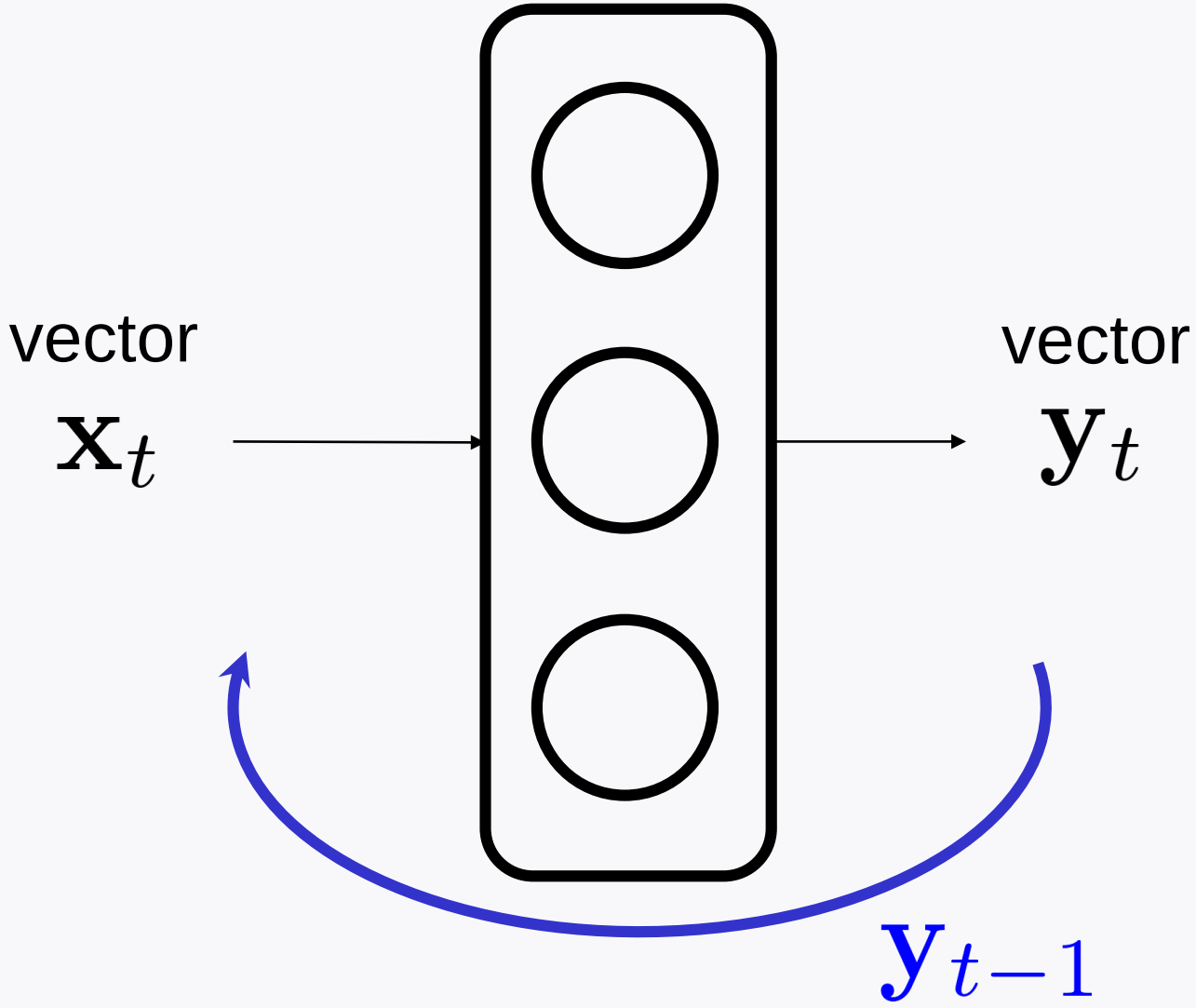
Recurrent neural networks

Practice Session 10

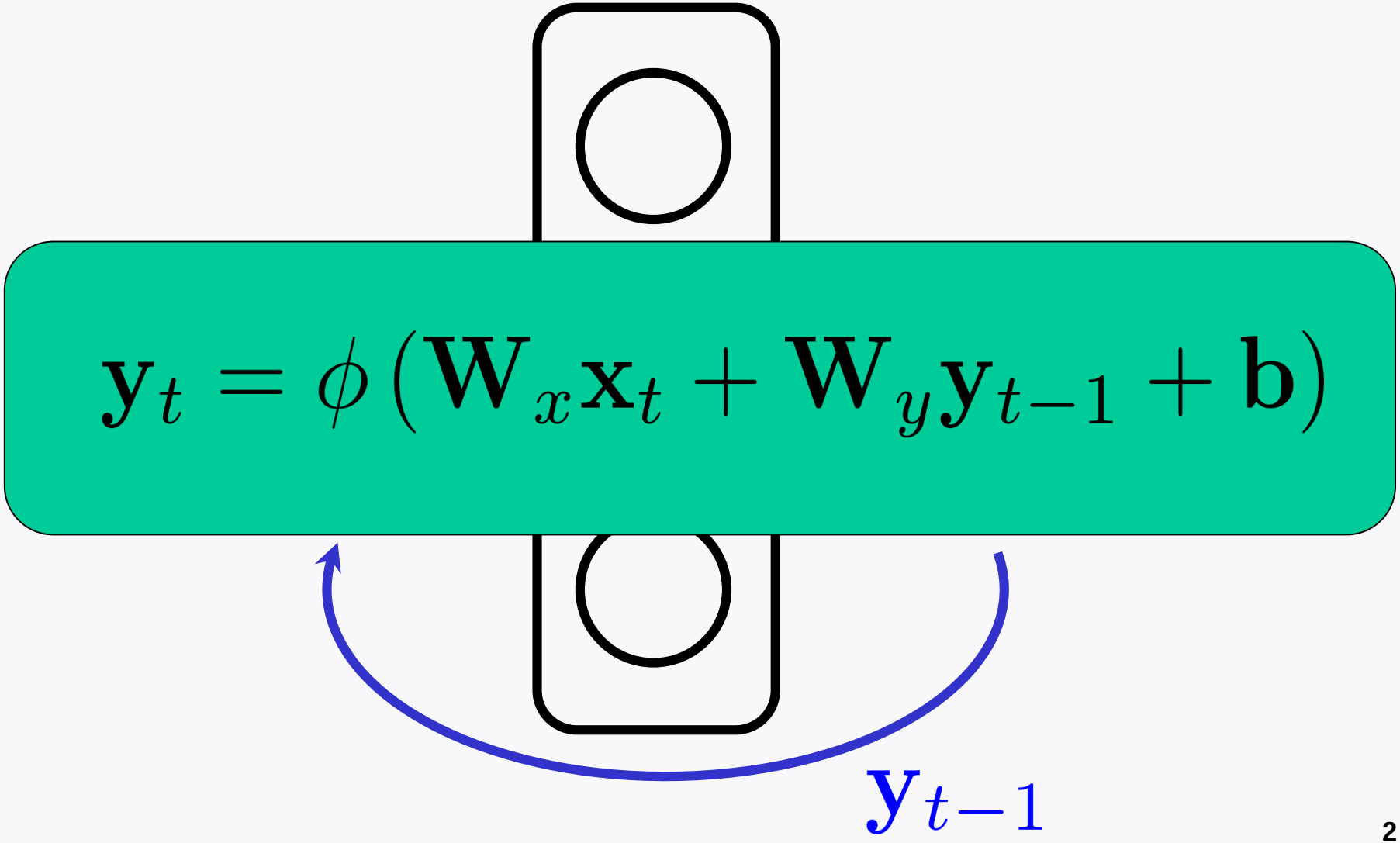
Changho Suh

January 25, 2024

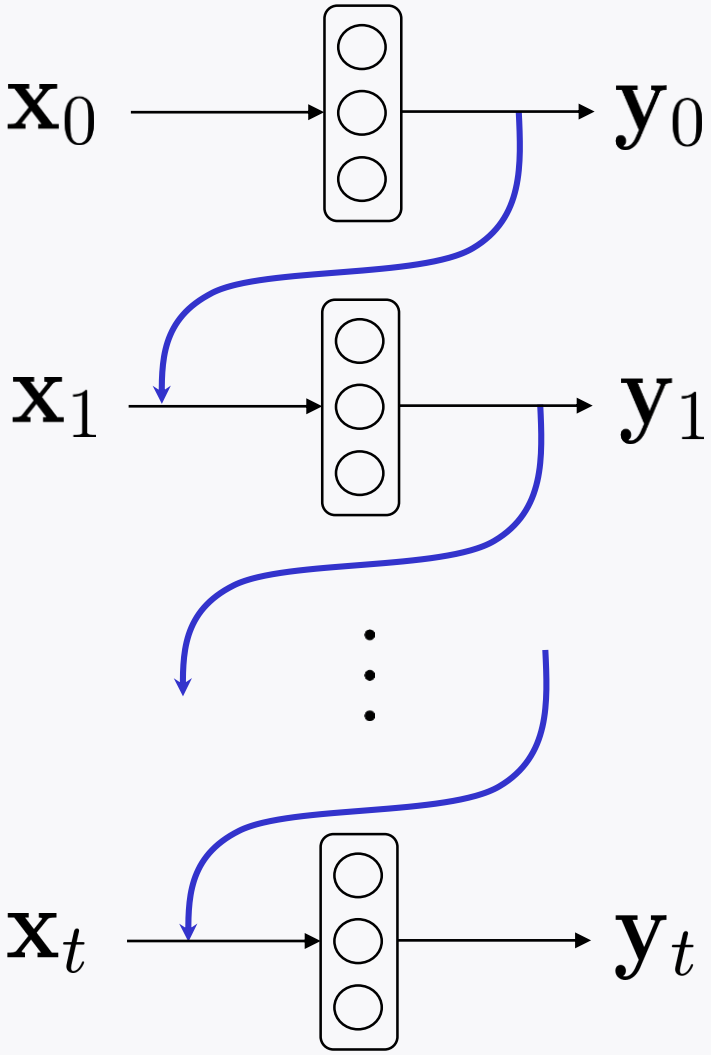
Recap: Recurrent neurons



Recap: Recurrent neurons



Recap: Unrolled version



Recap: A memory cell

An entity that preserves some state $\mathbf{h}_t$ (memory).

Simply called a cell.

A basic cell: A cell such that state = output

$$\mathbf{h}_t = \mathbf{y}_t$$

Basic RNNs: RNNs with basic cells.

Recap: LSTM cell

Performs much better relative to basic RNNs.

Offers faster training and detects long-term dependencies in data.

Idea: Split a state into two:

1. Short-term state $\mathbf{h}_t$
2. Long-term state $\mathbf{c}_t$

Design a cell so that the network can learn how to forget; how to input; and how to output.

