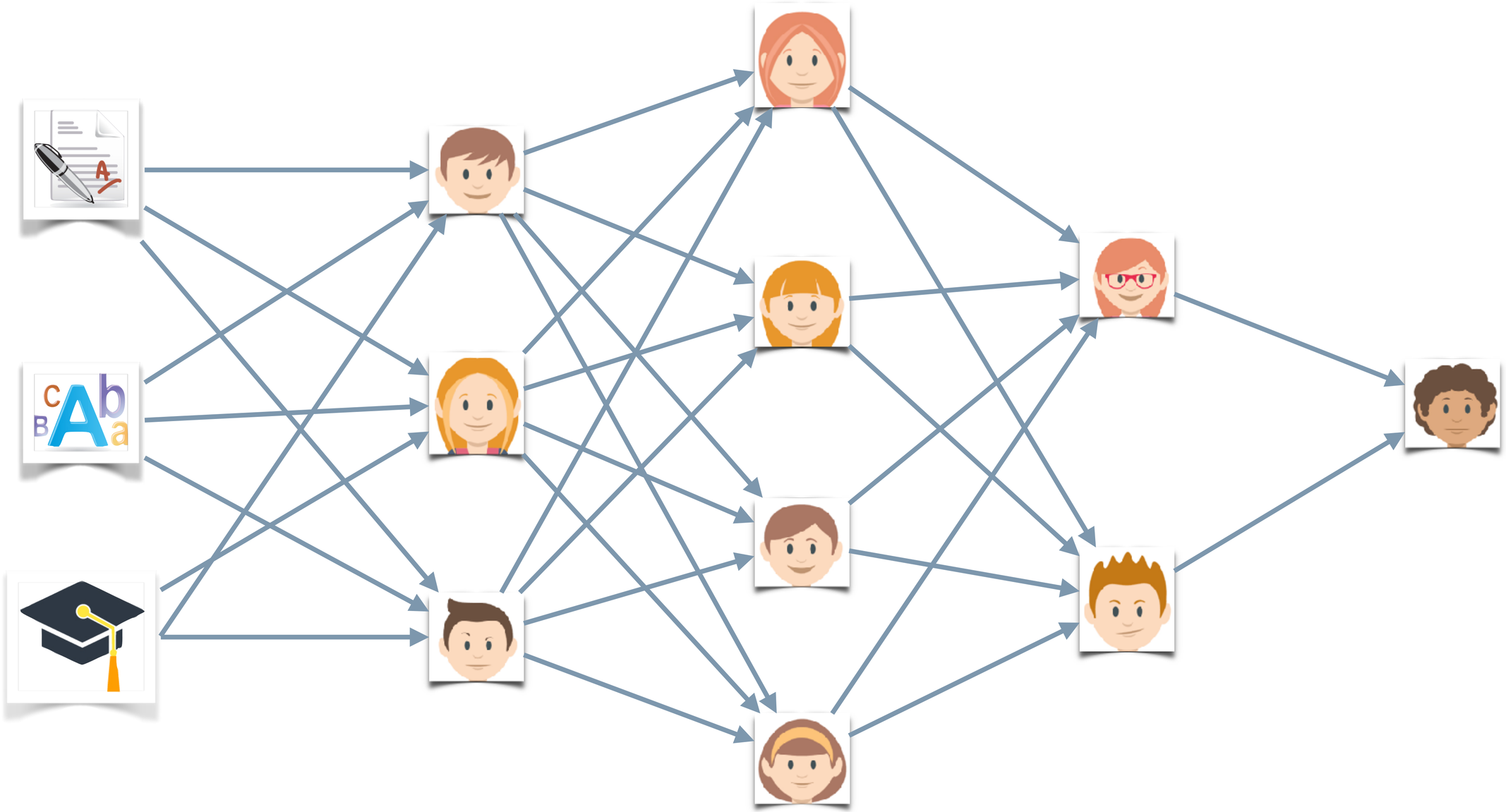

x1
x2
x3
Input layer
Hidden layer
Output layer

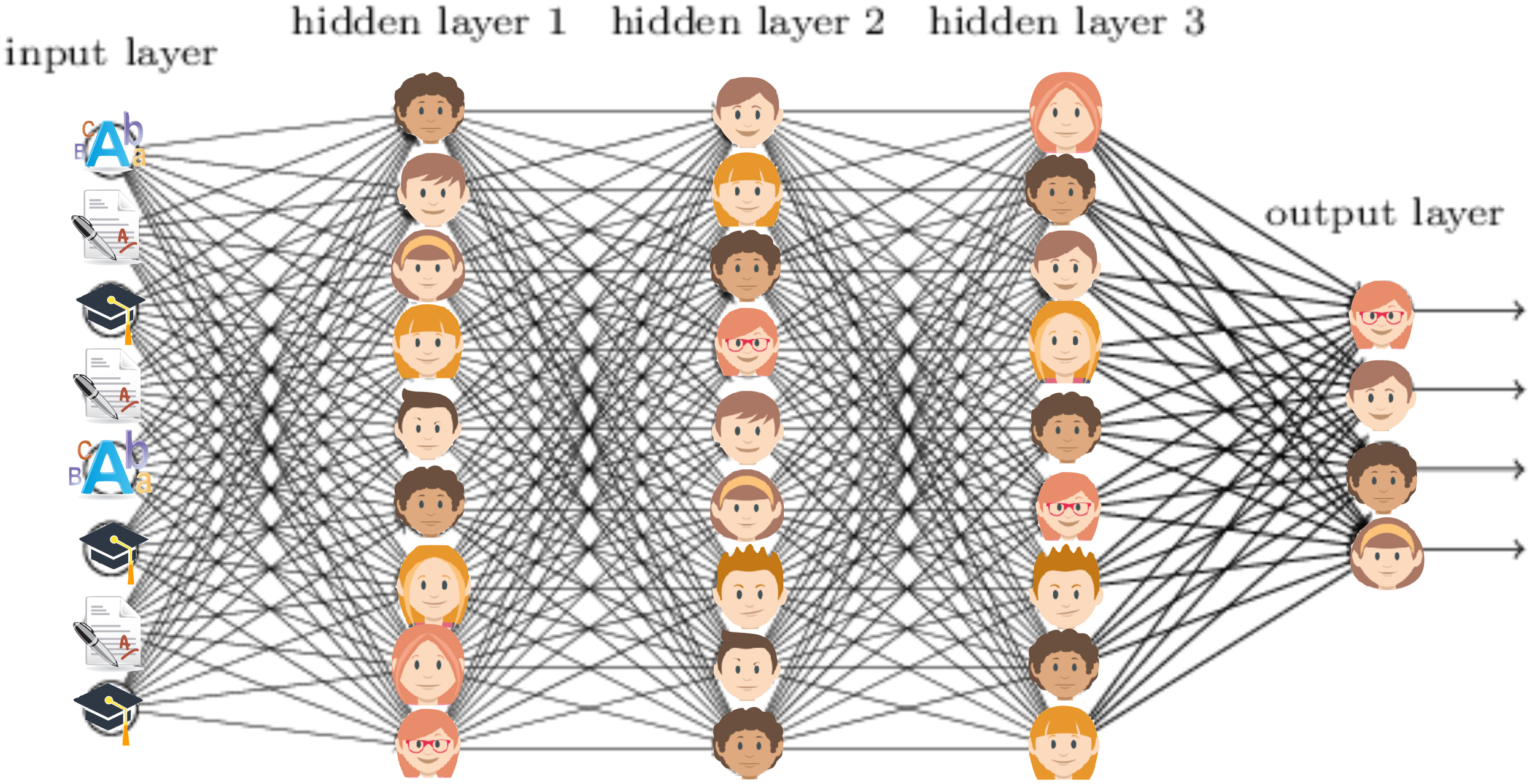

X1
X2
Dog
Cat
Bird
Input layer
Hidden layer
Output layer

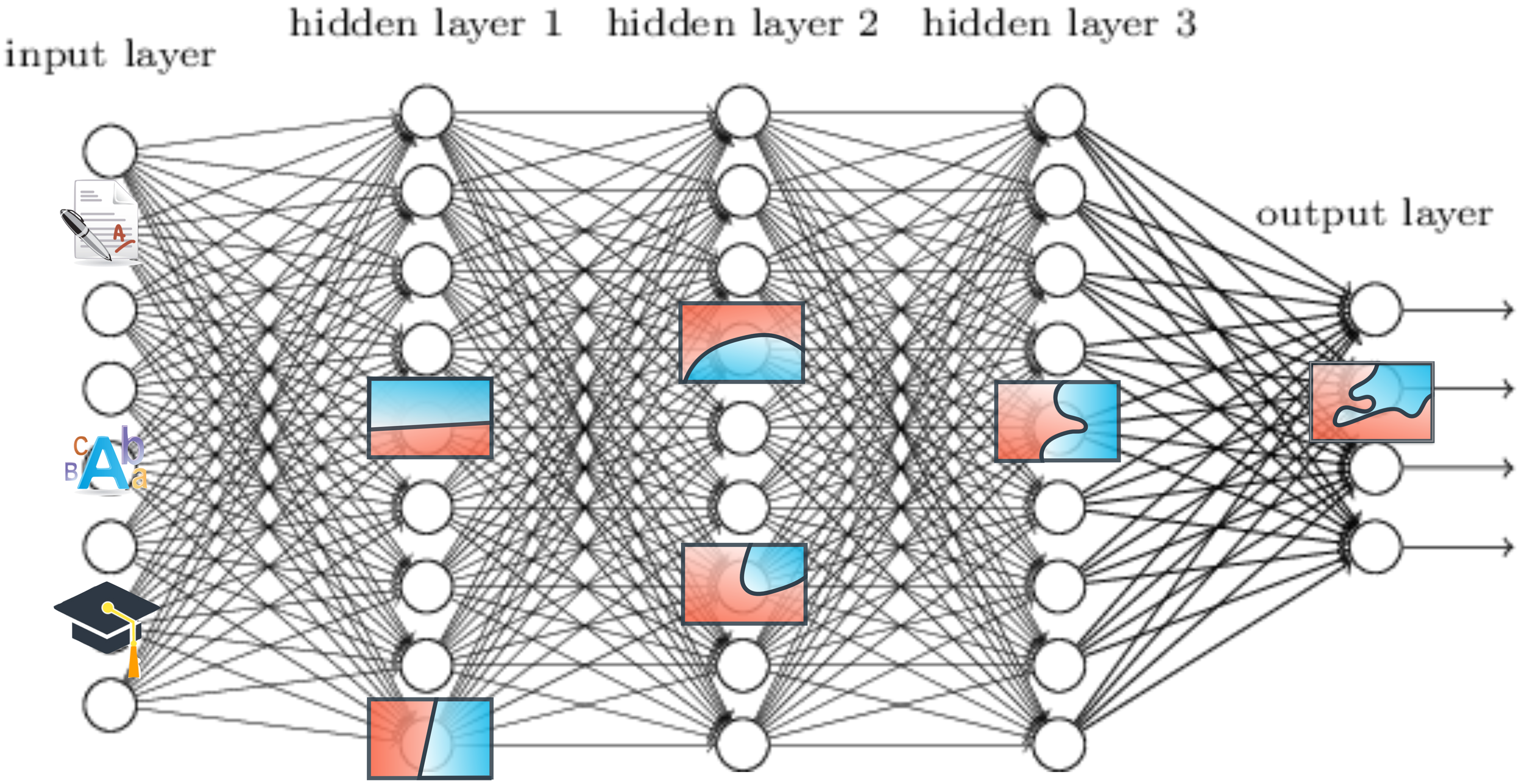
Deep Neural Network



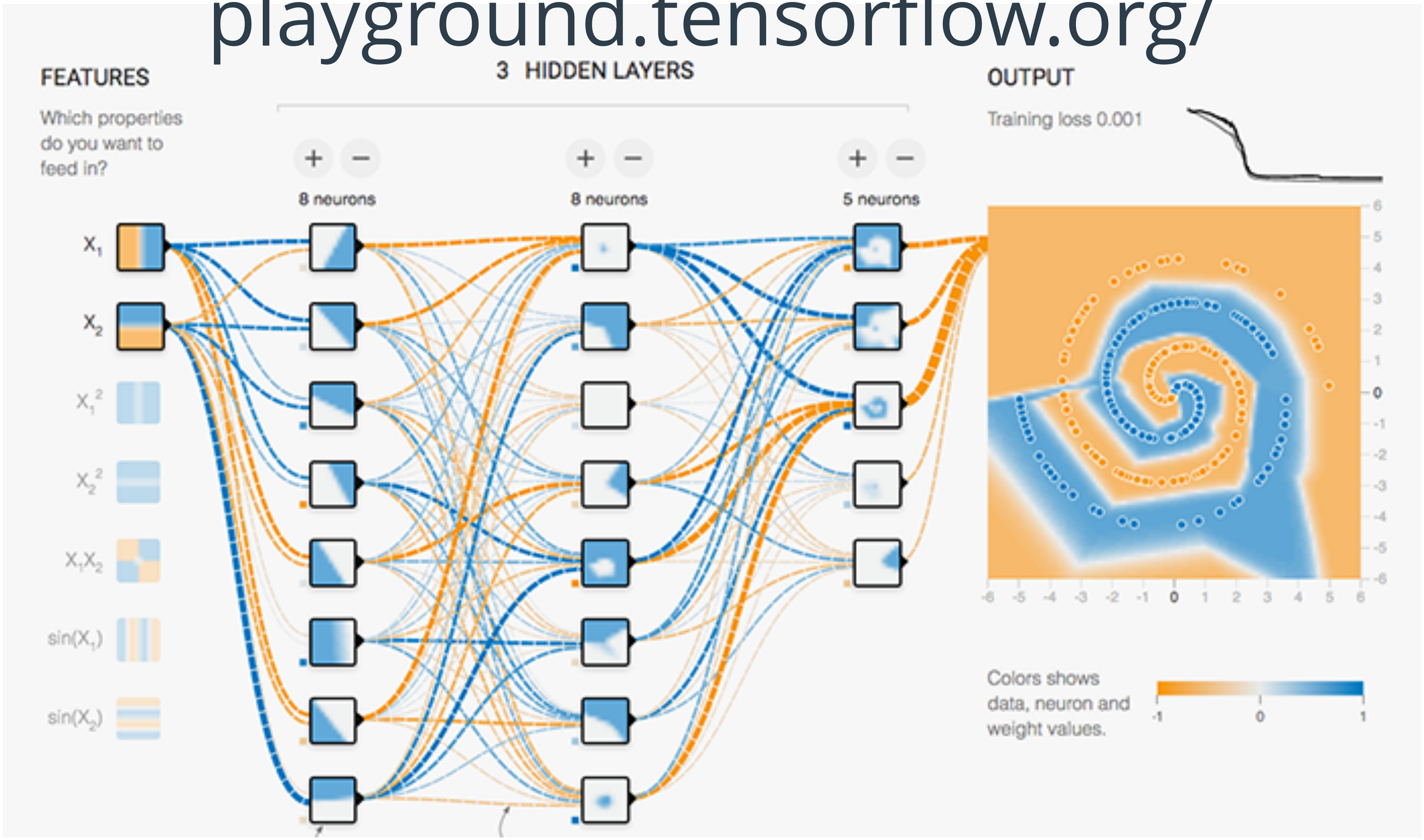
Neural Network



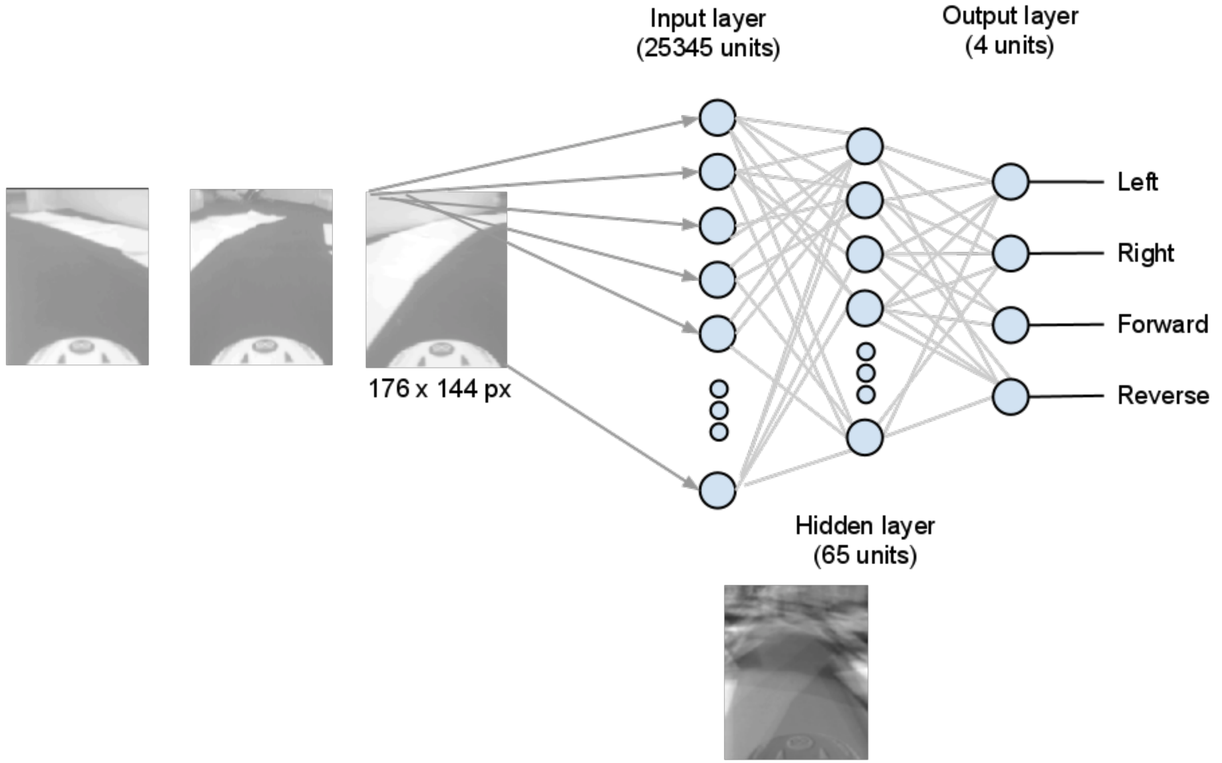

input layer
hidden layer 1
hidden layer 2
hidden layer 3
output layer


input layer
hidden layer 1
hidden layer 2
hidden layer 3
output layer

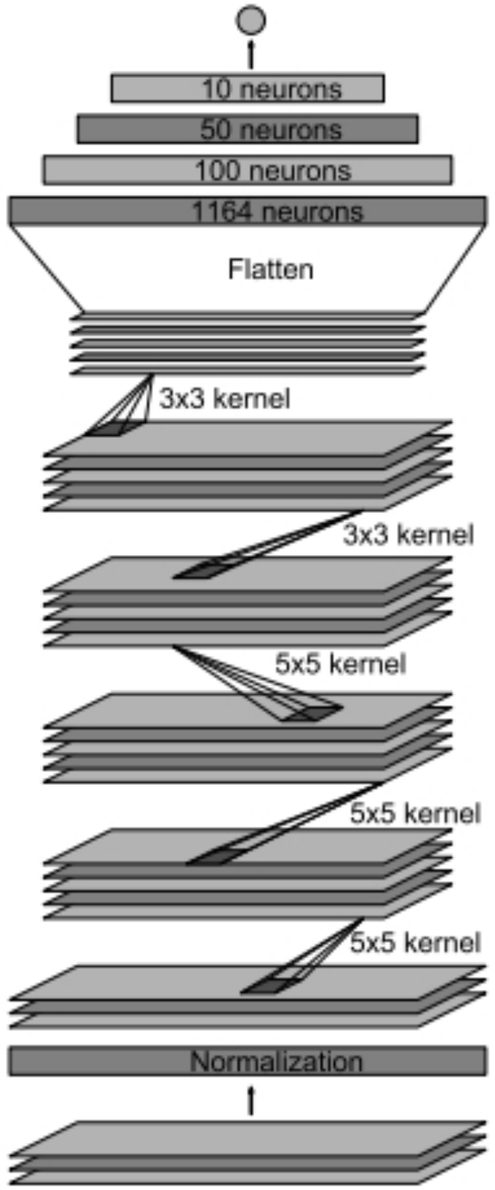
playground.tensorflow.org/



Self Driving Car

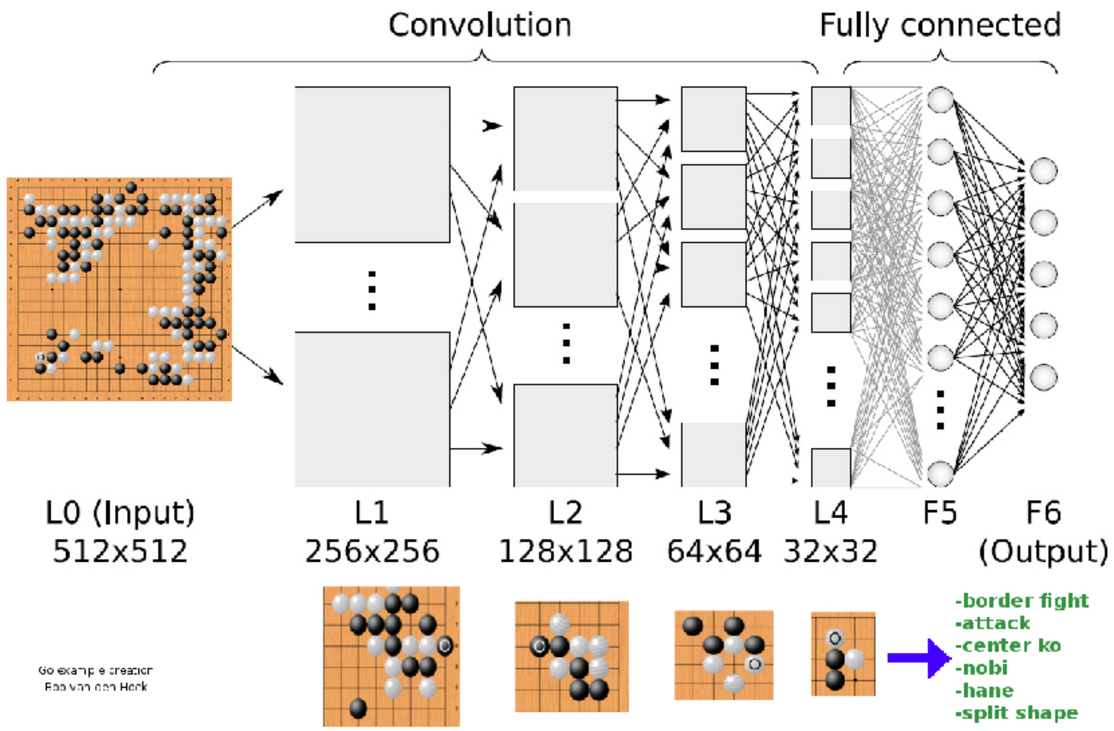


Self Driving Car



Playing Go


ALPHAGO
00:05:30
Google DeepMind
Challenge Match
LEE SEDOL
00:28:28


Convolution
Fully connected
L0 (Input)
512x512
L1
256x256
L2
128x128
L3
64x64
L4
32x32
F5
F6
(Output)
-border fight
-attack
-center ko
-nobi
-hane
-split shape

Magic Hat

Exam: 8

Grades: 7

Medical Applications

Temperature
Symptoms
Etc.

Stock Market

GOOG: 7.32

AAPL: 3.14

MSFT: 1.32

Computer Vision

YouTube

Age
Location
Watched: Despacito
Watched: Pitbull video

Spam Detection

Hello,
it’s grandma!

Spam Detection

E@rn c@$h quickly!

Lab: Sentiment Analysis

Classify Movie Reviews

Notebook: IMDB_Keras.ipynb

Sentiment Analysis Lab

Sentiment Analysis

What a great movie!

That was terrible.

One-hot encoding

What a great movie!

a	aardvark	cookie	great	figurine	movie	rhinoceros	what	zygote
1	0	0	**1**	0	**1**	0	**1**	0

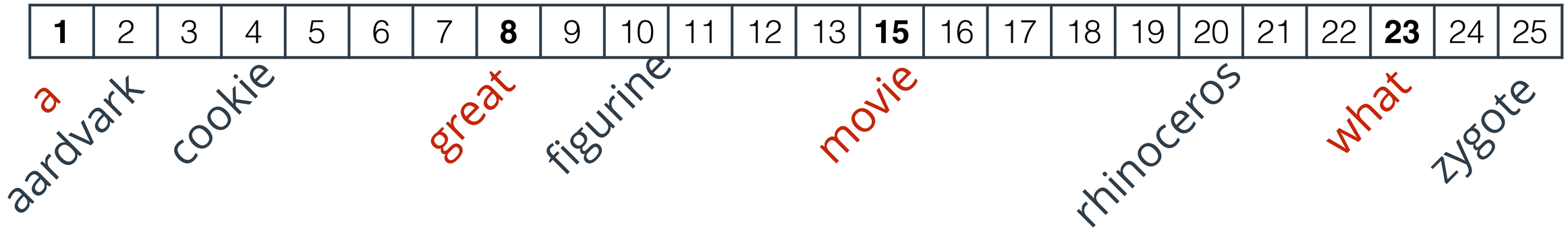
1. Loading the data

```
# Loading the data (it's preloaded in Keras)
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=1000)

print(x_train.shape)
print(x_test.shape)
```

2. Examining the data

What a great movie!



[1, 7, 15, 23]

3. One-hot encoding the input

What a great movie!

1	2	3	4	5	6	7	**8**	9	10	11	12	13	**15**	16	17	18	19	20	21	22	**23**	24	25
a	aardvark	cookie				great		figurine					movie				rhinoceros			what		zygote	

