

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:

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

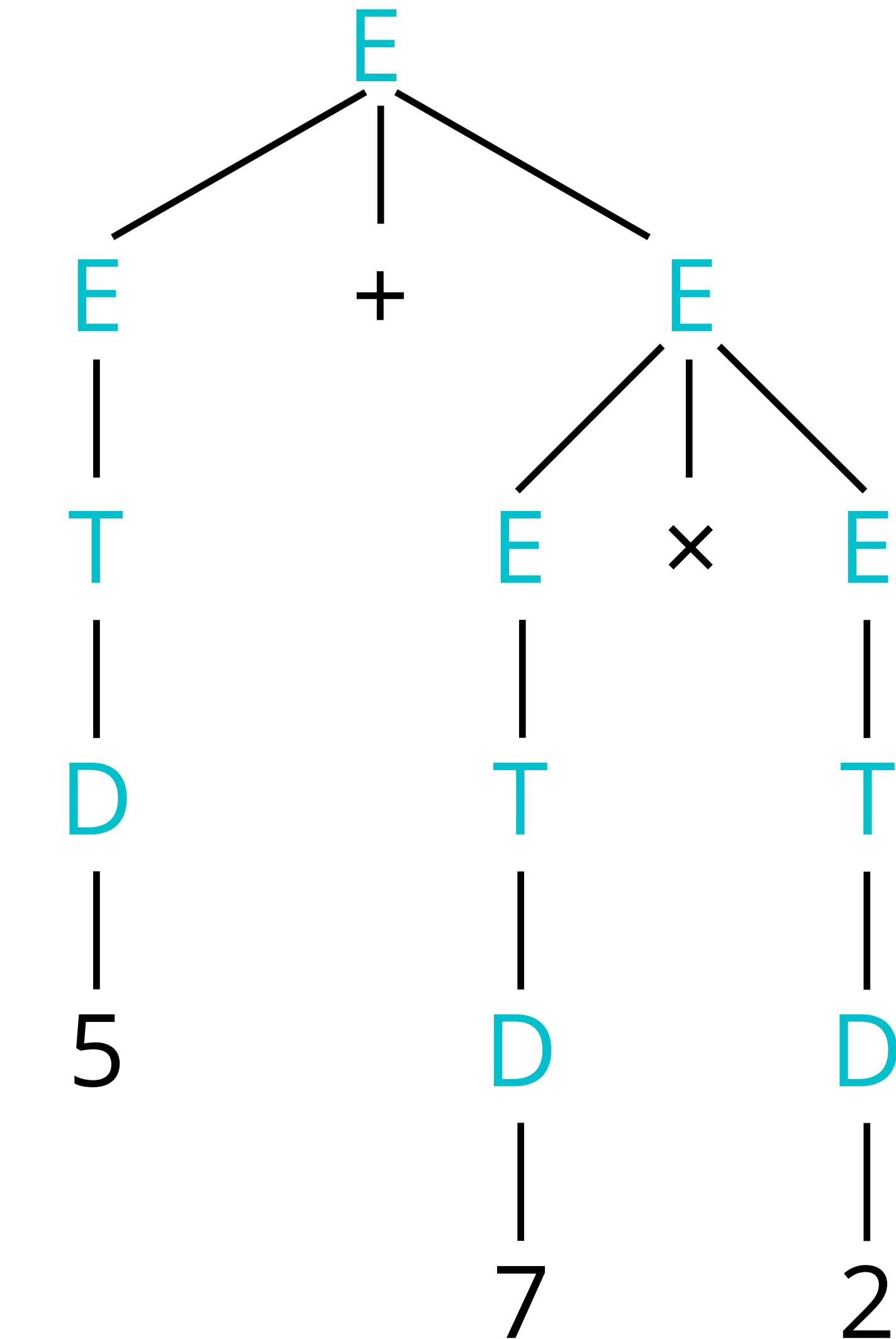
Structured Languages

- SQL:

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

Grammar

- Arithmetic (single digit):

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


Grammar

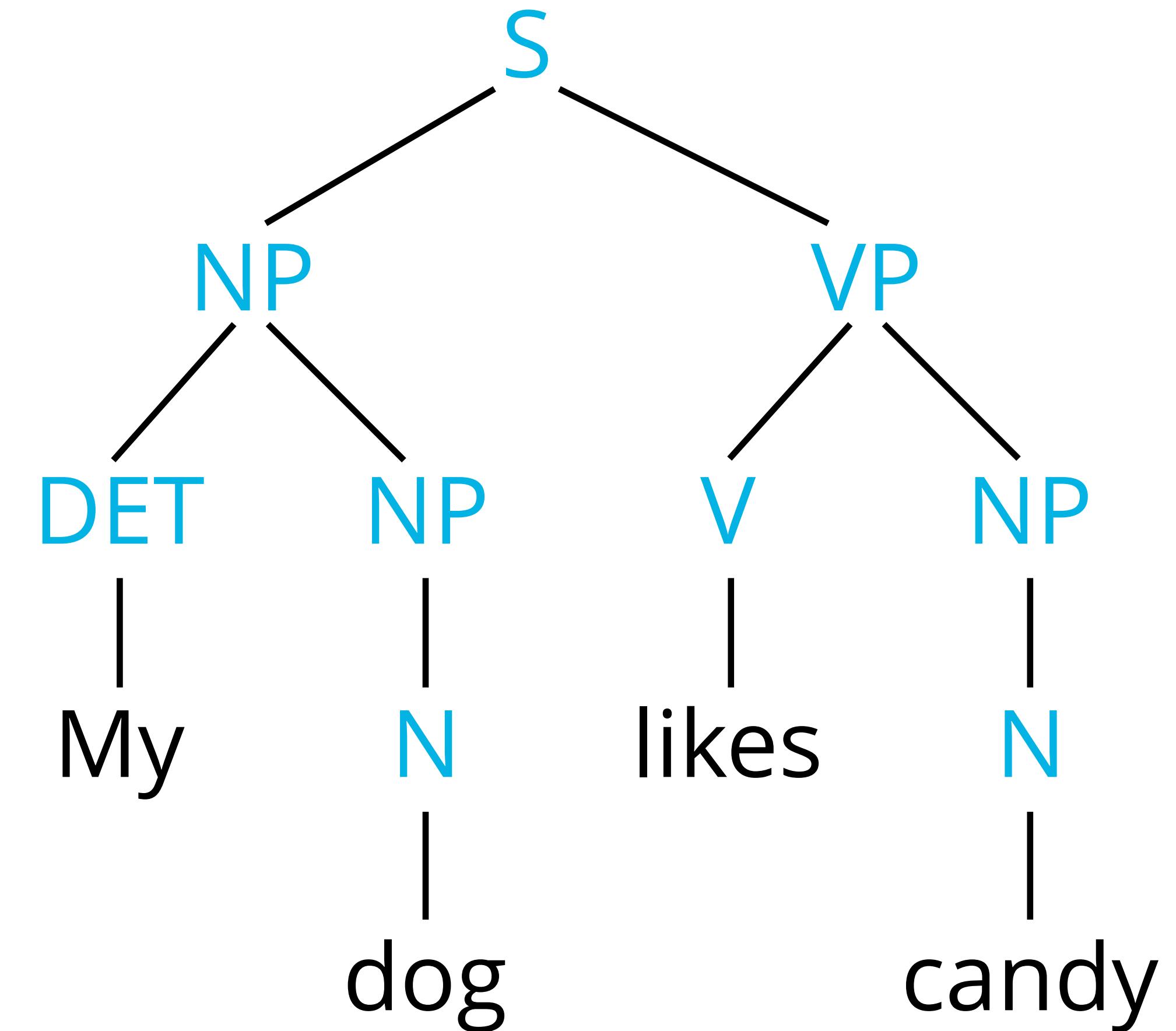
- English sentences (limited):

$S \rightarrow NP\ VP$

$NP \rightarrow N \mid DET\ NP \mid ADJ\ NP$

$VP \rightarrow V \mid V\ NP$

...



**“Because he was so small, Stuart was often hard to
find around the house.”**

verb

noun

– *Stuart Little*, E.B. White

Unstructured Text

the quick brown fox jumps over the lazy dog

Unstructured Text

Unstructured Text

jumps

the

fox

brown

over

dog

+ quick

the

lazy -

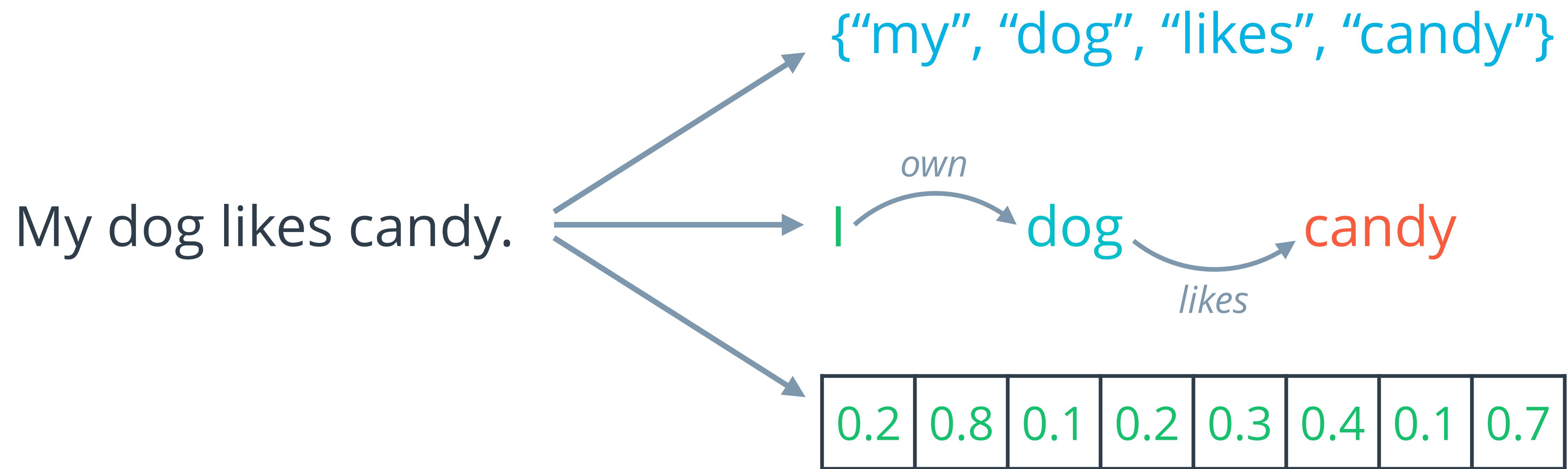
Applications



health
science
politics

what time is it?
¿que hora es?

Challenges: Representation



Challenges: Temporal Sequence

I want to buy a gallon of **milk**

water

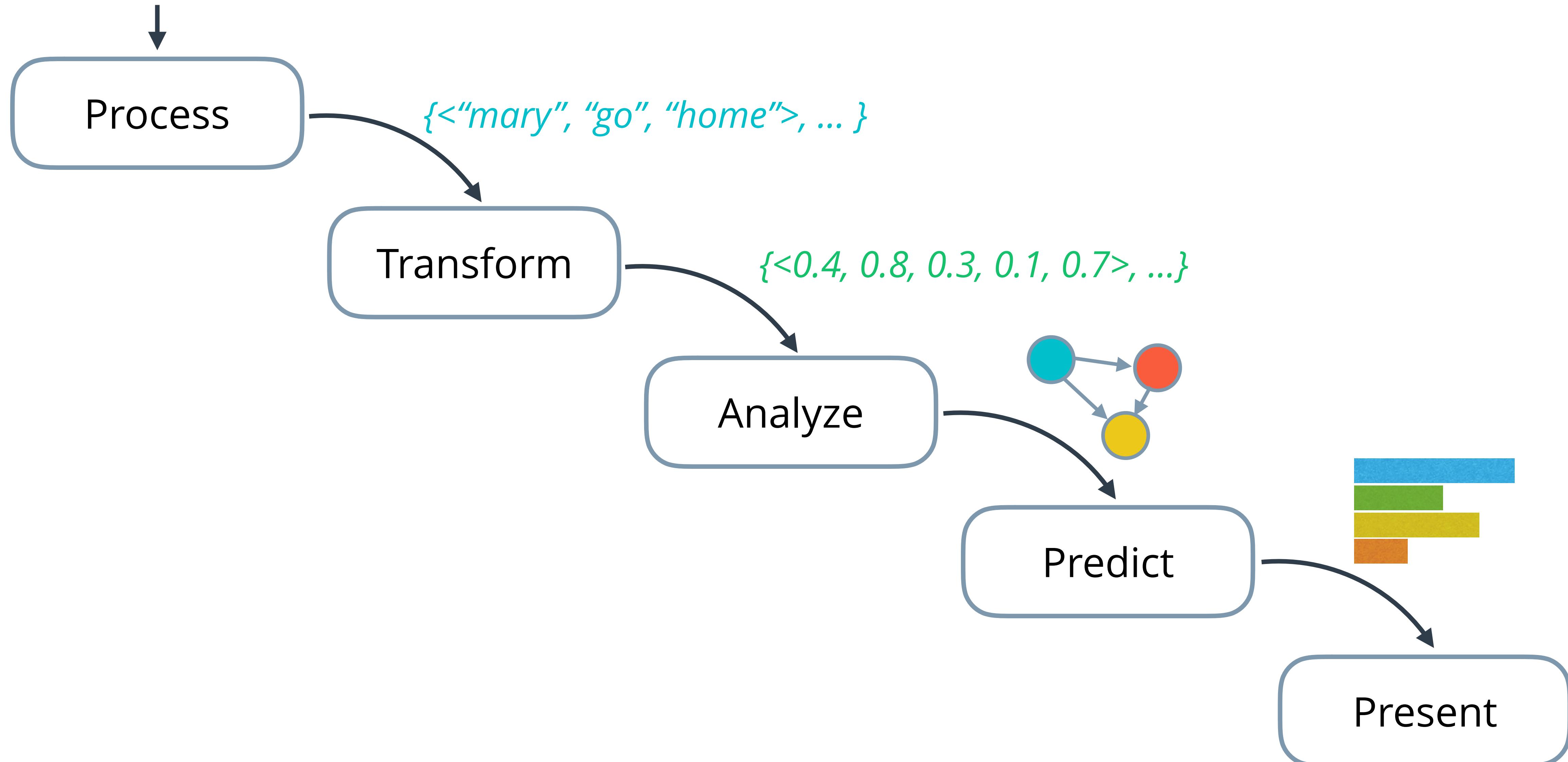
petrol

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. ..."



Classic NLP: Text Processing

Text Processing

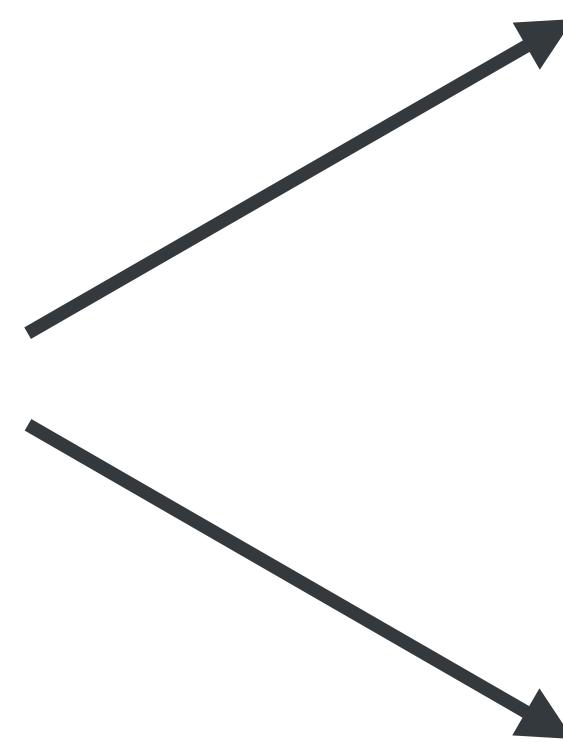
- Tokenization
- Stop Word Removal
- Stemming and Lemmatization

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”, “.”>

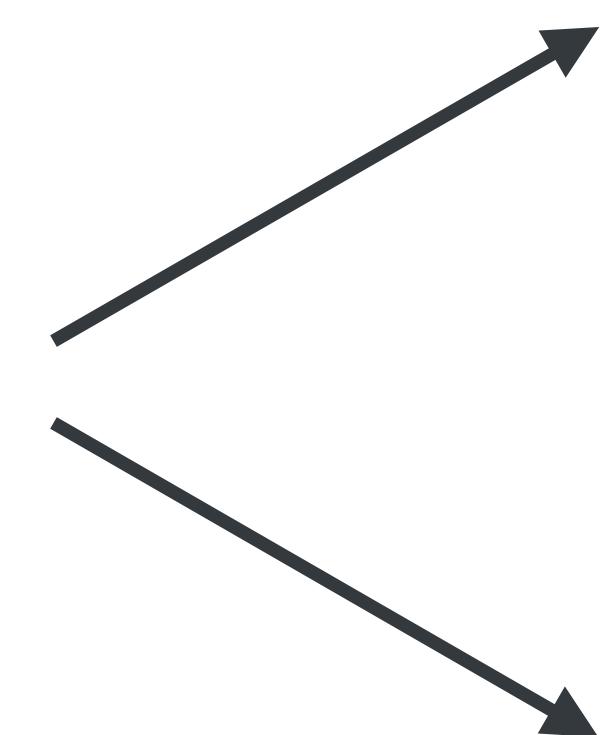
<“no”, “she”, “didnt”,
“do”, “it”>

?

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.

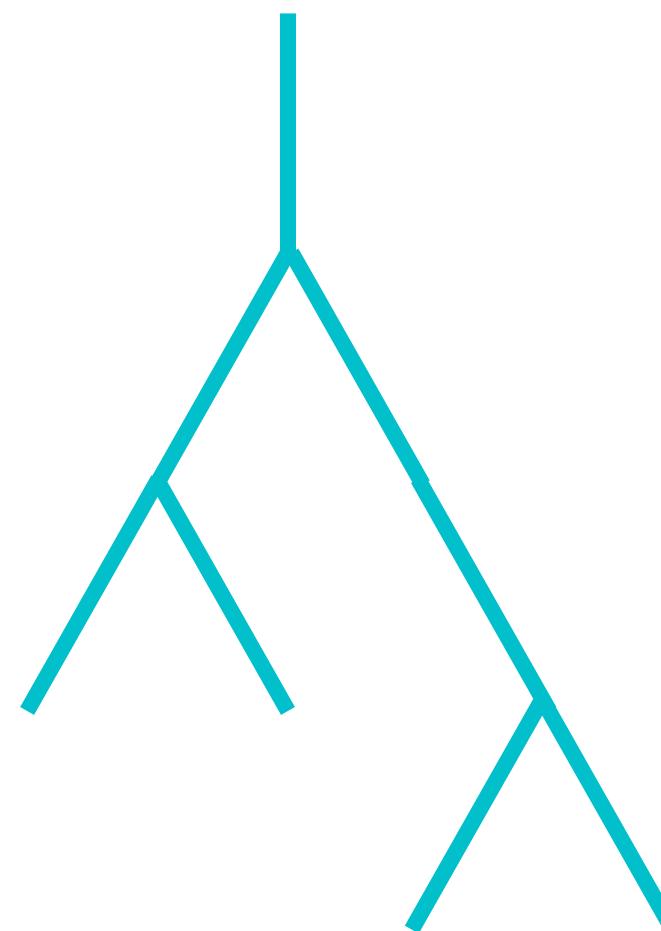
Stemming

branching

branched

branches

branch

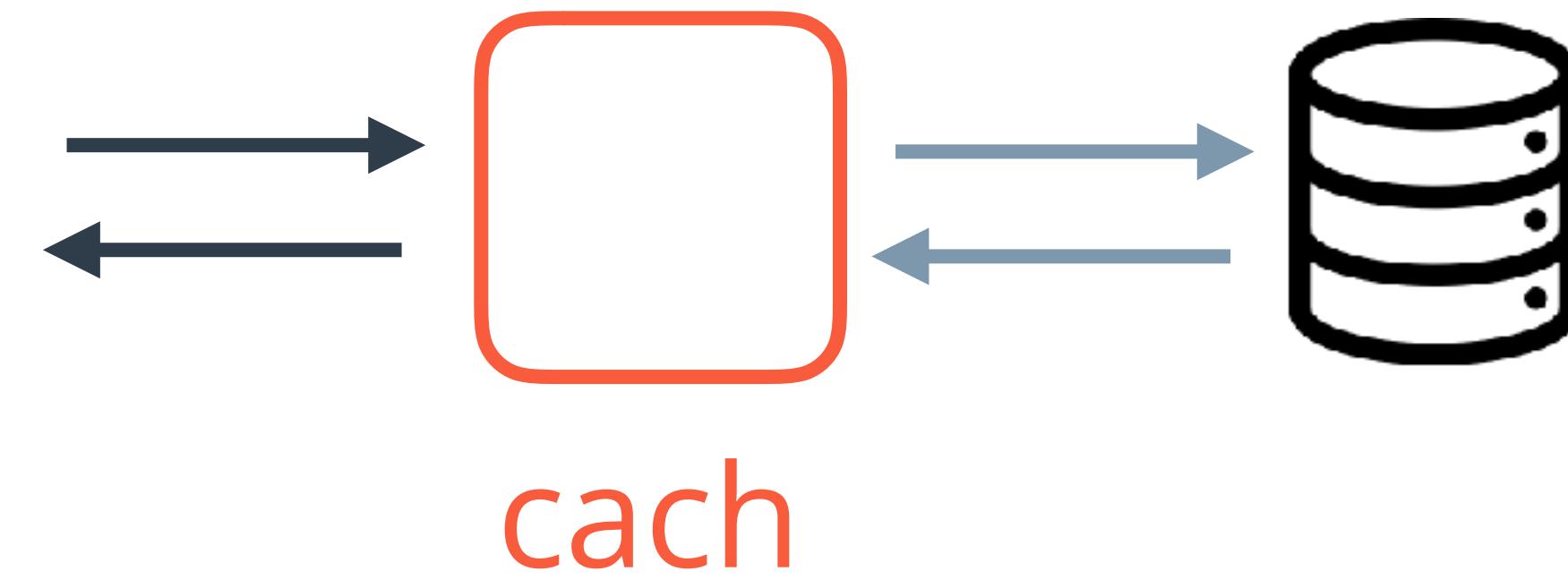
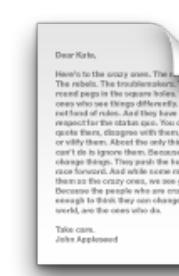


Stemming

caching

cached

caches

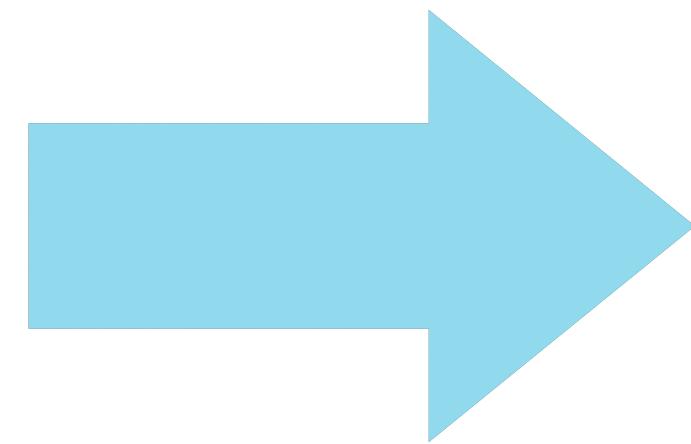


Lemmatization

is

was

were

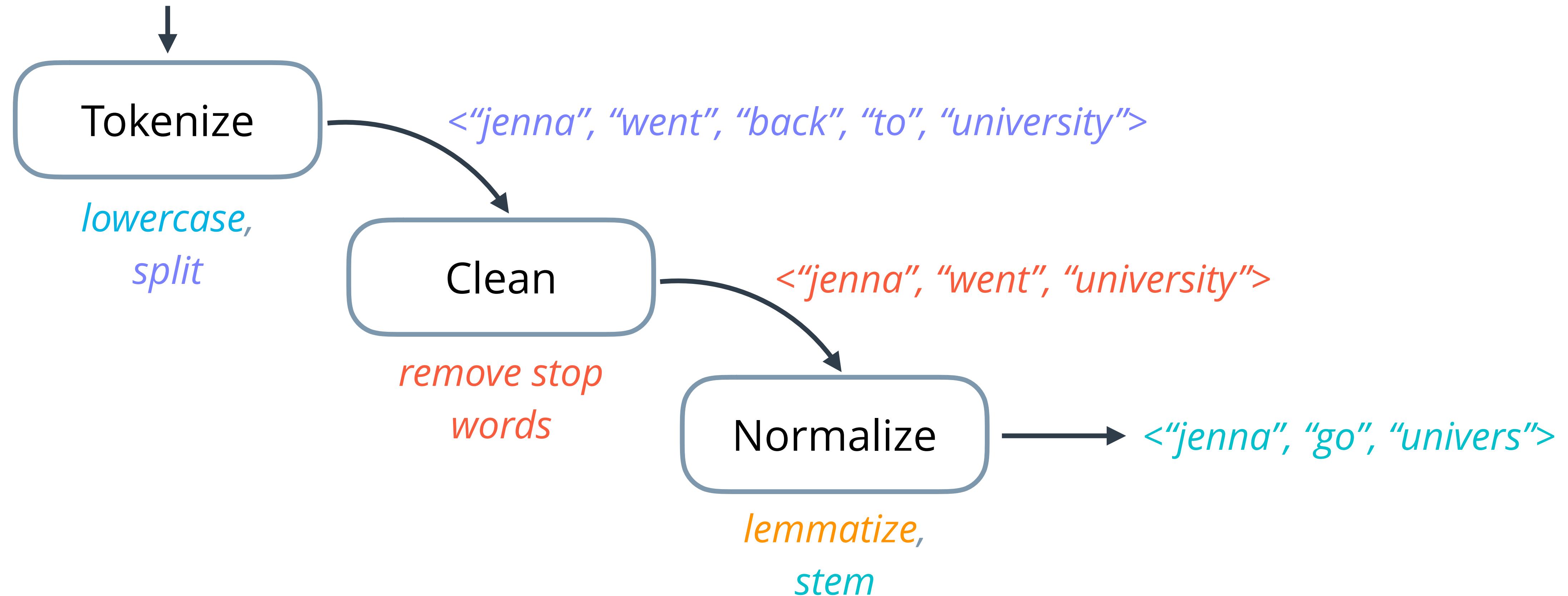


be



Text Processing Summary

"Jenna went back to University."

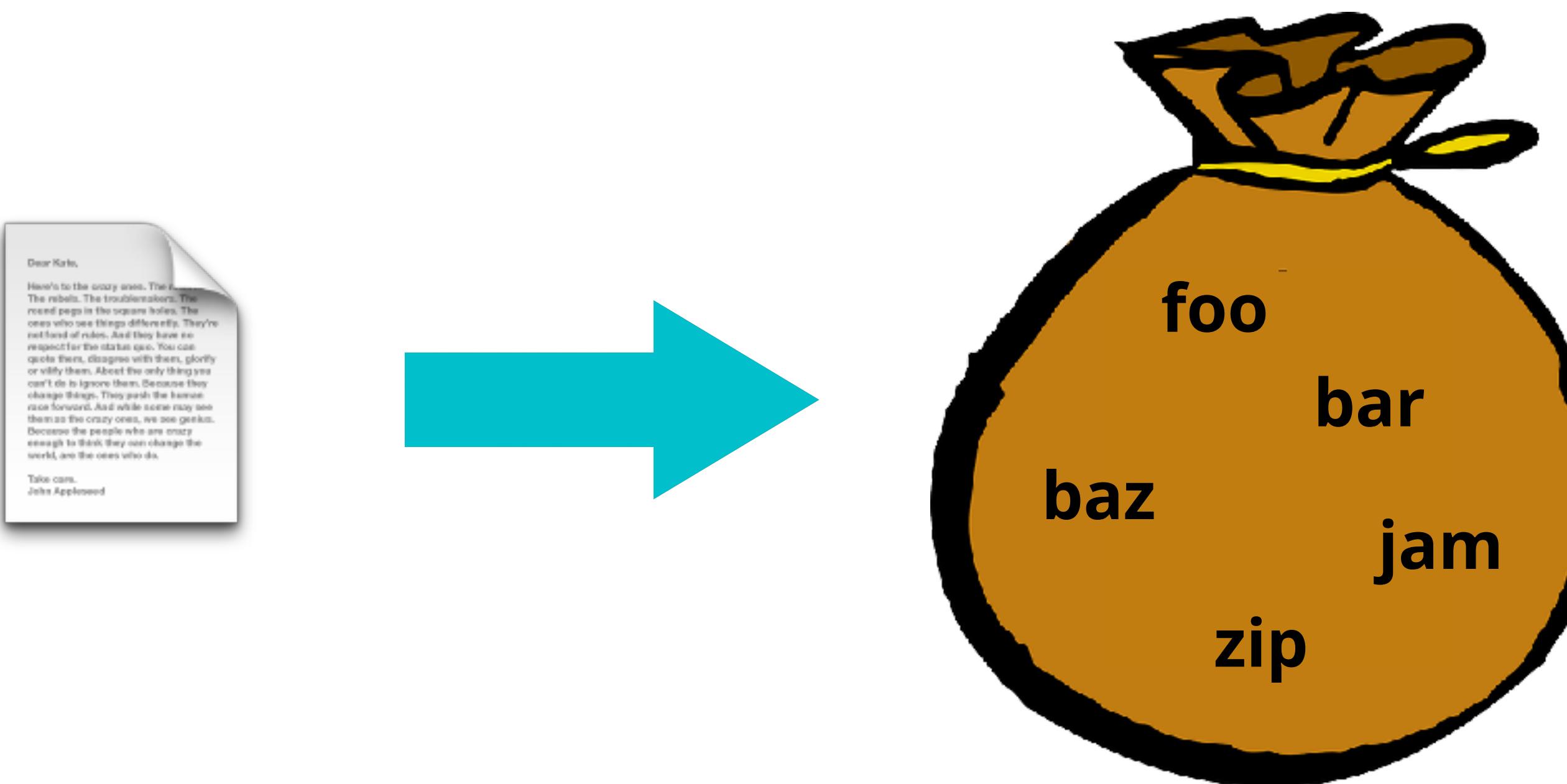


Classic NLP: Feature Extraction

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”

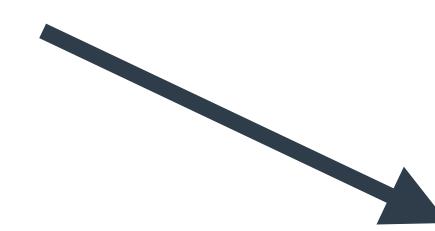


{"twinkl", "littl", "star"}

?

Bag of Words

“Little House on the Prairie”



littl hous prairi mari

“Mary had a Little Lamb”



lamb silenc twinkl star

“The Silence of the Lambs”



“Twinkle Twinkle Little Star”



corpus (D)

vocabulary (V)

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

Bæg værfn Vætrðsírm Matrix

term frequency

“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
1	1	1	0	0	0	0	0
1	0	0	1	1	0	0	0
0	0	0	0	1	1	0	0
1	0	0	0	0	0	2	1

Document Similarity

a “Little House on the Prairie”

	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

$$\mathbf{a} \cdot \mathbf{b} = \sum a_0 b_0 + a_1 b_1 + \dots + a_n b_n = 1 + 0 + 0 \text{ dot product} + 0 + 0 + 0 + 0$$

Document Similarity

a “Little House on the Prairie”

	littl	hous	prairi	mari	lamb	silenc	twinkl	star
a	1	1	1	0	0	0	0	0
b	1	0	0	1	1	0	0	0

$$\mathbf{a} \cdot \mathbf{b} = \sum a_0 b_0 + a_1 b_1 + \dots + a_n b_n = 1 \quad \text{dot product}$$

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

Term Specificity

“Little House on the Prairie”

	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
<i>document frequency</i> —	3	1	1	1	2	1	1	1

Term Specificity

“Little House on the Prairie”

littl	hous	prairi	mari	lamb	silenc	twinkl	star
1/3	1	1	0	0	0	0	0
1/3	0	0	1	1/2	0	0	0
0	0	0	0	1/2	1	0	0
1/3	0	0	0	0	0	2	1

“Mary had a Little Lamb”

“The Silence of the Lambs”

“Twinkle Twinkle Little Star”

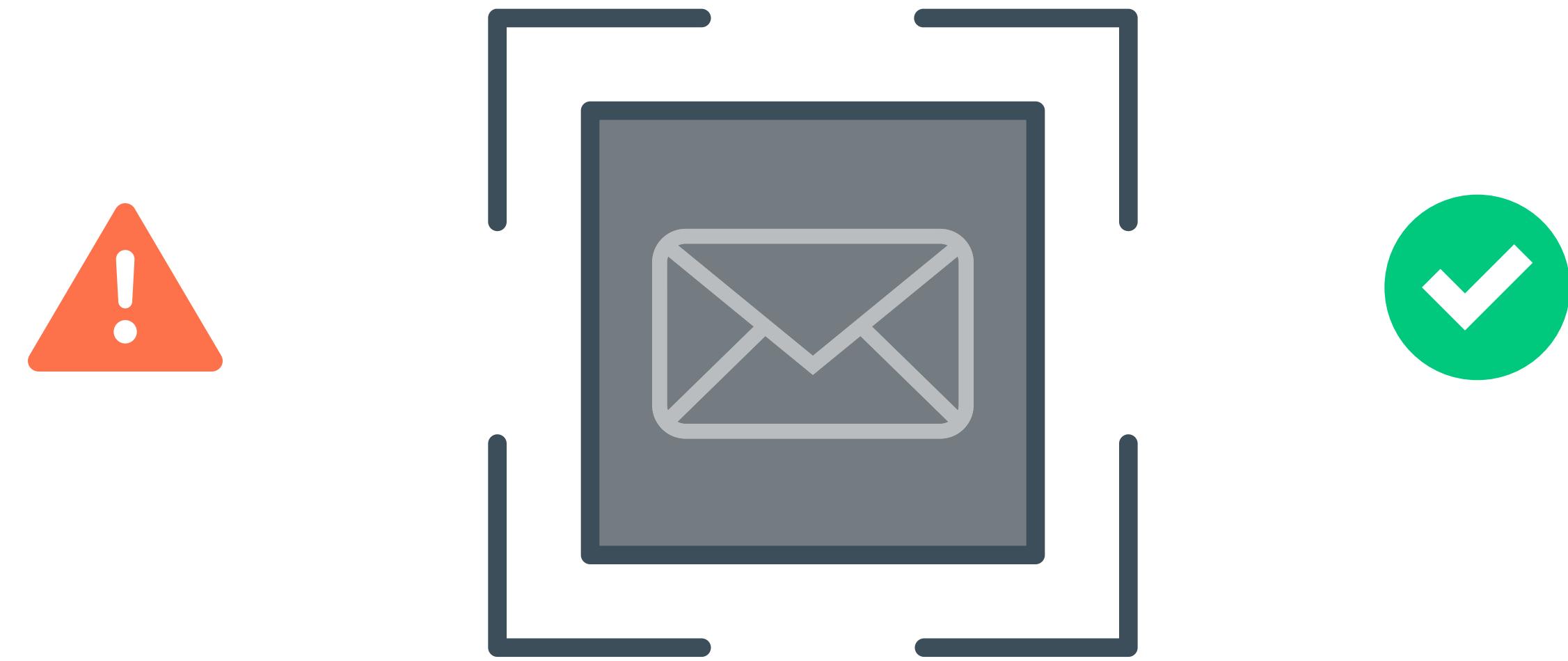
TF-IDF

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

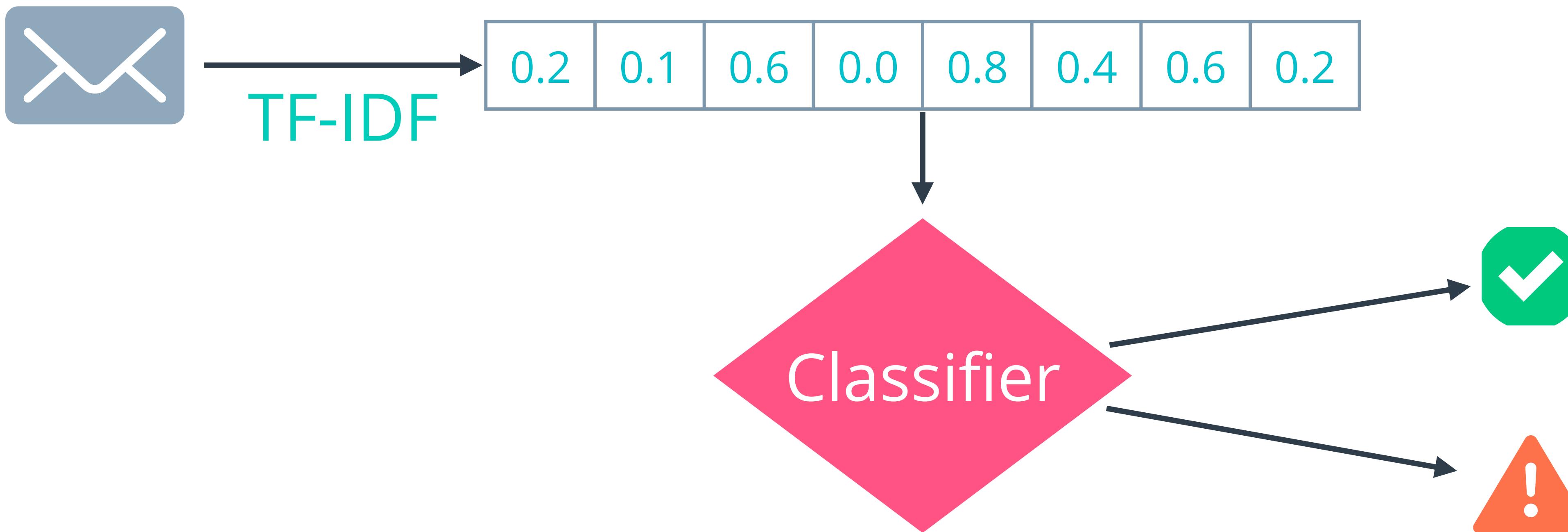
term frequency
 $\text{count}(t, d)/|d|$

