GGE 6341 Introduction to Artificial Intelligence in Geomatics
(Formerly, Advanced Techniques in Remote Sensing)

Presentation II

Long Short-Term Memory (LSTM) - RNN

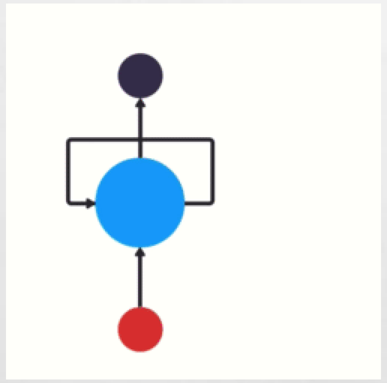
Vaasudevan Srinivasan

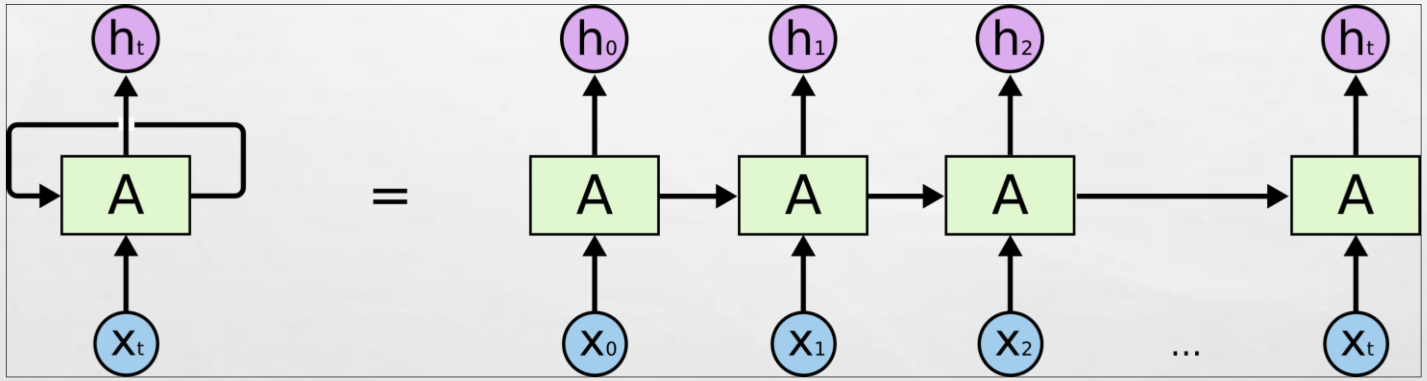
Nov 19, 2019

Glimpse from Presentation-1

What is Recurrent Neural Network (RNN) ?

It is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence



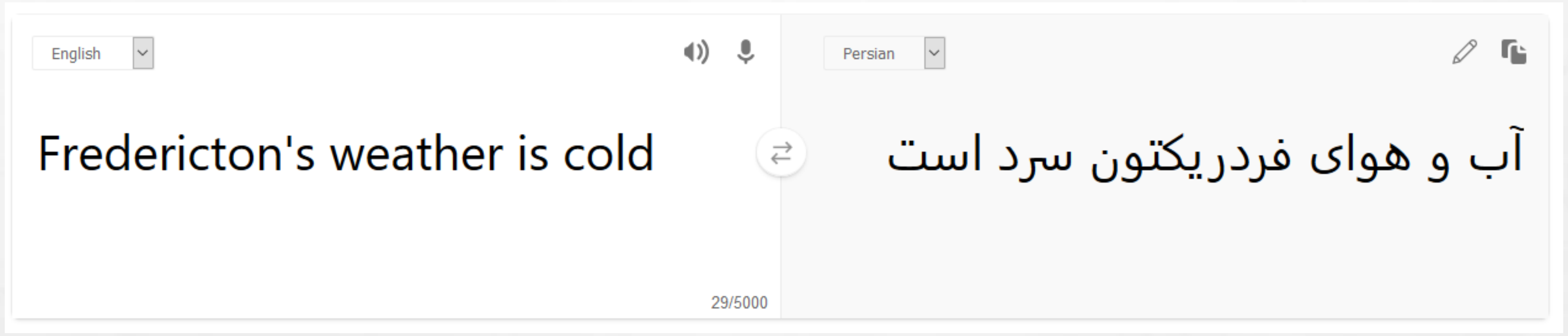


Unrolled RNN network

Perfect Roommate Example

How RNN can help predict the next food the roommate is going to cook (based on Weather and Sequence)

