Introduction to Deep Learning Part II

Jeff Liu and Ching Lee
2018/10/23

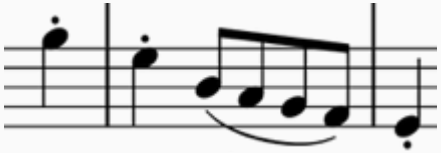
CloudMile

Variants of Neural Networks

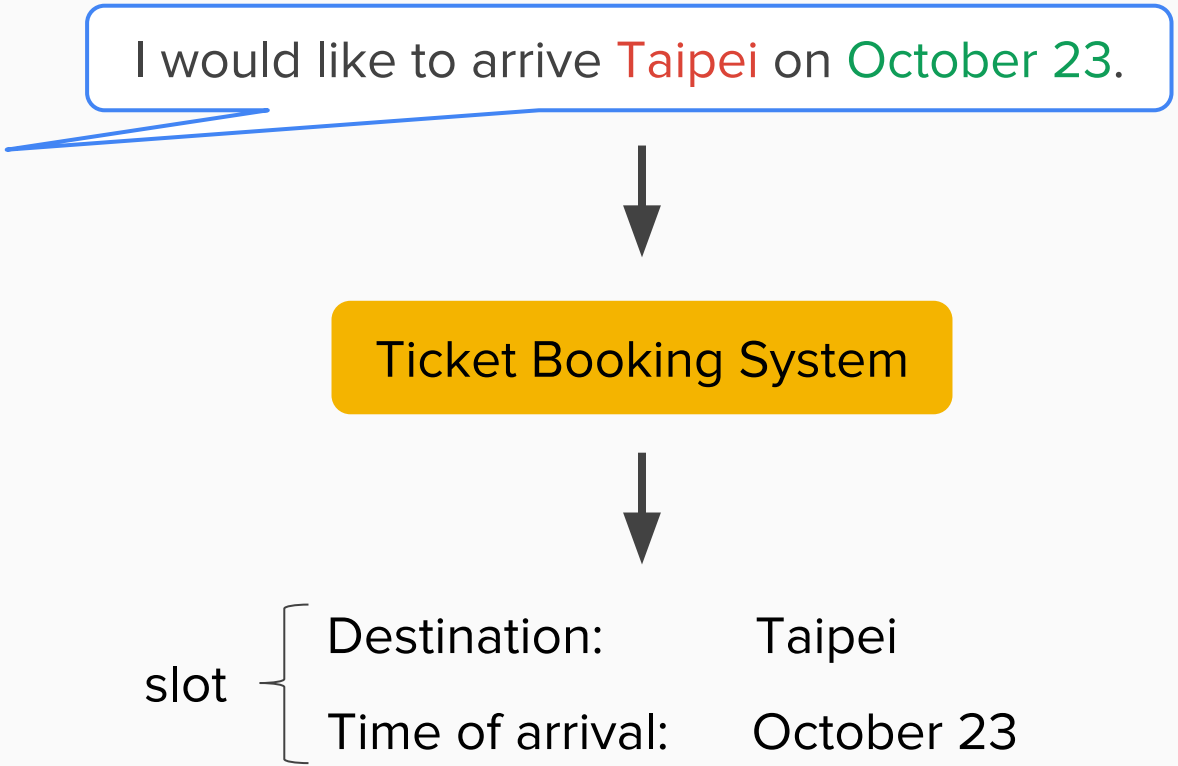
- Convolutional neural networks (CNN)
  → good for **images**

- Recurrent neural networks (RNN)
  → good for **sequential** data

Examples of sequence data

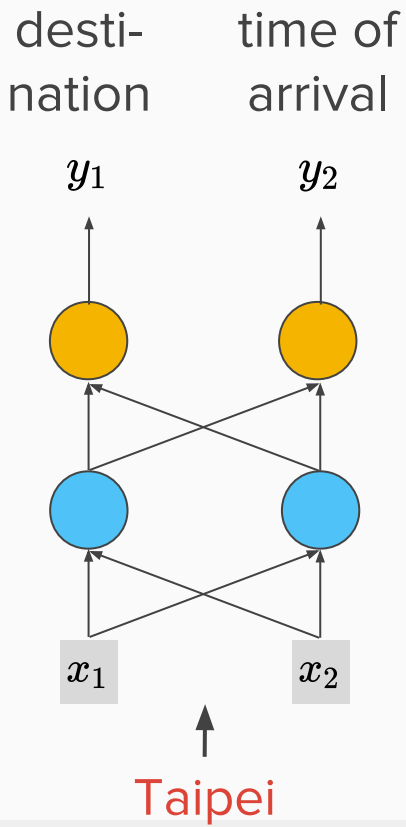
Speech recognition		→	你太skr了~
Music generation	ϕ	→	
Sentiment classification	看了這部電影覺得很高興.....	→	Positive (正雷)
DNA sequence analysis	AGCCCCTGTGAGGAACTAG	→	AGCCCCTGTGAGGAACTAG
Name entity recognition	柯文哲找北農總經理去市政府	→	柯文哲找北農總經理去市政府
Video activity recognition		→	running
Machine translation	安內母湯喔	→	這樣子不行喔

Why sequence models? Let's see a slot filling problem ...



Why sequence models?

arrive Taipei on October 23



Why sequence models?

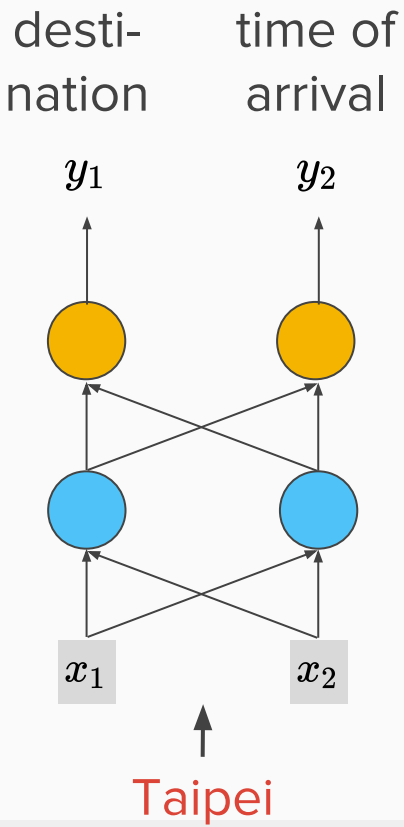
arrive Taipei on October 23

destination

leave Taipei on October 23

place of departure

Neural network needs memory!

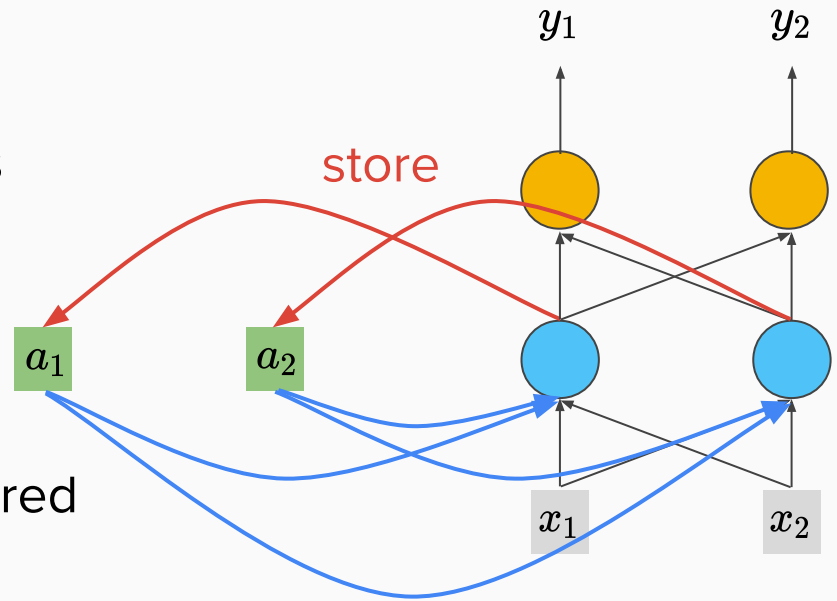


CloudMile

Recurrent Neural Network (RNN)

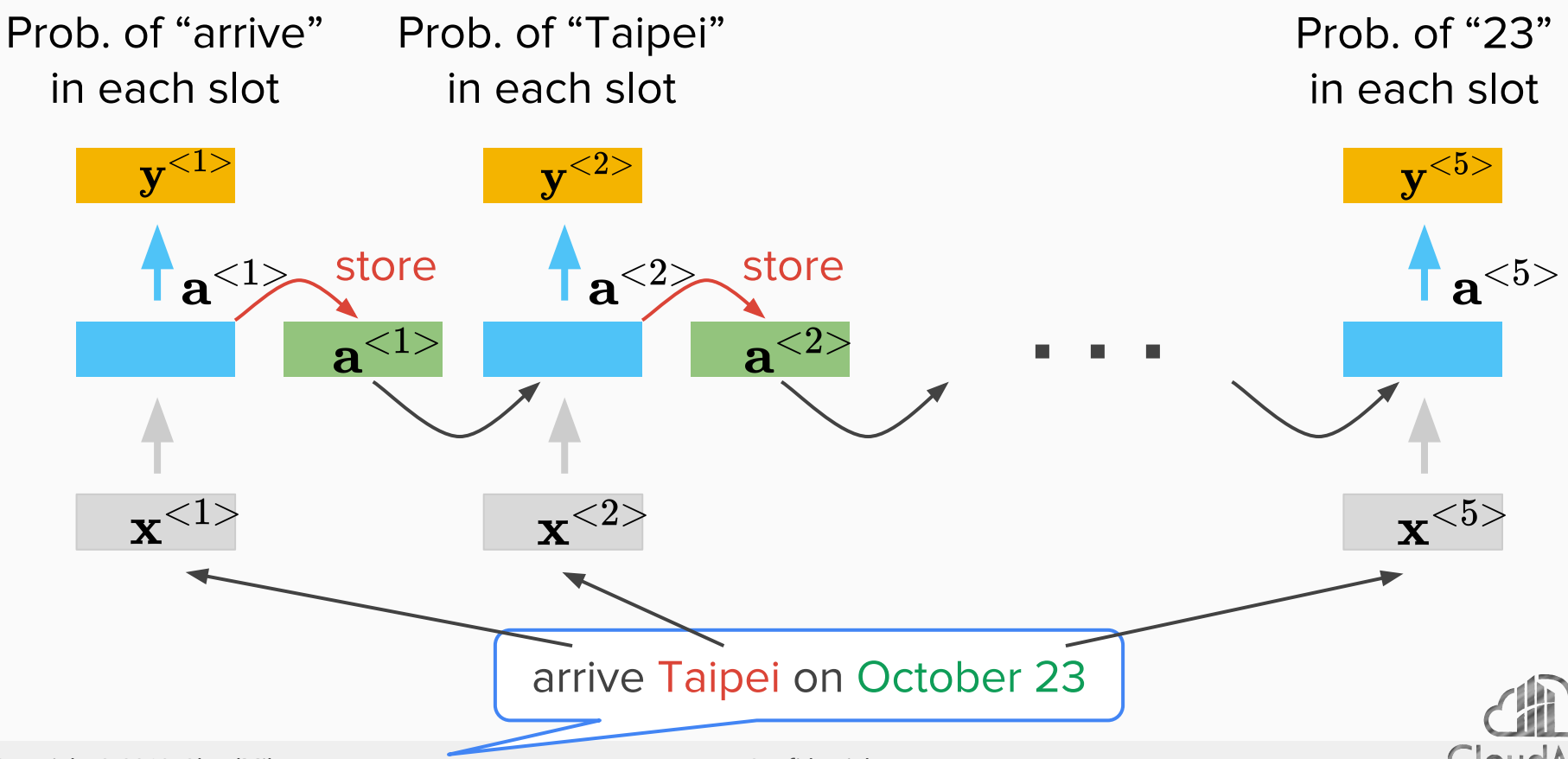
The outputs of hidden layers are stored in the memory.

The memory can be considered as another input.



RNN

The same network is used again and again.



RNN

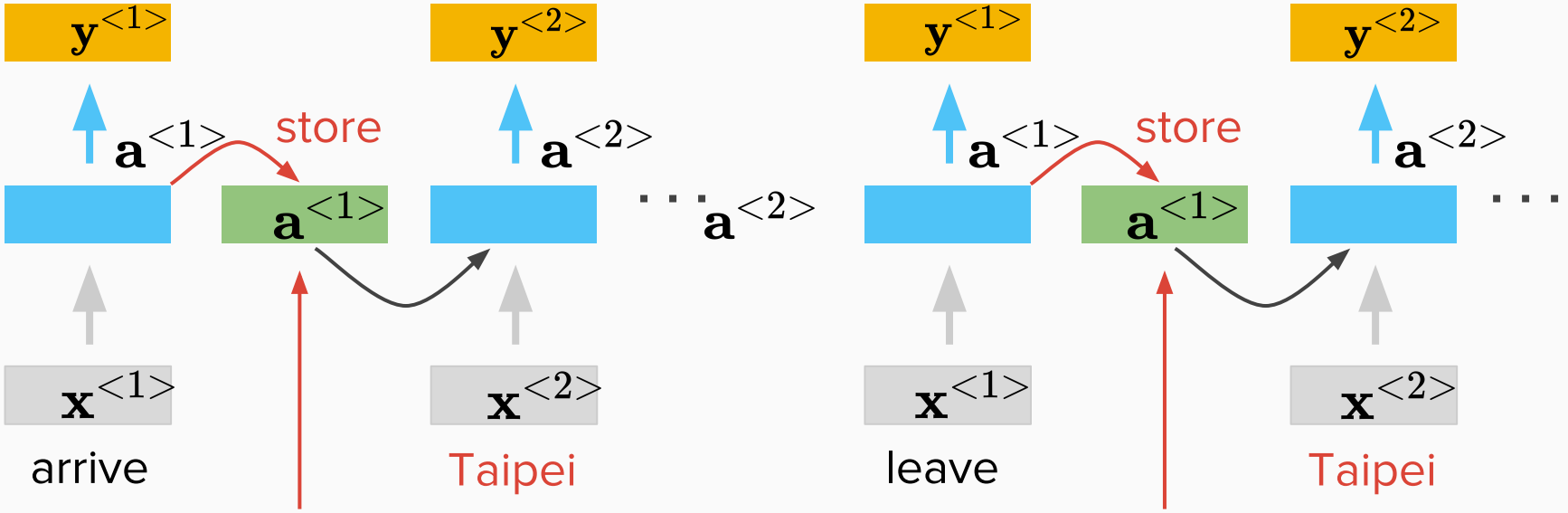
different!

