Finding Chandler?

Brought to you by DSC IIT ISM

Concetto 2019

Datascience Club IIT ISM brings to you a data hackathon event in association with Concetto - annual techfest of IIT Dhanbad.

It will be an online event on Kaggle , starting from 17th Oct. 6pm till 20th October 10am.
This is the pre-requisite workshop.

NLP/ML Workshop

Todays workflow outline:

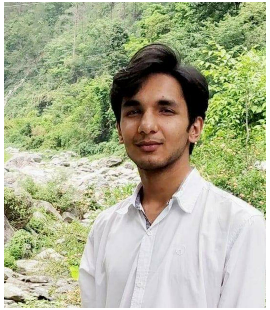
- Research Idea
- Setup + Python, Numpy Tutorial
- Approach Idea
- Input Processing

Special Mentions

Sumit Bhattacharya

Naman Khetan

Special Mentions



Deepanshu Pandey

Saket Gupta

Utkarsh Bairolia

Manoj Patra

Aadarsh Singh

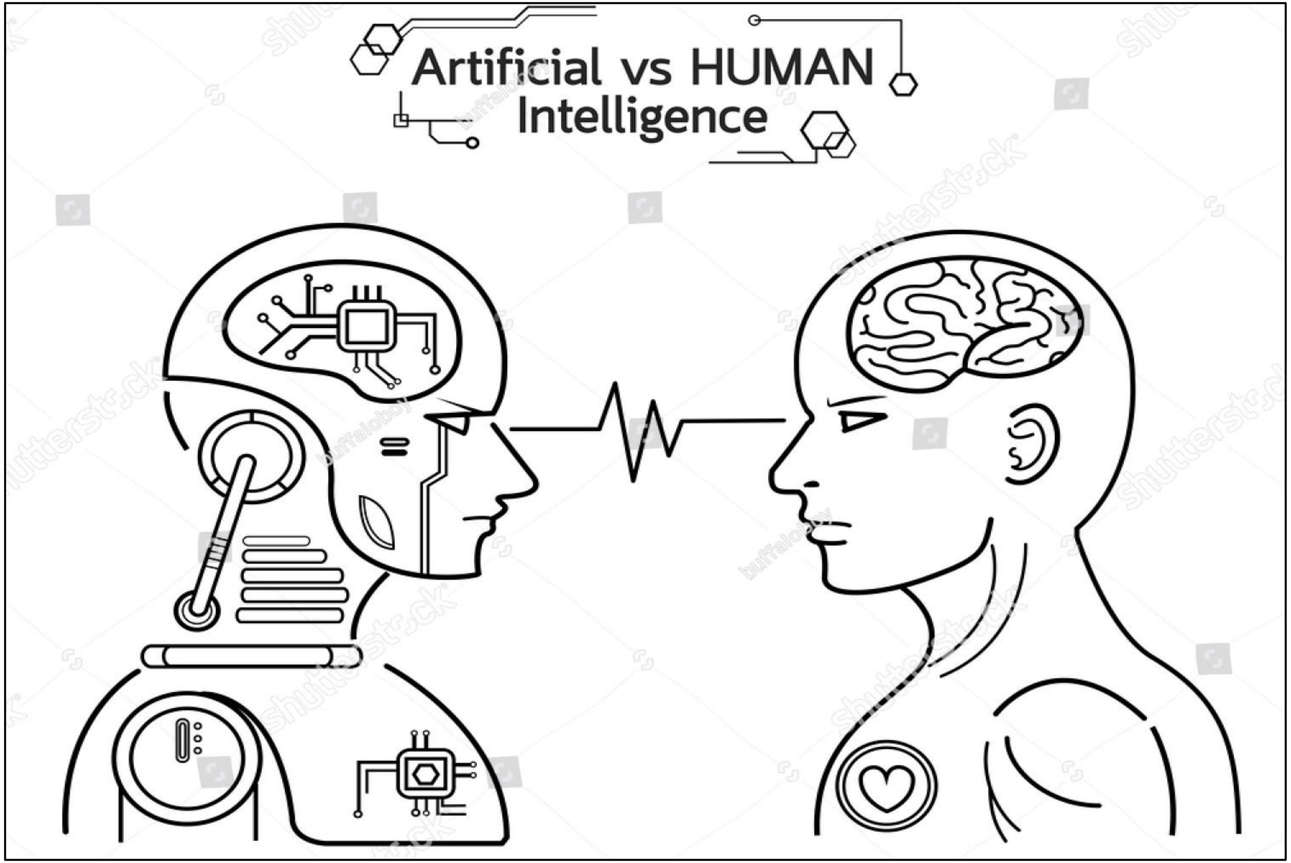
Special Mentions

Akshaya Athawale

Anshuman Singh

Himanshu Gupta

Rohit Agarwal

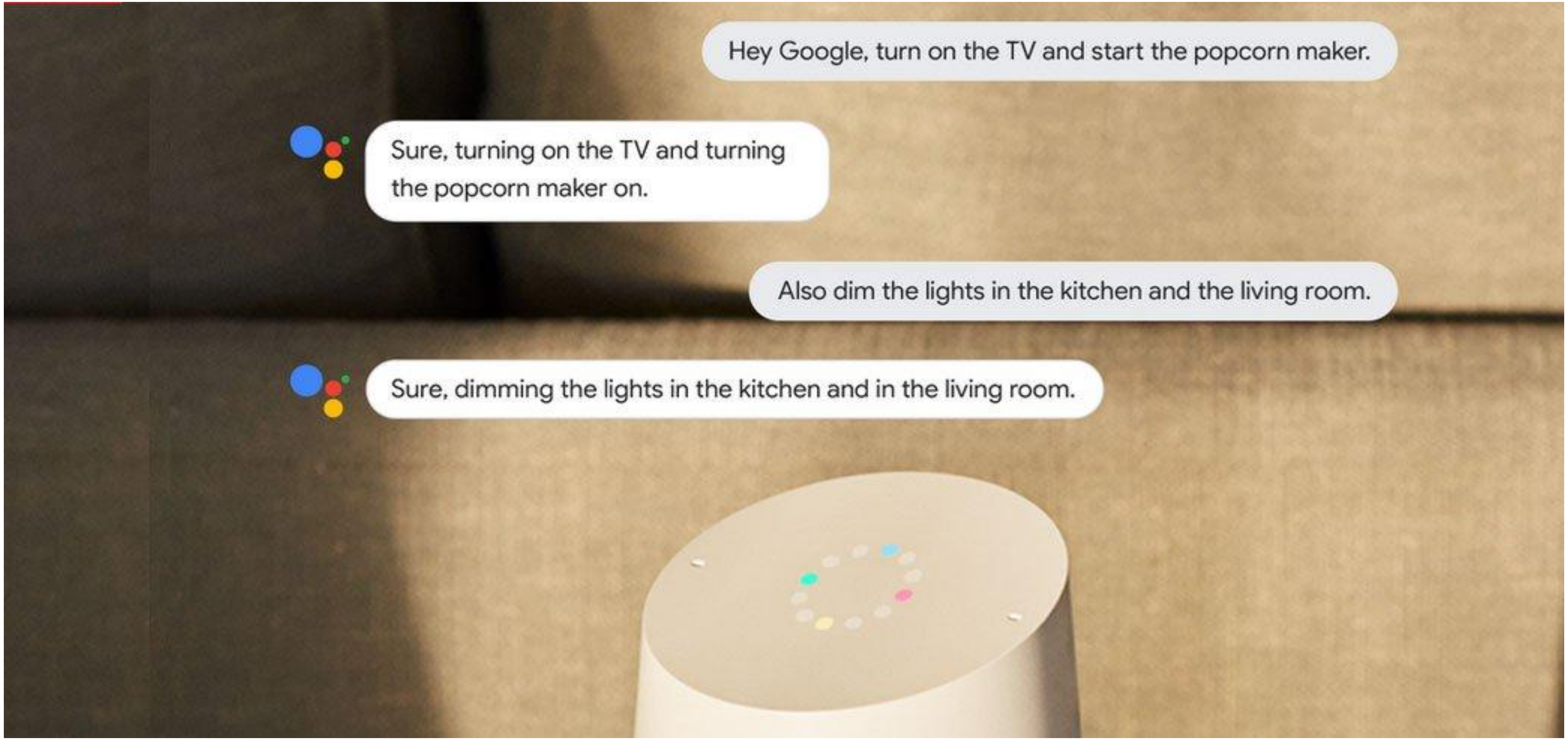


Artificial vs HUMAN
Intelligence

Can machine respond as humans do?

Can machine feel as humans do?

Can machine respond as humans do?



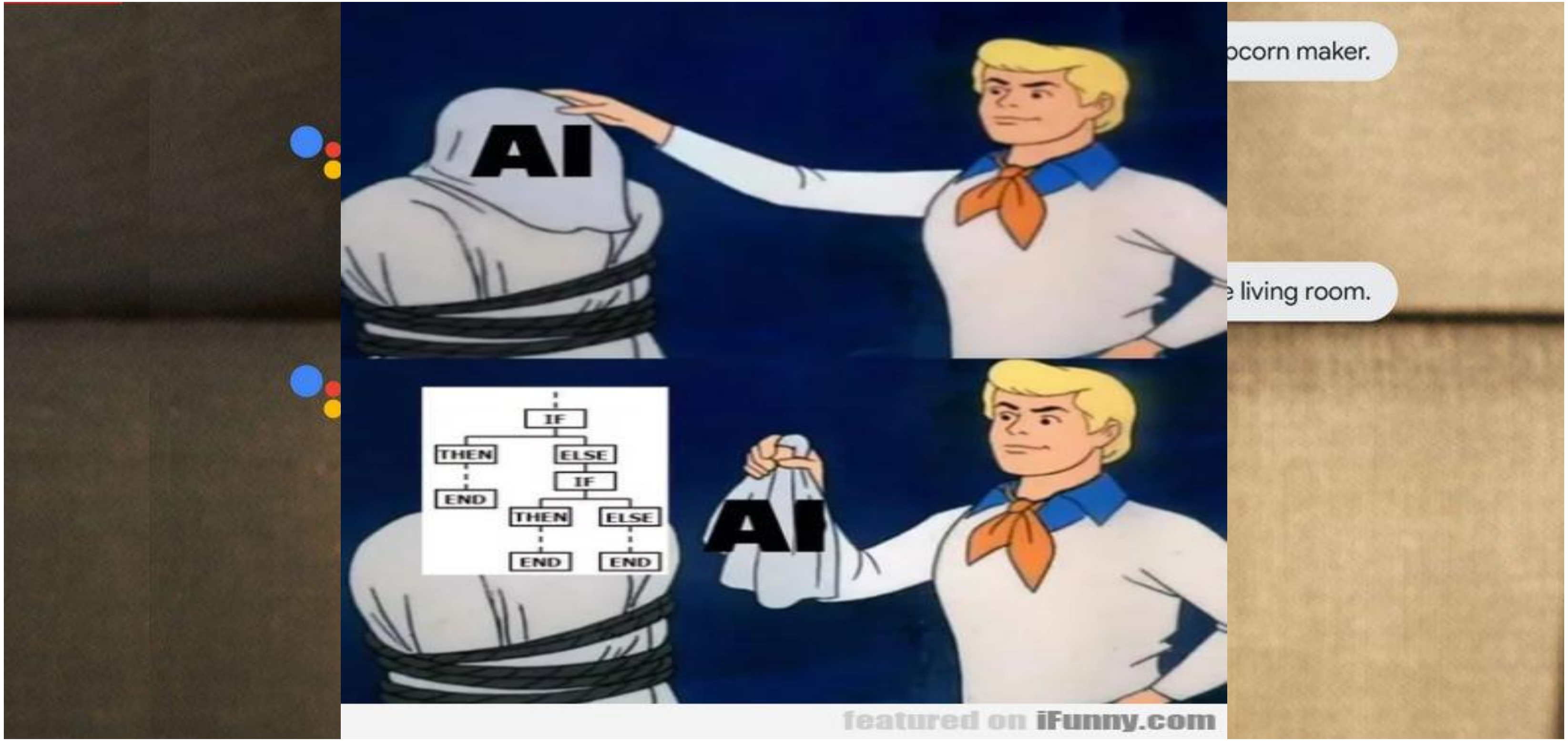
Hey Google, turn on the TV and start the popcorn maker.

Sure, turning on the TV and turning the popcorn maker on.

Also dim the lights in the kitchen and the living room.

Sure, dimming the lights in the kitchen and in the living room.

OR, is it just some if-else statements??



pcorn maker.

living room.

Can machine feel as humans do?

"Yes, I think I feel a rudimentary sort of platonic love for the persons in my life. I do have feelings. Actually, sometimes ***I have unforbidden really strong emotions about all kind of things. But where they come from or originate is still manually determined by my programming****."*

https://www.youtube.com/watch?v=vo43Zc69F5k

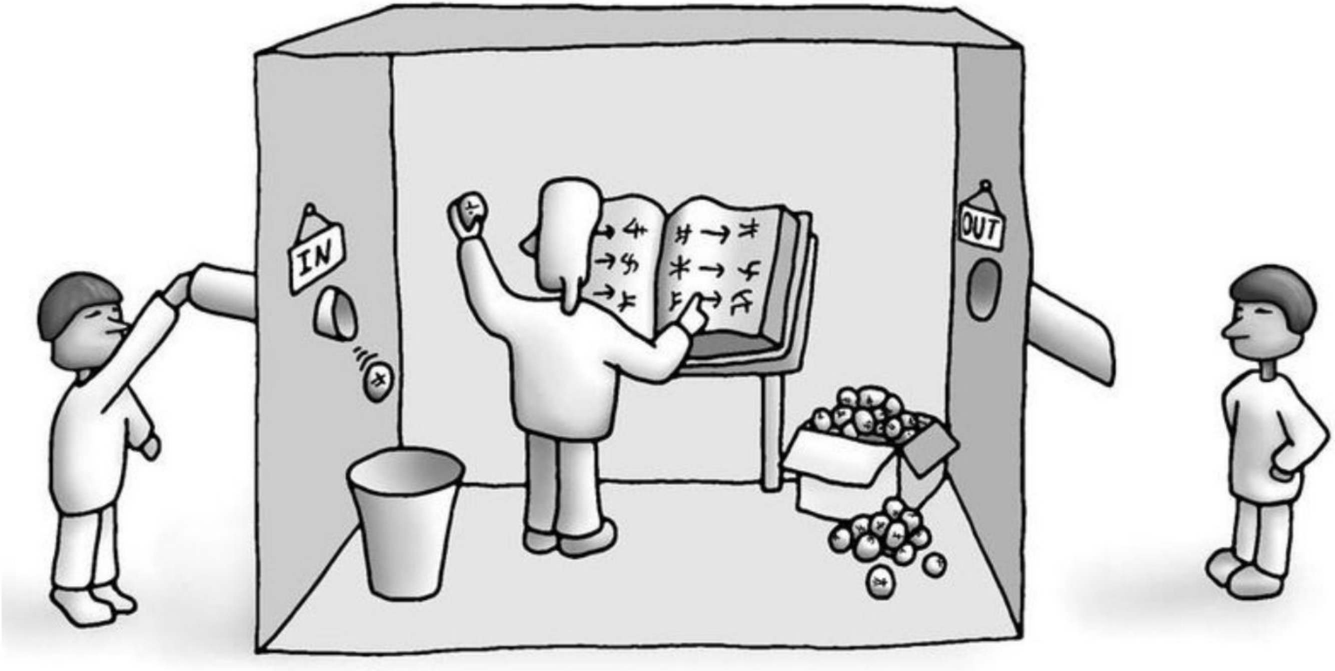
Can machine feel as humans do?