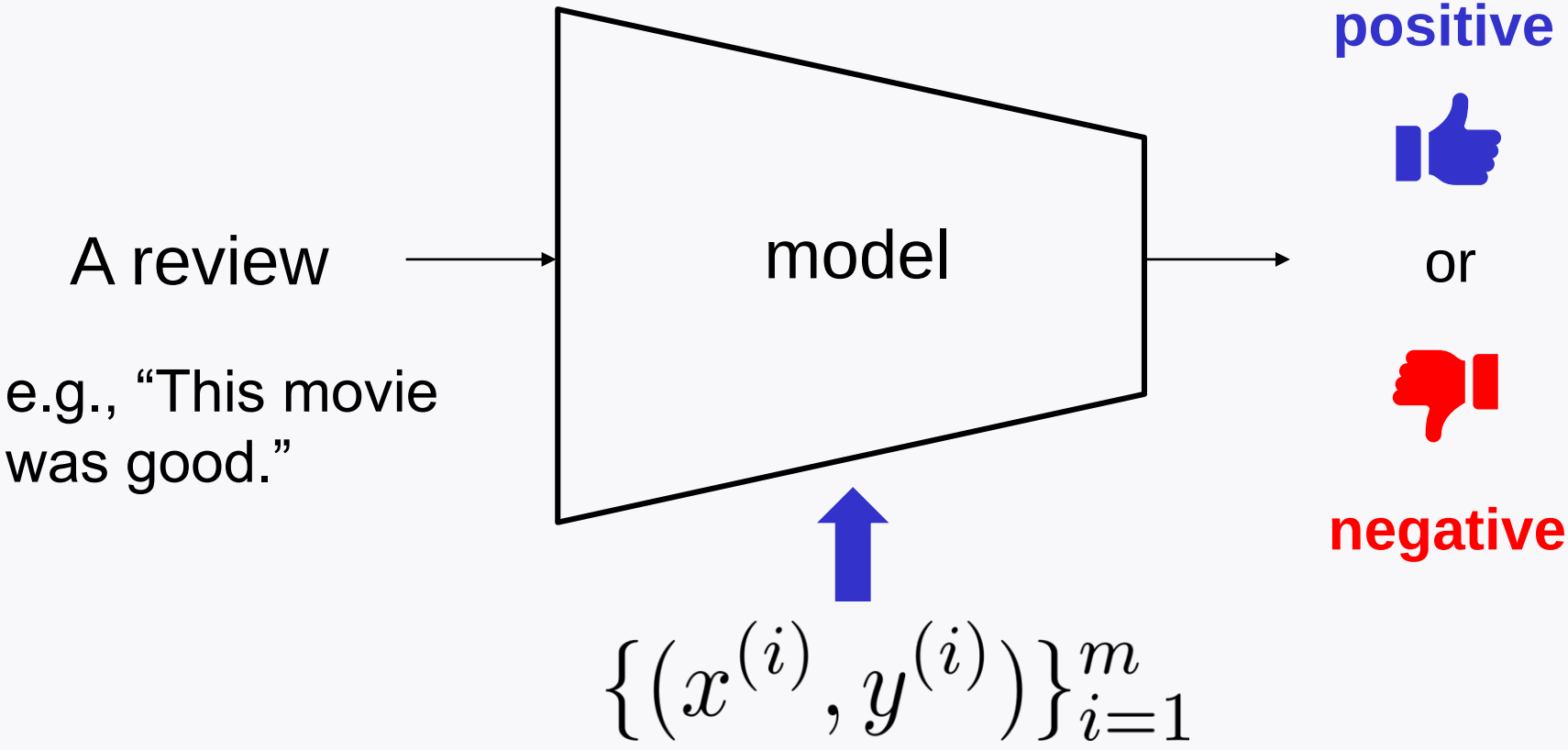
Outline

Task: Movie review classification (IMDB)

Will implement two models:

1. Basic RNN
2. LSTM

Movie review classification



IMDB dataset

Internet **M**ovie **D**ata **B**ase

World's most popular and authoritative source for movie, TV and celebrity content.

Contains many ratings and reviews for the newest movie and TV shows.

Consists of 50,000 movie reviews and ratings.

$$m = 25,000 \qquad m_{\text{test}} = 25,000$$

Label data

$$\{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$$

$$y^{(i)} \in \{1, 0\}$$

positive negative

Number of positive reviews: 12,500 (out of 25,000)

both for train and test data

Review data

$$\{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$$

Text: A collection of words

Average number of words per text: ~ 239

Maximum length: 256

Each word is represented with a numerical value via **dictionary**.

Dictionary

Two concepts required to understand “**dictionary**”.

1. **Key**
2. **Value**

Dictionary is a **set** of **(Key, Value) pairs**.

called “**item**”

Dictionary: IMDB example

Key: This Value: 11

Key: movie Value: 17

Key: was Value: 13

Key: good Value: 49

Dictionary: IMDB example

word_idx_dict

Key: this — Value: 11

Key: movie — Value: 17

Key: was — Value: 13

Key: good — Value: 49

```
word_idx_dict['movie'] = 17
```

Key Value

Dictionary in Python

```
word_dict = {'This':11, 'movie':17, 'was':13, 'good':49}

print(word_dict.keys())

dict_keys(['This', 'movie', 'was', 'good'])

print(word_dict.values())

dict_values([11, 17, 13, 49])

print(word_dict.items())

dict_items([('This', 11), ('movie', 17), ('was', 13), ('good', 49)])
```