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

Example Task: Spam Detection



Example Task: Spam Detection

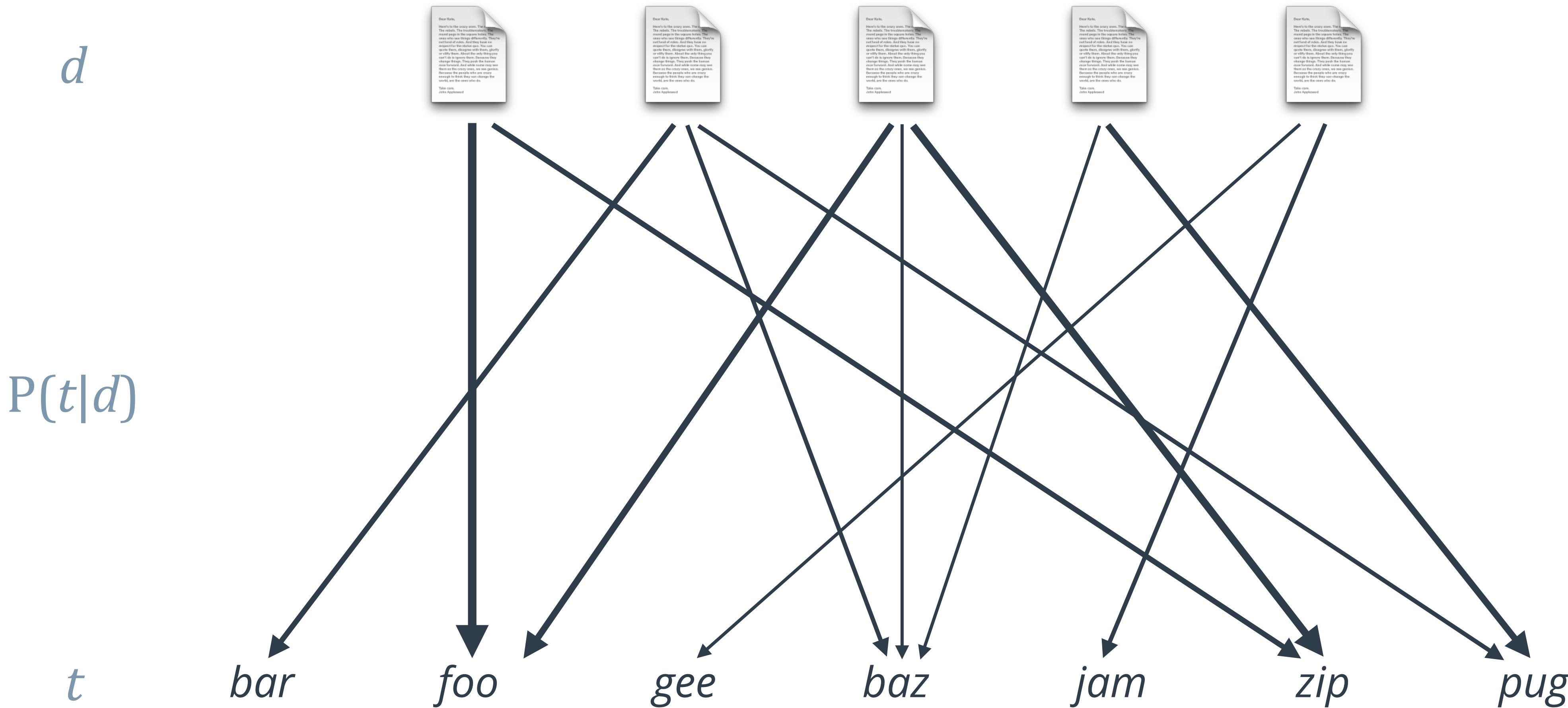


Classic NLP: Topic Modeling

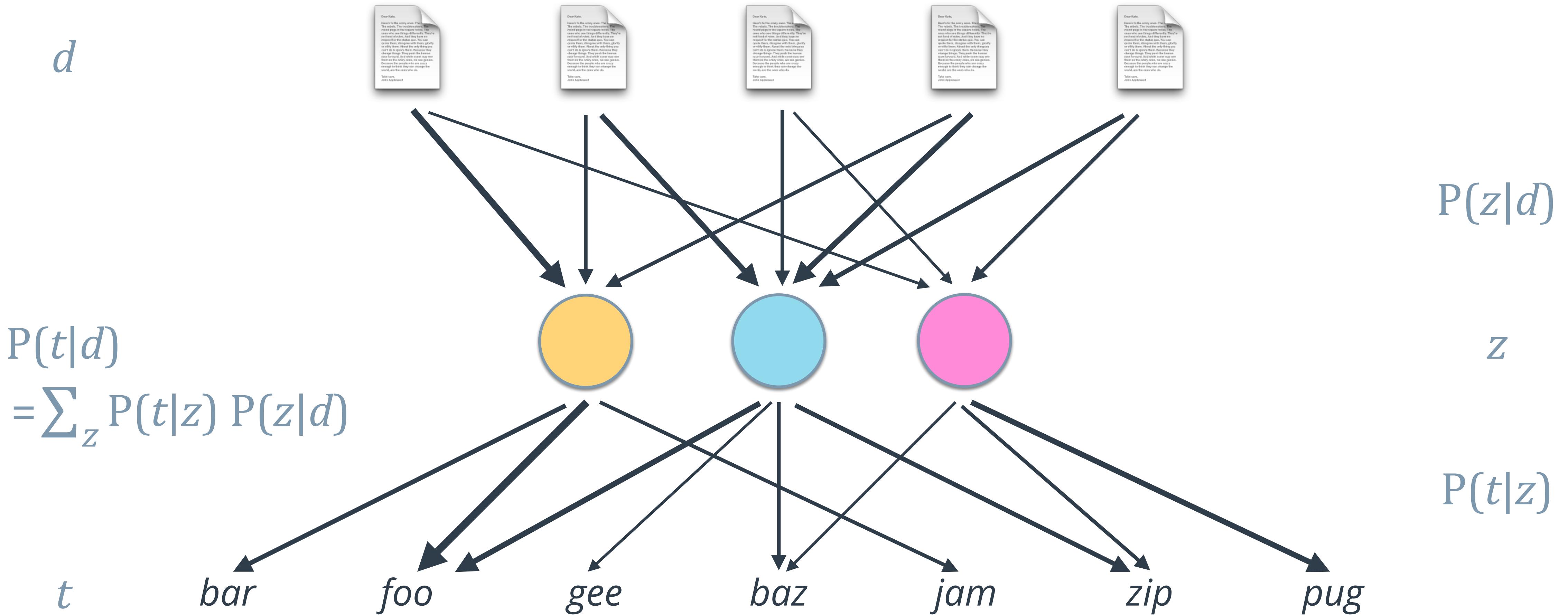
Topic Modeling

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

Bag of Words: Graphical Model



Latent Variables



Missing Priors

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

conditional probabilities

$$P(t, d) = ? \quad P(t, z) = ?$$

joint probabilities

$$P(d)$$

$$P(z)$$

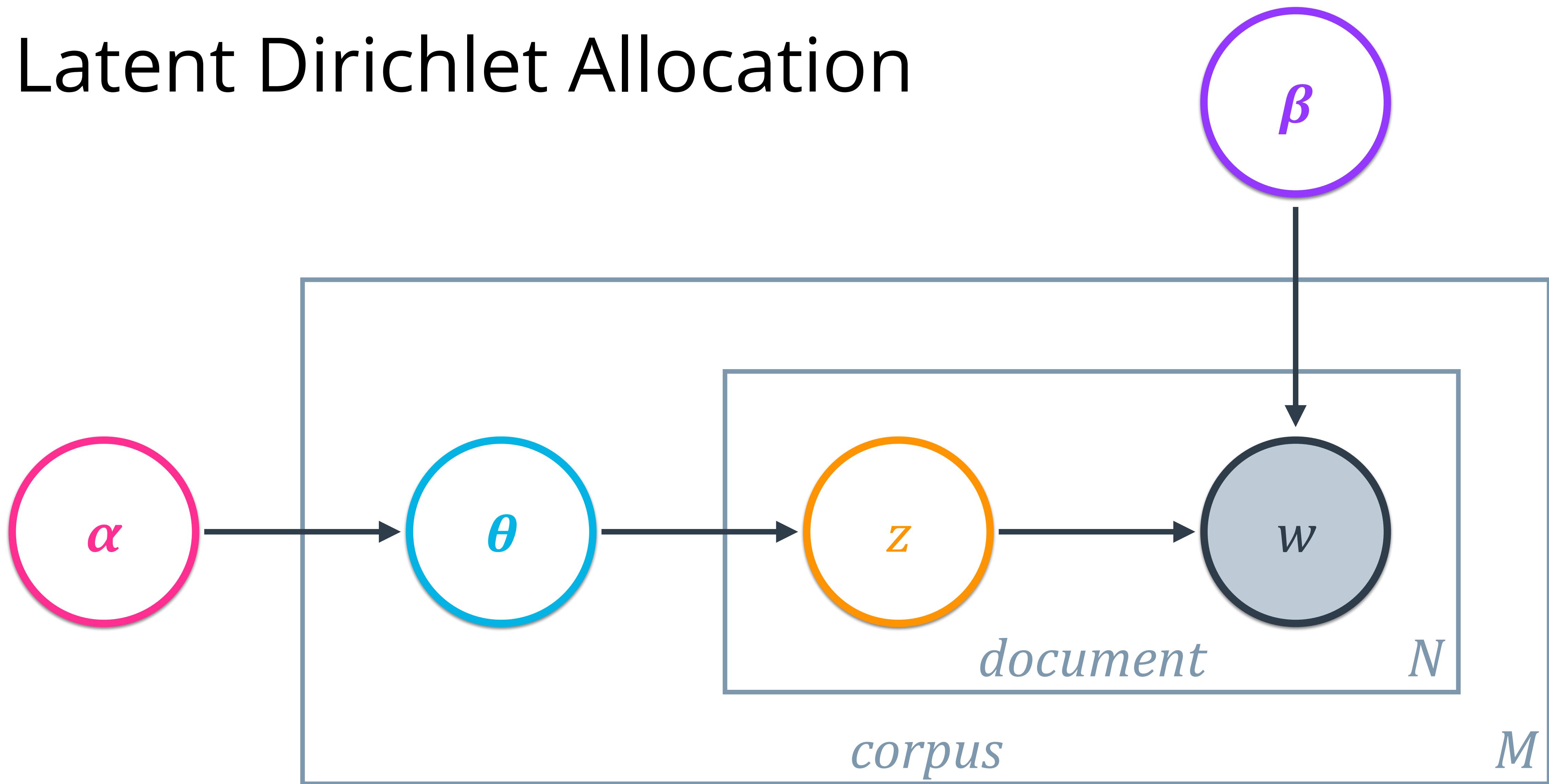
prior probabilities

$$\alpha$$

$$\beta$$

Dirichlet distributions

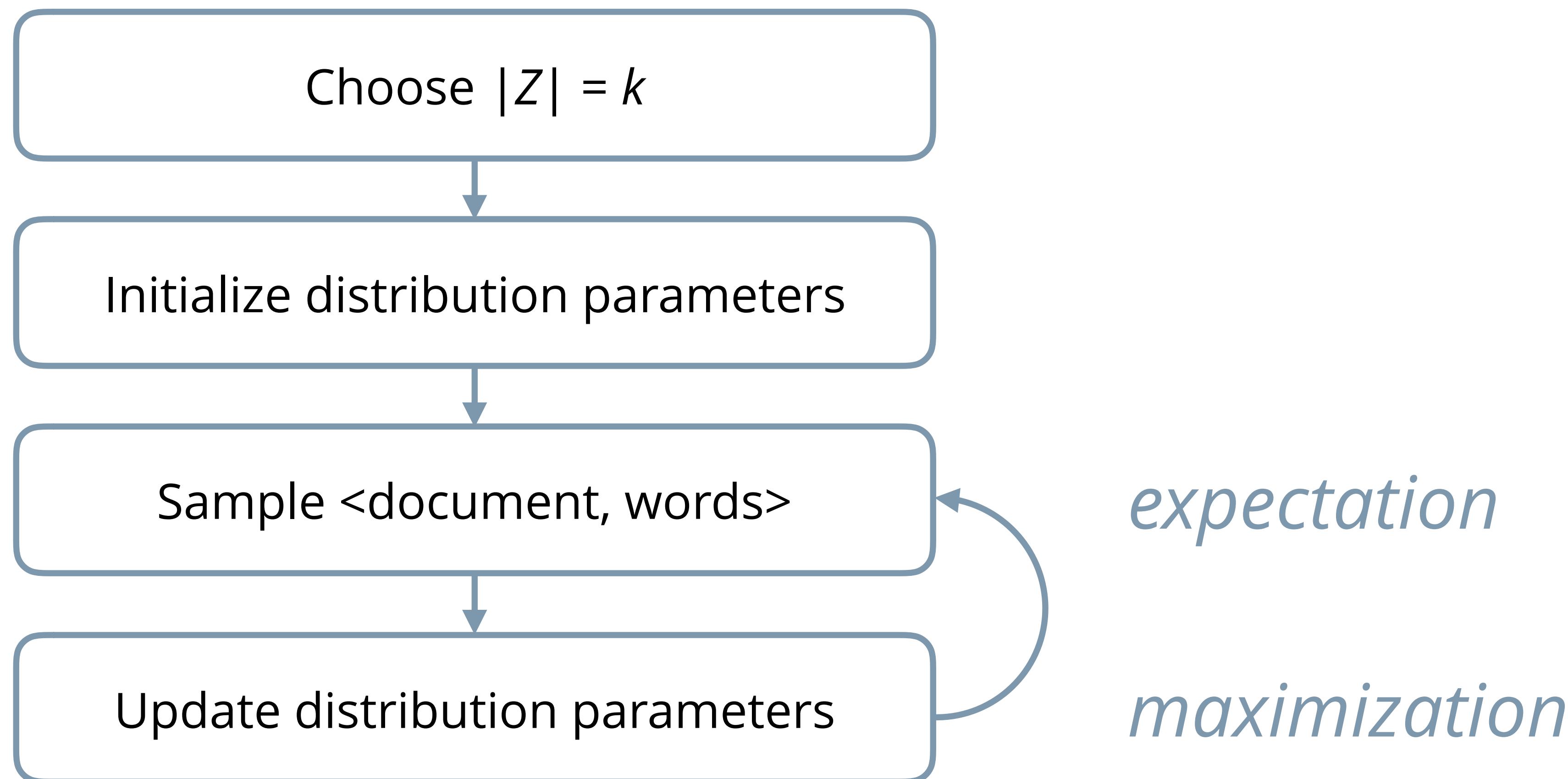
Latent Dirichlet Allocation



LDA: Use Cases

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

LDA: Parameter Estimation



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

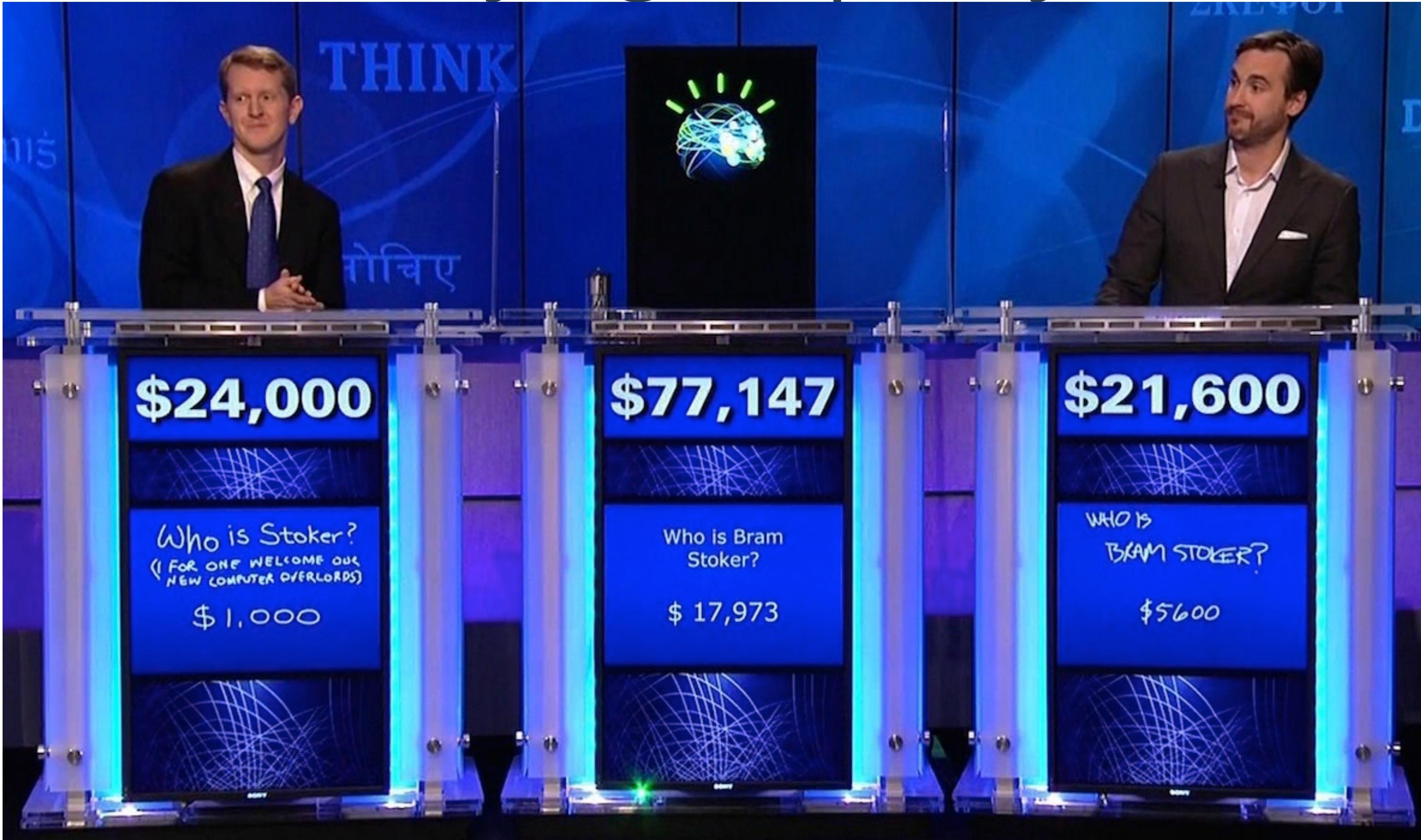
Categorize Newsgroups Data

Deep NLP: Neural Networks

Playing Go



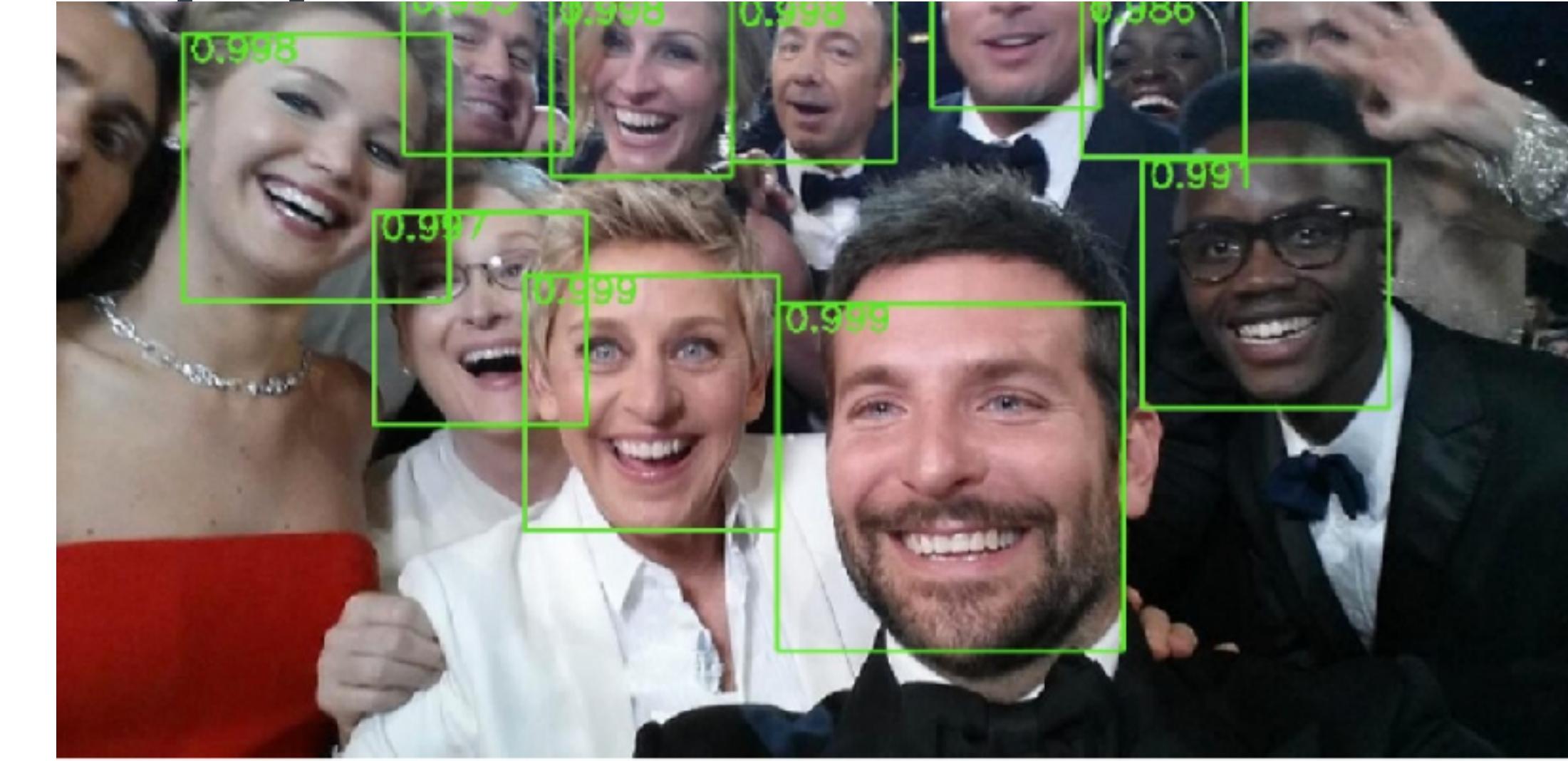
Playing Jeopardy



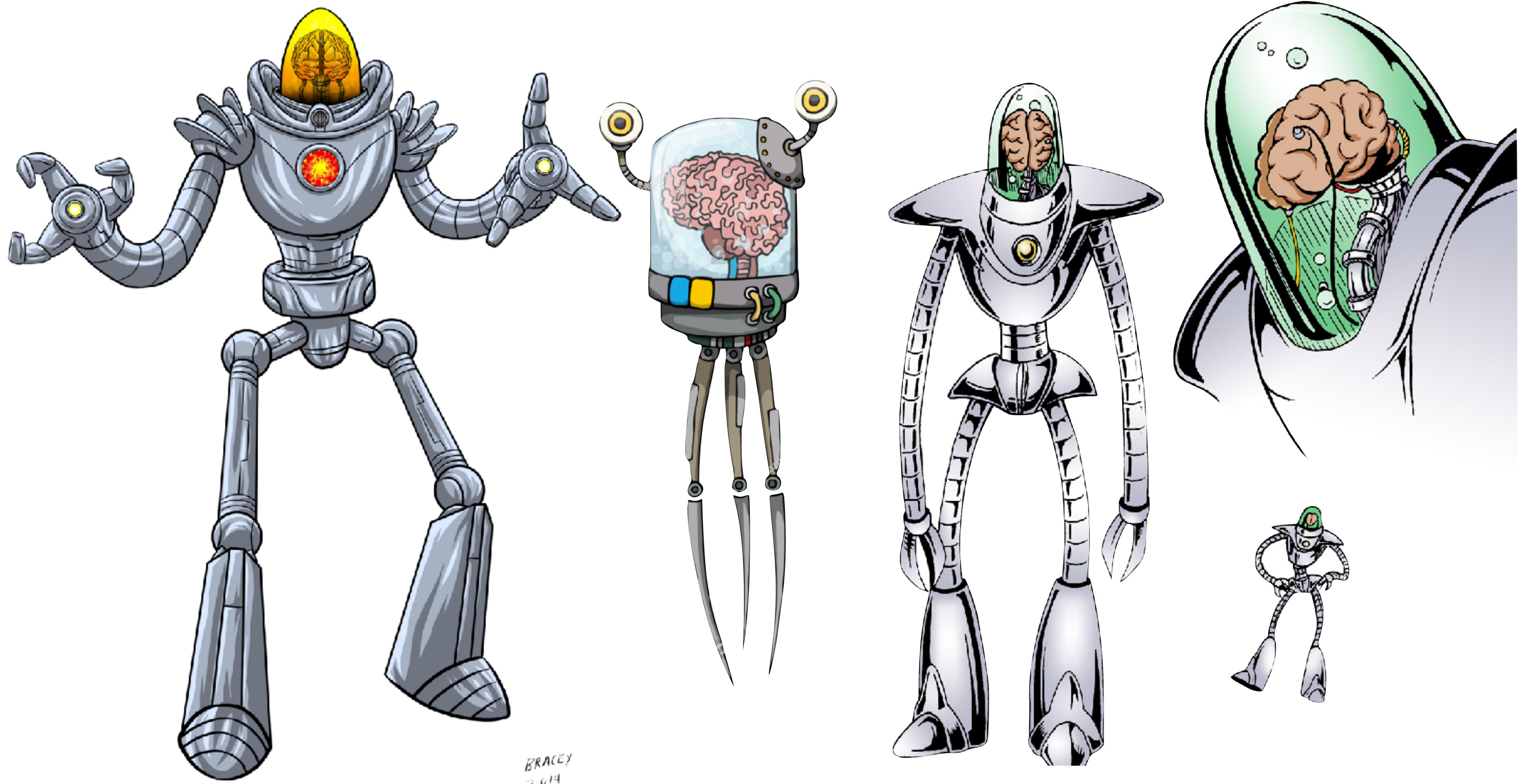
Self Driving Car



Many other applications



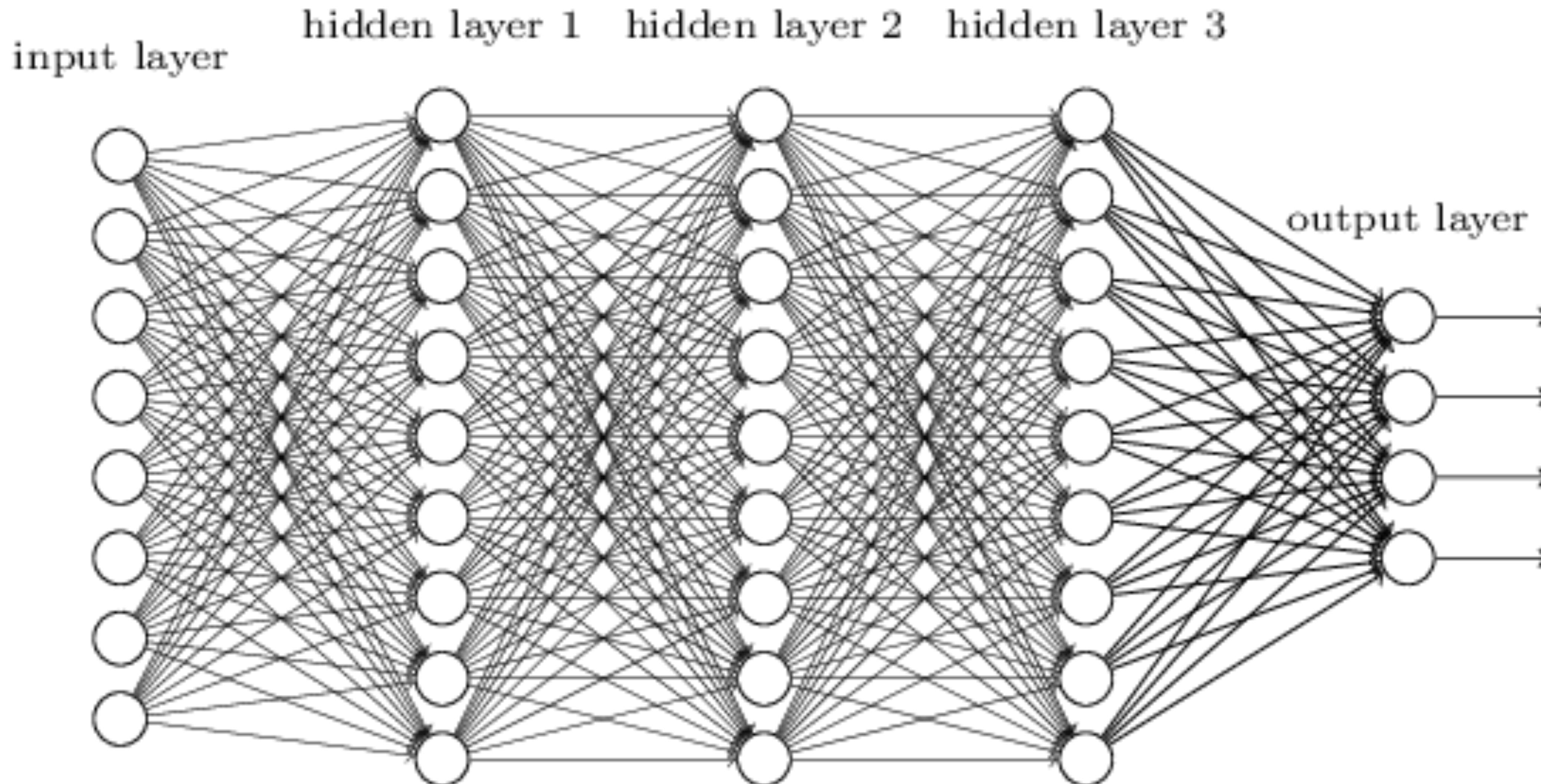
Neural Networks



Neural Networks



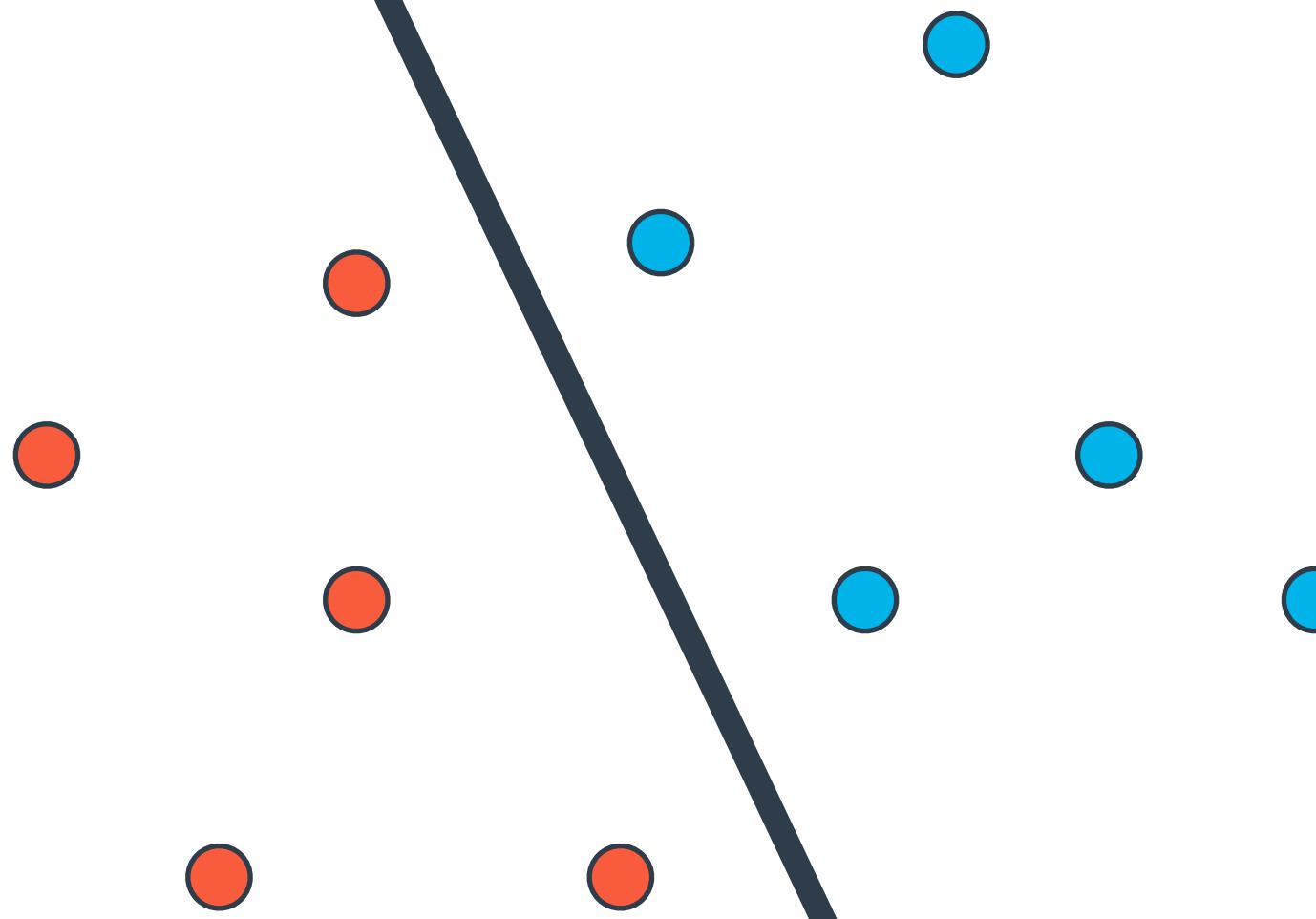
Neural Networks



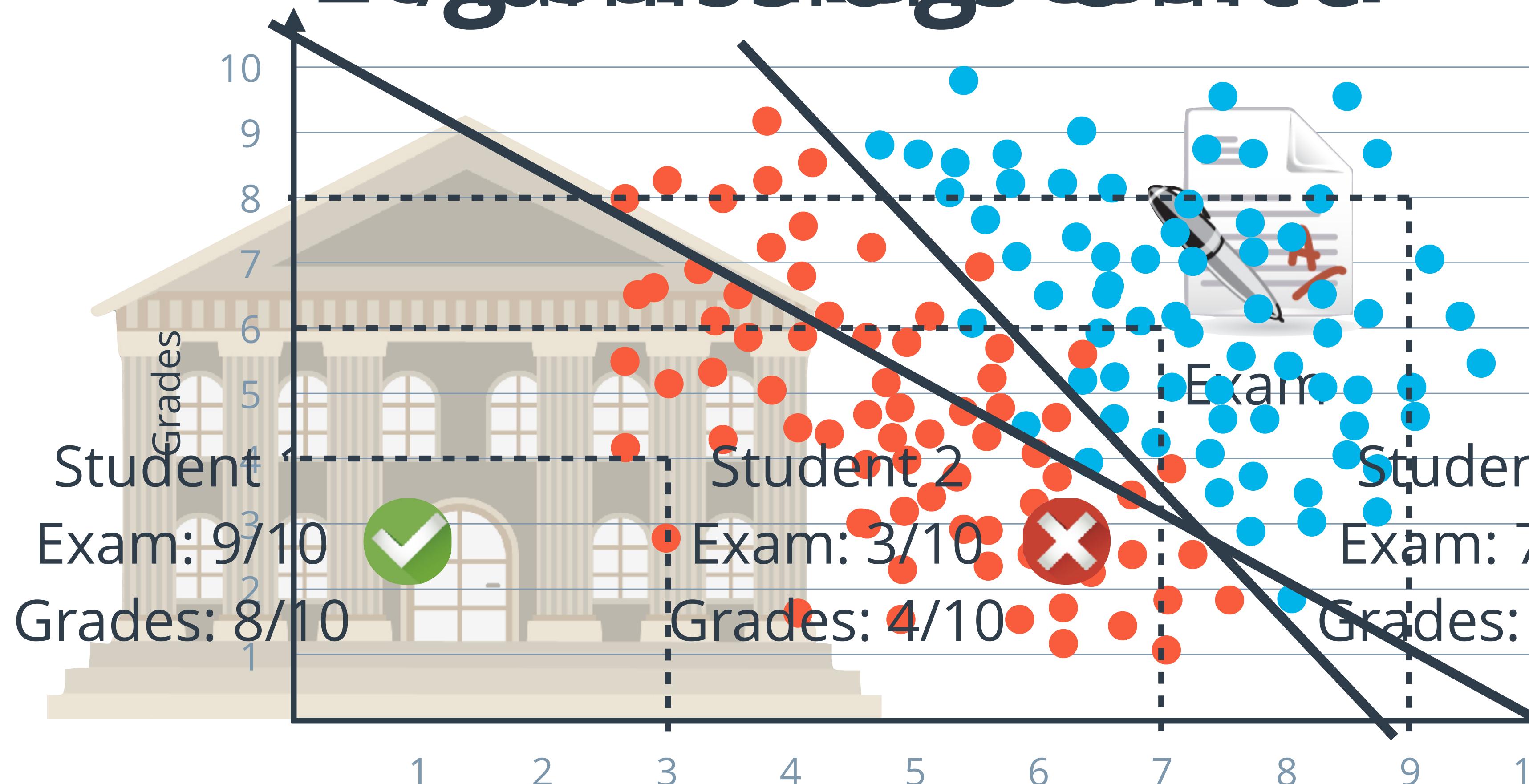
Neural Networks



Goal: Split Data



Logistics Regression



Score = Exam + Grades
Score > 10

Score = Exam + 2*Grades
Score > 18

Score = Exam - Grades
Score > 5

Admissions Office



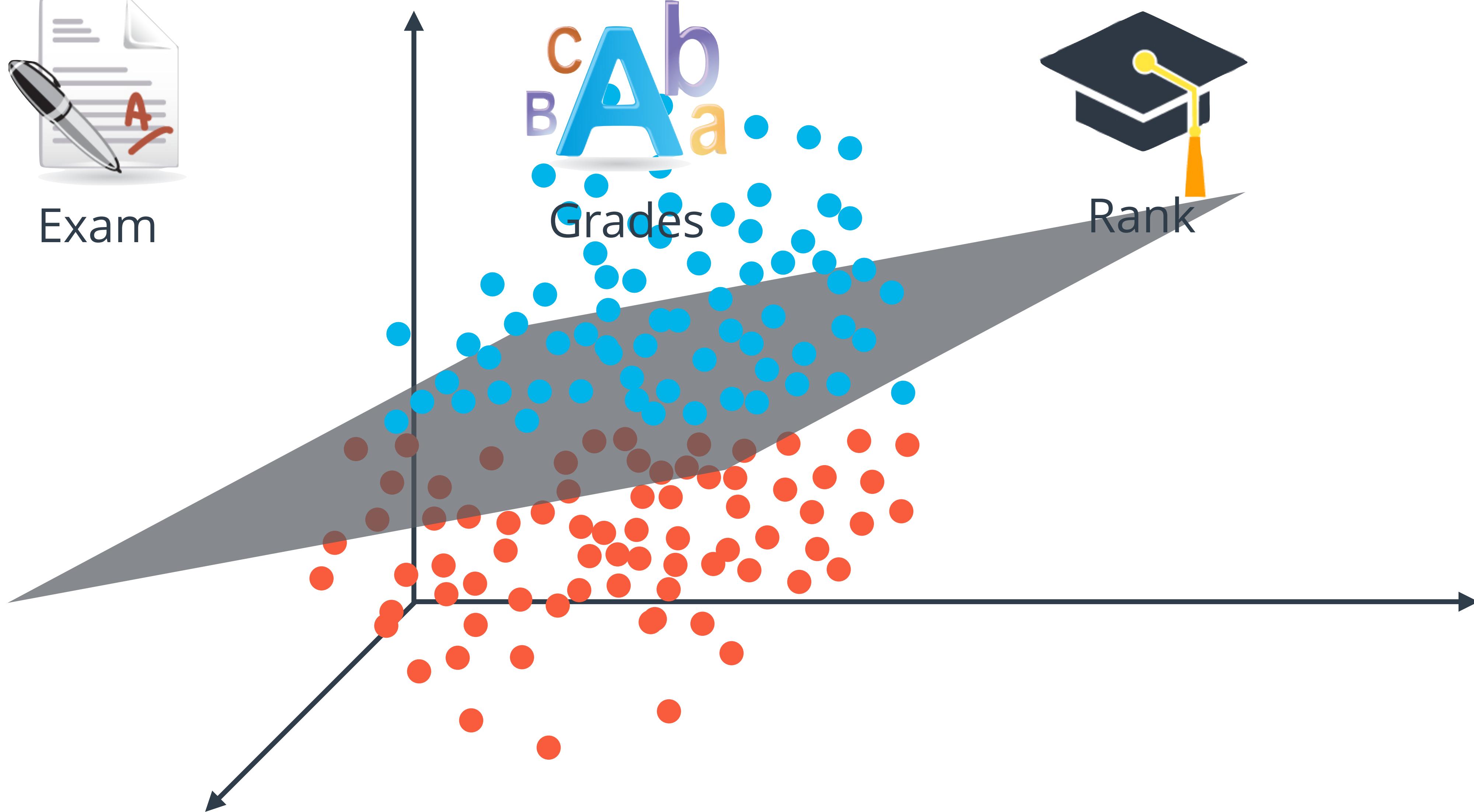
Exam

c
B
A
b
a

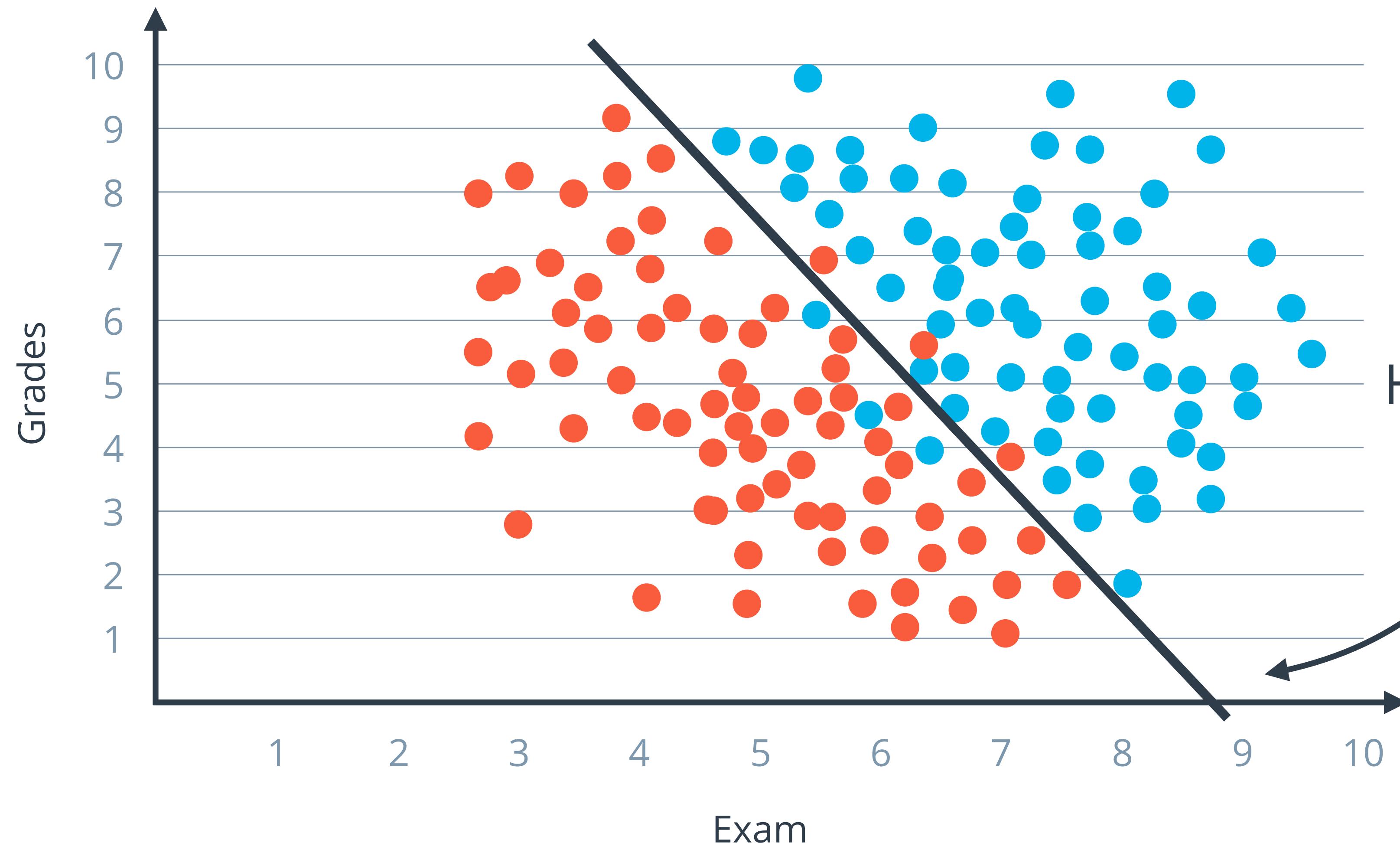
Grades



Rank

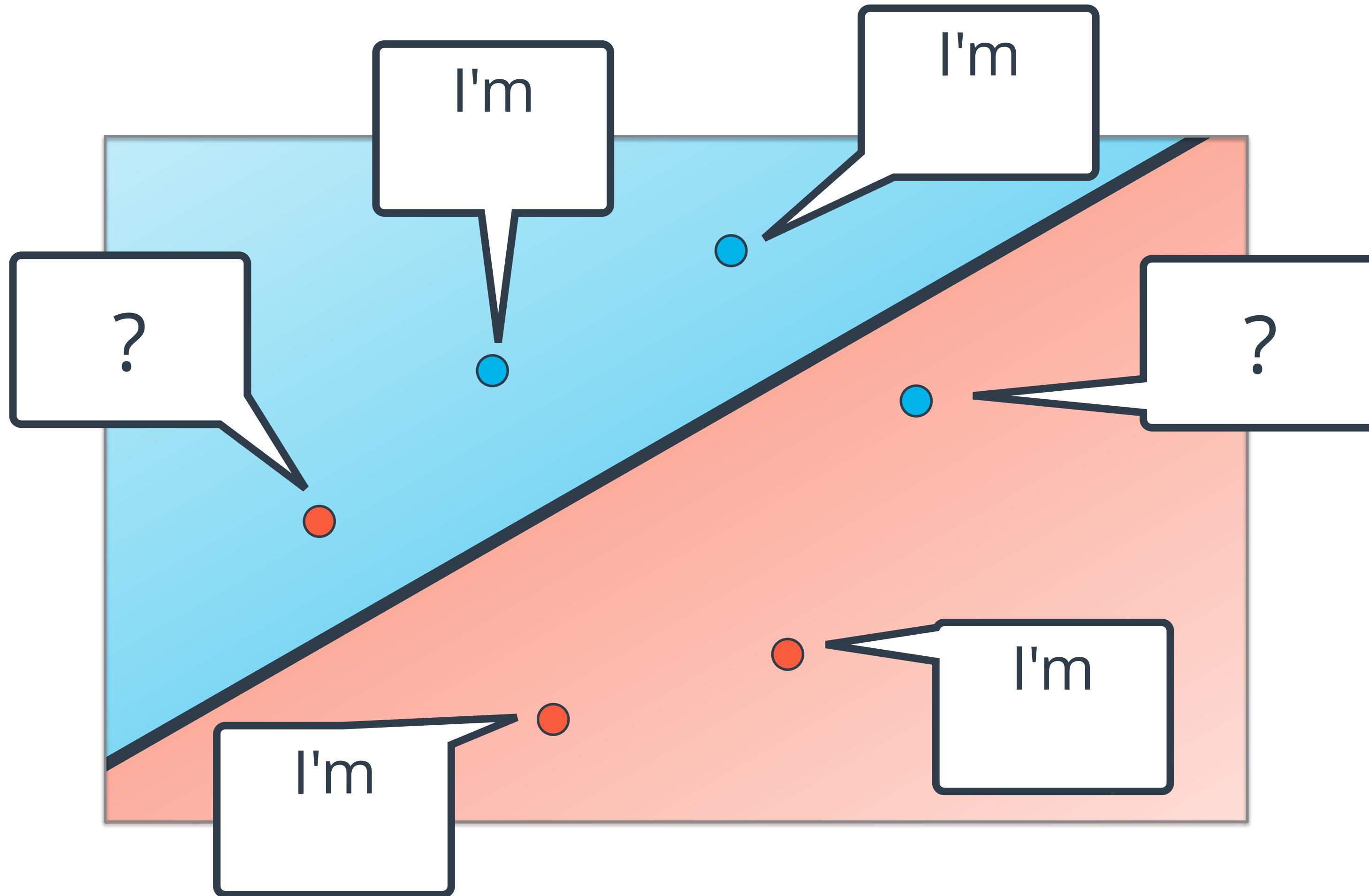


Admissions Office

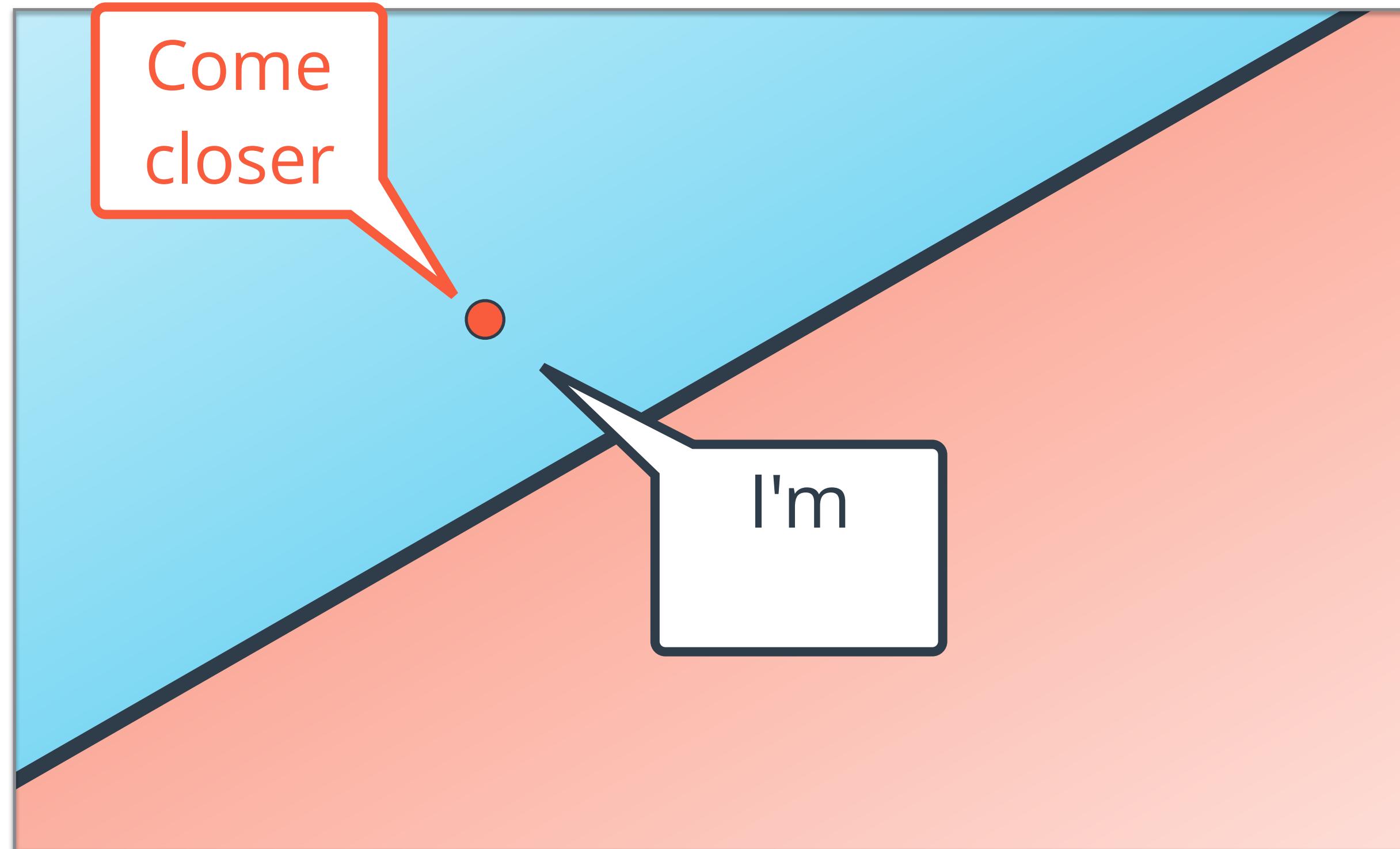


Question:
How do we find this line?

Goal: Split Data



Goal: Split Data

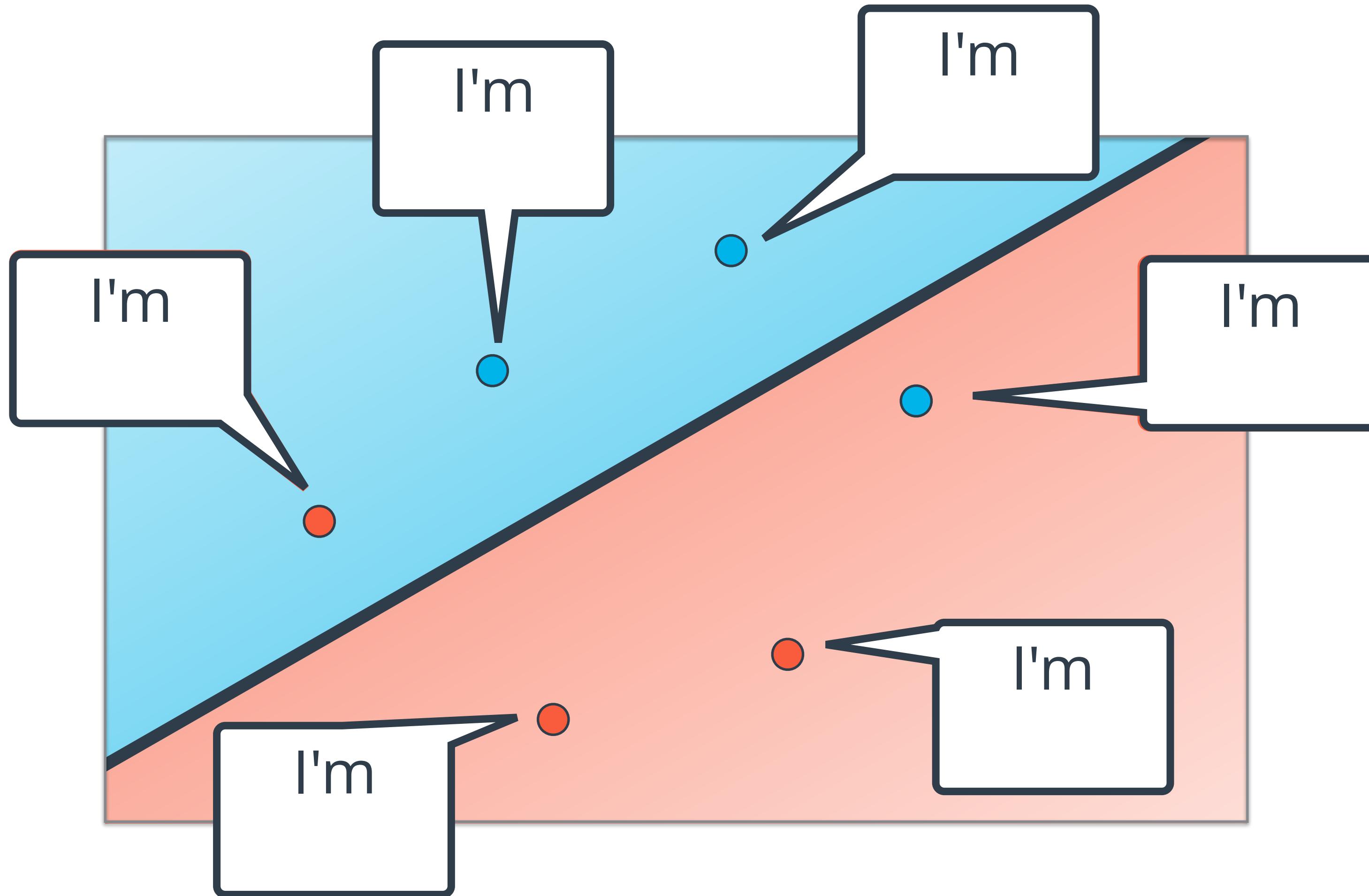


Quiz

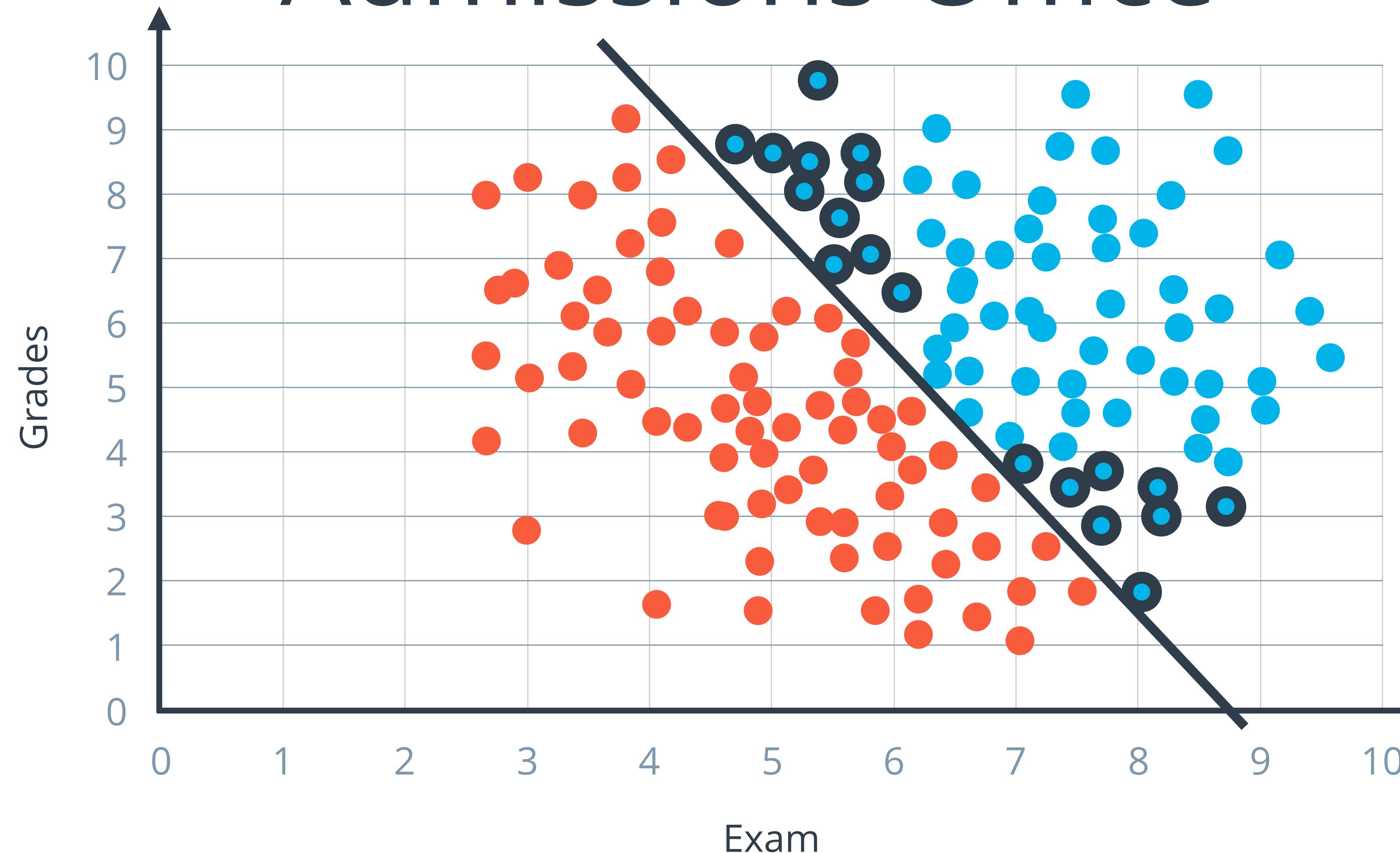
Where would the misclassified point want the line to move?