Prob. of “arrive” in each slot

Prob. of “Taipei” in each slot

Prob. of “leave” in each slot

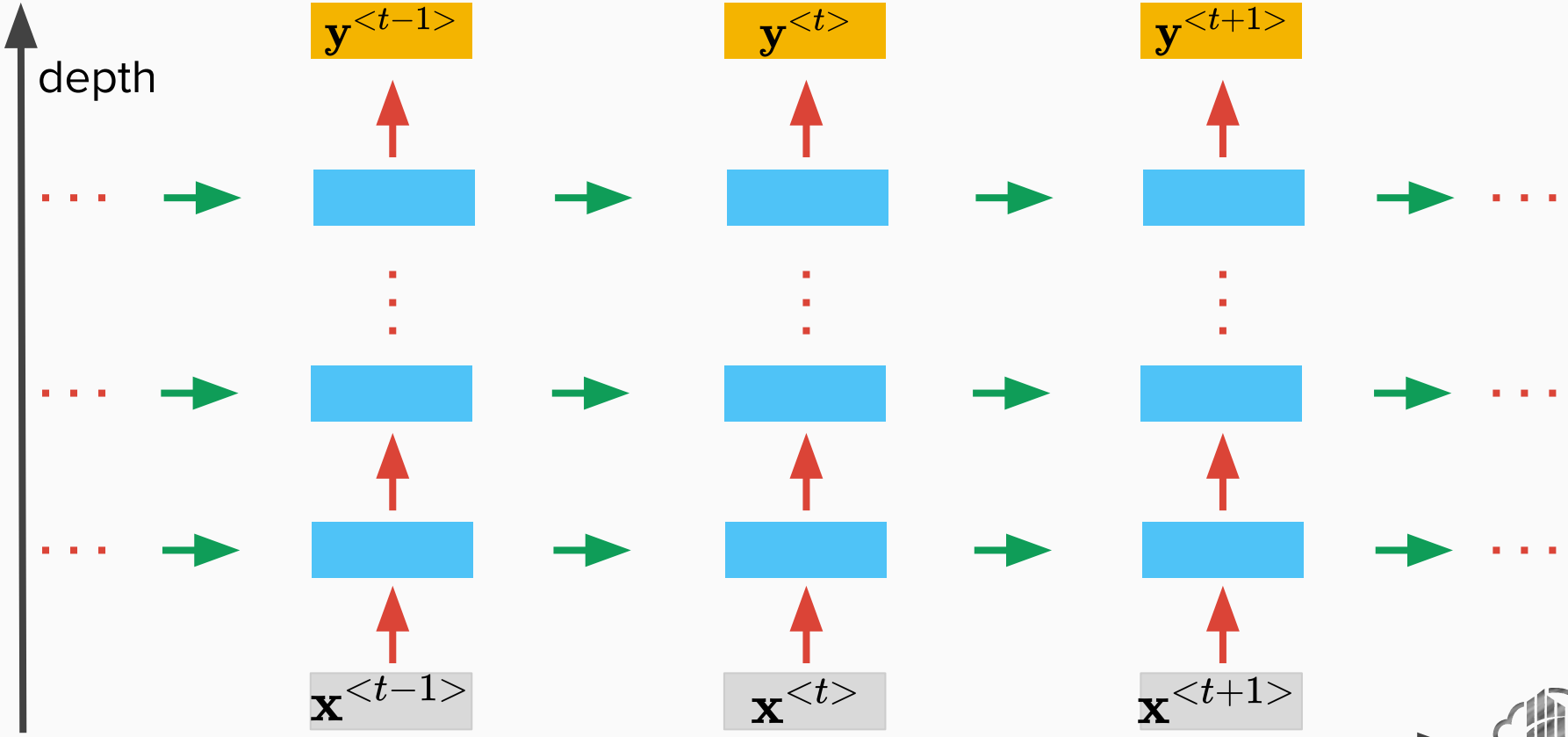
Prob. of “Taipei” in each slot



The values stored in the memory is different.

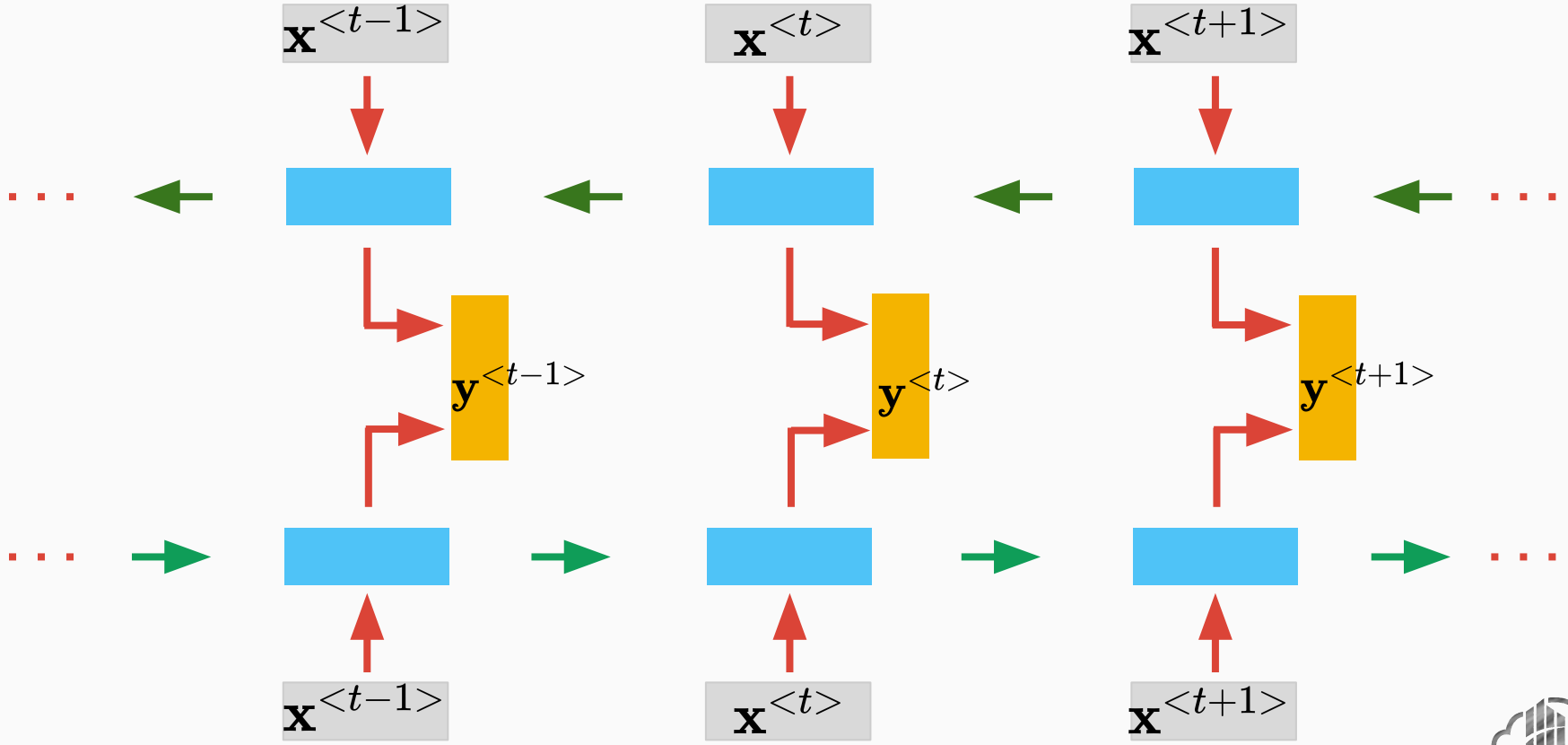
CloudMile

Deep RNN

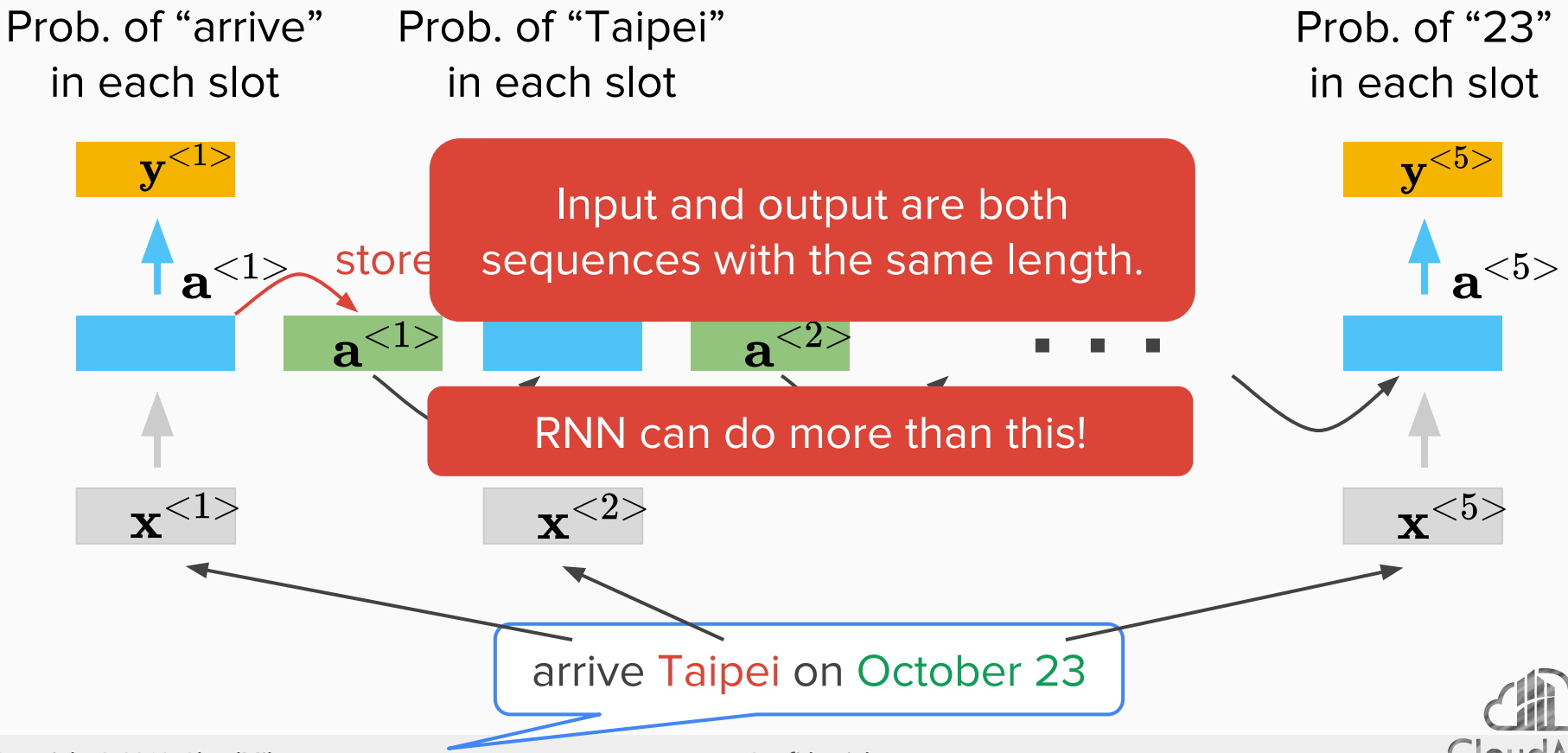


Bidirectional RNN

He said, “Teddy Roosevelt was a great President.”
He said, “Teddy bears are on sales!”