[1, 7, 15, 23]

1	0	0	0	0	0	0	**1**	0	0	0	0	0	**1**	0	0	0	0	0	0	0	**1**	0	0

3. One-hot encoding the output

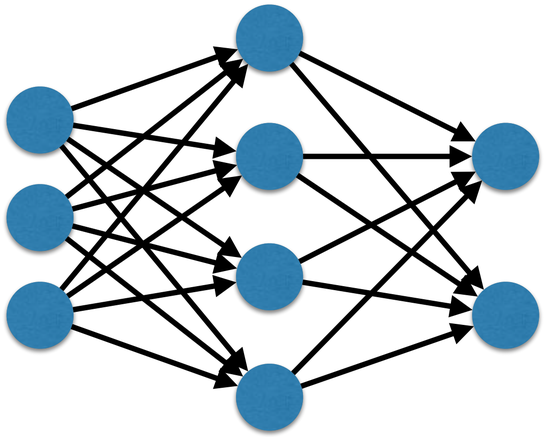
[1,0]

[0,1]

4. Building the Model Architecture

Build the Model

```
model = Sequential()
model.add(Dense(4, activation='relu', input_dim=3))
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))
model.summary()
```



4. Building the Model Architecture

Compile Model

```
model.compile(loss="categorical_crossentropy", optimizer="adam", metrics = ["accuracy"]

model.summary()
```

4. Solution

```
model = Sequential()
model.add(Dense(512, activation='relu', input_dim=1000))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
model.summary()

# Compiling the model using categorical_crossentropy loss, and rmsprop optimizer.
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

5. Training the Model

1. Compile Model

```
model.compile(loss="categorical_crossentropy", optimizer="adam", metrics = ["accuracy"]

model.summary()
```

2. Fit Model

```
model.fit(X, y, nb_epoch=1000, verbose=0)
```

```
Train on 25000 samples, validate on 25000 samples
Epoch 1/10
9s - loss: 0.3969 - acc: 0.8260 - val_loss: 0.3429 - val_acc: 0.8568
Epoch 2/10
9s - loss: 0.3339 - acc: 0.8670 - val_loss: 0.3413 - val_acc: 0.8632
Epoch 3/10
9s - loss: 0.3219 - acc: 0.8778 - val_loss: 0.3552 - val_acc: 0.8614
Epoch 4/10
9s - loss: 0.3110 - acc: 0.8853 - val_loss: 0.3718 - val_acc: 0.8602
Epoch 5/10
9s - loss: 0.3056 - acc: 0.8920 - val_loss: 0.4086 - val_acc: 0.8542
Epoch 6/10
10s - loss: 0.2951 - acc: 0.8983 - val_loss: 0.3938 - val_acc: 0.8608
Epoch 7/10
9s - loss: 0.2864 - acc: 0.9037 - val_loss: 0.4258 - val_acc: 0.8566
Epoch 8/10
9s - loss: 0.2738 - acc: 0.9100 - val_loss: 0.4733 - val_acc: 0.8509
Epoch 9/10
8s - loss: 0.2622 - acc: 0.9162 - val_loss: 0.4658 - val_acc: 0.8536
Epoch 10/10
12s - loss: 0.2520 - acc: 0.9216 - val_loss: 0.4877 - val_acc: 0.8583
```

Building the Neural Network