- Closer
- Farther

Algoritmo del Perceptron



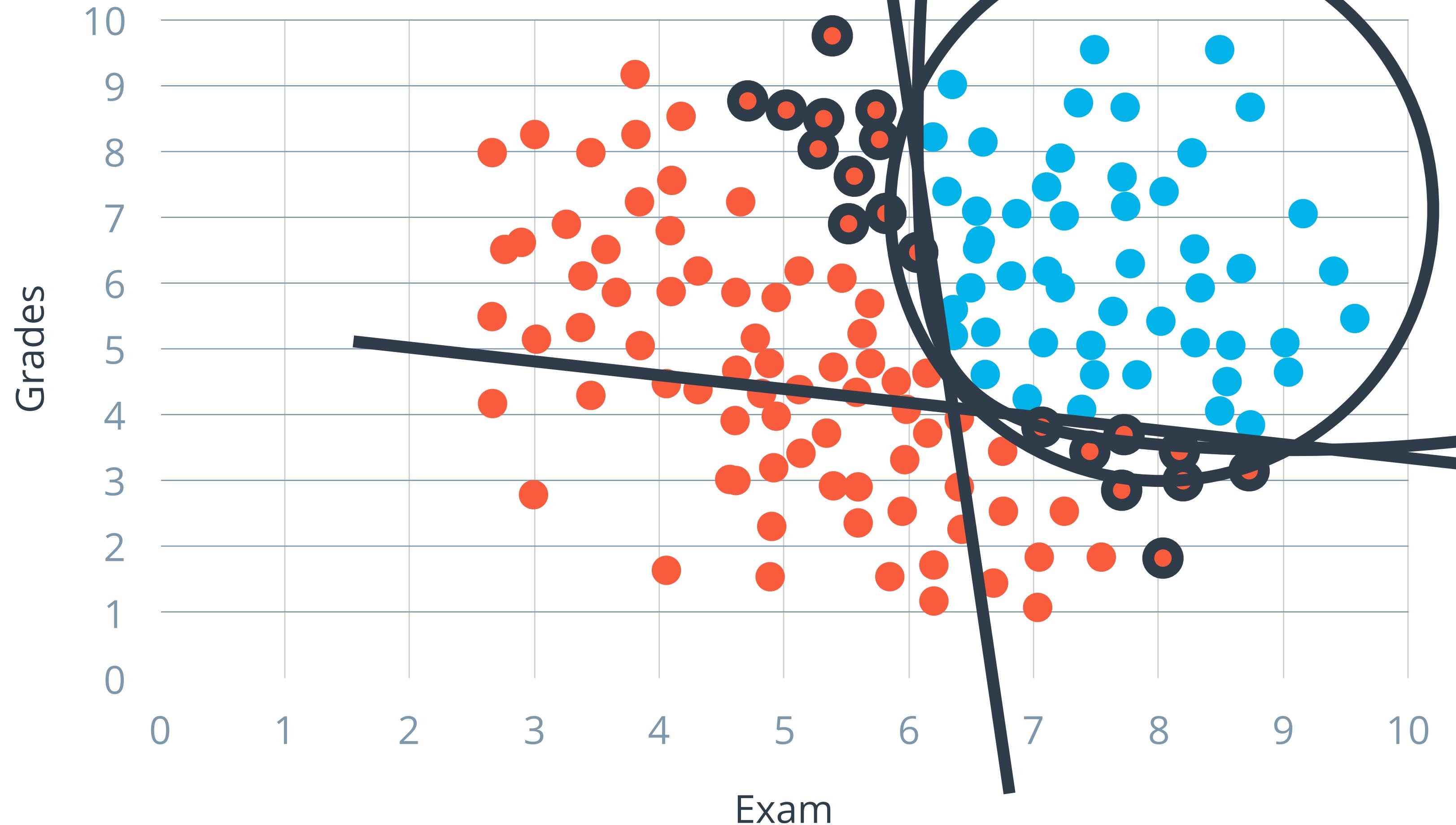
Admissions Office



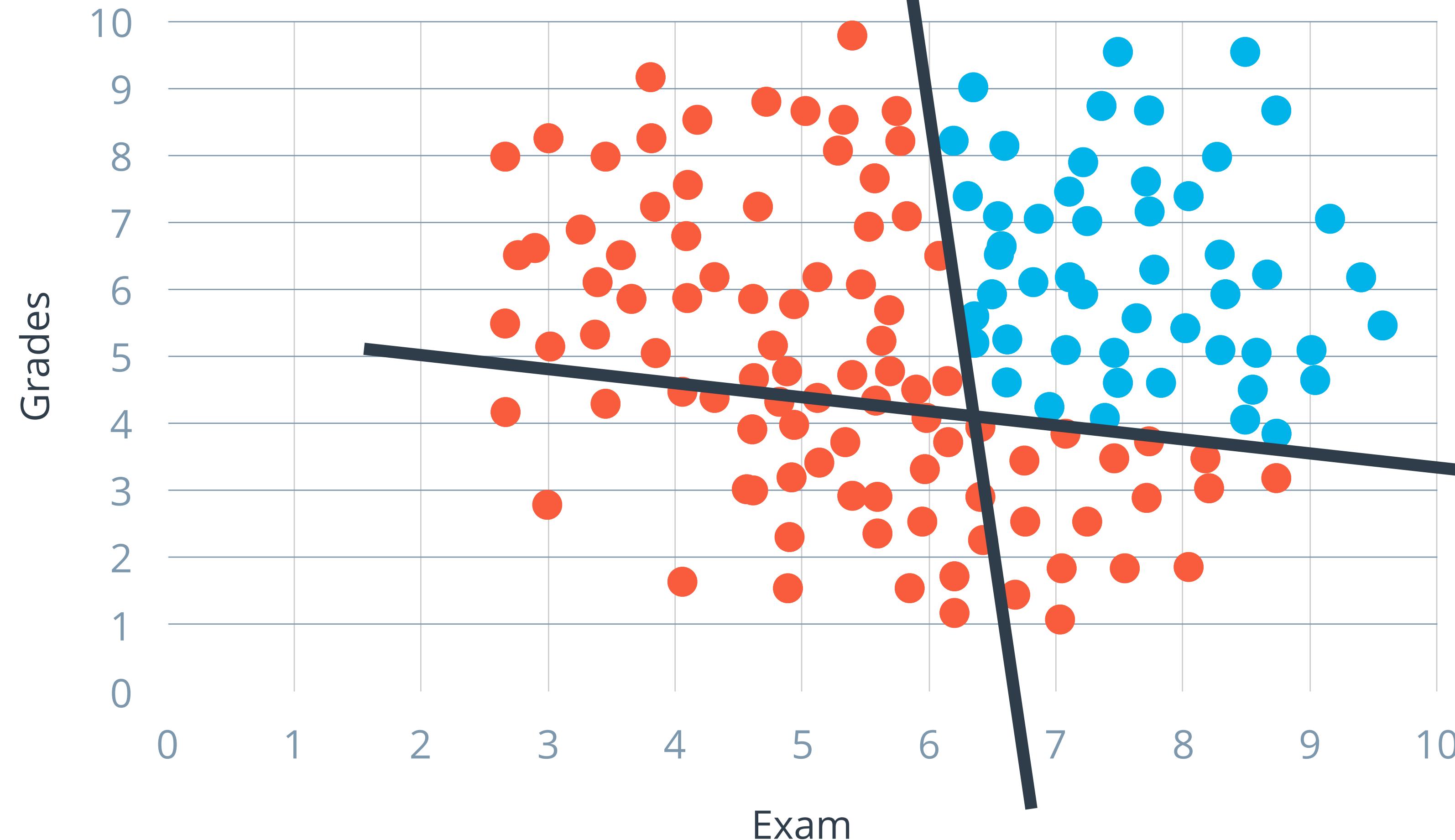
Student 4
Exam: 10
Grades: 0



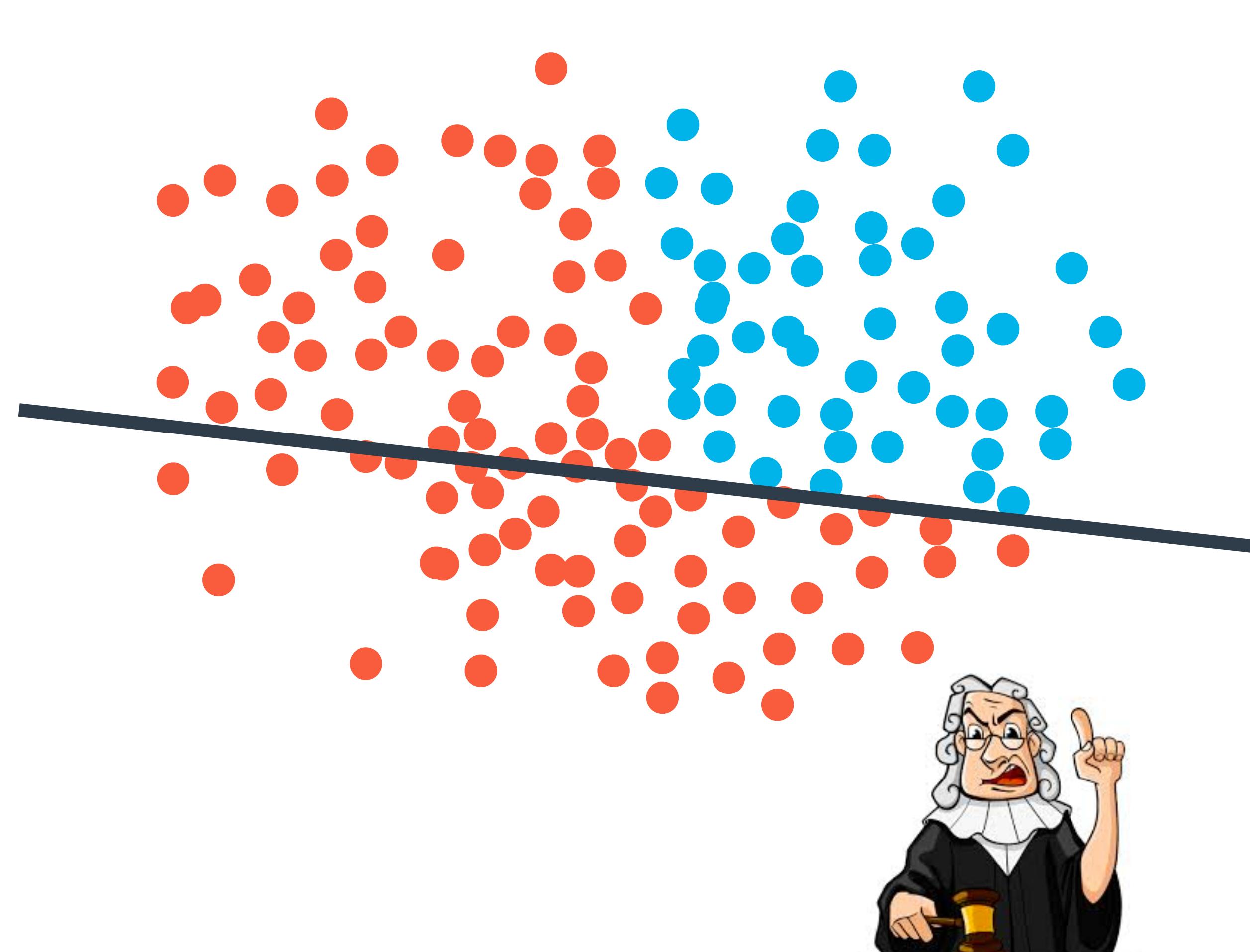
Admissions office



Admissions Office

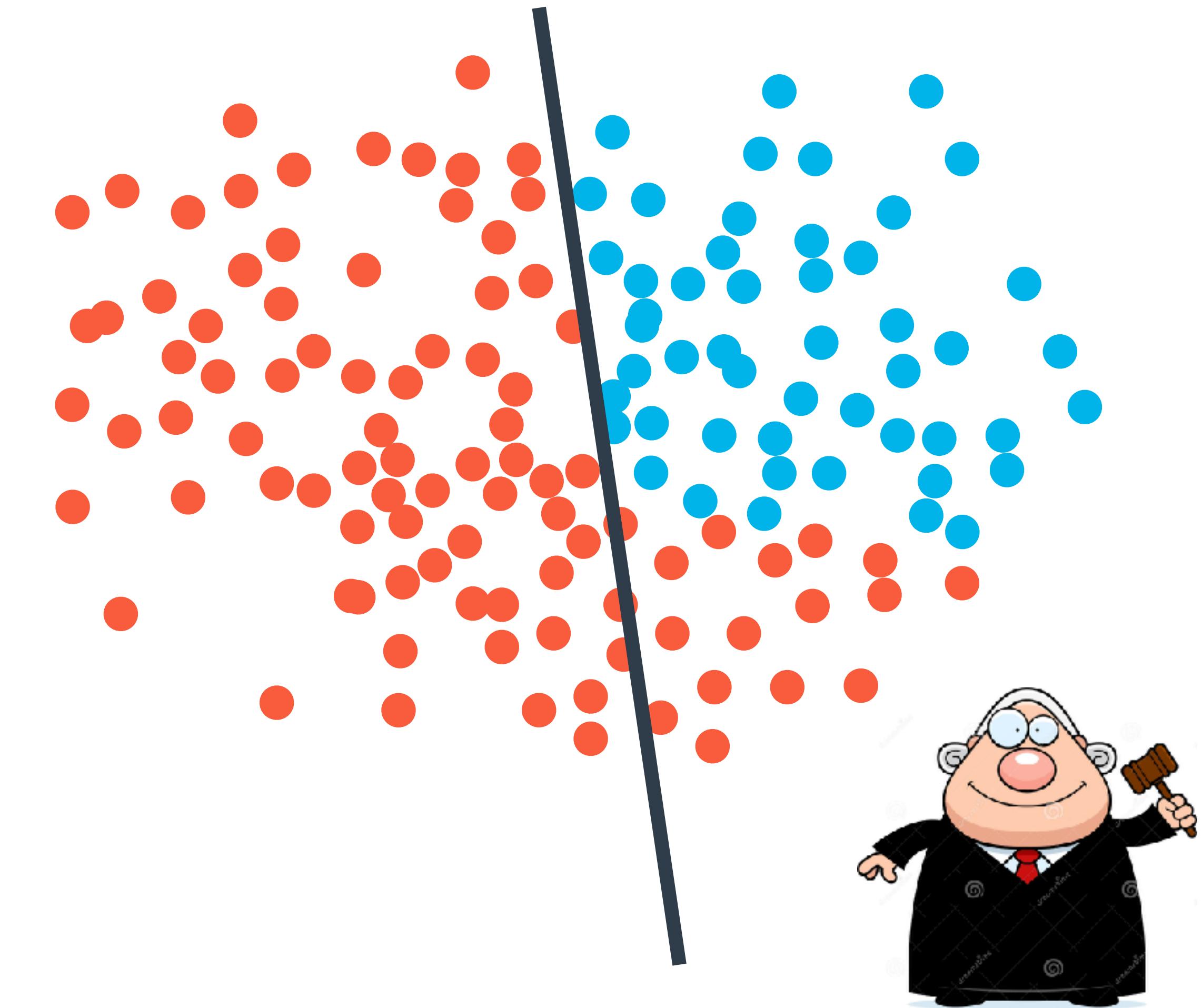


Admissions Office



Judge 1

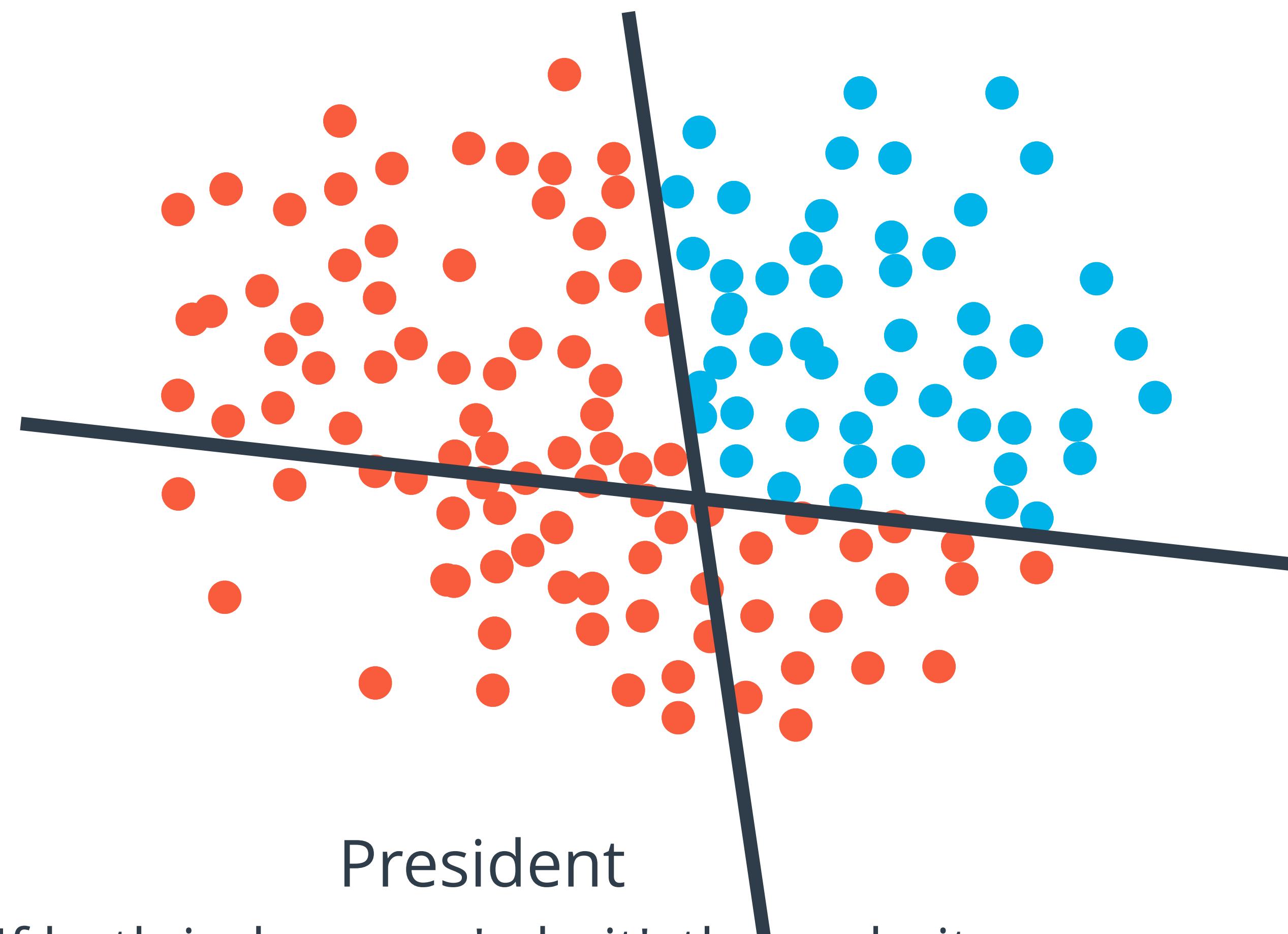
Score = 1*Exam + 8*Grades



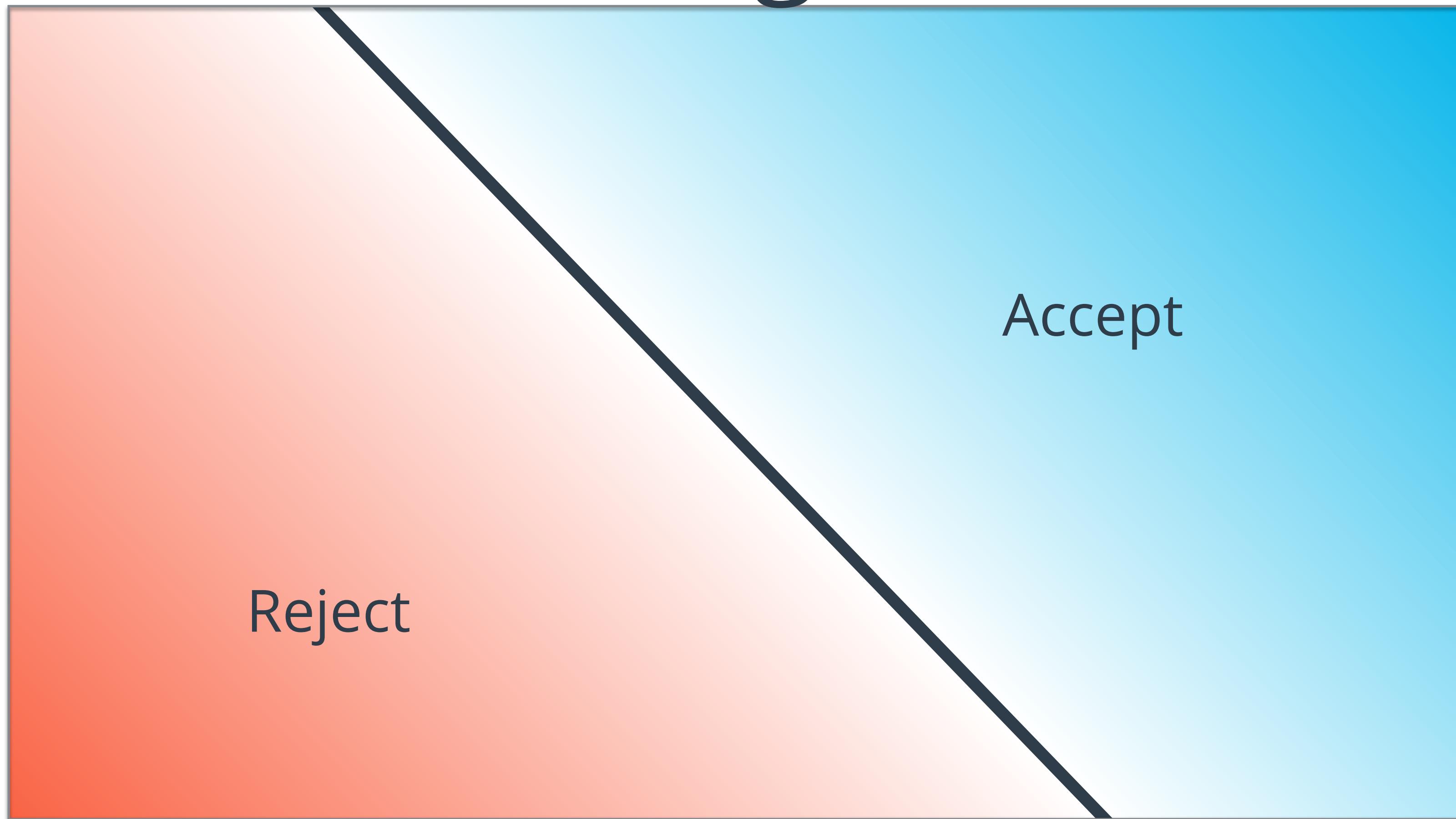
Judge 2

Score = 7*Exam + 2*Grades

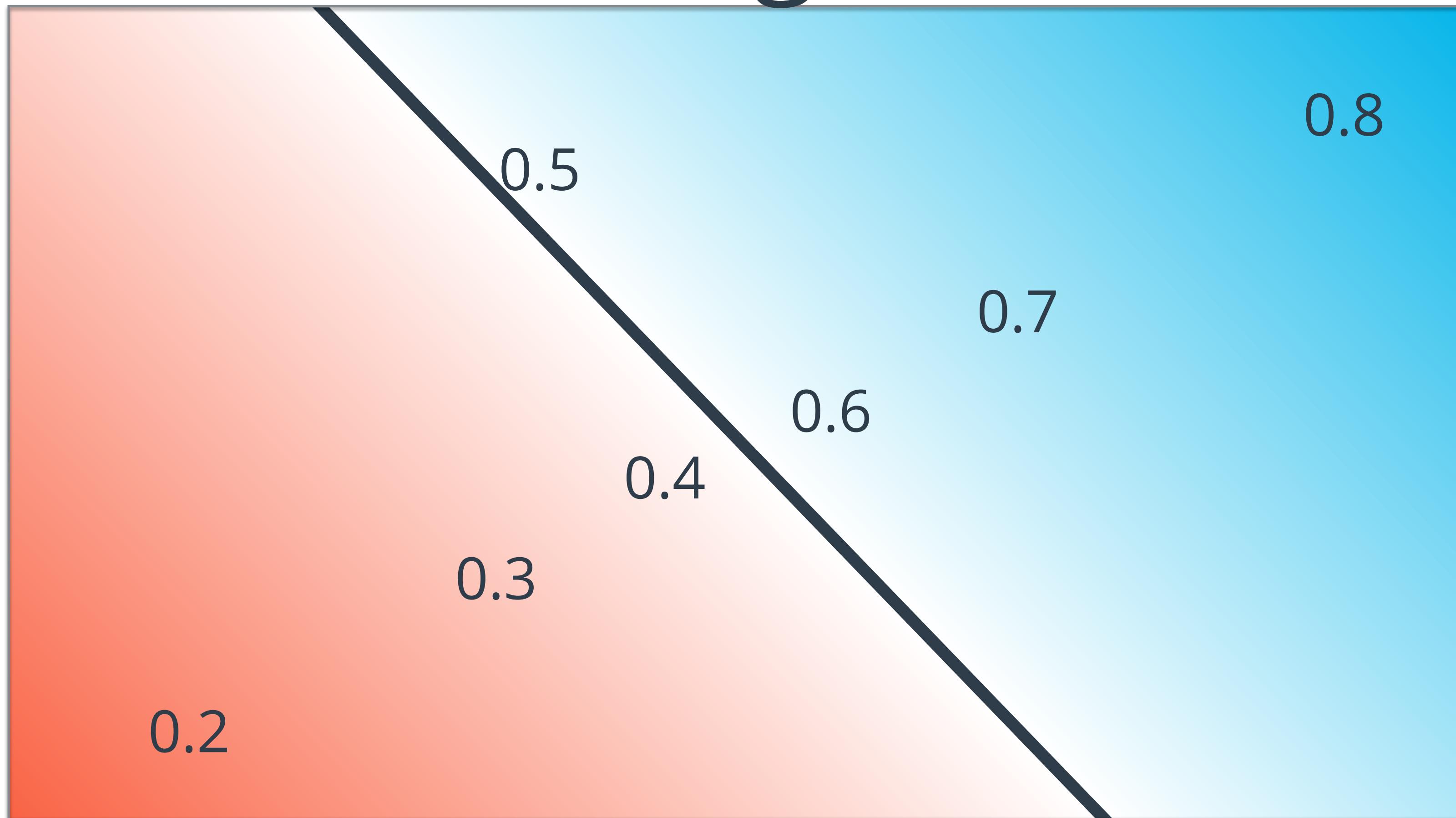
Admissions Office



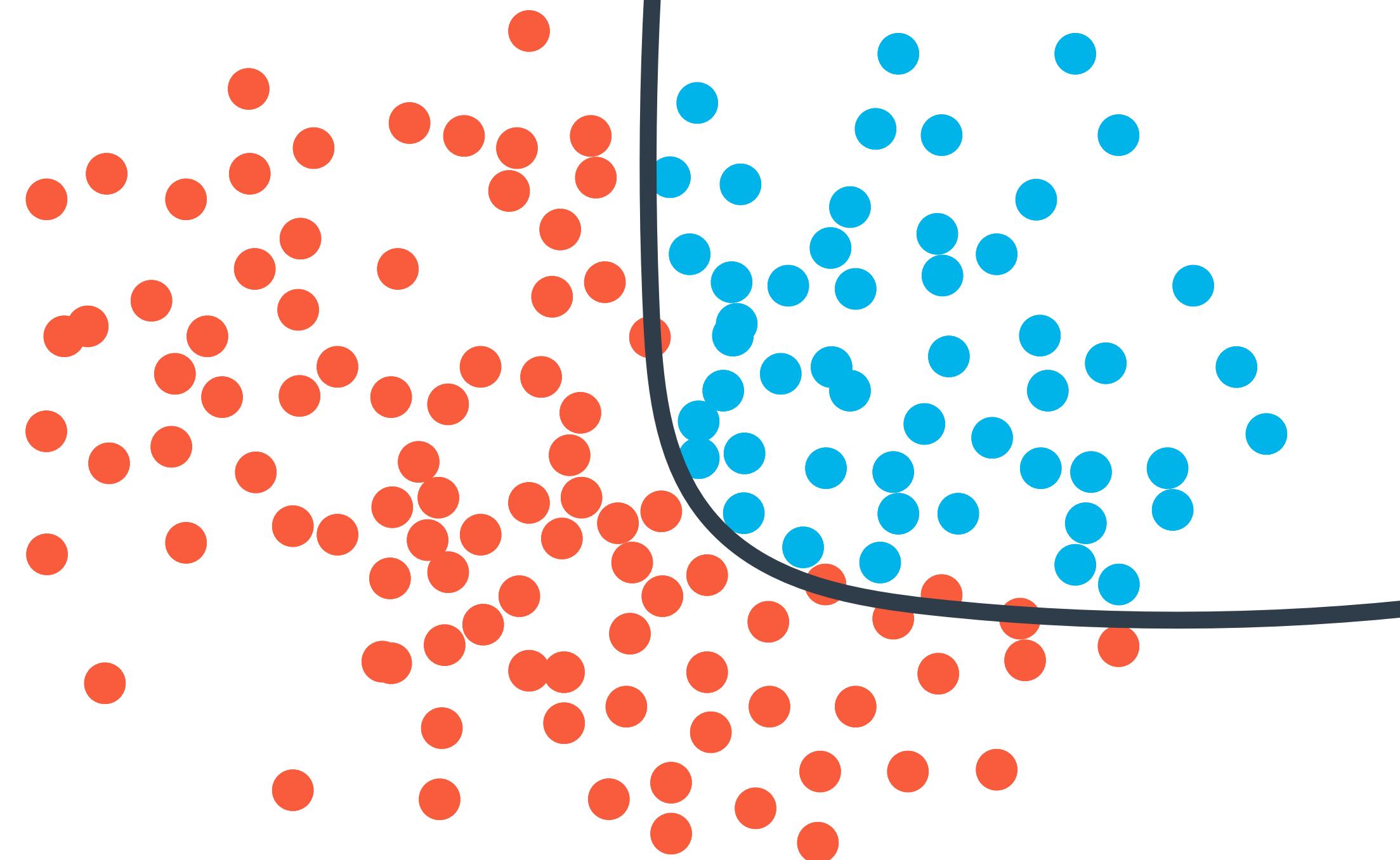
Normalizing the Score



Normalizing the Score



Admissions office

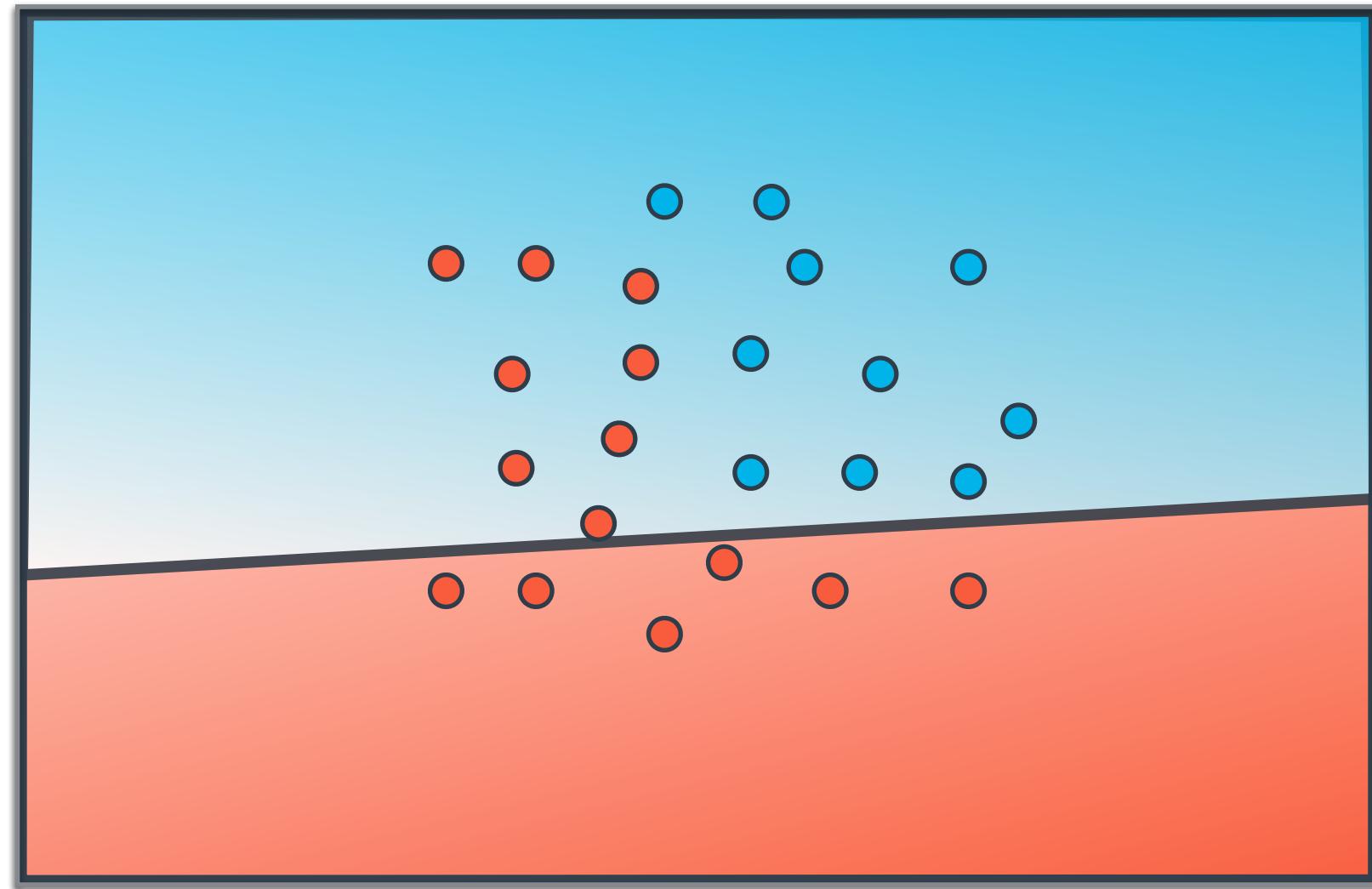


President

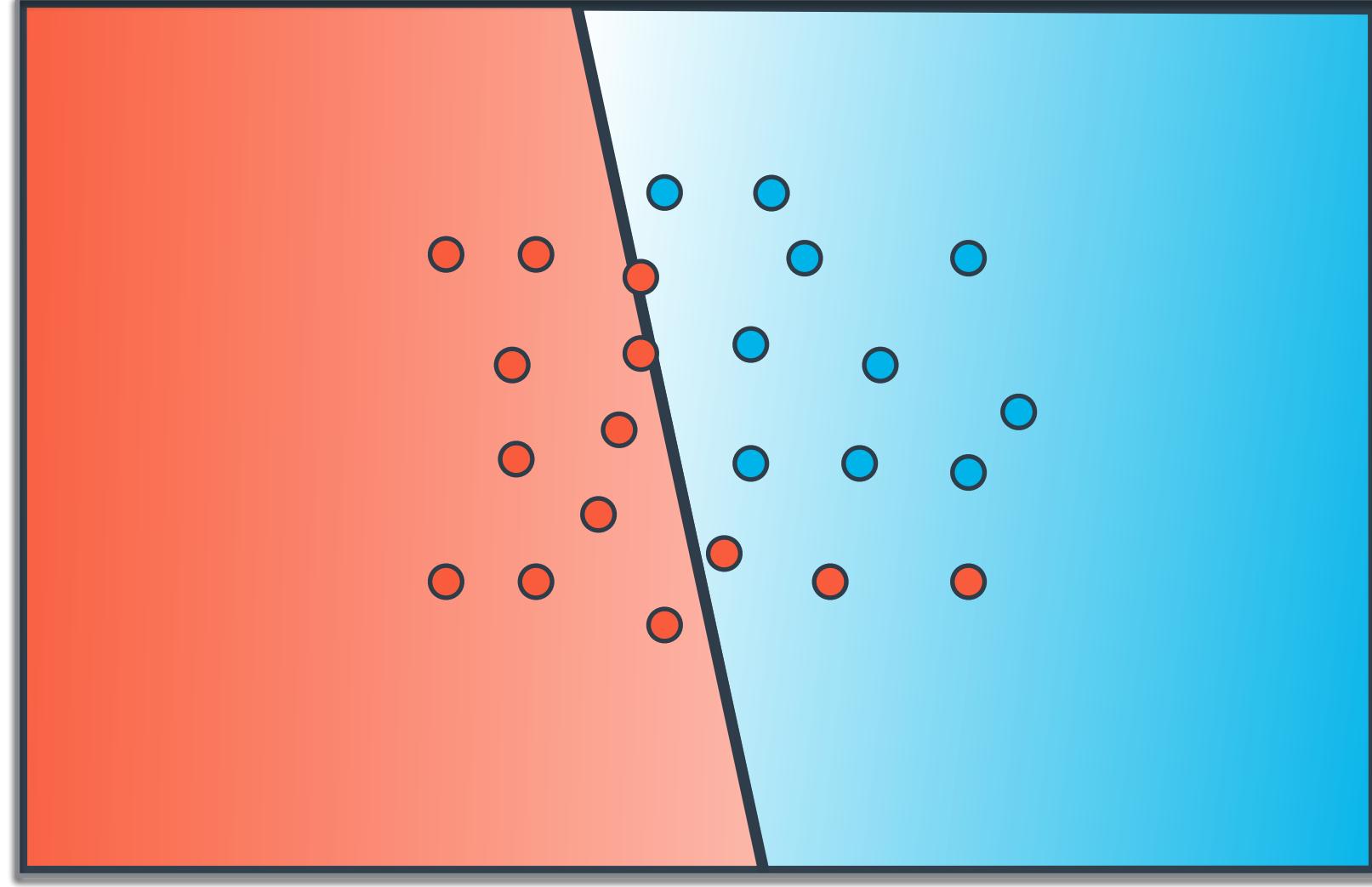
Score = 2*(Judge 1 Score) + 3*(Judge 2 Score)



