2022

Deep Learning based Text Processing

Lec 11: Introduction to Long-Short Term Memory

hsyi@kisti.re.kr
Hongsuk Yi (이홍석)

Overview of Course (III)

- Introduction to Recurrent Neural Network
  - ✓ Simple RNN, BPTT, Memory Cell
  - ✓ Code: Implementing an RNN with Keras
- Introduction to Long-Short Term Memroy
  - ✓ Cell state, LSTM, and GRU, and Applications
  - ✓ A Visual Guide to Recurrent Layers in Keras
  - ✓ Code: A simple LSTM layers
- Text generation with RNN
  - ✓ Tokenizer, Character-Level Language model
  - ✓ Code: Alice's Adventures in Wonderland
- Sequence to Sequence model with RNN
  - ✓ Introduction to Seq2Seq and Attention model
  - ✓ Code: Character-Level Neural Machine Translation

Reviewing the last class:

LSTM

Recurrent Neural Networks: Process Sequences

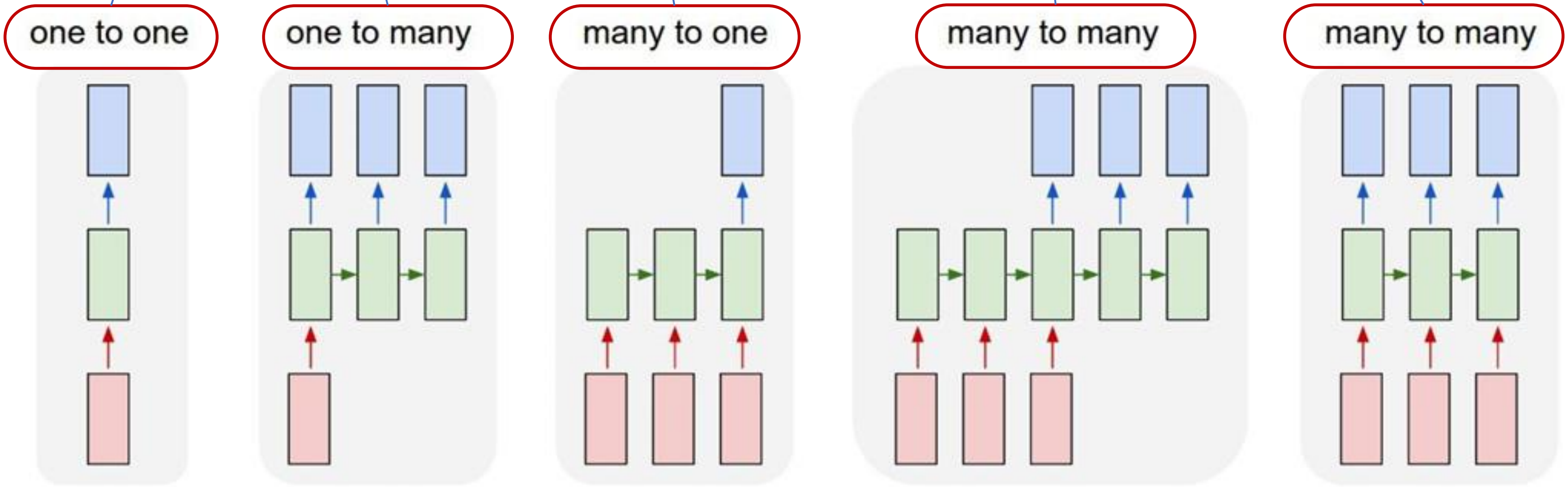
Vanilla
Neural Network

Image Captioning
(image→ seq of words)

Sentiment Classification
(Seq of words → sentiment)

Machine Translation
(Seq of words → seq of words)

Video classification
on frame level



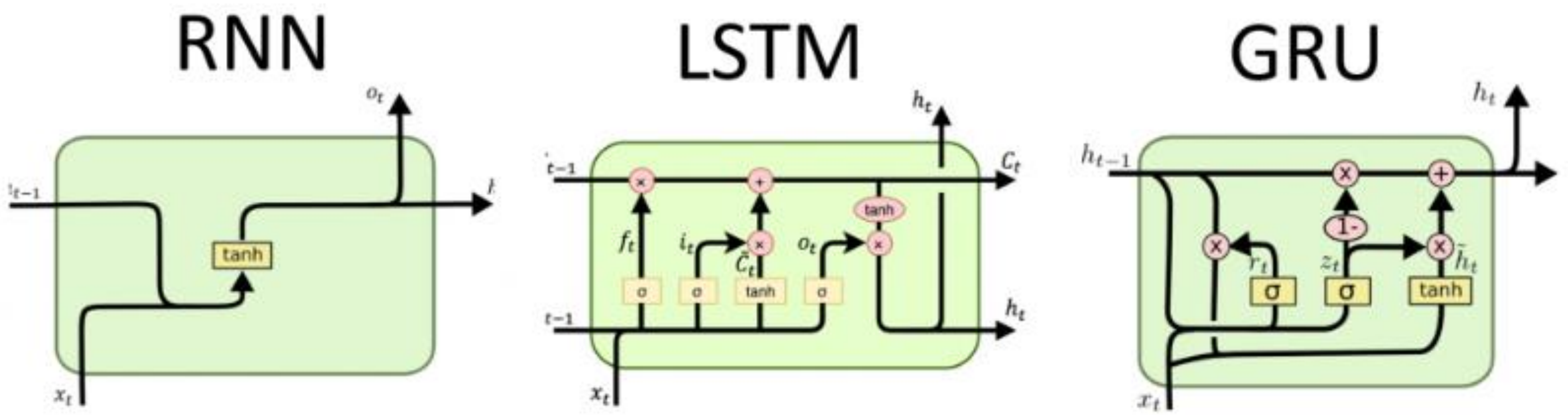
❖ Still popular and widely used today

✓ A recent, related approach is the Gated Recurrent Unit (GRU)

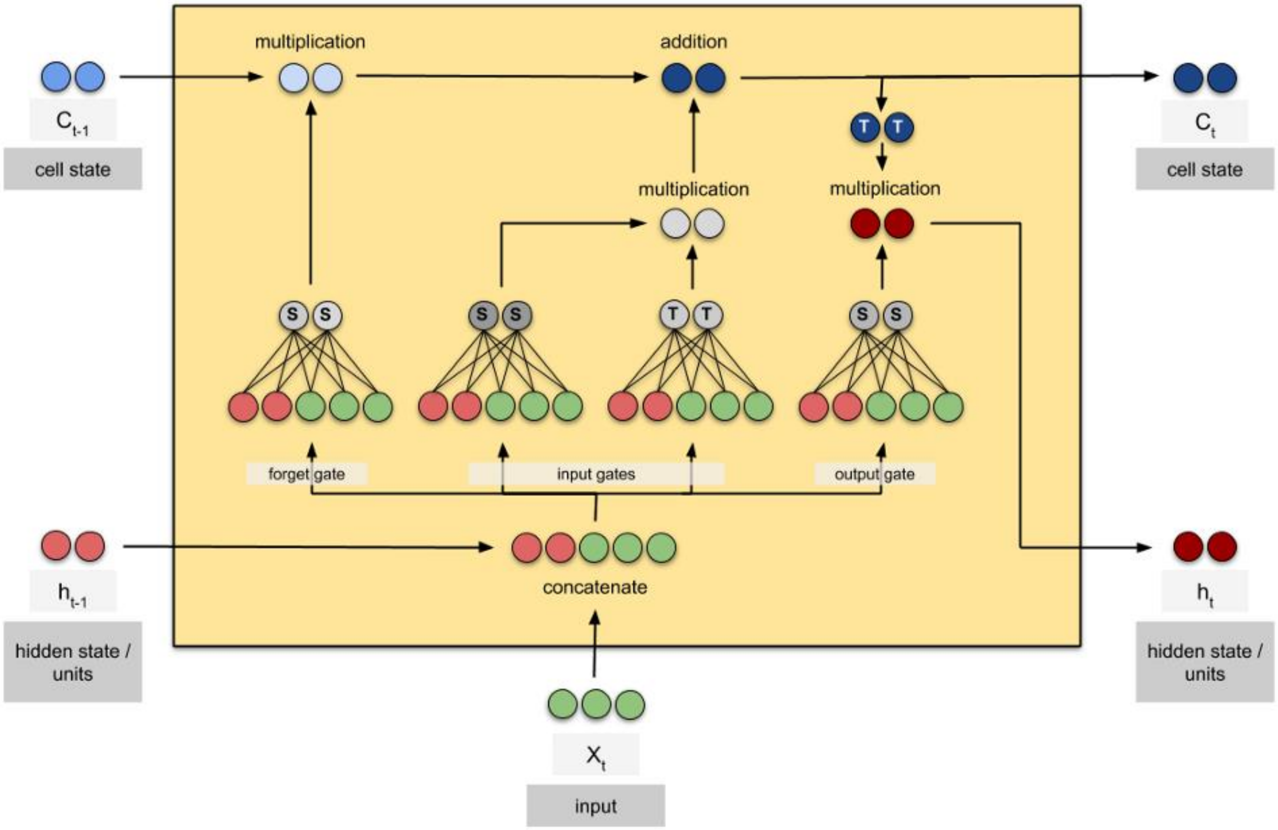
- Cho, Kyunghyun,et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014).

✓ Nice article exploring LSTMs and comparing them to GRUs

- Jozefowicz, et al. "An empirical exploration of recurrent network architectures." In International Conference on Machine Learning, pp. 2342-2350. 2015.