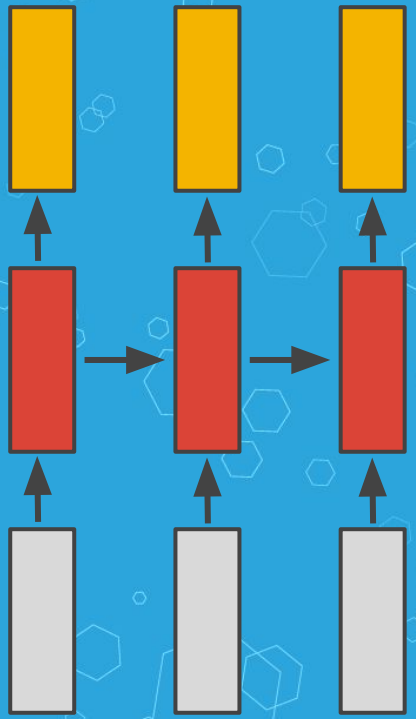


More Applications



CloudMile

Many-to-many

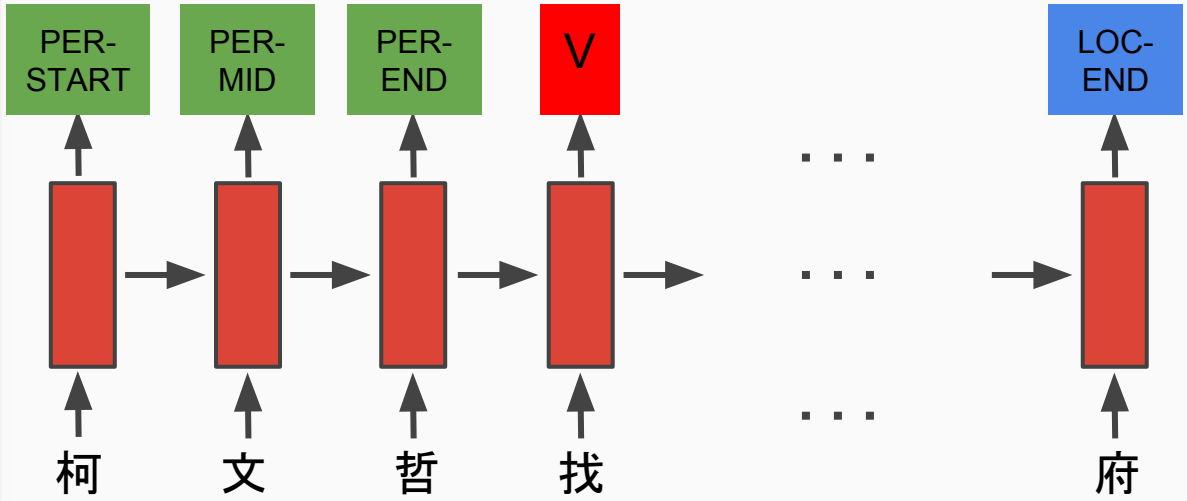


Name entity recognition

柯文哲找北農總經
理去市政府

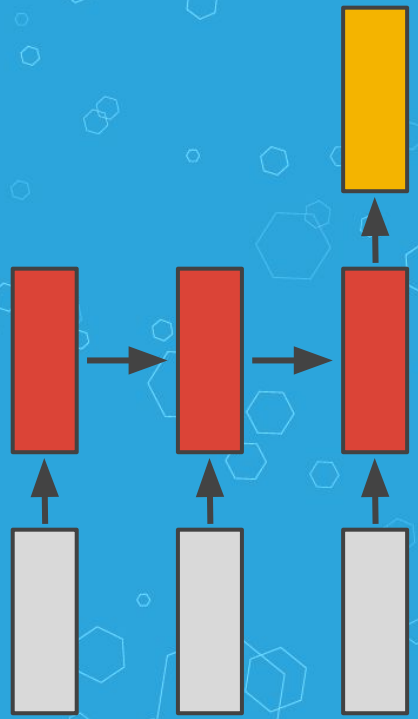
→

柯文哲找北農總經
理去市政府



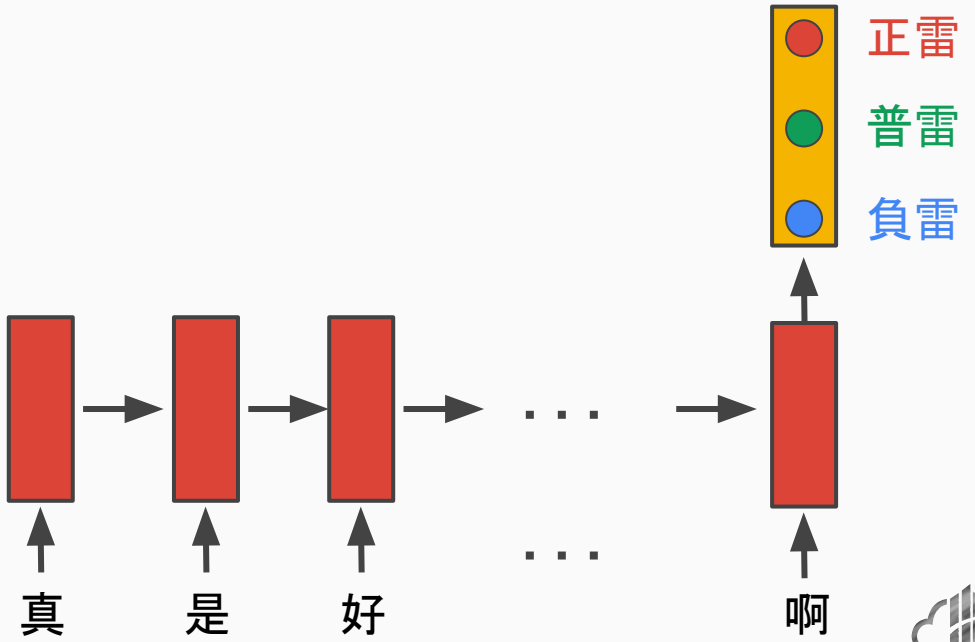
CloudMile

Many-to-one



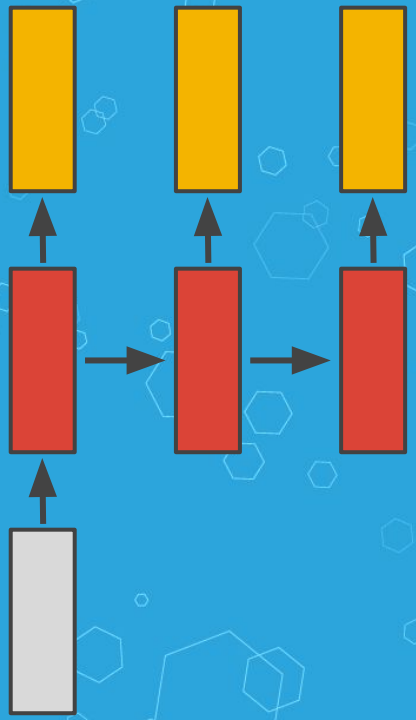
Sentiment Analysis

看了這部電影覺得很高興	這部電影太糟了	這部電影很棒 .....
Positive (正雷)	Negative (負雷)	Positive (正雷)

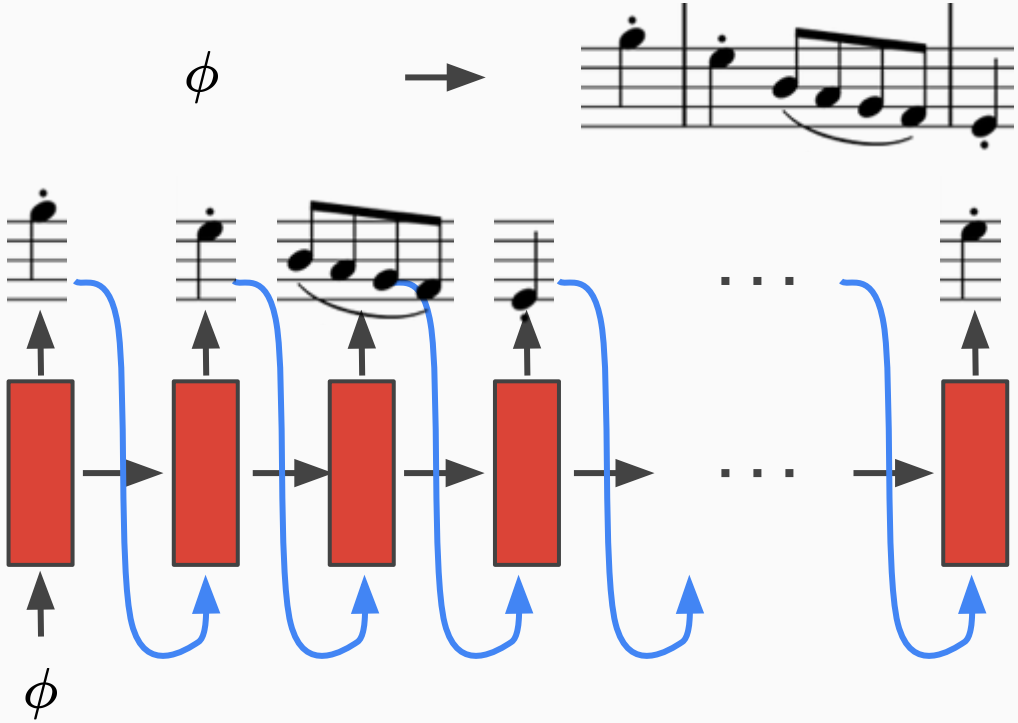


CloudMile

One-to-many

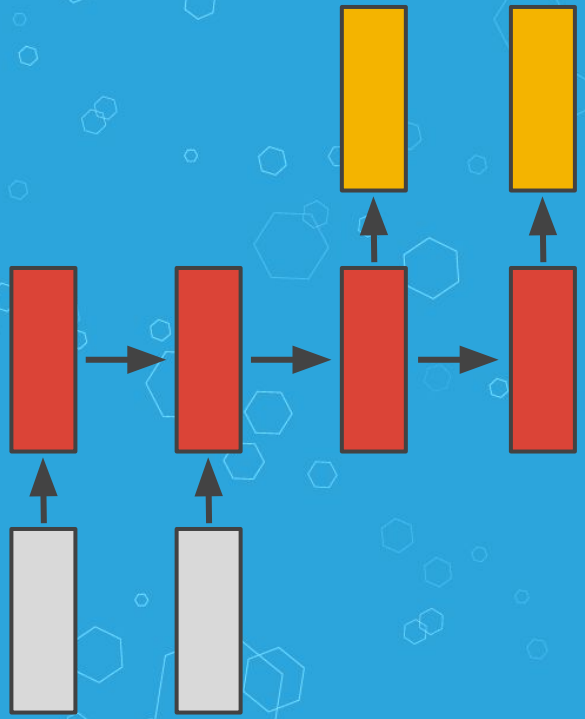


Music generation



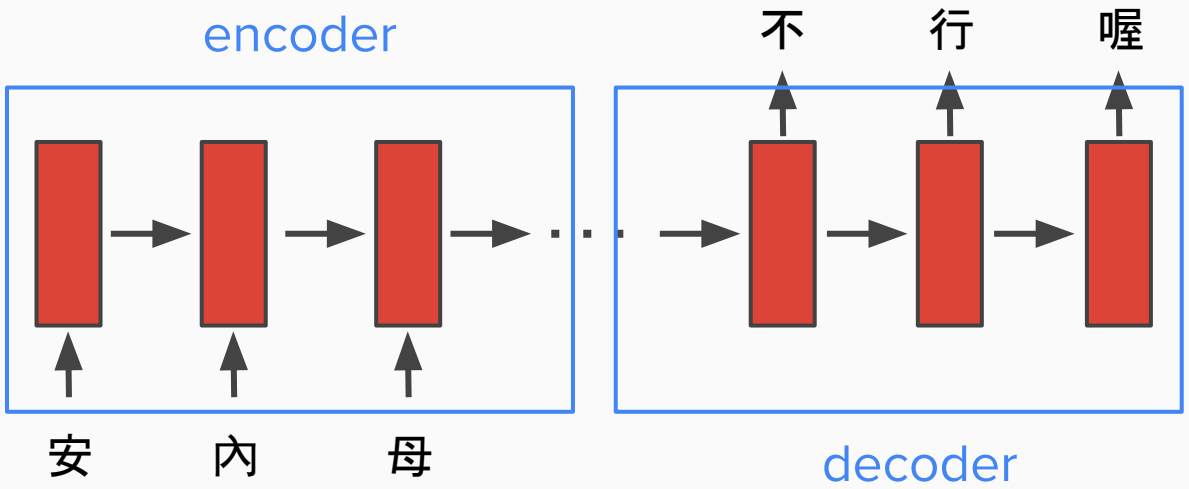
CloudMile

Many-to-many



Machine translation

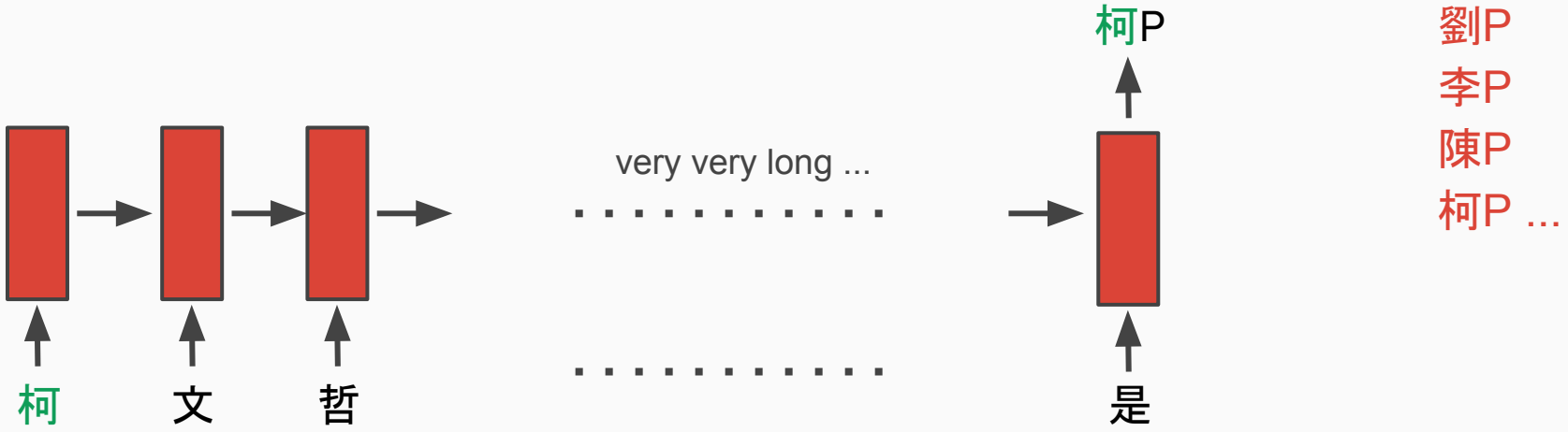
安內母湯喔 → 這樣子不行喔




IT'S NOT
THAT
SIMPLE

Long term dependencies

柯文哲(1959年8月6日－)，臺灣新竹市人，中華民國無黨籍政治人物、醫學家，現任臺北市市長。國立臺灣大學醫學院臨床醫學研究所博士畢業，曾任臺大醫院急診部醫師、臺大醫院創傷醫學部主任、臺大醫學院教授，專長為外科重症醫學、器官移植　、人工器官等，是臺灣第一個急診與重症加護專職醫生，為臺灣器官標準移植程序的建立者，也是首位將葉克膜(ECMO)技術引進臺灣的醫師。**他的綽號是？？？**



RNN is not always easy to learn ...