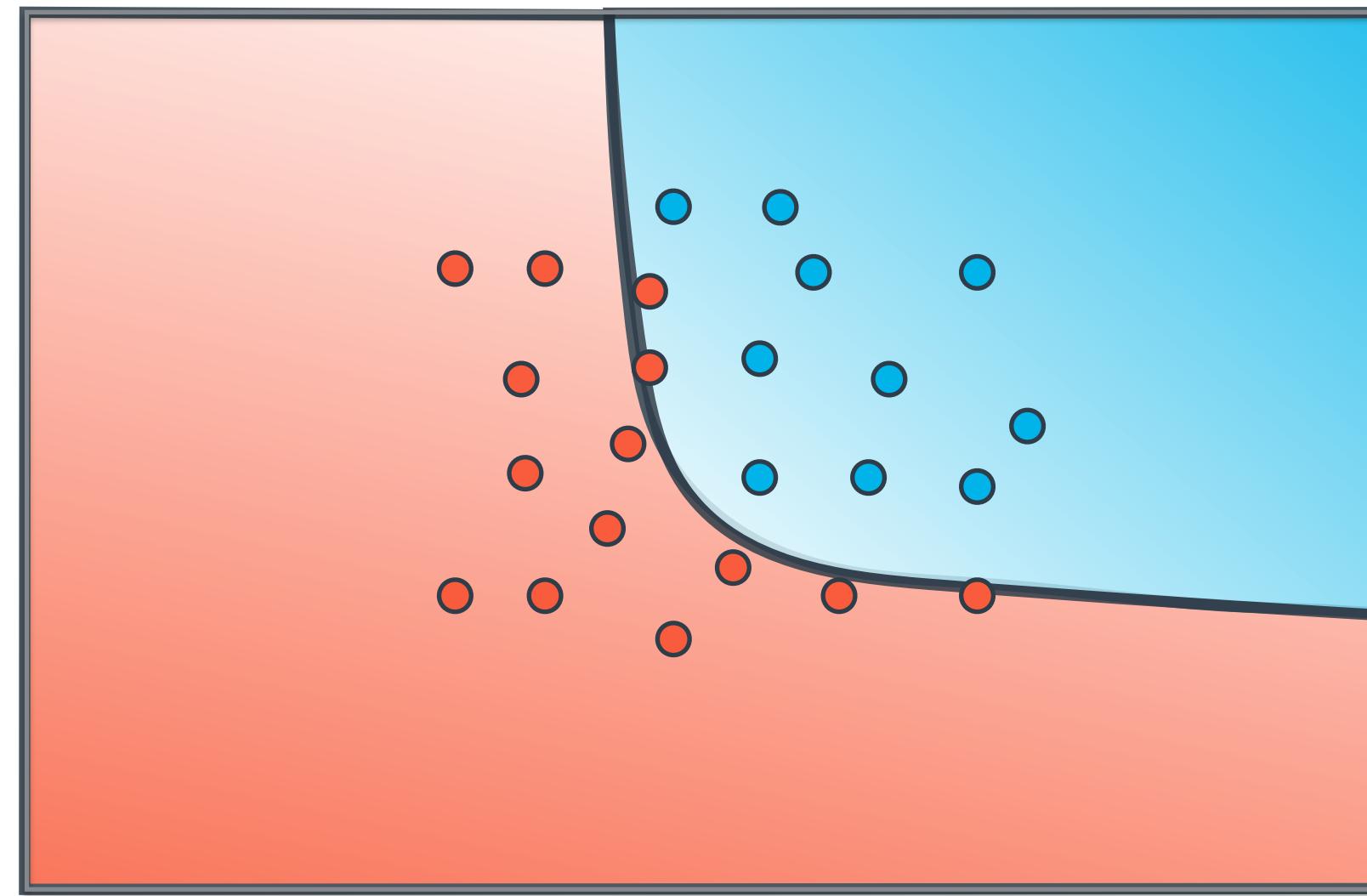
Judge 1



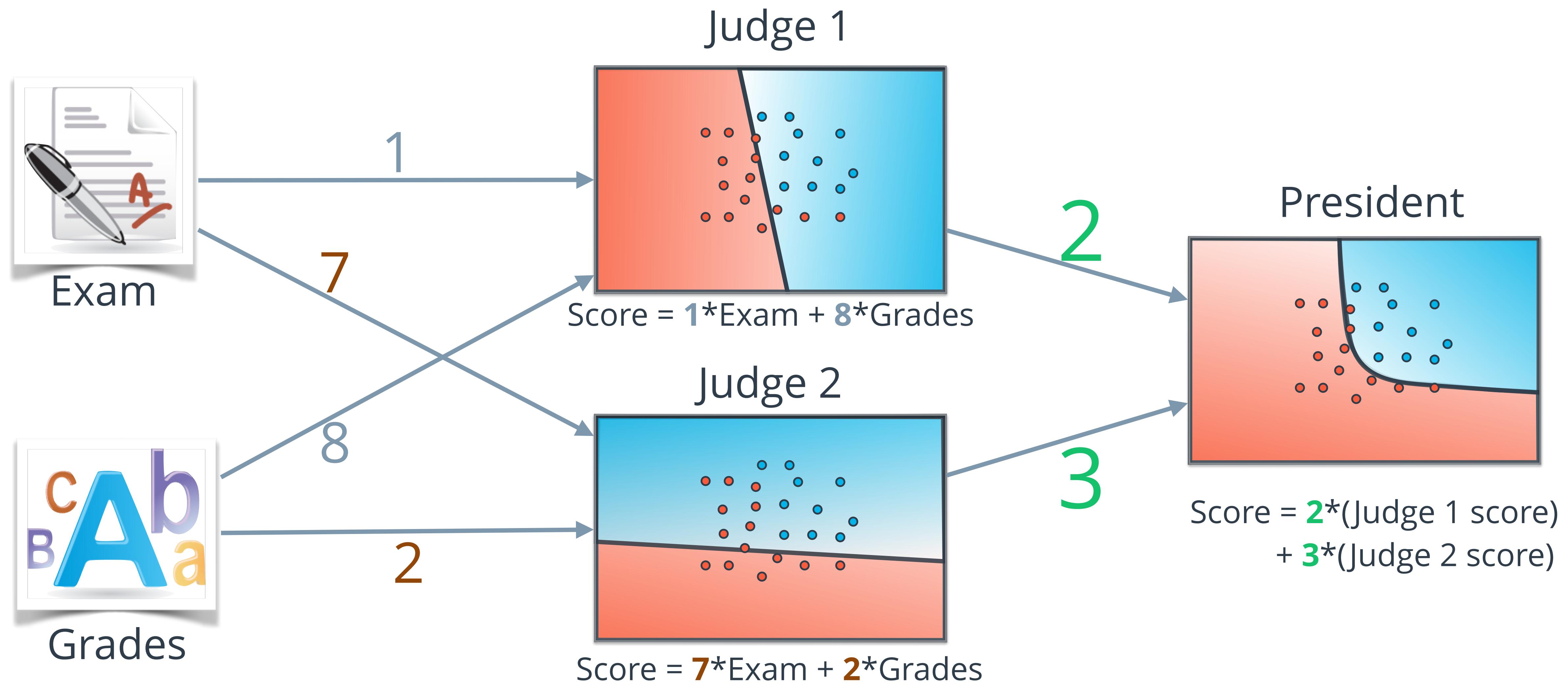
Judge 2



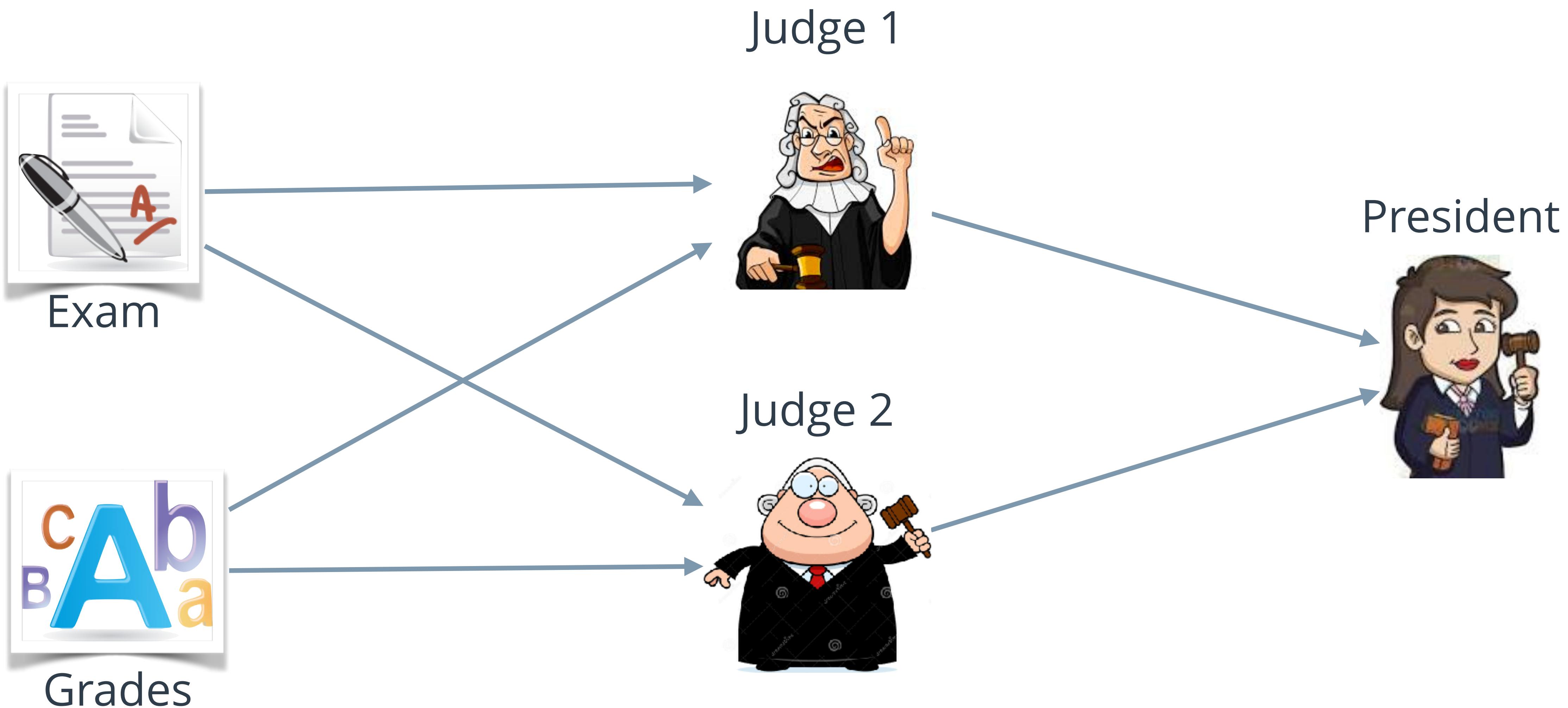
President



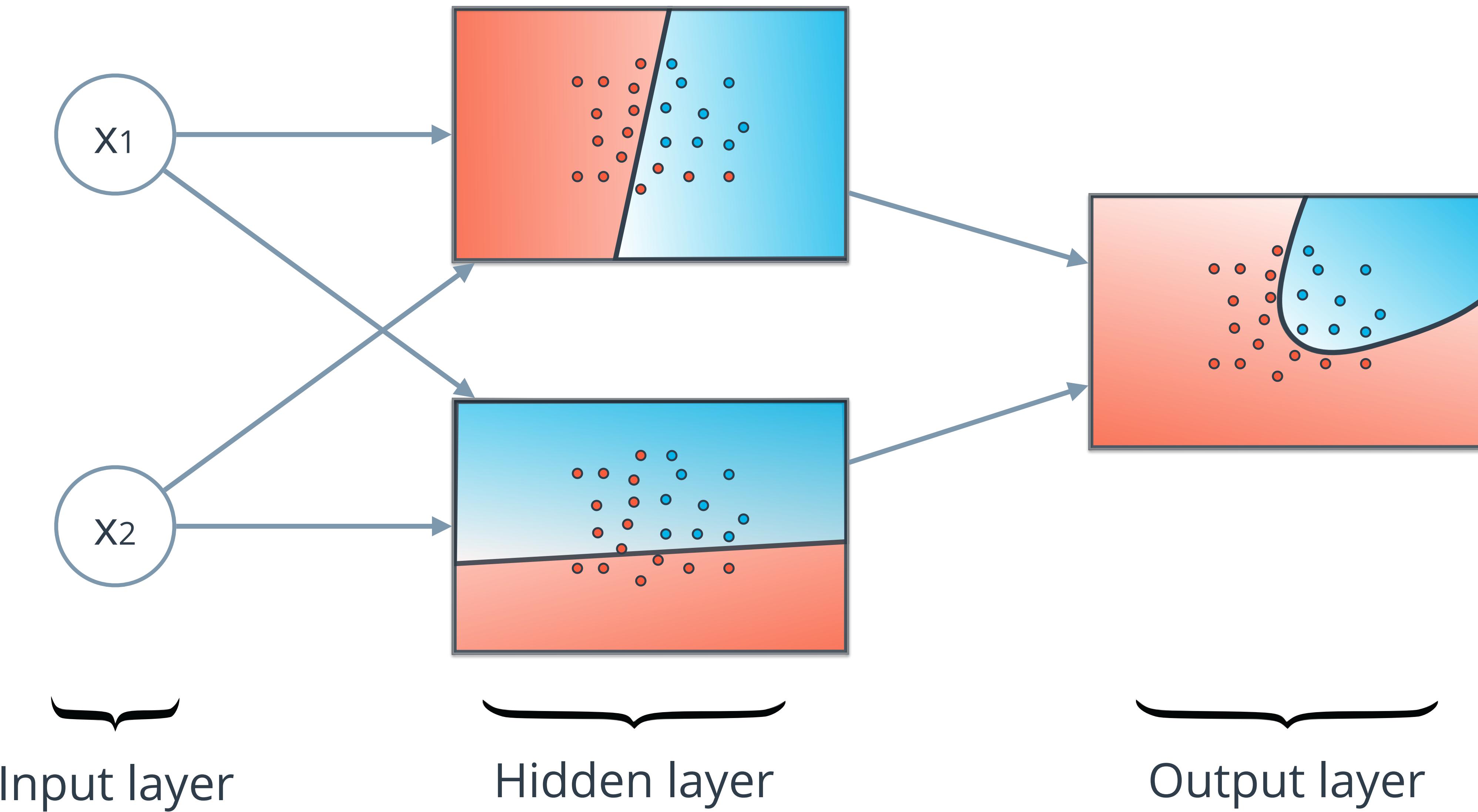
Neural Network

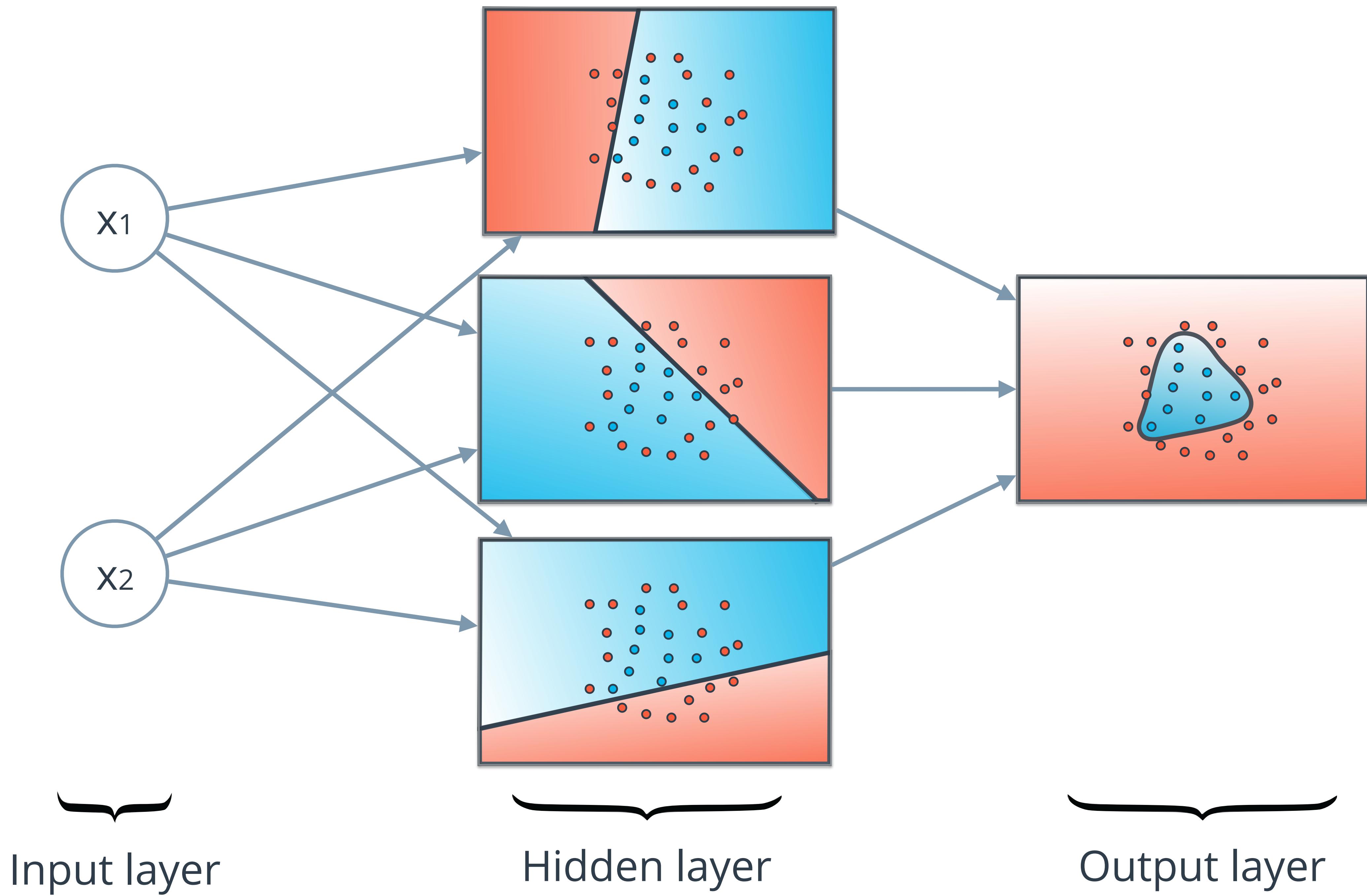


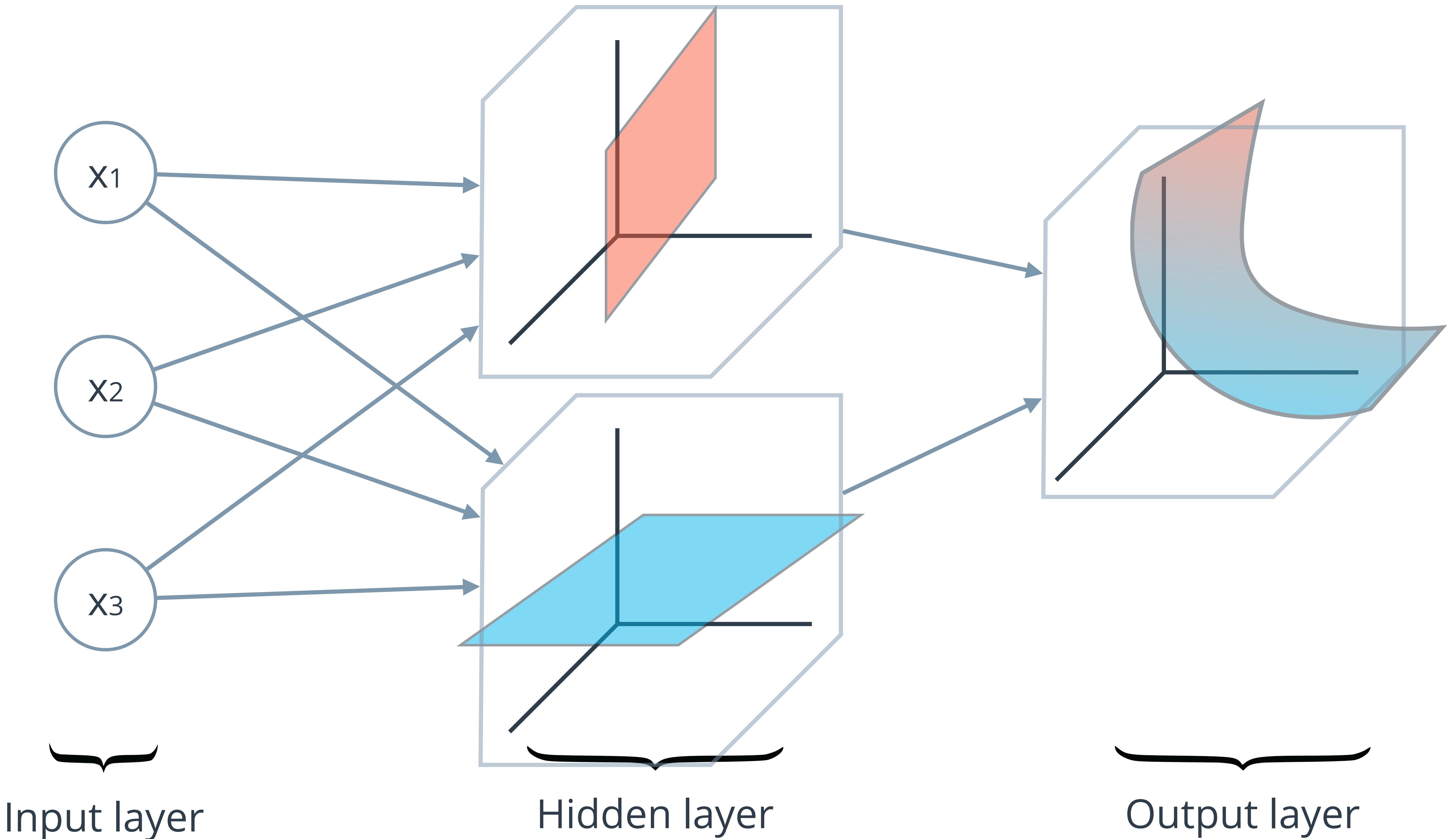
Neural Network

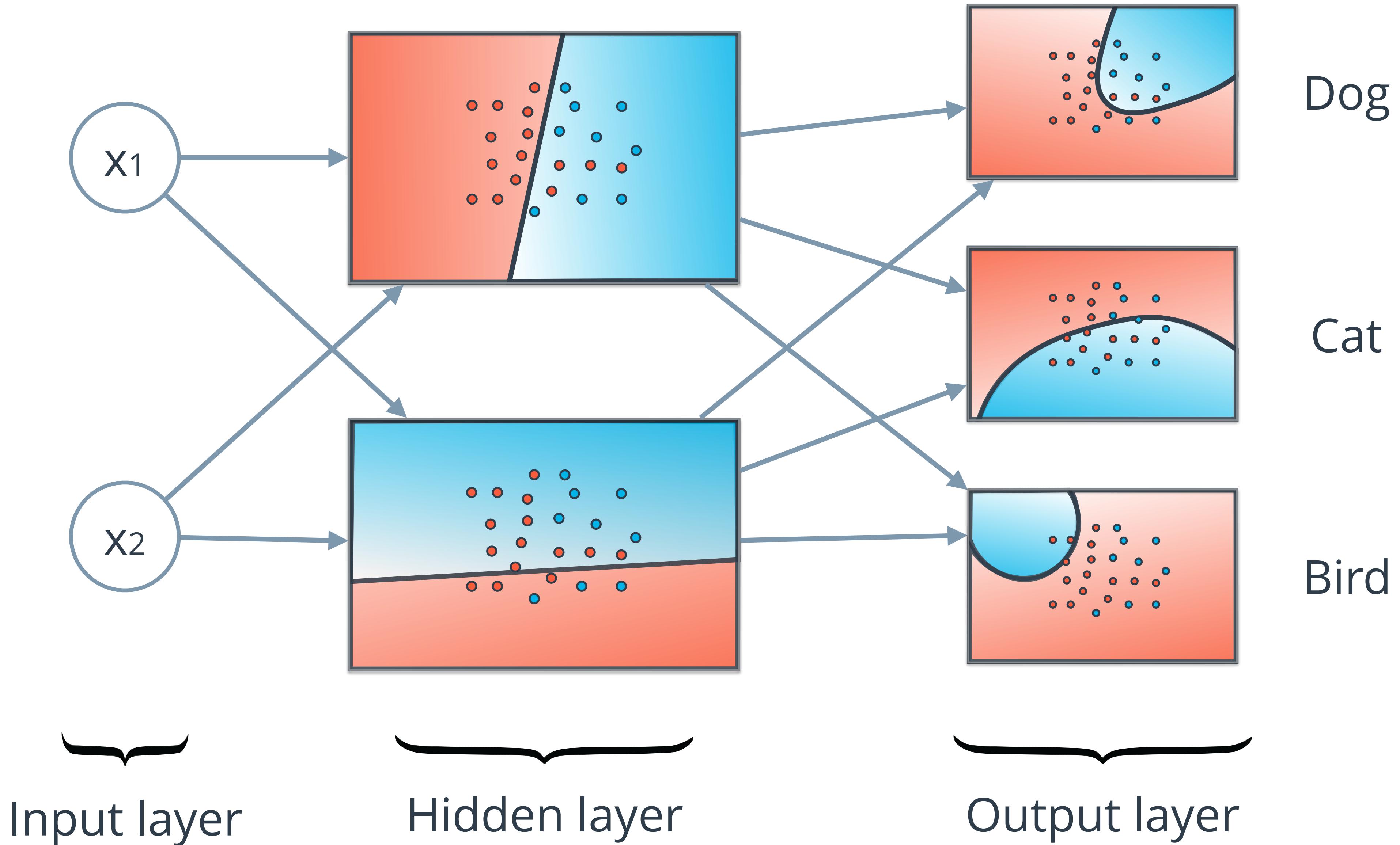


Neural Network

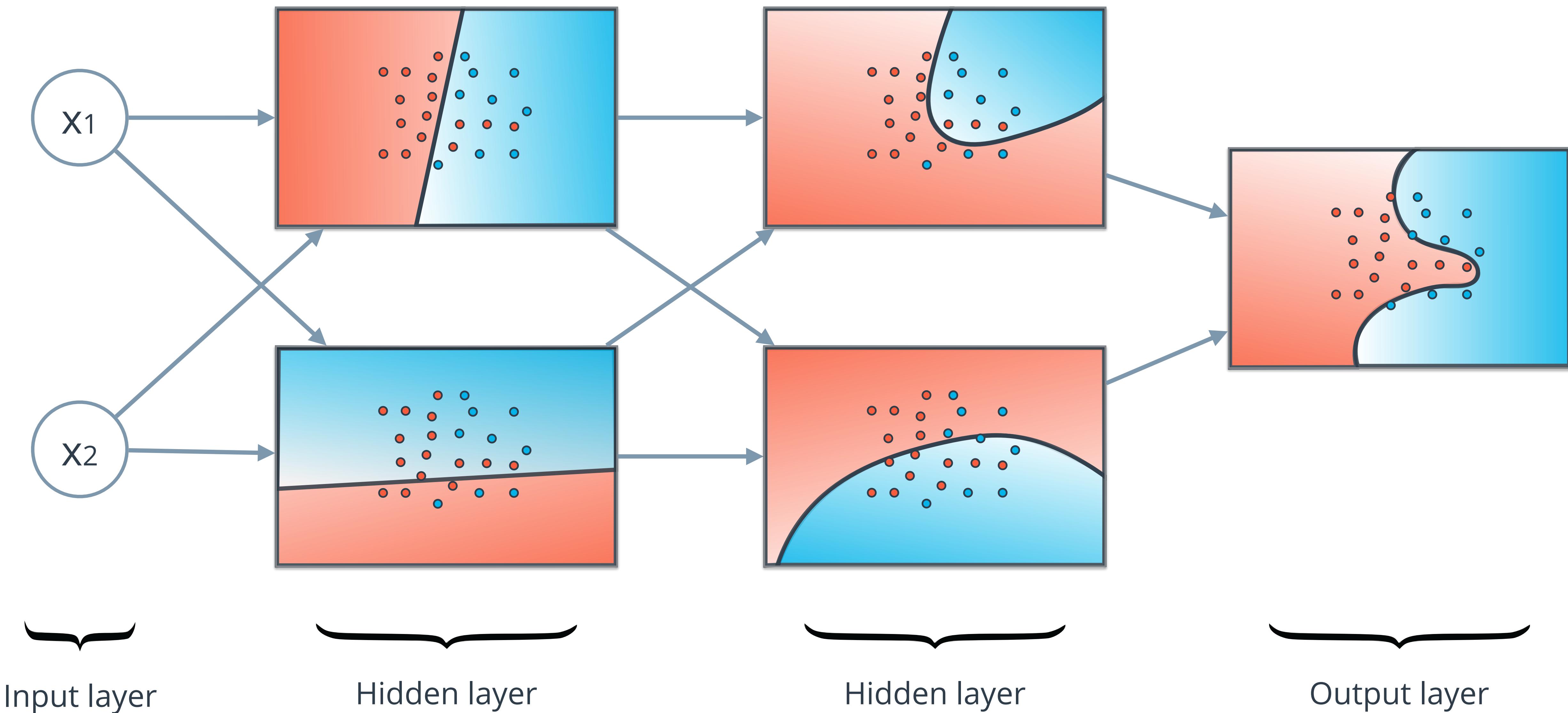




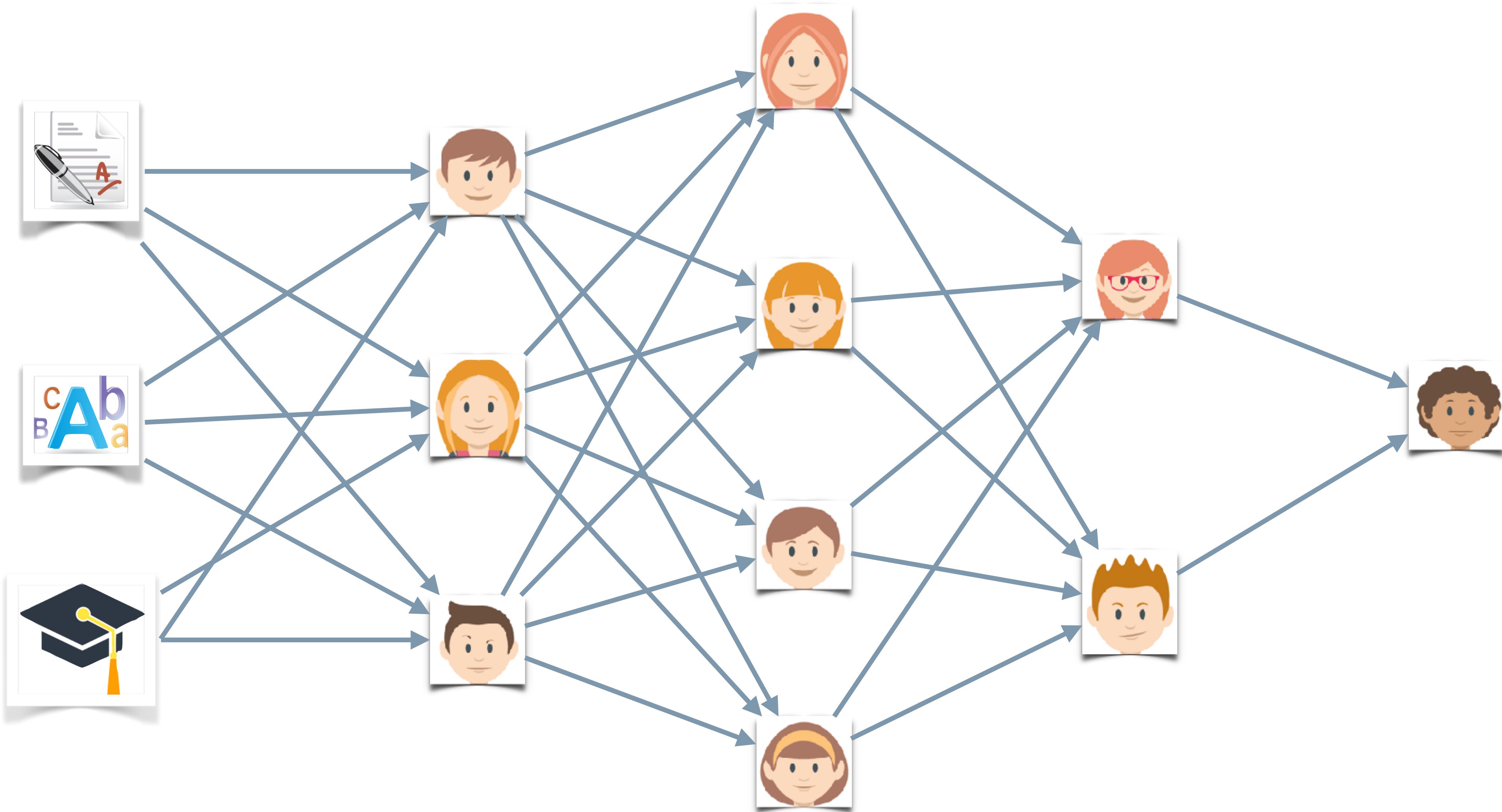




Deep Neural Network



Neural Network



input layer

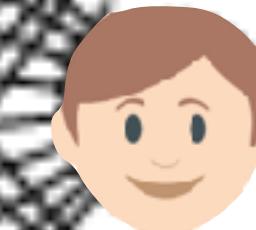
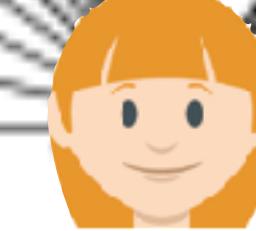
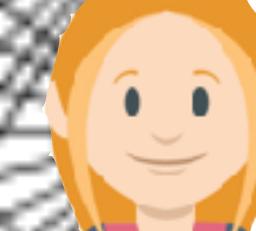
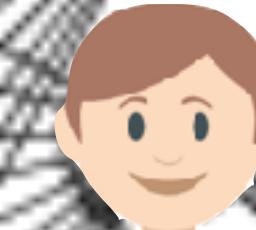
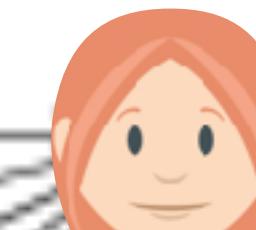
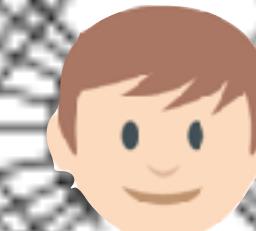
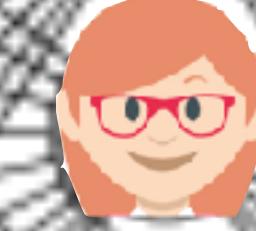
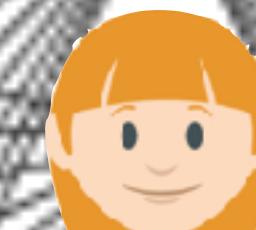
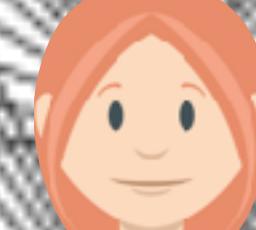
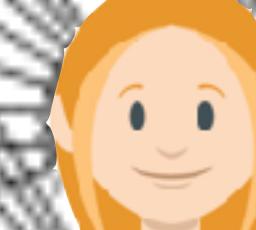
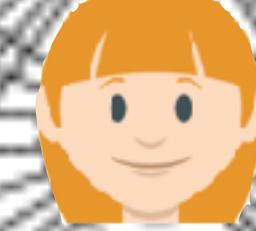
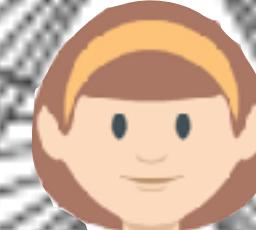
hidden layer 1 hidden layer 2 hidden layer 3

output layer

c
A
B
a