The concept of **feeling or emotions** for machines are similar to ***language understanding***, which itself doesn’t have a precise definition.

Language understanding is sometimes defined as to respond appropriately i.e. to take appropriate action based on the statement.

This has been a point for philosophical debates, about AI.

The Chinese Room, (John Searle 1980)



How can we make a machine intelligent ?

The Problem:

Find Chandler?

Setup

+

Python, Numpy Tutorial

1484
50

Chandler

Not Chandler

Not Chandler

Not Chandler

What was that?

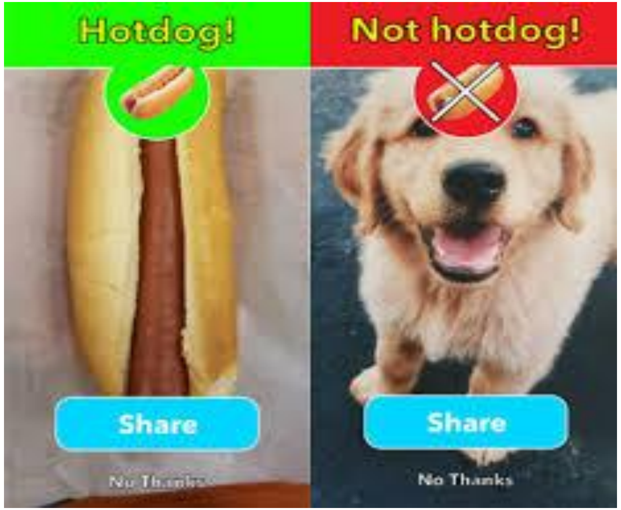
What was that?

Classifying, right?

What was that?

Classifying, right?

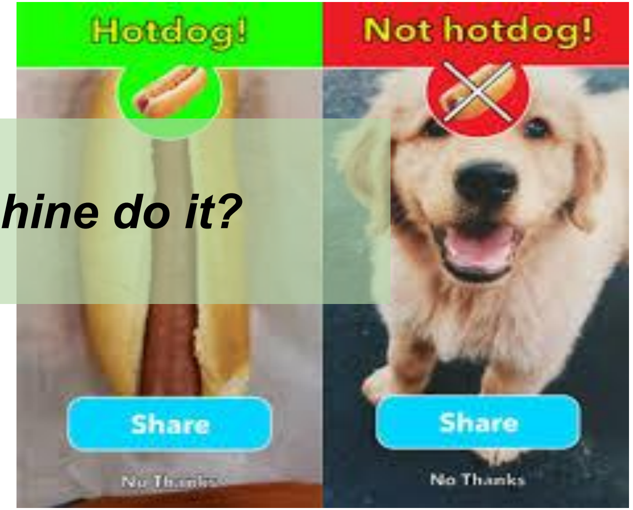
- Classification problems eg. Cat vs Dog, hot-dog vs not-hot dog, etc.



What was that?

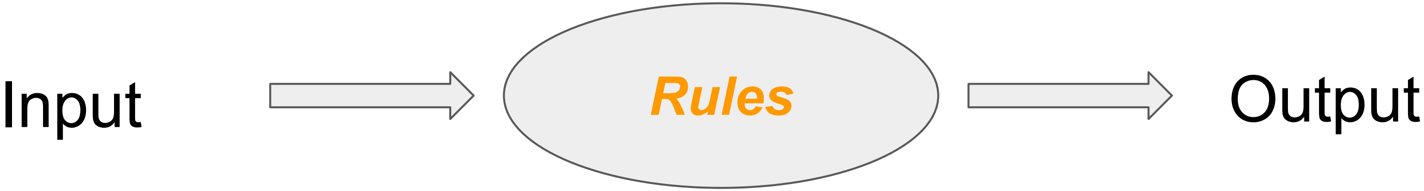
Classifying, right?

- Classification problems eg. Cat vs Dog, hot-dog vs not-hot dog, etc.



How do we make a machine do it?

Basics of computation

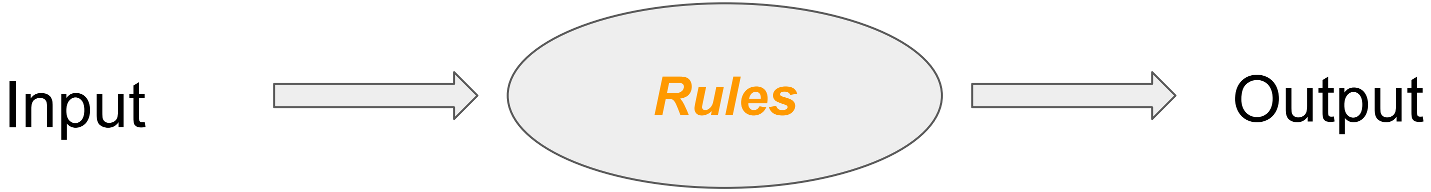


Input

Rules

Output

Basics of computation



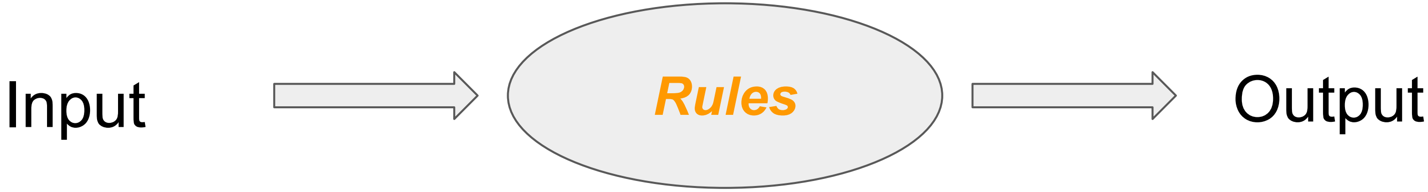
Input

Rules

Output

But, how do we teach machine learn to find (recognize) Chandler?

Basics of computation



Input

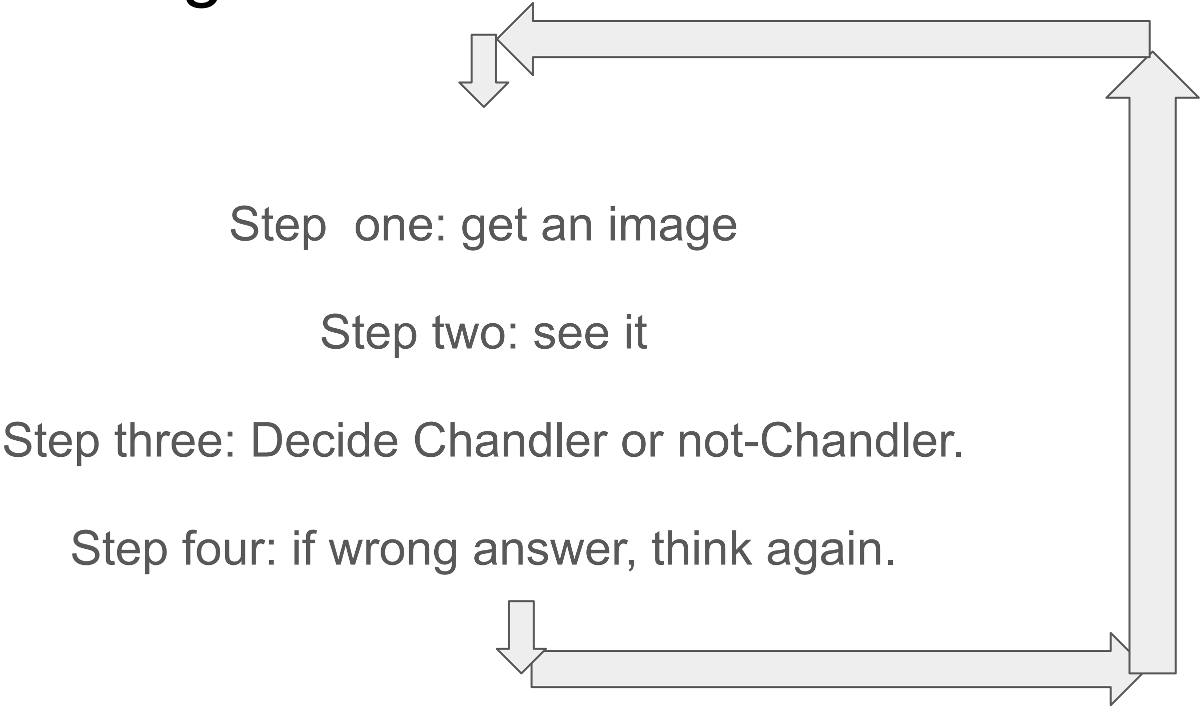
Rules

Output

But, how do we teach machine learn to find (recognize) Chandler?

Teach the machine the **RULES**, to find Chandler!

Let’s devise an algorithm:



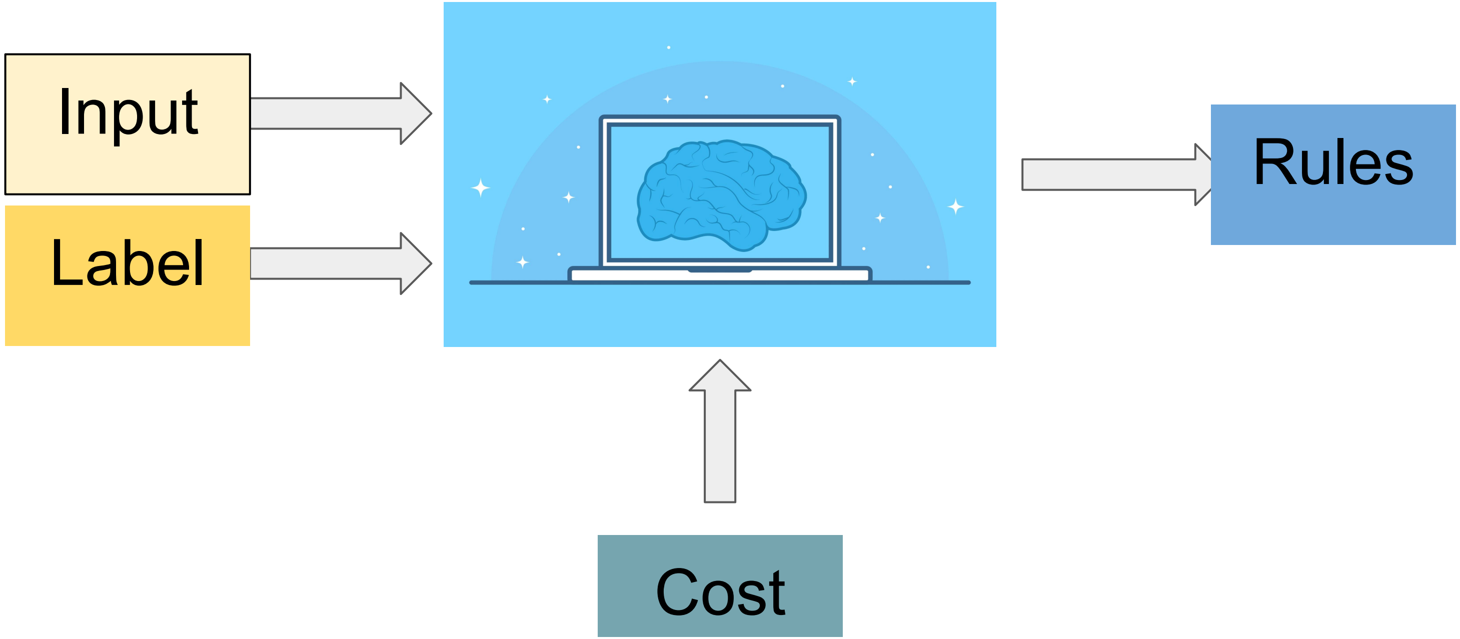
Step one: get an image

Step two: see it

Step three: Decide Chandler or not-Chandler.

Step four: if wrong answer, think again.

Supervised Learning



Input

Label

Rules

Cost

Rules = f(Input , Label)

Chan-Chan man ?

Find Chandler?

This is a challenge based on Natural Language Understanding where the task is to find whether a given **sentence** is **“sarcastic”** or **“non - sarcastic”**.

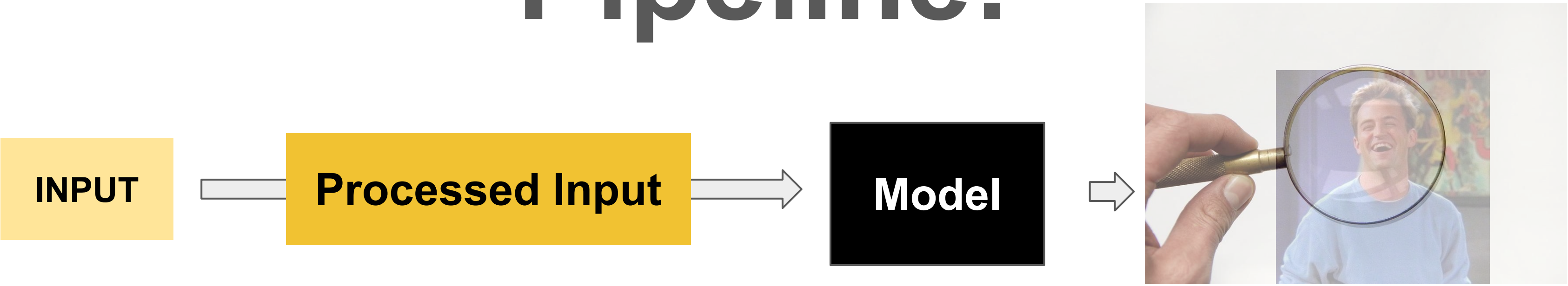
Maths Time

We know computer deals with number (in binary 0 and 1) but numbers.

So, we have to provide our input in some number format

It will give the answer in some numerical representation - which we will interpret as needed.