$w = 1 \rightarrow y_{1000} = 1$

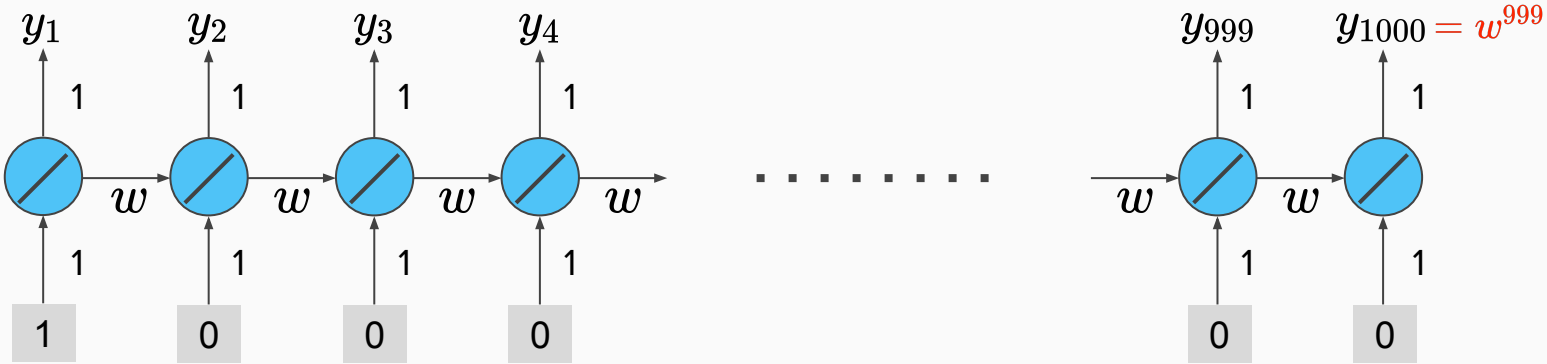
$w = 1.01 \rightarrow y_{1000} \approx 20000$

$w = 0.99 \rightarrow y_{1000} \approx 0$

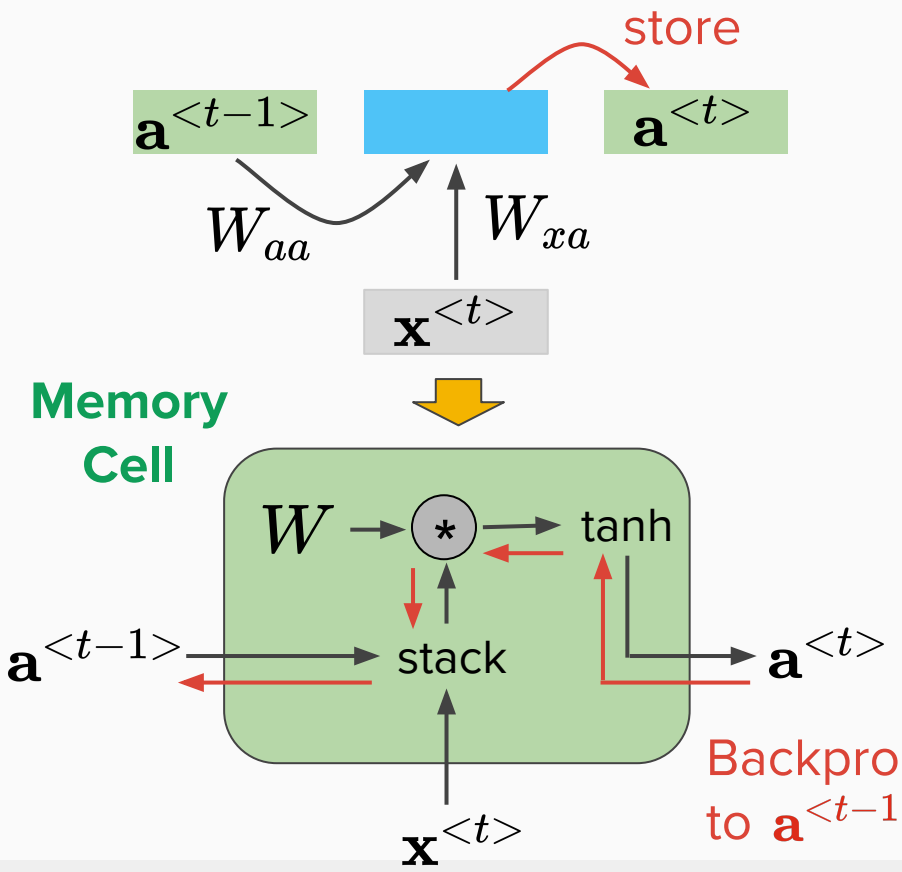
$w = 0.01 \rightarrow y_{1000} \approx 0$

large gradient → small learning rate?

small gradient → large learning rate?



Vanilla RNN Gradient Flow



$$\mathbf{a}^{<t>} = \tanh(W_{aa}\mathbf{a}^{<t-1>} + W_{xa}\mathbf{x}^{<t>})$$

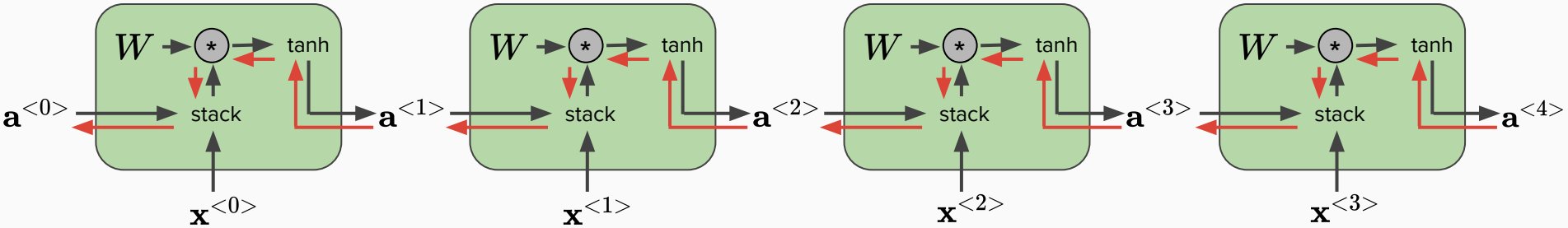
$$= \tanh\left((W_{aa} \;\; W_{ax})\begin{pmatrix}\mathbf{a}^{<t-1>}\\ \mathbf{x}^{<t>}\end{pmatrix}\right)$$

$$= \tanh\left(W\begin{pmatrix}\mathbf{a}^{<t-1>}\\ \mathbf{x}^{<t>}\end{pmatrix}\right)$$

activation — weighted sum

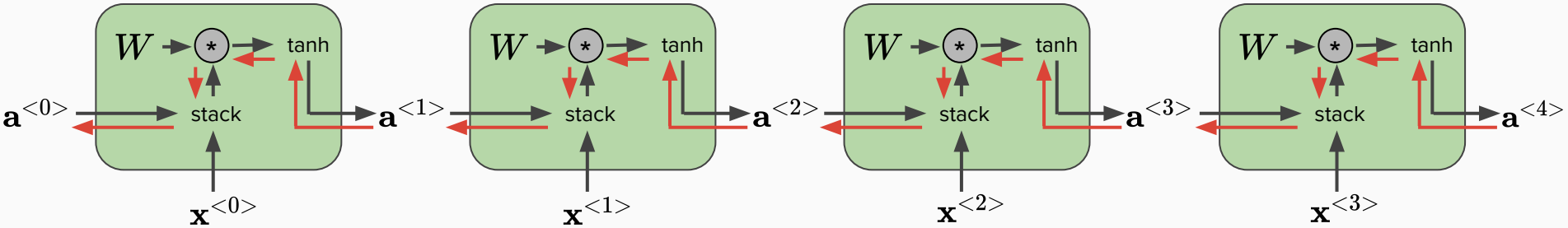
Backpropagation from $\mathbf{a}^{<t>}$ to $\mathbf{a}^{<t-1>}$ multiplies by W

Vanilla RNN Gradient Flow



Computing gradient of $\mathbf{a}^{<0>}$ involves many factors of W (and repeated tanh)

Vanilla RNN Gradient Flow



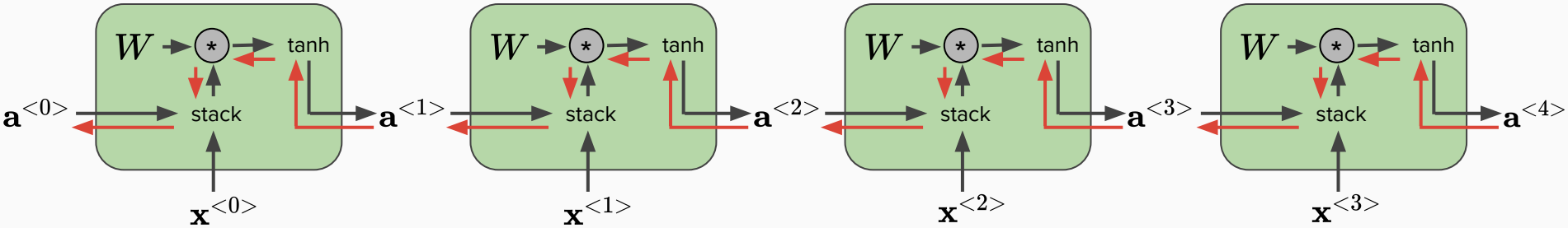
Computing gradient of $\mathbf{a}^{<0>}$ involves many factors of W (and repeated tanh)

$W > 1$:
Exploding gradients

$W < 1$:
Vanishing gradients

(Here, we assume W is a scalar for your better understanding)

Vanilla RNN Gradient Flow



Computing gradient of $\mathbf{a}^{<0>}$ involves many factors of W (and repeated tanh)

$W > 1$:
Exploding gradients

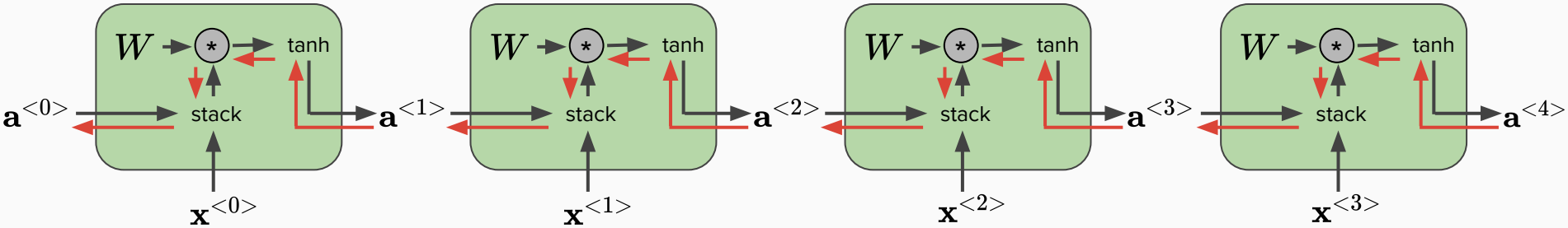
→ **Gradient clipping**: Scale gradient if its norm is too big

$W < 1$:
Vanishing gradients

(Here, we assume W is a scalar for your better understanding)

```
grad_norm = np.sum(grad * grad)
if grad_norm > threshold:
  grad *= (threshold / grad_norm)
```

Vanilla RNN Gradient Flow



Computing gradient of $\mathbf{a}^{<0>}$ involves many factors of W (and repeated tanh)

$W > 1$:
Exploding gradients

$W < 1$:
Vanishing gradients → Change RNN architecture

(Here, we assume W is a scalar for your better understanding)

Vanilla RNN v.s. Long Short-term Memory (LSTM)

Vanilla RNN

$$\mathbf{a}^{<t>} = \tanh\left(W\begin{pmatrix}\mathbf{a}^{<t-1>}\\ \mathbf{x}^{<t>}\end{pmatrix}\right)$$
$$\mathbf{a}^{<t>} = \mathbf{c}^{<t>}$$

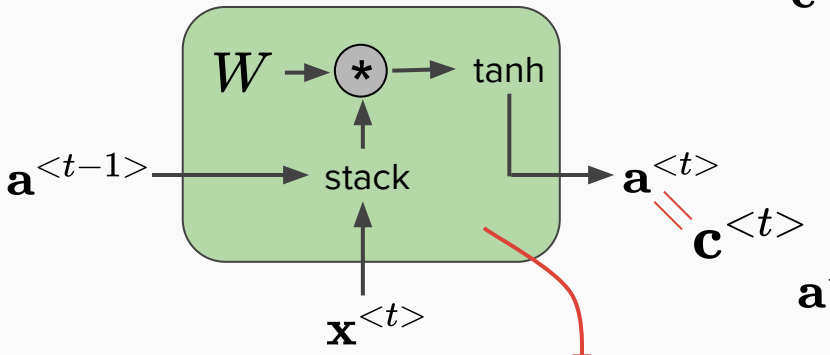
LSTM

$$\begin{pmatrix}\mathbf{i}\\ \mathbf{f}\\ \mathbf{o}\\ \mathbf{g}\end{pmatrix} = \begin{pmatrix}\sigma\\ \sigma\\ \sigma\\ \tanh\end{pmatrix} W\begin{pmatrix}\mathbf{a}^{<t-1>}\\ \mathbf{x}^{<t>}\end{pmatrix}$$
$$\mathbf{c}^{<t>} = \mathbf{f} \odot \mathbf{c}^{<t-1>} + \mathbf{i} \odot \mathbf{g}$$
$$\mathbf{a}^{<t>} = \mathbf{o} \odot \tanh(\mathbf{c}^{<t>})$$