```
# Building the model architecture with one layer of length 100
model = Sequential()
model.add(Dense(512, activation='relu', input_dim=1000))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
model.summary()

# Compiling the model using categorical_crossentropy loss, and rmsprop optimizer.
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

Running the Neural Network

```
# Running and evaluating the model
hist = model.fit(x_train, y_train,
          batch_size=32,
          epochs=10,
          validation_data=(x_test, y_test),
          verbose=2)
```

5. Training the Model

```
model.evaluate()
```

Deep NLP: Recurrent Neural Networks

One-hot encoding

“I expected this movie to be much better”

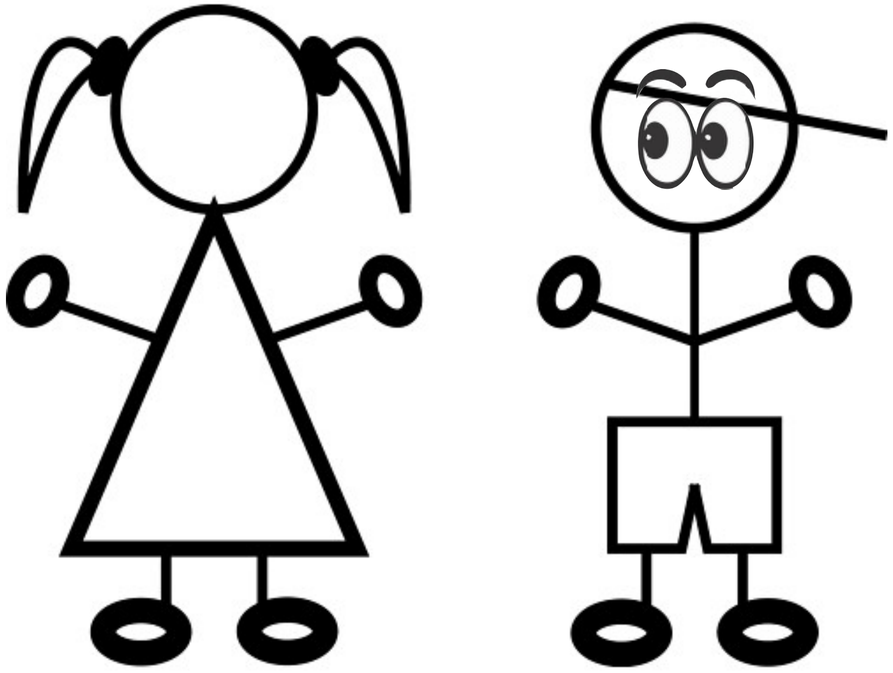
1	0	1	0	0	0	1	0	0	1	0	1	0	0	0	1	0	1	0	1	0	0	1	0
this	I	expected	to	be	movie	is	much	better

“This movie is much better than I expected”

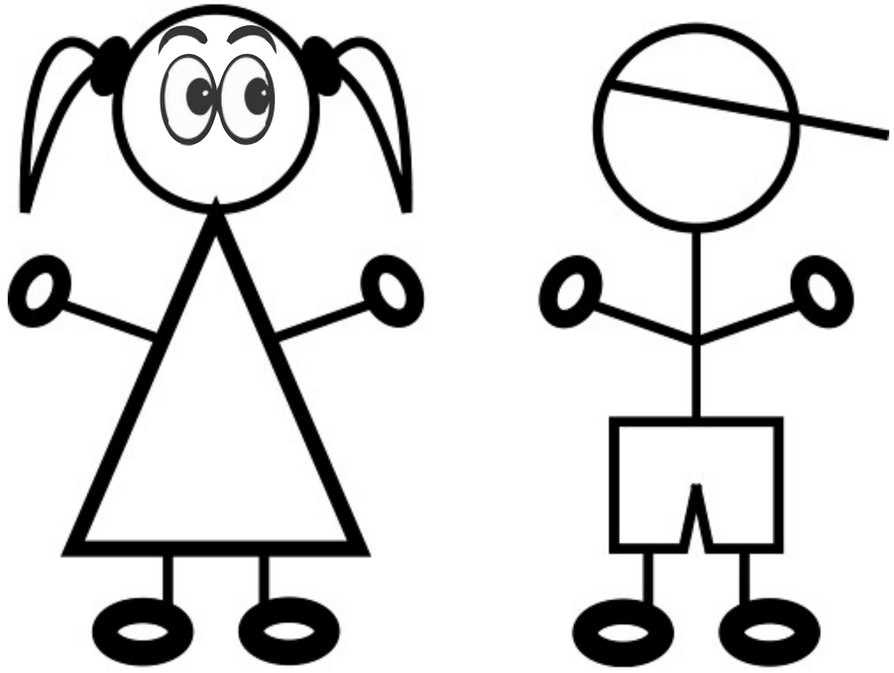
1	0	1	0	0	0	1	0	1	0	0	1	0	0	0	1	0	0	0	1	0	0	1	0
this	I	expected	than	be	movie	much	better

Recurrent Neural Networks

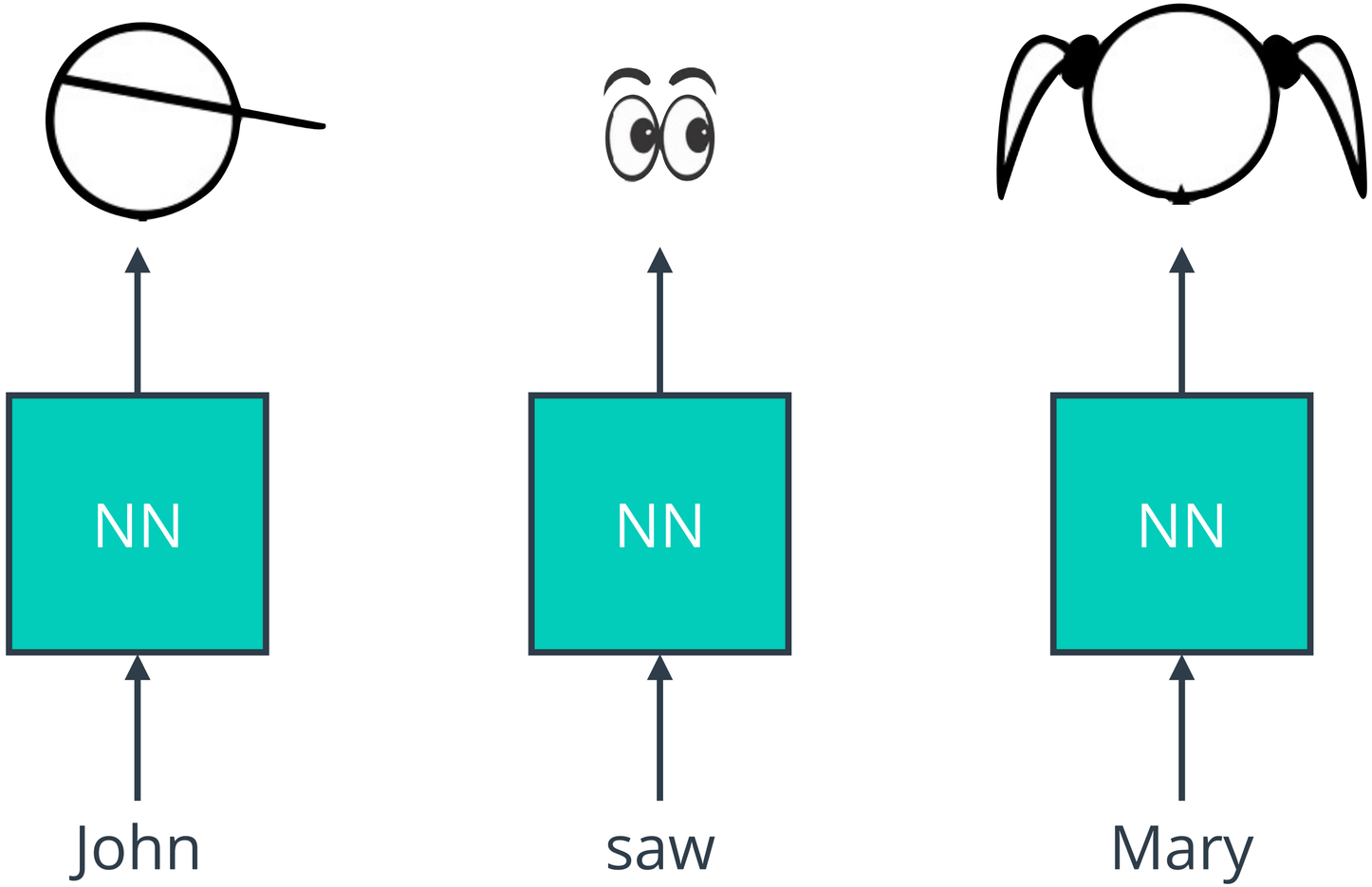
John saw Mary.



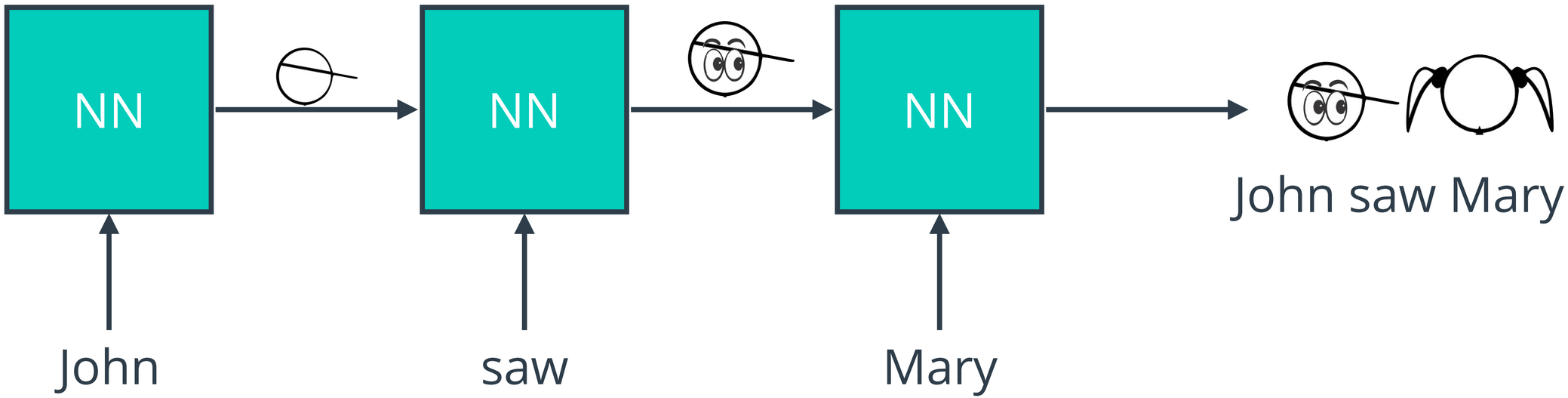
Mary saw John.