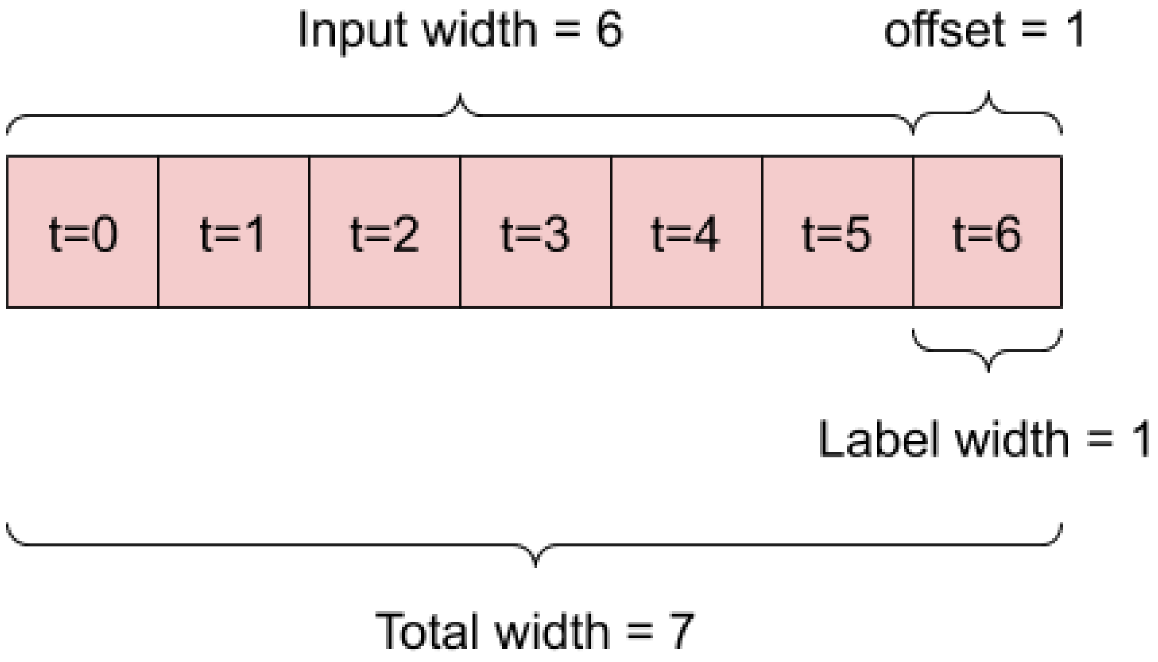
- Hidden State
- Cell State



Data windowing

✓ A model that makes a prediction one hour into the future, given six hours of history

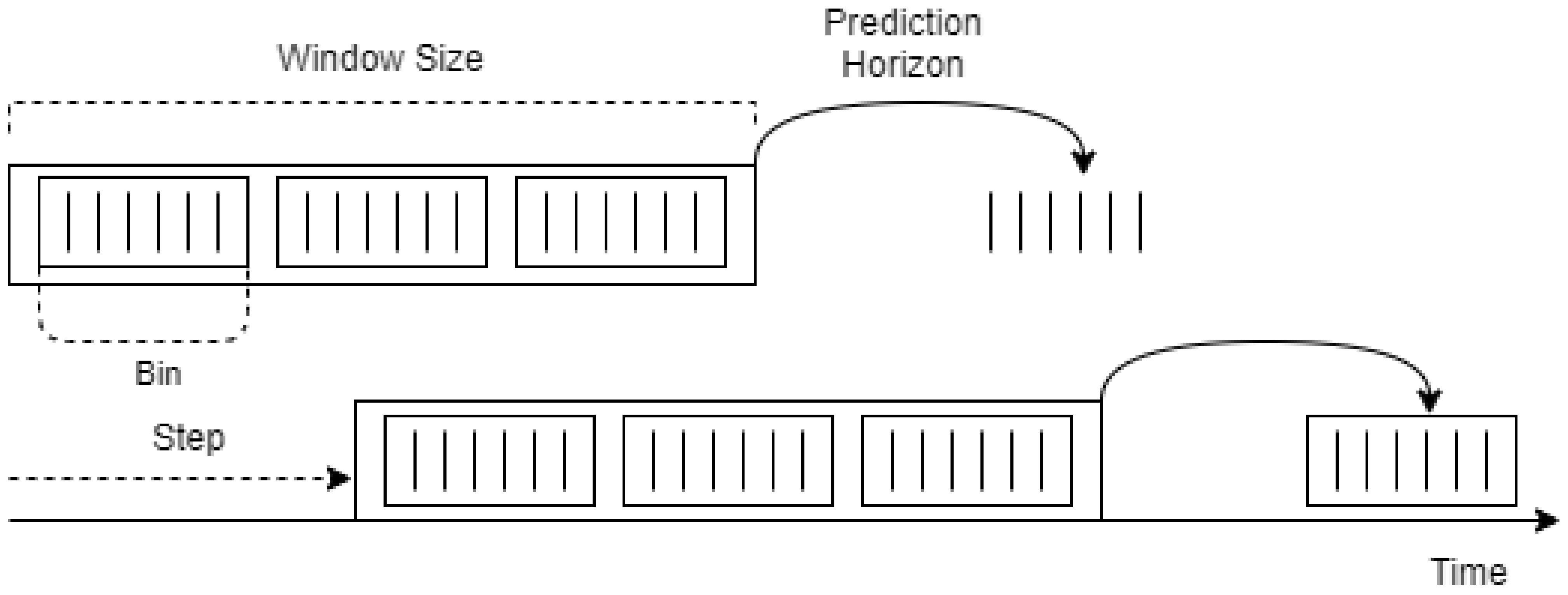
```
w2 = WindowGenerator(input_width=6, label_width=1, shift=1, label_columns=['sinefunction'])
```




```
Total window size: 7
Input indices: [0 1 2 3 4 5]
Label indices: [6]
Label column name(s): ['sinefunction']
```

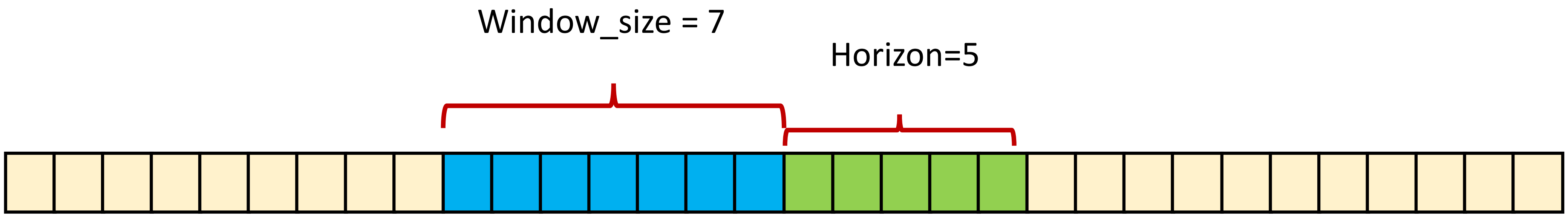
Modelling Experiments

- Two terms are really important in the type of forecasting model :
  - ✓ **Window Size** :The number of timesteps we take to predict into the future
  - ✓ **Horizon** : The number of timesteps ahead into the future we predict.