Glimpse from Presentation-1

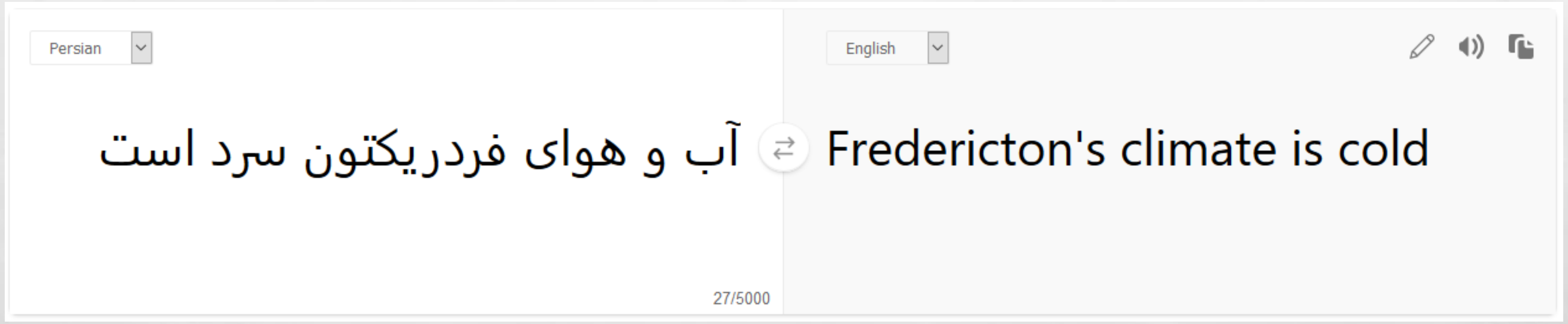

English

Fredericton's weather is cold

29/5000

Persian

آب و هوای فردریکتون سرد است


Persian

آب و هوای فردریکتون سرد است

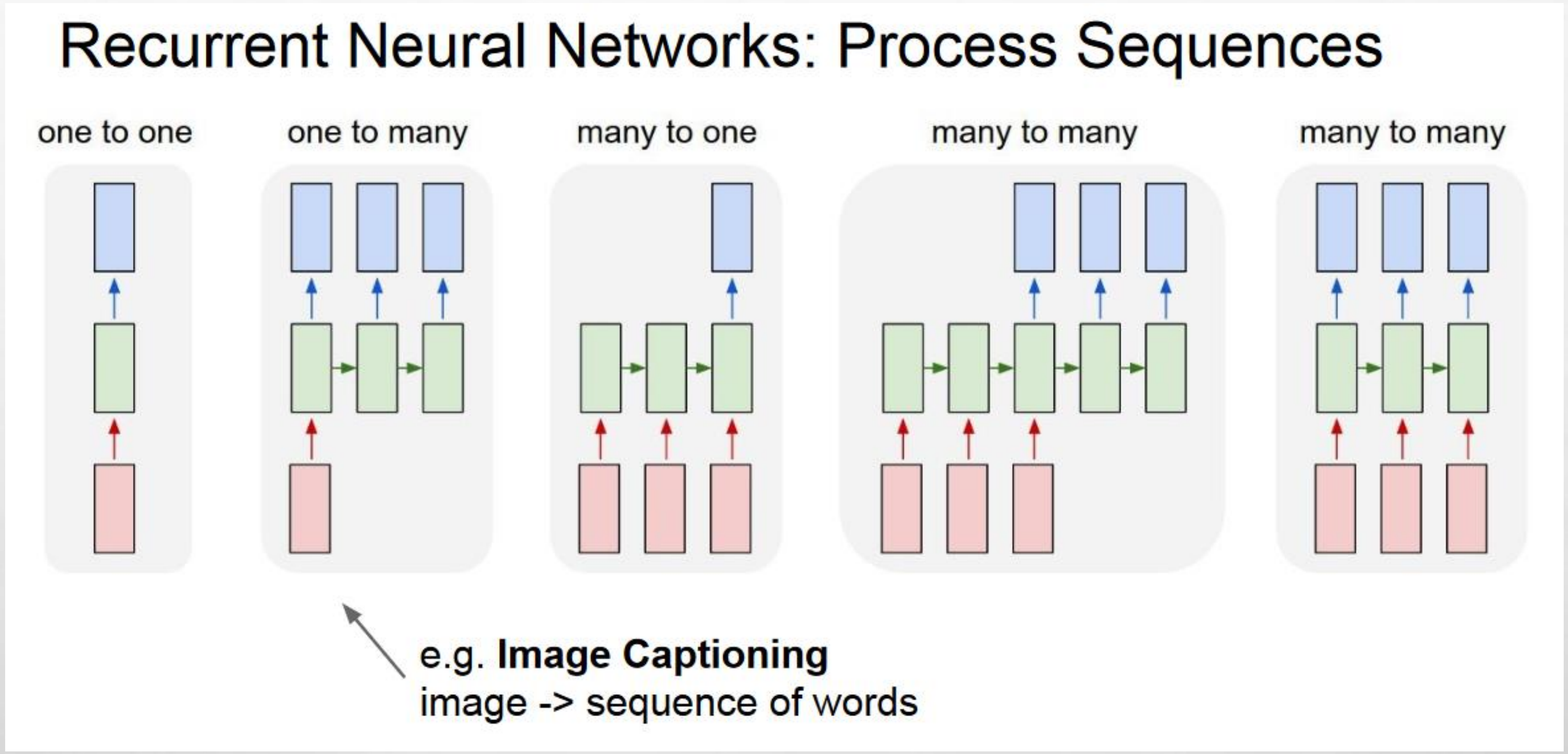
27/5000

English

Fredericton's climate is cold

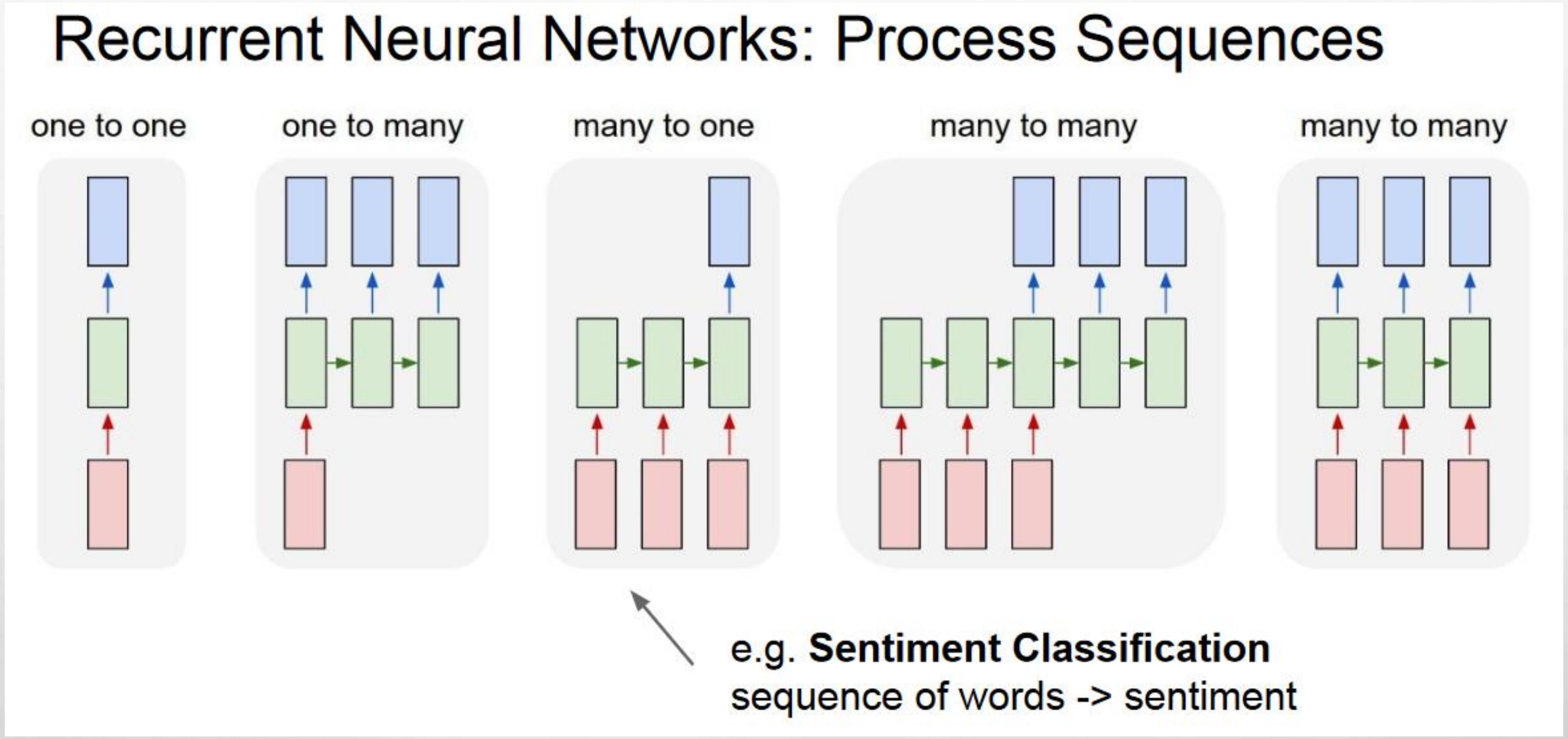
App: https://www.bing.com/translator

Glimpse from Presentation-1


Recurrent Neural Networks: Process Sequences

e.g. **Image Captioning**
image -> sequence of words

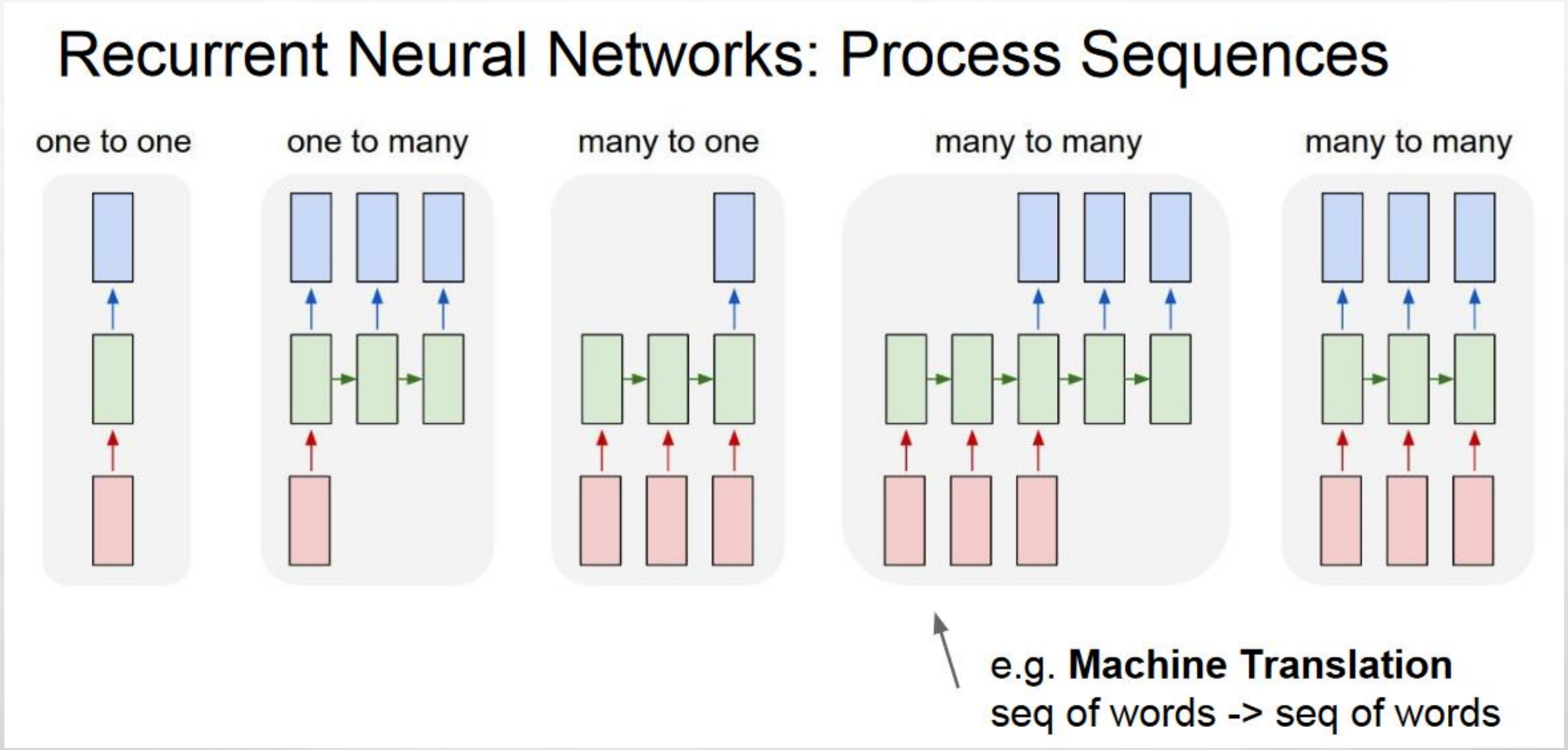
Glimpse from Presentation-1



Recurrent Neural Networks: Process Sequences

e.g. **Sentiment Classification**
sequence of words -> sentiment

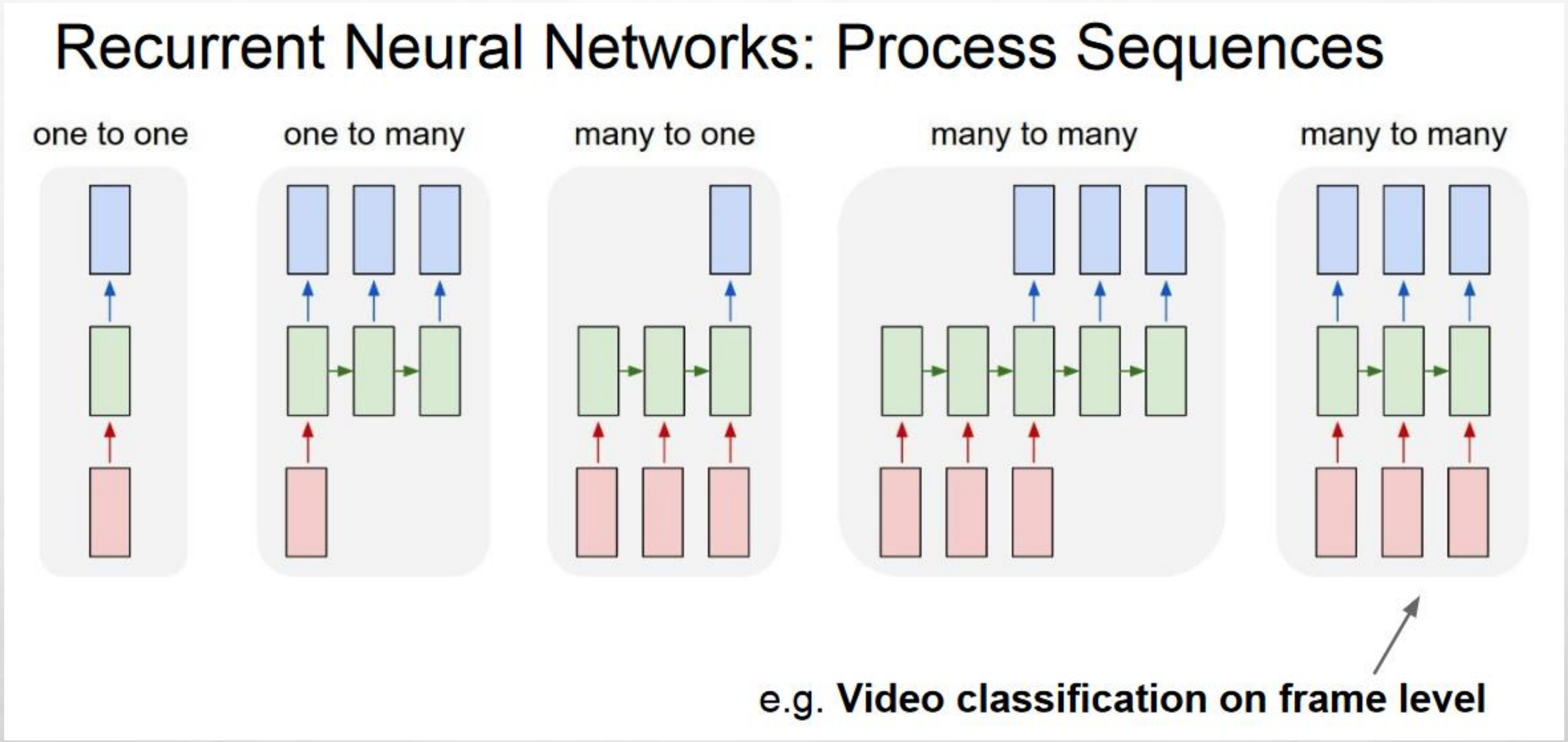
Glimpse from Presentation-1



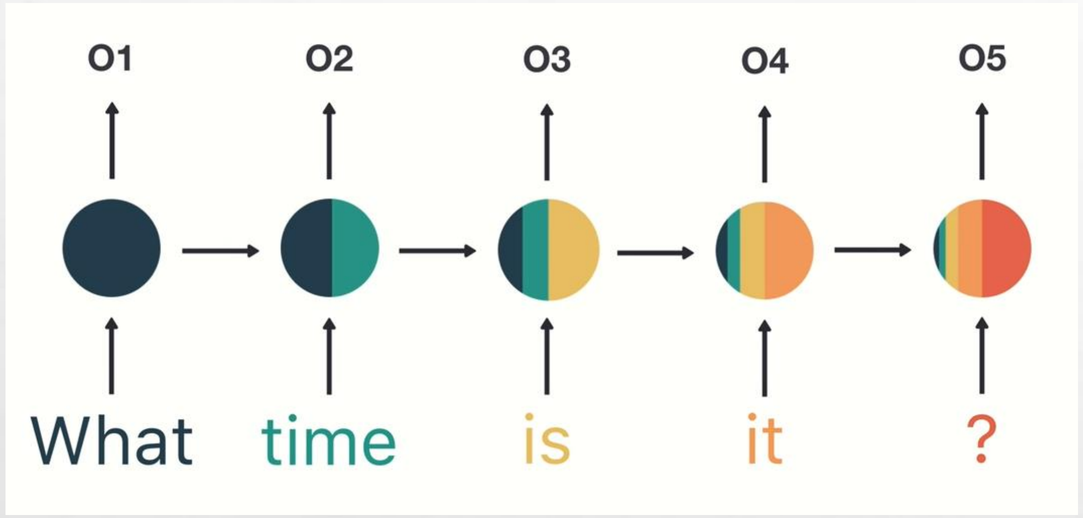
Recurrent Neural Networks: Process Sequences

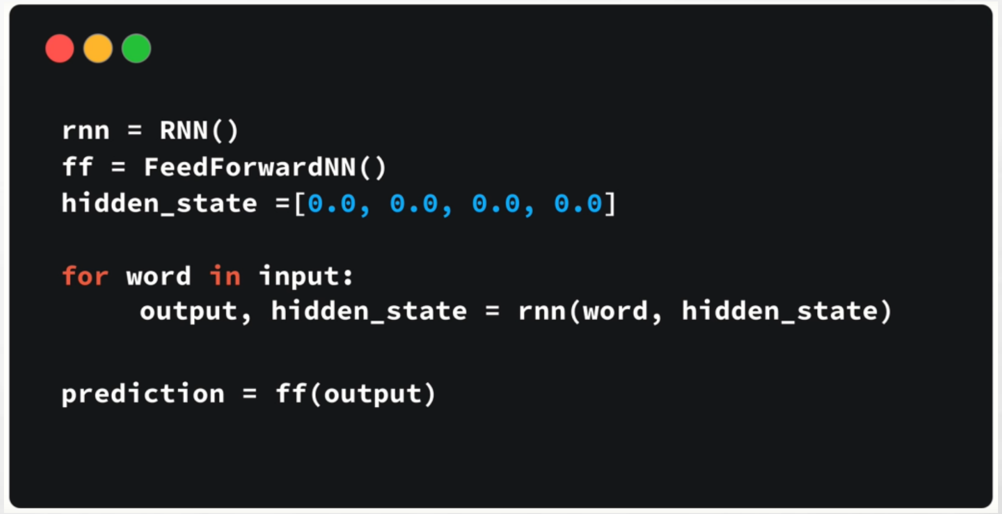
e.g. **Machine Translation**
seq of words -> seq of words