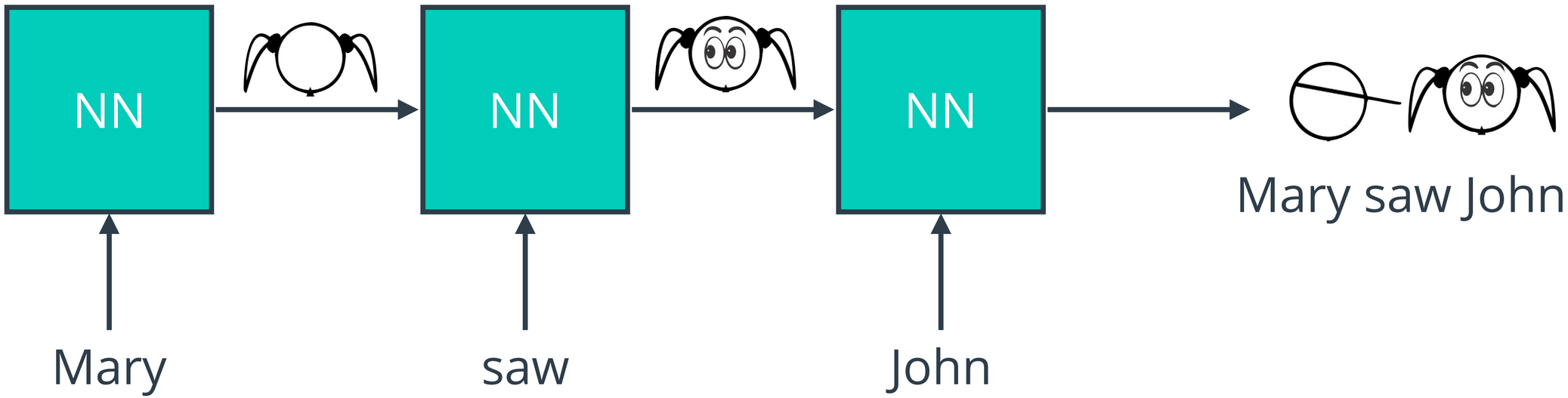
Neural Network



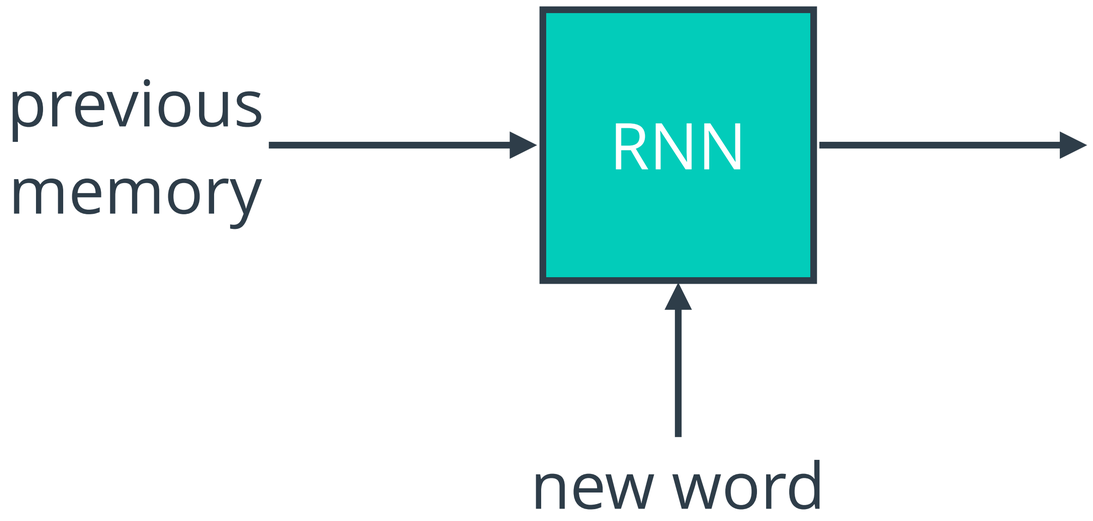
Recurrent Neural Network



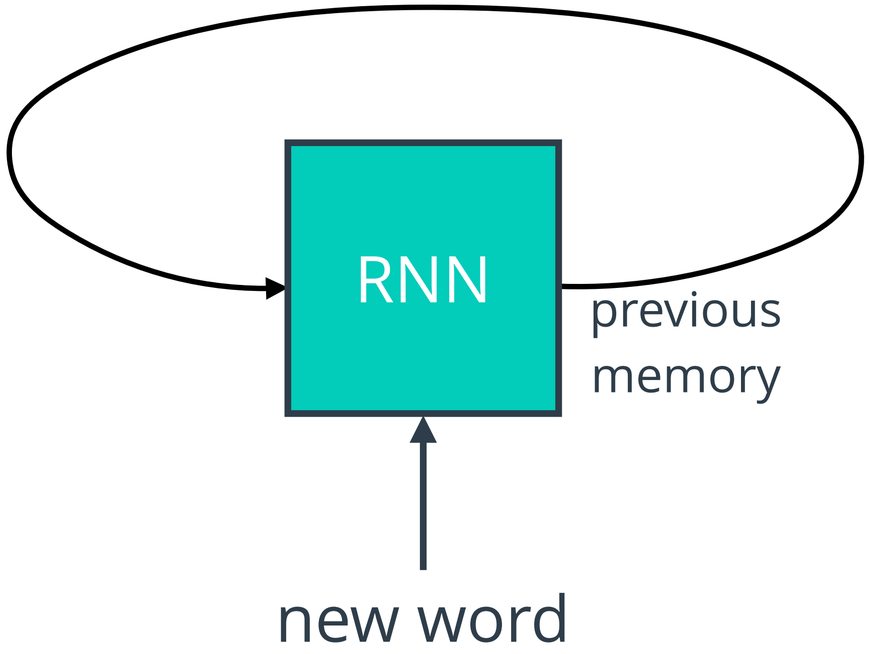
Recurrent Neural Network



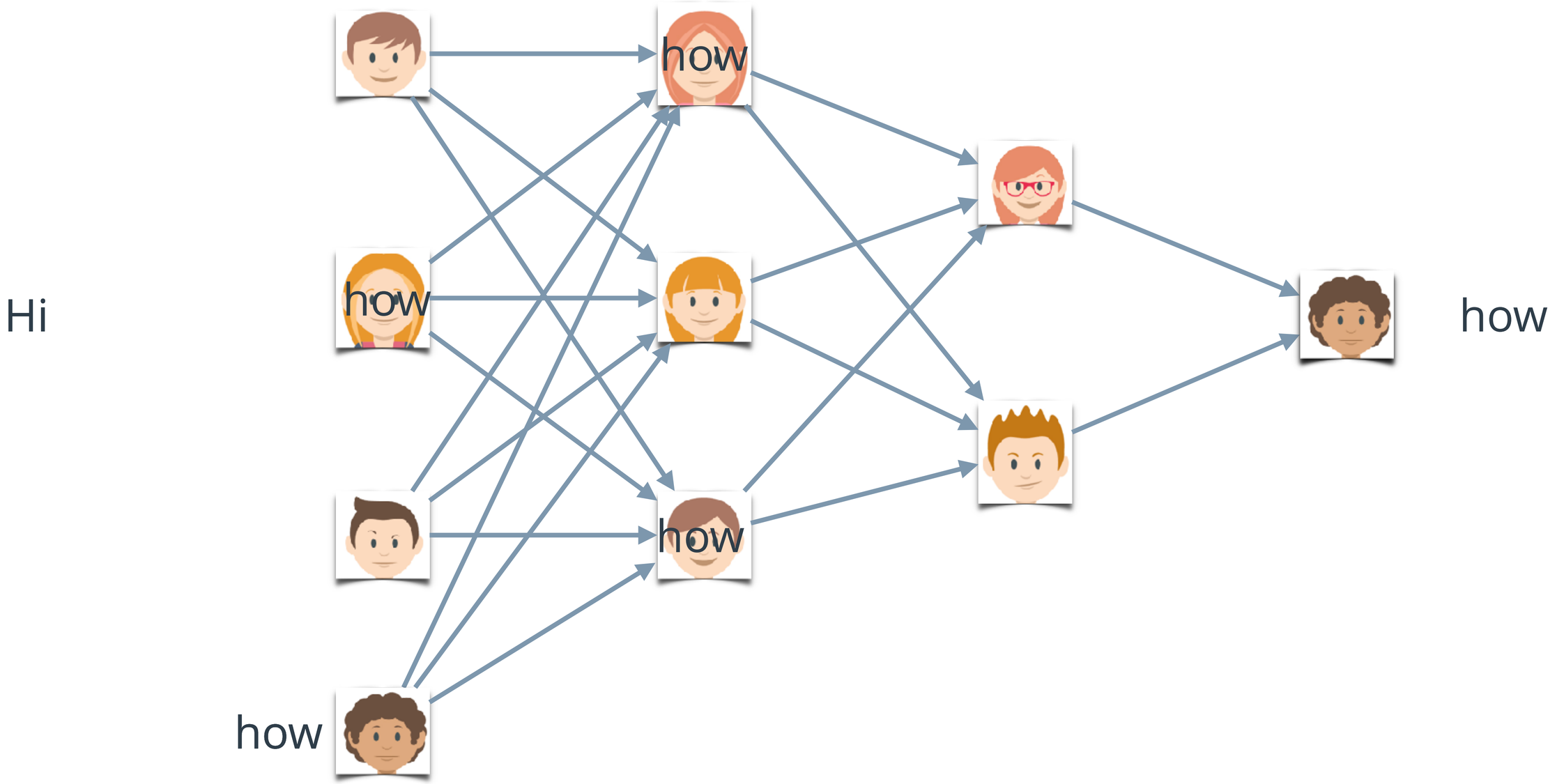
Recurrent Neural Network



Recurrent Neural Network



Recurrent Neural Network



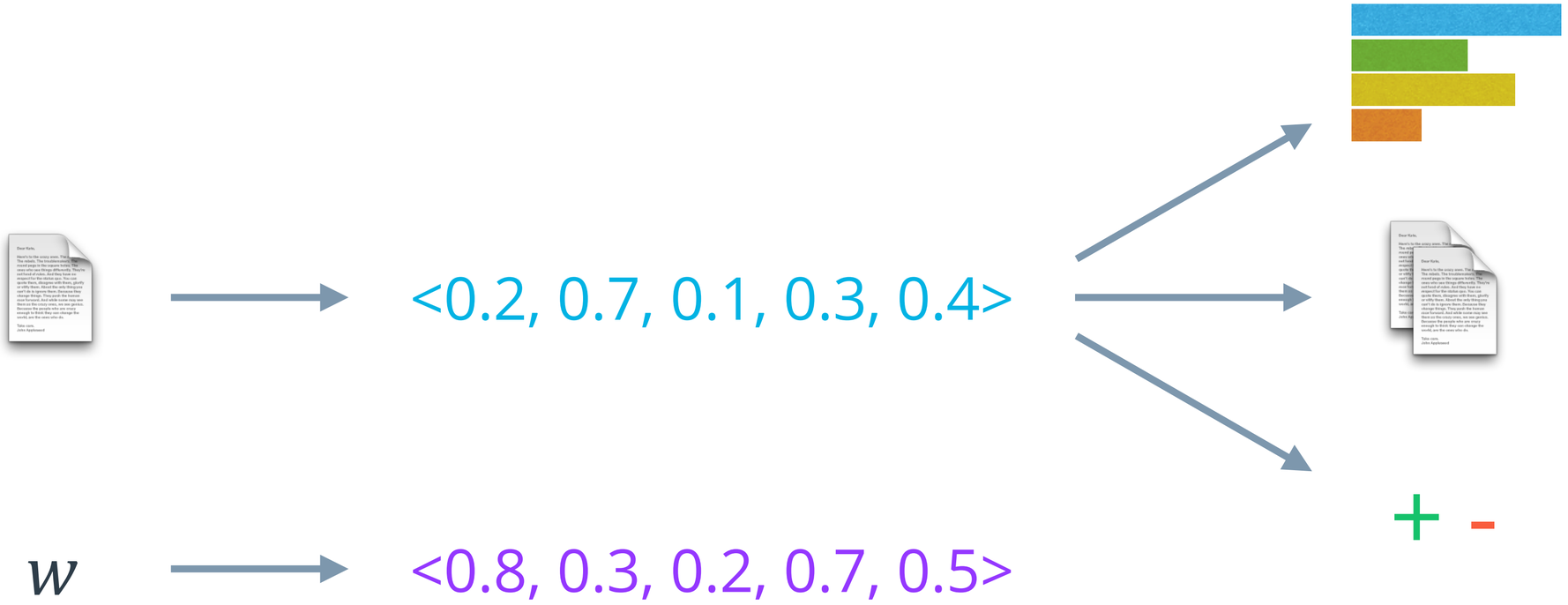
Recurrent Neural Network

Deep NLP: Word Embeddings

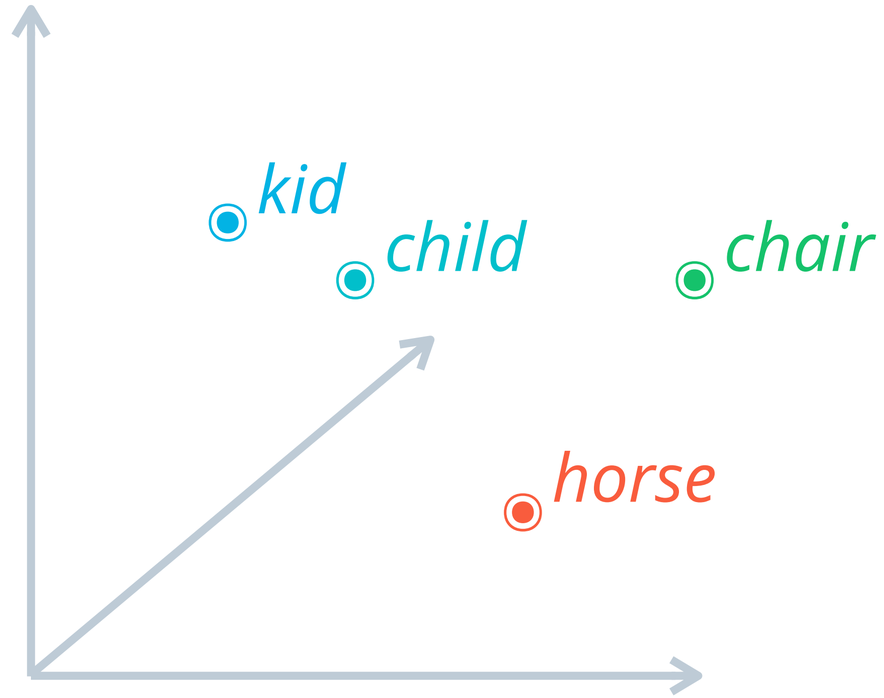
Word Embeddings

- Document vs. Word Representations
- Word2Vec
- GloVe
- Embeddings in Deep Learning
- Visualizing Word Vectors: tSNE

Document vs. Word Representations



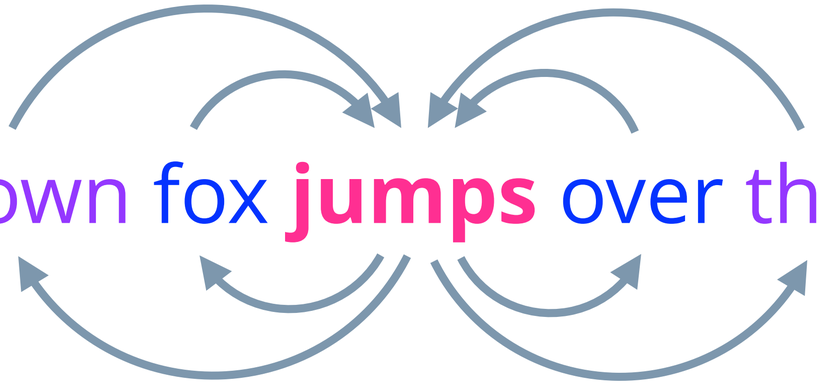
Word Embeddings



Word2Vec

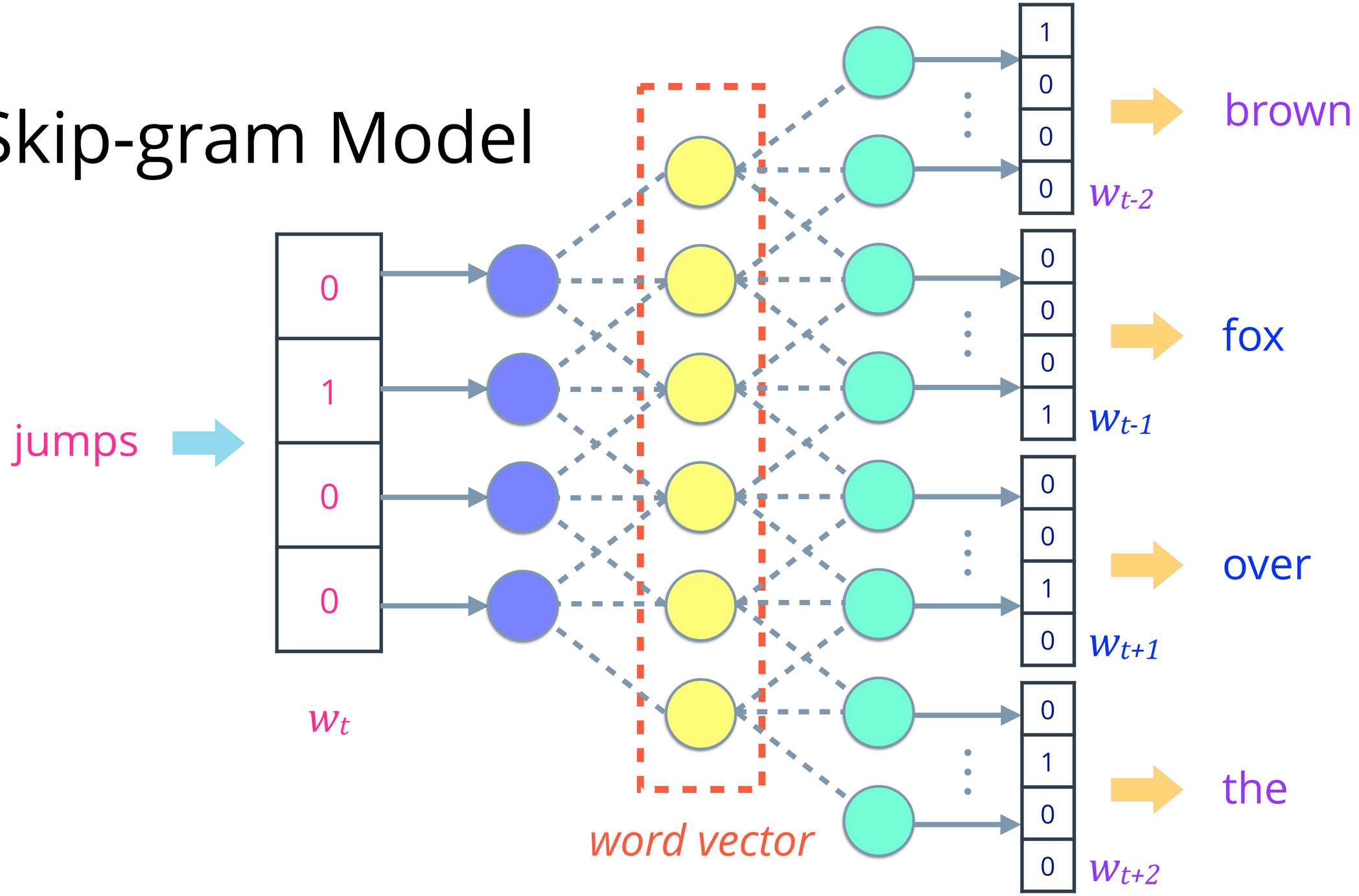
Word2Vec

Continuous Bag of Words (CBoW)