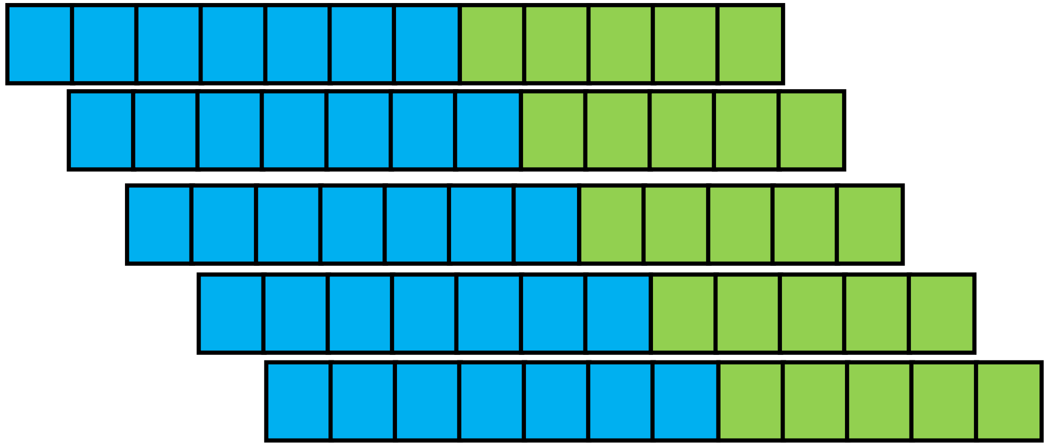


Many-to-One RNN Data Structure

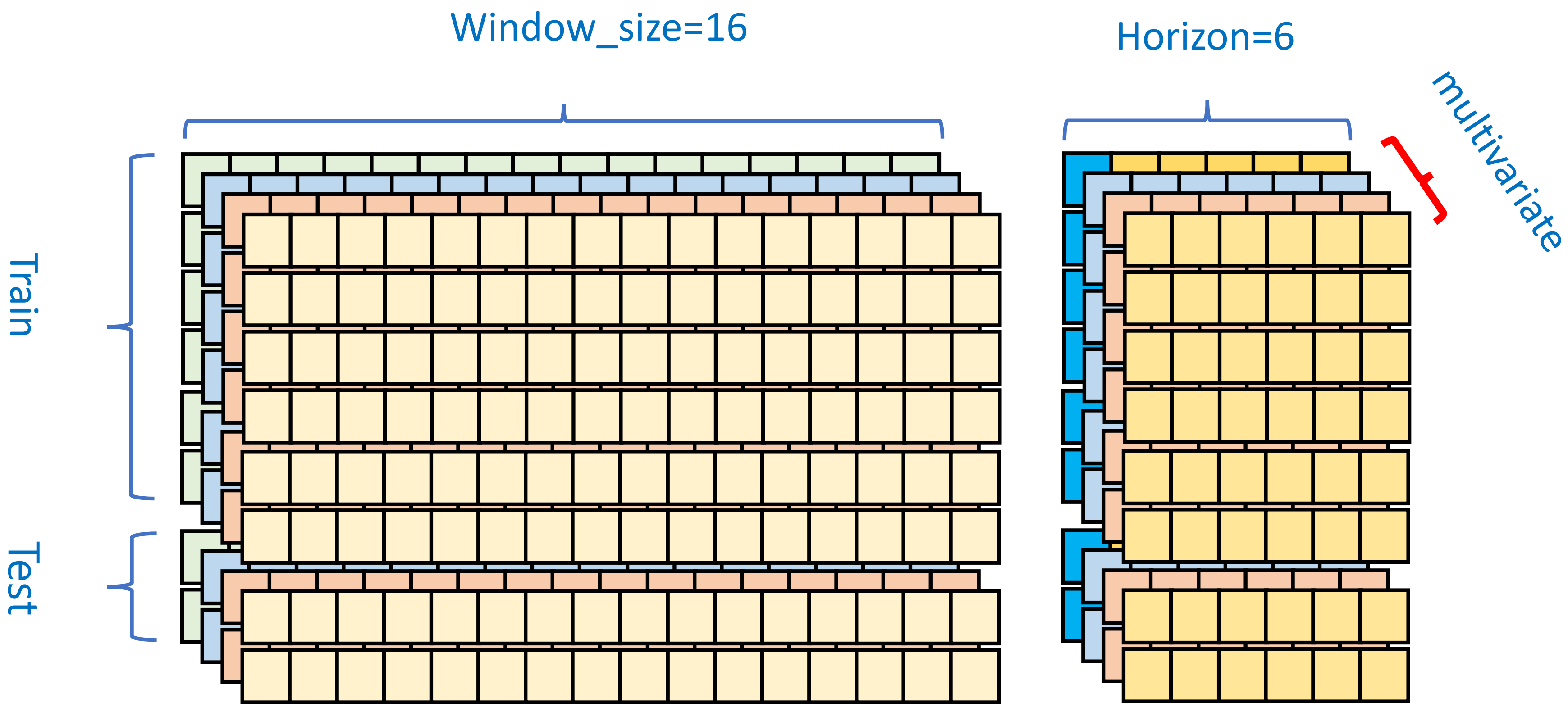
Step1: set the number of window_size, horizon



Step2: set the number of window_size, horizon



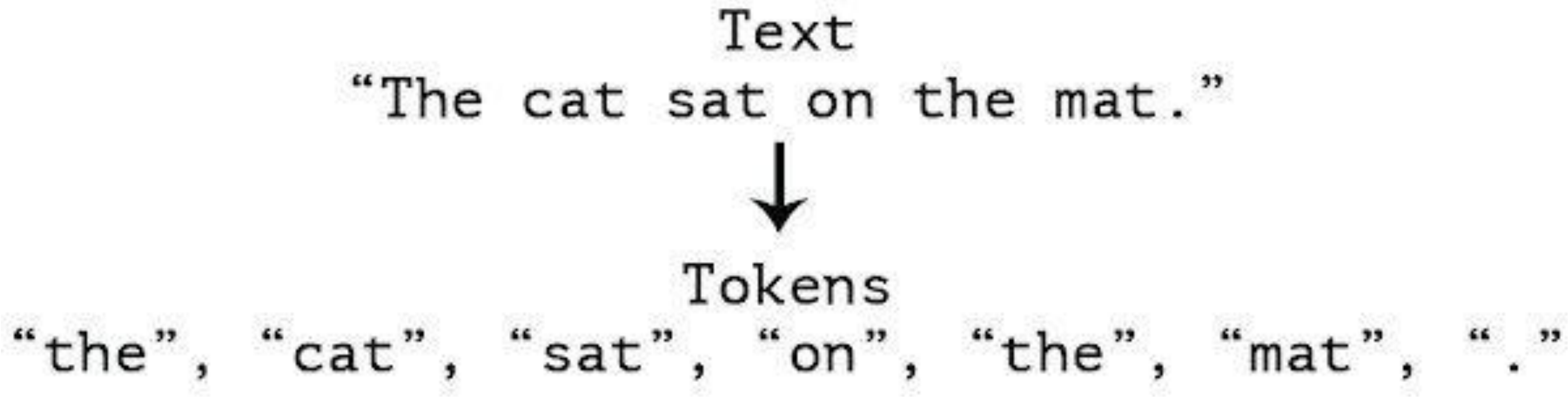
RNN Input-Output Data Structure



Text Tockenizer

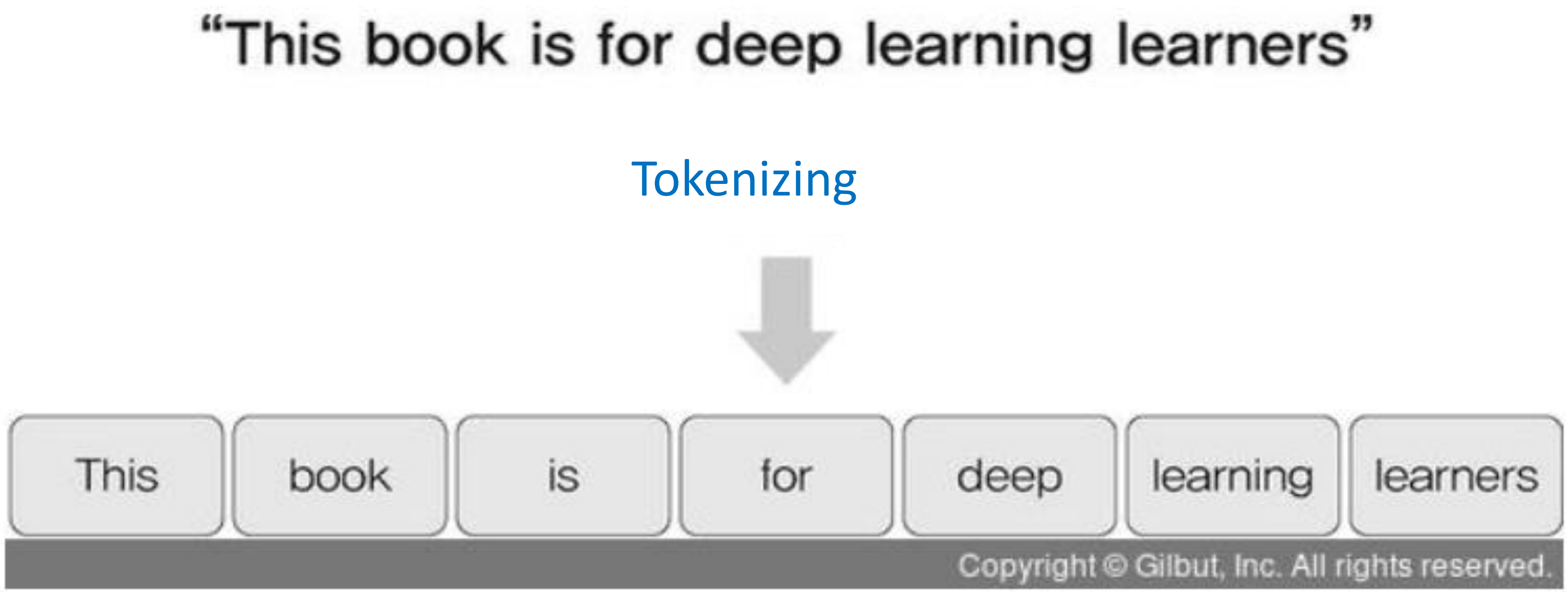
Understanding the principles

- Token : Language elements that we can’t share anymore
- Tokenizer
  - ✓ work to input text data into the neural network.
  - ✓ The preprocessing process that converts it into an appropriate form through encoding
- One-hot encoding
  - ✓ In the case of text data, an embedding layer is basically used.



KISTI SINCE 1962

- Word tokenization divides sentences based on spacing as follows.