Glimpse from Presentation-1


Recurrent Neural Networks: Process Sequences

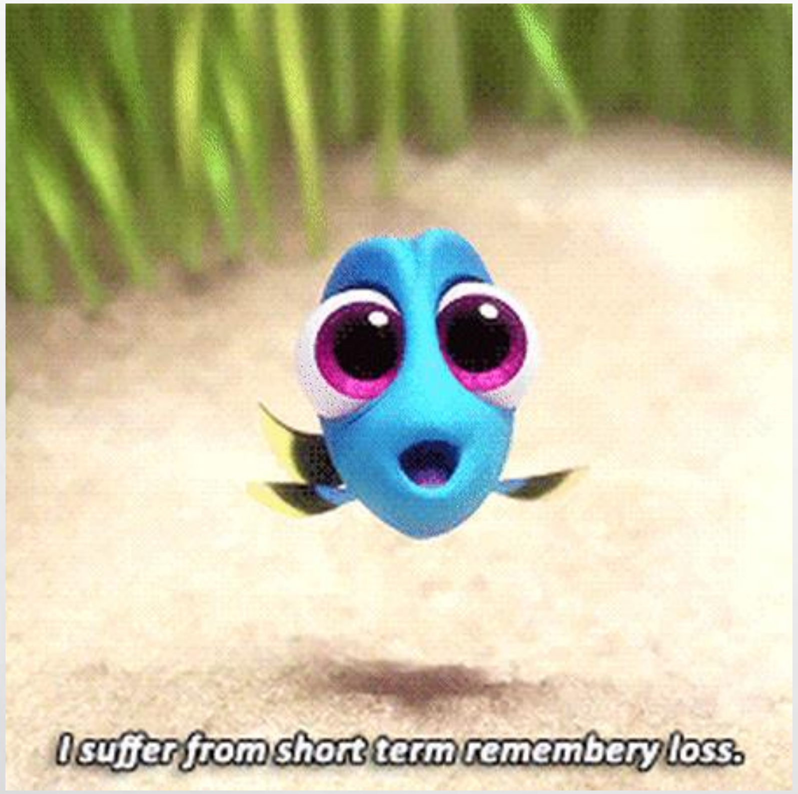
Example of RNN

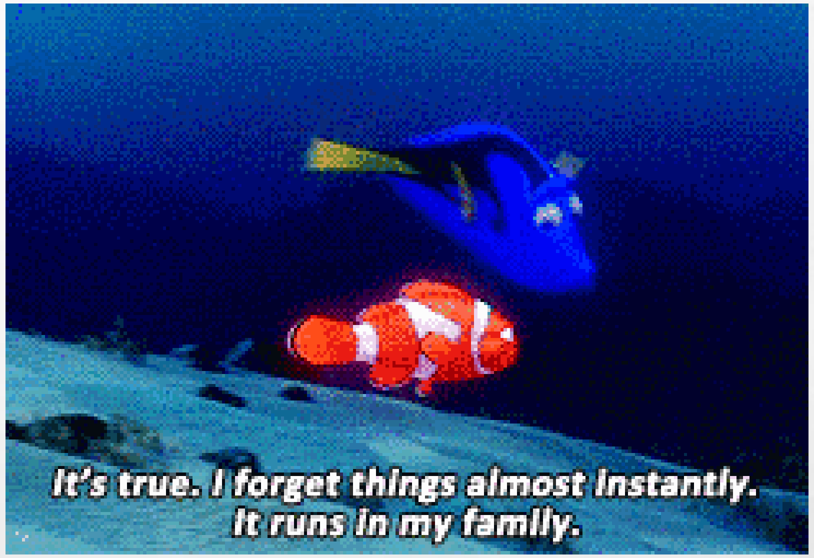




```
rnn = RNN()
ff = FeedForwardNN()
hidden_state =[0.0, 0.0, 0.0, 0.0]

for word in input:
     output, hidden_state = rnn(word, hidden_state)

prediction = ff(output)
```

RNN Problems





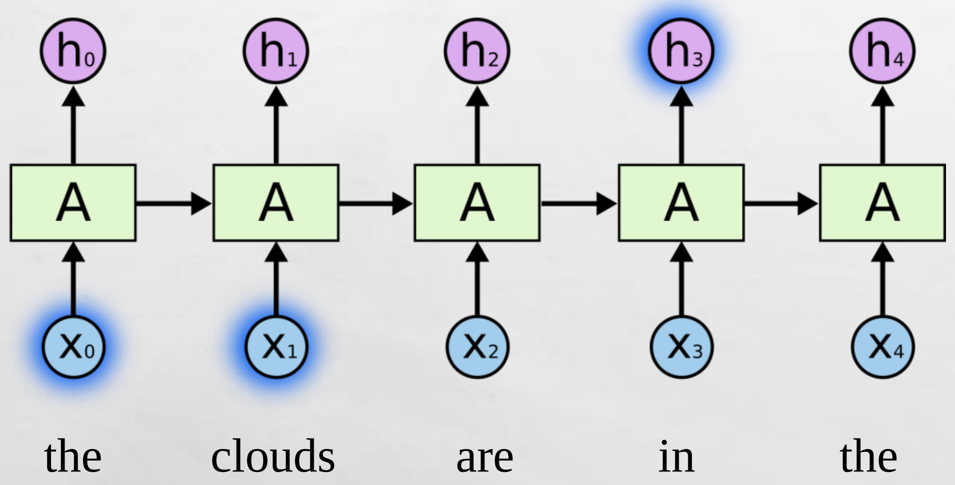
Vanishing Gradient Problem

Doesn't learn Long range dependencies across time

Why not RNN

Sometimes, we only need to look at recent information to perform the present task

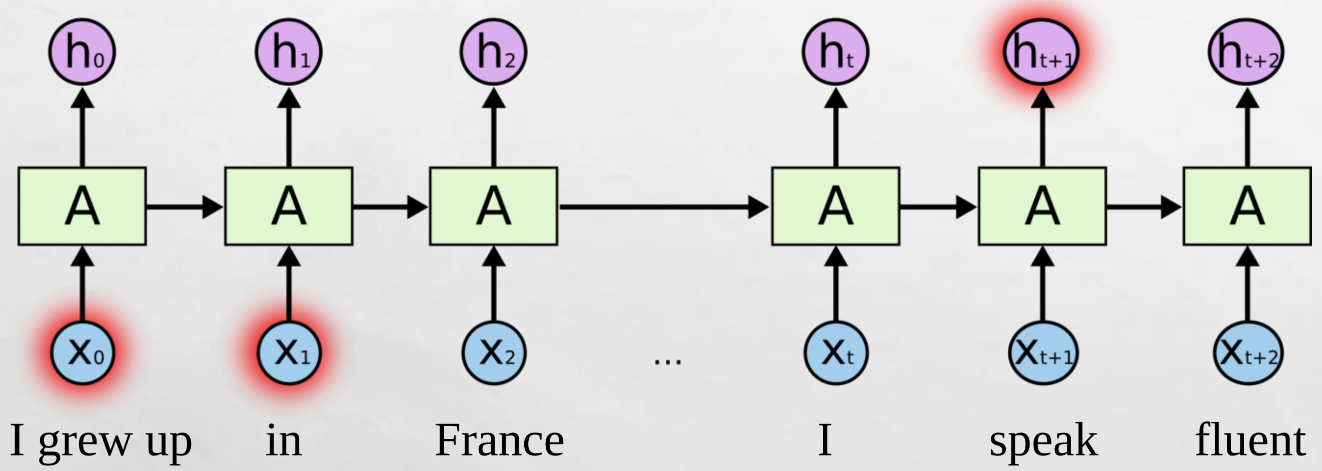
Example: the clouds are in the __________



Why not RNN

Sometimes, we need more context

Example: I grew up in France… I speak fluent __________



Why LSTM

In theory RNN are absolutely capable of handling such long-term dependencies. A **human** could carefully pick parameters for them to solve toy problems of this form.

Sadly, in practice, RNNs don't seem to be able to learn them.

The problem was explored in depth by Hochreiter (1991) [German] and Bengio, et al. (1994), who found some pretty fundamental reasons why it might be difficult.

LSTM's overcome both Vanishing Gradient Problem and also learns long-range dependencies through **Gating mechanism**

LSTM (Long Short-term memory)

Proposed in **1997** by Hochreiter and Schmidhuber

Some Applications:

As of 2016 Google used LSTM for speech recognition on the smartphone, for the smart assistant Allo and for Google Translate.

Apple uses LSTM for the "Quicktype" function on the iPhone and for Siri.

Amazon uses LSTM for Amazon Alexa

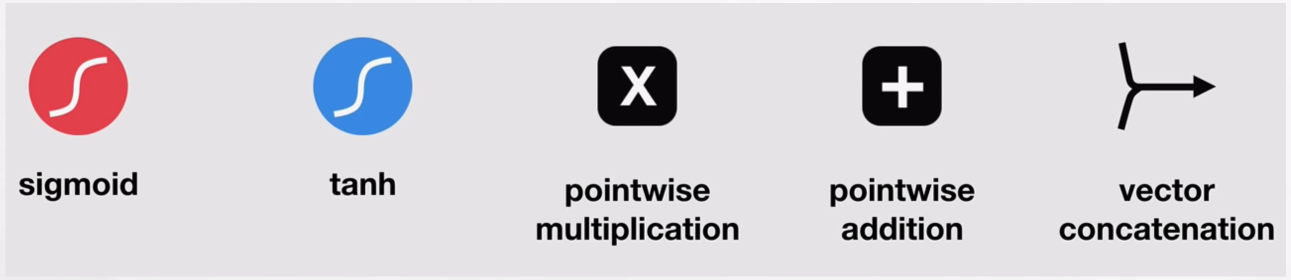
In 2017, Facebook performed some 4.5 billion automatic translations every day using long short-term memory networks.

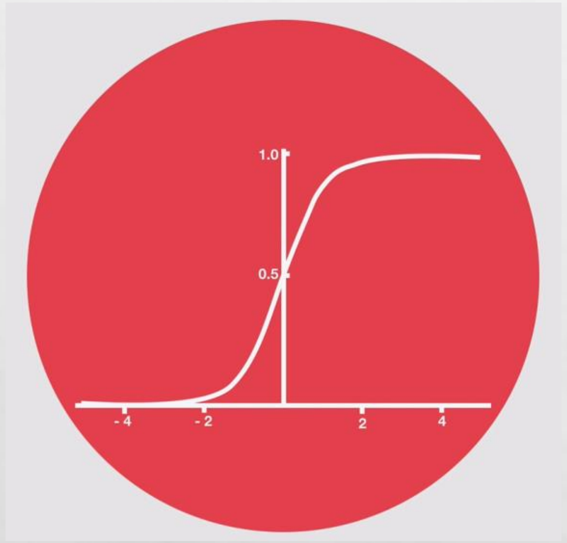
Intuition behind LSTM

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!

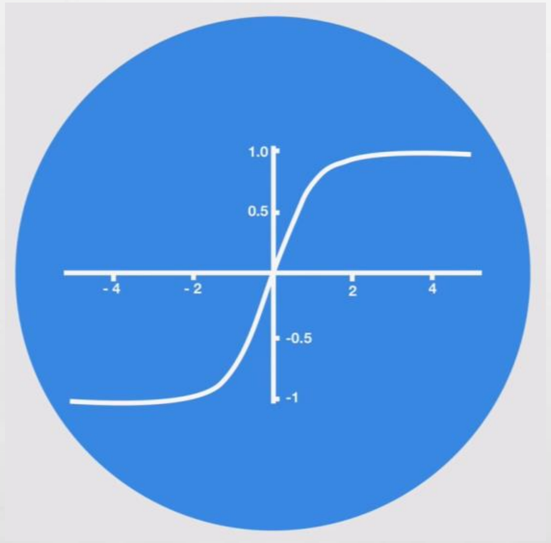
Amazing! This box of cereal gave me a **perfectly balanced breakfast**, as all things should be. I only ate half of it but **will definitely be buying again!**

LSTM terms to know



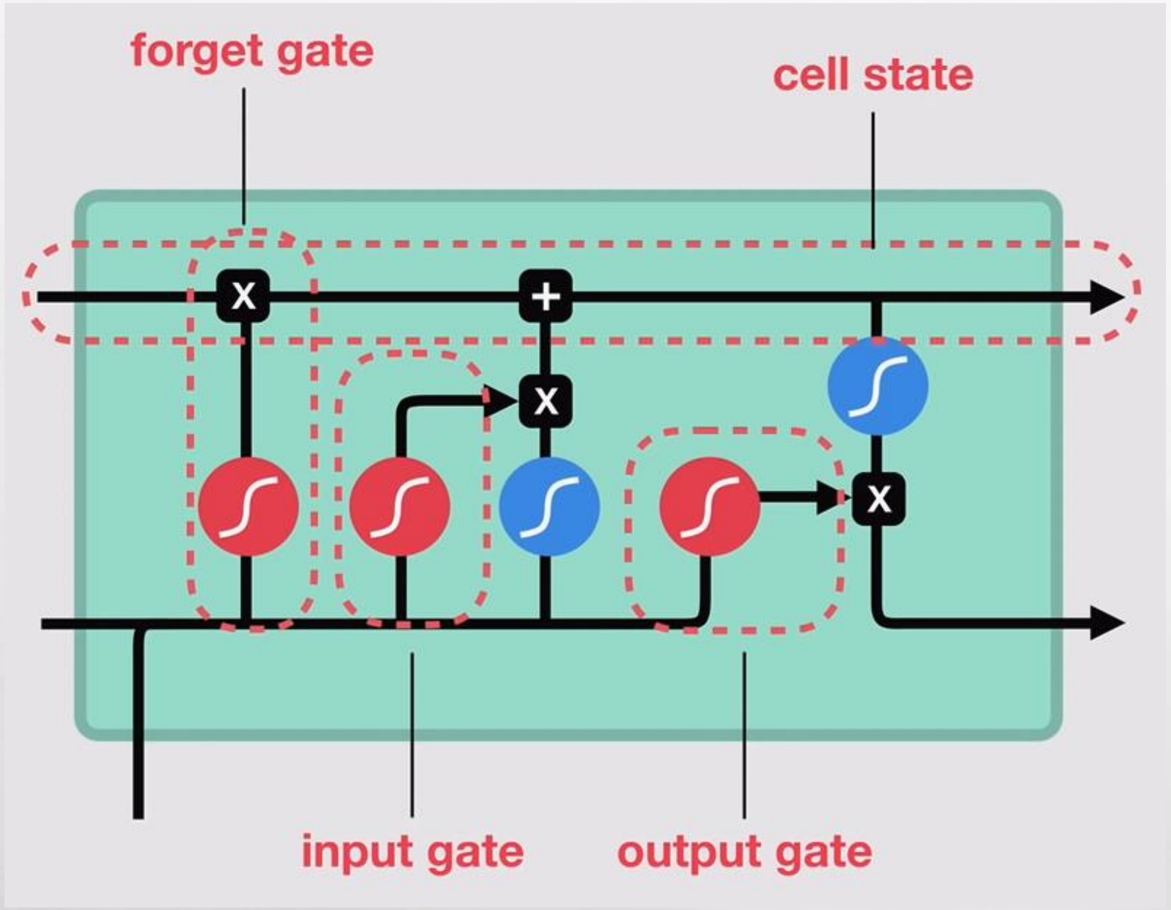


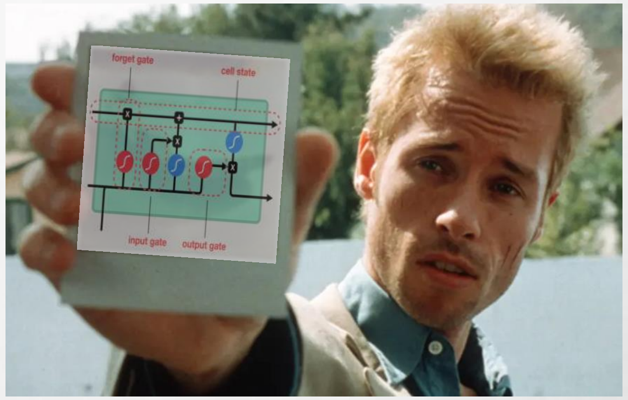
Sigmoid (0 to 1)