Type of word_dict.keys()?

```
word_dict = {'This':11, 'movie':17, 'was':13, 'good':49}
```

```
print(word_dict.keys())
```

```
dict_keys(['This', 'movie', 'was', 'good'])
```

```
print(type(word_dict.keys()))
```

```
<class 'dict_keys'>
```

‘tuple’

```
word_dict.keys()[0]='replace'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_38408/265923385.py in <module>
----> 1 word_dict.keys()[0]='replace'

TypeError: 'dict_keys' object does not support item assignment
```

Convert “tuple” to “list”

```
word_dict = {'This':11, 'movie':17, 'was':13, 'good':49}
```

```
print(type(word_dict.keys()))
```

```
<class 'dict_keys'>
```

‘tuple’

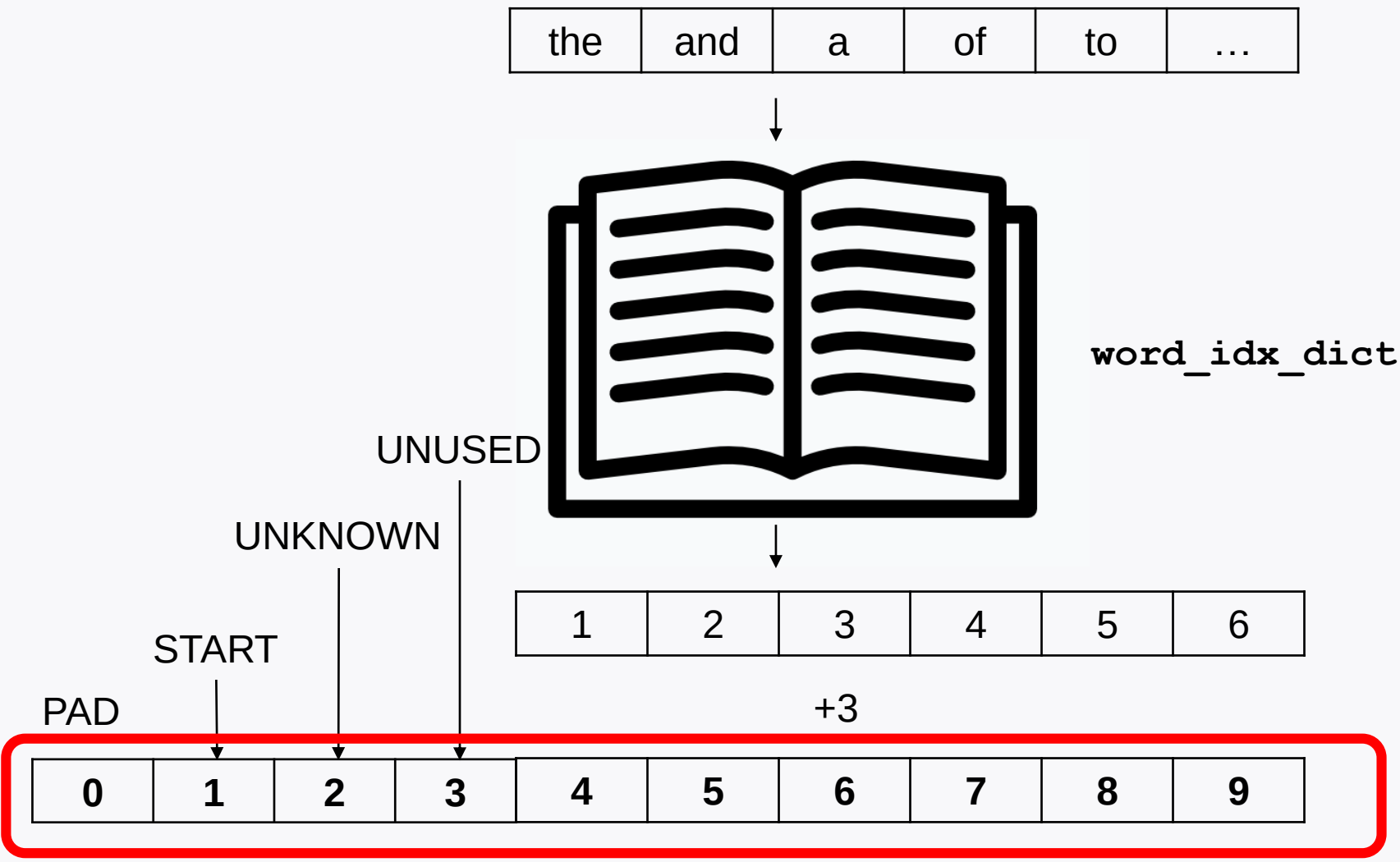
```
# convert "tuple" to "list"
b = list(word_dict.keys())
print(type(b))
```

```
<class 'list'>
```

```
b[0]='replace'
print(b)
```

```
['replace', 'movie', 'was', 'good']
```