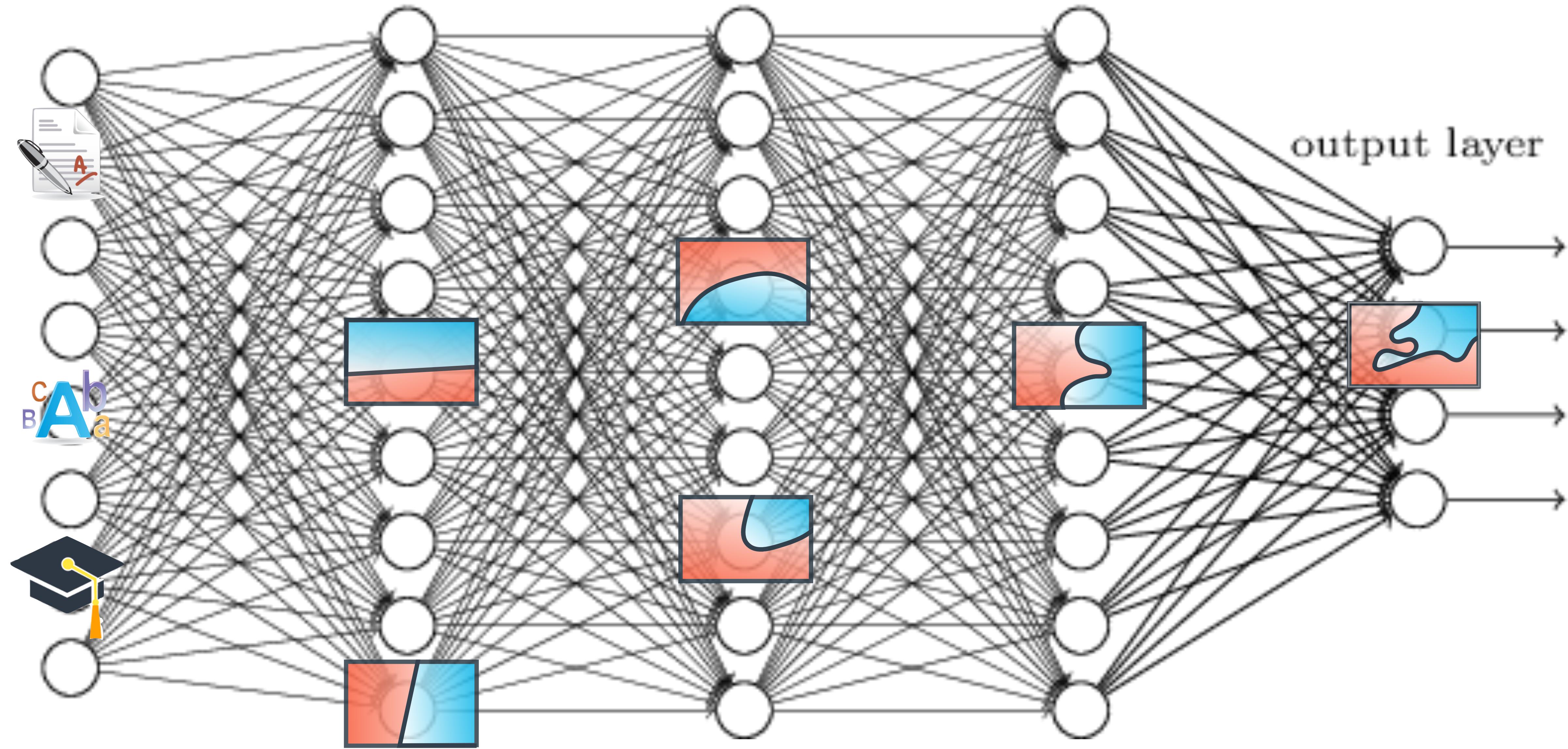
C
A
B
a



input layer

hidden layer 1 hidden layer 2 hidden layer 3

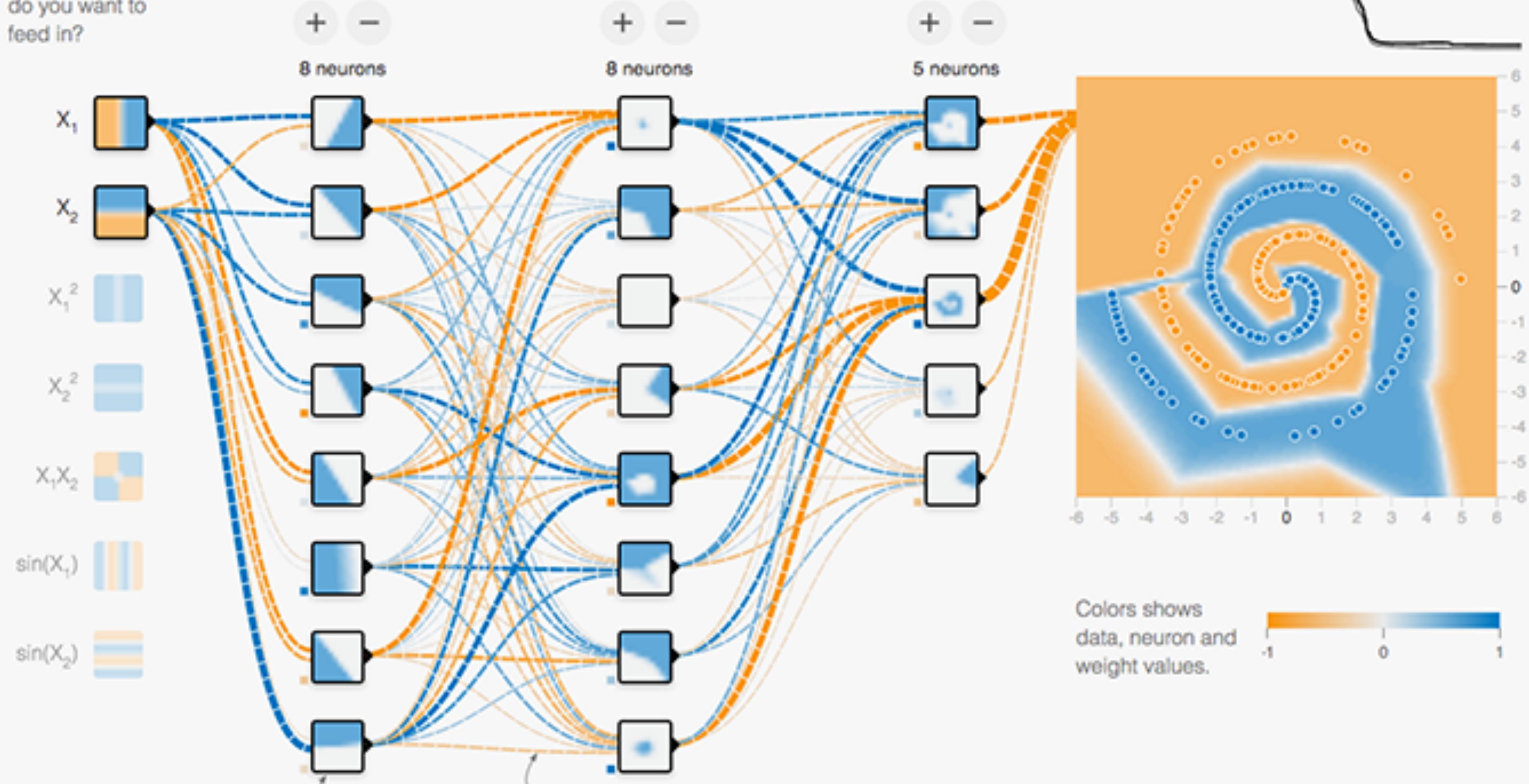
output layer



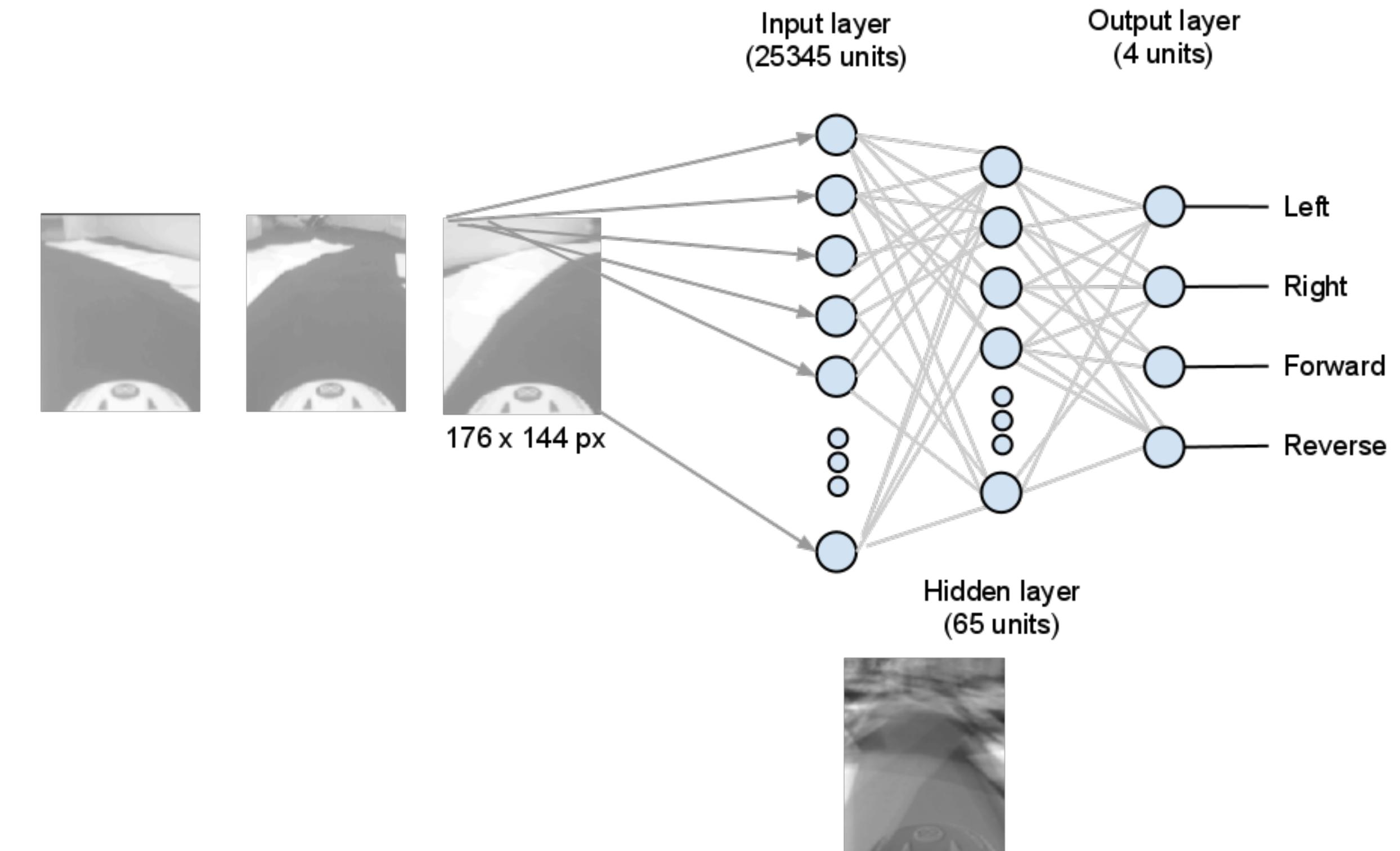
playground.tensorflow.org/

FEATURES

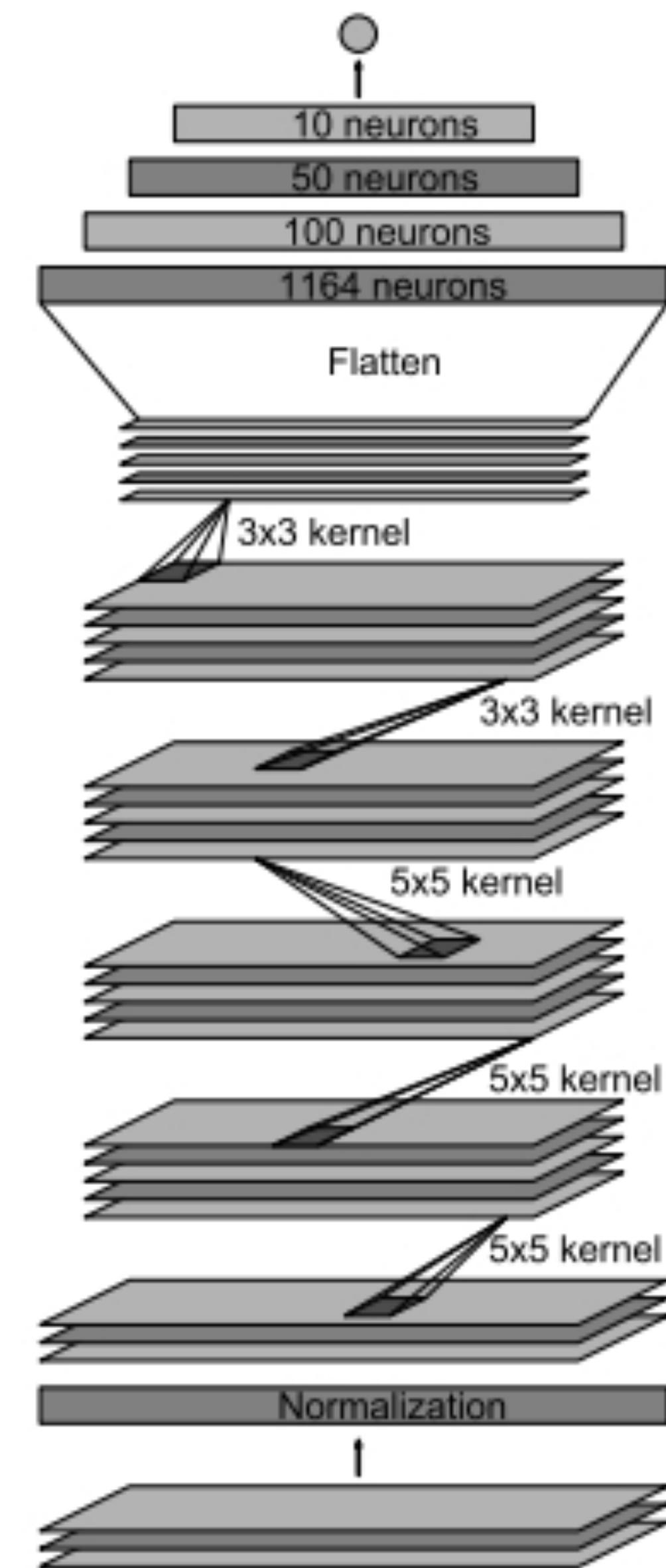
Which properties do you want to feed in?



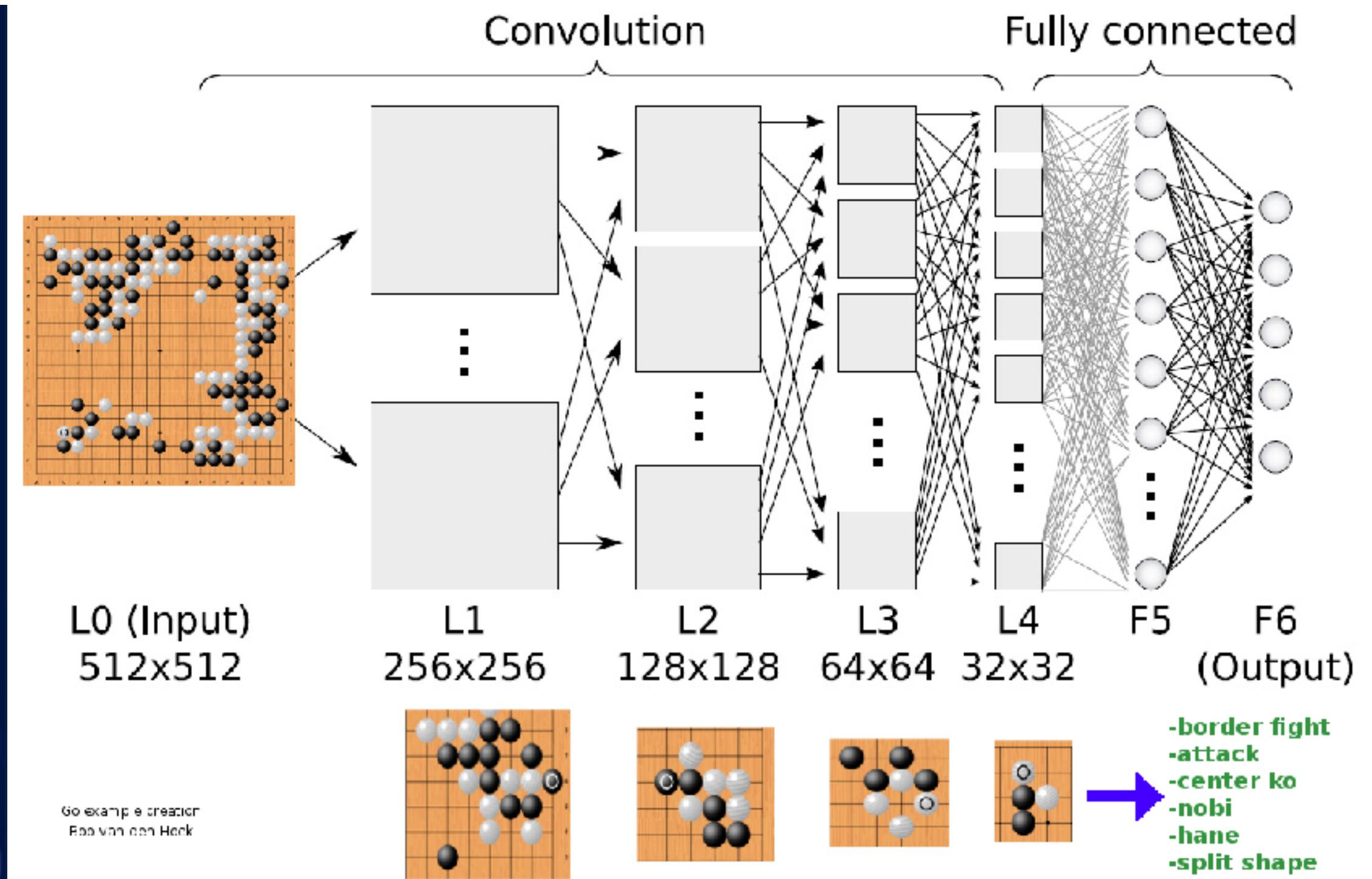
Self Driving Car



Self Driving Car



Playing Go





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!





Sentiment Analysis Lab

Sentiment Analysis

What a great movie!

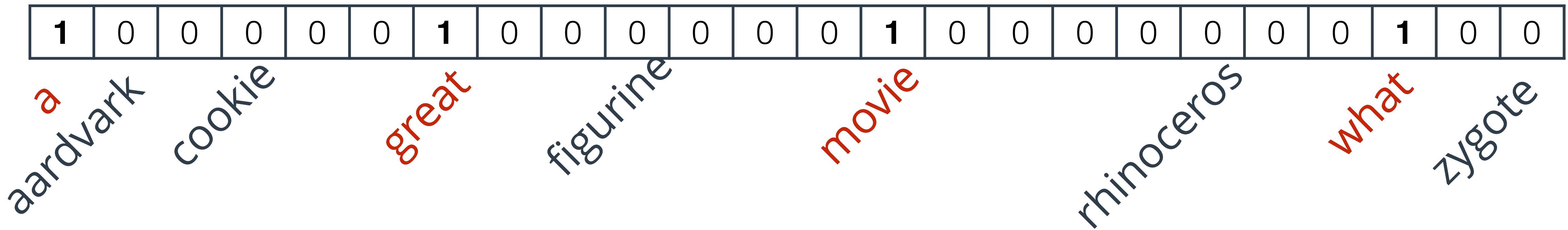


That was terrible.



One-hot encoding

What a great movie!



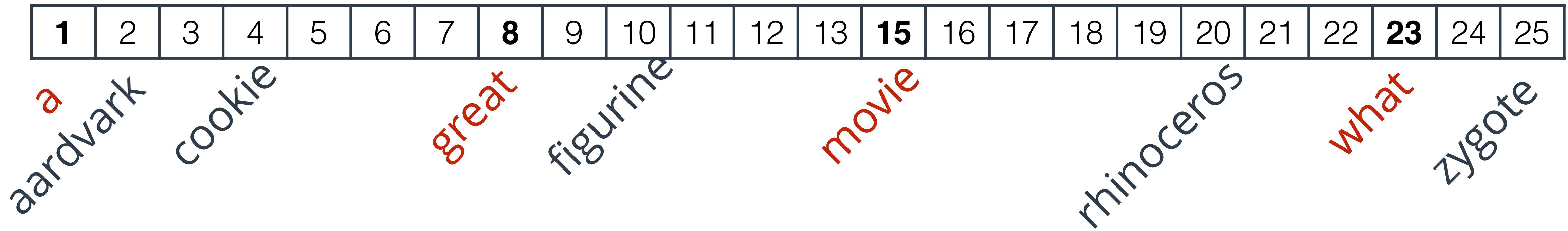
1. Loading the data

```
# 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

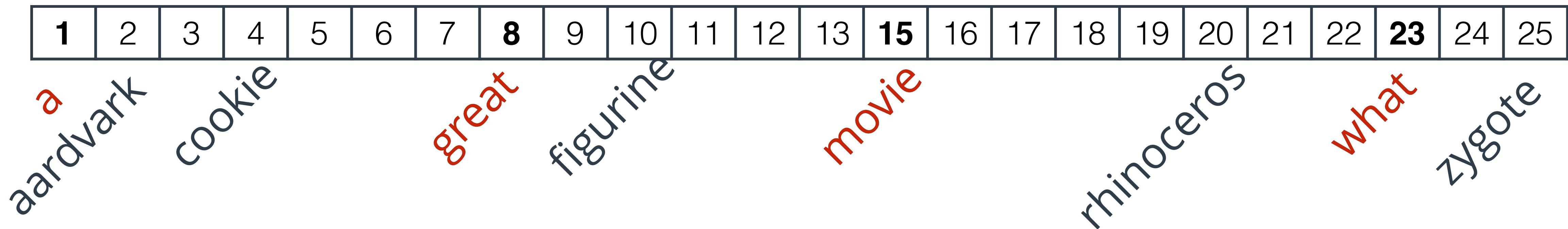
What a great movie!



[1, 7, 15, 23]

3. One-hot encoding the input

What a great movie!



[1, 7, 15, 23]



3. One-hot encoding the output



[1,0]

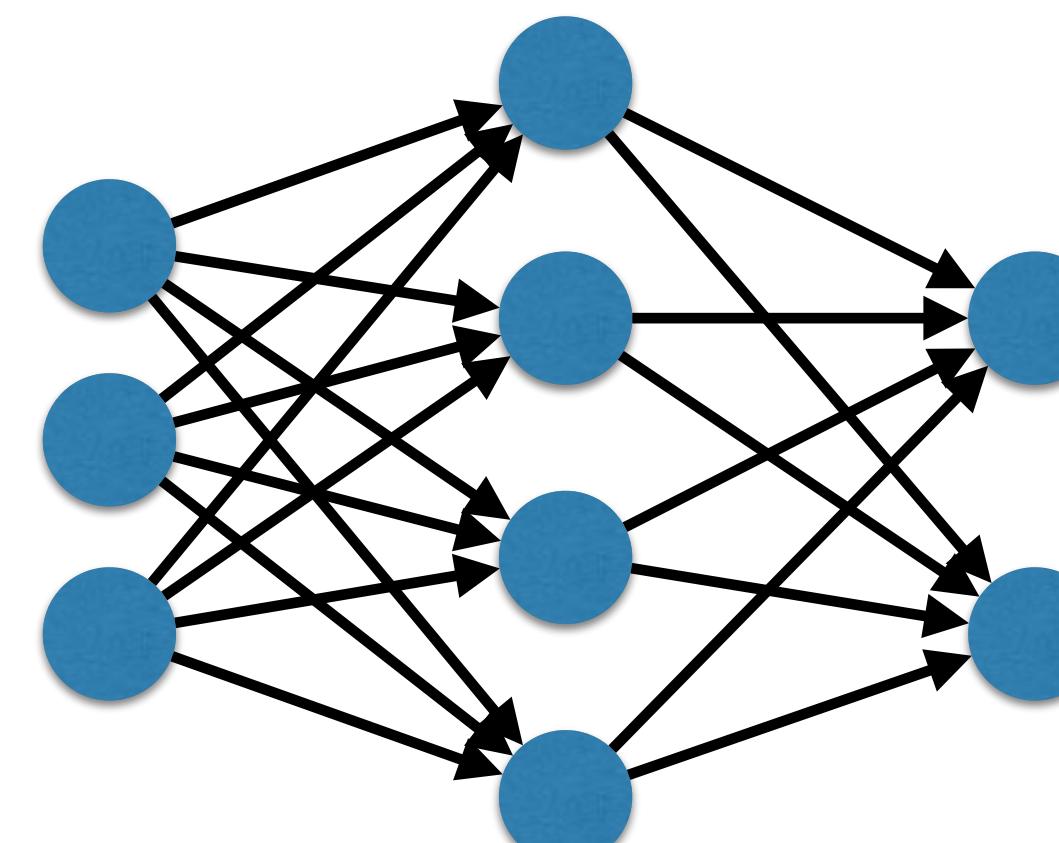


[0,1]

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)
```