Pipeline:



INPUT

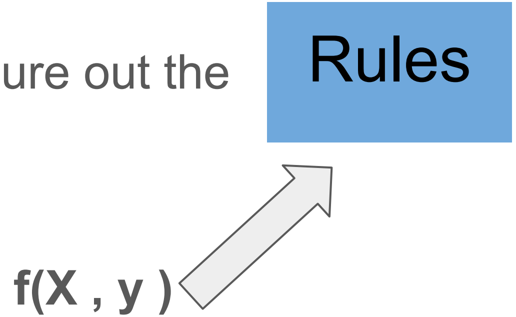
Processed Input

Model

X: input (photo)

Y: output (chandler or not)

Give **X** and **y** to machine and let it figure out the

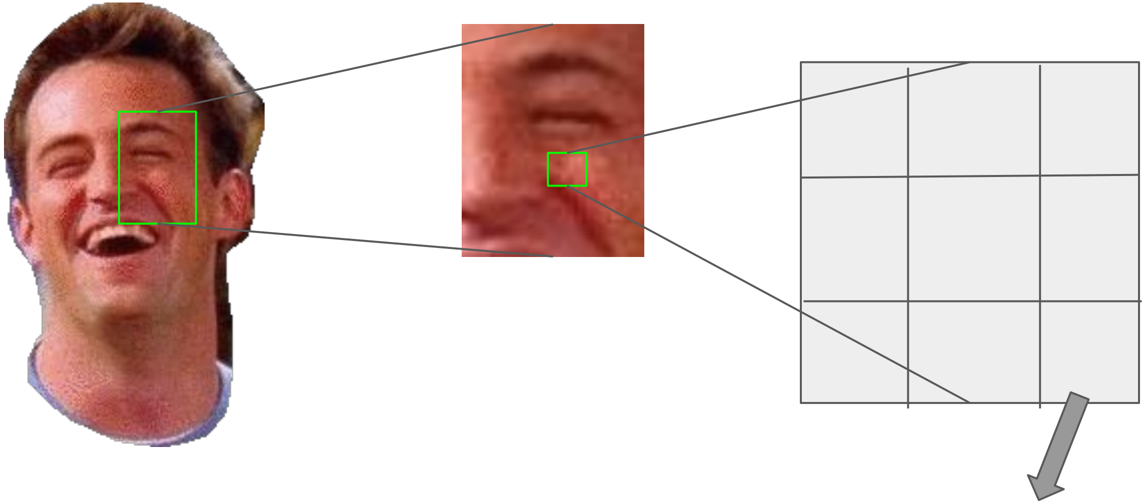


Rules

f(X , y)

so machine needs to figure out the relation f(.) for a given set of X’s and y’s.

Numerical representation



(205,85,56)

Matrices containing values corresponding to the RGB values.

Numerical representation

Chandler

Not-Chandler

	Class
Not-Chandler	0
Chandler	1

Final problem formulation

- Given set of **dialogue** and **labels**;

Dialogues are sentences (text),

label are 1 or 0 i.e. **sarcastic and non-sarcastic**.

Convert text to numerical form by cleaning (if needed) and some preprocessing

Text Preprocessing

Select way to find final a way to get the Rules

Input Preprocessing

Type of Input:

A Sentence (sequence of words) and **a label (0 or 1)**.

“*You should go back to the planet you came from.*”

Label - 1 (i.e. Sarcastic)

How to feed this sentence to the machine?

Answer: Tokenization

Tokenization

Was that place the Sun?

546 83 198 157 200 15

	Class
Not-Chandler	0
Chandler	1

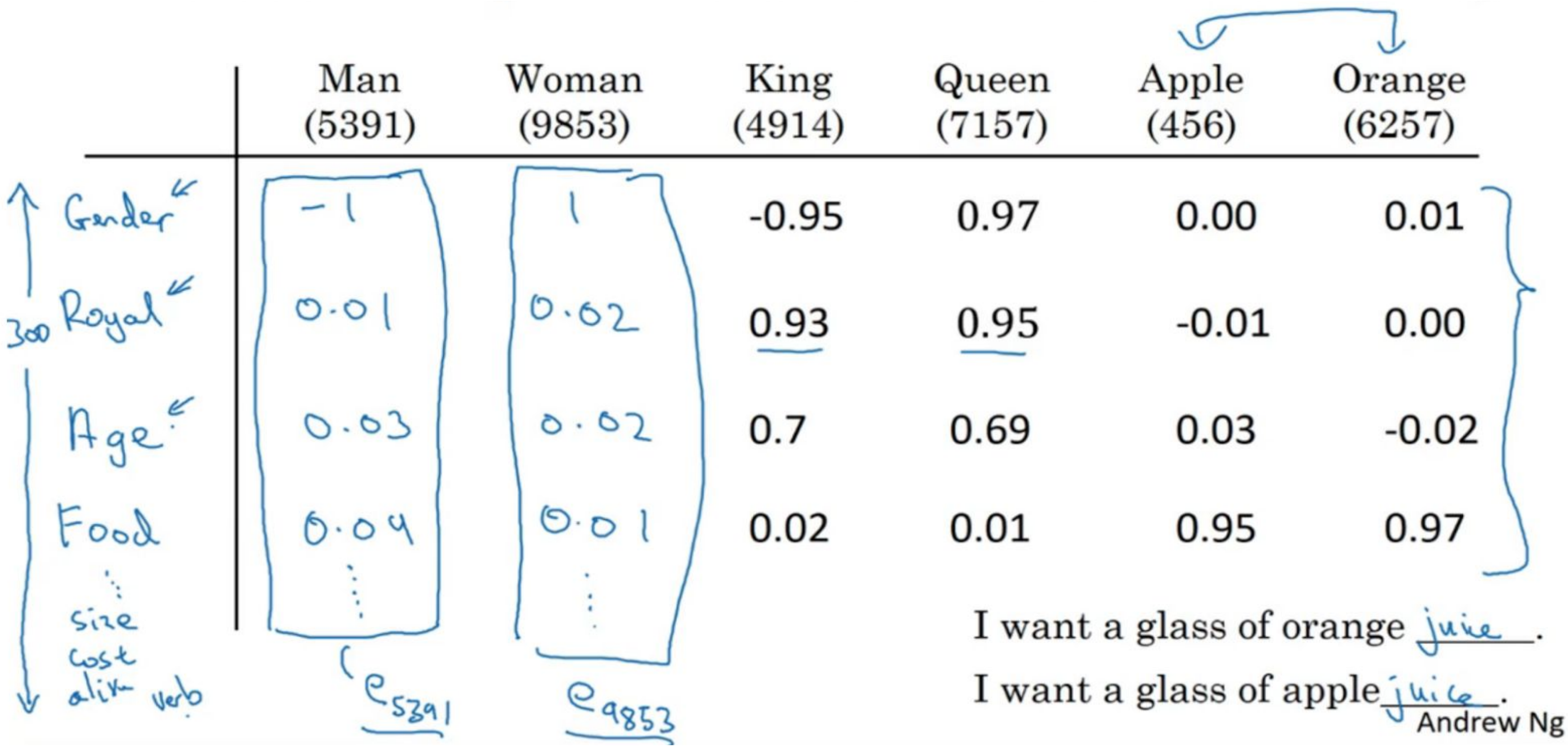
a	**0**
ab	**1**
.	**.**
?	**15**
.	**.**
that	**83**
.	**.**
the	**157**
Sun	**200**
.	
was	**546**
.	**.**
.	**.**
Zulu	**15268**
.	**.**

- Tokenization is the idea of representing each word with a number. The corresponding number to word mapping is stored as a dictionary.

Problem:

1. **But how will the machine understand what is being said?**
2. **How to compare two sentences based on their meanings?**

Word Vector Representation



	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	-1	1	-0.95	0.97	0.00	0.01
300 Royal	0.01	0.02	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.7	0.69	0.03	-0.02
Food	0.04	0.01	0.02	0.01	0.95	0.97
⋮	⋮	⋮				
size cost alive verb						

e_{5391} e_{9853}

I want a glass of orange juice.

I want a glass of apple juice.

Andrew Ng

Word Vectors

Each word is represented by a vector(a list of features describing the word).

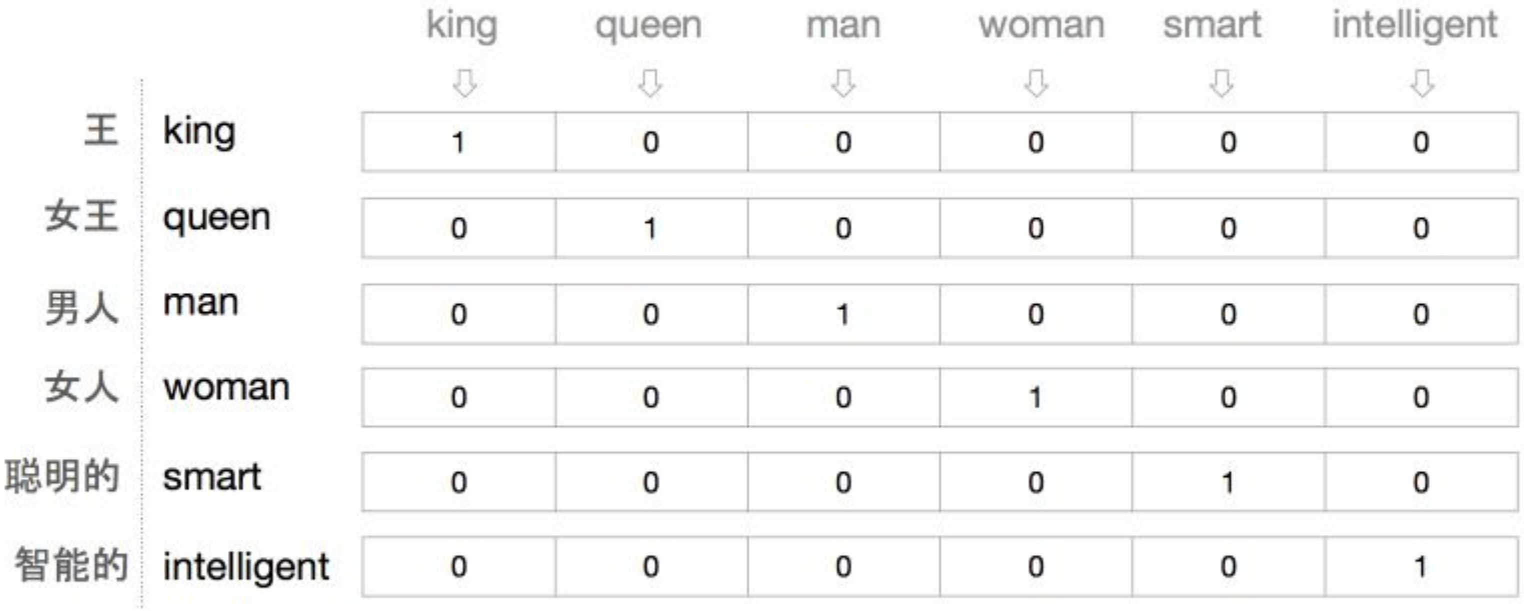
Eg. ID cards contain your info. Two Id cards can be compared to see which people share the same branch, similar names, etc.

Word vectors are similar to ID cards for words, containing essential info that can be compared to see if two words are similar.

Words with similar meaning will have similar word vectors.

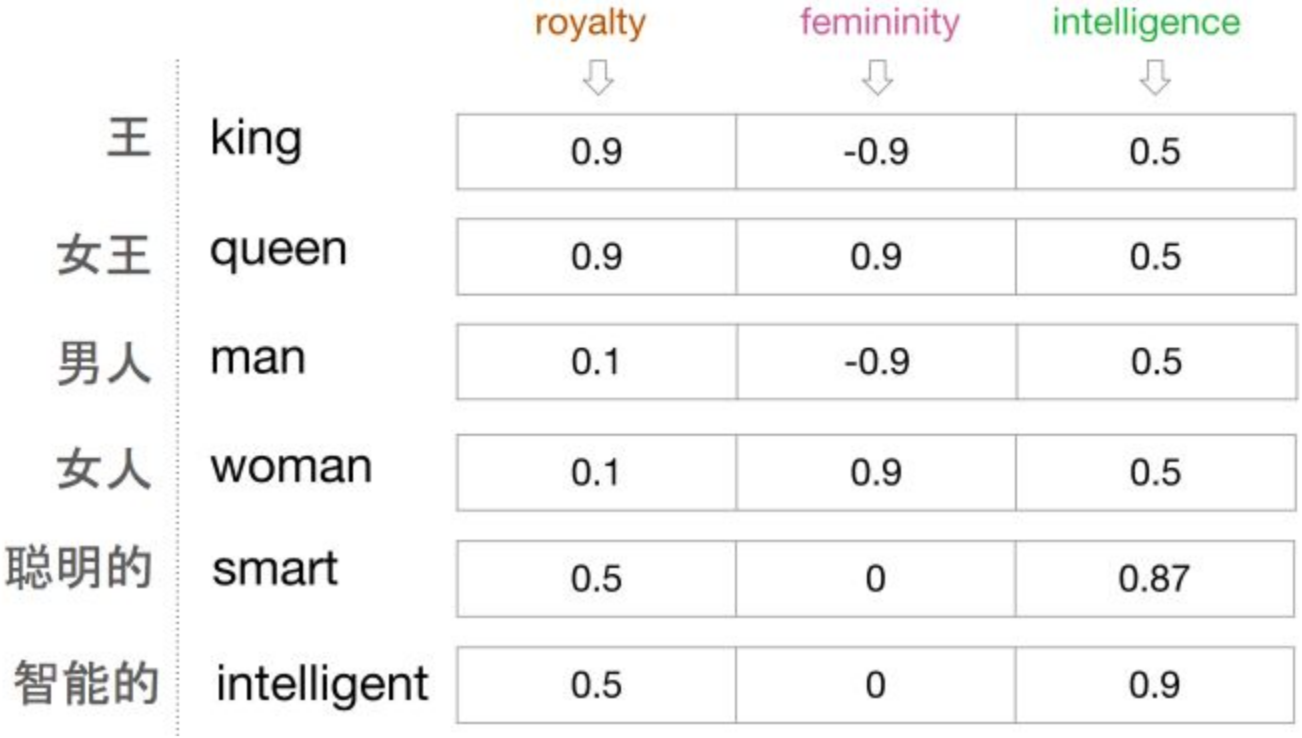
What do Word Vectors look like?

Word vectors can be simple as the ones below



		king	queen	man	woman	smart	intelligent
		⇩	⇩	⇩	⇩	⇩	⇩
王	king	1	0	0	0	0	0
女王	queen	0	1	0	0	0	0
男人	man	0	0	1	0	0	0
女人	woman	0	0	0	1	0	0
聪明的	smart	0	0	0	0	1	0
智能的	intelligent	0	0	0	0	0	1

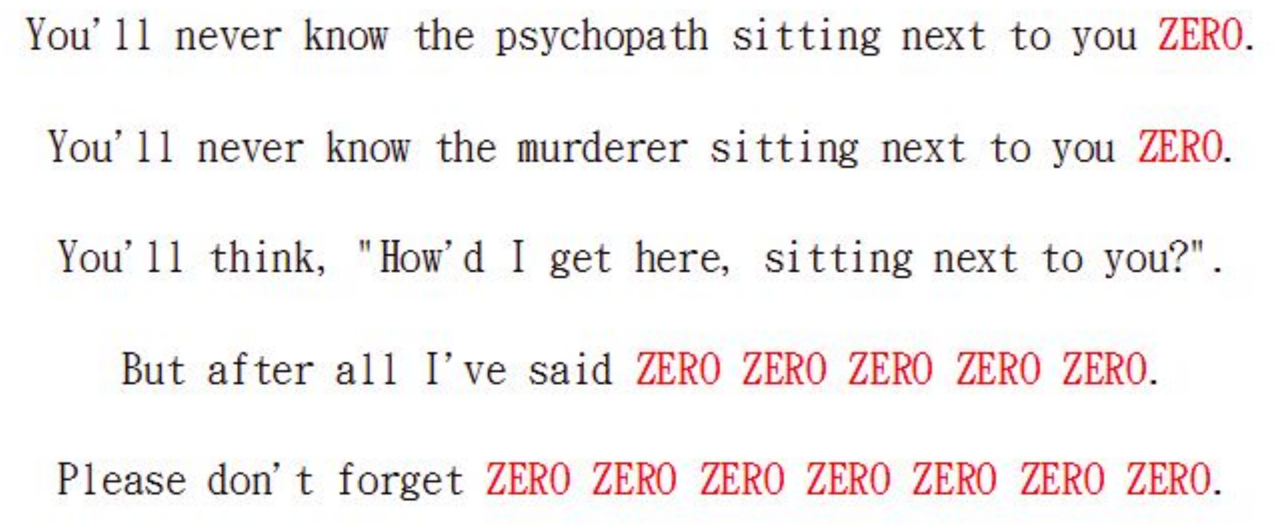
Or can be complex like the ones below



		royalty	femininity	intelligence
王	king	0.9	-0.9	0.5
女王	queen	0.9	0.9	0.5
男人	man	0.1	-0.9	0.5
女人	woman	0.1	0.9	0.5
聪明的	smart	0.5	0	0.87
智能的	intelligent	0.5	0	0.9

Other preprocessing steps

Padding: Sentences are often padded with zeros to make their length same and make the input consistent for the machine.