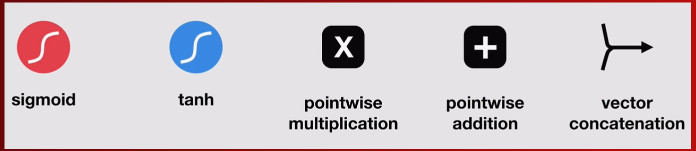


tanh (-1 to 1)

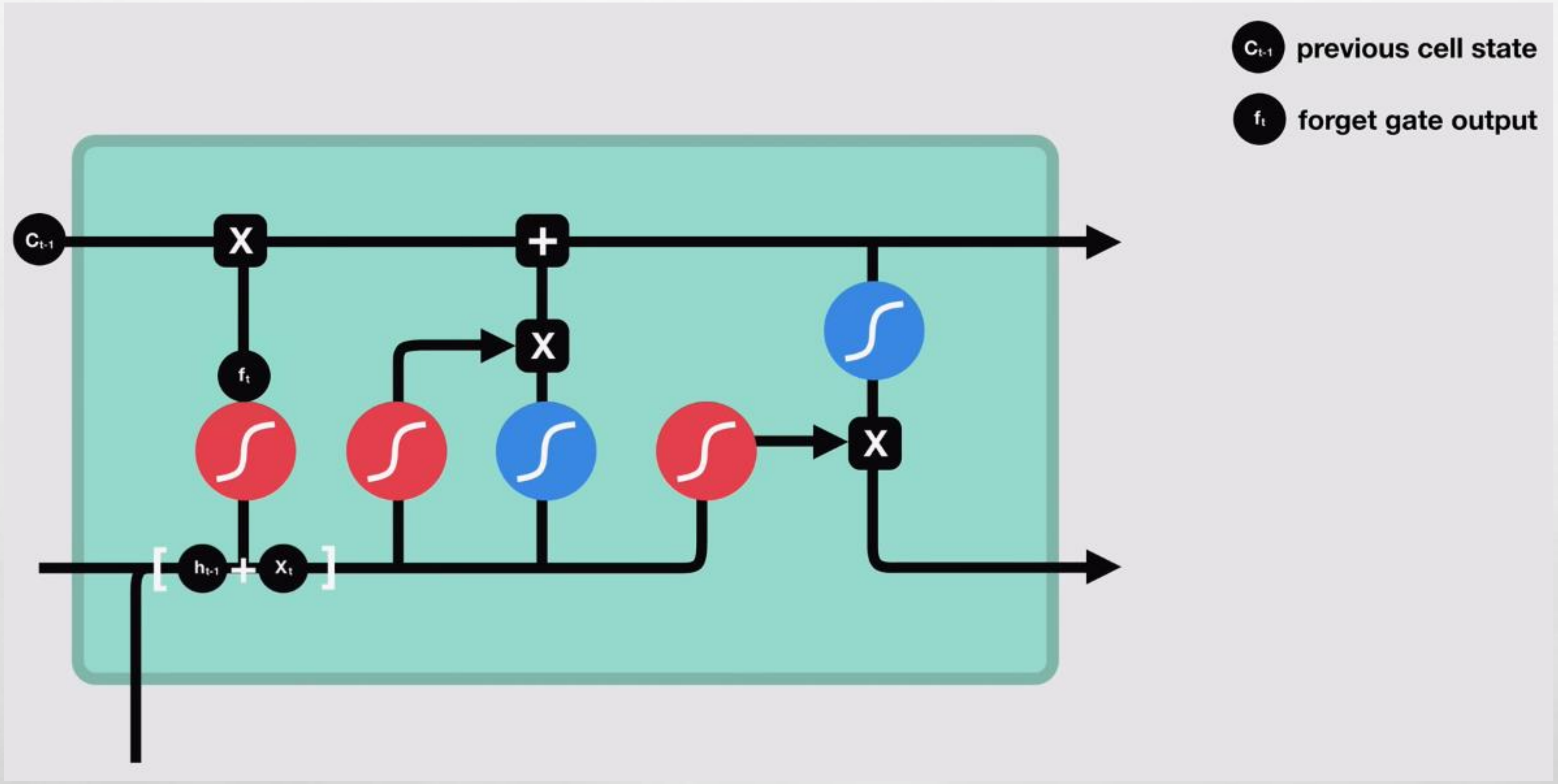
What is LSTM on the inside ?