Continuous Skip-gram

Skip-gram Model



Word2Vec: Recap

- Robust, distributed representation.
- Vector size independent of vocabulary.
- Train once, store in lookup table.
- Deep learning ready!

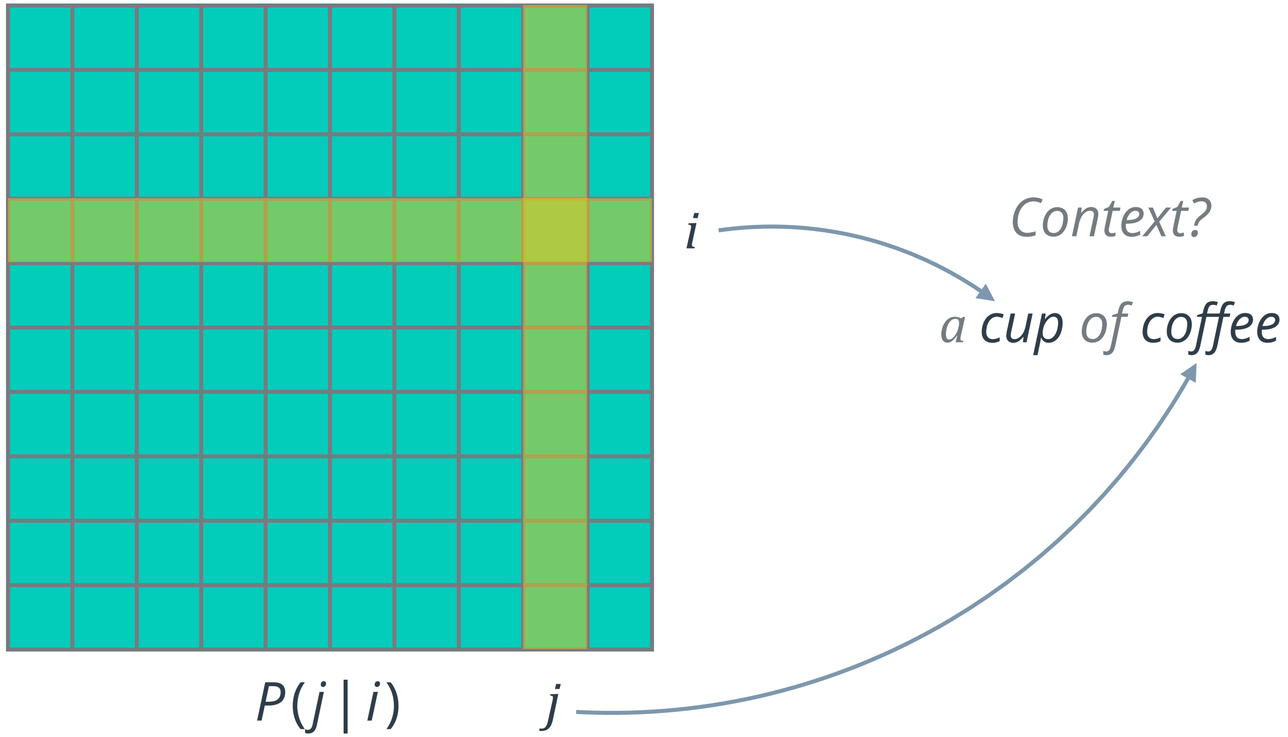
Word2Vec: Further Reading

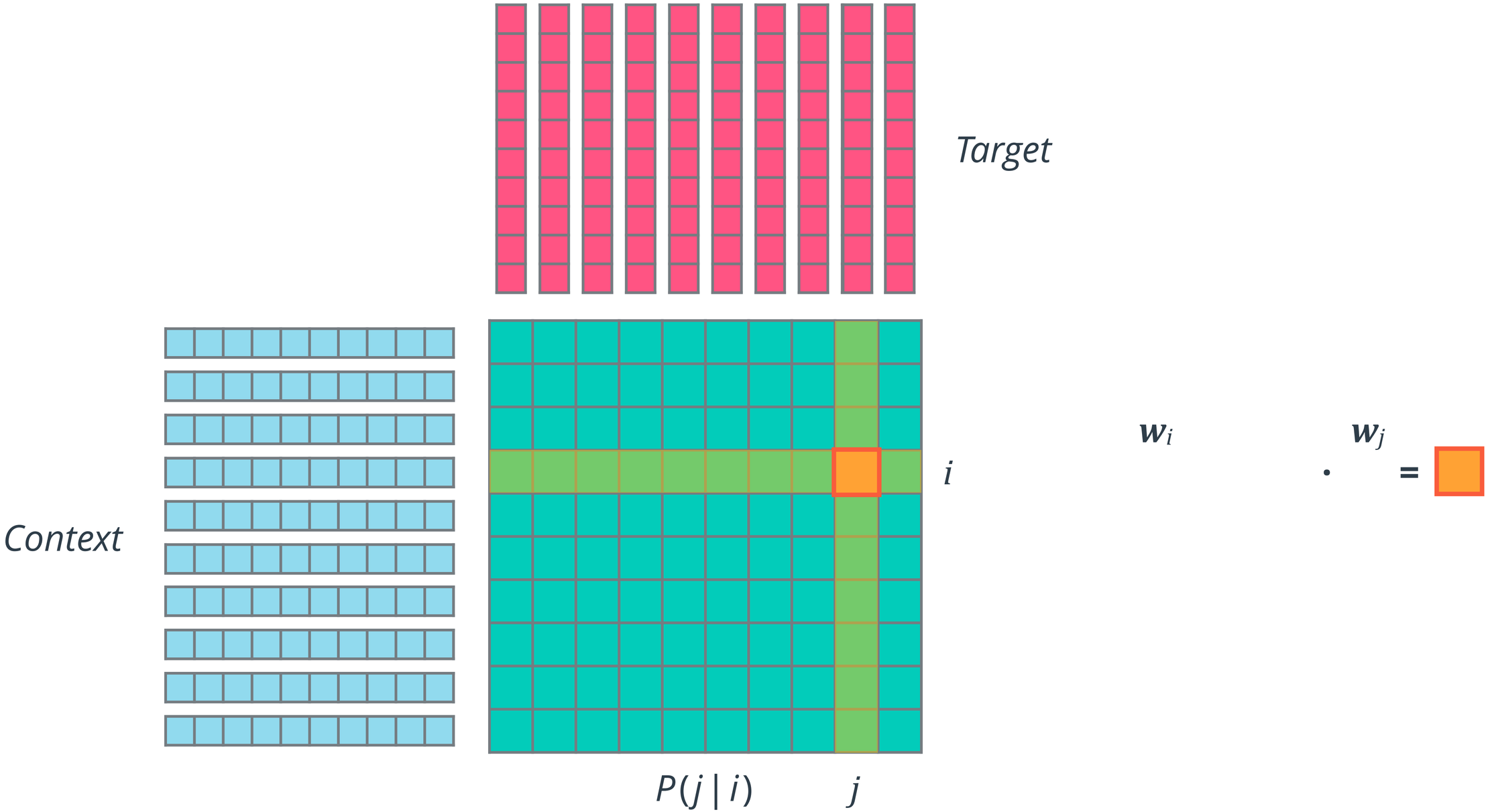
Tomas Mikolov, et al., 2013. Distributed Representation of Words and Phrases and their Compositionality, In *Advances of Neural Information Processing Systems (NIPS)*, pp. 3111-3119.

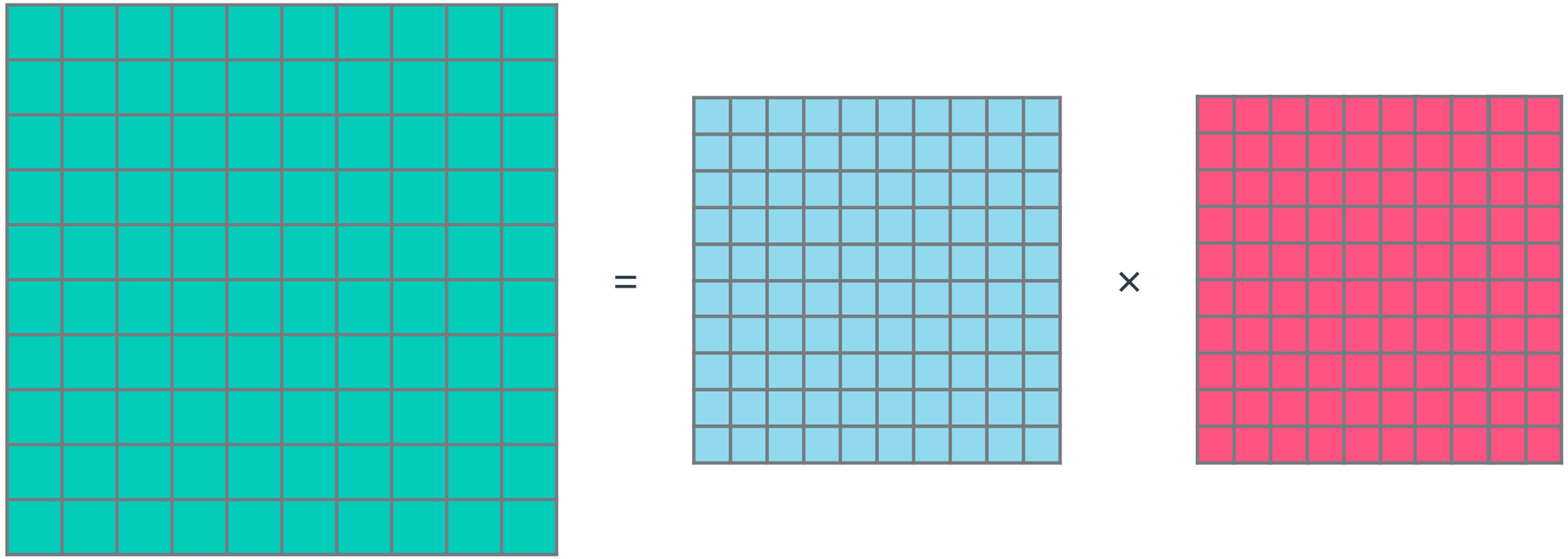
Adrian Colyer, 2016. The amazing power of word vectors.

GloVe

Global Vectors for Word Representation


i
Context?
a cup of coffee
P(j | i)
j


Target
Context
i
w_i
w_j
$P(j \mid i)$
j



$P(j \mid i)$

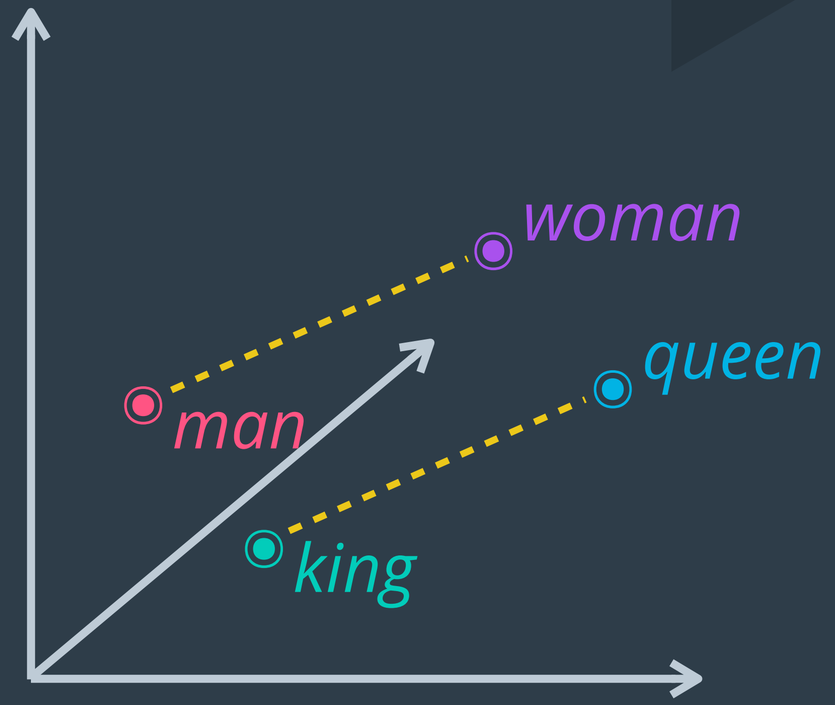
Co-occurrence Probabilities

	solid	*water*
 *ice*	$P(solid \mid ice)$	$P(water \mid ice)$
 *steam*	$P(solid \mid steam)$	$P(water \mid steam)$

$$\frac{P(solid \mid ice)}{P(solid \mid steam)} \gg 1 \qquad \frac{P(water \mid ice)}{P(water \mid steam)} \approx 1$$

Embeddings in Deep Learning

woman - man + king = queen



Distributional Hypothesis

“Would you like a cup of _____?”