Actual values stored in IMDB dataset



Actual values stored in IMDB dataset

Loading IMDB

```
from tensorflow.keras.datasets import imdb

(X_train,y_train), (X_test,y_test) = imdb.load_data(num_words=10000)
```

total number of vocabularies employed

```
print(X_train.shape)
print(X_test.shape)
```

```
(25000,)
(25000,)
```

IMDB: Average length of words per text

```
print(len(X_train[0]))
print(len(X_train[1]))
print(len(X_train[2]))
```

```
218
189
141
```

```
import numpy as np
len_list = [len(s) for s in X_train]
print(np.mean(len_list))
```

```
238.71364
```

IMDB: Vocabularies

```
# total number of vocabularies = 10,000
print(np.max(X_train[3]))
```

```
9941
```

```
# total number of vocabularies = 10,000
max_list = [np.max(s) for s in X_train]
print(np.max(max_list))
```

```
9999
```

IMDB: Review stats

```
# How many positive reviews?
print(np.sum(y_train))
print(np.sum(y_test))
```

```
12500
12500
```

IMDB dictionary

```
# Get IMDB dictionary
word_idx_dict = imdb.get_word_index()

print(word_idx_dict.keys())
```

```
dict_keys(['fawn', 'tsukino', 'nunnery', 'sonja', 'vani', 'woods', 'spiders', 'hangi
ng', 'woody', 'trawling', "hold's", 'comically', 'localized', 'disobeying', "'royal
e", "harpo's", 'canet', 'aileen', 'acurately', "diplomat's", 'rickman', 'arranged',
'rumbustious', 'familiarness', "spider'", 'hahahah', "wood'", 'transvestism', "hangi
n'", 'bringing', 'seamier', 'wooded', 'bravora', 'grueling', 'wooden', 'wednesday',
"'prix", 'altagracia', 'circuitry', 'crotch', 'busybody', "tart'n'tangy", 'burgade',
'thrace', "tom's", 'snuggles', 'francesco', 'complainers', 'templarios', '272', '27
3', 'zaniacs', '275', 'consenting', 'snuggled', 'inanimate', 'uality', 'bronte', 'er
rors', 'dialogs', "yomada's", "madman's", 'dialoge', 'usenet', 'videodrome', "kid'",
'pawed', "'girlfriend'", "'pleasure", "'reloaded'", "kazakos'", 'rocque', 'mailing
s', 'brainwashed', 'mcanally', "tom'", 'kurupt', 'affiliated', 'babaganoosh', "no
e's", 'quart', 'kids', 'uplifting', 'controversy', 'kida', 'kidd', "error'", 'neurol
ogist', 'spotty', 'cobblers', 'projection', 'fastforwarding', 'sters', "eggar's", 'e
therything', 'gateshead', 'airball', 'unsinkable', 'stern', "cervi's", 'dnd', 'dna',
'insecurity', "'reboot'", 'trelkovsky', 'jaekel', 'sidebars', "sforza's", 'distortio
ns', 'mutinies', 'sermons', '7ft', 'boobage', "o'bannon's", 'populations', 'chulak',
'mesmerize', 'quinnell', 'yahoo', 'meteorologist', 'beswick', 'boorman', 'voicewor
k', "ster'", 'blustering', 'hj', 'intake', 'morally', 'jumbling', 'bowersock', "'por
ky's'", 'gershon', 'ludicrosity', 'coprophilia', 'expressively', "india's", "pos
```

Look ahead

Will learn more about IMDB dataset.

Will do data processing.