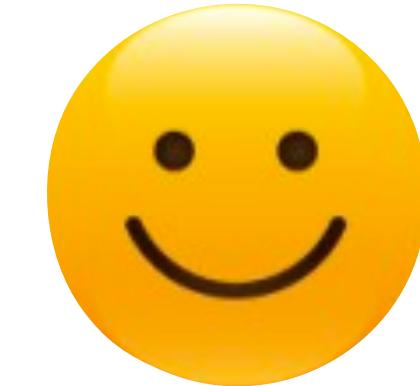
5. Training the Model

```
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	0	1	0	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”

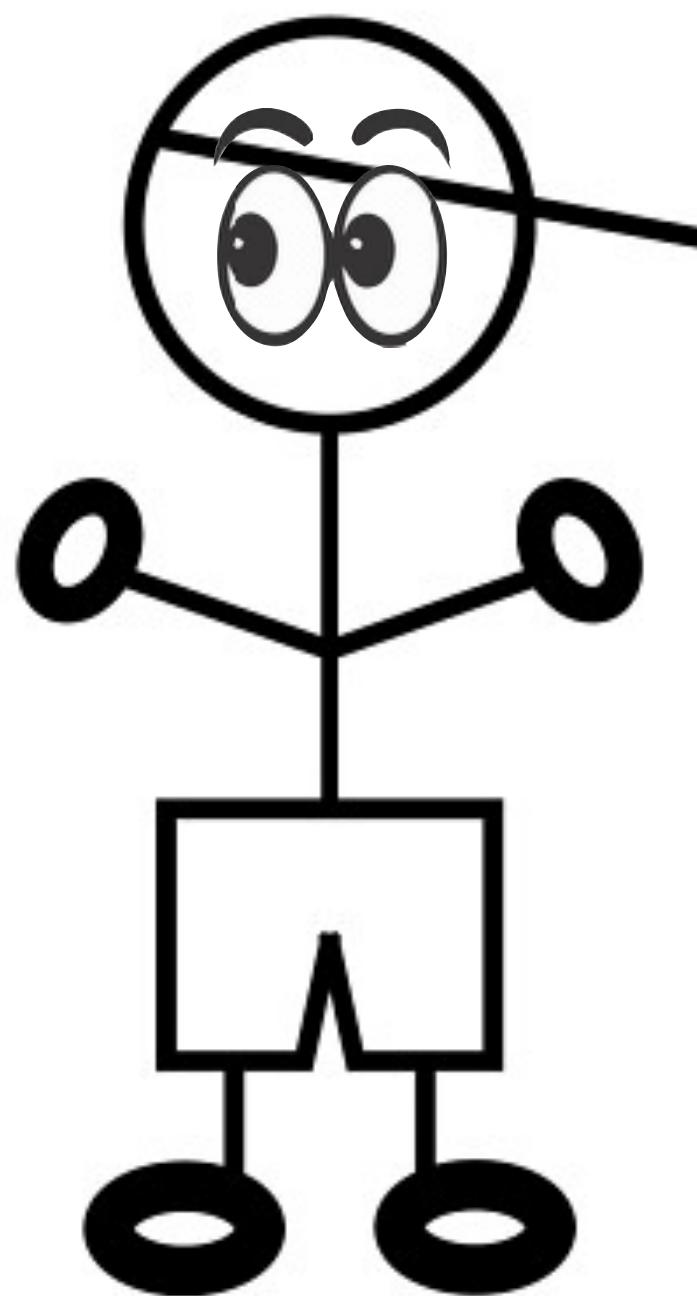
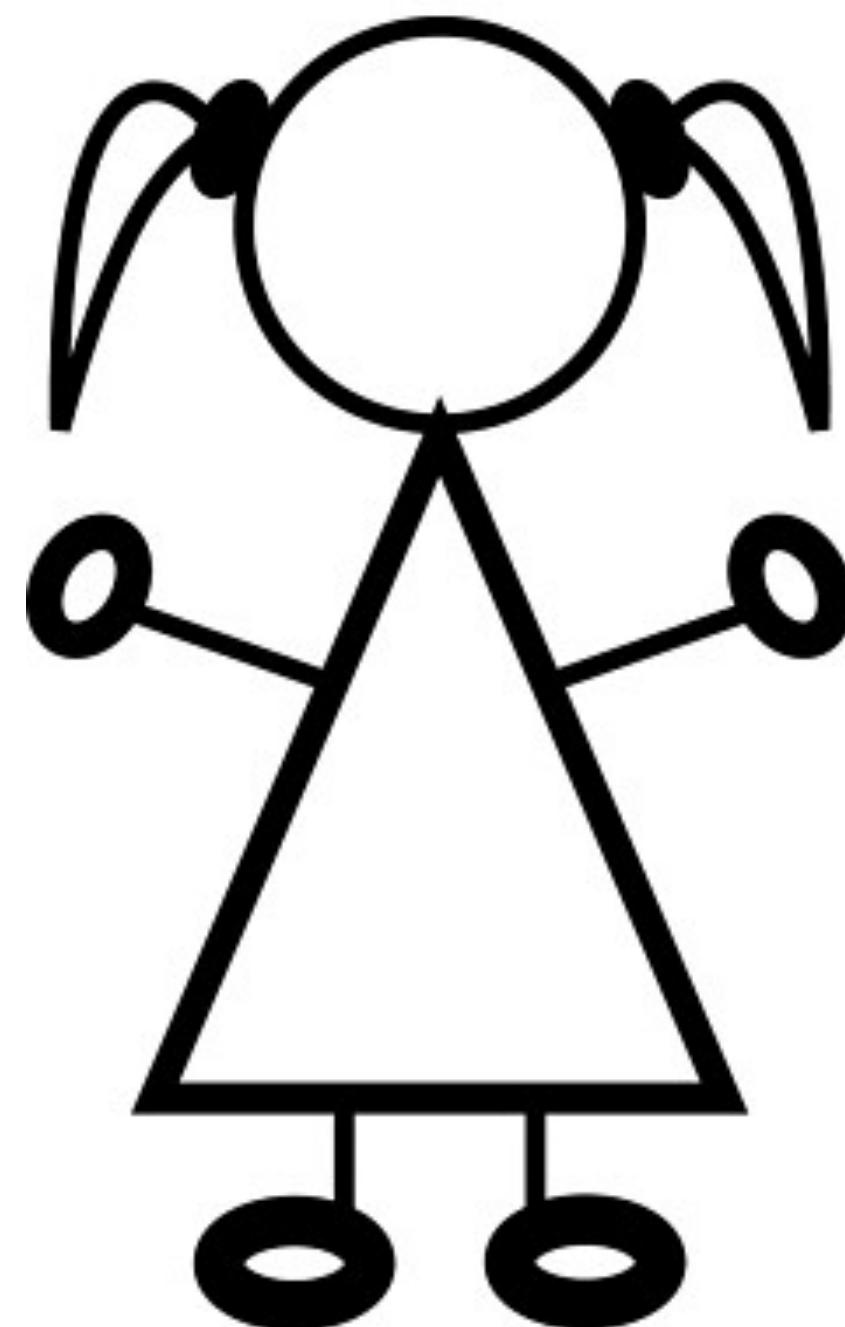


1	0	1	0	0	0	1	0	1	0	0	0	1	0	0	0	1	0	0	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

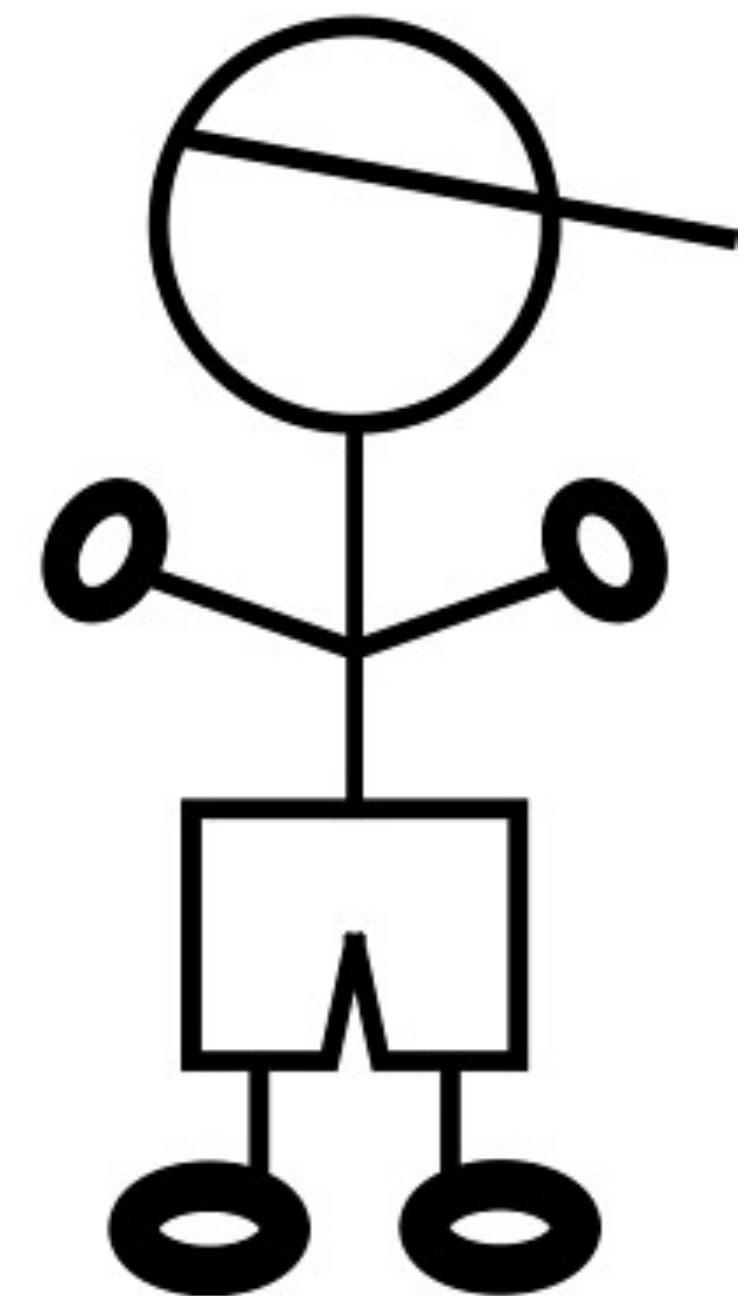
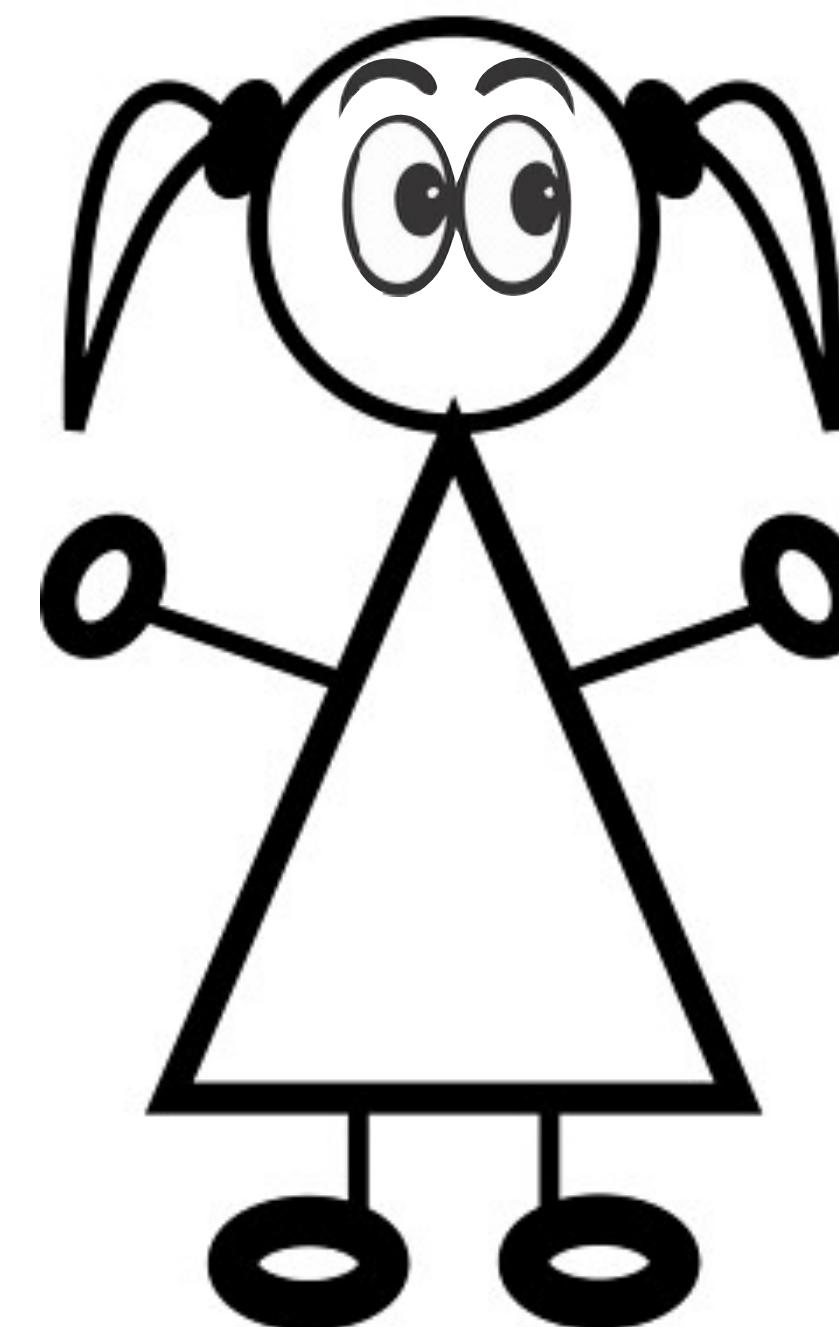
this I expected than be movie much better

Recurrent Neural Networks

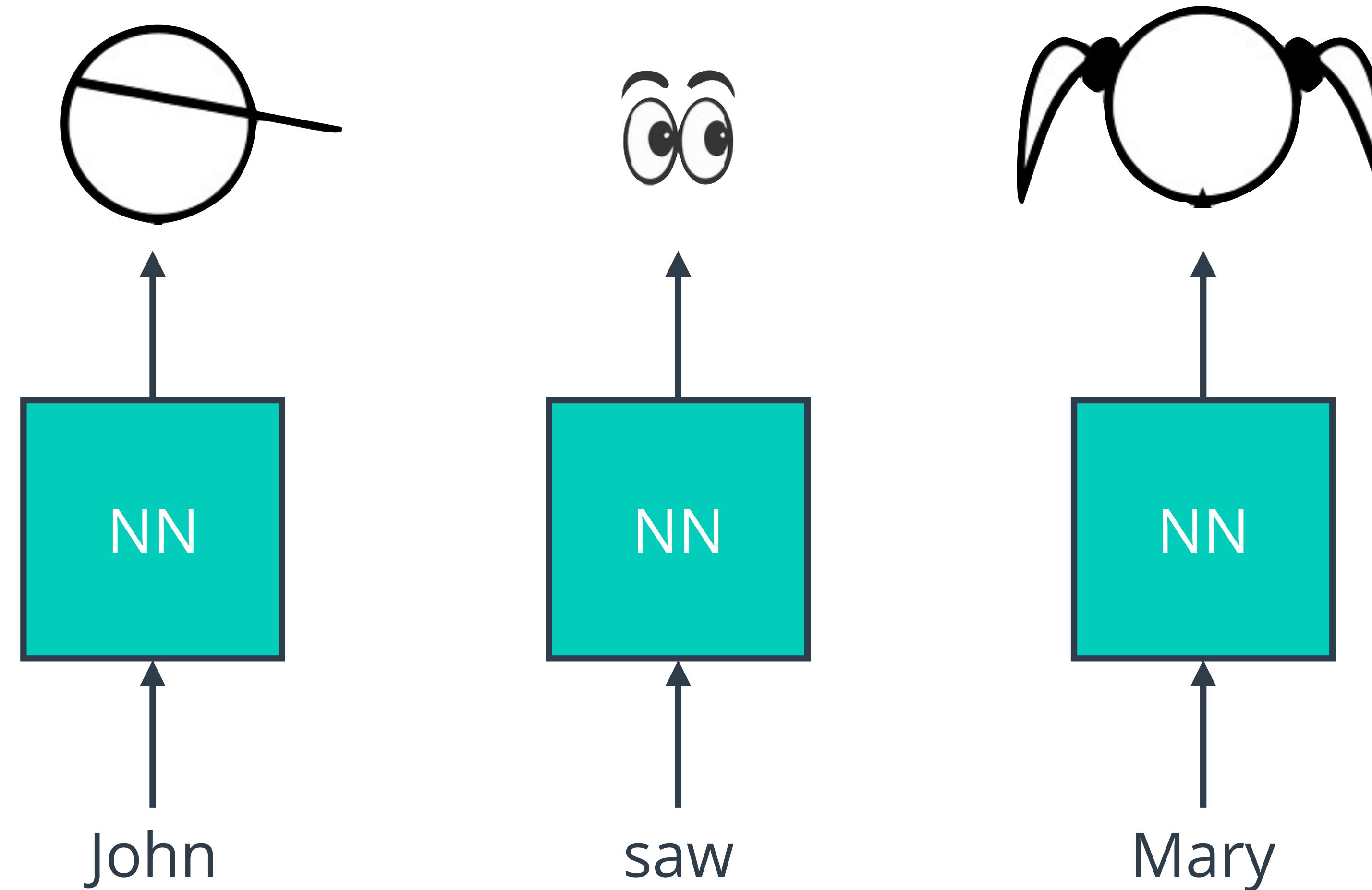
John saw Mary.



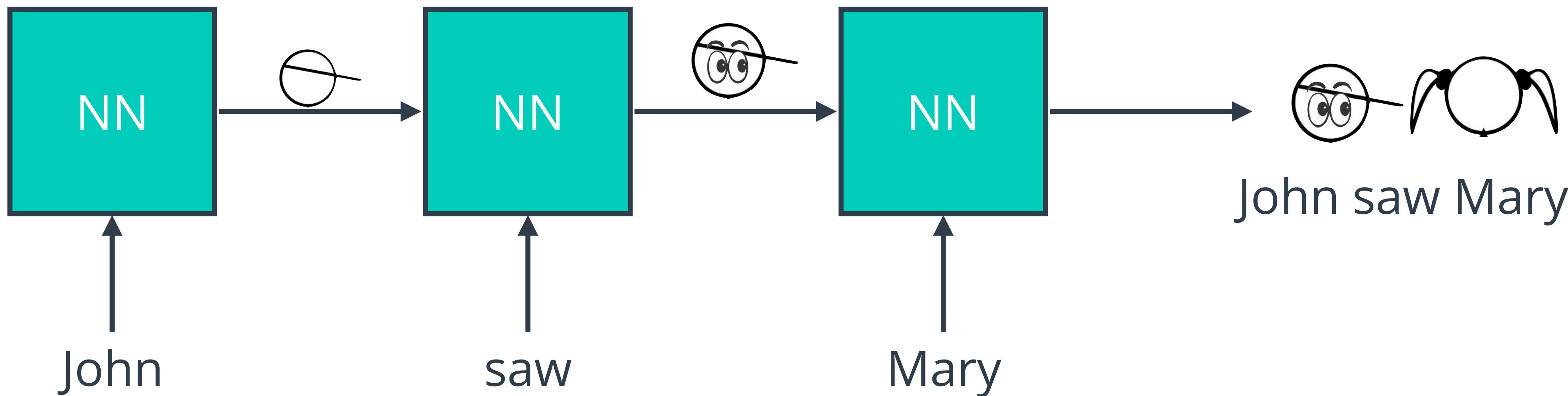
Mary saw John.



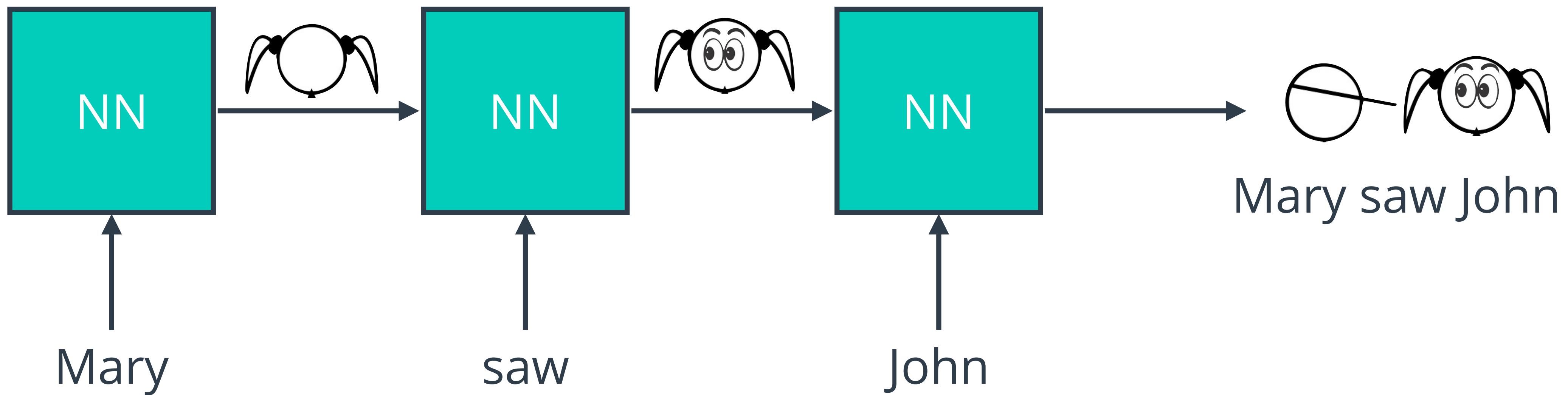
Neural Network



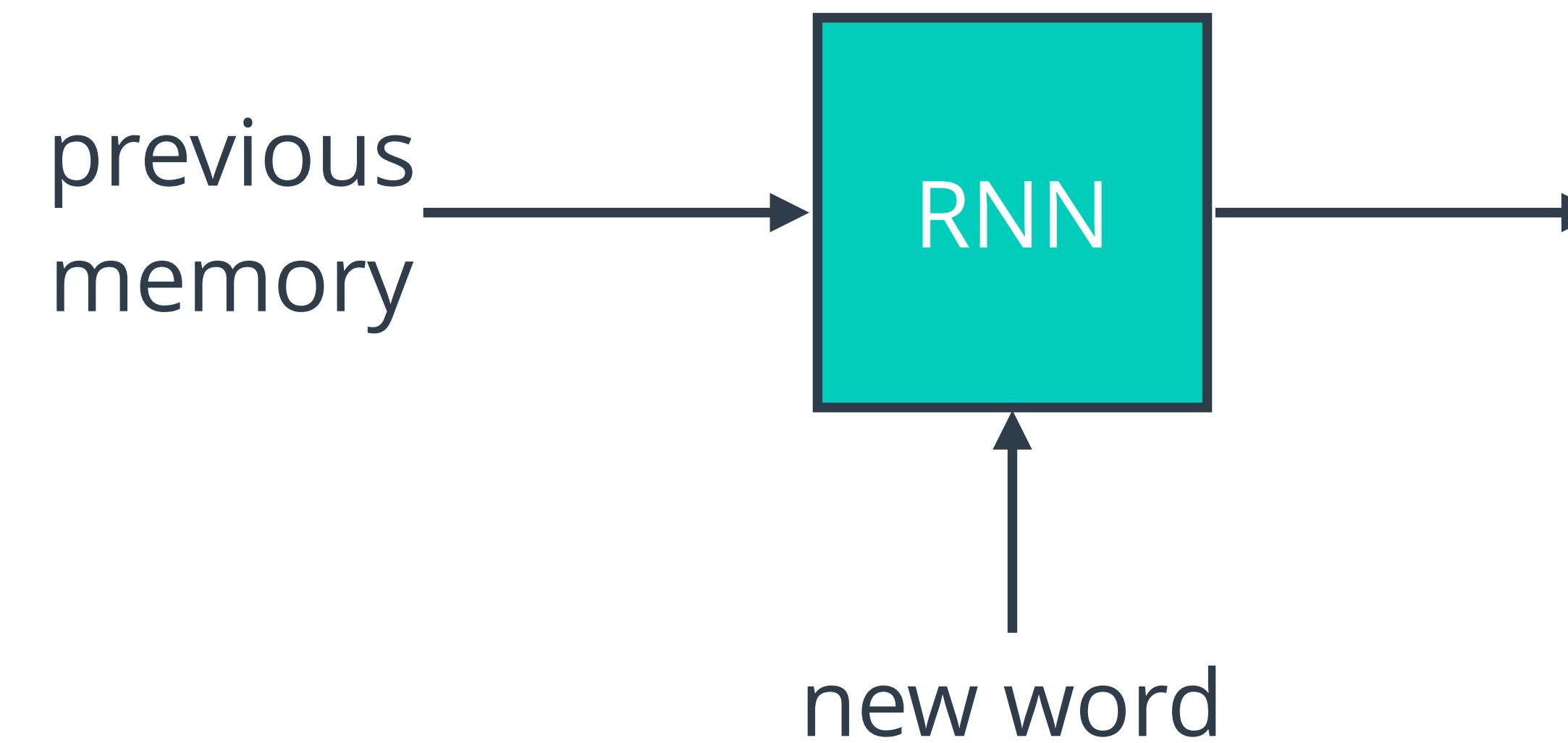
Recurrent Neural Network



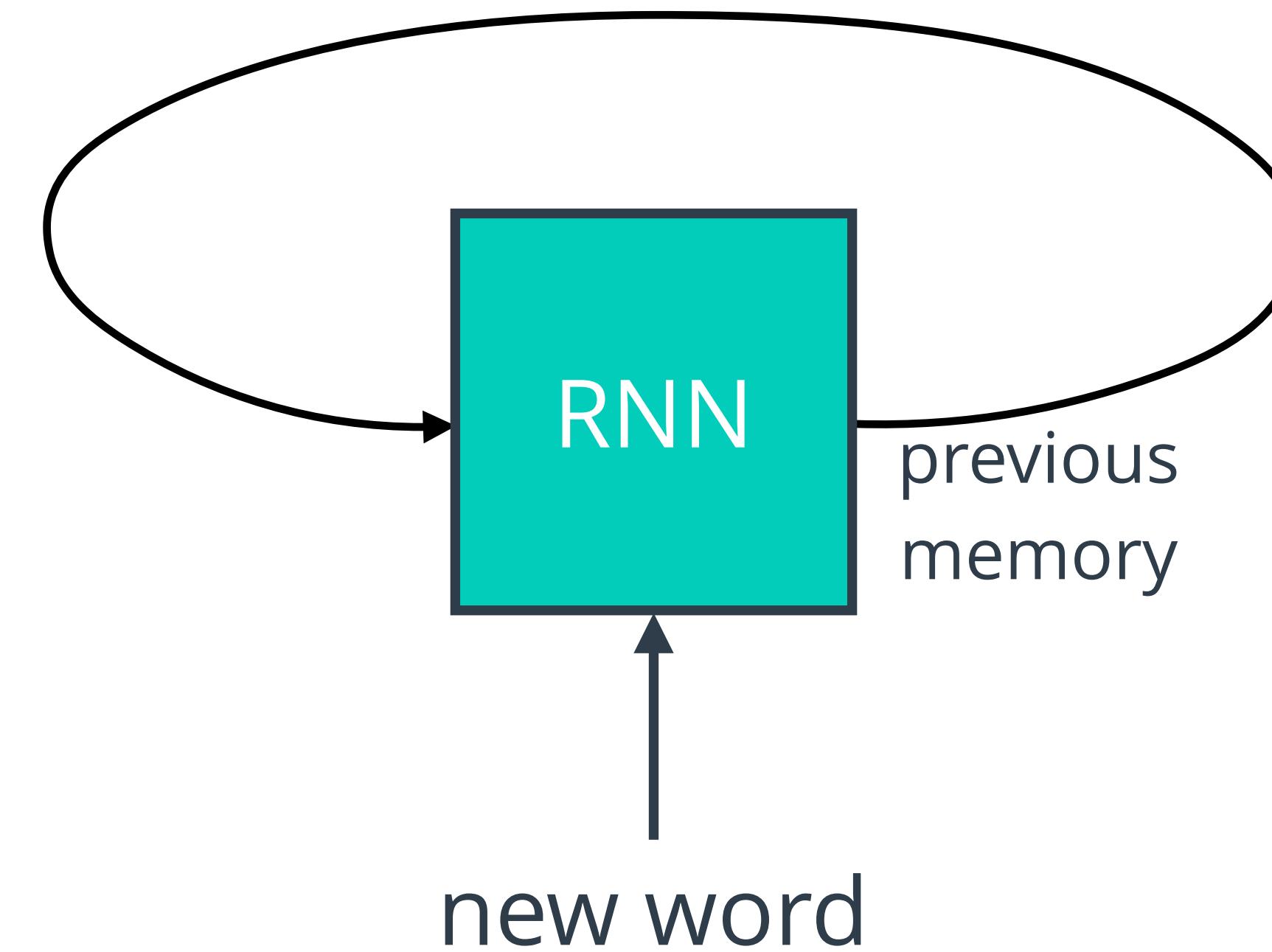
Recurrent Neural Network



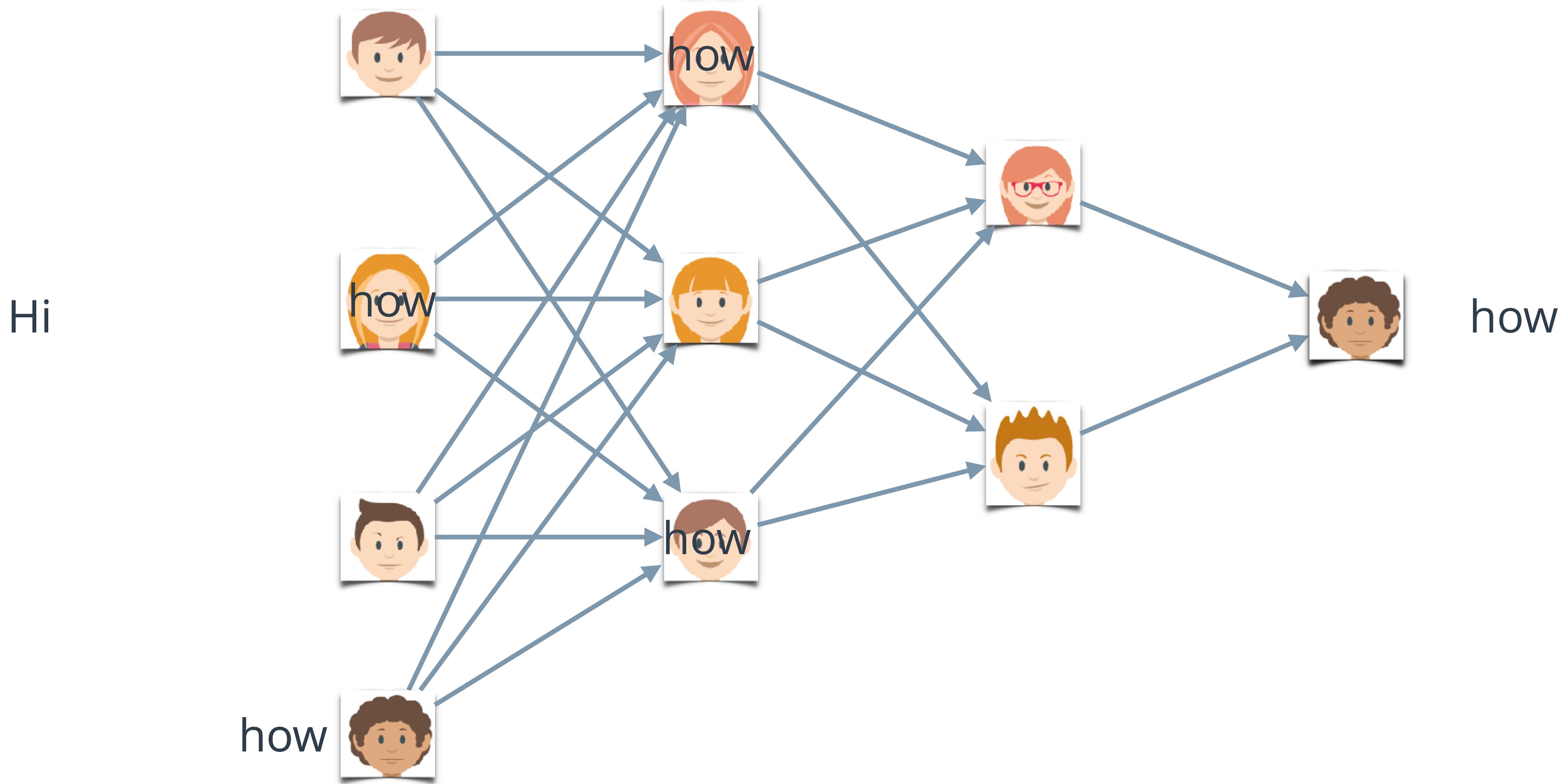
Recurrent Neural Network



Recurrent Neural Network



Recurrent Neural Network



Recurrent Neural Network

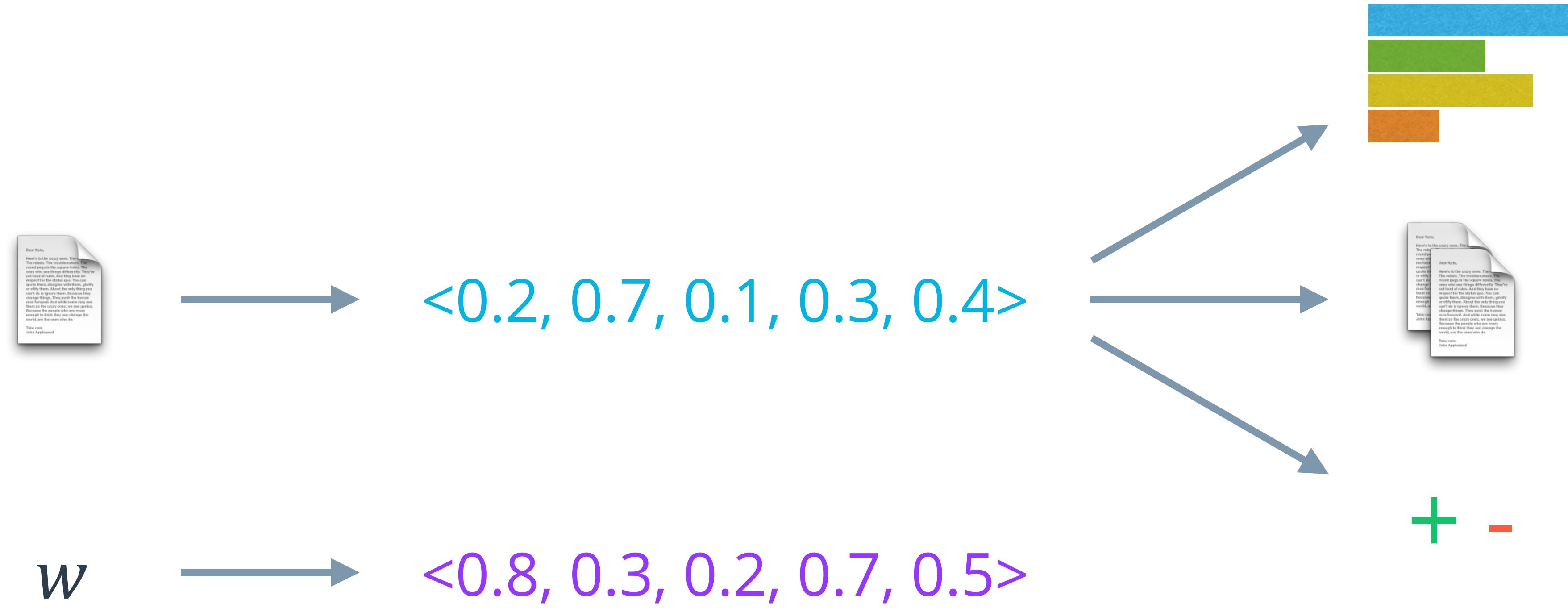


Deep NLP: Word Embeddings

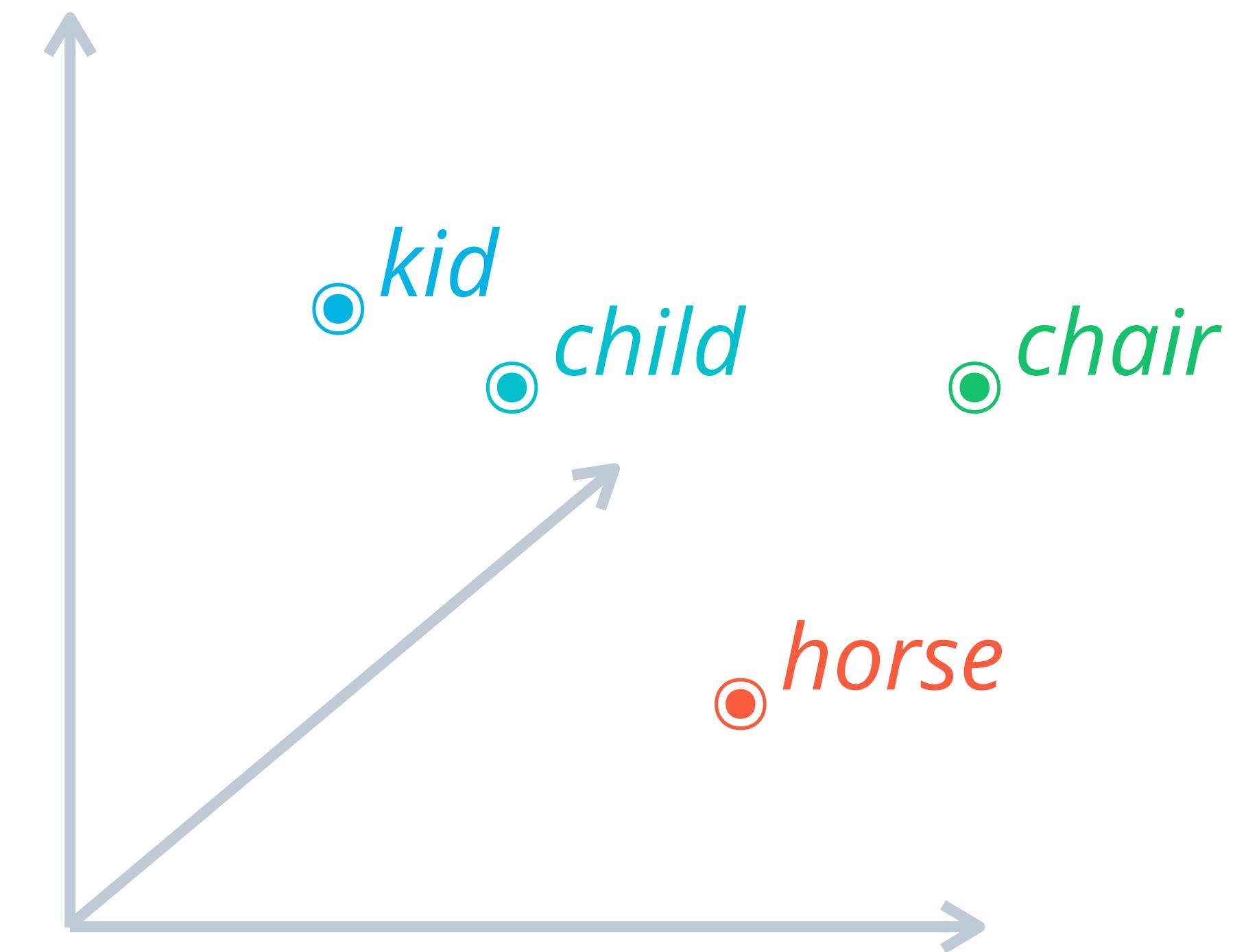
Word Embeddings

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

Document vs. Word Representations



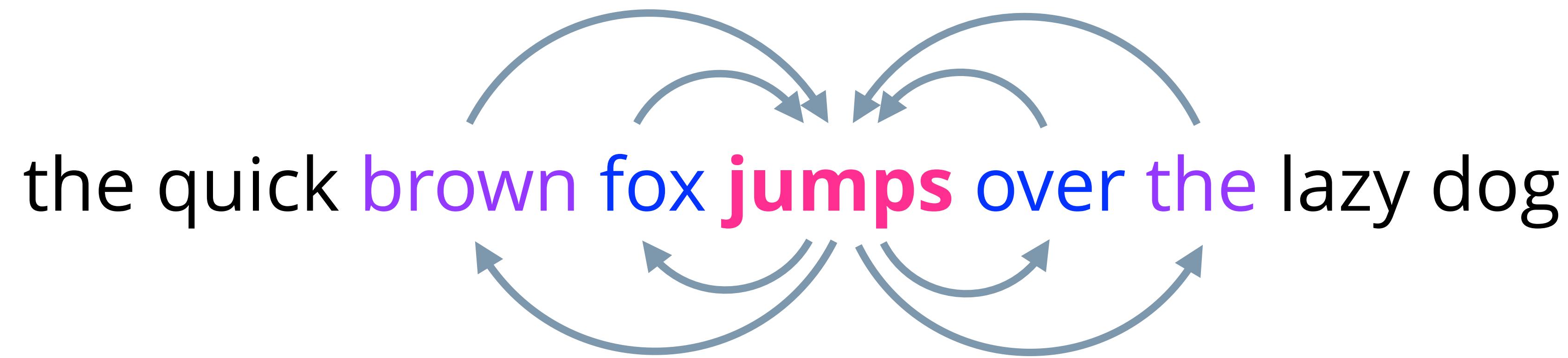
Word Embeddings



Word2Vec

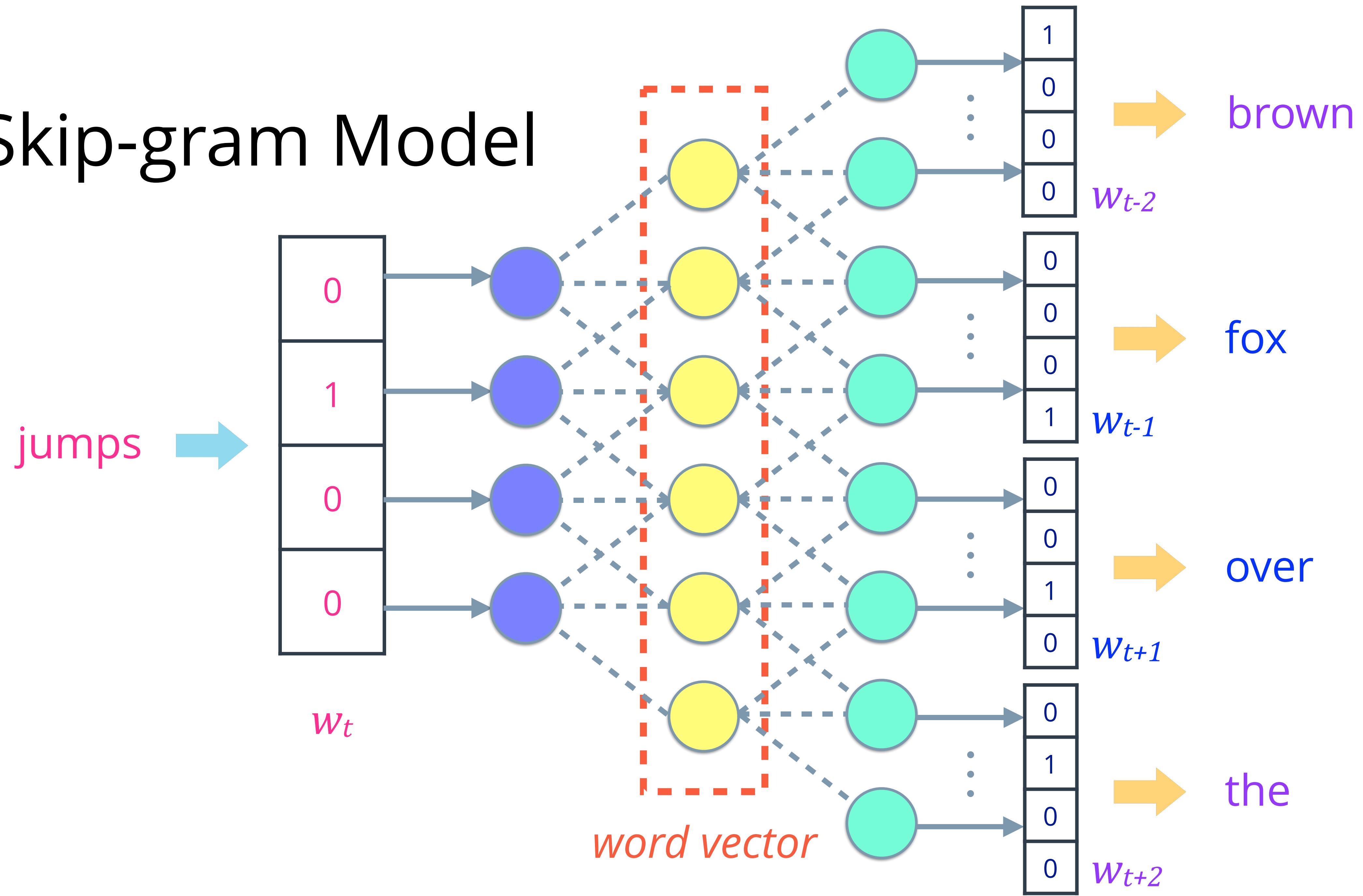
Word2Vec

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!

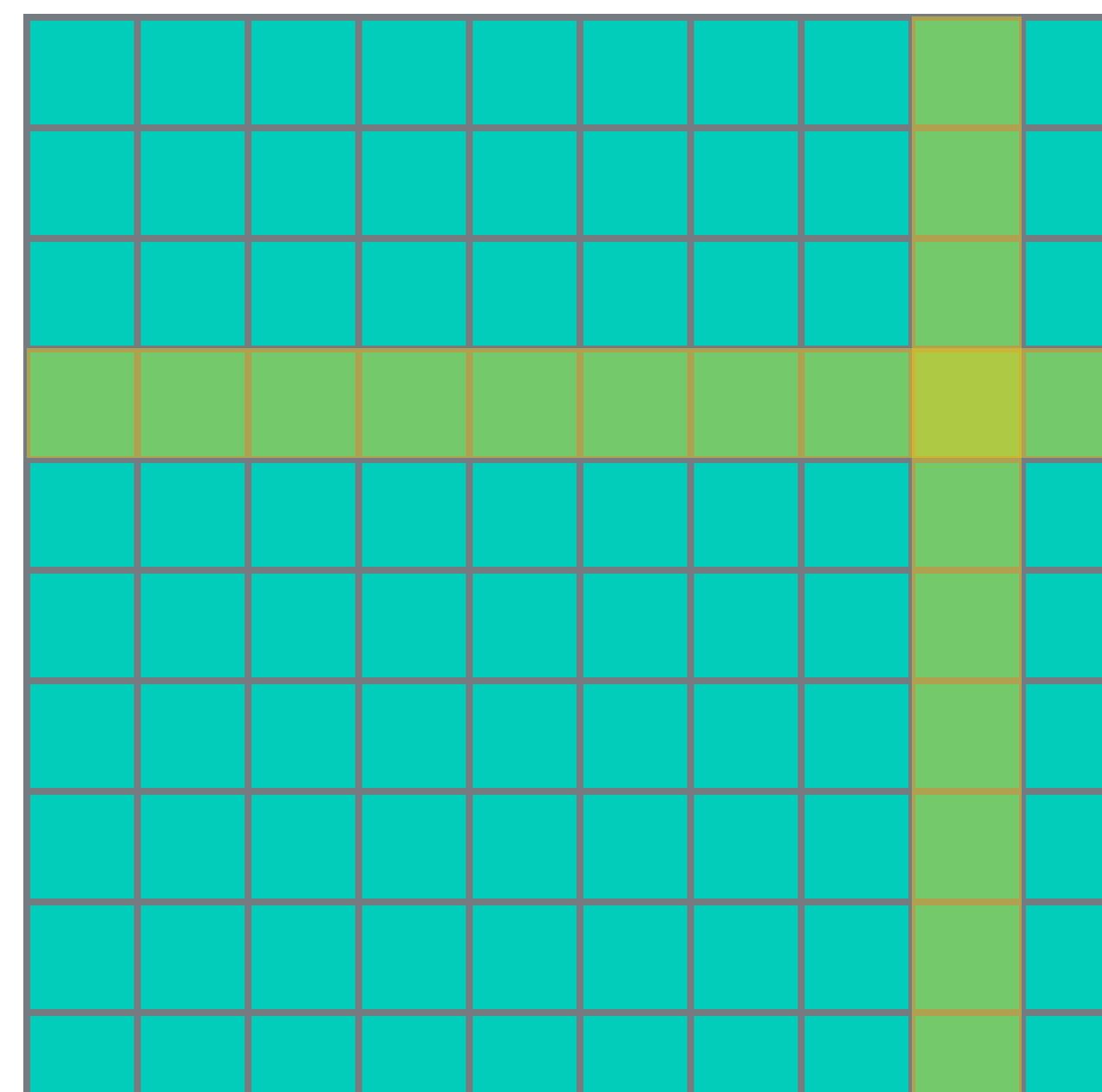
Word2Vec: Further Reading

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.

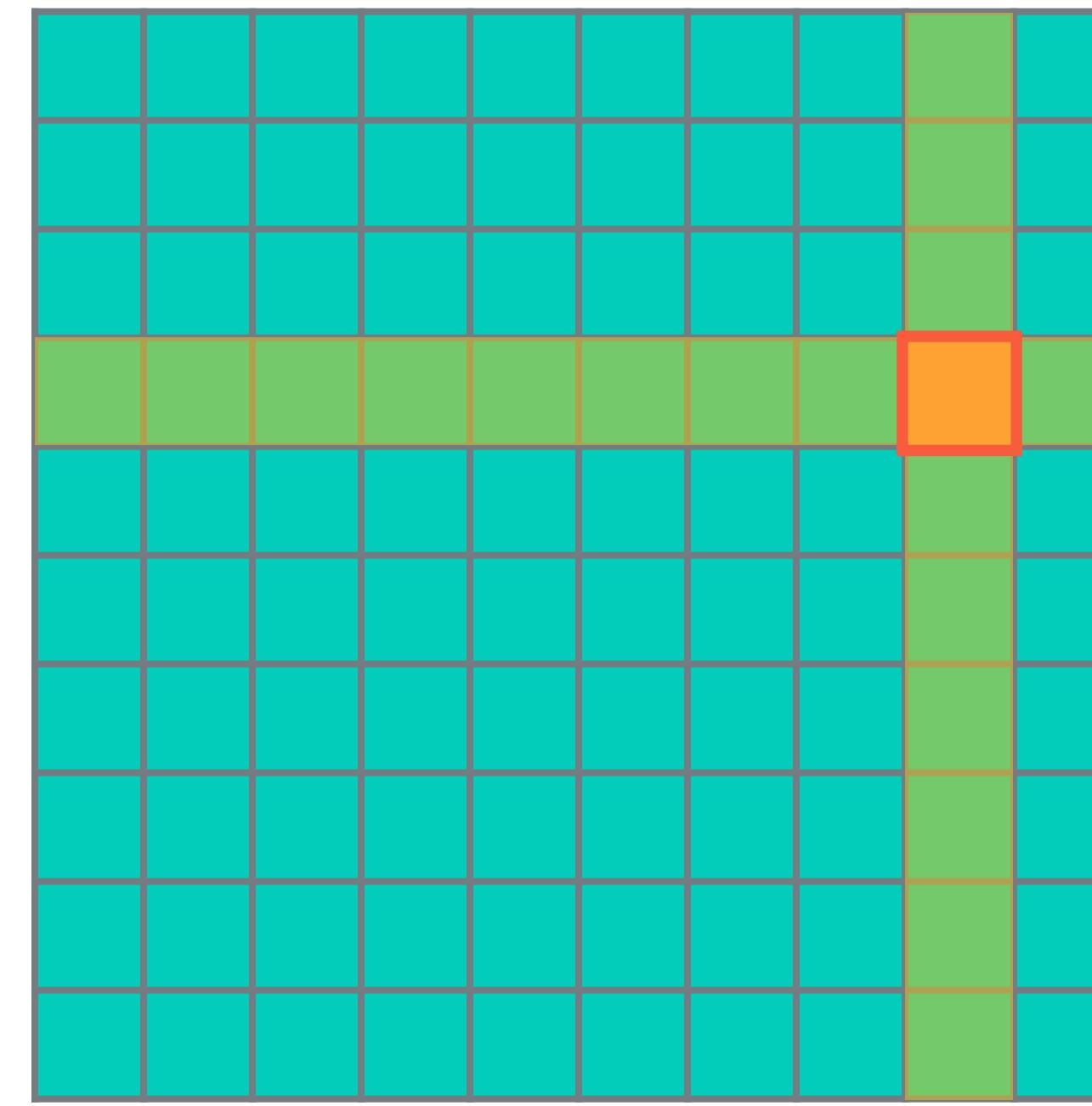