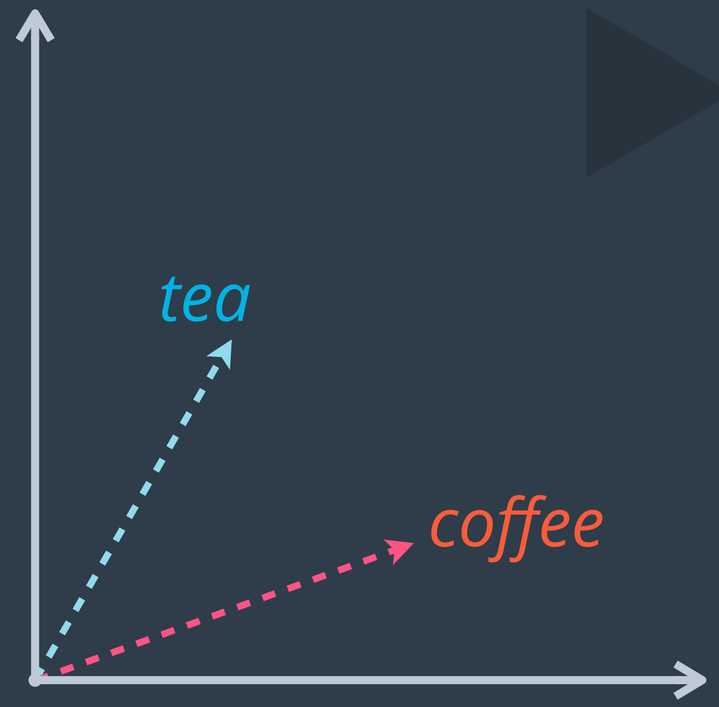
“I like my _____ black.”

“I need my morning _____ before I can do anything!”

“Would you like a cup of _____?”

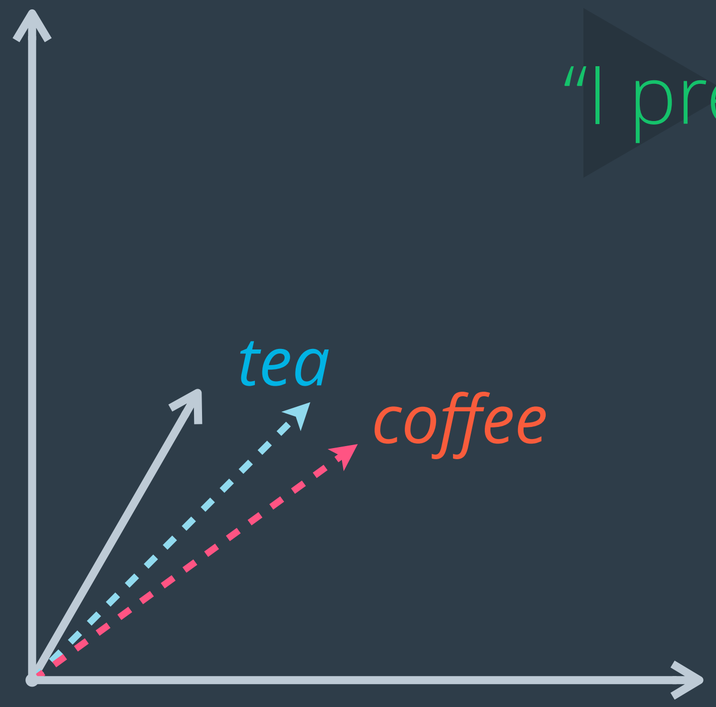
“I like my _____ black.”

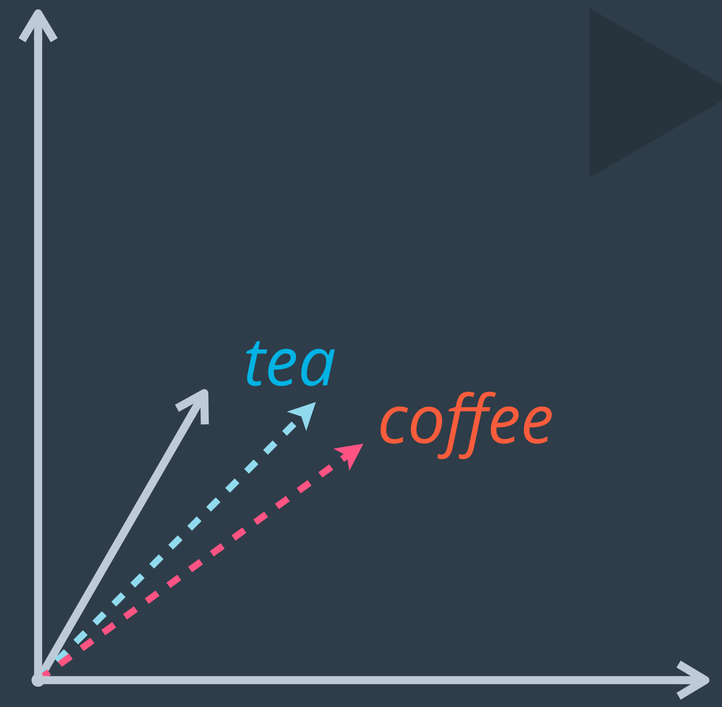
“I need my morning _____ before I can do anything!”

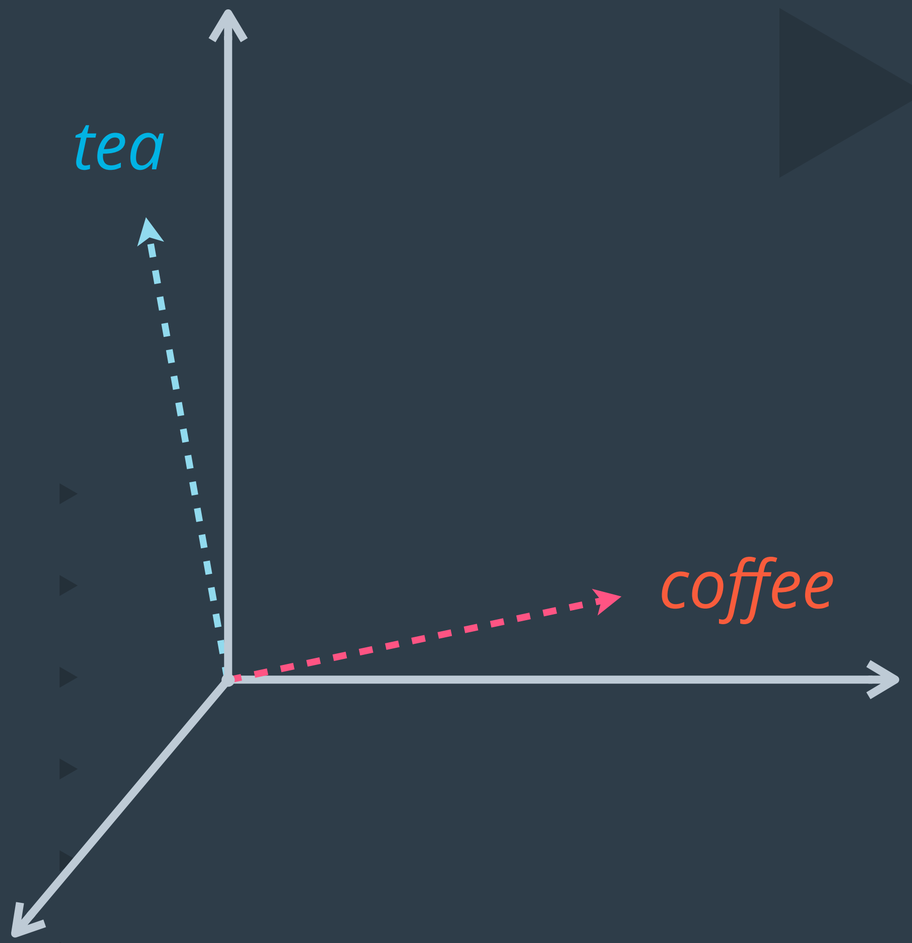

tea
coffee

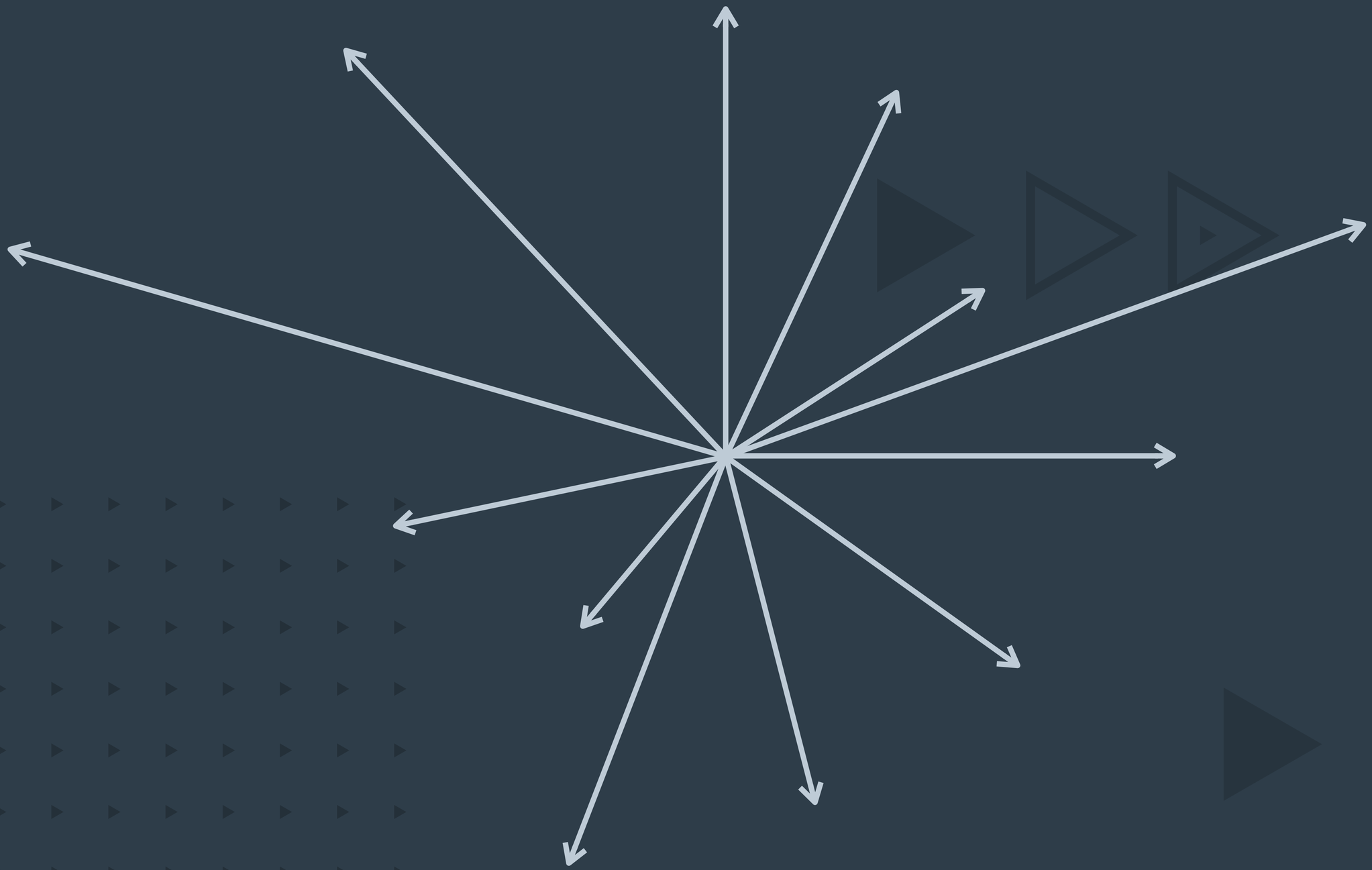
“ Coffee grounds are great for composting!”

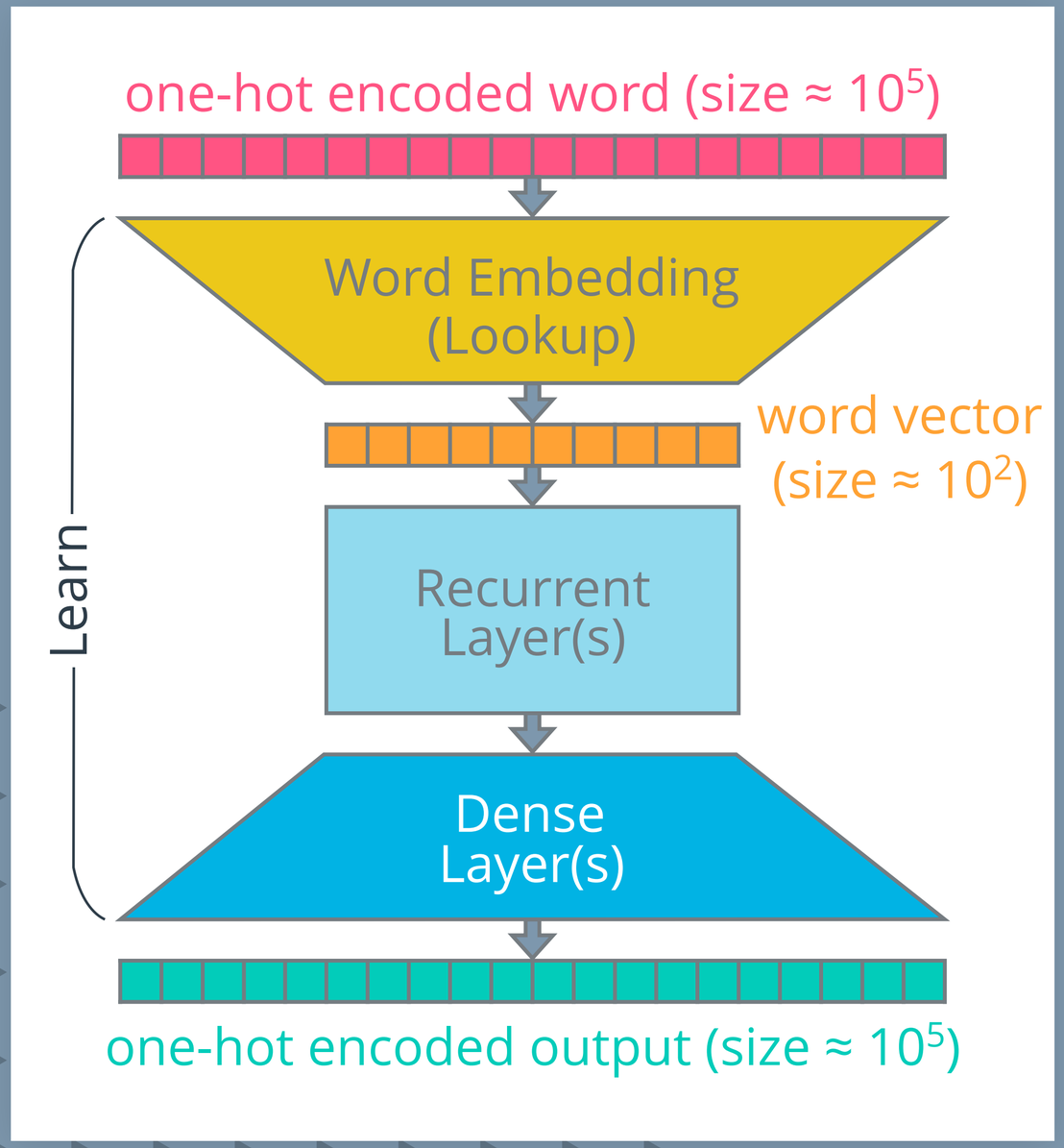
“I prefer loose leaf _tea_.”