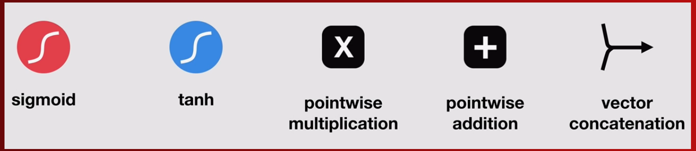




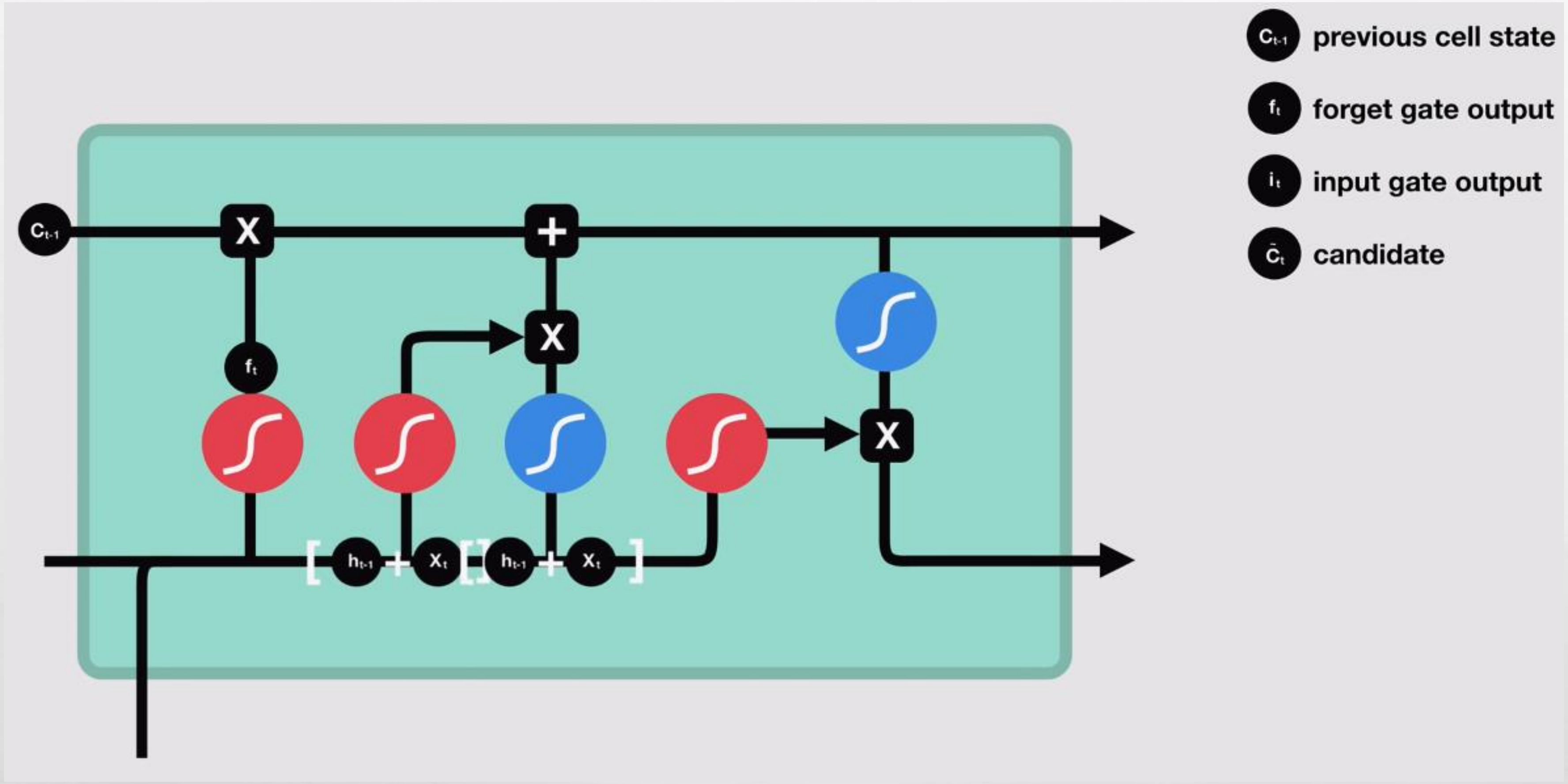


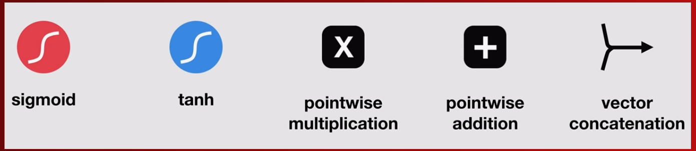
How LSTM works ? { Step 1 }



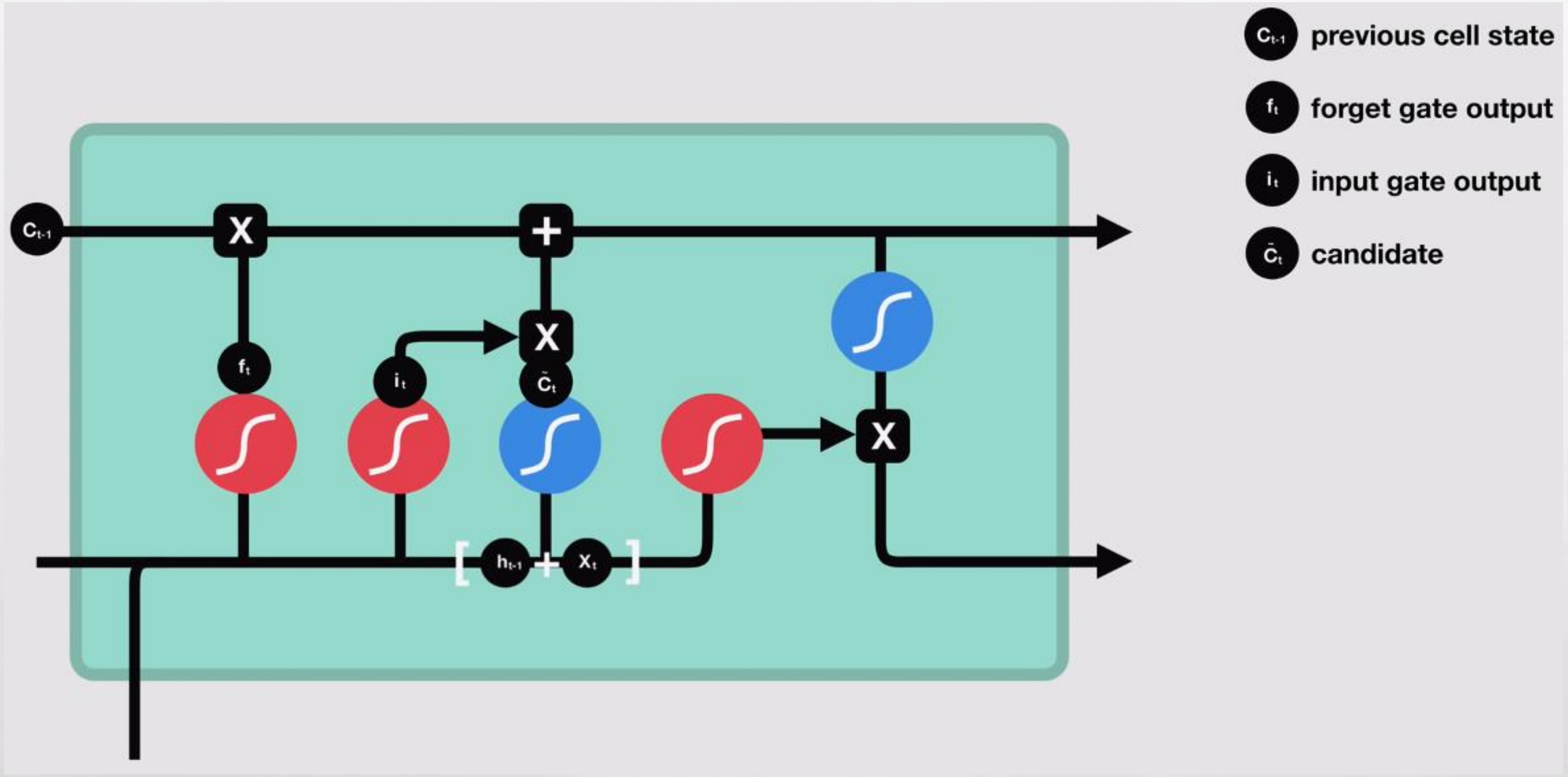


How LSTM works ? { Step 2}



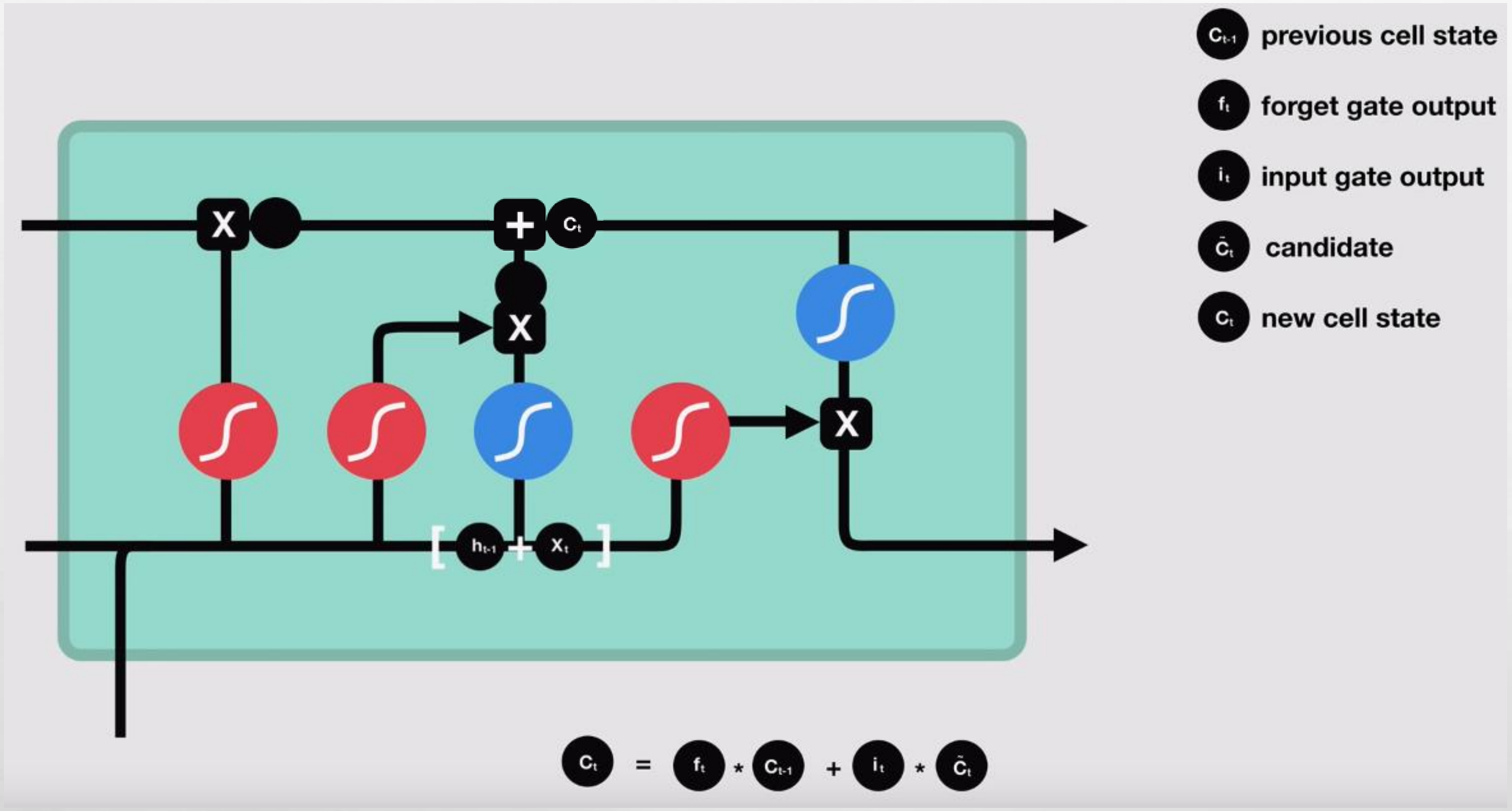


How LSTM works ? { Step 3}



sigmoid
tanh
pointwise multiplication
pointwise addition
vector concatenation

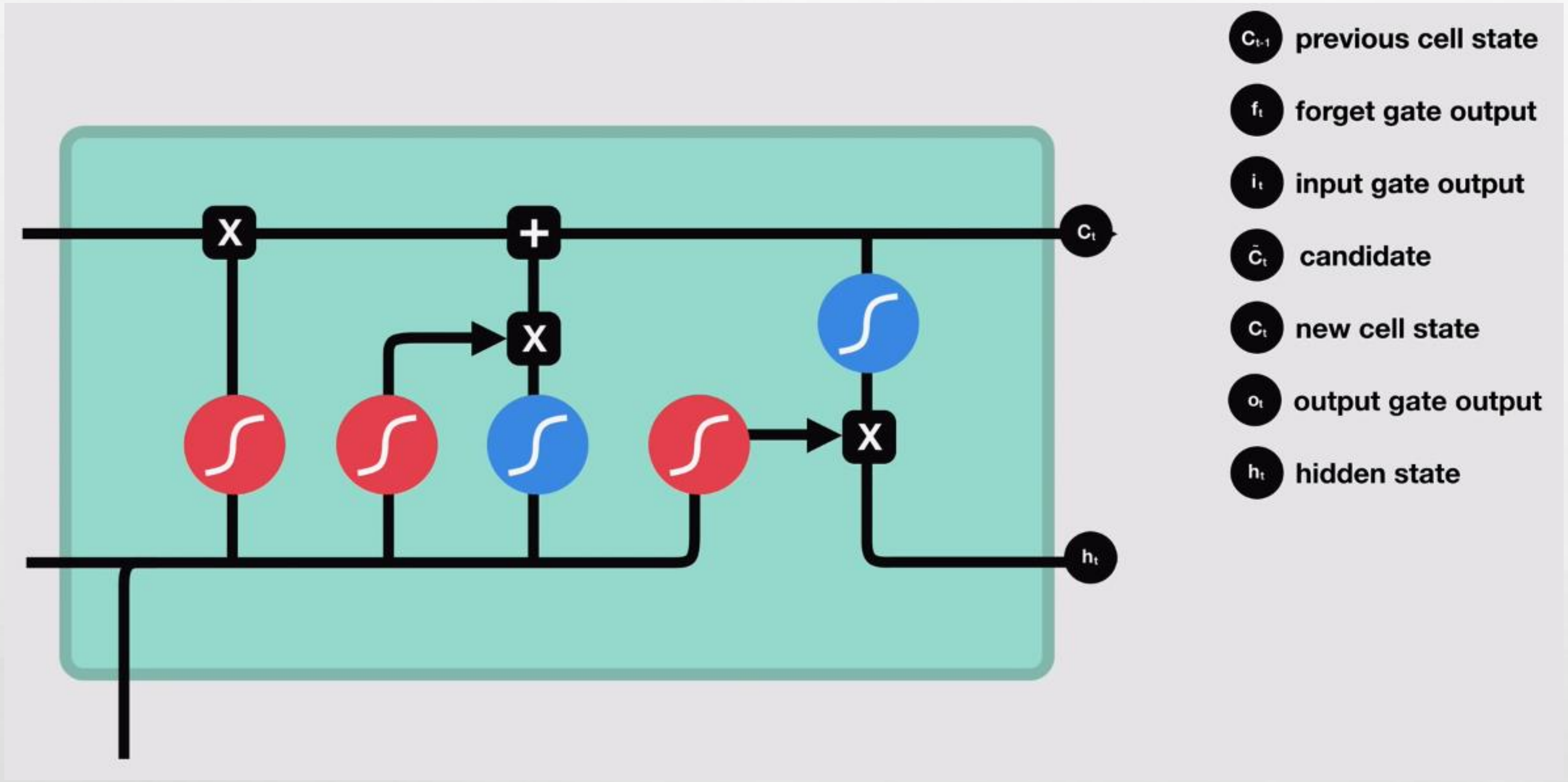
How LSTM works ? { Step 4}

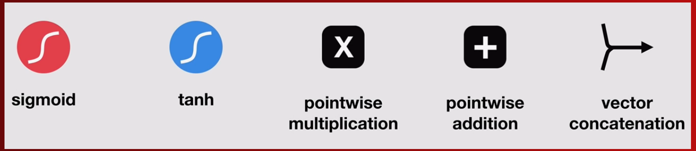


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

sigmoid
tanh
pointwise multiplication
pointwise addition
vector concatenation

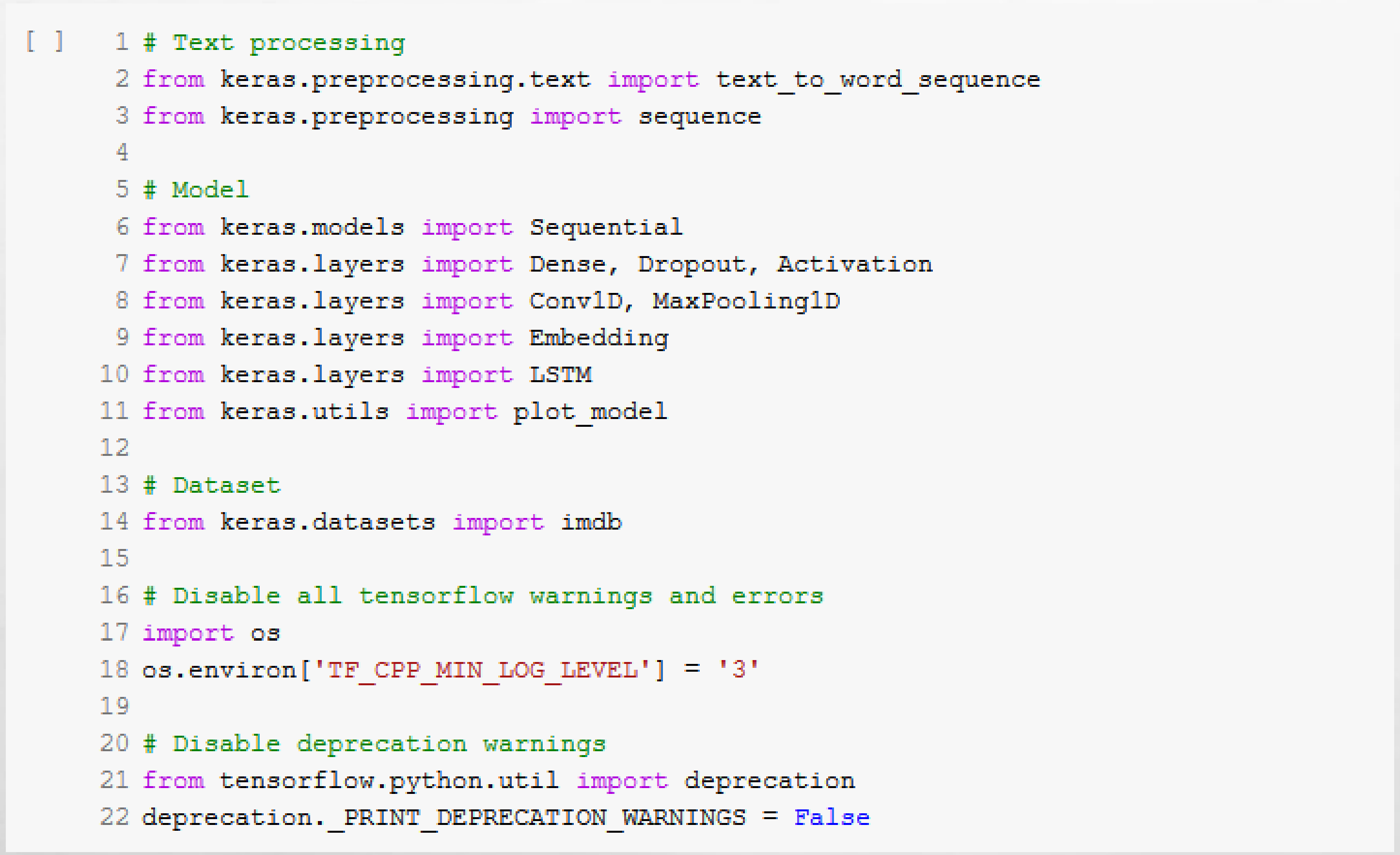
How LSTM works ? { Step 5}





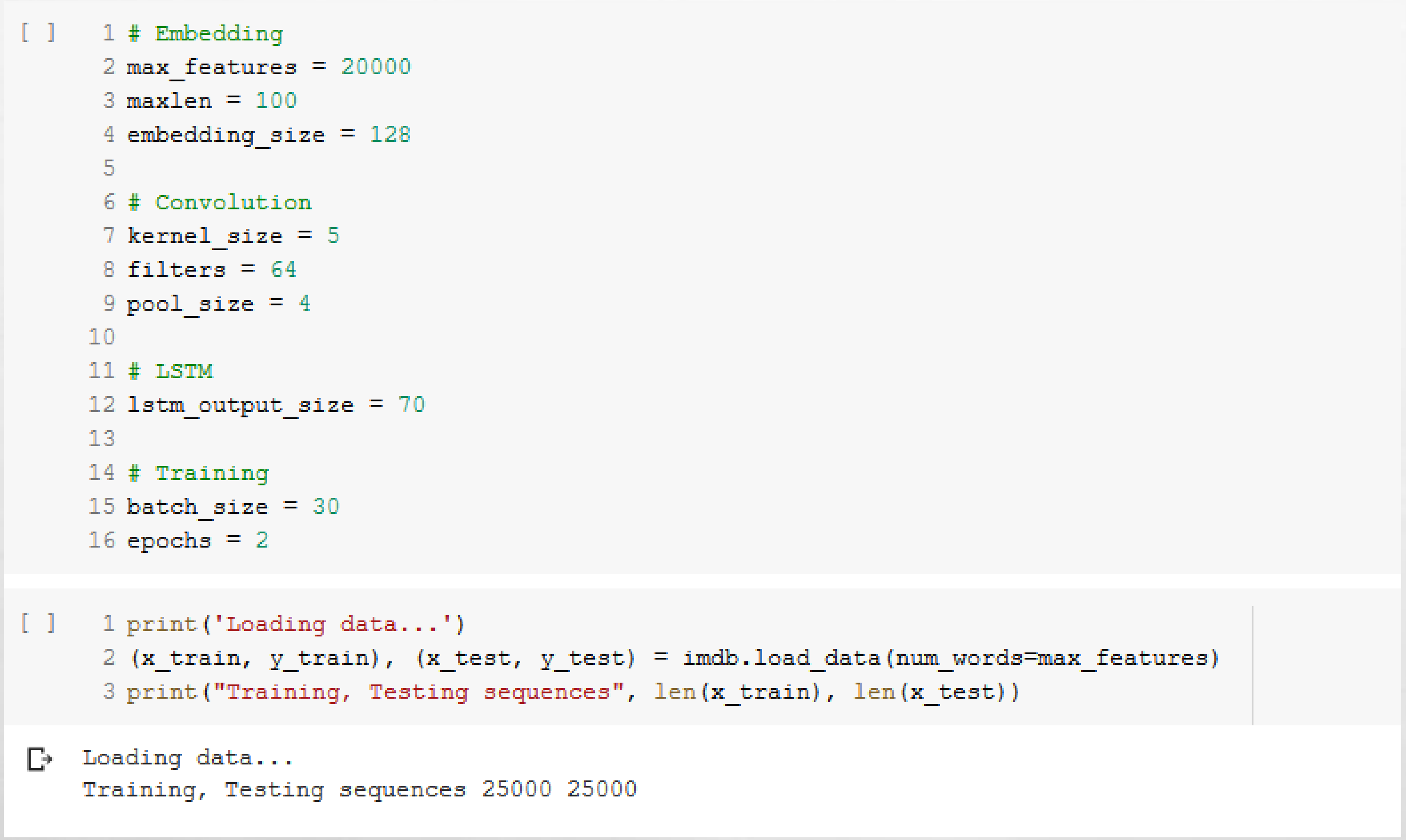
Example - Colab - https://colab.research.google.com/drive/1YRdRViykuWm9dIk2NVDKovDx7PDEwrmF

Example - Colab - https://colab.research.google.com/drive/1YRdRViykuWm9dIk2NVDKovDx7PDEwrmF



```
[ ]
# Text processing
from keras.preprocessing.text import text_to_word_sequence
from keras.preprocessing import sequence

# Model
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.layers import Conv1D, MaxPooling1D