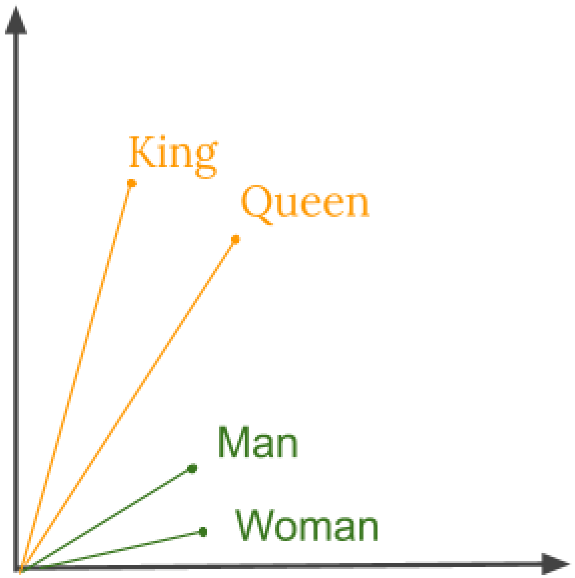
One-Hot Encoding

	1	0	0	0	…	0	0
Index:	0	1	2	3	…	99998	99999

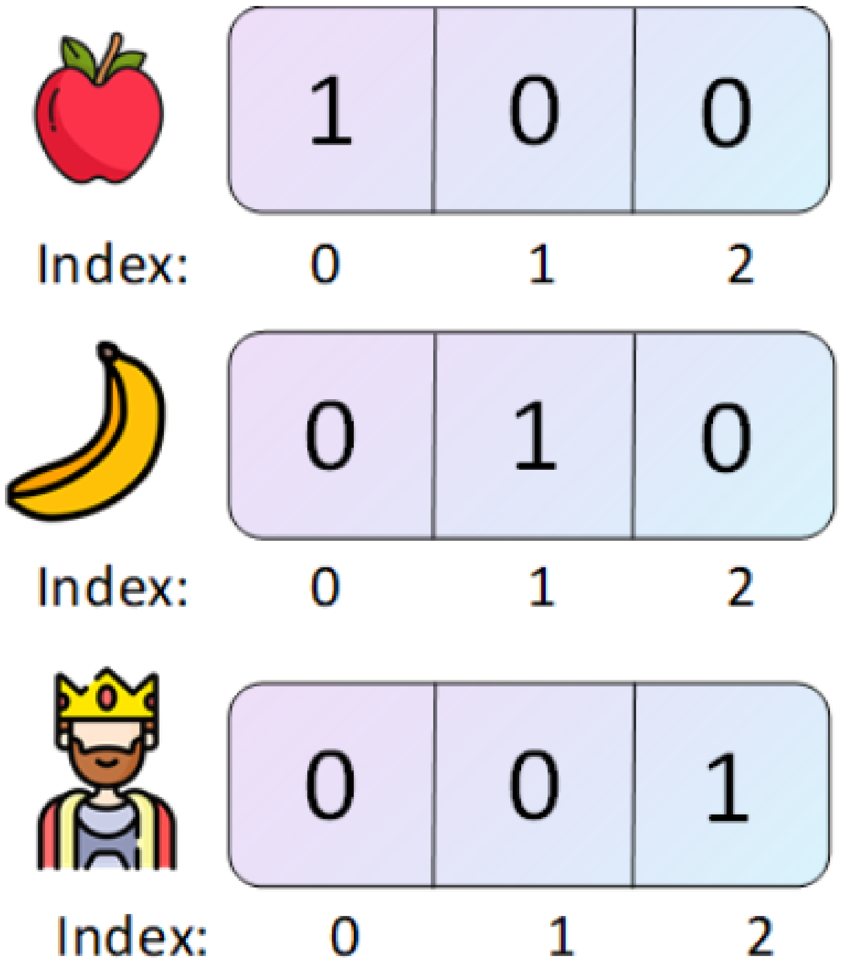
	0	1	0	0	…	0	0
Index:	0	1	2	3	…	99998	99999

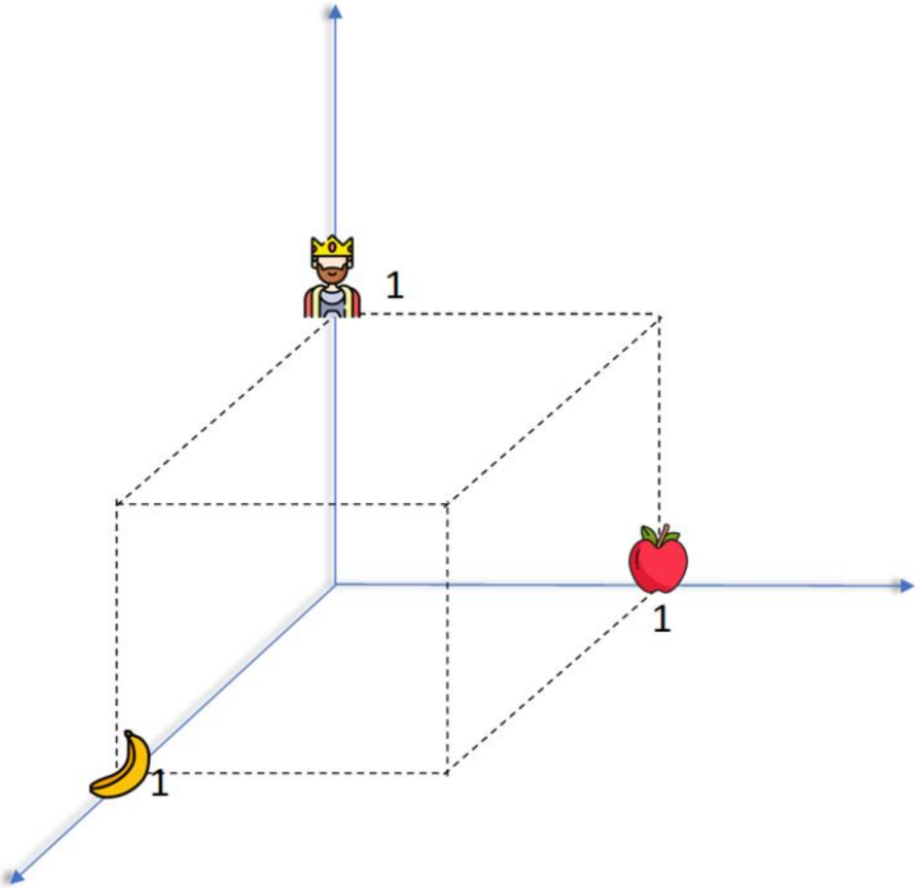
What is Word Embedding?

- Word embeddings are basically a form of word representation that bridges the human understanding of language to that of a machine.
  - ✓ They have learned representations of text in an n-dimensional space where words that have the same meaning have a similar representation.
  - ✓ Meaning that two similar words are represented by almost similar vectors that are very closely placed in a vector space.