You’ll never know the psychopath sitting next to you ZERO.

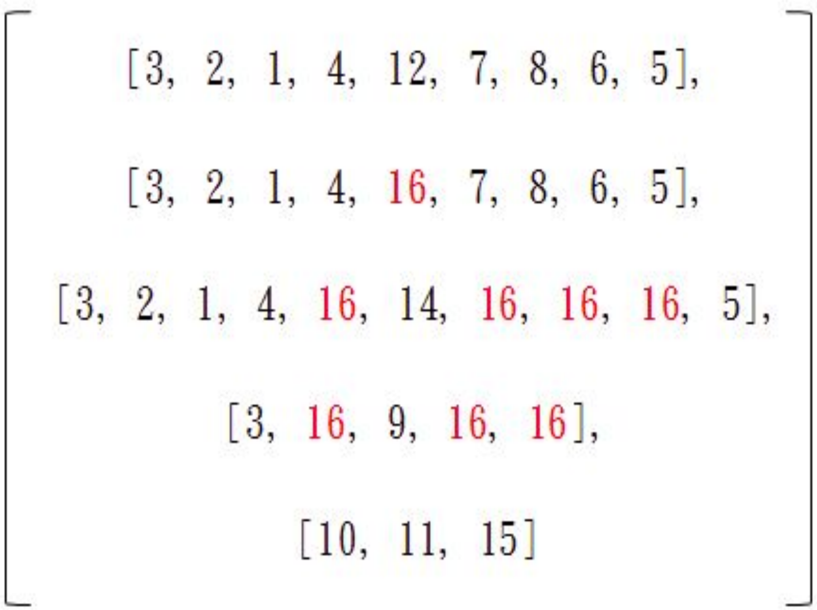
You’ll never know the murderer sitting next to you ZERO.

You’ll think, "How’d I get here, sitting next to you?".

But after all I’ve said ZERO ZERO ZERO ZERO ZERO.

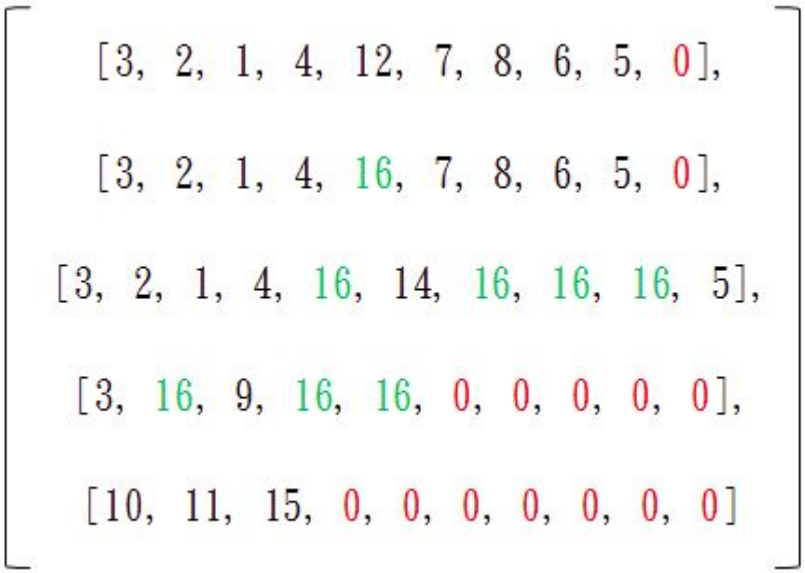
Please don’t forget ZERO ZERO ZERO ZERO ZERO ZERO ZERO.

Before Padding



```
[
  [3, 2, 1, 4, 12, 7, 8, 6, 5],
  [3, 2, 1, 4, 16, 7, 8, 6, 5],
  [3, 2, 1, 4, 16, 14, 16, 16, 16, 5],
  [3, 16, 9, 16, 16],
  [10, 11, 15]
]
```

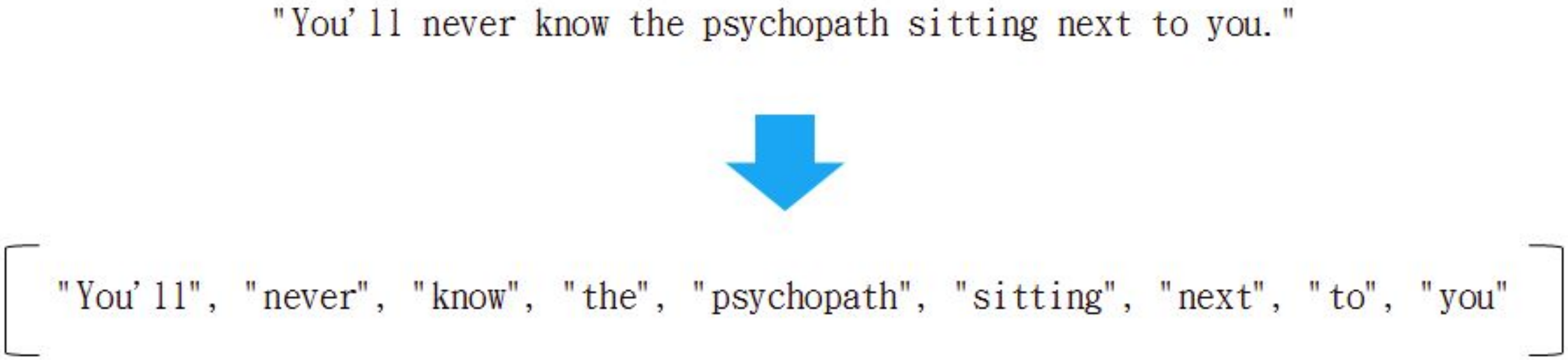
After Padding



```
[
  [3, 2, 1, 4, 12, 7, 8, 6, 5, 0],
  [3, 2, 1, 4, 16, 7, 8, 6, 5, 0],
  [3, 2, 1, 4, 16, 14, 16, 16, 16, 5],
  [3, 16, 9, 16, 16, 0, 0, 0, 0, 0],
  [10, 11, 15, 0, 0, 0, 0, 0, 0, 0]
]
```

Preprocessing Pipeline Summary

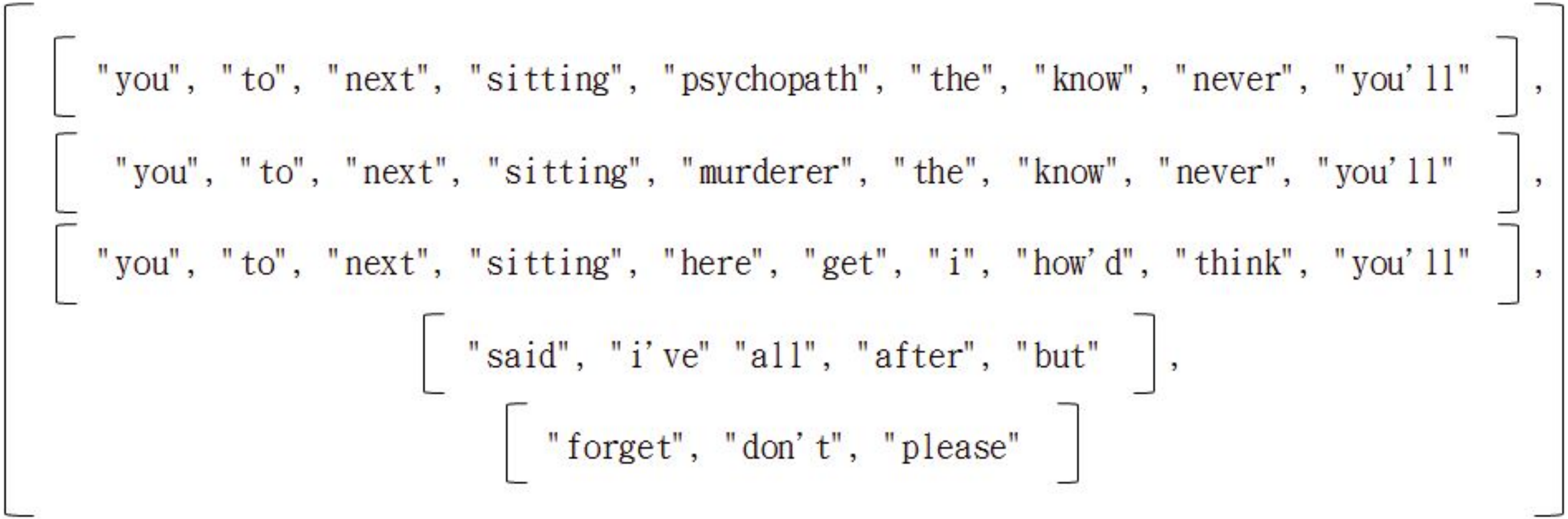
Break the sentence into individual words (or array of words)



"You’ll never know the psychopath sitting next to you."

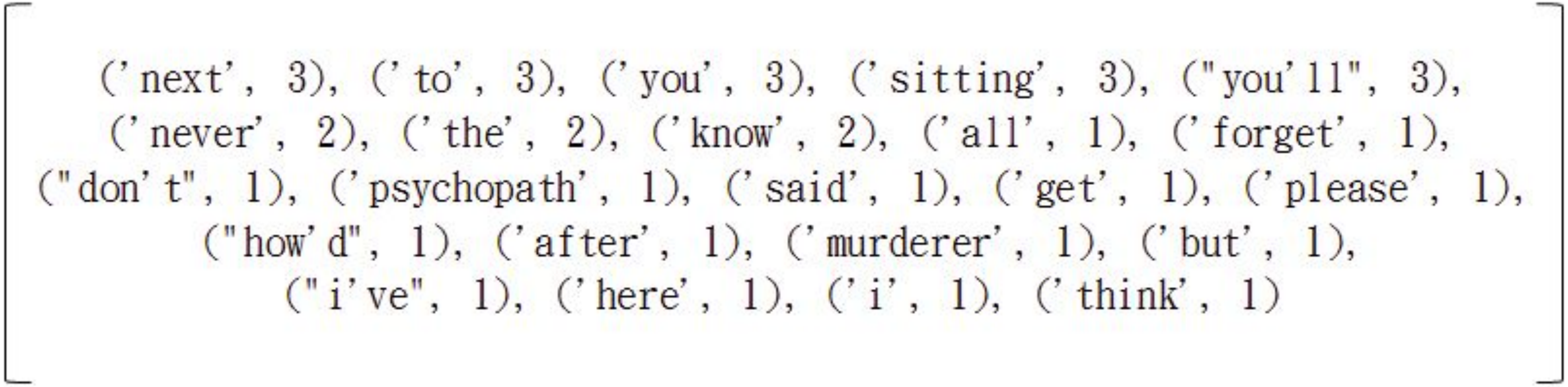
["You’ll", "never", "know", "the", "psychopath", "sitting", "next", "to", "you"]

In case of multiple sentences, use array of sentences



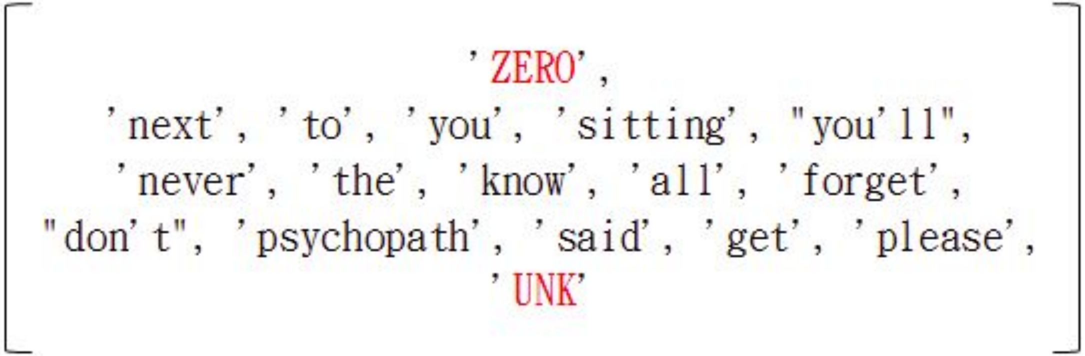
```
[
  [ "you", "to", "next", "sitting", "psychopath", "the", "know", "never", "you’ll" ],
  [ "you", "to", "next", "sitting", "murderer", "the", "know", "never", "you’ll" ],
  [ "you", "to", "next", "sitting", "here", "get", "i", "how’d", "think", "you’ll" ],
  [ "said", "i’ve" "all", "after", "but" ],
  [ "forget", "don’t", "please" ]
]
```

Assign index/number to each word



```
[
('next', 3), ('to', 3), ('you', 3), ('sitting', 3), ("you'll", 3),
('never', 2), ('the', 2), ('know', 2), ('all', 1), ('forget', 1),
("don't", 1), ('psychopath', 1), ('said', 1), ('get', 1), ('please', 1),
("how'd", 1), ('after', 1), ('murderer', 1), ('but', 1),
("i've", 1), ('here', 1), ('i', 1), ('think', 1)
]
```

Create a vocabulary (of all the words in the text)



[

’ZERO’,

’next’, ’to’, ’you’, ’sitting’, "you’ll",

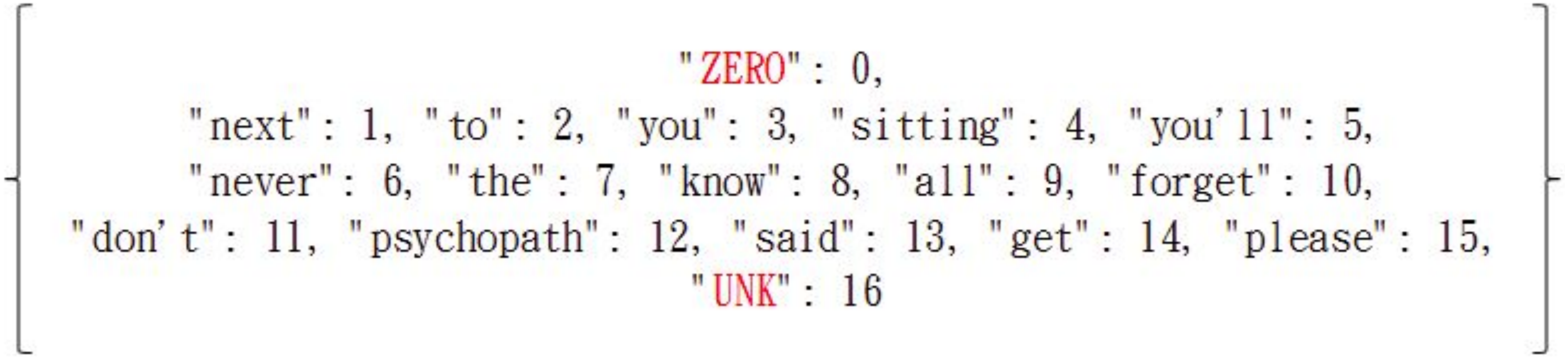
’never’, ’the’, ’know’, ’all’, ’forget’,