from keras.layers import Embedding
from keras.layers import LSTM
from keras.utils import plot_model

# Dataset
from keras.datasets import imdb

# Disable all tensorflow warnings and errors
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

# Disable deprecation warnings
from tensorflow.python.util import deprecation
deprecation._PRINT_DEPRECATION_WARNINGS = False
```

Example - Colab - https://colab.research.google.com/drive/1YRdRViykuWm9dIk2NVDKovDx7PDEwrmF

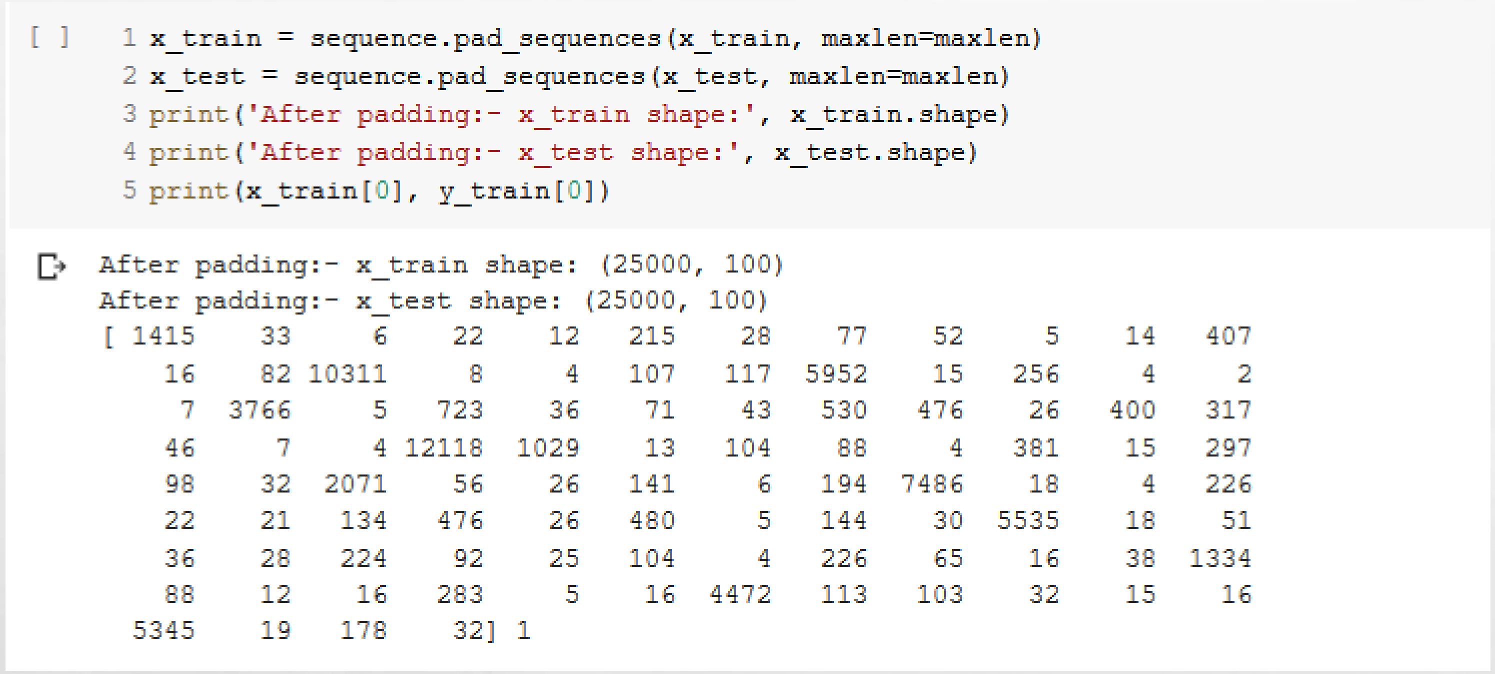


```
[ ]
# Embedding
max_features = 20000
maxlen = 100
embedding_size = 128

# Convolution
kernel_size = 5
filters = 64
pool_size = 4

# LSTM
lstm_output_size = 70

# Training
batch_size = 30
epochs = 2
```

```
[ ]
print('Loading data...')
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)
print("Training, Testing sequences", len(x_train), len(x_test))
```

```
Loading data...
Training, Testing sequences 25000 25000
```

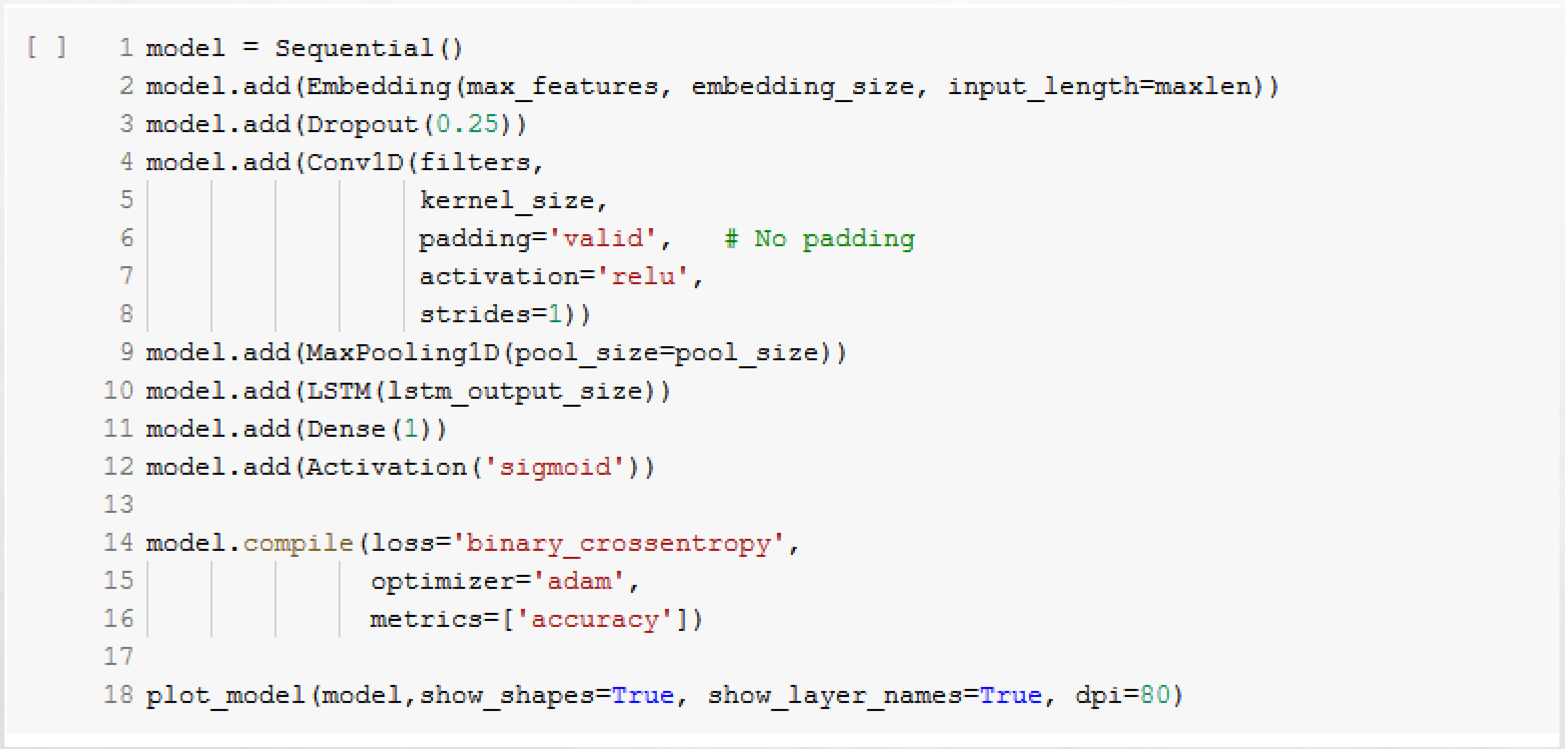
Example - Colab - https://colab.research.google.com/drive/1YRdRViykuWm9dIk2NVDKovDx7PDEwrmF