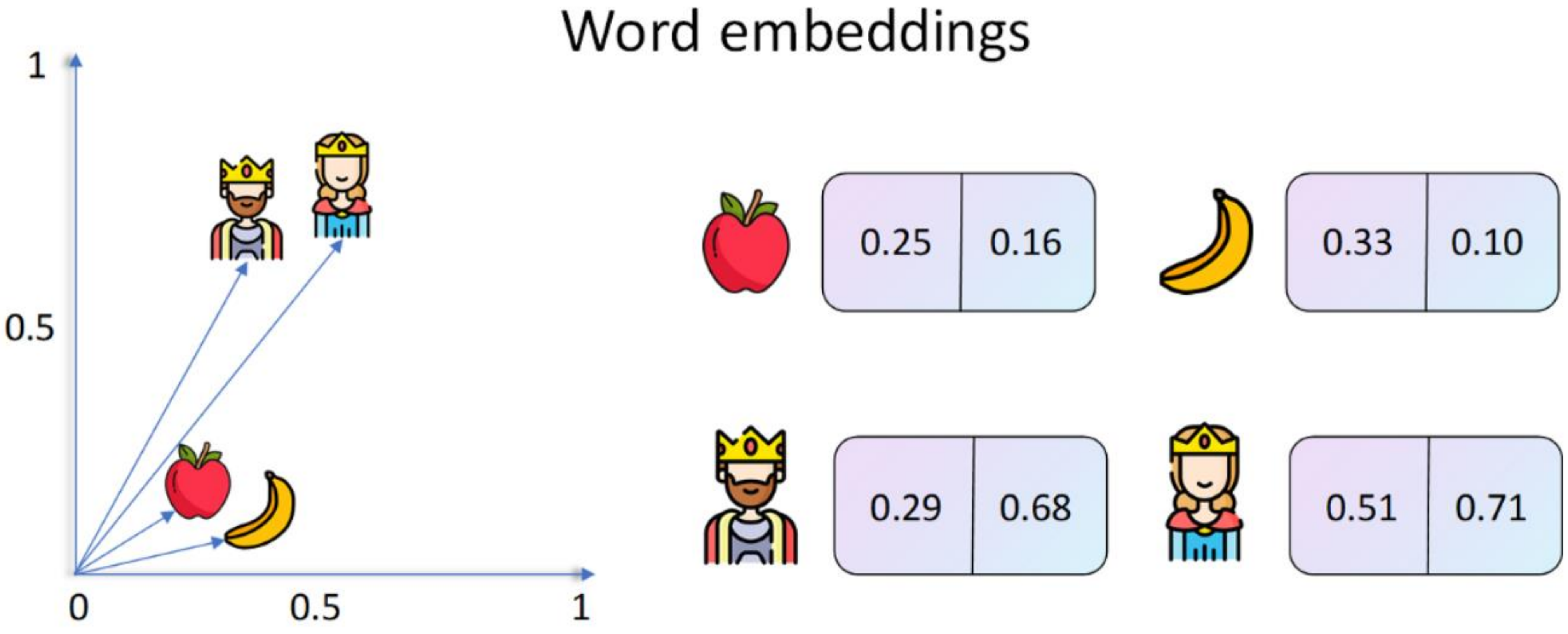
Word Embedding





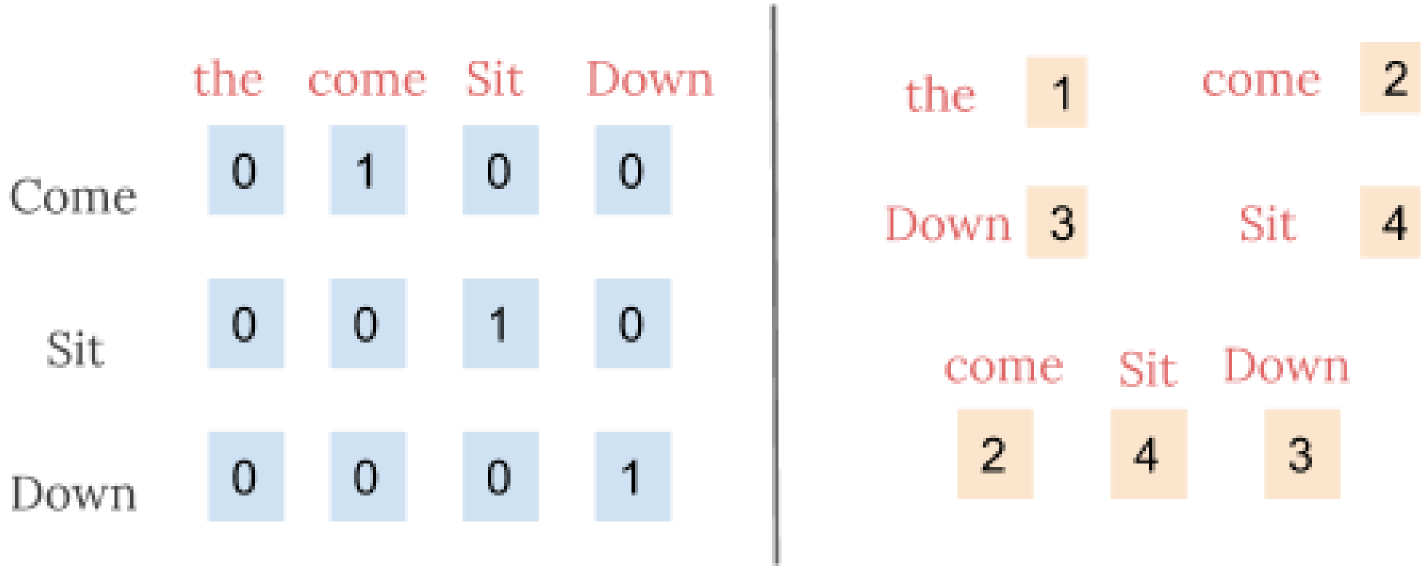
Word Embedding

❖ 2 dimensional word embedding representation of our example words



Why Word Embeddings are used?

- we can use two more techniques
  - ✓ one-hot encoding
  - ✓ we can use unique numbers to represent words in a vocabulary.
- we assume we have a small vocabulary containing just four words, using the two techniques we represent the sentence ‘Come sit down’.



A simple example: Word embedding

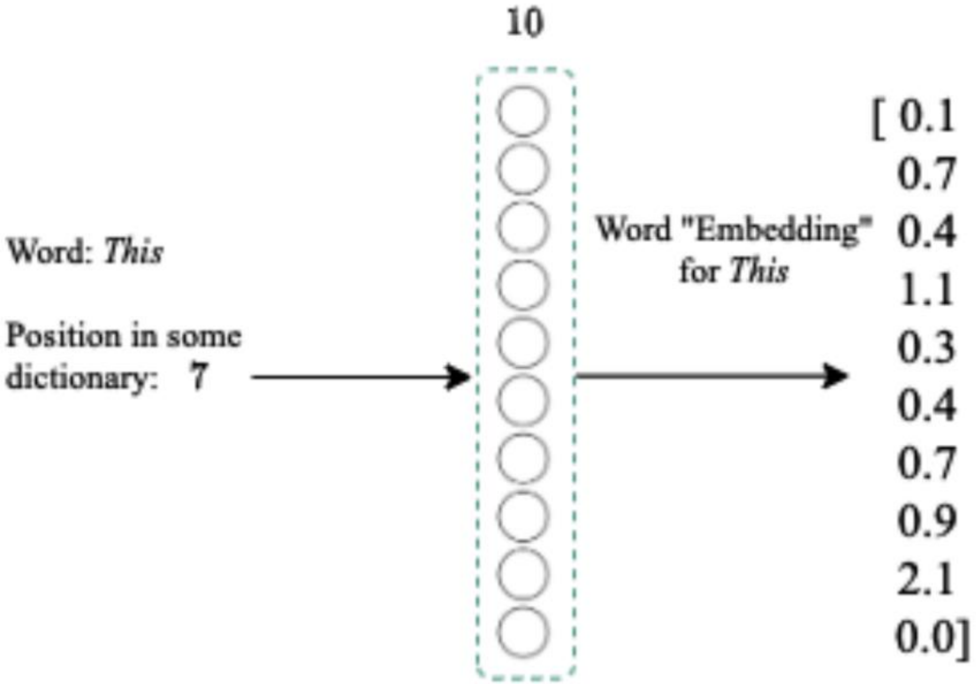
"This is a small vector"

This

index : 7,

vector size : 10

embeddng_dim = 10



Sequence of vectors

Usually the vocabulary size is thousands : **vocab_size = 30**

Usually, sentences consist of more than five words. **seq_length = 5**

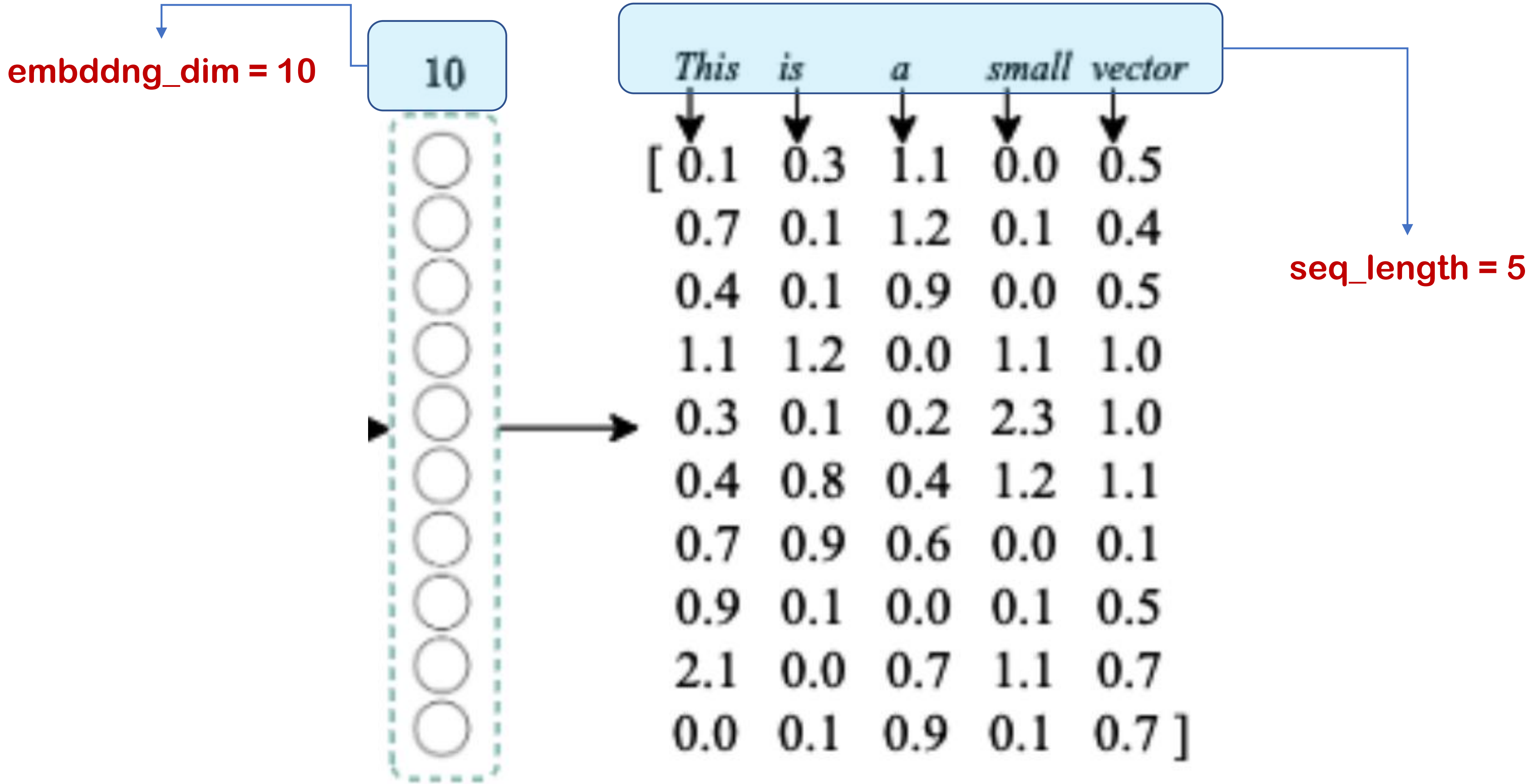
"This is a small vector"

[7 0 6 1 28]

The embedding dimension is about 50 dimensions for small models.

There are also 500 to 1,000 dimensions of a good model.

Sequence of word embeddings



Let’s Code!
A simple example of Tokenizer

- The example below shows how to encode two sentences
  - ✓ 'You are the Best', and 'You are the Nice' based on words using TensorFlow.

1. Word-based encoding

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.utils import to_categorical
```

```
sentences = [
  'You are the Best',
  'You are the Nice'
]
```

KISTI SINCE 1962

❖ fit_on_texts() 메서드는 문자 데이터를 입력받아서 리스트의 형태로 변환합니다.