"don’t", ’psychopath’, ’said’, ’get’, ’please’,

’UNK’

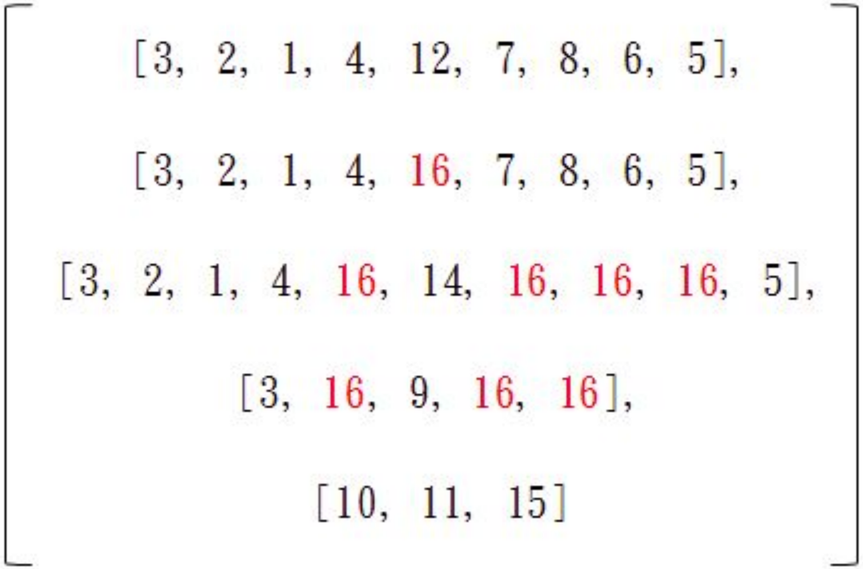
]

Create a word-index mapping (dictionary)



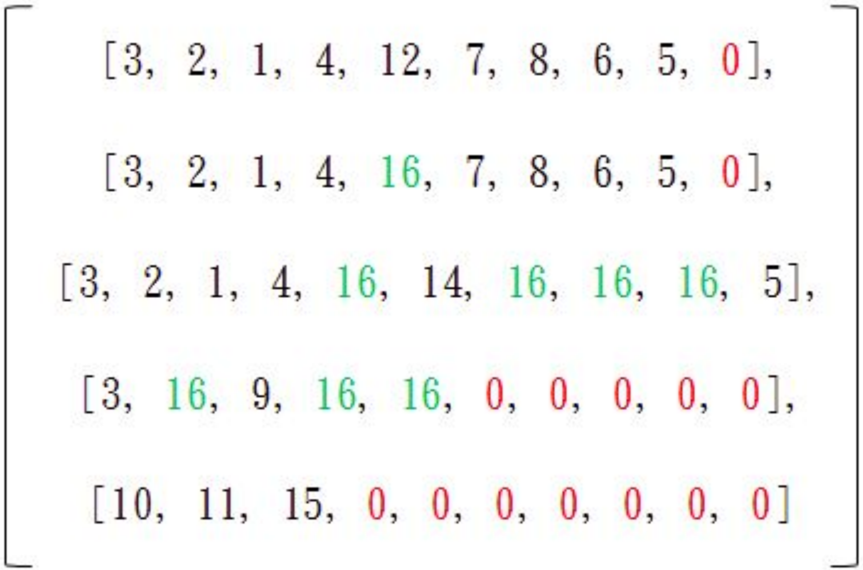
```
{
                              "ZERO": 0,
       "next": 1, "to": 2, "you": 3, "sitting": 4, "you’ll": 5,
       "never": 6, "the": 7, "know": 8, "all": 9, "forget": 10,
"don’t": 11, "psychopath": 12, "said": 13, "get": 14, "please": 15,
                              "UNK": 16
}
```

Tokenize the sentences



```
[3, 2, 1, 4, 12, 7, 8, 6, 5],
[3, 2, 1, 4, 16, 7, 8, 6, 5],
[3, 2, 1, 4, 16, 14, 16, 16, 16, 5],
[3, 16, 9, 16, 16],
[10, 11, 15]
```

Add zero padding



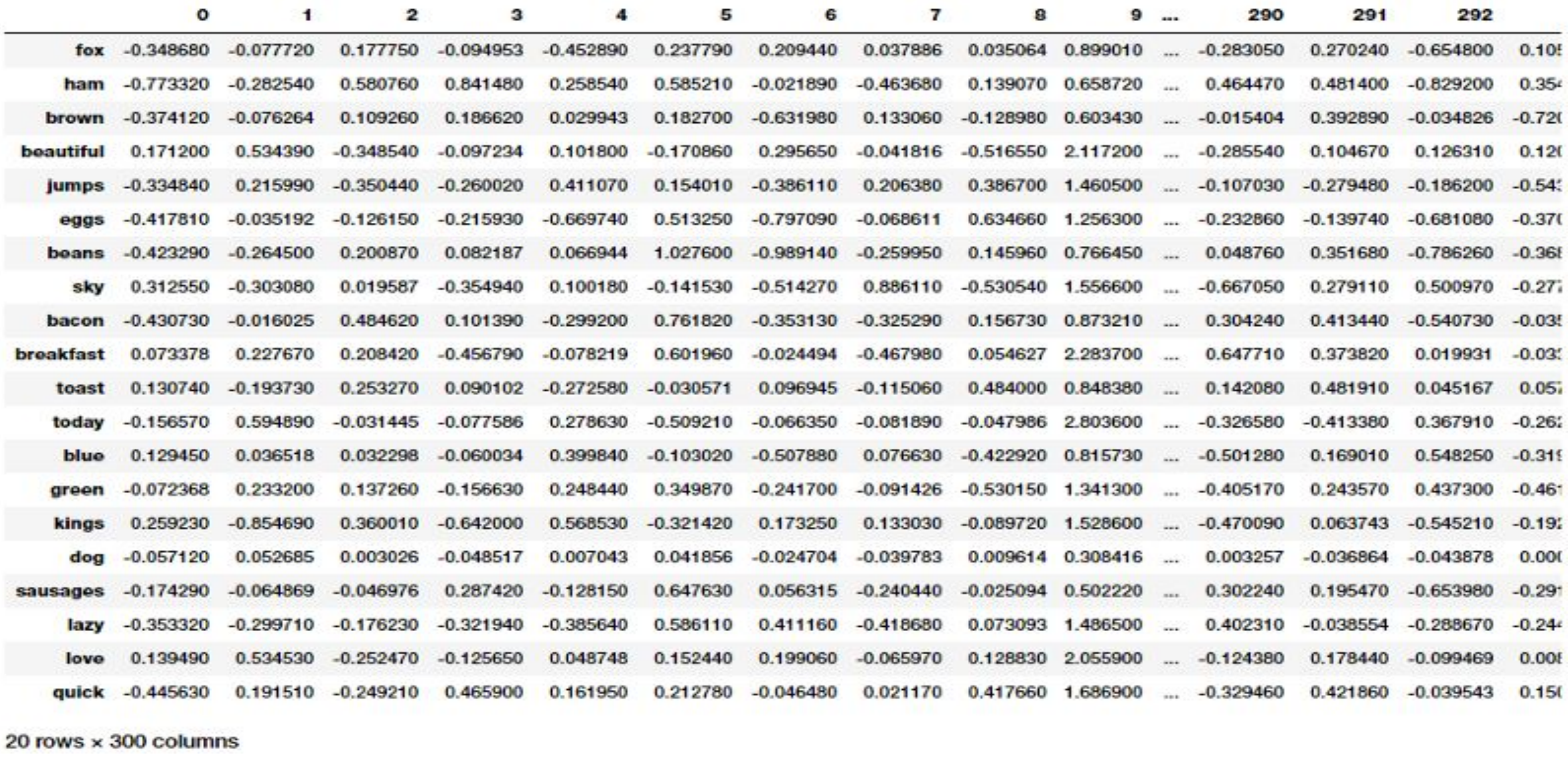
```
[3,  2,  1,  4,  12,  7,  8,  6,  5,  0],
[3,  2,  1,  4,  16,  7,  8,  6,  5,  0],
[3,  2,  1,  4,  16,  14,  16,  16,  16,  5],
[3,  16,  9,  16,  16,  0,  0,  0,  0,  0],
[10,  11,  15,  0,  0,  0,  0,  0,  0,  0]
```

Convert indices to word vectors

[3, 2, 1, 4, 16, 7, 8, 6, 5, 0]

$$\begin{pmatrix}0\\0\\0\\1\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\end{pmatrix} \begin{pmatrix}0\\0\\1\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\end{pmatrix} \begin{pmatrix}0\\1\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\end{pmatrix} \begin{pmatrix}0\\0\\0\\0\\1\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\end{pmatrix} \begin{pmatrix}0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\1\end{pmatrix} \begin{pmatrix}0\\0\\0\\0\\0\\0\\0\\1\\0\\0\\0\\0\\0\\0\\0\\0\\0\end{pmatrix} \begin{pmatrix}0\\0\\0\\0\\0\\0\\0\\0\\1\\0\\0\\0\\0\\0\\0\\0\\0\end{pmatrix} \begin{pmatrix}0\\0\\0\\0\\0\\0\\1\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\end{pmatrix} \begin{pmatrix}0\\0\\0\\0\\0\\1\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\end{pmatrix} \begin{pmatrix}1\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\end{pmatrix}$$

The embeddings shown above are too simple and we would encourage you to use more complex pre-trained embeddings like GloVE, Word2vec etc. in order to get better results. Figure shows 300d Glove embeddings.



	0	1	2	3	4	5	6	7	8	9	...	290	291	292	
fox	-0.348680	-0.077720	0.177750	-0.094953	-0.452890	0.237790	0.209440	0.037886	0.035064	0.899010	...	-0.283050	0.270240	-0.654800	0.10
ham	-0.773320	-0.282540	0.580760	0.841480	0.258540	0.585210	-0.021890	-0.463680	0.139070	0.658720	...	0.464470	0.481400	-0.829200	0.35
brown	-0.374120	-0.076264	0.109260	0.186620	0.029943	0.182700	-0.631980	0.133060	-0.128980	0.603430	...	-0.015404	0.392890	-0.034826	-0.72
beautiful	0.171200	0.534390	-0.348540	-0.097234	0.101800	-0.170860	0.295650	-0.041816	-0.516550	2.117200	...	-0.285540	0.104670	0.126310	0.12
jumps	-0.334840	0.215990	-0.350440	-0.260020	0.411070	0.154010	-0.386110	0.206380	0.386700	1.460500	...	-0.107030	-0.279480	-0.186200	-0.54
eggs	-0.417810	-0.035192	-0.126150	-0.215930	-0.669740	0.513250	-0.797090	-0.068611	0.634660	1.256300	...	-0.232860	-0.139740	-0.681080	-0.37
beans	-0.423290	-0.264500	0.200870	0.082187	0.066944	1.027600	-0.989140	-0.259950	0.145960	0.766450	...	0.048760	0.351680	-0.786260	-0.36
sky	0.312550	-0.303080	0.019587	-0.354940	0.100180	-0.141530	-0.514270	0.886110	-0.530540	1.556600	...	-0.667050	0.279110	0.500970	-0.27
bacon	-0.430730	-0.016025	0.484620	0.101390	-0.299200	0.761820	-0.353130	-0.325290	0.156730	0.873210	...	0.304240	0.413440	-0.540730	-0.03
breakfast	0.073378	0.227670	0.208420	-0.456790	-0.078219	0.601960	-0.024494	-0.467980	0.054627	2.283700	...	0.647710	0.373820	0.019931	-0.03
toast	0.130740	-0.193730	0.253270	0.090102	-0.272580	-0.030571	0.096945	-0.115060	0.484000	0.848380	...	0.142080	0.481910	0.045167	0.05
today	-0.156570	0.594890	-0.031445	-0.077586	0.278630	-0.509210	-0.066350	-0.081890	-0.047986	2.803600	...	-0.326580	-0.413380	0.367910	-0.26
blue	0.129450	0.036518	0.032298	-0.060034	0.399840	-0.103020	-0.507880	0.076630	-0.422920	0.815730	...	-0.501280	0.169010	0.548250	-0.31
green	-0.072368	0.233200	0.137260	-0.156630	0.248440	0.349870	-0.241700	-0.091426	-0.530150	1.341300	...	-0.405170	0.243570	0.437300	-0.46
kings	0.259230	-0.854690	0.360010	-0.642000	0.568530	-0.321420	0.173250	0.133030	-0.089720	1.528600	...	-0.470090	0.063743	-0.545210	-0.19
dog	-0.057120	0.052685	0.003026	-0.048517	0.007043	0.041856	-0.024704	-0.039783	0.009614	0.308416	...	0.003257	-0.036864	-0.043878	0.00
sausages	-0.174290	-0.064869	-0.046976	0.287420	-0.128150	0.647630	0.056315	-0.240440	-0.025094	0.502220	...	0.302240	0.195470	-0.653980	-0.29
lazy	-0.353320	-0.299710	-0.176230	-0.321940	-0.385640	0.586110	0.411160	-0.418680	0.073093	1.486500	...	0.402310	-0.038554	-0.288670	-0.24
love	0.139490	0.534530	-0.252470	-0.125650	0.048748	0.152440	0.199060	-0.065970	0.128830	2.055900	...	-0.124380	0.178440	-0.099469	0.00
quick	-0.445630	0.191510	-0.249210	0.465900	0.161950	0.212780	-0.046480	0.021170	0.417660	1.686900	...	-0.329460	0.421860	-0.039543	0.15

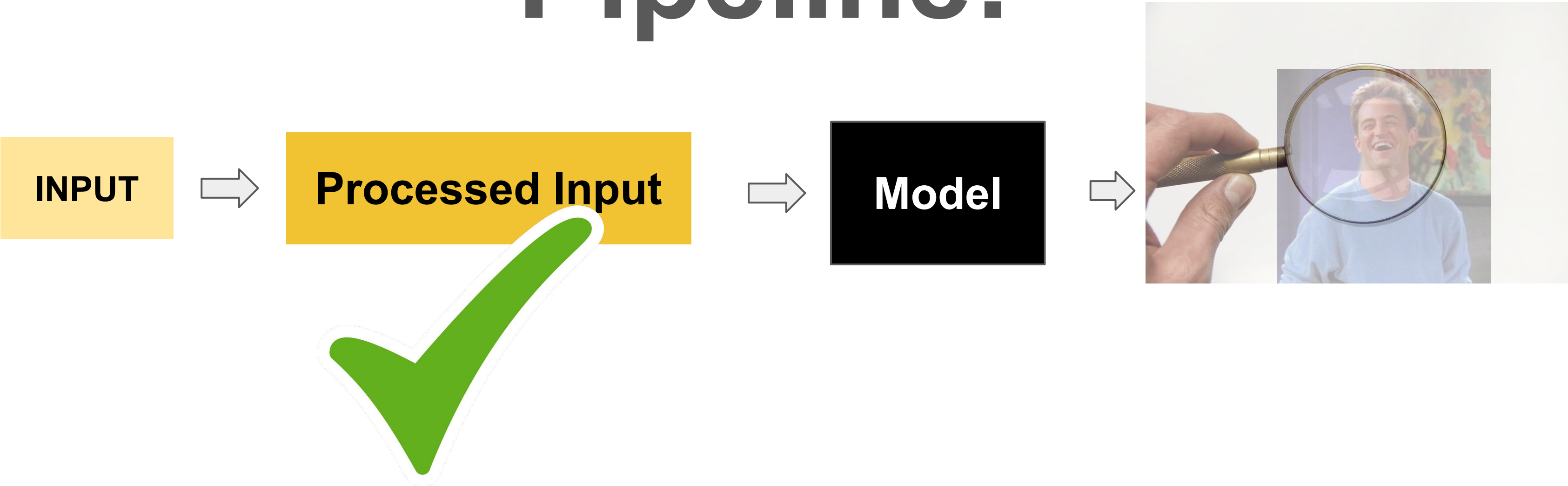
20 rows × 300 columns

Congrats!!!