Values stored in the memory cell

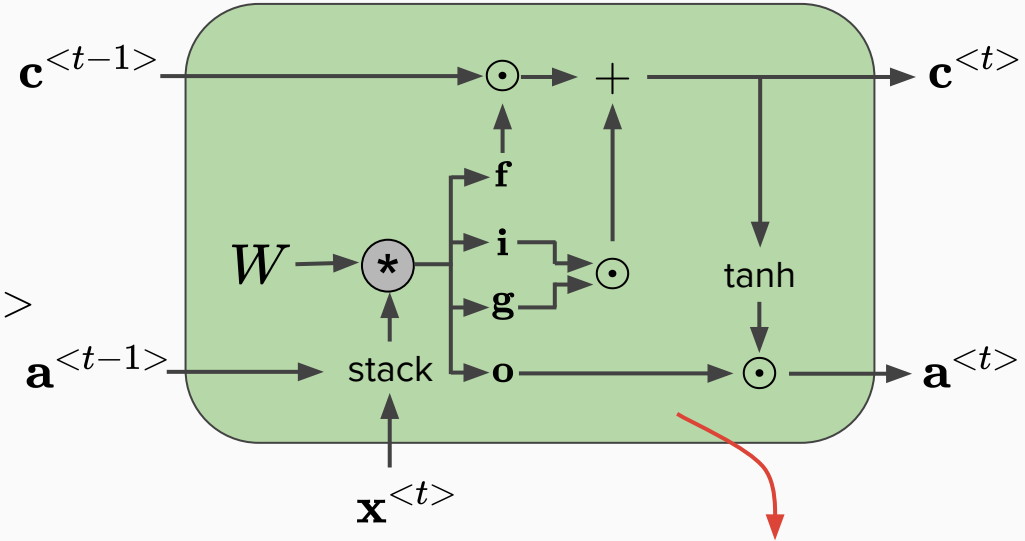
Vanilla RNN v.s. LSTM in Pictures

Vanilla RNN



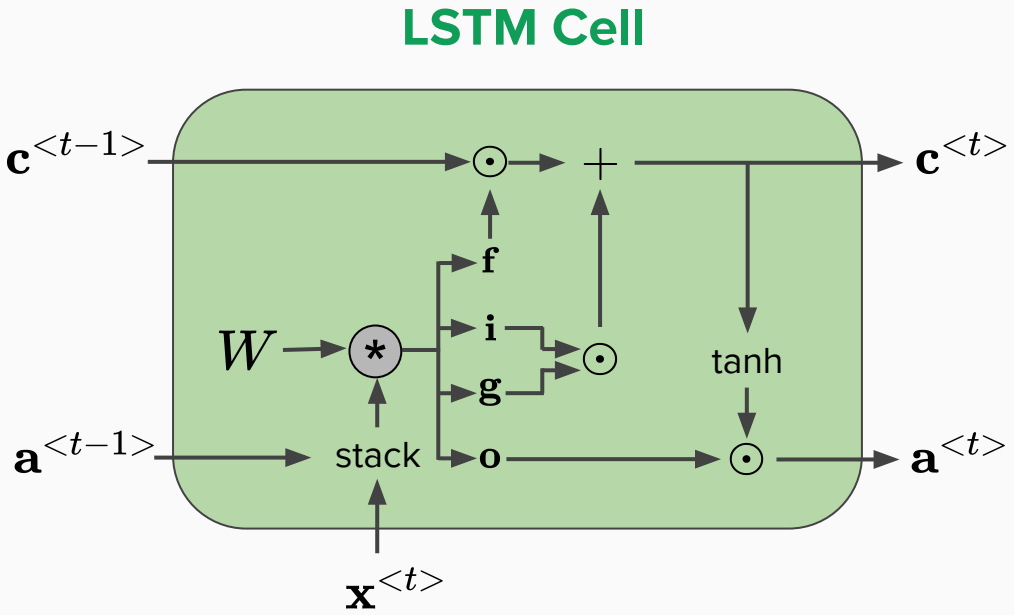
The memory cell will be updated every time step and only stores the information of previous time step.

LSTM



Can control the input and output of the memory cell, or erase the information stored inside.

Long Short-term Memory (LSTM)



$\mathbf{f}$: Forget gate, whether to erase cell
$\mathbf{i}$: Input gate, whether to write to cell
$\mathbf{g}$: Gate gate (?), how much to write to cell
$\mathbf{o}$: Output gate, how much to reveal cell

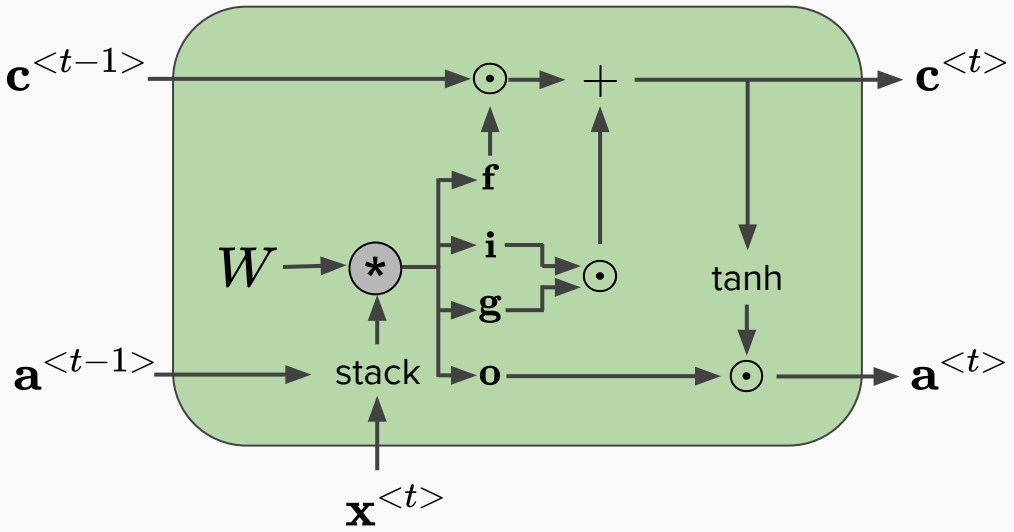
$$\begin{pmatrix} \mathbf{i} \\ \mathbf{f} \\ \mathbf{o} \\ \mathbf{g} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} \mathbf{a}^{<t-1>} \\ \mathbf{x}^{<t>} \end{pmatrix}$$

$$\mathbf{c}^{<t>} = \mathbf{f} \odot \mathbf{c}^{<t-1>} + \mathbf{i} \odot \mathbf{g}$$

$$\mathbf{a}^{<t>} = \mathbf{o} \odot \tanh(\mathbf{c}^{<t>})$$

Long Short-term Memory (LSTM)

LSTM Cell



Amazing! This box of cereal gave me a perfectly balanced breakfast, as all things should be. I only ate half of it but will definitely be buying again!

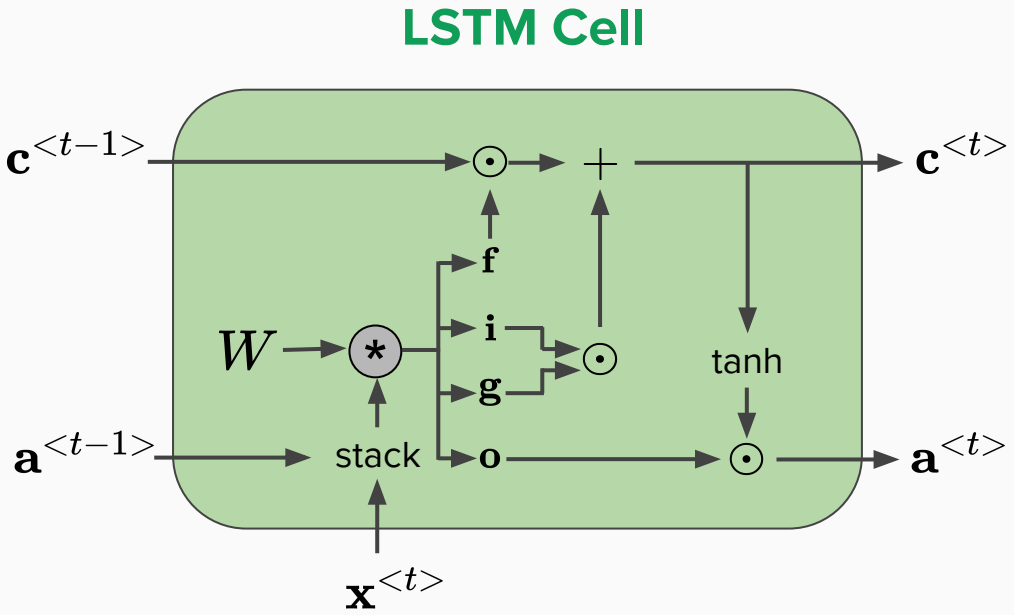
$\mathbf{f}$: Forget gate, whether to erase cell
$\mathbf{i}$: Input gate, whether to write to cell
$\mathbf{g}$: Gate gate (?), how much to write to cell
$\mathbf{o}$: Output gate, how much to reveal cell

$$\begin{pmatrix} \mathbf{i} \\ \mathbf{f} \\ \mathbf{o} \\ \mathbf{g} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} \mathbf{a}^{<t-1>} \\ \mathbf{x}^{<t>} \end{pmatrix}$$

$$\mathbf{c}^{<t>} = \mathbf{f} \odot \mathbf{c}^{<t-1>} + \mathbf{i} \odot \mathbf{g}$$

$$\mathbf{a}^{<t>} = \mathbf{o} \odot \tanh(\mathbf{c}^{<t>})$$

Long Short-term Memory (LSTM)



Amazing! This box of cereal gave me a **perfectly balanced breakfast,** as all things should be. I only ate half of it but **will definitely be buying again!**

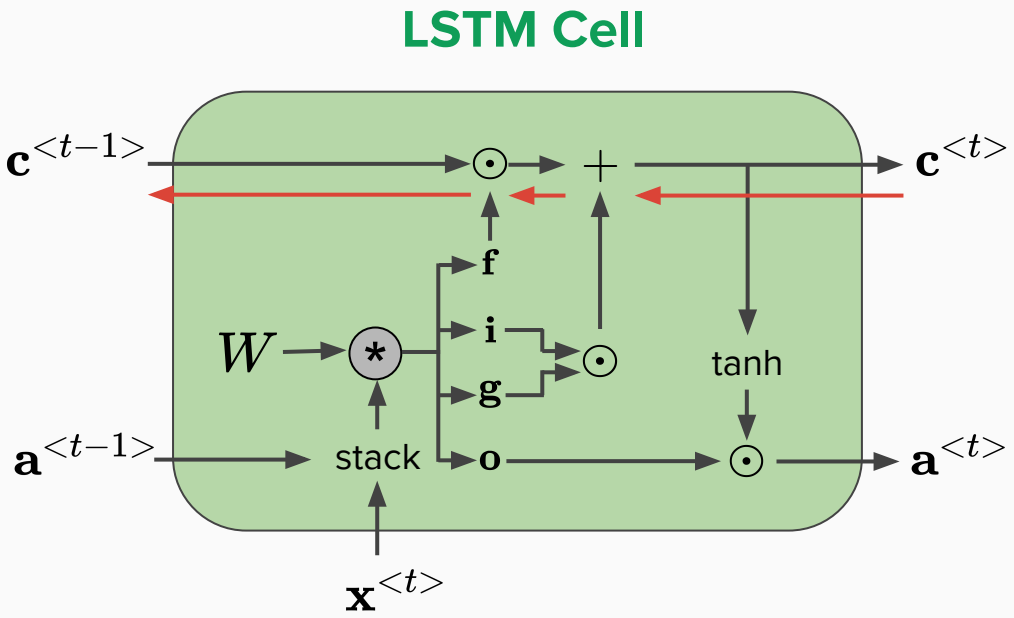
$\mathbf{f}$: Forget gate, whether to erase cell
$\mathbf{i}$: Input gate, whether to write to cell
$\mathbf{g}$: Gate gate (?), how much to write to cell
$\mathbf{o}$: Output gate, how much to reveal cell

$$\begin{pmatrix} \mathbf{i} \\ \mathbf{f} \\ \mathbf{o} \\ \mathbf{g} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} \mathbf{a}^{<t-1>} \\ \mathbf{x}^{<t>} \end{pmatrix}$$

$$\mathbf{c}^{<t>} = \mathbf{f} \odot \mathbf{c}^{<t-1>} + \mathbf{i} \odot \mathbf{g}$$

$$\mathbf{a}^{<t>} = \mathbf{o} \odot \tanh(\mathbf{c}^{<t>})$$

Long Short-term Memory (LSTM)



Backpropagation from $\mathbf{c}^{<t>}$ to $\mathbf{c}^{<t-1>}$ only element-wise multiplication by $\mathbf{f}$, not multiplied by W

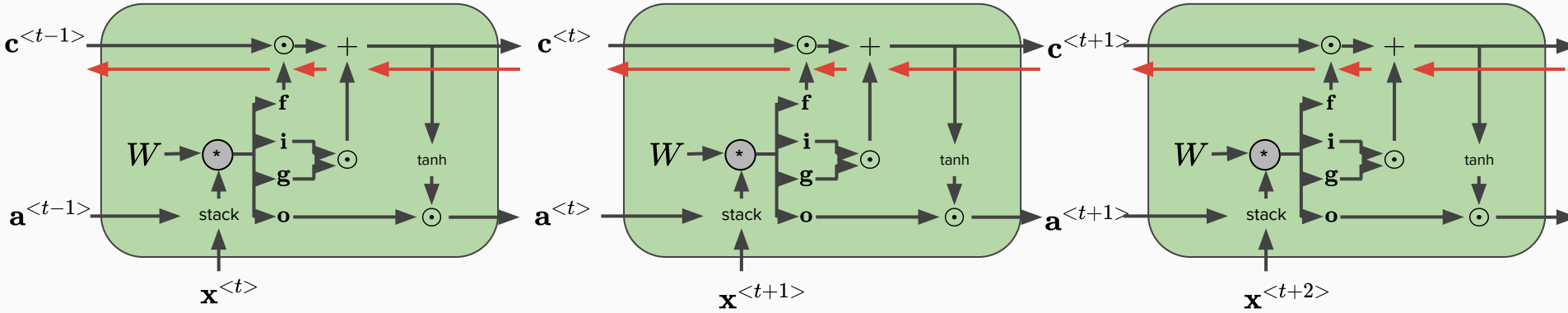
$$\begin{pmatrix} \mathbf{i} \\ \mathbf{f} \\ \mathbf{o} \\ \mathbf{g} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} \mathbf{a}^{<t-1>} \\ \mathbf{x}^{<t>} \end{pmatrix}$$

$$\mathbf{c}^{<t>} = \mathbf{f} \odot \mathbf{c}^{<t-1>} + \mathbf{i} \odot \mathbf{g}$$

$$\mathbf{a}^{<t>} = \mathbf{o} \odot \tanh(\mathbf{c}^{<t>})$$

Long Short-Term Memory (LSTM): Gradient Flow

Uninterrupted gradient flow!



How to represent words in computers?

Word representation by one-hot vectors

V = [a, aaron, ..., zulu, <UNK>], |V| = 10,000

one-hot representation

Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)

I want a glass of **orange juice**.

I want a glass of **apple ___?___**.

There is no natural notion of **similarity** for one-hot vectors!