```
tokenizer = Tokenizer(num_words = 100, oov_token="<OOV>")
tokenizer.fit_on_texts(sentences)
word_index = tokenizer.word_index
```

Words that have not been indexed in advance are indexed as "OOV"

```
print(word_index)
print('----------------------------------------')
total_words = len(tokenizer.word_index) + 1
print('total_words=',total_words)
```

```
{'<OOV>': 1, 'you': 2, 'are': 3, 'the': 4, 'best': 5, 'nice': 6}
----------------------------------------
total_words= 7
```

The word_index attribute of the tokenizer returns a dictionary containing a pair of key-values of words and numbers.

The output results show that the upper case 'I' has been converted to the lower case 'i'.

2) Converting text into a sequence

텍스트를 시퀀스로 변환하기

```
sequences = tokenizer.texts_to_sequences(sentences)

print(word_index)
print(sequences)
```

```
{'<OOV>': 1, 'you': 2, 'are': 3, 'the': 4, 'best': 5, 'nice': 6}
[[2, 3, 4, 5], [2, 3, 4, 6]]
```

3. Setting up padding

- You have to padding to make the sentence the same length.
  - ✓ Padding uses the pad_sequences function.

```
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

Sequences are text sentences converted into sequences of integers

- Since the longest sequence is 7, it has all been converted into sequences of the same length

```
padded = pad_sequences(sequences)

print(word_index)
print(sequences)
print(padded)
```

```
{'<OOV>': 1, 'you': 2, 'are': 3, 'the': 4, 'best': 5, 'nice': 6}
[[2, 3, 4, 5], [2, 3, 4, 6]]
[[2 3 4 5]
 [2 3 4 6]]
```

3. Setting up padding

❖ padding parameter : 'pre', 'post’

✓ If the padding parameter is specified as 'post', padding is filled after the sequence. The default is "pre"

```
padded = pad_sequences(sequences, padding='post')
print(padded)
```

```
[[2 3 4 5]
 [2 3 4 6]]
```

4) Encoding in binary form

```
# 이진 형태로 인코딩합니다.
binary_results = tokenizer.sequences_to_matrix(sequences, mode = 'binary')
print(f'binary_vectors:₩n {binary_results}₩n')
```

```
binary_vectors:
 [[0. 0. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0.]
 [0. 0. 1. 1. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0.]]
```

4) Encoding in binary form

```
print(f'One-Hot Encodeing:',to_categorical(sequences))
```

```
One-Hot Encodeing: [[[0. 0. 1. 0. 0. 0. 0.]
  [0. 0. 0. 1. 0. 0. 0.]
  [0. 0. 0. 0. 1. 0. 0.]
  [0. 0. 0. 0. 0. 1. 0.]]

 [[0. 0. 1. 0. 0. 0. 0.]
  [0. 0. 0. 1. 0. 0. 0.]
  [0. 0. 0. 0. 1. 0. 0.]
  [0. 0. 0. 0. 0. 0. 1.]]]
```

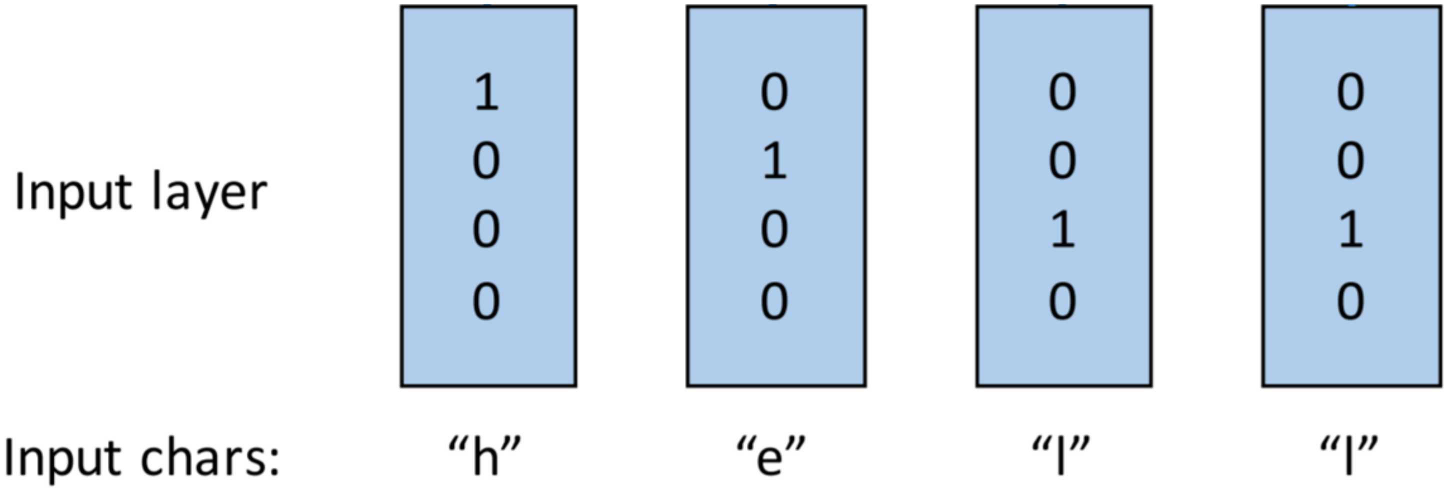
```
test_text = ['You are the One']
test_seq = tokenizer.texts_to_sequences(test_text)

print(f'test sequences: {test_seq}')
```

```
test sequences: [[2, 3, 4, 1]]
```

Character-level language model

Example training sequence : “hello”

- “The Unreasonable Effectiveness of Recurrent Neural Networks,”
  - ✓ Andrej Karpathy In 2015
- Character-level language model
  - ✓ Suppose we only had a vocabulary of four letters “hell”: [h,e,l,o]
  - ✓ We will encode each character into a vector using 1-of-k encoding

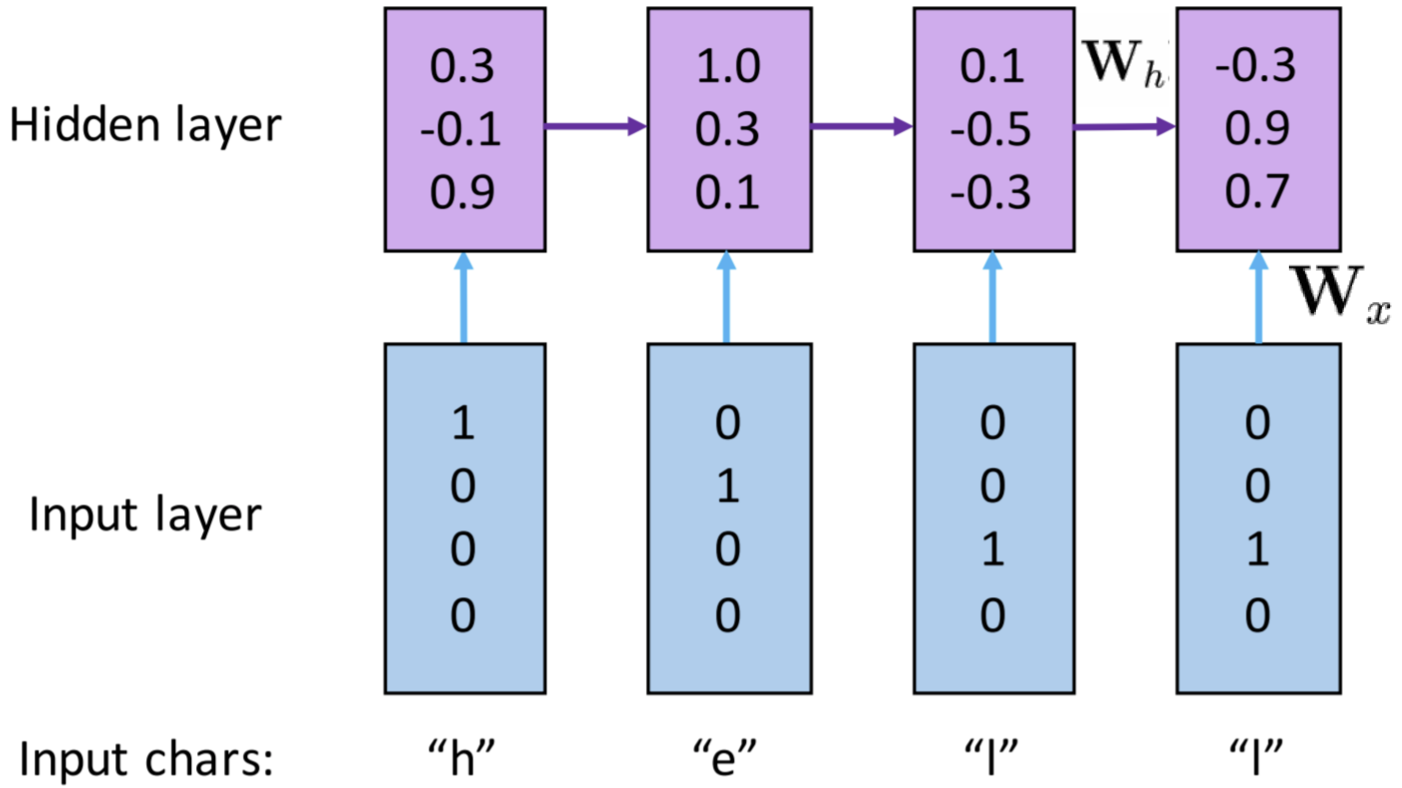


$$\mathbf{h}_t = \tanh(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \mathbf{x}_t)$$