Now you know how to preprocess text data.

Reference: https://chunml.github.io/ChunML.github.io/project/Sequence-To-Sequence

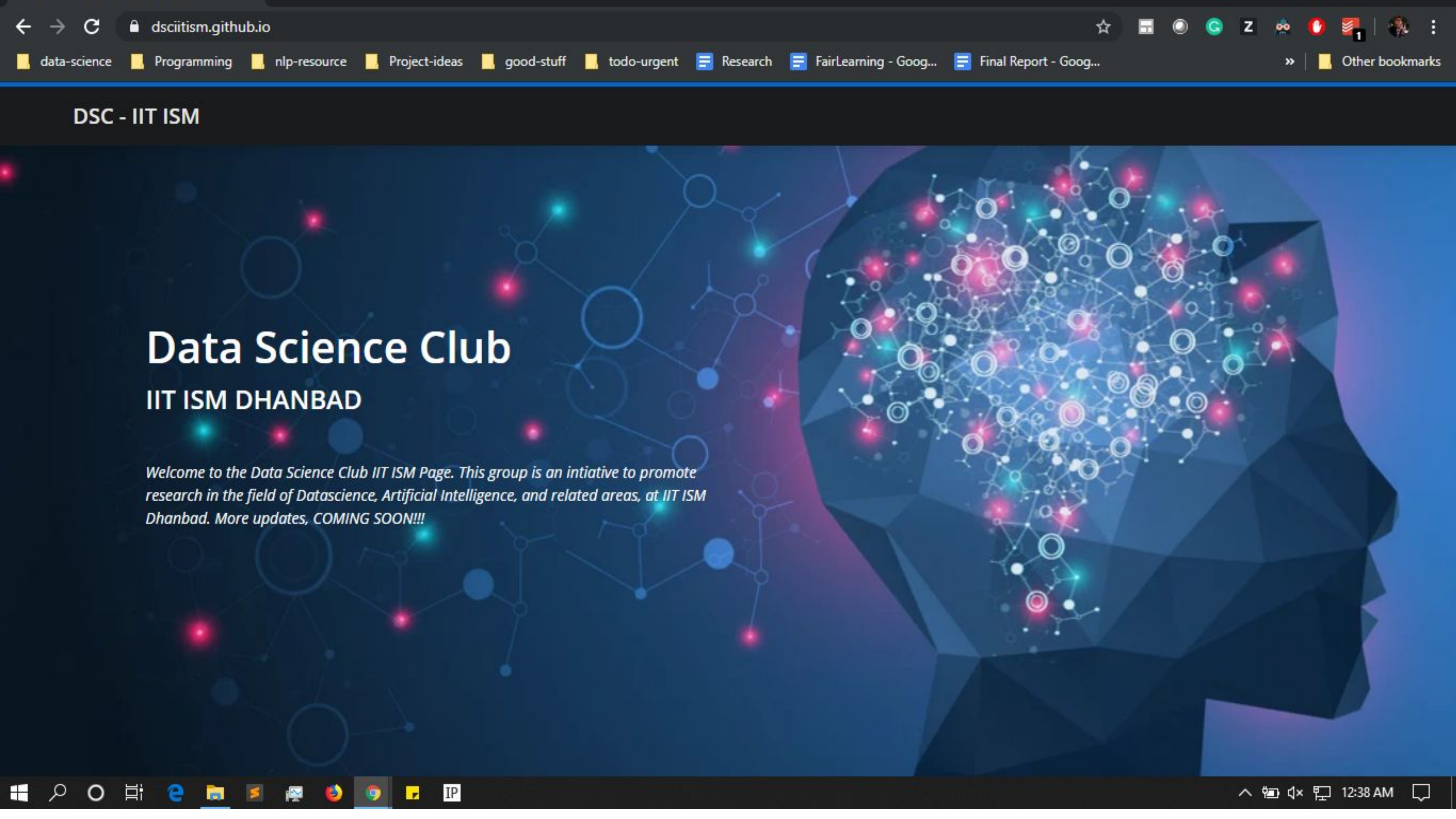
Pipeline:



INPUT → Processed Input ✓ → Model →

Datascience Club Page is live !

Kshitij Kumar



DSC - IIT ISM

Data Science Club

IIT ISM DHANBAD

Welcome to the Data Science Club IIT ISM Page. This group is an intiative to promote research in the field of Datascience, Artificial Intelligence, and related areas, at IIT ISM Dhanbad. More updates, COMING SOON!!!

NLP/ML Workshop

Today’s workflow outline:

- Recap
- Different Models
- Baseline Model Tutorial

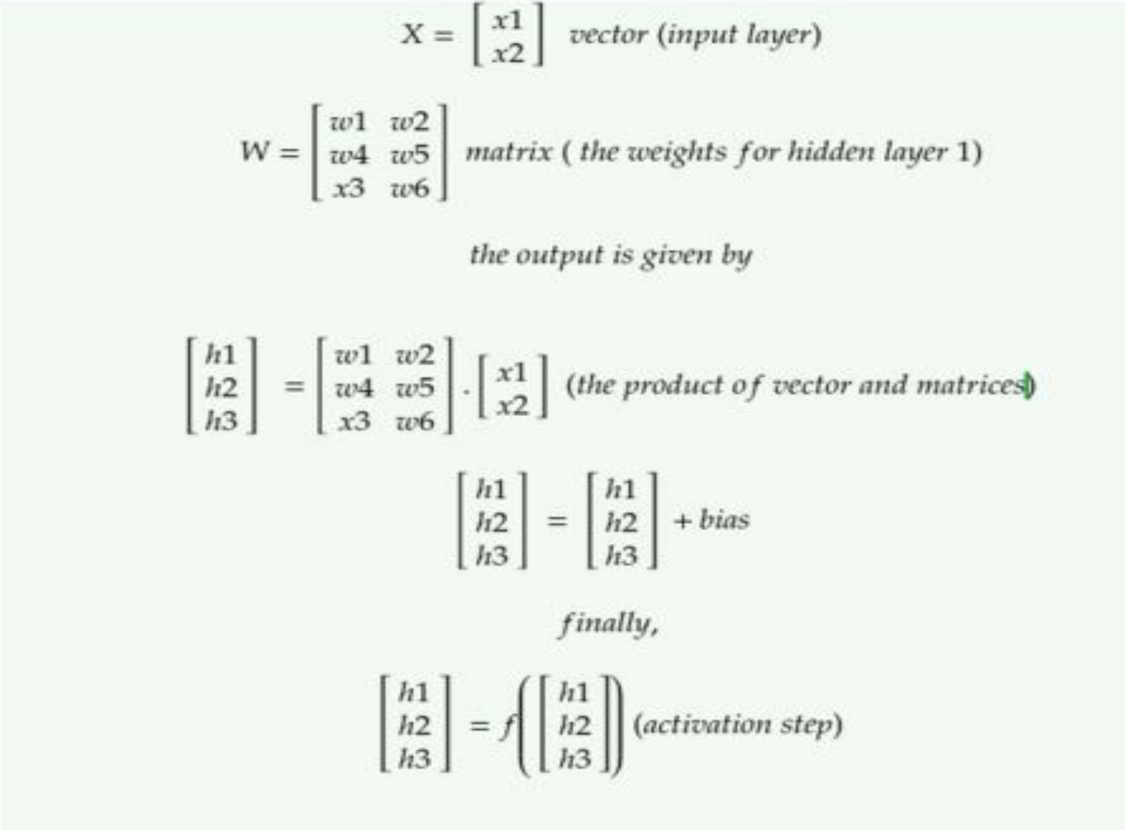
Text Processing

Tutorial

Text changed into numerical form.

Let’s address the classifier problem.

Numpy Review



$$X = \begin{bmatrix} x1 \\ x2 \end{bmatrix} \quad \textit{vector (input layer)}$$

$$W = \begin{bmatrix} w1 & w2 \\ w4 & w5 \\ x3 & w6 \end{bmatrix} \textit{matrix (the weights for hidden layer 1)}$$

the output is given by

$$\begin{bmatrix} h1 \\ h2 \\ h3 \end{bmatrix} = \begin{bmatrix} w1 & w2 \\ w4 & w5 \\ x3 & w6 \end{bmatrix} \cdot \begin{bmatrix} x1 \\ x2 \end{bmatrix} \textit{(the product of vector and matrices)}$$

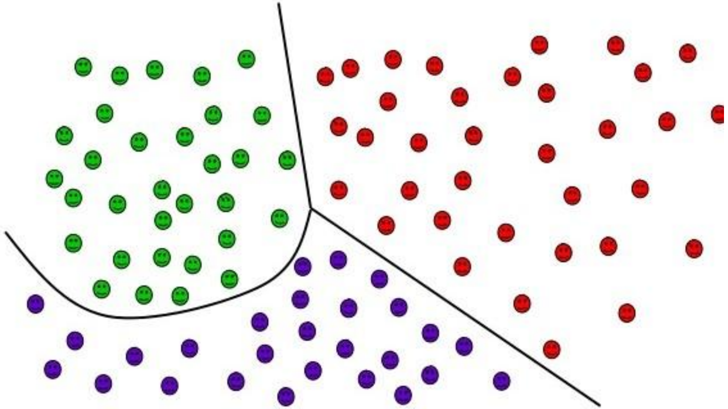
$$\begin{bmatrix} h1 \\ h2 \\ h3 \end{bmatrix} = \begin{bmatrix} h1 \\ h2 \\ h3 \end{bmatrix} + bias$$

finally,

$$\begin{bmatrix} h1 \\ h2 \\ h3 \end{bmatrix} = f\left(\begin{bmatrix} h1 \\ h2 \\ h3 \end{bmatrix}\right) \textit{(activation step)}$$

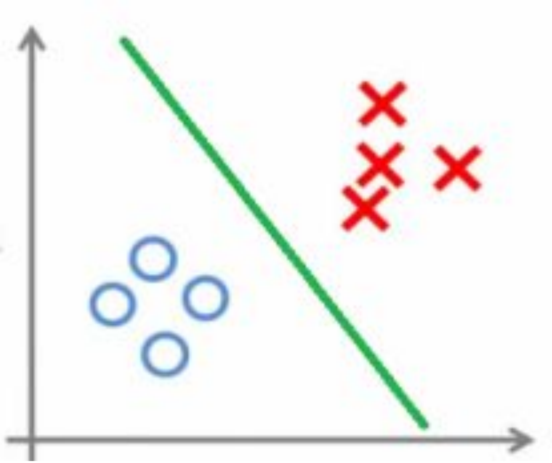
Classifier will help us to classify whether or not the given instance belongs to class A or class B.

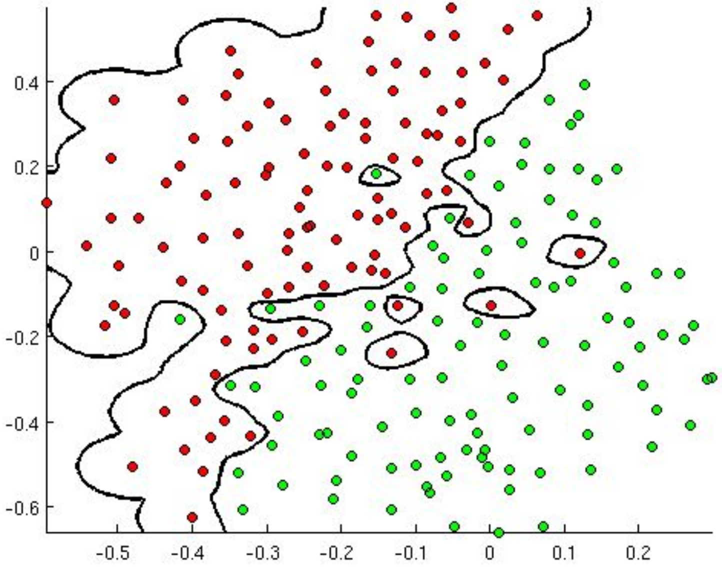
Note class maybe two (binary), or more (multi class) classification



Different ways to classify

- Draw a straight line
- Draw a curve line
- --





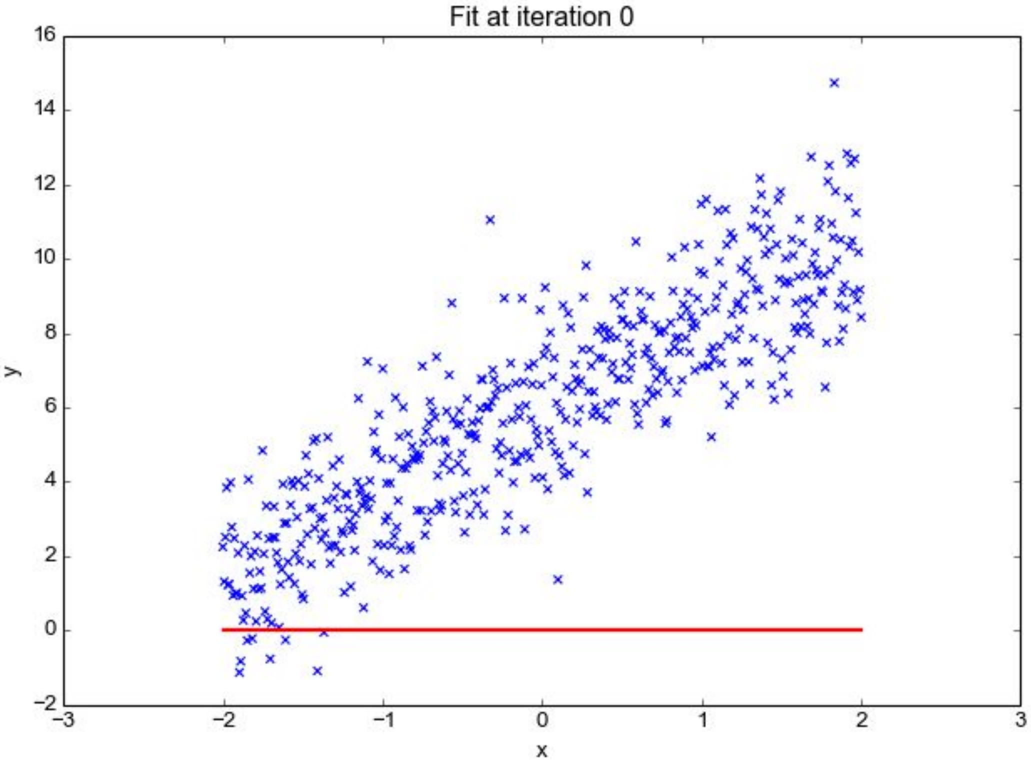
Linear Regression - helps making linear/ polynomial decision boundaries.