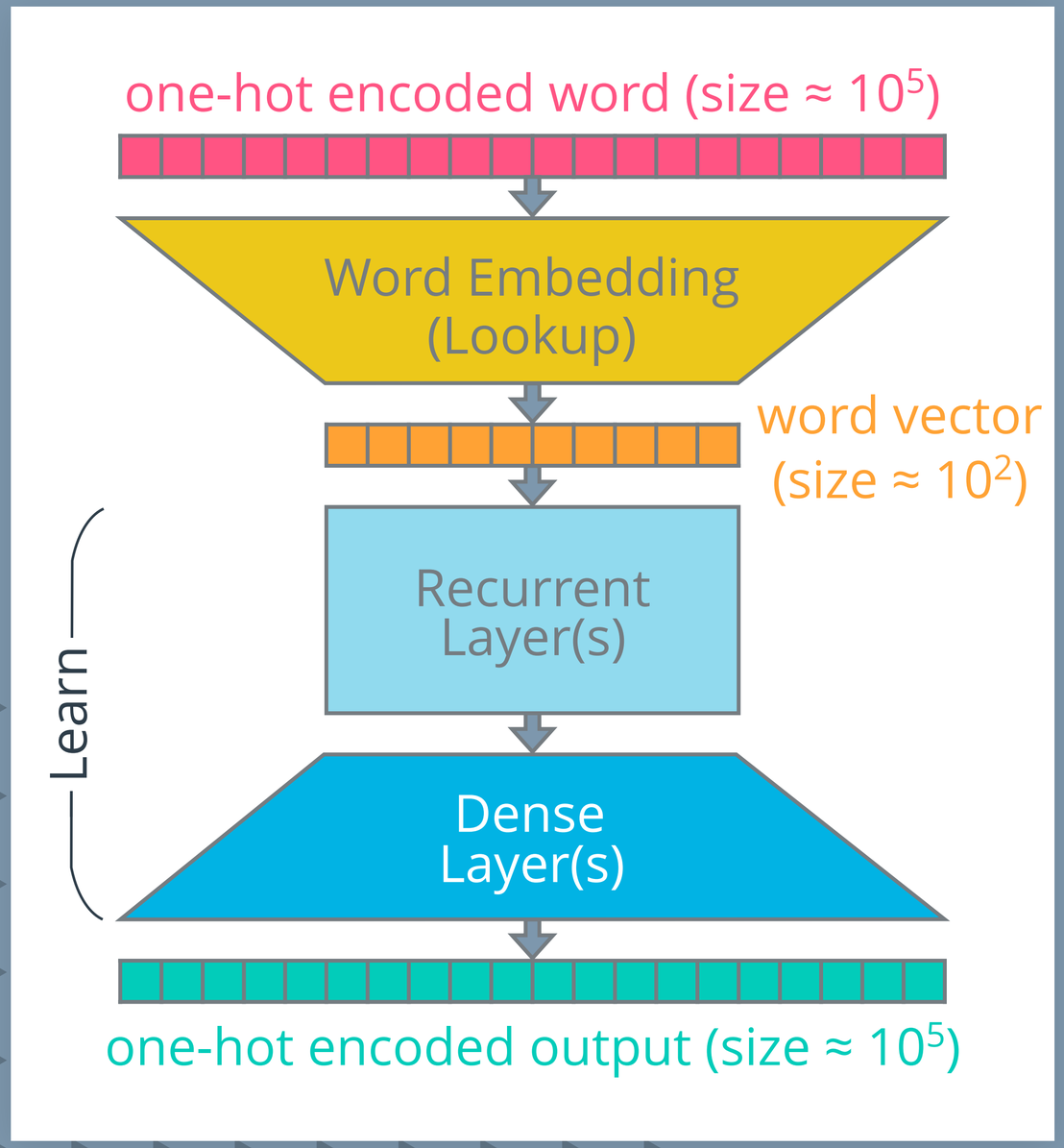


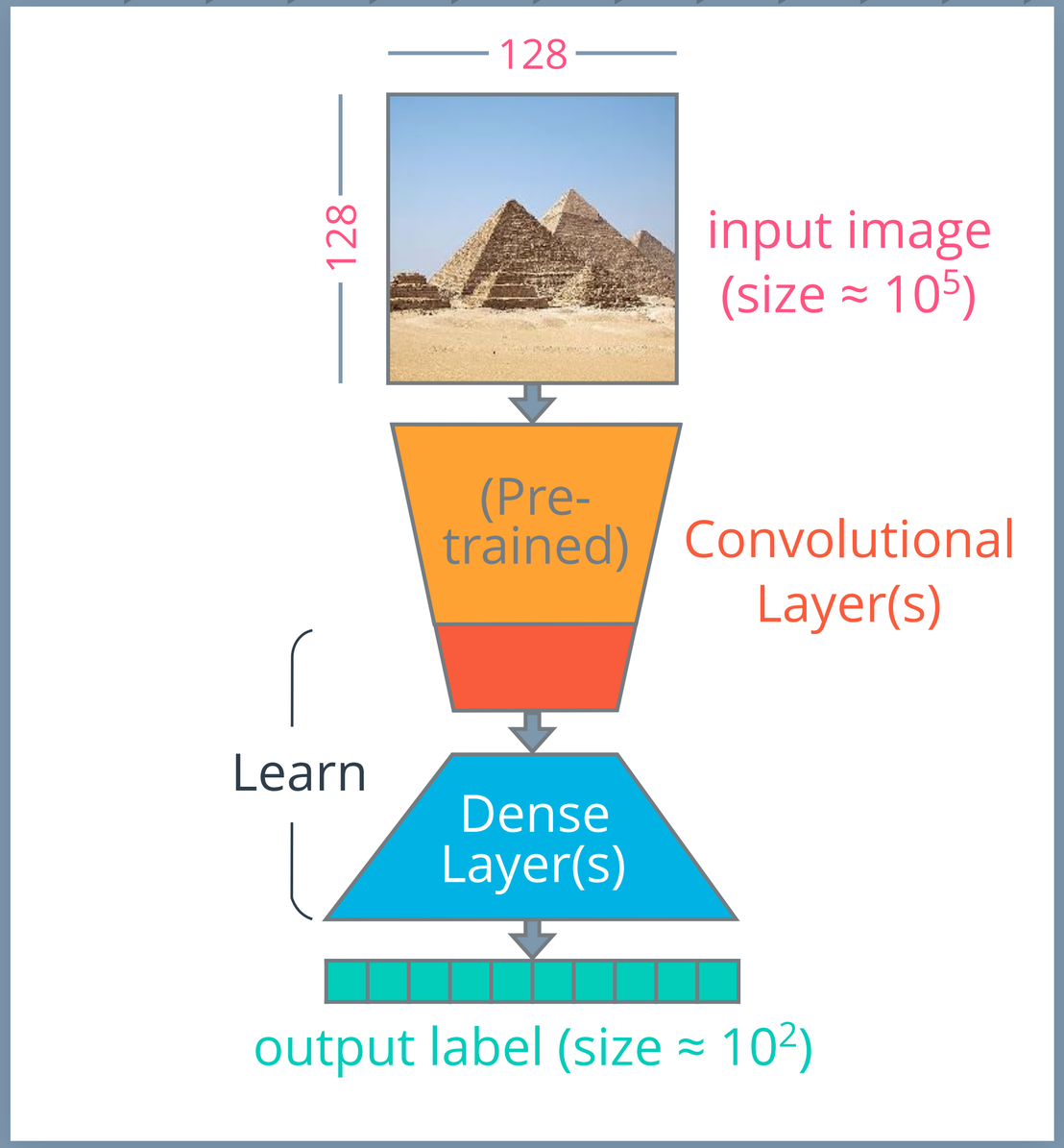

tea
coffee


tea
coffee




one-hot encoded word (size ≈ 10^5)
Word Embedding
(Lookup)
word vector
(size ≈ 10^2)
Recurrent
Layer(s)
Dense
Layer(s)
Learn
one-hot encoded output (size ≈ 10^5)


one-hot encoded word (size ≈ 10^5)
Word Embedding (Lookup)
word vector (size ≈ 10^2)
Recurrent Layer(s)
Learn
Dense Layer(s)
one-hot encoded output (size ≈ 10^5)

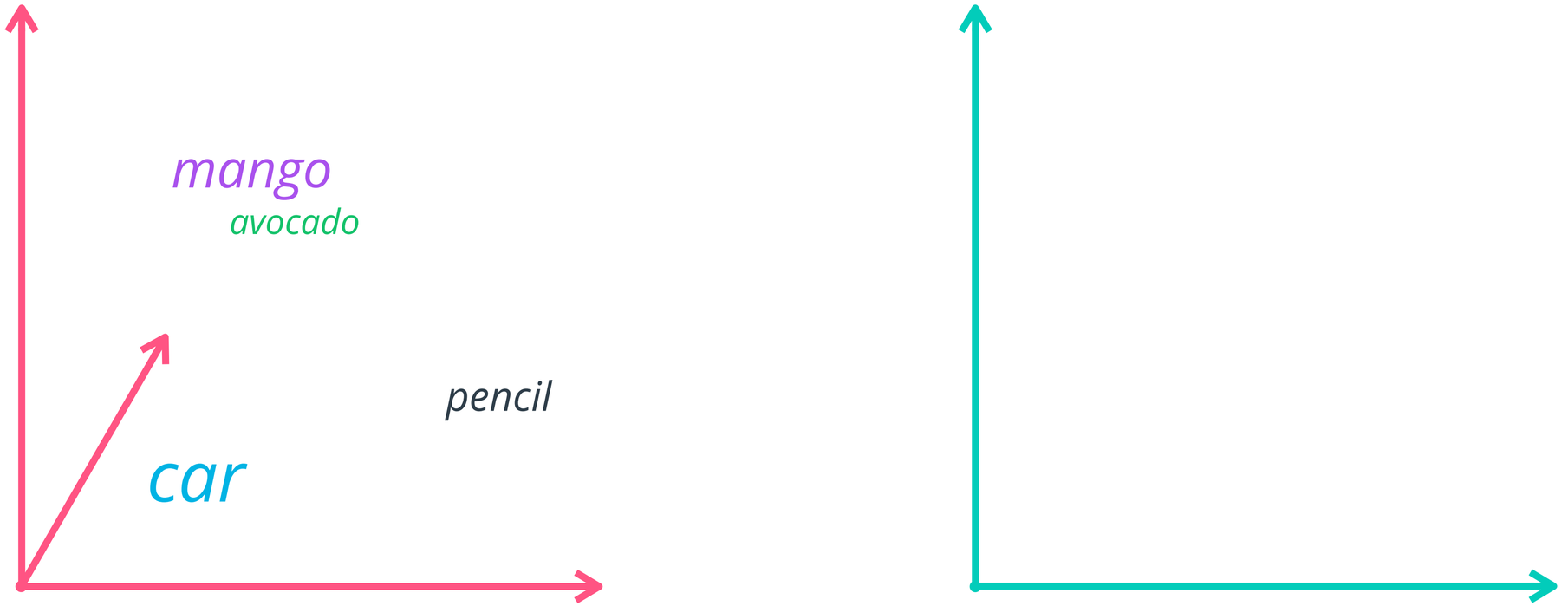

128
128
input image (size ≈ 10^5)
(Pre-trained)
Convolutional Layer(s)
Learn
Dense Layer(s)
output label (size ≈ 10^2)

