[Char-RNN을 사용해 셰익스피어 문체 생성하기]

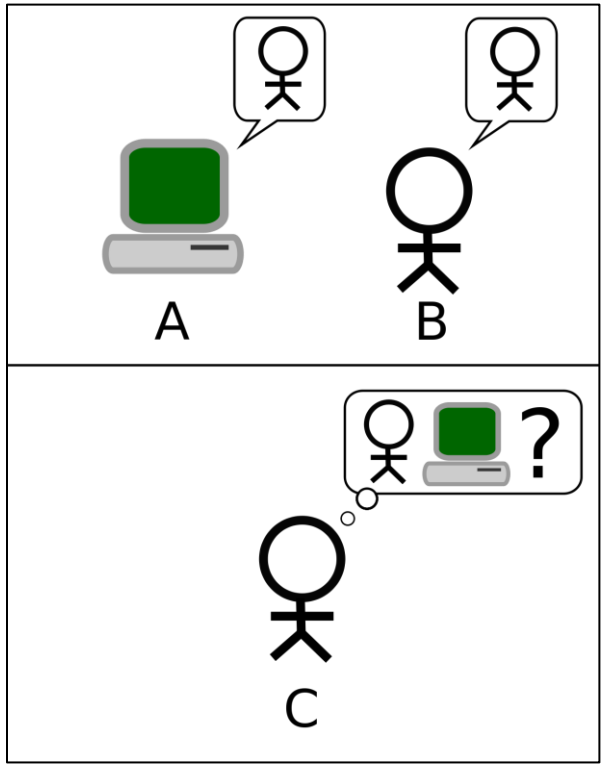
16.1.1 ~ 16.1.2

훈련 데이터셋 만들기 ~ 순차 데이터셋 나누기

2021.05.09 백관구

16. RNN과 어텐션을 사용한 자연어 처리

- 앨런 튜링(Alan Turing)
  - 언어를 마스터하는 것이 인간(호모 사피엔스)의 놀라운 인지 능력
  - 튜링 테스트(1950): 대화의 상대방이 자신을 사람이라고 생각하도록 속일 수 있는 챗봇(chatbot)인지 시험