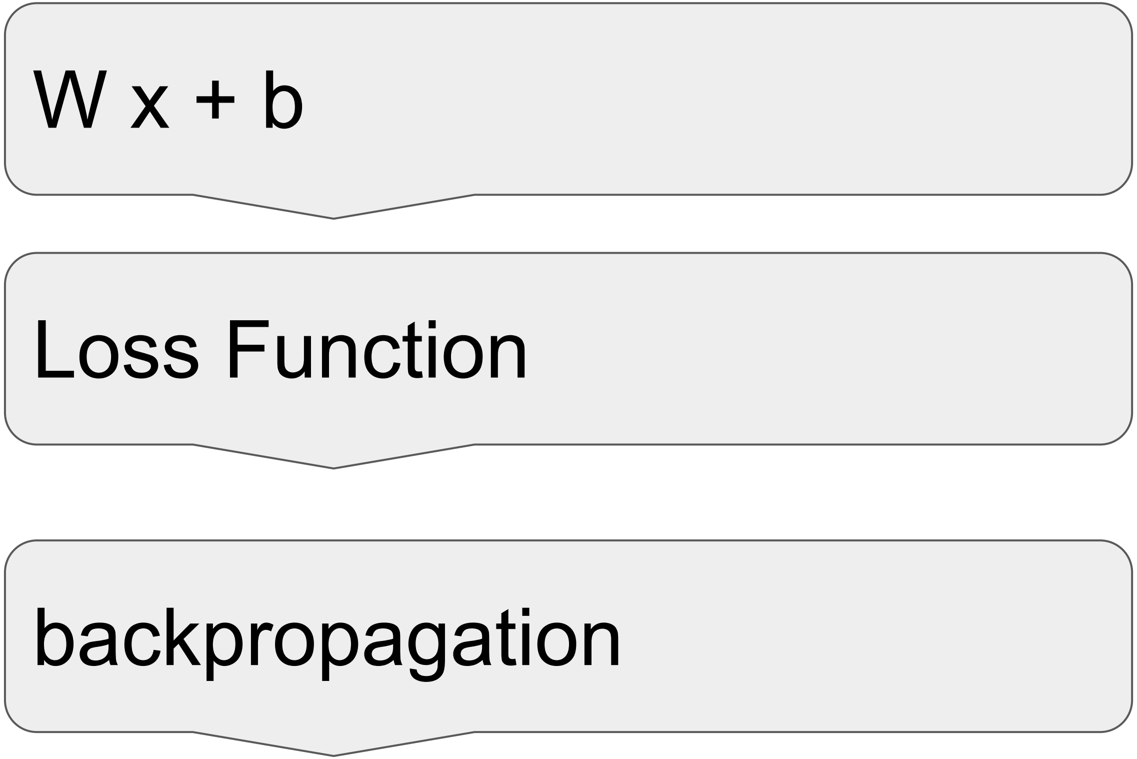
Logistic Regression - categorical values

Neural network - helps with more complex boundaries

Linear Regression

Concept





W x + b

Loss Function

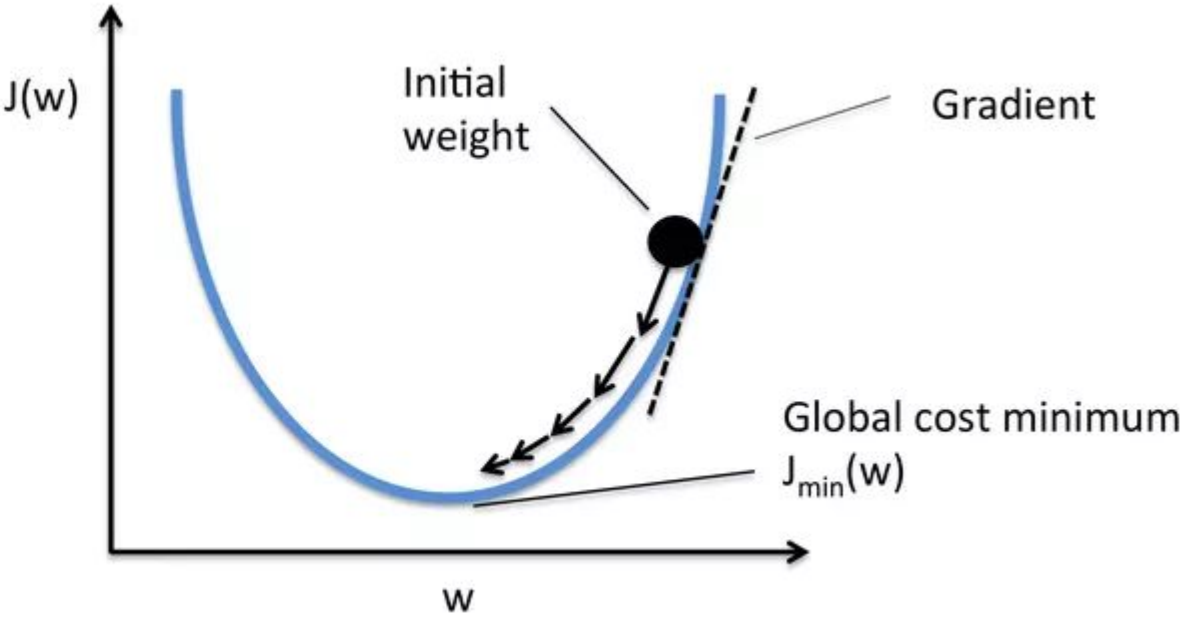
backpropagation

Loss/Cost Function

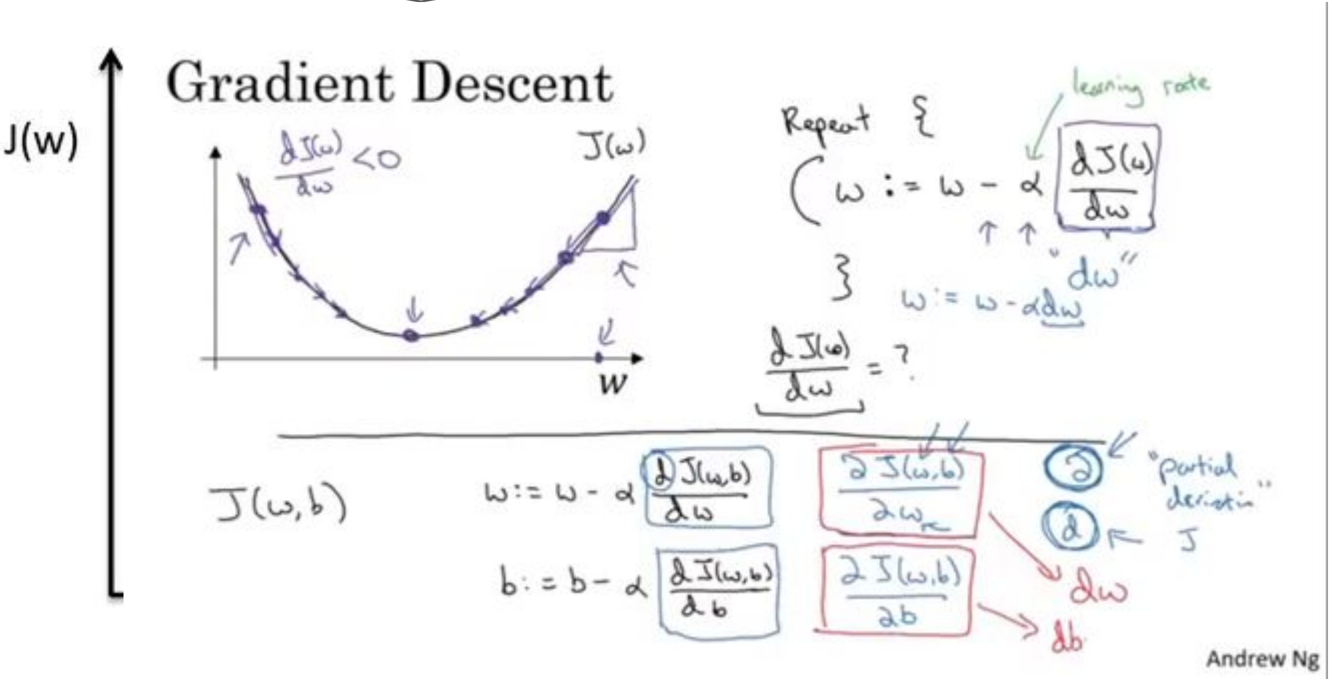
$$\mathit{Least\ Squared\ Error} = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$\mathit{Cost\ Function} = J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2, \quad m = \mathit{number\ of\ sample\ data}$$

backpropagation



backpropagation



Gradient Descent

learning rate

Repeat {

$$w := w - \alpha \boxed{\frac{dJ(w)}{dw}}$$

"dw"

}

$$w := w - \alpha dw$$

$$\underbrace{\frac{dJ(w)}{dw}} = ?$$

$J(w,b)$

$$w := w - \alpha \boxed{\frac{dJ(w,b)}{dw}} \quad \boxed{\frac{\partial J(w,b)}{\partial w}} \rightarrow dw$$

$$b := b - \alpha \boxed{\frac{dJ(w,b)}{db}} \quad \boxed{\frac{\partial J(w,b)}{\partial b}} \rightarrow db$$

∂ "partial derivatin"

d J

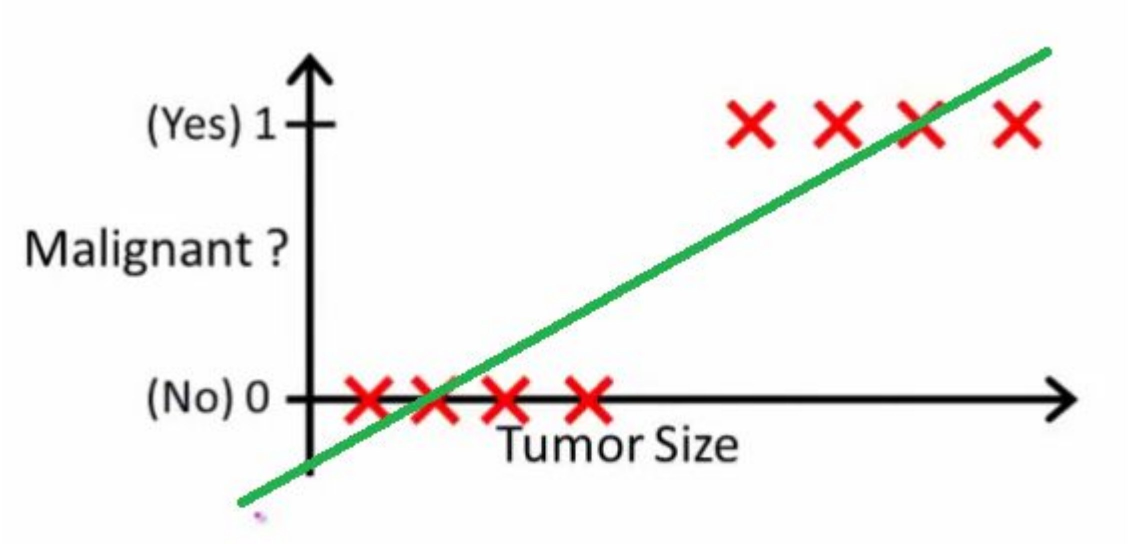
Andrew Ng

Linear Regression:

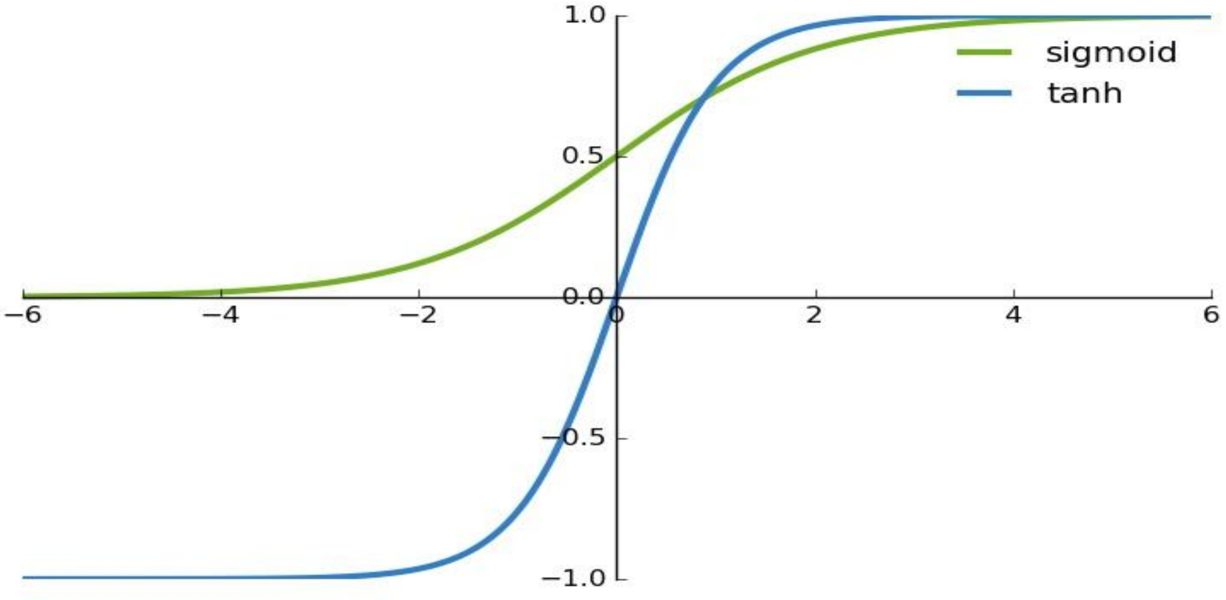
Notebook

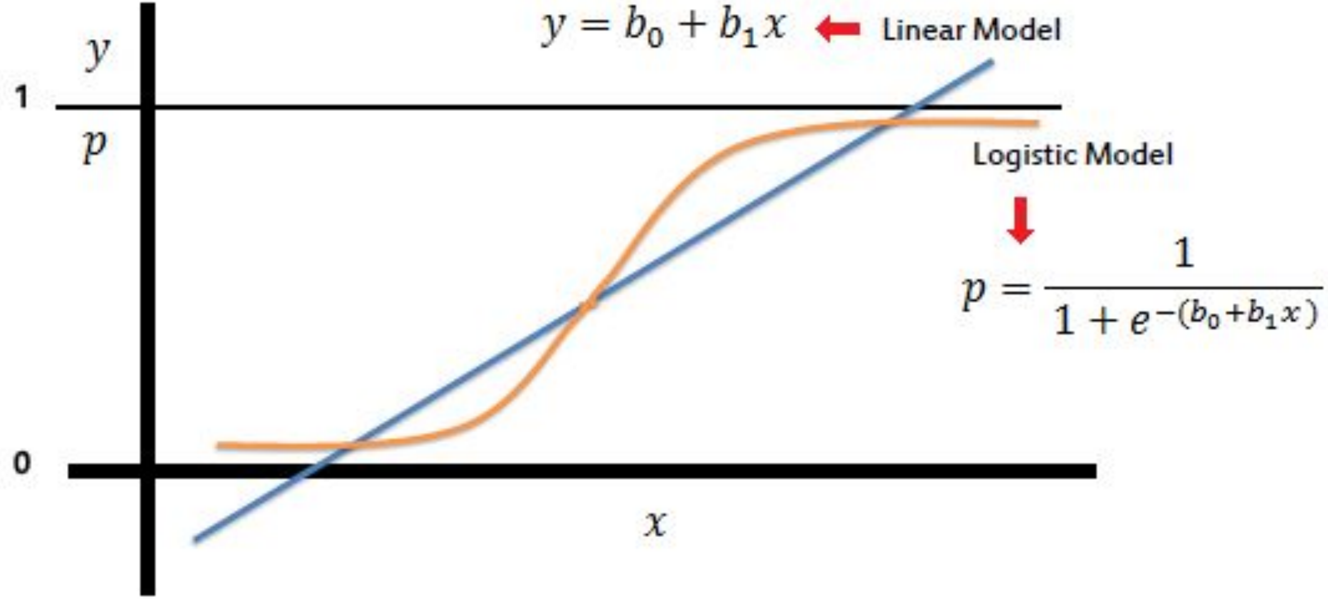
Logistic Regression

Why Logistic Regression?