Example: Character-level Language Model

Vocabulary: [h,e,l,o]

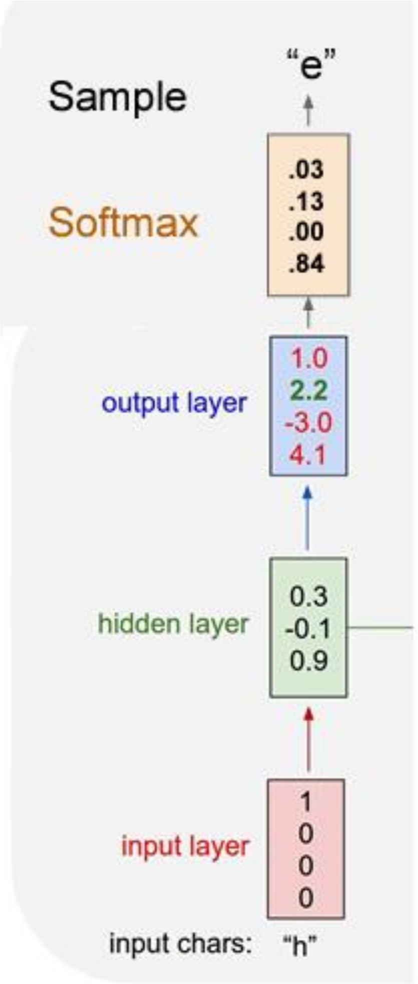
Example training sequence: **"hello"**

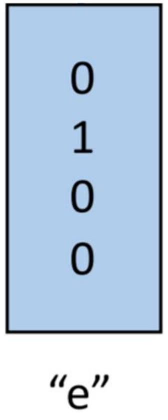


Example:
Character-level
Language Model
Sampling

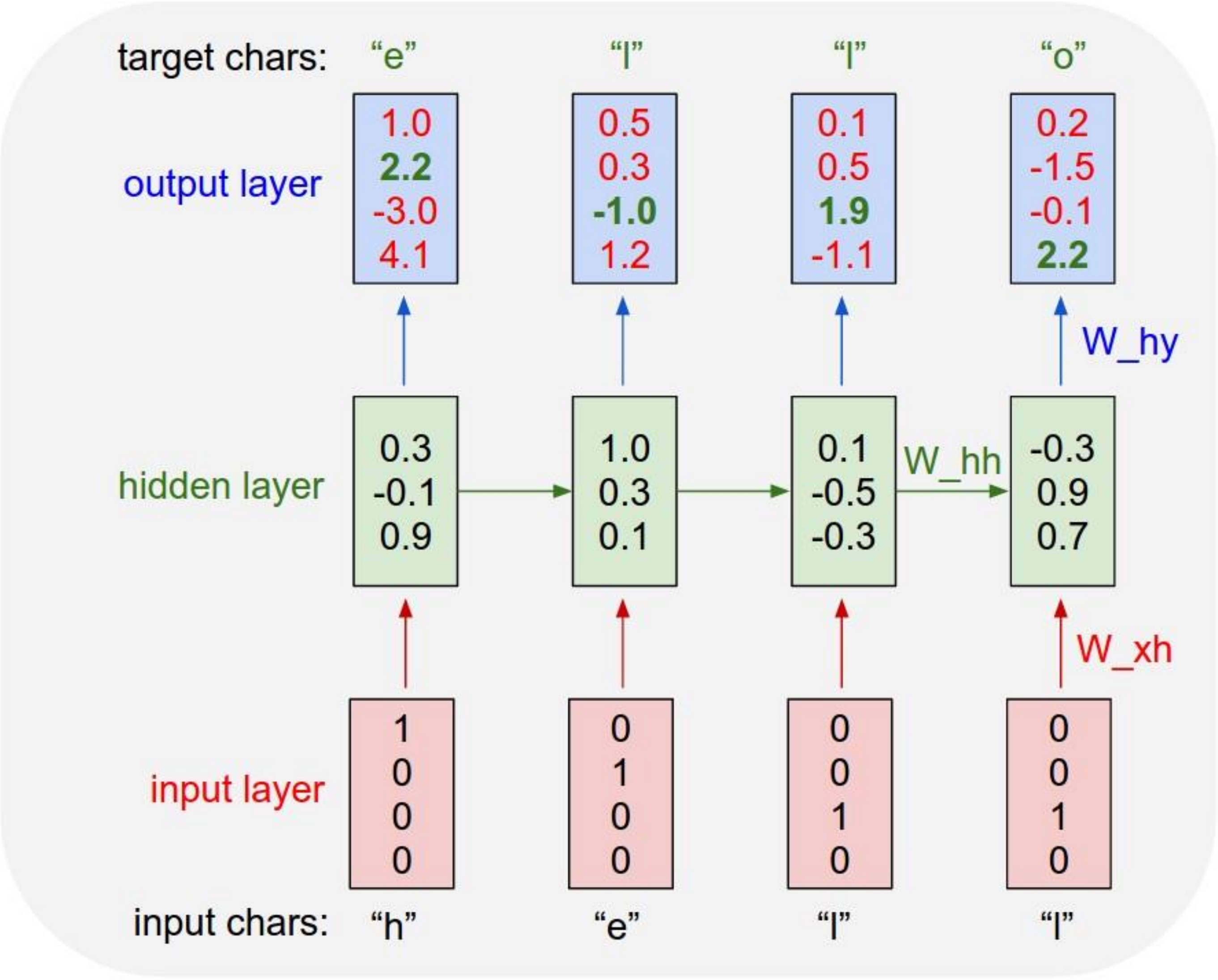
Vocabulary:
[h,e,l,o]

At test-time sample characters one at a time, feed back to model



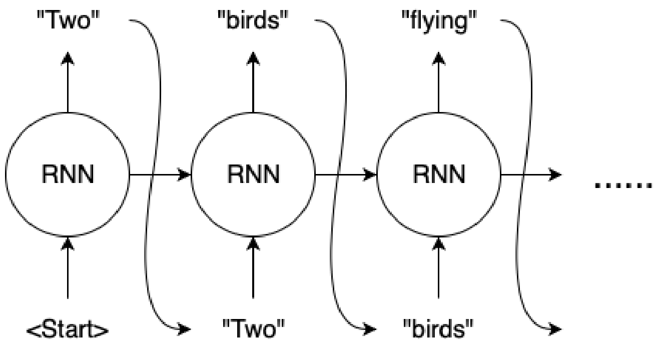


Char-RNN

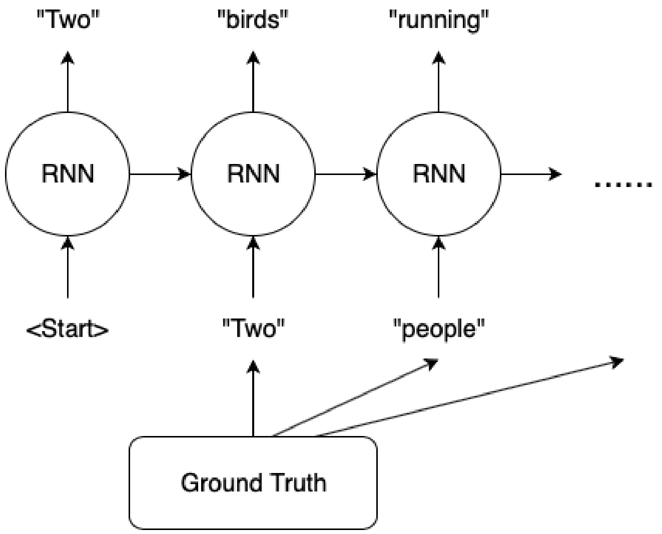


We want the green numbers to be high and red numbers to be low.