CloudMile

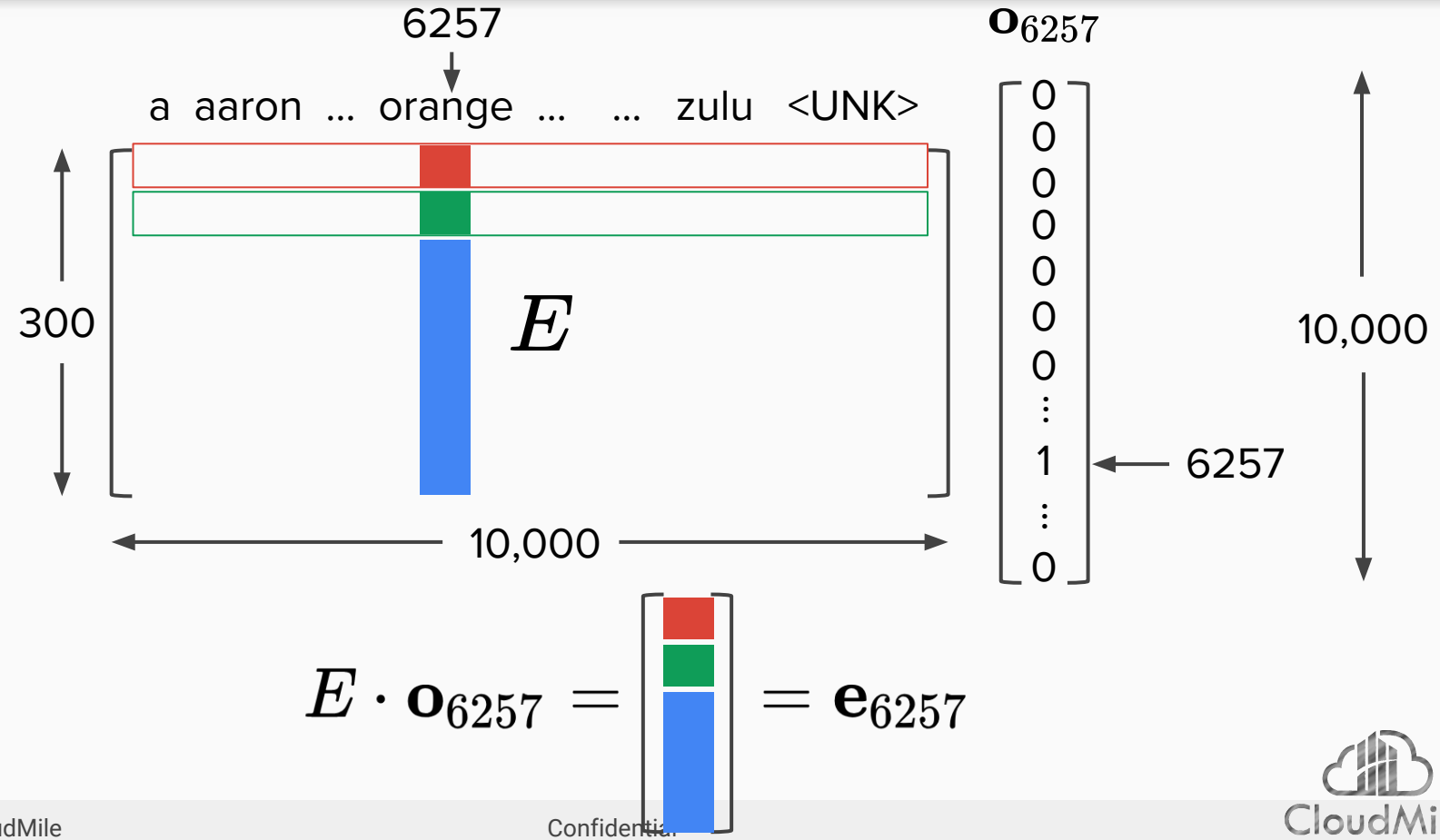
Featurized representation: word embedding

	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	-1	1	-0.95	0.97	0.00	0.01
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	0.09	0.01	0.02	0.01	0.95	0.97
⋮	⋮	⋮	⋮			
	e_{5391}	e_{9853}				

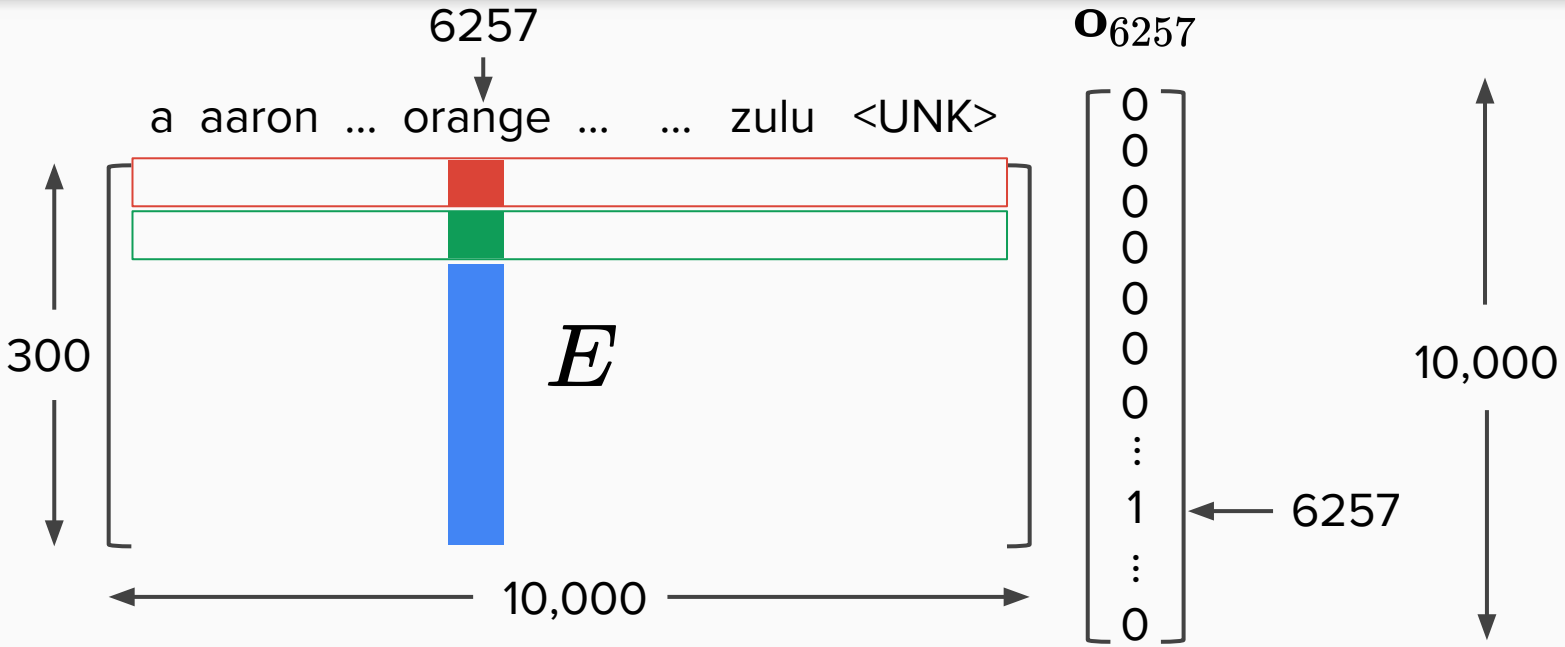
less than 10,000 e.g. 300

I want a glass of **orange juice**.
I want a glass of **apple juice**.

Embedding Matrix



Embedding Matrix

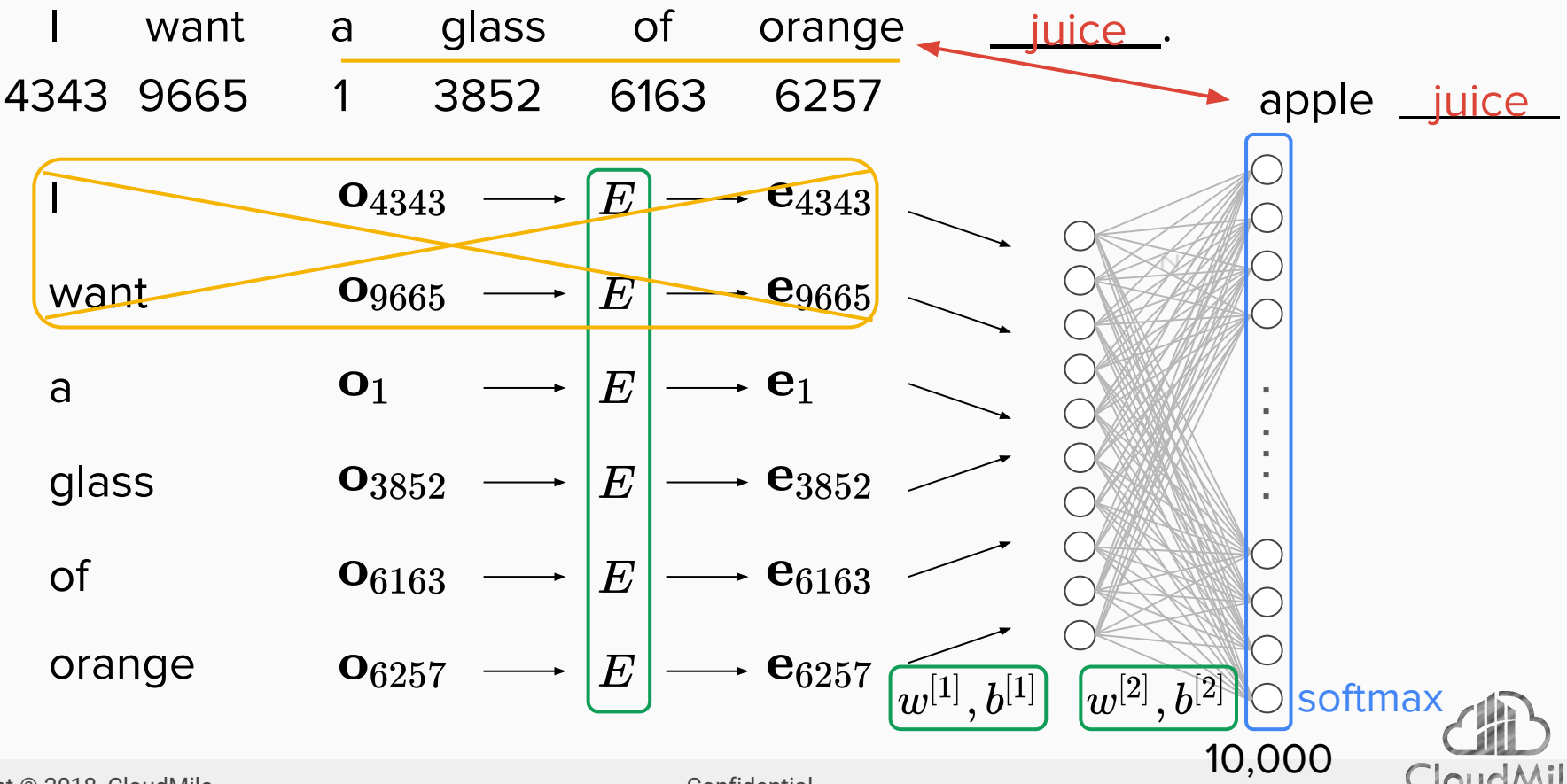


In practice, use specialized function to **look up** an embedding.

keras.layers.Embedding

CloudMile

Learning Word Embeddings



Other context/target pairs

I want a glass of orange juice to go along with my cereal.

target

Context: Last 4 words

a glass of orange ___?___ to go along with

Other context/target pairs

I want a glass of orange juice to go along with my cereal.

target

Context: Last 4 words

4 words on left & right a glass of orange ___?___ to go along with

Other context/target pairs

I want a glass of orange juice to go along with my cereal.

target

Context: Last 4 words

4 words on left & right

Last 1 word a glass of orange ___?___

Other context/target pairs

I want a glass of orange juice to go along with my cereal.

target

Context: Last 4 words

4 words on left & right

Last 1 word

Nearby 1 word

a glass of ?

skip-gram

Word2Vec Skip-gram Model

I want a glass of orange juice to go along with my cereal.

randomly pick a word as context word

Context	Target	
orange	juice	randomly pick
orange	glass	
orange	my	
⋮	⋮	

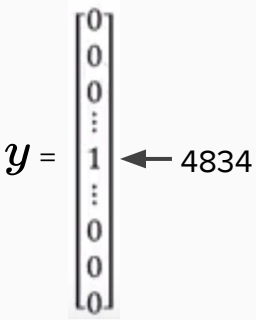
Skip-gram Model

x ⟶ y

Context, c
("orange")
6257

⟶

Target, t
("juice")
4834



$\mathbf{o}_c$ ⟶ E ⟶ $\mathbf{e}_c$ ⟶ ◯ ⟶ $\hat{y}$

softmax

softmax: $p(t|c) = \frac{e^{\theta_t^T \mathbf{e}_c}}{\sum_{j=1}^{10000} e^{\theta_j^T \mathbf{e}_c}}$

➔ θ_t : parameter associated with output *t*

loss: $L(\hat{y}, y) = -\sum_{i=1}^{10000} y_i \log \hat{y}_i$

Do not fully exploit statistical information regarding word co-occurrences.