Visualizing Word Vectors: t-SNE

t-Distributed Stochastic Neighbor Embedding

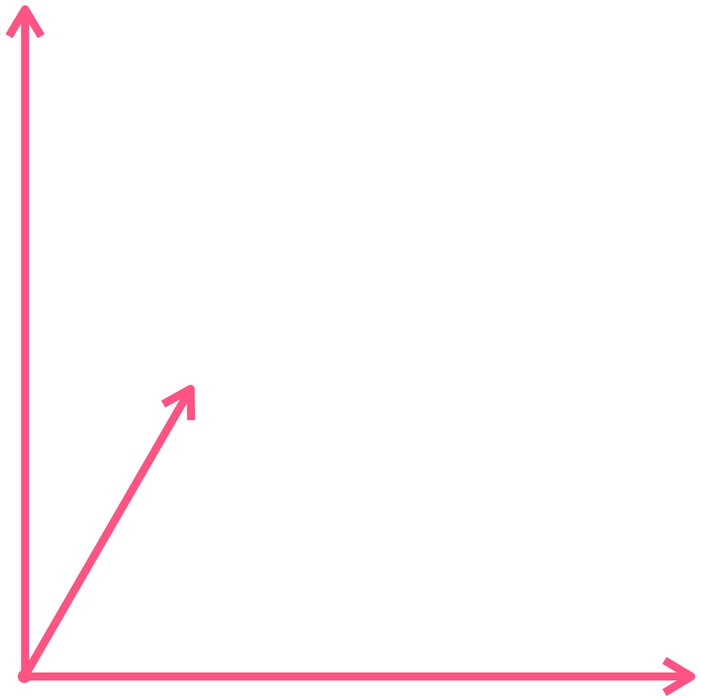
t-SNE

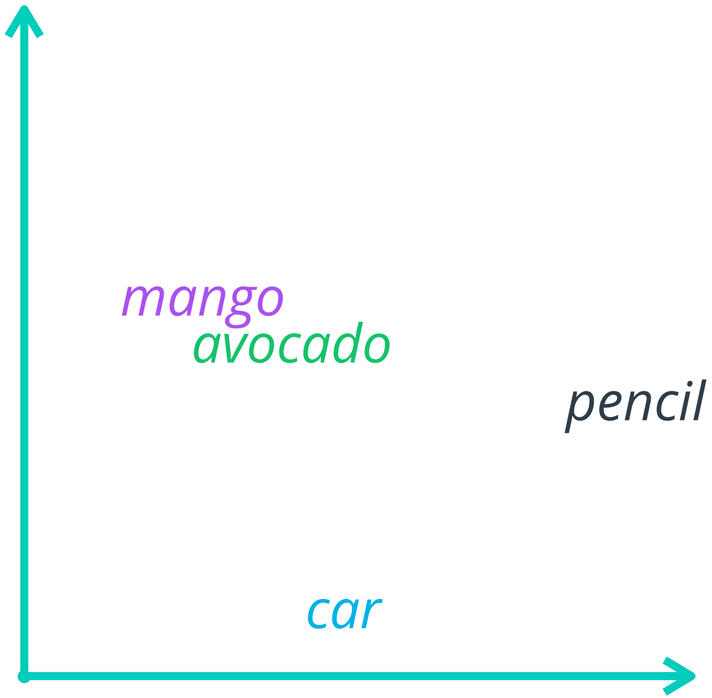
t-Distributed Stochastic Neighbor Embedding

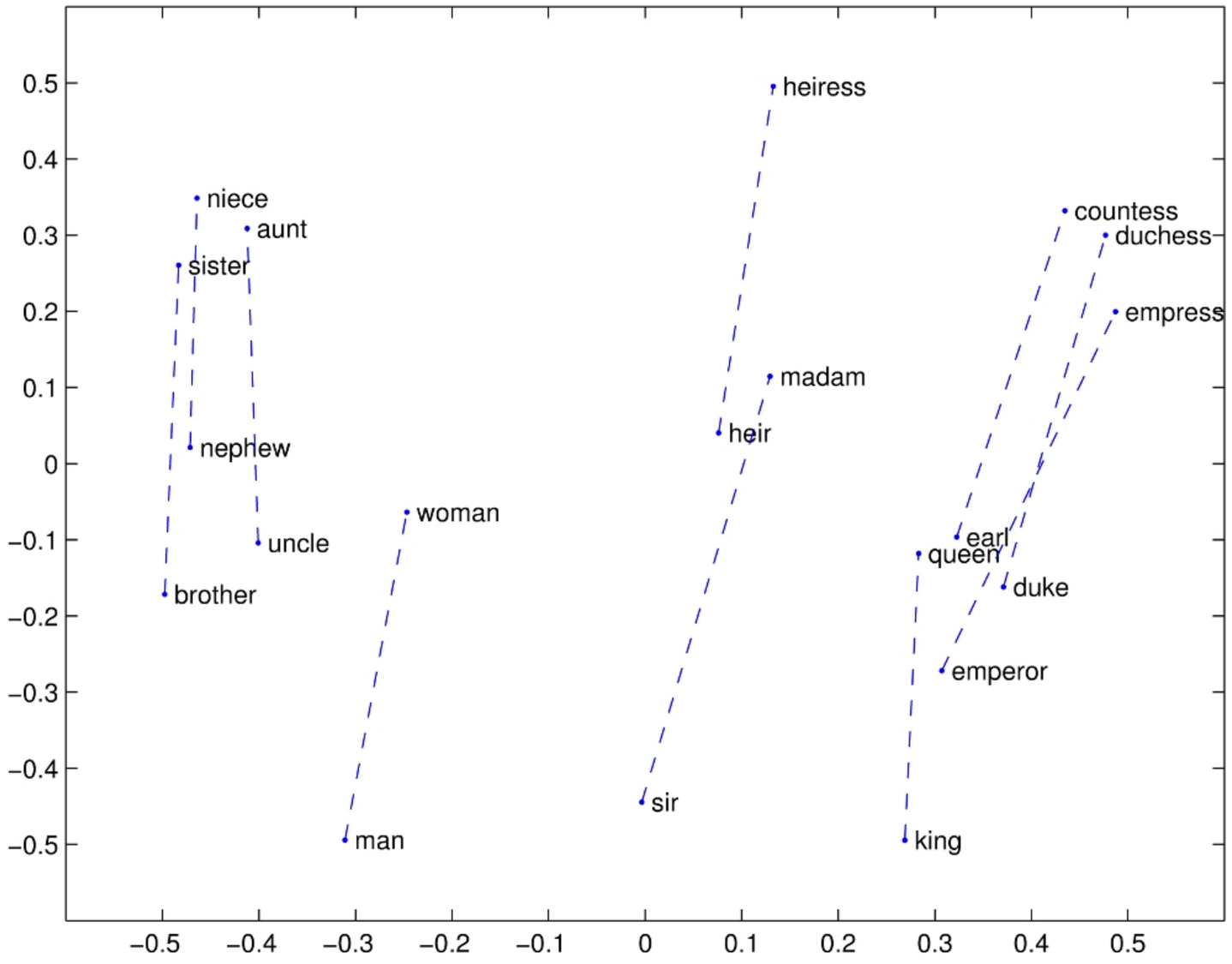


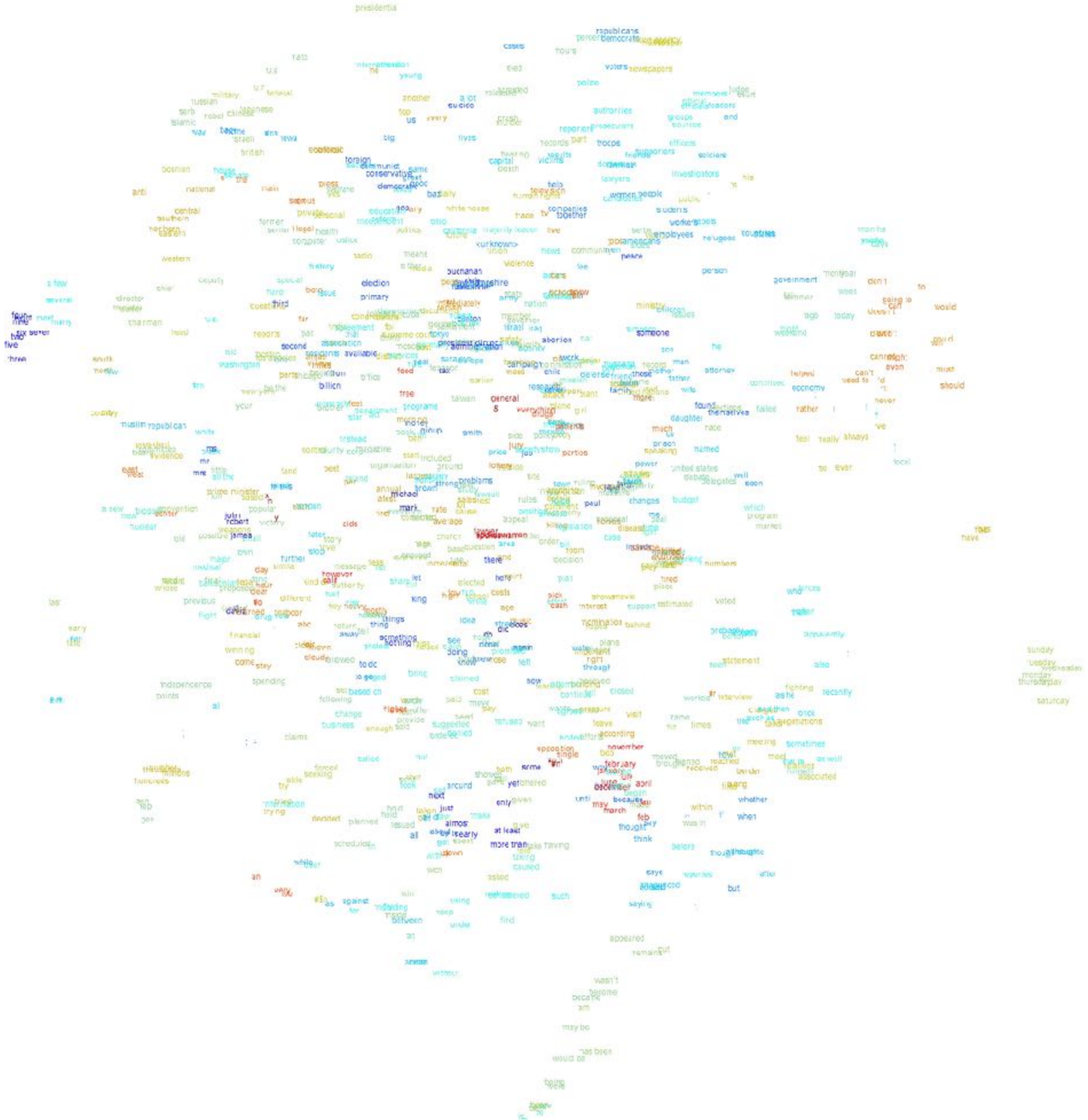
t-SNE

t-Distributed Stochastic Neighbor Embedding






0.5
0.4
0.3
0.2
0.1
0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.5
−0.4
−0.3
−0.2
−0.1
0
0.1
0.2
0.3
0.4
0.5
niece
aunt
sister
nephew
uncle
brother
woman
man
heiress
madam
heir
sir
countess
duchess
empress
earl
queen
duke
emperor
king




presidential
republicans
percent
democrats
cases
hours
nato
u.s
international
american
no
young
died
voters
newspapers
police
u.n
military
federal
russian
another
a lot
suicide
released
arrested
serb
rebel
chinese
japanese
top
us
every
authorities
islamic
crash
murder
way
back
home
#na
israeli
iowa
big
lives
reporters
prosecutors
records
part
troops
british
economic
political
foreign
security
communist
hearing
results
capital
victims
friends
supporters
bosnian
house
senate
a
the
conservative
same
great
death
families
lawyers
press
main
national
anti
democratic
good
texas
welfare
yes
bad
daily
help
television
human rights
women
people
candidates
serious
central
southern
education
gop
any
white house
private
personal
reform
independent
trade
tv
companies
together
former
ohio
northern
eastern
senior
illegal
health
politics
california
future
majority leader
live
serbs
jobs
americans
sides
computer
justice
<unknown>
union
news
community
men
peace
western
radio
means
history
either
media
violence
buchanan
lee
deputy
special
election
dole
chief
hard
born
issue
primary
director
third
questions
state
army
nation
ministry
minister
leader
immediately
remain
development
documents
congressional
cuba
russia
clinton
member
governor
chairman
u.s.
far
india
agreement
fbi
parliament
israel
iraq
simpson
head
reports
trial
someone

ohio
politics
california
majority leader
live
workers
rebels
months
serbs
employees
refugees
future
<unknown>
jobs
americans
sides
means
union
news
peace
violence
lee
person
buchanan
government
don't
to
week
summer
going to
can
would
doesn't
ministry
children
issues
ago
today
nation
army
member
governor
clinton
israel
iraq
parliament
fbi
tokyo
supreme court
abortion
car
someone
weekend
could
will
administration
agency
son
he
past
europe
campaign
commission
dollar
husband
record
man
cannot
might
even
food
tax
mass
child
those
mother
attorney
helped
can't
must
defense
friend
father
earlier
mission
team
family
used to
continued
economy
'd
should
'll
research
airport
plant
wife
free
taiwan
general
more
never
programs
found
elections
themselves
failed
rather
&
everything
plane
girl
drugs
daughter
morning
money
patients
group
smith
much
race
book
side
mexico
feel
really
always
ban
co
prison
jury
price
show
job
parties
speaking
named
start
included
power
local
ground
lottery
outside
united states
so
ever
site
well
problems
debate
strong
brown
delegates
soon
town
ruling
party
michael
lawsuit
measure
rules
sales
rest
changes
budget
paul
which
mark
rate
lot
cause
comment
average
appeal
process
me
deal
program
proposal
homes
killing
legislation
disease
market
time
fight
church
lawyer
spokesman
letter
case
have
order
area
question
base
bill
room
include
damage

return
thing
things
idea
music
does
did
nomination
behind
probably
apparently
away
tell
do
hopes
something
nothing
see
plans
better
protect
raised
claim
done
water
important
doing
promised
right
statement
allowed
to do
knew
rose
left
through
seen
also
know
bring
to go
claimed
believed
now
fighting
set
based on
cost
closed
recently
following
paid
continue
fell
interview
move
worked
as he
pay
change
offer
visit
once
provide
need
came
business
suggested
refused
want
leave
like
negotiations
enough
sold
times
hit
denied
according
ordered
ended
efforts
meeting
sometimes
opposition
november
single
bob
moved
meet
how
as well
called
put
some
february
received
border
reached
relations
himself
both
july
associated
showed
april
along
around
yet
offered
took
next
began
until
because
whether
only
given
may
march
within
hold
just
held
make
if
when
decided
planned
feb
say
issued
almost
thought
was in
all
nearly
at least
give
think
scheduled
spent
more than
having
before
down
told
taking
with
caused
won
says
reported
after
asked
but
win
considered
such
against
using
saying
including
for
keep
inside
between
under
find
at
appeared
out


Highlights of
Current Research

Lab: Sentiment Analysis using RNNs

Classify Movie Reviews

Notebook: IMDB_Keras_RNN.ipynb

Workshop Summary

Classic NLP

- Text Processing: Stop word removal, stemming, lemmatization
- Feature Extraction: Bag-of-Words, TF-IDF
- Topic Modeling: Latent Dirichlet Allocation
- **Lab: Topic modeling using LDA**

Deep NLP

- Neural Networks
- Recurrent Neural Networks
- Word Embeddings: Word2Vec, GloVe, tSNE
- **Lab: Sentiment Analysis using RNNs**

Additional Resources

- Recurrent Neural Networks (RNNs)
- Long Short-Term Memory Networks (LSTMs)
- Visual Question-Answering

Adarsh Nair

Luis Serrano

Arpan Chakraborty

udacity.com/school-of-ai