With Teacher Forcing Learing

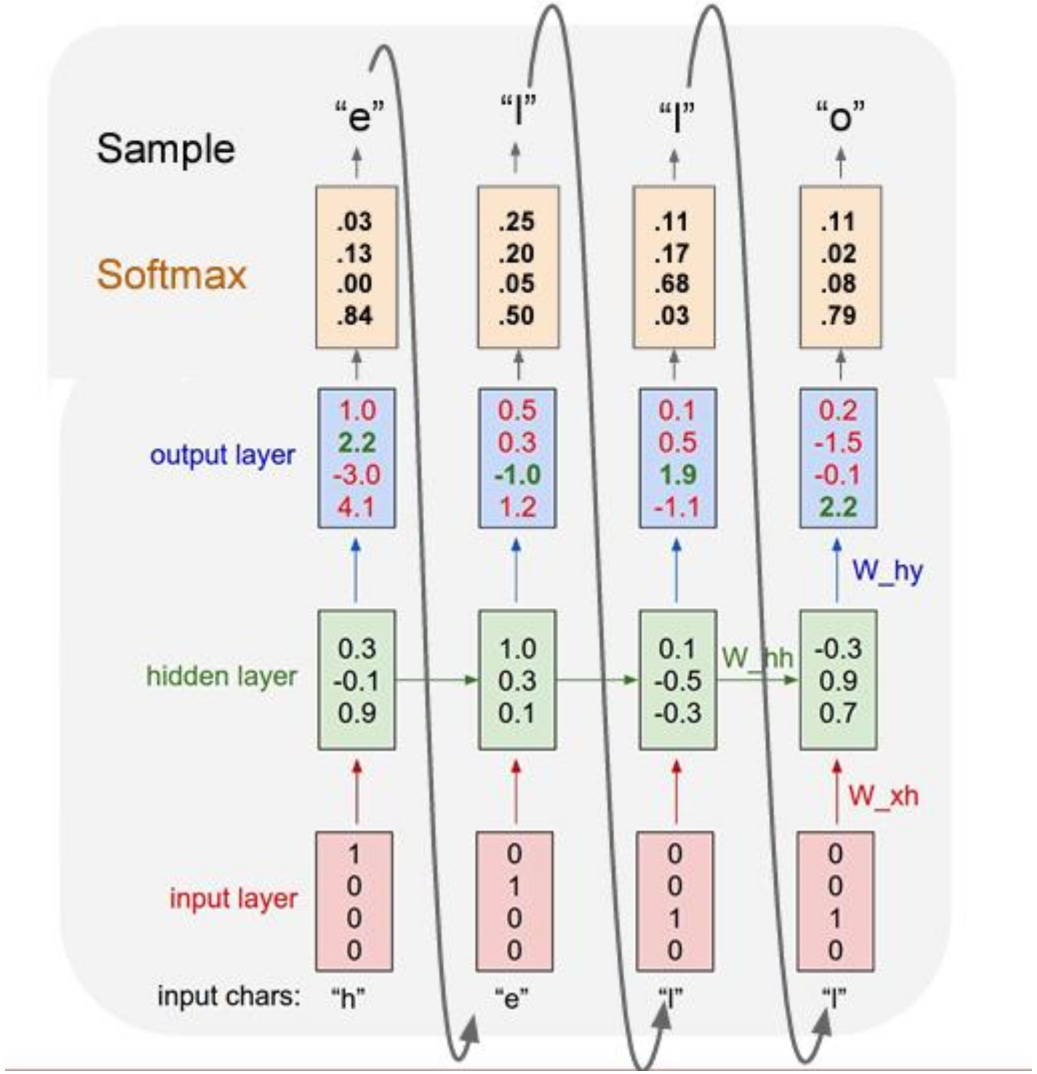


Without Teacher Forcing



With Teacher Forcing

Teacher forcing : Character-level Language Model



[HW] Let's Code! :

Alice's Adventures in Wonderland

❖ Data download : Alice’s Adventures in Wonderland

✓ Classical novels are not protected by copyright

In [1]:
```
import numpy as np
import urllib.request
from tensorflow.keras.utils import to_categorical
```

In [2]:
```
urllib.request.urlretrieve("http://www.gutenberg.org/files/11/11-0.txt", filename="11-0.txt")
f = open('11-0.txt', 'rb')
sentences = []
for sentence in f:                    # 데이터를 한 줄씩 읽는다.
    sentence = sentence.strip() # strip()을 통해 ₩r, ₩n을 제거한다.
    sentence = sentence.lower() # 소문자화.
    sentence = sentence.decode('ascii', 'ignore') # ₩xe2₩x80₩x99 등과 같은 바이트 열 제거
    if len(sentence) > 0:
        sentences.append(sentence)
f.close()
```

Char RNNLM

The elements in the list are composed of strings, but they are not tokenized. put it together as a string

In [3]:
```
sentences[:5]
```

Out[3]:
```
['the project gutenberg ebook of alices adventures in wonderland, by lewis carroll',
 'this ebook is for the use of anyone anywhere in the united states and',
 'most other parts of the world at no cost and with almost no restrictions',
 'whatsoever. you may copy it, give it away or re-use it under the terms',
 'of the project gutenberg license included with this ebook or online at']
```

In [4]:
```
total_data = ' '.join(sentences)
print('문자열의 길이 또는 총 글자의 개수: %d' % len(total_data))
```

```
문자열의 길이 또는 총 글자의 개수: 159484
```

In [5]:
```
print(total_data[:20])
```

```
the project gutenber
```

In [6]:
```
char_vocab = sorted(list(set(total_data)))
vocab_size = len(char_vocab)
print ('글자 집합의 크기 : {}'.format(vocab_size))
```

```
글자 집합의 크기 : 56
```

In [7]:
```
char_to_index = dict((char, index) for index, char in enumerate(char_vocab)) # 글자에 고유한 정수 인덱스 부여
print(char_to_index)
```

```
{' ': 0, '!': 1, '"': 2, '#': 3, '$': 4, '%': 5, "'": 6, '(': 7, ')': 8, '*': 9, ',': 10, '-': 11, '.': 12, '/': 13, '0': 14, '1': 15, '2': 16, '3': 17, '4': 18, '5': 19, '6': 20,
28, '_': 29, 'a': 30, 'b': 31, 'c': 32, 'd': 33, 'e': 34, 'f': 35, 'g': 36, 'h': 37, 'i': 38, 'j': 39, 'k': 40, 'l': 41, 'm': 42, 'n': 43, 'o': 44, 'p': 45, 'q': 46, 'r': 4
'z': 55}
```

In [8]:
```
index_to_char={}
for key, value in char_to_index.items():
    index_to_char[value] = key
```

In [9]:
```
seq_length = 60 # 문장의 길이를 60으로 한다.
n_samples = int(np.floor((len(total_data) - 1) / seq_length)) # 문자열을 60등분한다. 그러면 즉, 총 샘플의 개수
print ('문장 샘플의 수 : {}'.format(n_samples))
```

```
문장 샘플의 수 : 2658
```

KISTI SINCE 1962

```
train_X = []
train_y = []

for i in range(n_samples):
    # 0:60 -> 60:120 -> 120:180로 loop를 돌면서 문장 샘플을 1개씩 픽한다.
    X_sample = total_data[i * seq_length: (i + 1) * seq_length]
    # 정수 인코딩
    X_encoded = [char_to_index[c] for c in X_sample]
    train_X.append(X_encoded)
    # 오른쪽으로 1칸 쉬프트
    y_sample = total_data[i * seq_length + 1: (i + 1) * seq_length + 1]
    y_encoded = [char_to_index[c] for c in y_sample]
    train_y.append(y_encoded)
```

```
print('X 데이터의 첫번째 샘플 :',train_X[0])
print('y 데이터의 첫번째 샘플 :',train_y[0])
print('-'*50)
print('X 데이터의 첫번째 샘플 디코딩 :',[index_to_char[i] for i in train_X[0]])
print('y 데이터의 첫번째 샘플 디코딩 :',[index_to_char[i] for i in train_y[0]])
```

```
X 데이터의 첫번째 샘플 : [49, 37, 34, 0, 45, 47, 44, 39, 34, 32, 49, 0, 36, 50, 49, 34, 43, 31, 34, 47]
y 데이터의 첫번째 샘플 : [37, 34, 0, 45, 47, 44, 39, 34, 32, 49, 0, 36, 50, 49, 34, 43, 31, 34, 47, 36]
--------------------------------------------------
X 데이터의 첫번째 샘플 디코딩 : ['t', 'h', 'e', ' ', 'p', 'r', 'o', 'j', 'e', 'c', 't', ' ', 'g', 'u', 't', 'e', 'n', 'b', 'e', 'r']
y 데이터의 첫번째 샘플 디코딩 : ['h', 'e', ' ', 'p', 'r', 'o', 'j', 'e', 'c', 't', ' ', 'g', 'u', 't', 'e', 'n', 'b', 'e', 'r', 'g']
```

```
train_X = to_categorical(train_X)
train_y = to_categorical(train_y)
```

```
print('train_X의 크기(shape) : {}'.format(train_X.shape)) # 원-핫 인코딩
print('train_y의 크기(shape) : {}'.format(train_y.shape)) # 원-핫 인코딩
```

```
train_X의 크기(shape) : (7974, 20, 56)
train_y의 크기(shape) : (7974, 20, 56)
```

Char RNNLM

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, TimeDistributed

hidden_units = 256

model = Sequential()
model.add(LSTM(hidden_units, input_shape=(None, train_X.shape[2]), return_sequences=True))
model.add(LSTM(hidden_units, return_sequences=True))
model.add(TimeDistributed(Dense(vocab_size, activation='softmax')))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, None, 256)         320512

 lstm_1 (LSTM)               (None, None, 256)         525312

 time_distributed (TimeDistr  (None, None, 56)         14392
```

Char RNNLM

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(train_X, train_y, epochs=80, verbose=2)
```

```
def sentence_generation(model, length):
    # 글자에 대한 랜덤 인덱스 생성
    ix = [np.random.randint(vocab_size)]
    # 랜덤 익덱스로부터 글자 생성
    y_char = [index_to_char[ix[-1]]]
    print(ix[-1],'번 글자',y_char[-1],'로 예측을 시작!')
    # (1, length, 55) 크기의 X 생성. 즉, LSTM의 입력 시퀀스 생성
    X = np.zeros((1, length, vocab_size))
    for i in range(length):
        # X[0][i][예측한 글자의 인덱스] = 1, 즉, 예측 글자를 다음 입력 시퀀스에 추가
        X[0][i][ix[-1]] = 1
        print(index_to_char[ix[-1]], end="")
        ix = np.argmax(model.predict(X[:, :i+1, :])[0], 1)
        y_char.append(index_to_char[ix[-1]])
    return ('').join(y_char)
```

Char RNNLM

```
result = sentence_generation(model, 100)
print(result)
```

```
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 31ms/step
```

```
result = sentence_generation(model, 100)
print(result)
```

```
8 번 글자 ) 로 예측을 시작!
1/1 [==============================] - 1s 837ms/step
```

```
50ms/step
32ms/step
50ms/step
50ms/step
28ms/step
39ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
) is accessed, displaying, were took the labses, and began singing it, sure a grow himgely, anxiously
```

KISTI SINCE 1962

2022

Korea Institute of Science and Technology Information

TRUST KISTI