GloVe (global vectors for word representation)

- Co-occurrence probabilities for context words ice and steam with selected target words from a 6 billion token corpus:

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

- Objective function:

how related are those two words

how often they occur with each other

$$\min \sum_{i,j=1}^{|V|} (\theta_i^T e_j - \log X_{i,j})^2$$

$X_{i,j}$ = # of times word i appears in context of word j

target context

GloVe (global vectors for word representation)

- Co-occurrence probabilities for context words ice and steam with selected target words from a 6 billion token corpus:

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

- Objective function:

$$\min \sum_{i,j=1}^{|V|} f(X_{i,j})(\theta_i^T e_j - \log X_{i,j})^2$$

weighting term

$X_{i,j}$ = # of times word i appears in context of word j

target context

GloVe (global vectors for word representation)

- Co-occurrence probabilities for context words ice and steam with selected target words from a 6 billion token corpus:

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

- Objective function:

$$\min \sum_{i,j=1}^{|V|} f(X_{i,j})(\theta_i^T e_j + b_i + b'_j - \log X_{i,j})^2$$

bias

$X_{i,j}$ = # of times word i appears in context of word j

target context

Hands-on Time

Lab 3

Topic : Emoji classification with LSTM

Filename	lab3.ipynb
Data	Emoji Dataset
Target	➔ Import dataset ➔ Data exploration ➔ Data preprocessing ➔ Train a LSTM classification model
Duration	~ 20 mins

Lab 3

Classes:

code	emoji	label
:heart:	❤️	0
:baseball:	⚾	1
:smile:	😀	2
:disappointed:	😔	3
:fork_and_knife:	🍴	4

	Text	Label	emoji
0	never talk to me again	3	😞
1	I am proud of your achievements	2	😄
2	It is the worst day in my life	3	😞
3	Miss you so much	0	❤️
4	food is life	4	🍴
5	I love you mum	0	❤️
6	Stop saying bullshit	3	😞
7	congratulations on your acceptance	2	😄
8	The assignment is too long	3	😞
9	I want to go play	1	⚾

188 sentences in total

80% for Training

20% for Testing

10 Words in Longest sentence

CloudMile

Lab 3

Text
never talk to me again
I am proud of your achievements
It is the worst day in my life
Miss you so much
food is life
I love you mum
Stop saying bullshit
congratulations on your acceptance
The assignment is too long
I want to go play

Word Embeddings

Vectors
Vector 1
Vector 2
Vector 3
Vector 4
Vector 5
Vector 6
Vector 7
Vector 8
Vector 9
Vector 10

Classification (LSTM NN)

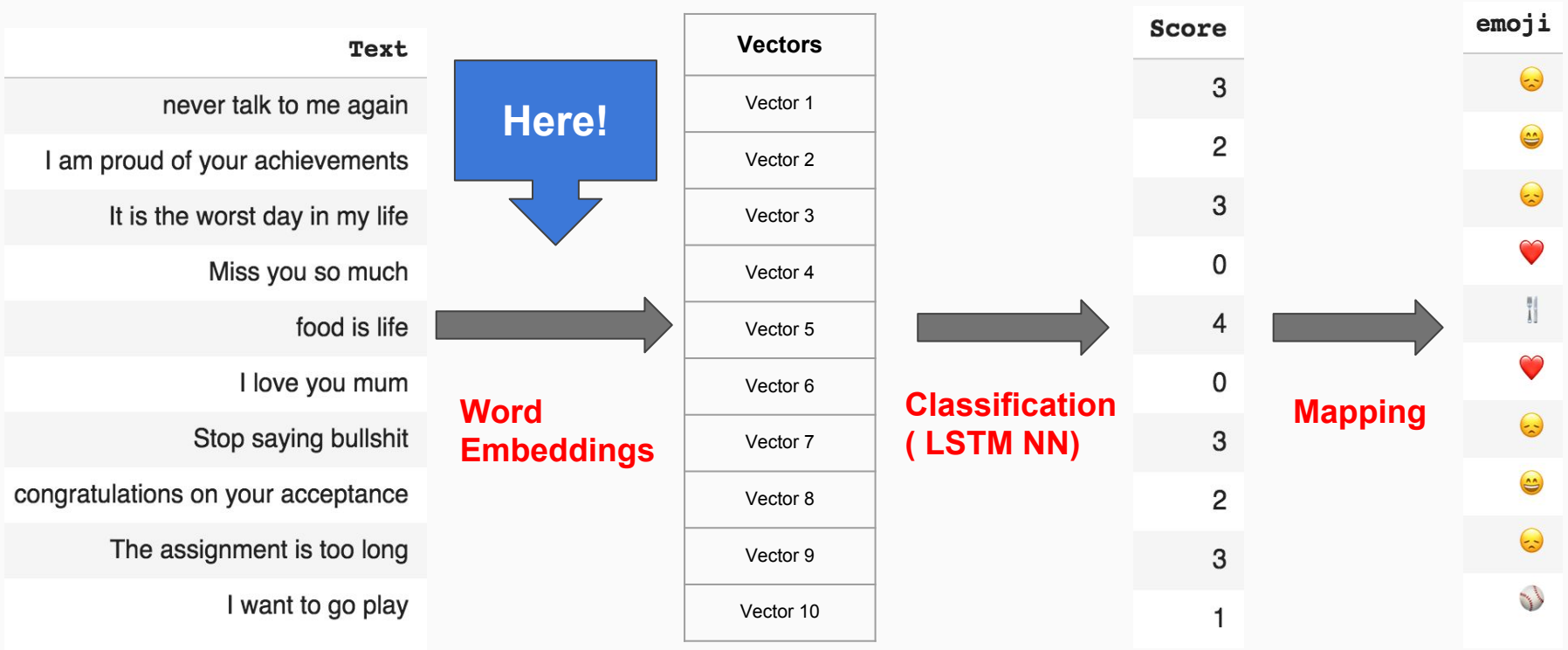
Score
3
2
3
0
4
0
3
2
3
1

Mapping

emoji
😞
😄
😞
❤️
🍴
❤️
😞
😄
😞
⚾

CloudMile

Lab 3

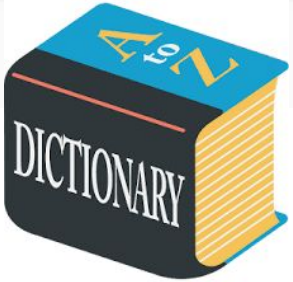


CloudMile

Lab 3

Step 1: Index words

Step 2: Create embedding layer



"I love you"

"I will be ready"

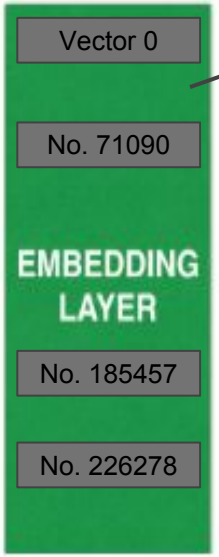
word_to_index

[185457, 226278, 394475, 0, 0]

[185457, 387696, 71090, 302254, 0]

array of indexes: shape = (2, maxLen)

Step 3: Map the input word to index



Step 4: Get the vectors of input Sentence

$[e_{185457}, e_{226278}, e_{394475}, \vec{0}, \vec{0}]$

$[e_{185457}, e_{387696}, e_{71090}, e_{302254}, \vec{0}]$

array of word vectors: shape = (2, maxLen, 50)

CloudMile

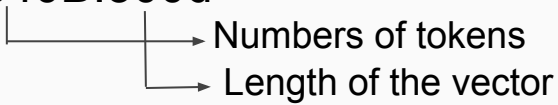
GloVe: Global Vectors for Word Representation

- Unsupervised learning algorithm for obtaining vector representations for word
- Built by Stanford

Download pre-trained word vectors

- Pre-trained word vectors. This data is made available under the Public Domain Dedication and License v1.0 whose full text can be found at: http://www.opendatacommons.org/licenses/pddl/1.0/.
  - Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 50d, 100d, 200d, & 300d vectors, 822 MB download): glove.6B.zip
  - Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors, 1.75 GB download): glove.42B.300d.zip
  - Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors, 2.03 GB download): glove.840B.300d.zip
  - Twitter (2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 100d, & 200d vectors, 1.42 GB download): glove.twitter.27B.zip
- Ruby script for preprocessing Twitter data

- Files:
  glove.6B.50d
  glove.6B.100d
  glove.840B.300d



- Link: https://nlp.stanford.edu/projects/glove/

GloVe

1. Nearest neighbors

Words closest to **frog**…...

- frogs
- toad
- litoria
- leptodactylidae
- rana
- lizard
- eleutherodactylus

3. litoria

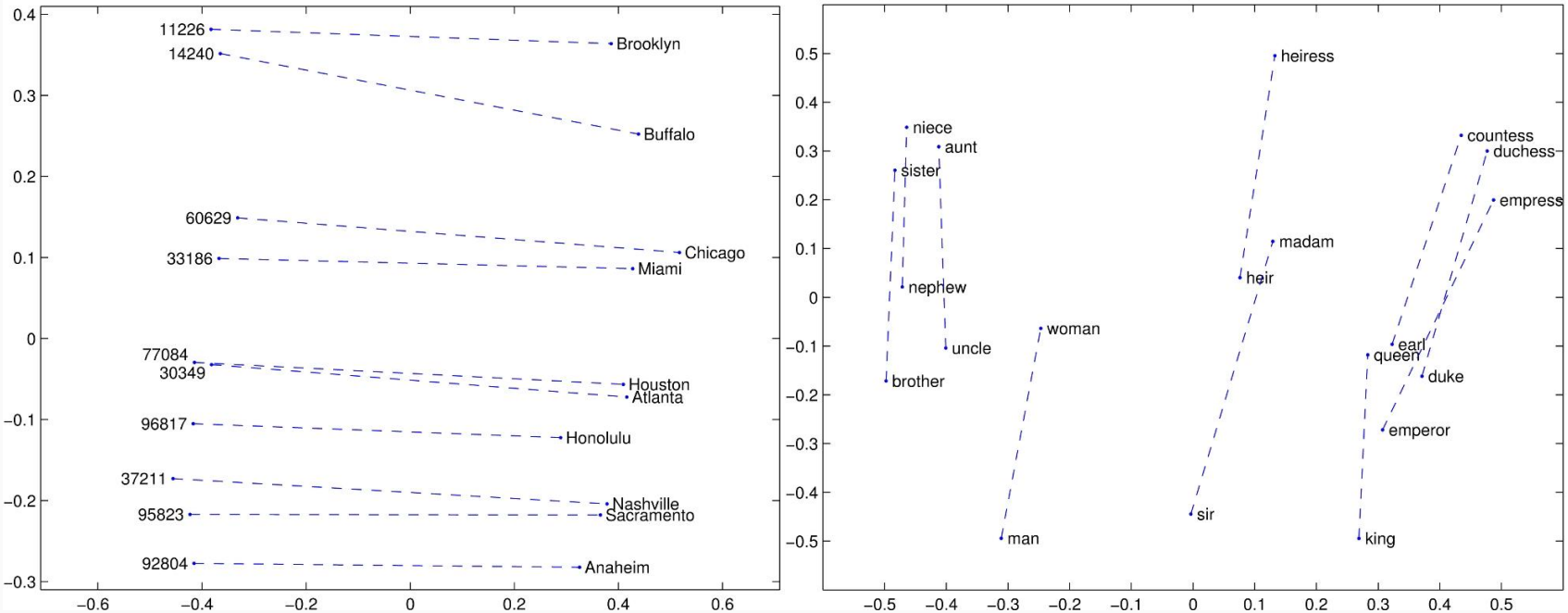
4. leptodactylidae

5. rana

7. eleutherodactylus

Lab 3 - Glove

2. Linear substructures



CloudMile

Lab 3 - What's in GloVe

```
'for': array([ 0.15272 ,  0.36181 , -0.22168 ,  0.066051,  0.13029 ,  0.37075 ,
       -0.75874 , -0.44722 ,  0.22563 ,  0.10208 ,  0.054225,  0.13494 ,
       -0.43052 , -0.2134  ,  0.56139 , -0.21445 ,  0.077974,  0.10137 ,
       -0.51306 , -0.40295 ,  0.40639 ,  0.23309 ,  0.20696 , -0.12668 ,
       -0.50634 , -1.7131  ,  0.077183, -0.39138 , -0.10594 , -0.23743 ,
        3.9552  ,  0.66596 , -0.61841 , -0.3268  ,  0.37021 ,  0.25764 ,
        0.38977 ,  0.27121 ,  0.043024, -0.34322 ,  0.020339,  0.2142  ,
        0.044097,  0.14003 , -0.20079 ,  0.074794, -0.36076 ,  0.43382 ,
       -0.084617,  0.1214  ]),
'is': array([ 6.1850e-01,  6.4254e-01, -4.6552e-01,  3.7570e-01,  7.4838e-01,
        5.3739e-01,  2.2239e-03, -6.0577e-01,  2.6408e-01,  1.1703e-01,
        4.3722e-01,  2.0092e-01, -5.7859e-02, -3.4589e-01,  2.1664e-01,
        5.8573e-01,  5.3919e-01,  6.9490e-01, -1.5618e-01,  5.5830e-02,
       -6.0515e-01, -2.8997e-01, -2.5594e-02,  5.5593e-01,  2.5356e-01,
       -1.9612e+00, -5.1381e-01,  6.9096e-01,  6.6246e-02, -5.4224e-02,
        3.7871e+00, -7.7403e-01, -1.2689e-01, -5.1465e-01,  6.6705e-02,
       -3.2933e-01,  1.3483e-01,  1.9049e-01,  1.3812e-01, -2.1503e-01,
       -1.6573e-02,  3.1200e-01, -3.3189e-01, -2.6001e-02, -3.8203e-01,
        1.9403e-01, -1.2466e-01, -2.7557e-01,  3.0899e-01,  4.8497e-01]),
```

```
print('There are {} words in the dictionary'.format(len(pretrained_dic)))
print('Length of vector for a word: {}'.format(len(pretrained_dic['is'])))
```

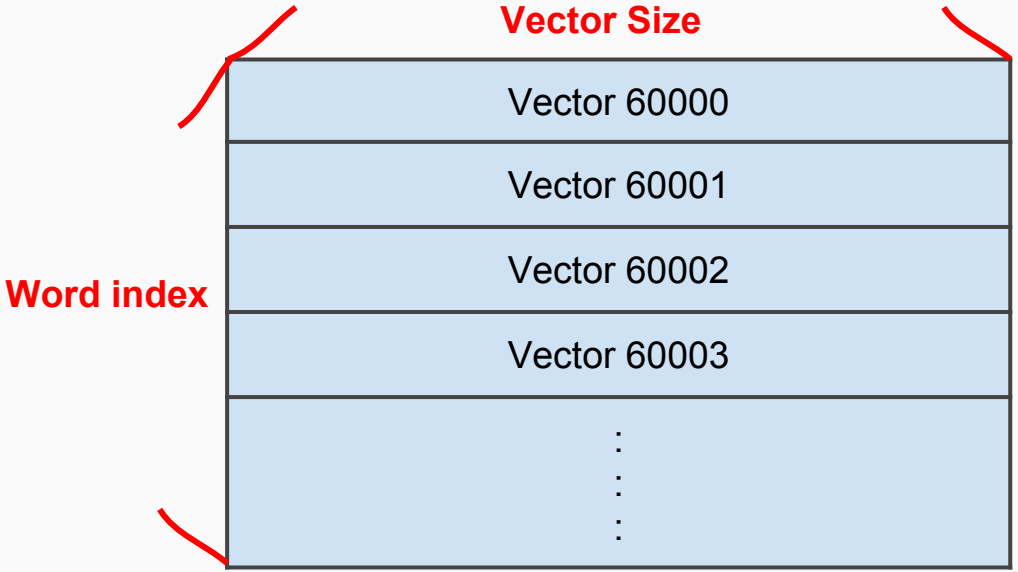
```
There are 400000 words in the dictionary
Length of vector for a word: 50
```