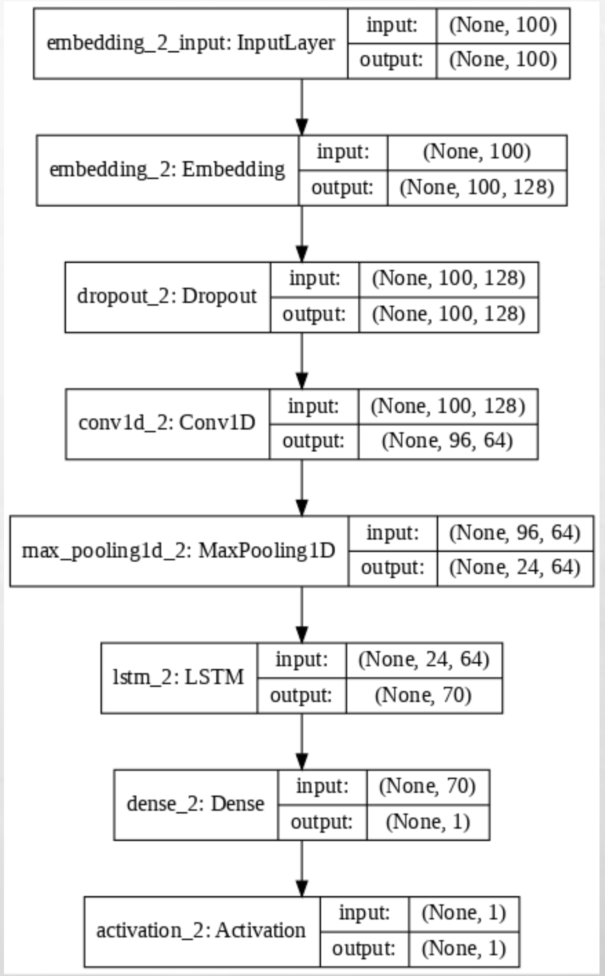
```
[ ]
x_train = sequence.pad_sequences(x_train, maxlen=maxlen)
x_test = sequence.pad_sequences(x_test, maxlen=maxlen)
print('After padding:- x_train shape:', x_train.shape)
print('After padding:- x_test shape:', x_test.shape)
print(x_train[0], y_train[0])
```

```
After padding:- x_train shape: (25000, 100)
After padding:- x_test shape: (25000, 100)
[ 1415    33     6    22    12   215    28    77    52     5    14   407
    16    82 10311     8     4   107   117  5952    15   256     4     2
     7  3766     5   723    36    71    43   530   476    26   400   317
    46     7     4 12118  1029    13   104    88     4   381    15   297
    98    32  2071    56    26   141     6   194  7486    18     4   226
    22    21   134   476    26   480     5   144    30  5535    18    51
    36    28   224    92    25   104     4   226    65    16    38  1334
    88    12    16   283     5    16  4472   113   103    32    15    16
  5345    19   178    32] 1
```

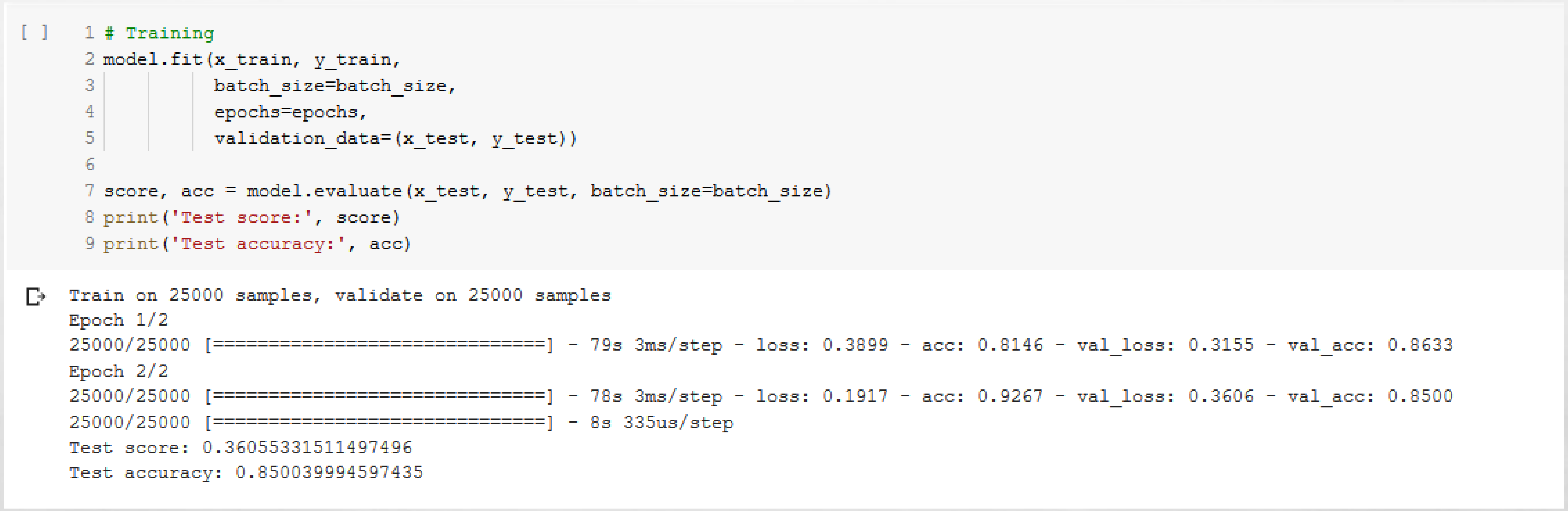
Example - Colab - https://colab.research.google.com/drive/1YRdRViykuWm9dIk2NVDKovDx7PDEwrmF



```
[ ]
model = Sequential()
model.add(Embedding(max_features, embedding_size, input_length=maxlen))
model.add(Dropout(0.25))
model.add(Conv1D(filters,
                 kernel_size,
                 padding='valid',    # No padding
                 activation='relu',
                 strides=1))
model.add(MaxPooling1D(pool_size=pool_size))
model.add(LSTM(lstm_output_size))
model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

plot_model(model,show_shapes=True, show_layer_names=True, dpi=80)
```



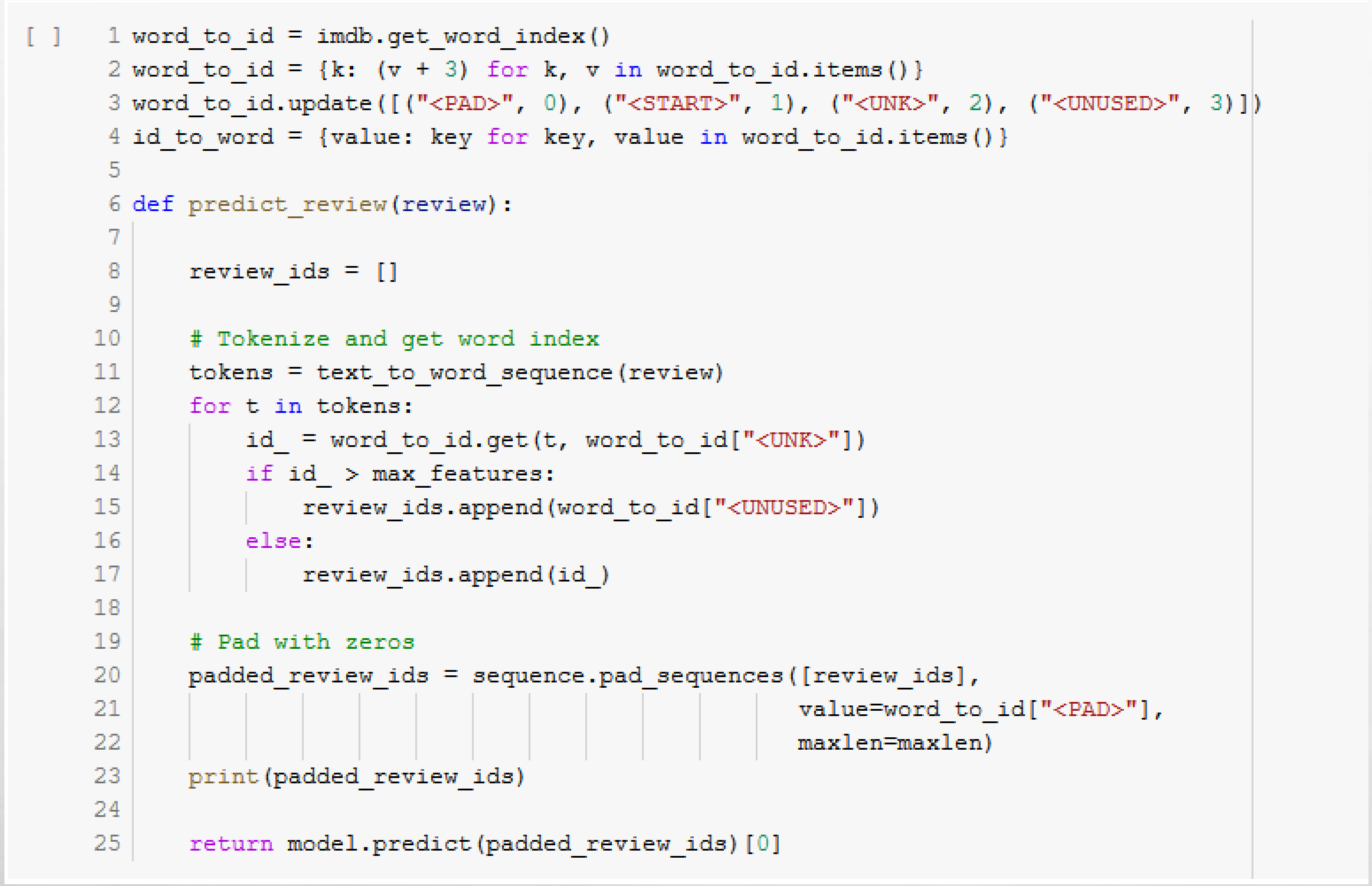
Example - Colab - https://colab.research.google.com/drive/1YRdRViykuWm9dIk2NVDKovDx7PDEwrmF



```
# Training
model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          validation_data=(x_test, y_test))

score, acc = model.evaluate(x_test, y_test, batch_size=batch_size)
print('Test score:', score)
print('Test accuracy:', acc)
```

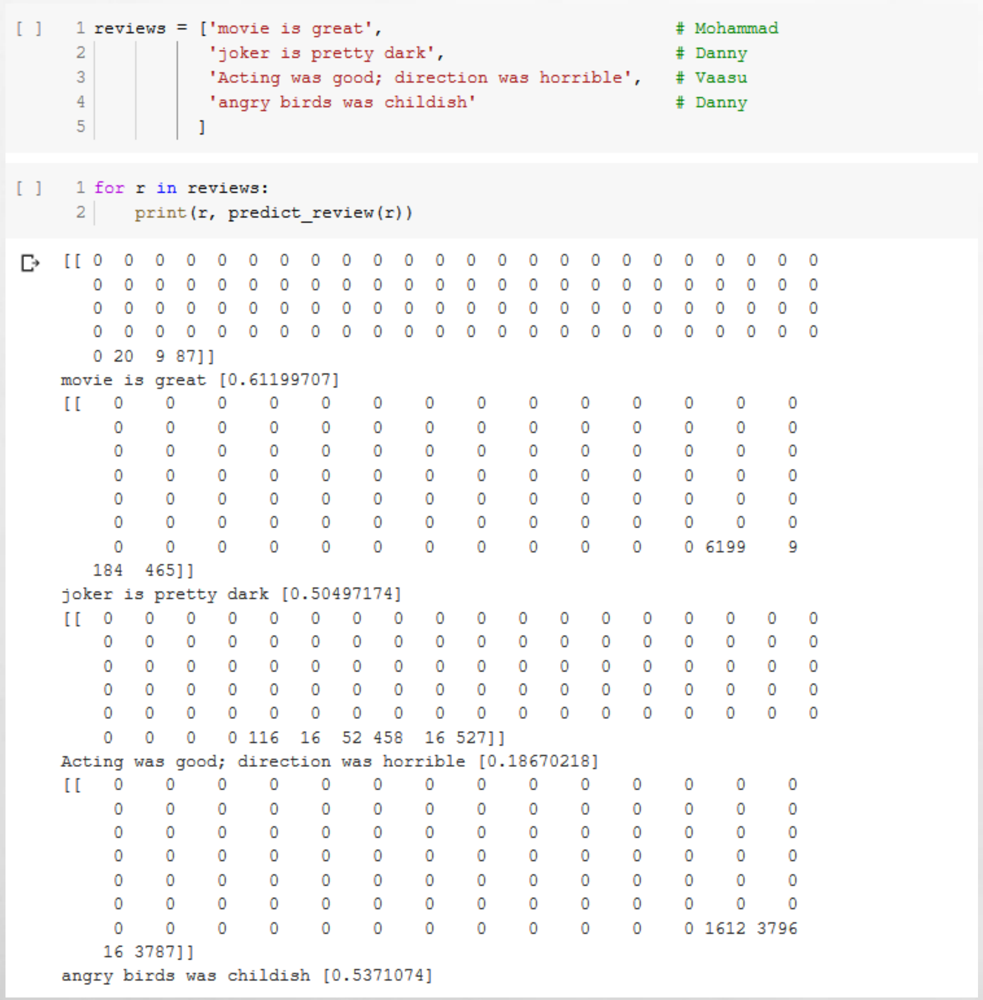
```
Train on 25000 samples, validate on 25000 samples
Epoch 1/2
25000/25000 [==============================] - 79s 3ms/step - loss: 0.3899 - acc: 0.8146 - val_loss: 0.3155 - val_acc: 0.8633
Epoch 2/2
25000/25000 [==============================] - 78s 3ms/step - loss: 0.1917 - acc: 0.9267 - val_loss: 0.3606 - val_acc: 0.8500
25000/25000 [==============================] - 8s 335us/step
Test score: 0.36055331511497496
Test accuracy: 0.850039994597435
```