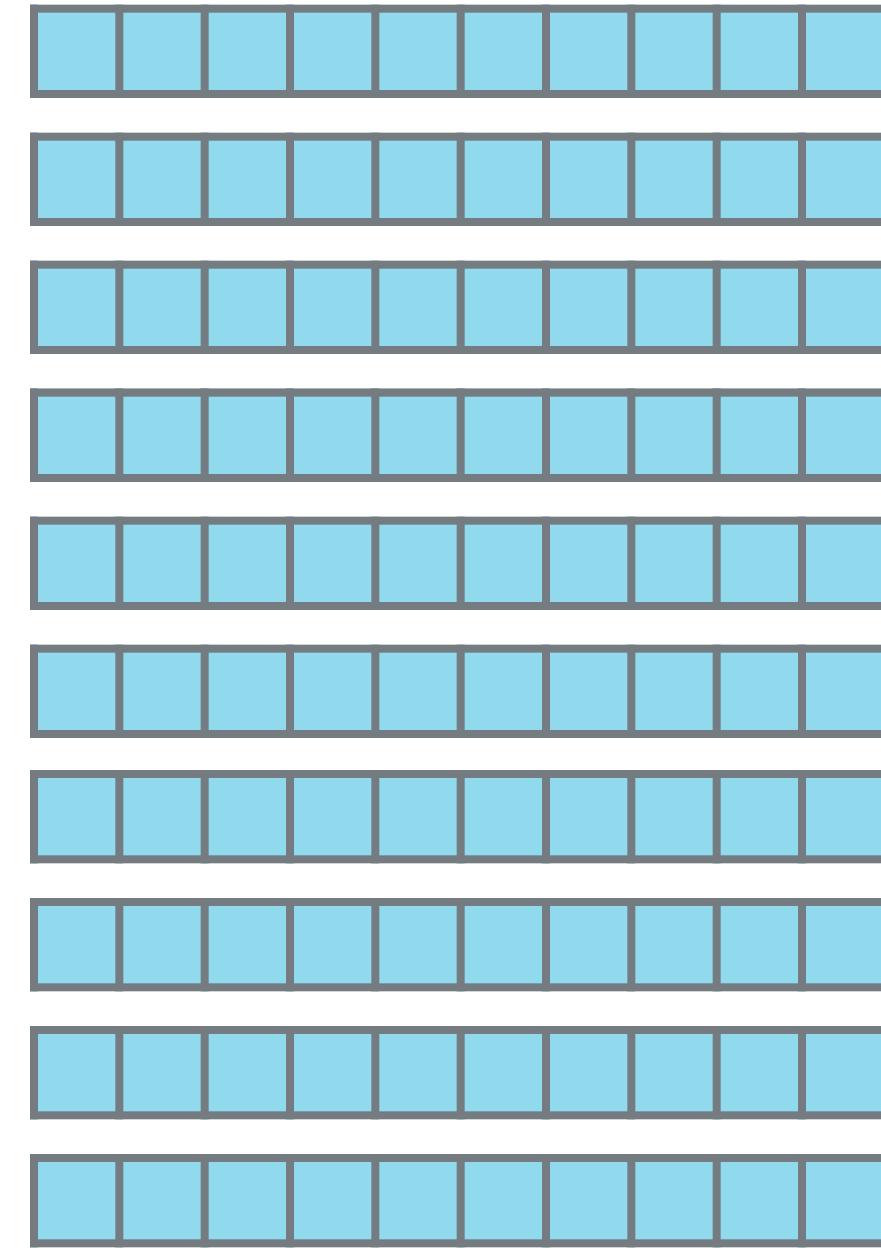
GloVe

Global Vectors for Word Representation

 $P(j | i)$ j i

*Context?
a cup of coffee*

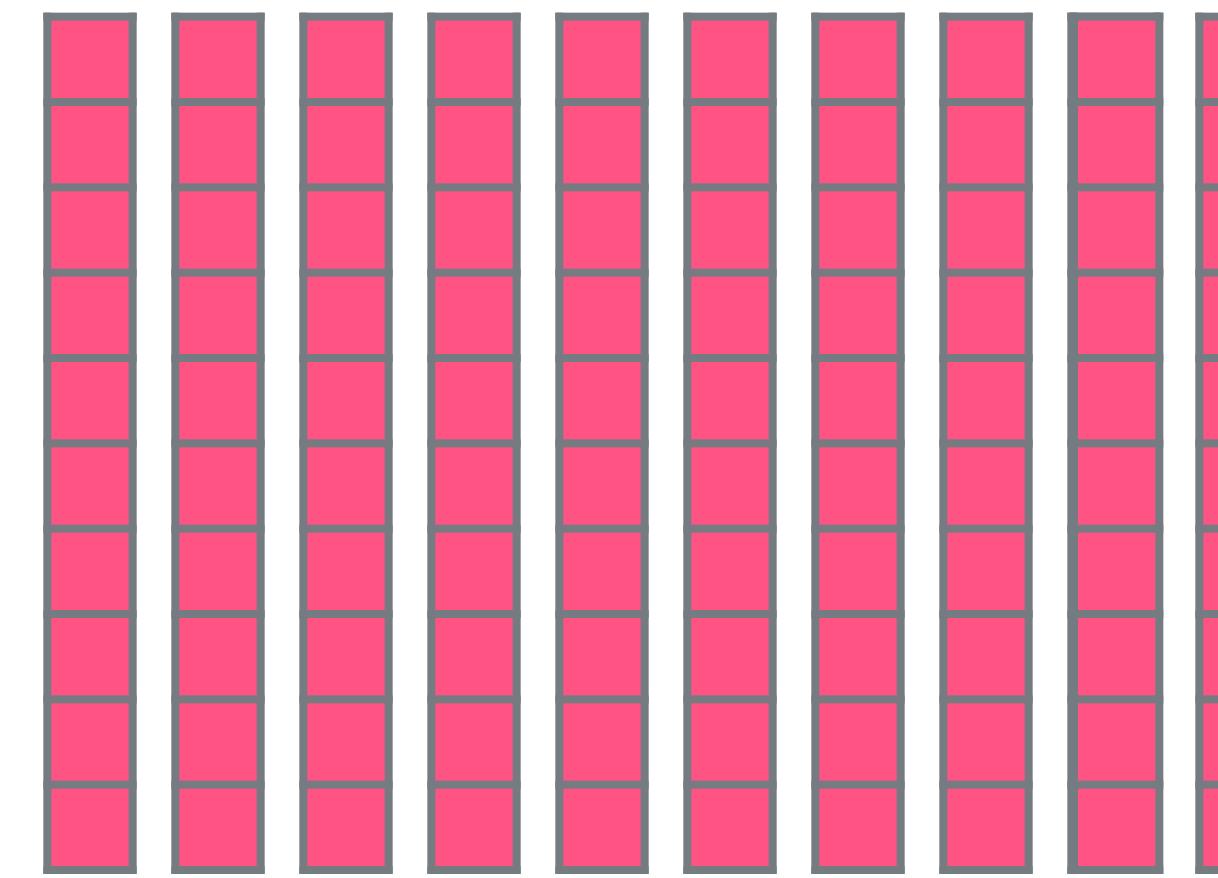
Context



$P(j | i)$

j

Target

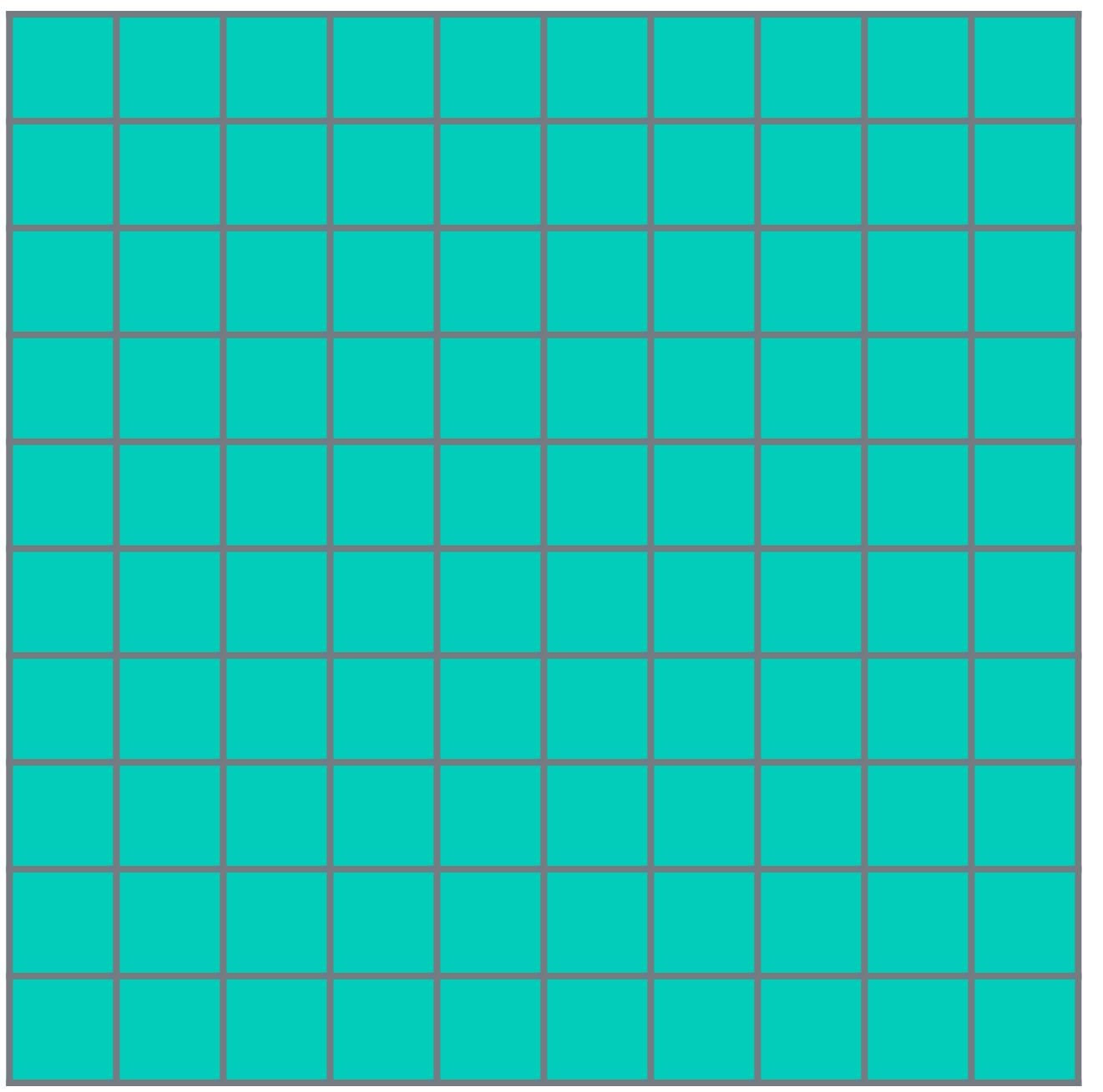


w_i

w_j

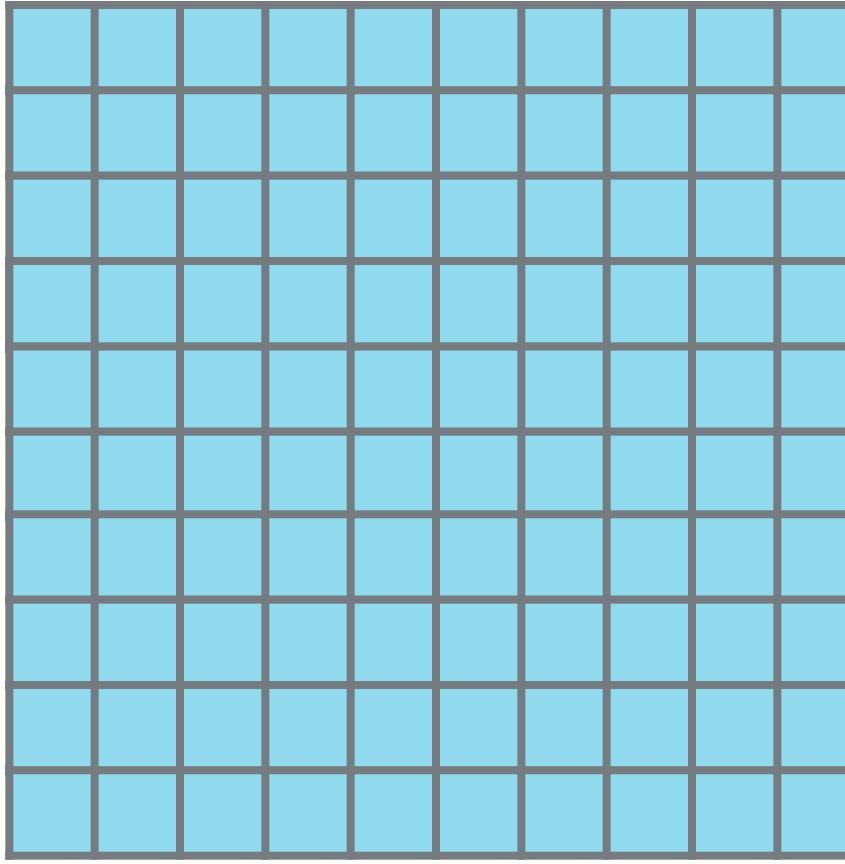
$\bullet =$



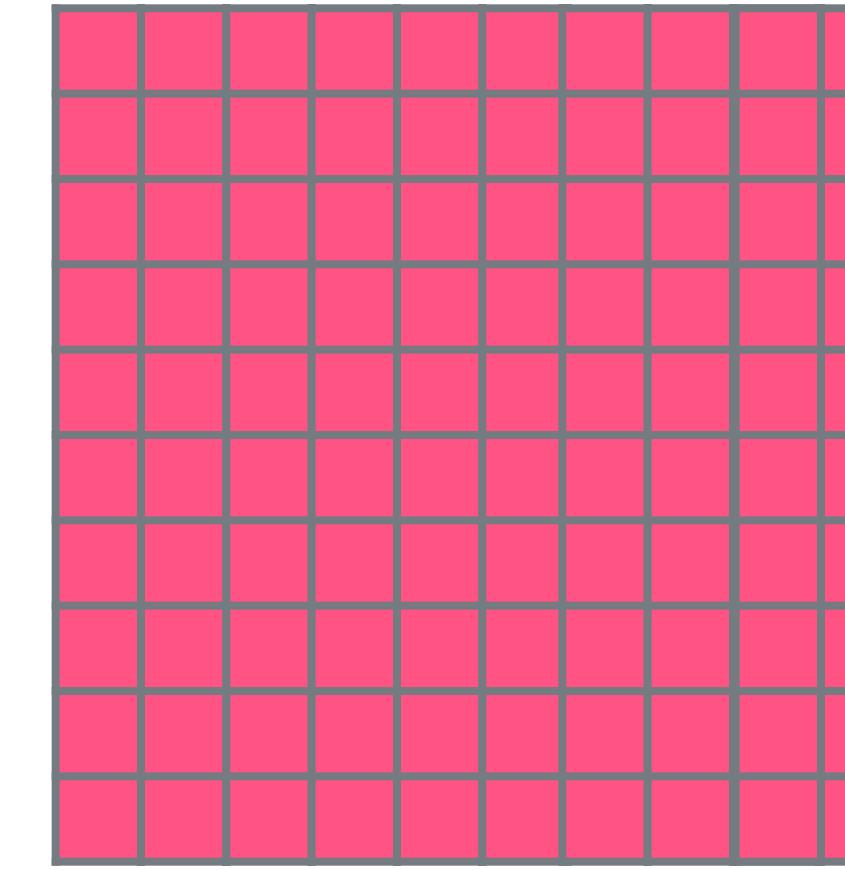


$$P(j \mid i)$$

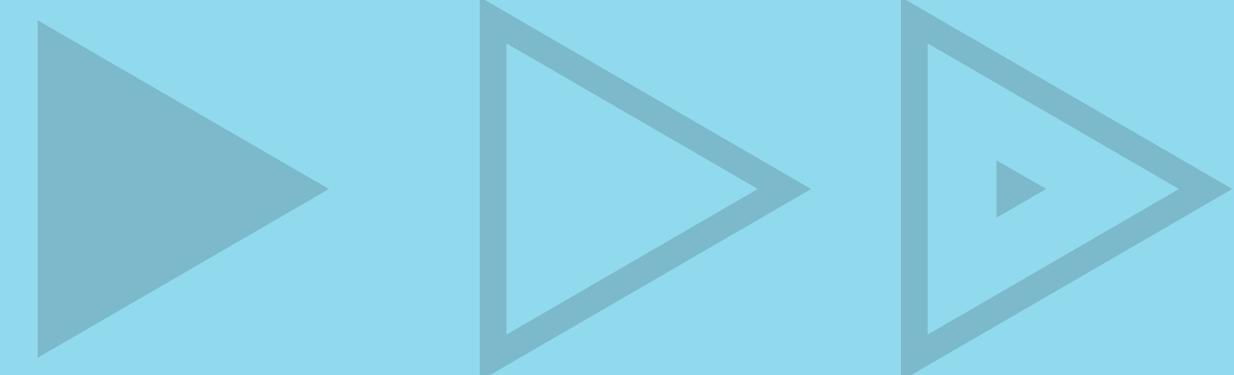
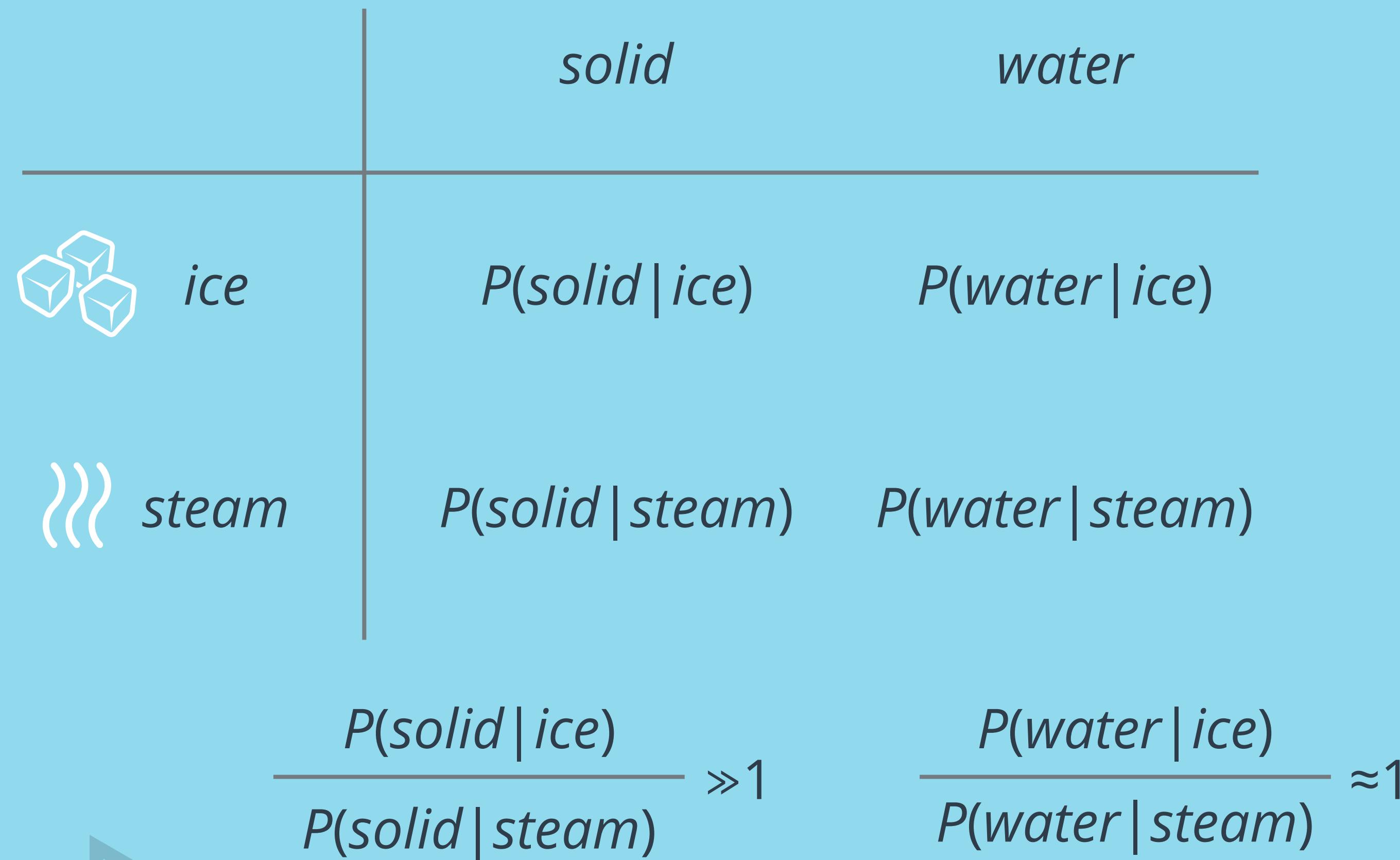
=



×

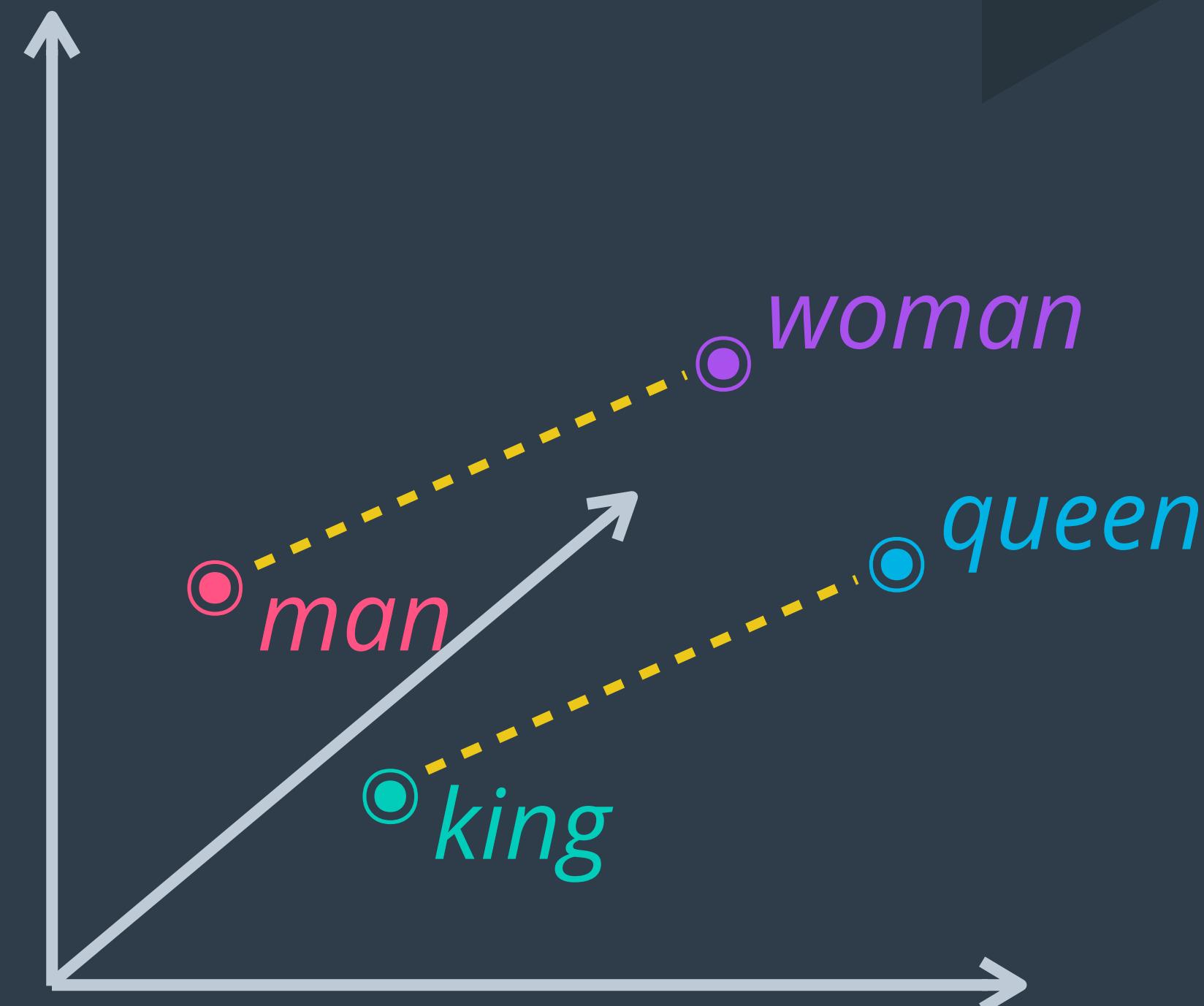


Co-occurrence Probabilities



Embeddings in Deep Learning

woman - man + king = queen



Distributional Hypothesis

“Would you like a cup of _____?”

“I like my _____ black.”

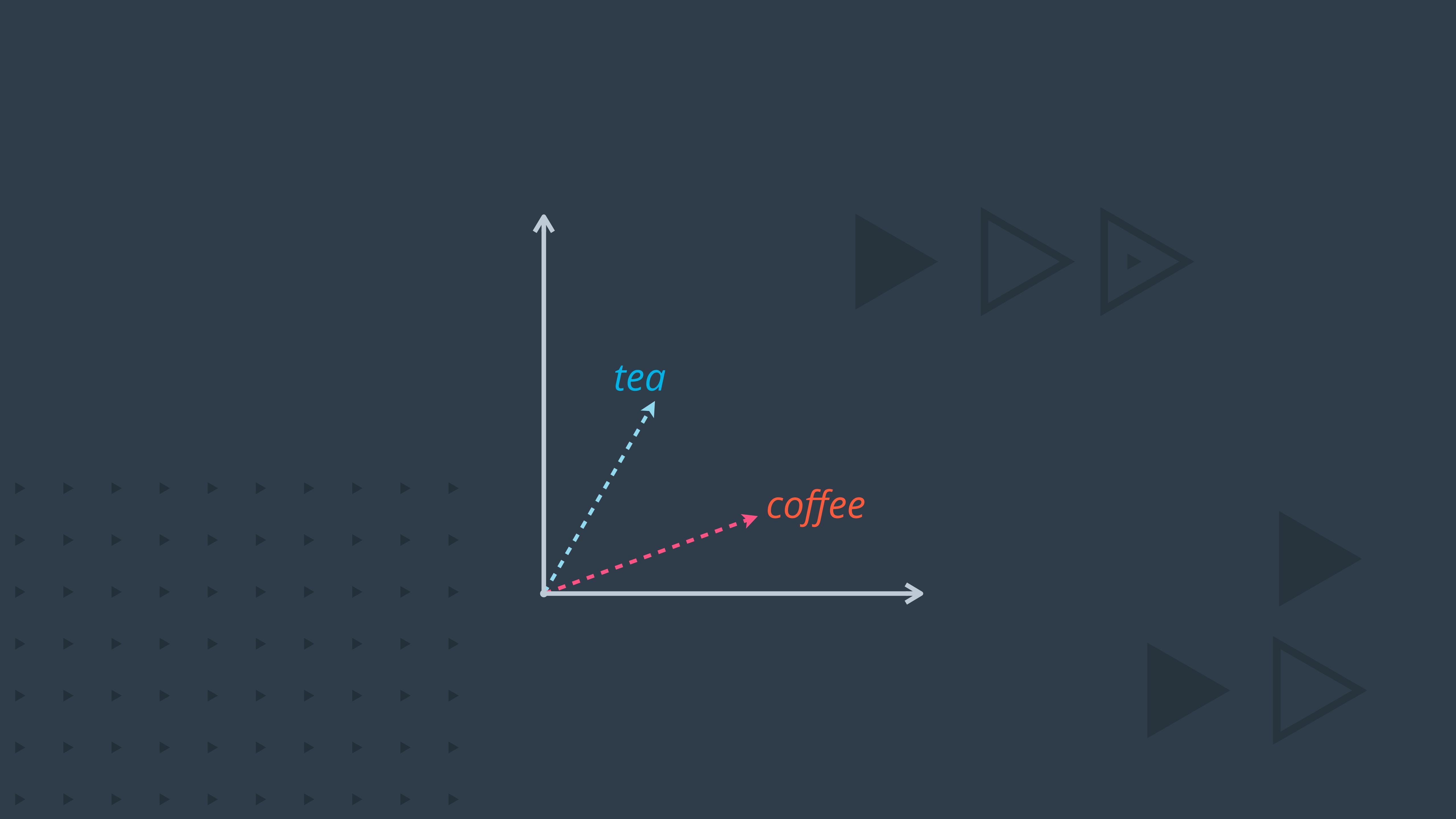
“I need my morning _____ before I can do anything!”



"Would you like a cup of _____?"

"I like my _____ black."

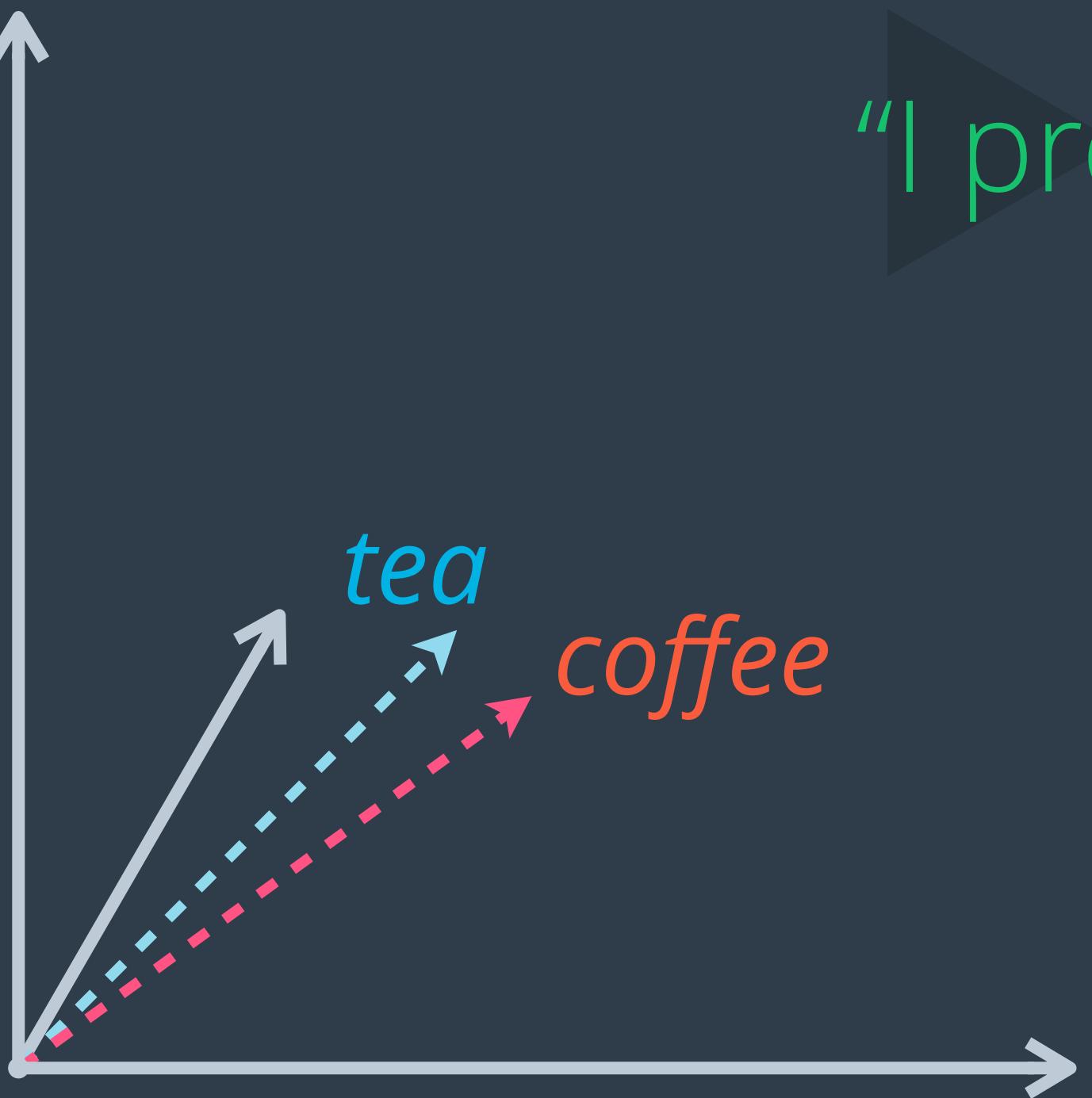
"I need my morning _____ before I can do anything!"



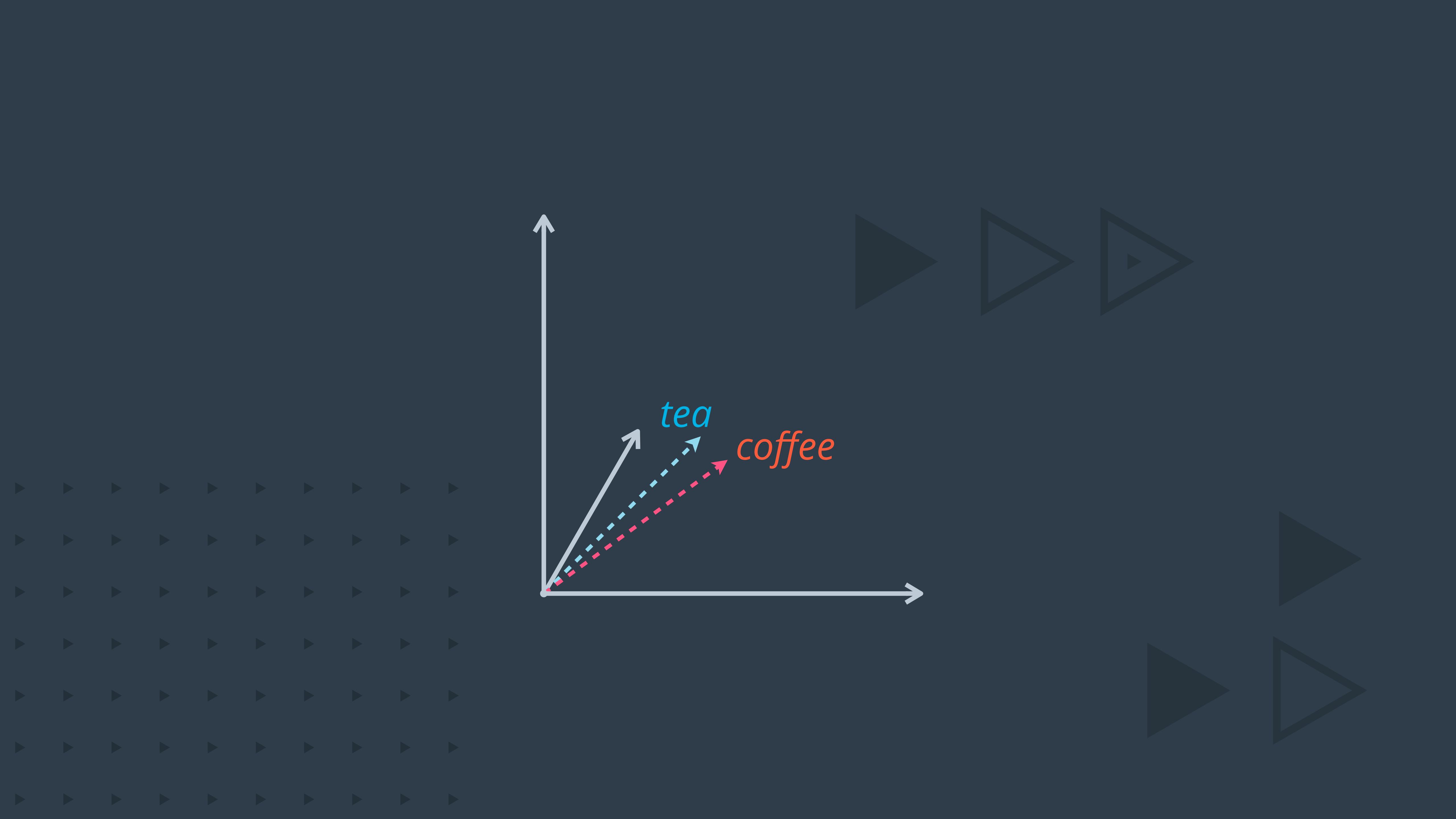
tea

coffee

"Coffee grounds are great for composting!"

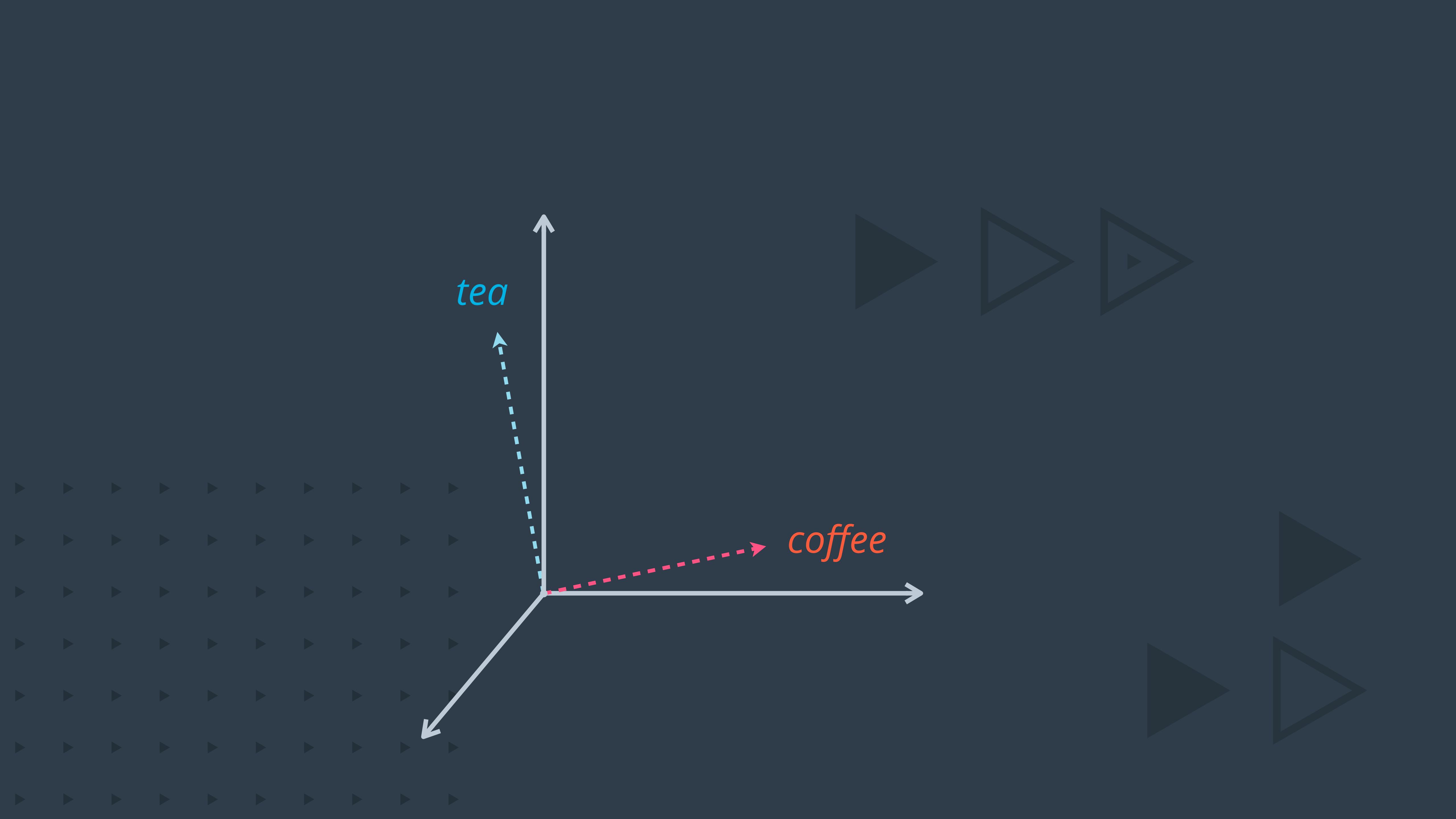


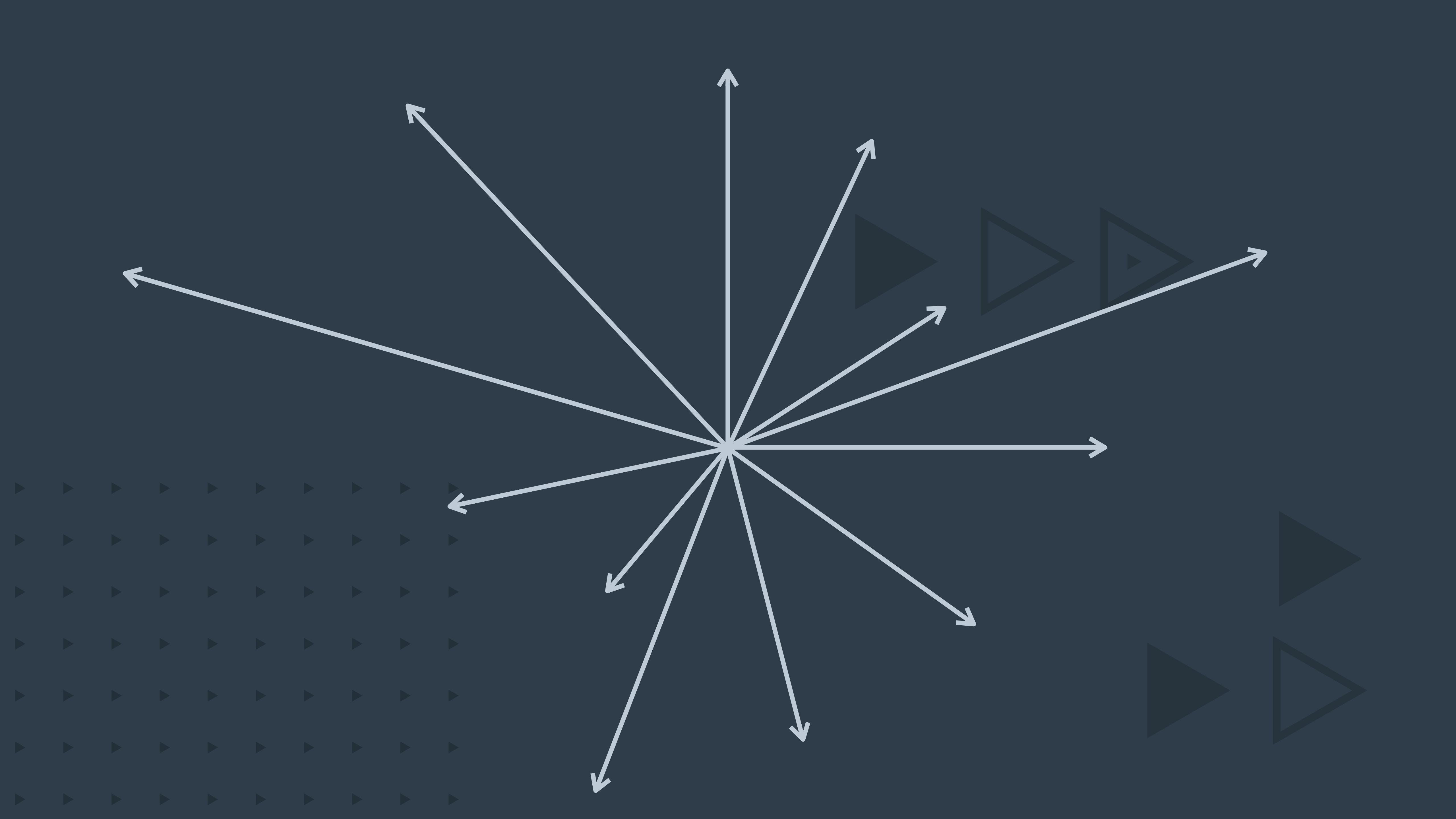
"I prefer loose leaf tea."

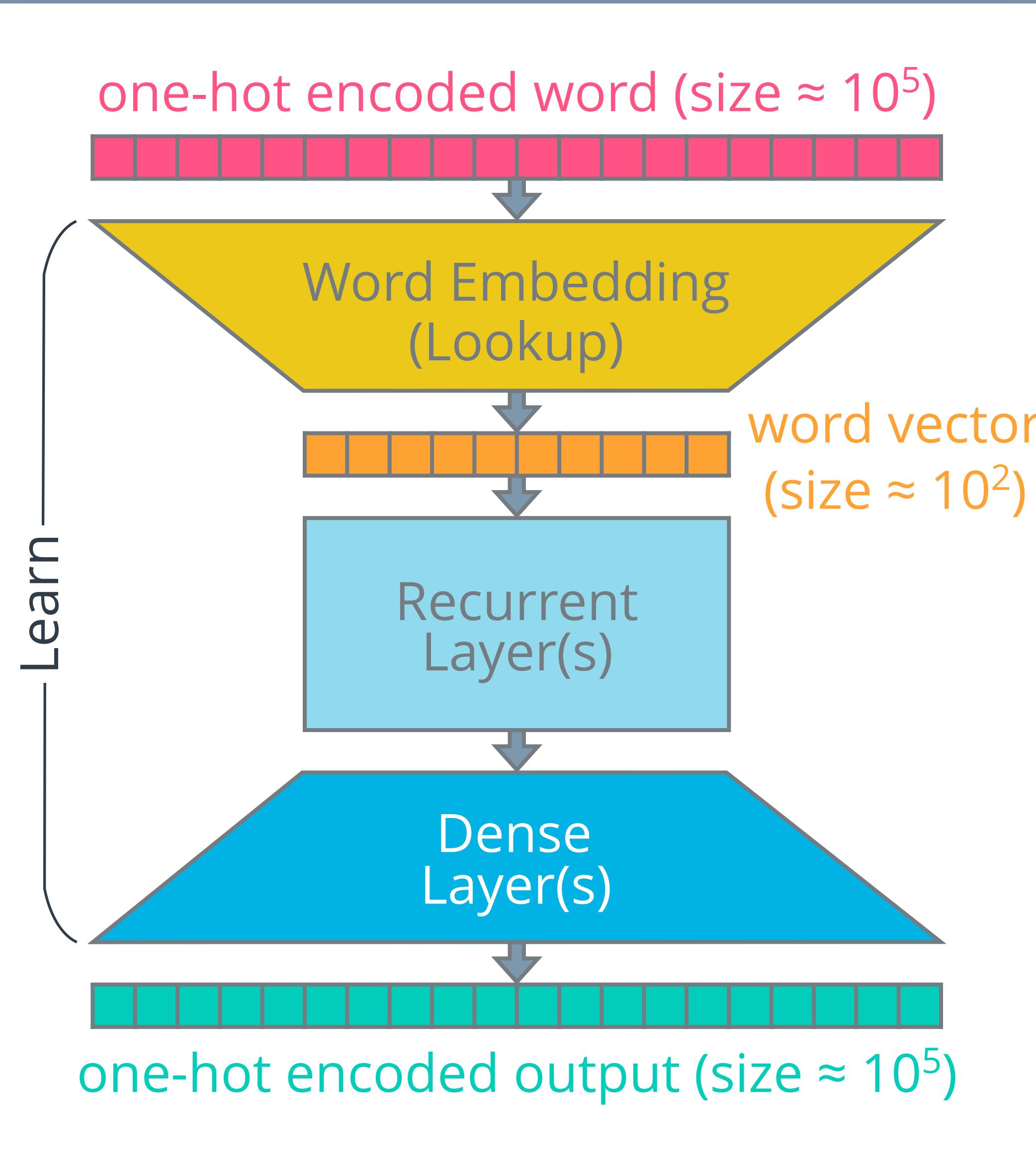


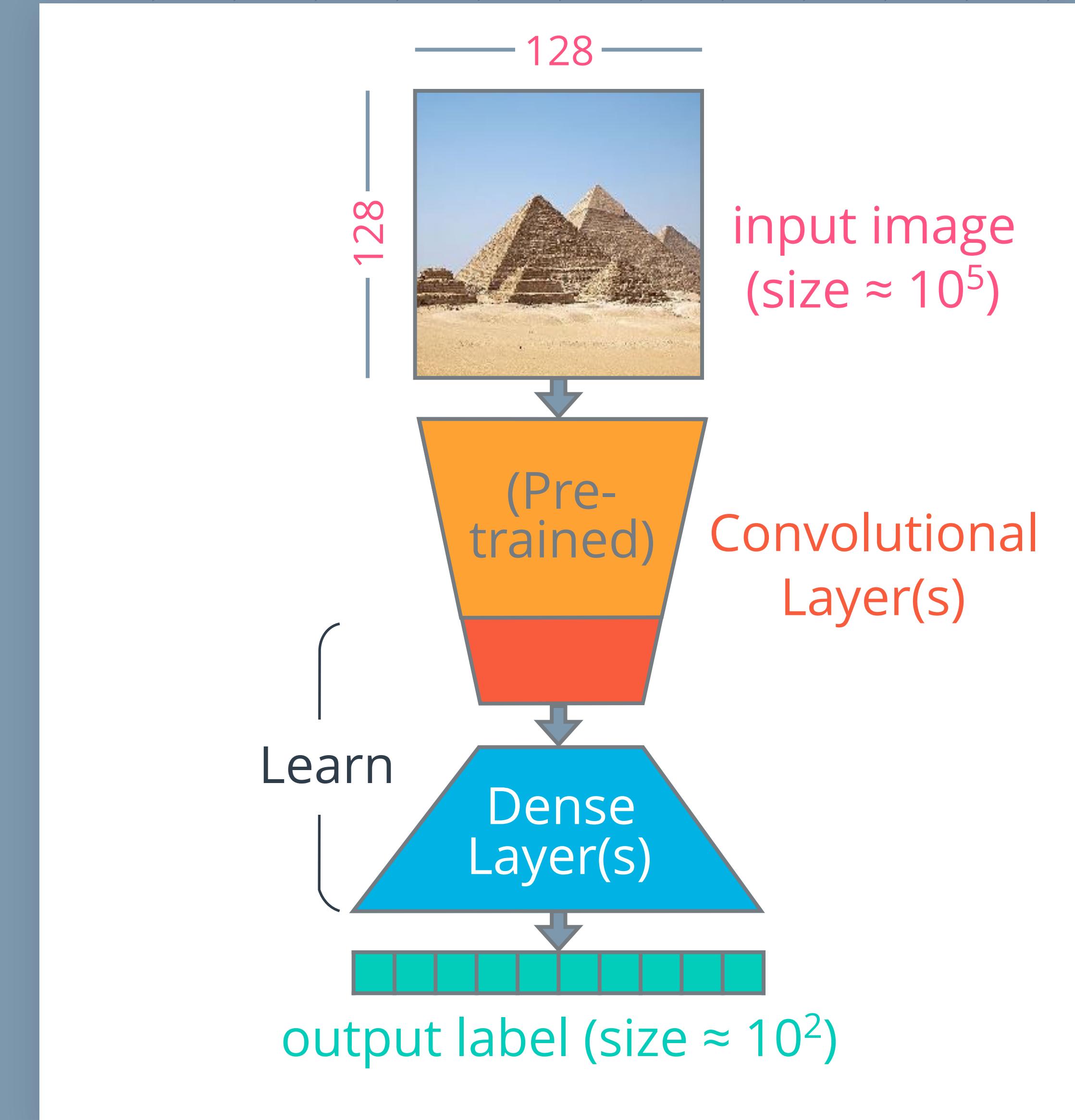
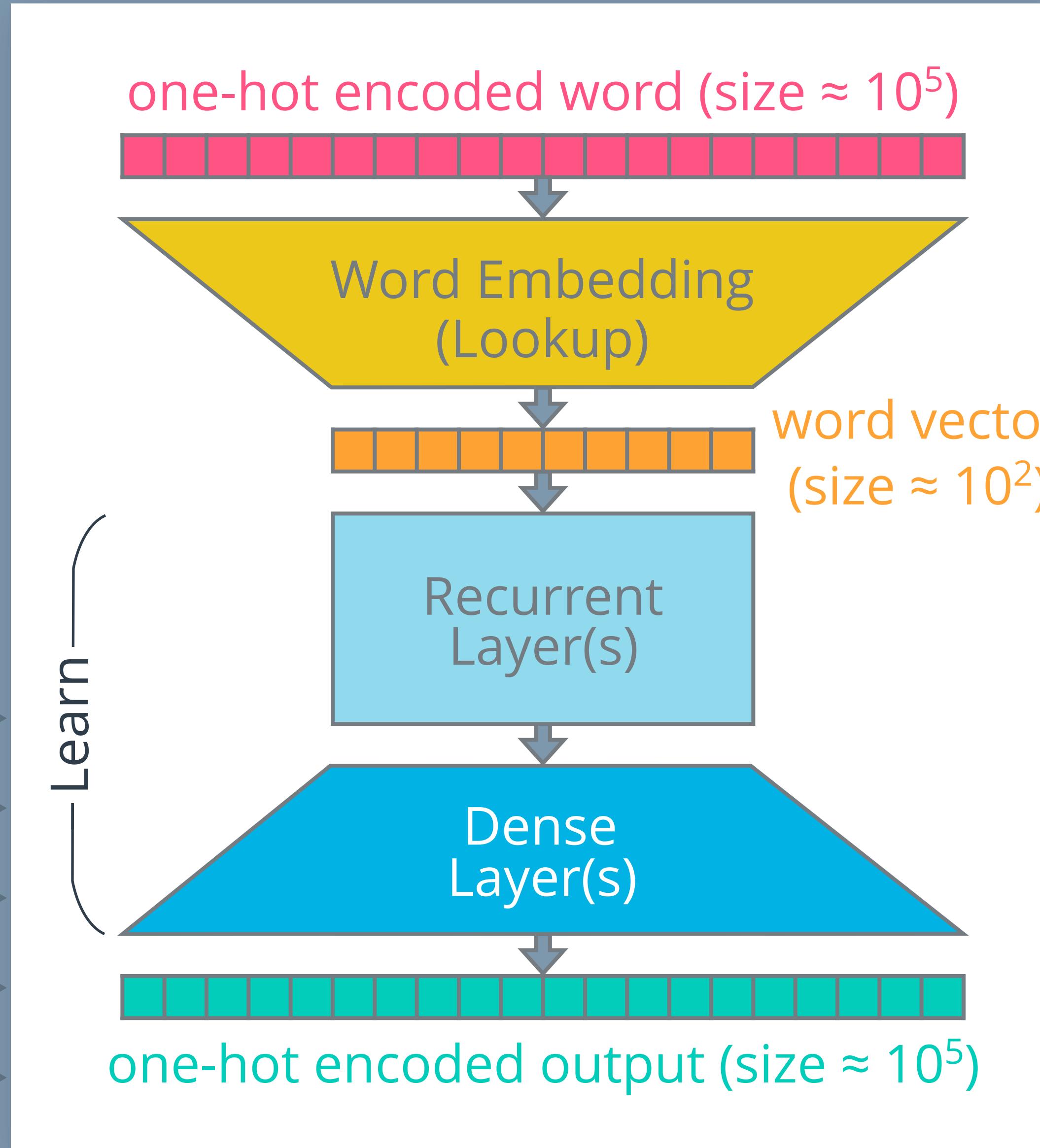
tea

coffee







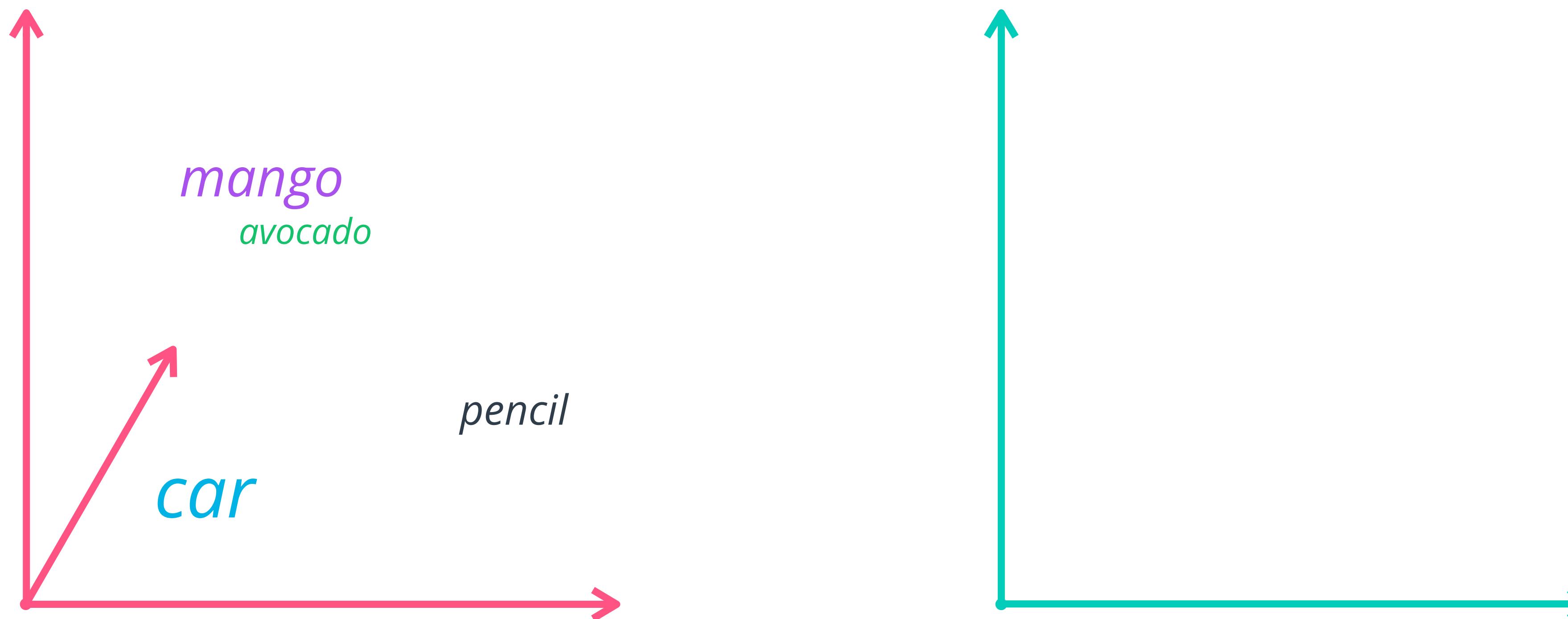


Visualizing Word Vectors: t-SNE

t-Distributed Stochastic Neighbor Embedding

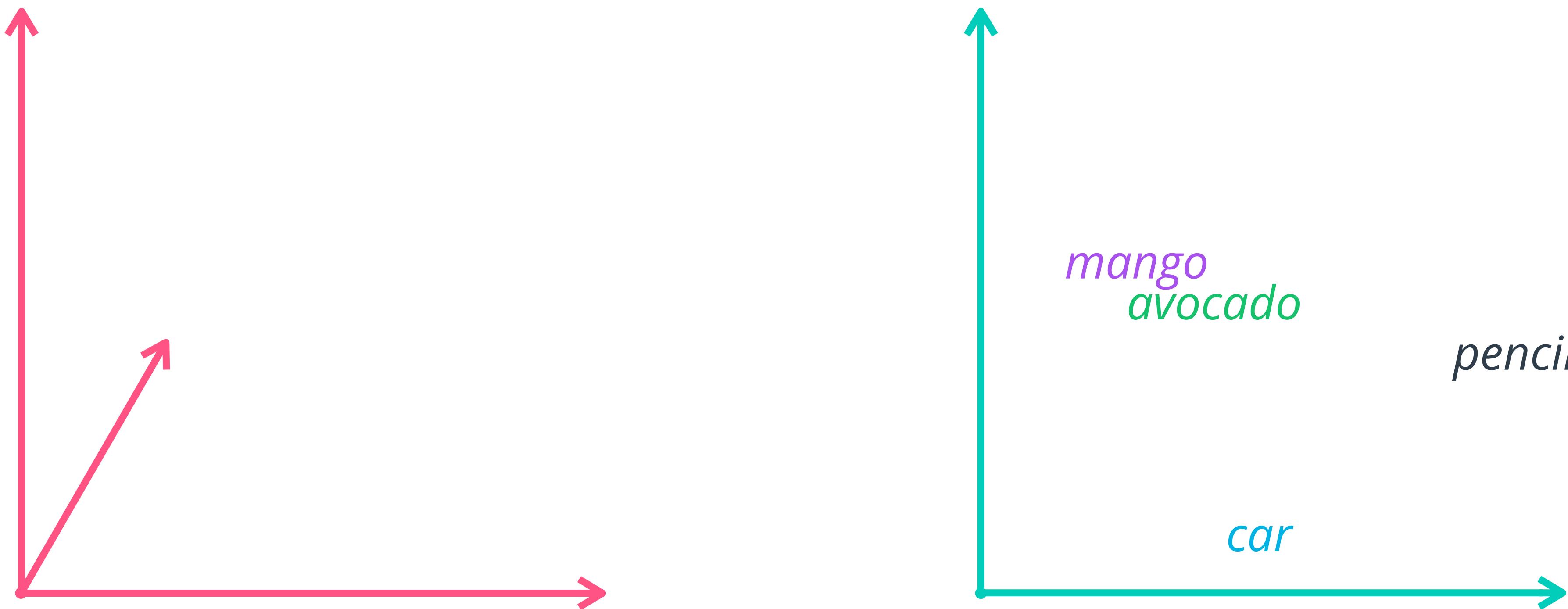
t-SNE

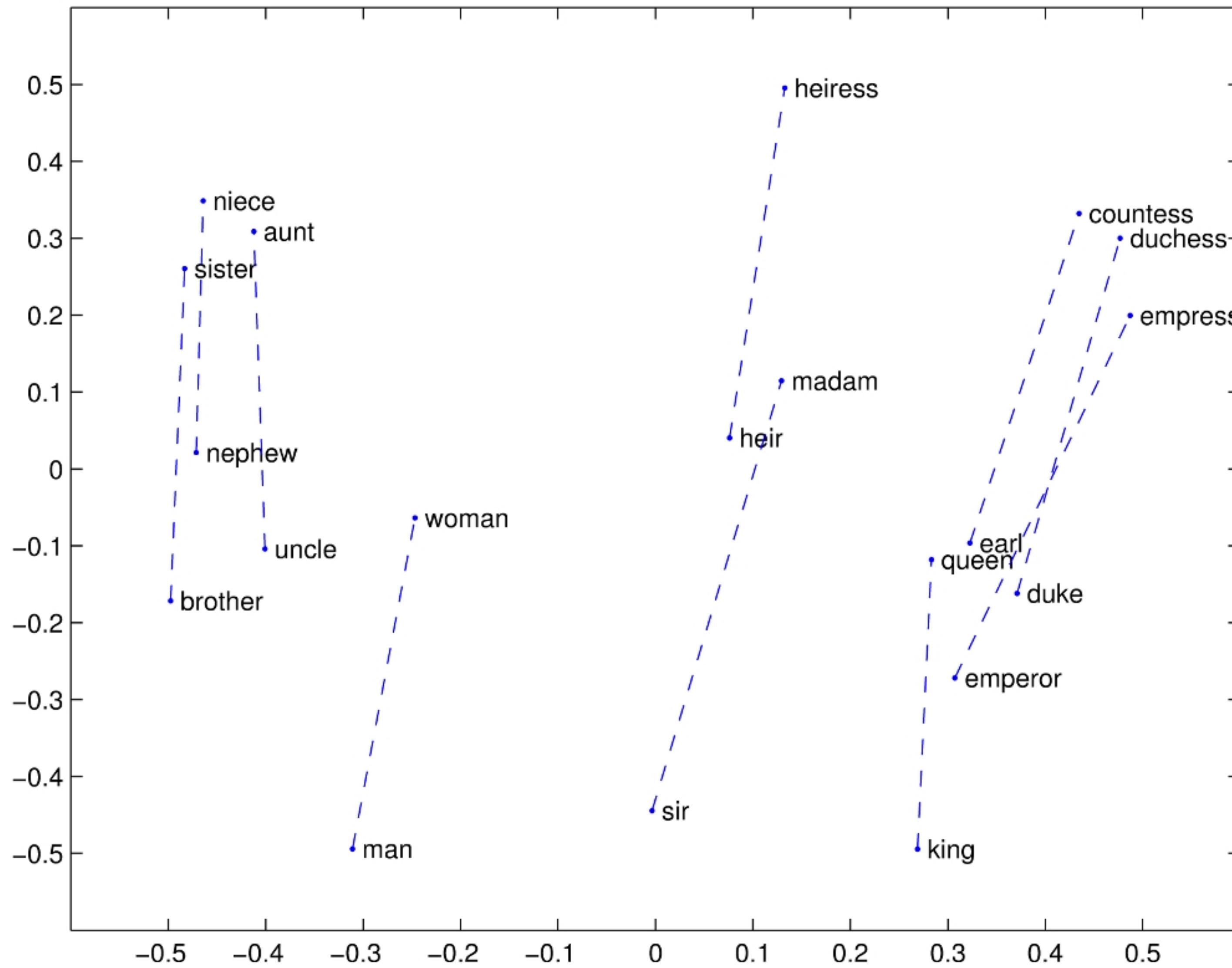
t-Distributed Stochastic Neighbor Embedding