Lab 3

Step 1:
Index the words in GloVe

```
word_to_index

{'arraba': 60000,
 'arrabal': 60001,
 'arrack': 60002,
 'arraf': 60003,
 'arrah': 60004,
 'arraial': 60005,
 'arraign': 60006,
 'arraigned': 60007,
 'arraignment': 60008,
 'arraignments': 60009}
```

Step 2:
Create a embedding matrix ordered by index



Lab 3

Step 3: Map the input word to index

```
def sentences_to_indices(X, word_to_index, max_len):
    m = X.shape[0]
    X_indices = np.zeros((m, max_len))
    for i in range(m):
        sentence_words = X[i].lower().split()
        j = 0
        for w in sentence_words:
            X_indices[i, j] = word_to_index[w]
            j = j + 1
    return X_indices
```

```
test_sentence = np.array(['this book is nice', 'lets go shopping', 'hello'])
max_len = get_max_length(test_sentence)
test_indices = sentences_to_indices(test_sentence, word_to_index, max_len=max_len)
```

```
test_indices
```

```
array([[358159.,  80585., 192972., 260759.],
       [220929., 163236., 329657.,      0.],
       [176467.,      0.,      0.,      0.]])
```

Lab 3

‘ The book is nice ’

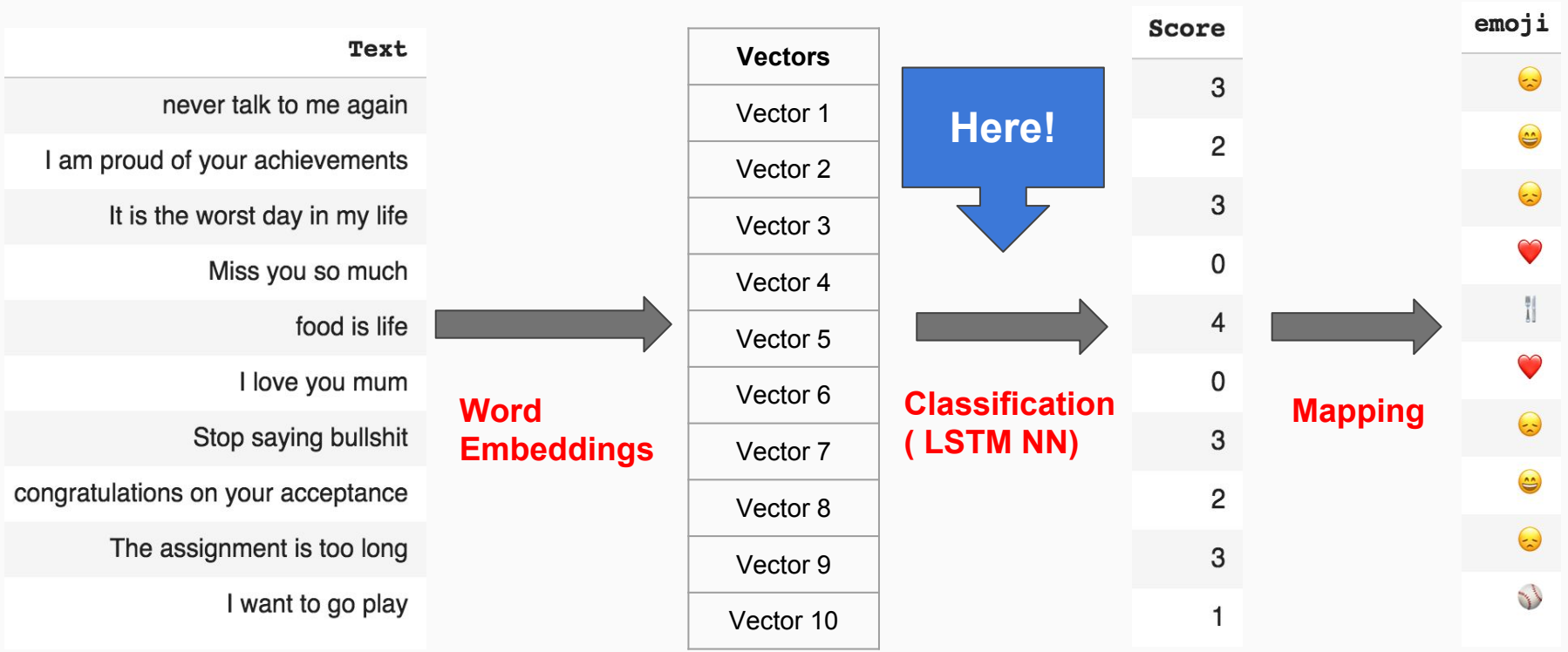
[358159 , 80585 , 192972 , 260759]

```
embedding_layer = Embedding(len(word_to_index),
                            EMBEDDING_DIM,
                            weights = [embedding_matrix],
                            input_length = MAX_SEQUENCE_LENGTH,
                            trainable = False)
```

[Vector 358159 , Vector 80585 , Vector 192972 , Vector 260759]

CloudMile

Lab 3



CloudMile

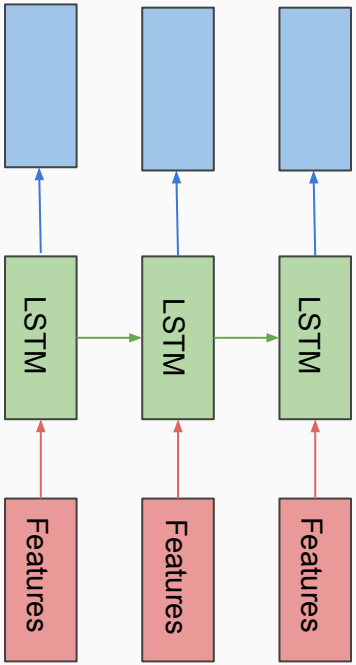
Lab 3

```
K.clear_session()

sequence_input = Input(shape = (MAX_SEQUENCE_LENGTH,), dtype = 'int32')

embedding_layer = Embedding(len(word_to_index),
                            EMBEDDING_DIM,
                            weights = [embedding_matrix],
                            input_length = MAX_SEQUENCE_LENGTH,
                            trainable = False)

embeddings = embedding_layer(sequence_input)

X = LSTM(128, return_sequences=True)(embeddings)
X = Dropout(0.5)(X)
X = LSTM(128, return_sequences=False)(X)
X = Dropout(0.5)(X)
X = Dense(5, activation='softmax')(X)

model = Model(sequence_input, X)
```

Lab 3

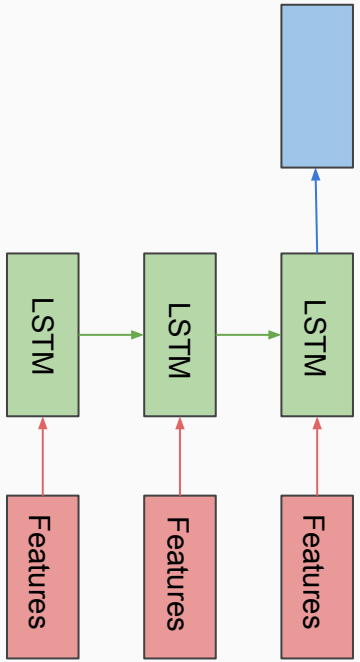
Many to many

return_sequence = True

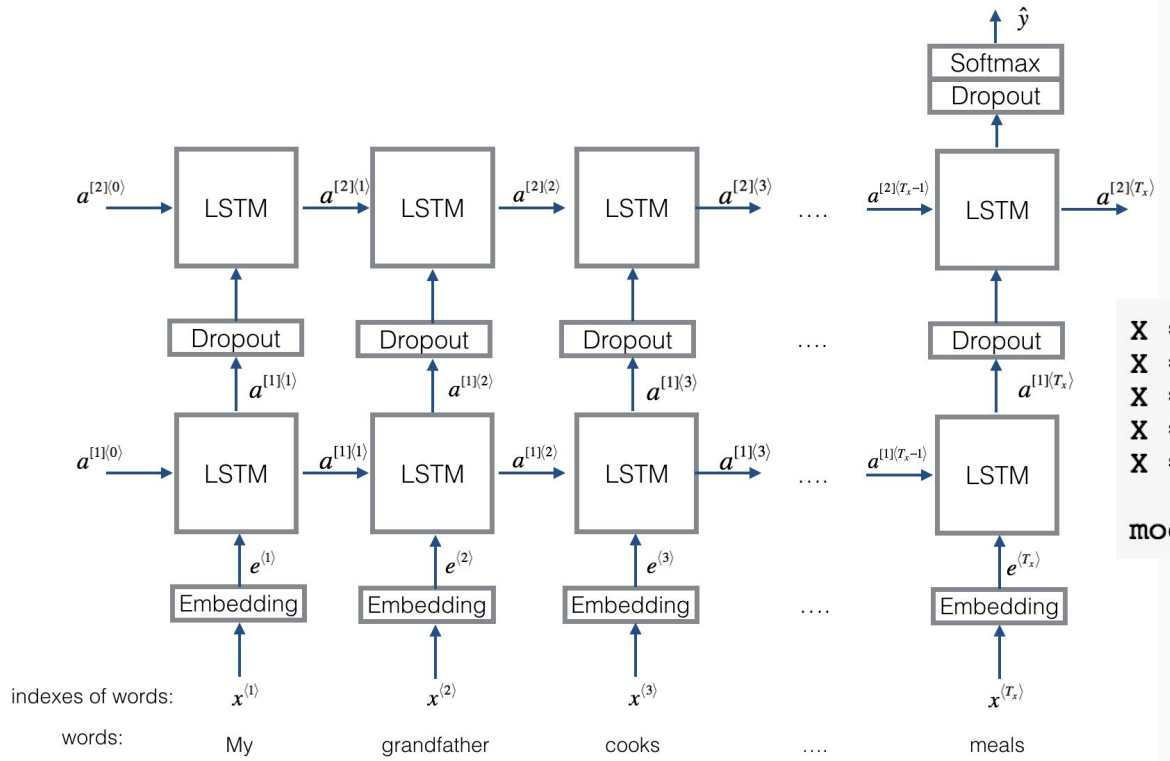


Many to one

return_sequence = False



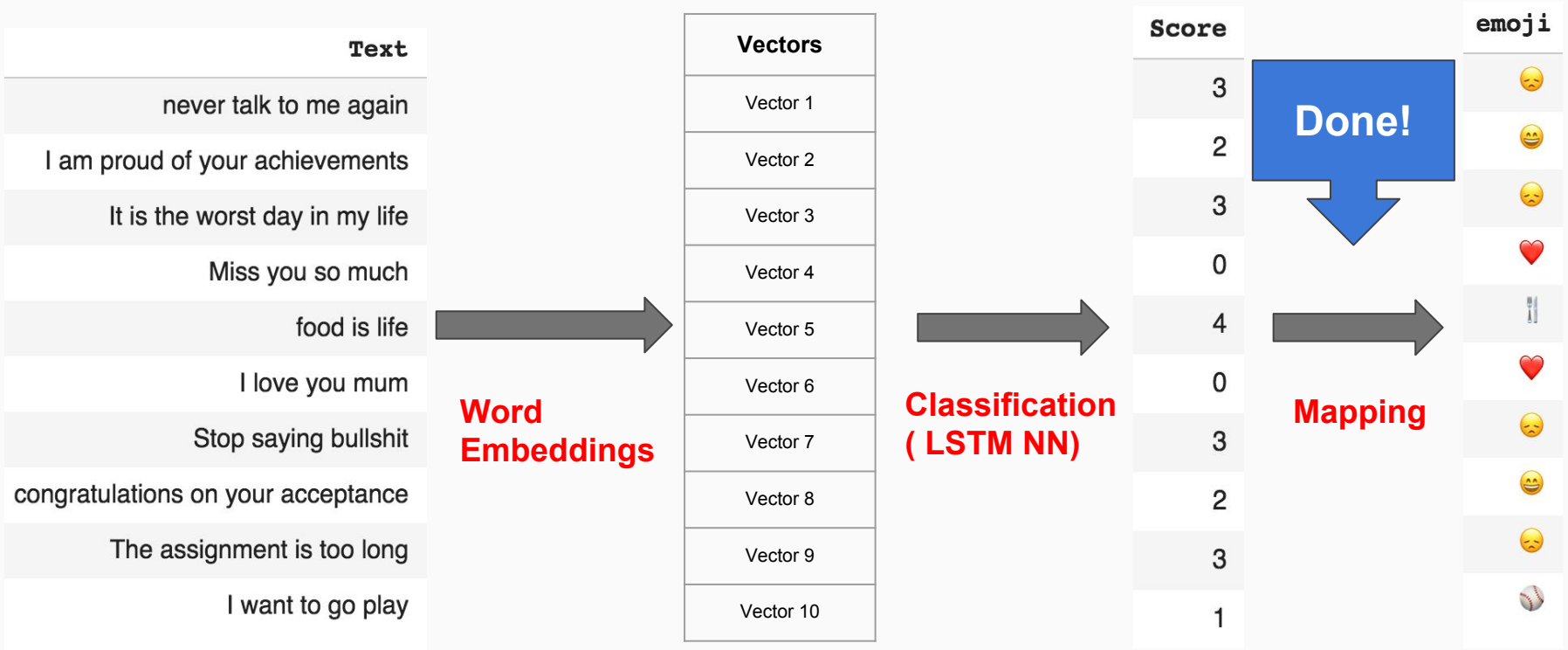
CloudMile




```
X = LSTM(128, return_sequences=True)(embe
X = Dropout(0.5)(X)
X = LSTM(128, return_sequences=False)(X)
X = Dropout(0.5)(X)
X = Dense(5, activation='softmax')(X)

model = Model(sequence_input, X)
```


CloudMile

Lab 3



CloudMile

lab3

Topic : Emoji classification with LSTM

Start coding !

Filename	lab3.ipynb
Data	Emoji Dataset
Target	➔ Import dataset ➔ Data exploration ➔ Data preprocessing ➔ Train a LSTM classification model
Duration	~ 20 mins