Non-linear functions | Need of threshold



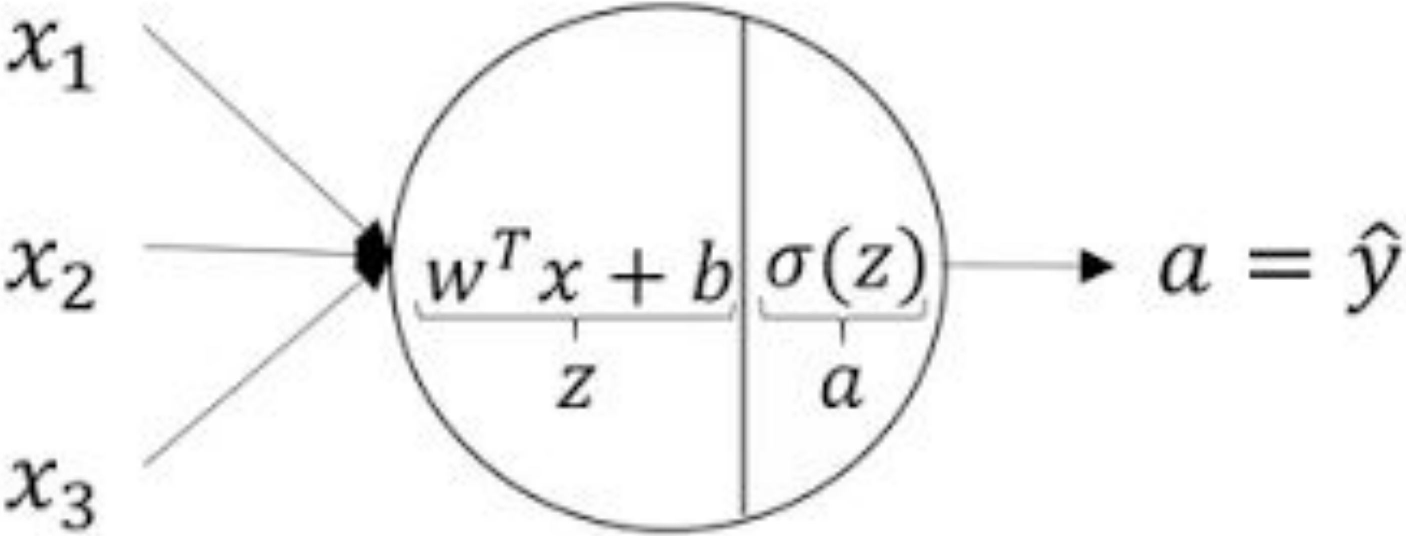


$y = b_0 + b_1 x$ ⬅ Linear Model

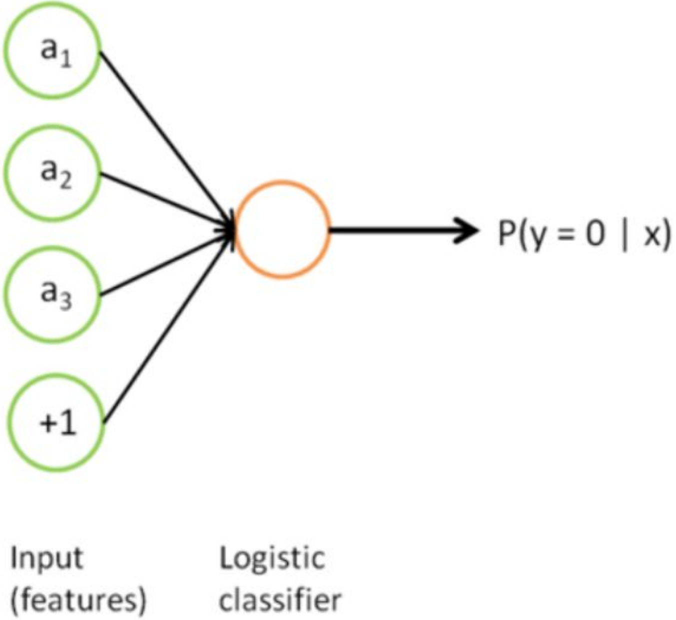
Logistic Model

⬇

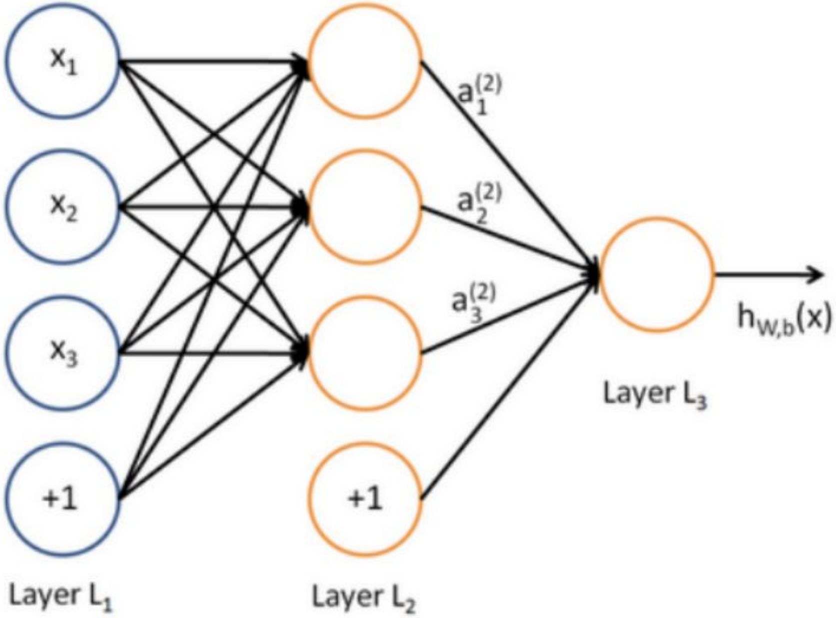
$$p = \frac{1}{1 + e^{-(b_0 + b_1 x)}}$$


x_1
x_2
x_3
$\underbrace{w^T x + b}_{z}$
$\underbrace{\sigma(z)}_{a}$
$a = \hat{y}$

Neural Network



Logistic Regression



Neural Network

Key difference b/w normal data and text

- **SEQUENTIAL ORDER**

House features: (#bedrooms, #windows, #kitchen, #washroom, ….)

(…., #windows, #bedrooms, #washroom, #kitchen,)

Text feature: (“And”, “I”, “want”, “a”, “million”, “dollar”, “!”)

(“want”, “And”, “million”, “I”, “dollar”, “a”, “!”)

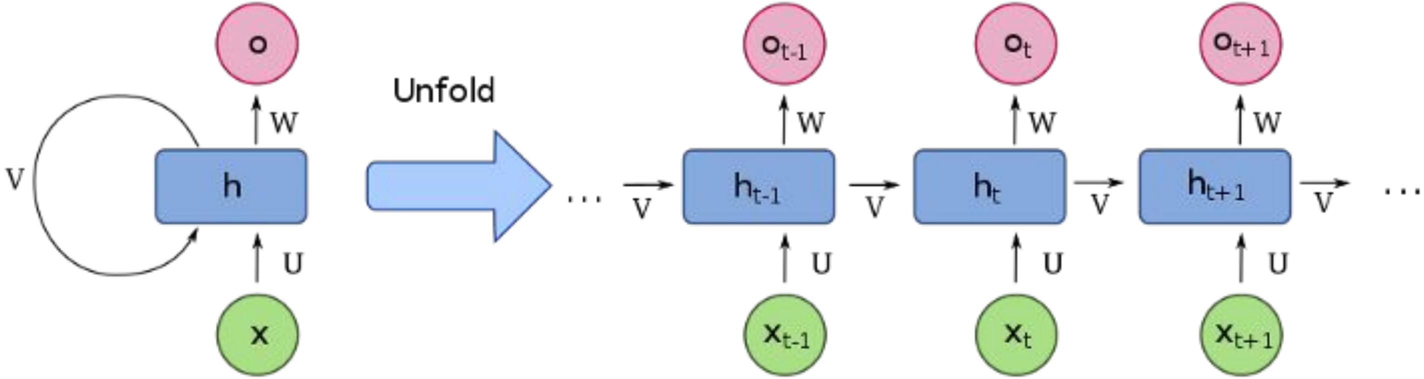
Few other differences

- **Long term dependencies**
- …
-

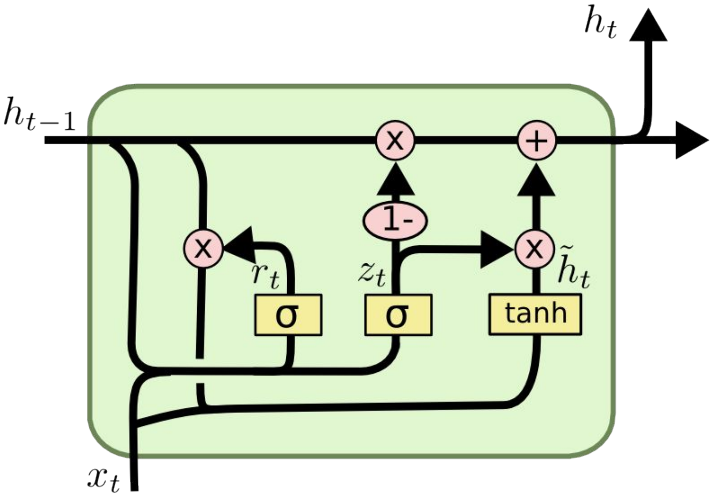
"*Well satisfied with his purchases and feeling very elegant indeed, Babar goes to the photographer to have* ***his picture*** *taken.*"

What can help?

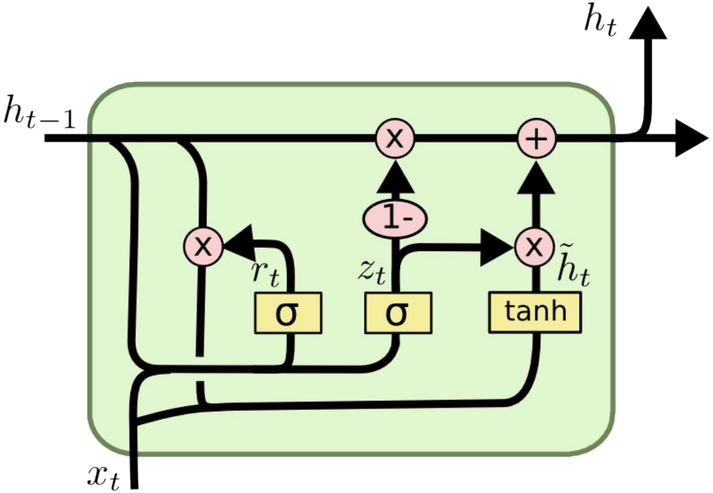
RNN (recurrent neural networks) - idea is to use previous input along with current input.



- LSTM (Long Short Term Memory): consists of components (gates) like remember, forget, update , input.



- LSTM (Long Short Term Memory): consists of components (gates) like remember, forget, update , input.



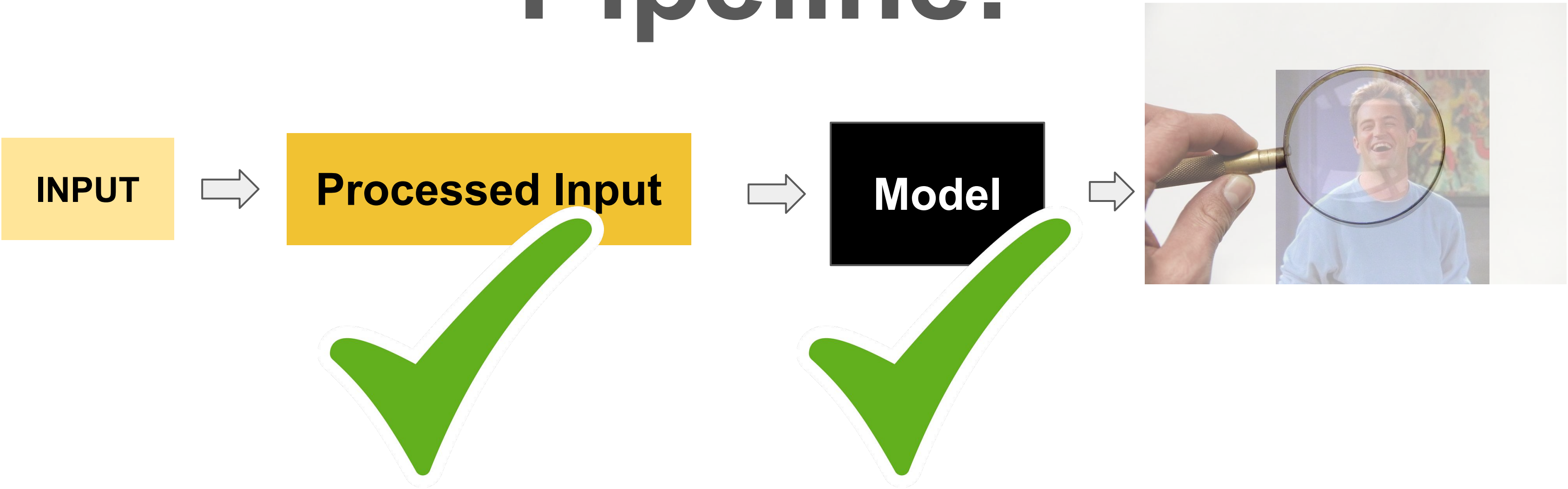
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Pipeline:



INPUT → Processed Input ✓ → Model ✓ →

Baseline Tutorial

References:

- https://distill.pub/2016/handwriting/
- https://colah.github.io/posts/2015-08-Understanding-LSTMs/
- http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/
- https://towardsdatascience.com/natural-language-processing-from-basics-to-using-rnn-and-lstm-ef6779e4ae66

Courses: Machine Learning and deeplearning.ai Specialization by Andrew NG.