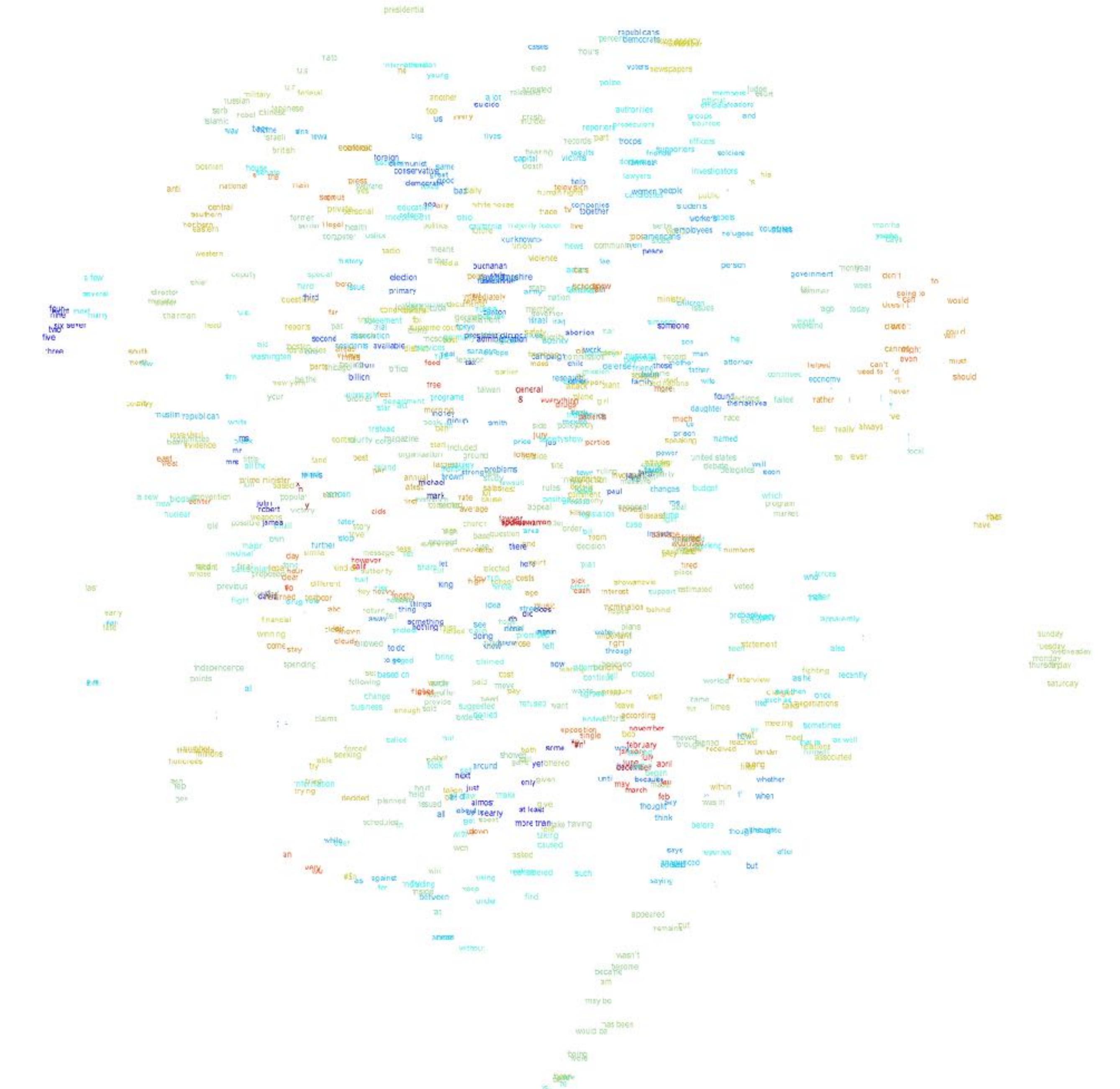


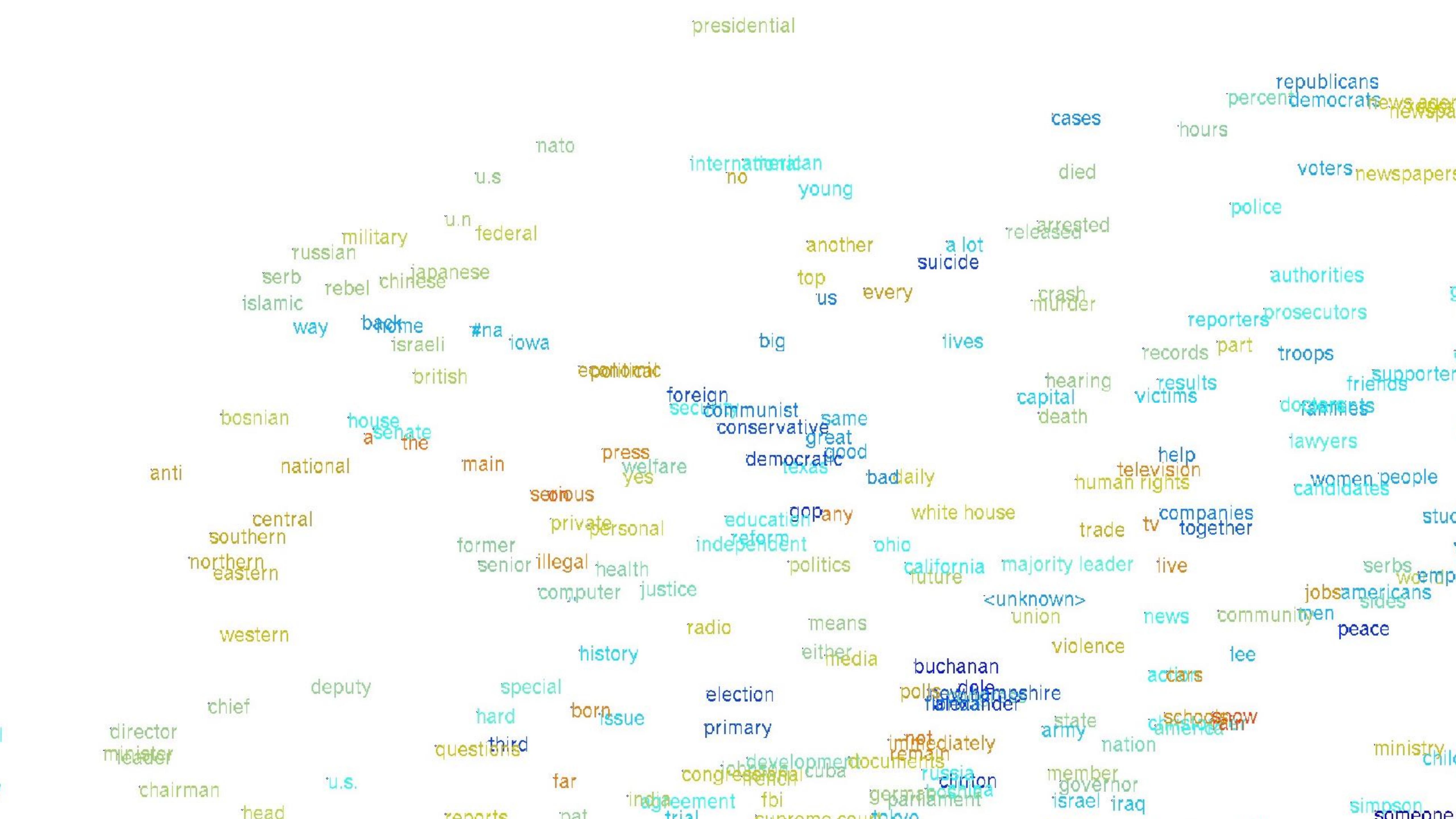
t-SNE

t-Distributed Stochastic Neighbor Embedding









things
return away tell
own
Judy allowed to do
to go
set based on following
change business
enough
called
forced king
decided
scheduled
\$n as

idea
protect
nothing
raise
see
doing
know
bring
cost
move
pay
need
refused
want
opposition
single
both
some
shot
took
get
next
hold
held
taken
just
off
make
almost
all
about
up
early
got
spent
with
down
won
win
against
for
including
inside
between
keep
using
under
find
street
do
hope
done
promised
again
claim
knew
rose
left
claimed
paid
move
pay
suggested
ordered
put
showed
gave
call
yet
offered
only
given
at least
give
more than
take
having
told
taking
caused
asked
making
considered
such
at

music
did
hopes
plans
importance
right
through
believed
attempting
continuing
closed
wants
pressure
agreed
leave
according
ended
efforts
november
bob
february
january
july
june
begin
april
december
began
made
march
may
began
march
feb
thought
say
think
says
reported
announced
saying
appeared
remaining
out
probably
better
already
seen
statement
also
fighting
as he
recently
changed
then
once
like such as
talk negotiations
meeting sometimes
meet that is
as well
relations
himself
associated
along
whether
within
if
when
before
thought
thought
after
but









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



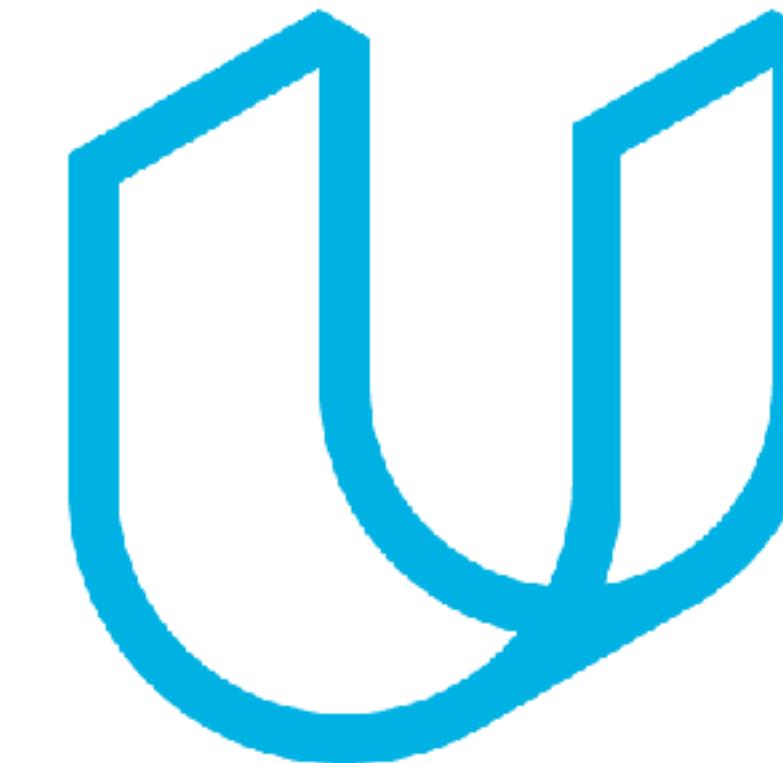
Adarsh Nair



Luis Serrano



Arpan Chakraborty



udacity.com/ai
udacity.com/ml