Example - Colab - https://colab.research.google.com/drive/1YRdRViykuWm9dIk2NVDKovDx7PDEwrmF



```
word_to_id = imdb.get_word_index()
word_to_id = {k: (v + 3) for k, v in word_to_id.items()}
word_to_id.update([("<PAD>", 0), ("<START>", 1), ("<UNK>", 2), ("<UNUSED>", 3)])
id_to_word = {value: key for key, value in word_to_id.items()}

def predict_review(review):

    review_ids = []

    # Tokenize and get word index
    tokens = text_to_word_sequence(review)
    for t in tokens:
        id_ = word_to_id.get(t, word_to_id["<UNK>"])
        if id_ > max_features:
            review_ids.append(word_to_id["<UNUSED>"])
        else:
            review_ids.append(id_)

    # Pad with zeros
    padded_review_ids = sequence.pad_sequences([review_ids],
                                               value=word_to_id["<PAD>"],
                                               maxlen=maxlen)
    print(padded_review_ids)

    return model.predict(padded_review_ids)[0]
```

Example - Colab - https://colab.research.google.com/drive/1YRdRViykuWm9dIk2NVDKovDx7PDEwrmF

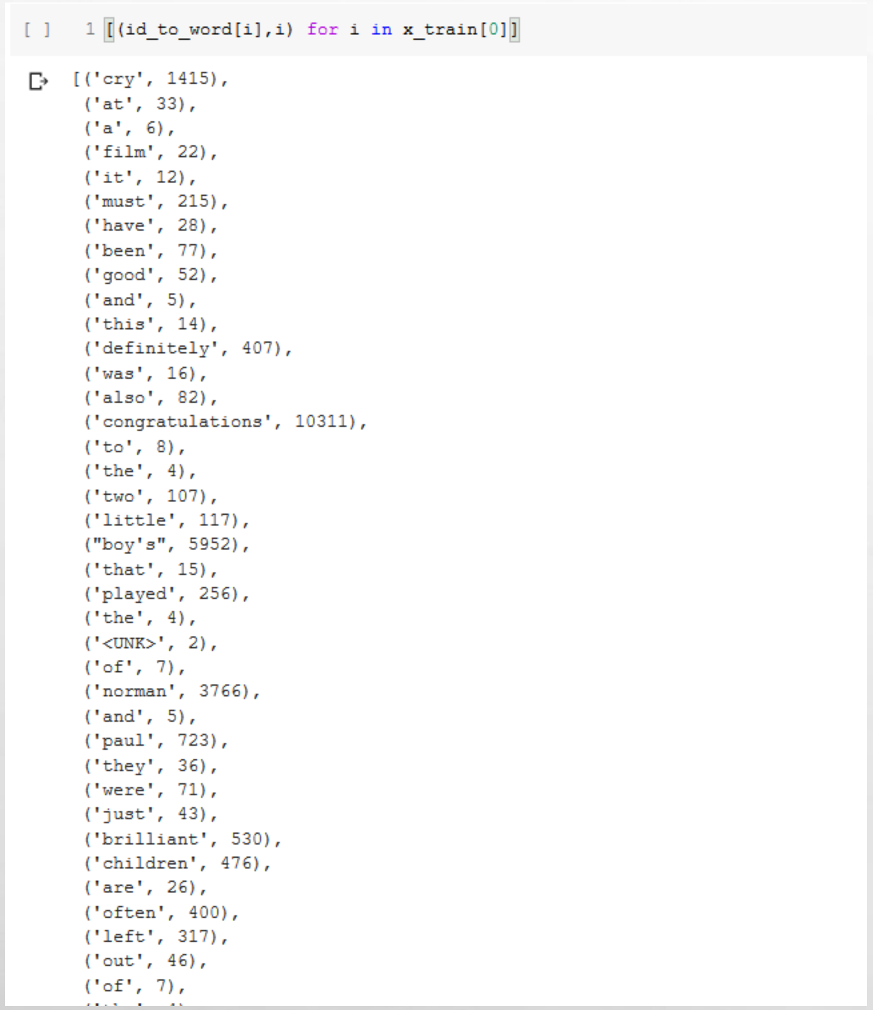


```
[ ] reviews = ['movie is great',                          # Mohammad
               'joker is pretty dark',                    # Danny
               'Acting was good; direction was horrible', # Vaasu
               'angry birds was childish'                 # Danny
              ]
```

```
[ ] for r in reviews:
        print(r, predict_review(r))
```

```
[[ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0 20  9 87]]
movie is great [0.61199707]
[[   0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0 6199    9
   184  465]]
joker is pretty dark [0.50497174]
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0 116  16  52 458  16 527]]
Acting was good; direction was horrible [0.18670218]
[[   0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0 1612 3796
    16 3787]]
angry birds was childish [0.5371074]
```

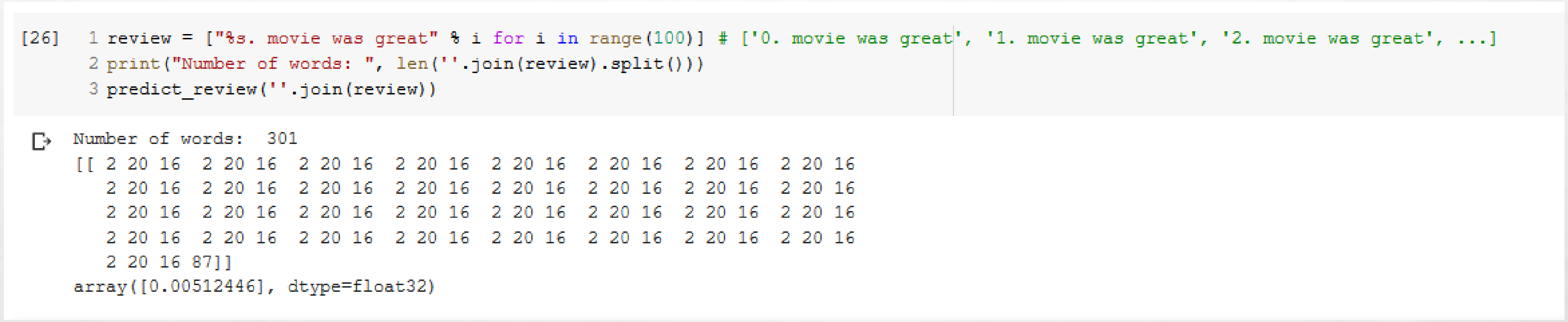
Example - Colab - https://colab.research.google.com/drive/1YRdRViykuWm9dIk2NVDKovDx7PDEwrmF



```
[(id_to_word[i],i) for i in x_train[0]]
```

```
[('cry', 1415),
 ('at', 33),
 ('a', 6),
 ('film', 22),
 ('it', 12),
 ('must', 215),
 ('have', 28),
 ('been', 77),
 ('good', 52),
 ('and', 5),
 ('this', 14),
 ('definitely', 407),
 ('was', 16),
 ('also', 82),
 ('congratulations', 10311),
 ('to', 8),
 ('the', 4),
 ('two', 107),
 ('little', 117),
 ("boy's", 5952),
 ('that', 15),
 ('played', 256),
 ('the', 4),
 ('<UNK>', 2),
 ('of', 7),
 ('norman', 3766),
 ('and', 5),
 ('paul', 723),
 ('they', 36),
 ('were', 71),
 ('just', 43),
 ('brilliant', 530),
 ('children', 476),
 ('are', 26),
 ('often', 400),
 ('left', 317),
 ('out', 46),
 ('of', 7),
```

Example - Colab - https://colab.research.google.com/drive/1YRdRViykuWm9dIk2NVDKovDx7PDEwrmF



```
[26]  review = ["%s. movie was great" % i for i in range(100)] # ['0. movie was great', '1. movie was great', '2. movie was great', ...]
      print("Number of words: ", len(''.join(review).split()))
      predict_review(''.join(review))
```

```
Number of words:  301
[[ 2 20 16  2 20 16  2 20 16  2 20 16  2 20 16  2 20 16  2 20 16  2 20 16
   2 20 16  2 20 16  2 20 16  2 20 16  2 20 16  2 20 16  2 20 16  2 20 16
   2 20 16  2 20 16  2 20 16  2 20 16  2 20 16  2 20 16  2 20 16  2 20 16
   2 20 16  2 20 16  2 20 16  2 20 16  2 20 16  2 20 16  2 20 16  2 20 16
   2 20 16 87]]
array([0.00512446], dtype=float32)
```

Question about predicting for a review with more than 100 words:

If you look at the predicted ids, it’s length is 100 which means it just neglects the rest of the words in the review. Here 2 is nothing but <UNK> which is unknown word since numbers like 0, 1, 2 … are not in the word_index

Where to get vectors for words ?

fastText

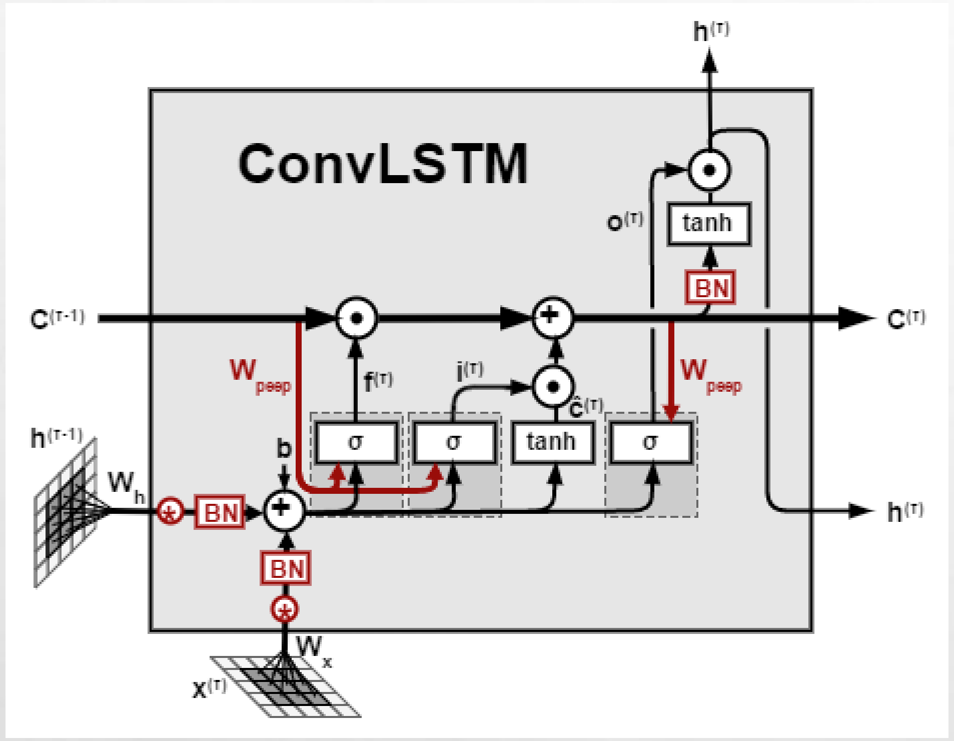
Library for efficient text classification and representation learning

Word vectors for 157 languages

English word vectors

Pre-trained on English webcrawl and Wikipedia

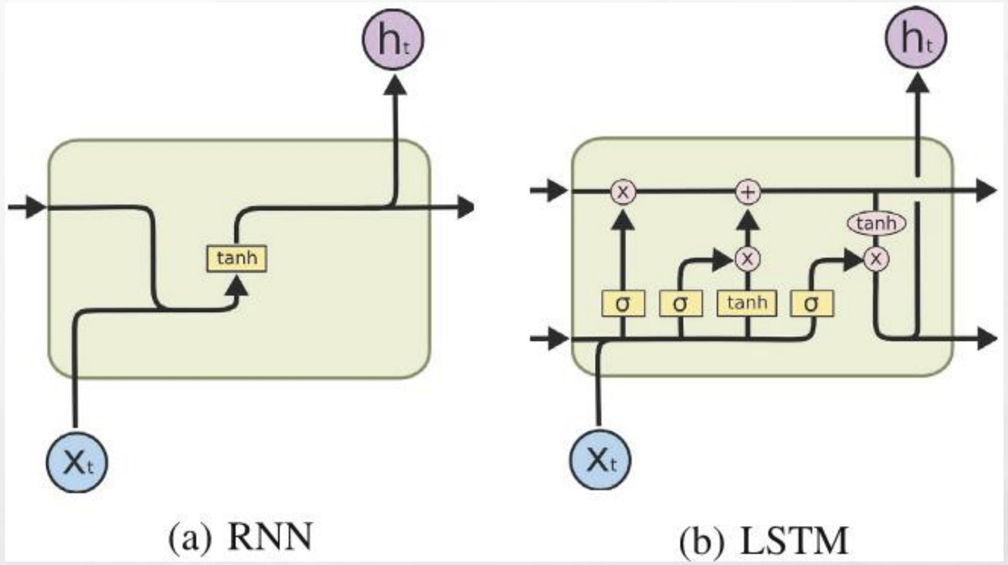
LSTM for Images ???



Key Takeaways

RNNs are used to process and predict sequences

LSTM's are great 😎



(a) RNN (b) LSTM

References

https://en.wikipedia.org/wiki/Long_short-term_memory
http://colah.github.io/posts/2015-08-Understanding-LSTMs/
https://www.youtube.com/watch?v=LHXXI4-IEns (Illustrated Guide to Recurrent Neural Networks: Understanding the Intuition)
https://www.youtube.com/watch?v=8HyCNIVRbSU (Illustrated Guide to LSTM's and GRU's: A step by step explanation)
https://medium.com/datathings/the-magic-of-lstm-neural-networks-6775e8b540cd

Visit this website to see some amazing deep learning works that can be experienced on the browser:
https://www.dlology.com/blog/top-10-deep-learning-experiences-run-on-your-browser/