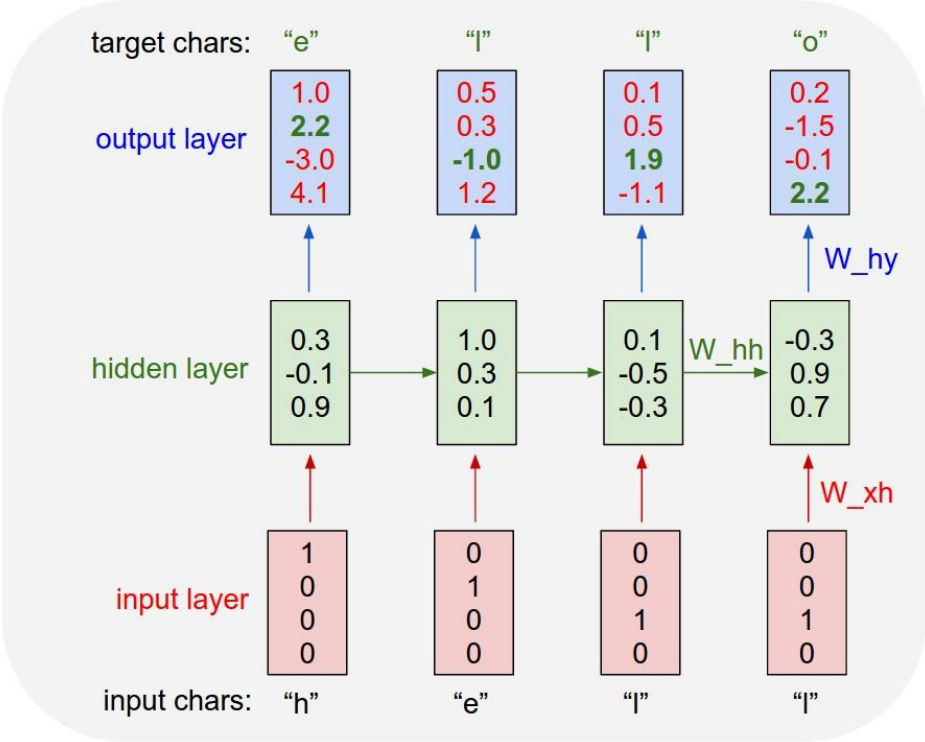


튜링 테스트

- 자연어 문제를 위해 많이 사용하는 방법이 순환 신경망(RNN)
  - 자연어(Natural language): 사람들이 일상적으로 쓰는 언어(예. 한국어, 영어)
  - 16단원에서는 아래와 같은 자연어 모델들을 살펴볼 예정
    - 문자단위 RNN (Character RNN)
    - 상태가 없는 RNN (Stateless RNN)
    - 상태가 있는 RNN (Stateful RNN)
    - 인코더-디코더 네트워크
    - 트랜스포머 (어텐션 메커니즘)

16.1. Char-RNN을 사용해 셰익스피어 문체 생성하기

- 문자단위 RNN (Character RNN; Char-RNN)
  - RNN을 훈련하여 문장에서 다음 글자를 예측하는 방법
  - 한 글자씩 새로운 텍스트를 생성



16.1.1. 훈련 데이터셋 만들기 (Char-RNN을 사용한 셰익스피어 문체 생성)

- 셰익스피어 작품 다운로드
  - tensorflow.keras.utils.get_file(파일명, 다운로드 경로) 을 사용하여 url 상의 텍스트를 받음

```
from tensorflow import keras

url = 'https://homl.info/shakespeare'
path = keras.utils.get_file('shakespeare.txt', url)
with open(path) as f:
    text = f.read()
```

[2] ✓ 2.9s

```
Downloading data from https://homl.info/shakespeare
1122304/1115394 [==============================] - 0s 0us/step
```

```
text
```

[5] ✓ 0.3s

```
"First Citizen:\nBefore we proceed any further, hear me speak.\n\nAll:\nSpeak, speak.\n\nFirst Citizen:\nYou are all resolved rather to die
Citizen:\nFirst, you know Caius Marcius is chief enemy to the people.\n\nAll:\nWe know't, we know't.\n\nFirst Citizen:\nLet us kill him, and
\n\nAll:\nNo more talking on't; let it be done: away, away!\n\nSecond Citizen:\nOne word, good citizens.\n\nFirst Citizen:\nWe are accounted
surfeits on would relieve us: if they\nwould yield us but the superfluity, while it were\nwholesome, we might guess they relieved us humanel
that\nafflicts us, the object of our misery, is as an\ninventory to particularise their abundance; our\nsufferance is a gain to them Let us
```

16.1.1. 훈련 데이터셋 만들기 (Char-RNN을 사용한 셰익스피어 문체 생성)

- 인코딩: 글자 → 정수
  - tensorflow.keras.preprocessing.text.Tokenizer(char_level = True) 를 사용하여 인코딩
  - char_level = True 시 단어 단위 대신 글자 단위로 인코딩

```
[7]  tokenizer = keras.preprocessing.text.Tokenizer(char_level = True)
     tokenizer.fit_on_texts(text)
     tokenizer.word_index
     ✓ 1.3s

{' ': 1,
 'e': 2,
 't': 3,
 'o': 4,
 'a': 5,
 'i': 6,
 'h': 7,
 's': 8,
 'r': 9,
 'n': 10,
```

```
[8]  import numpy as np

     [encoded] = np.array(tokenizer.texts_to_sequences([text])) - 1
     [encoded]
     ✓ 0.5s

[array([19,  5,  8, ..., 20, 26, 10])]
```

16.1.2. 순차 데이터셋을 나누는 방법 (Char-RNN을 사용한 셰익스피어 문체 생성)

- 훈련, 검증, 테스트 세트가 중복되지 않도록 데이터셋을 만들어야 함
  - 시계열 데이터의 경우, 일반적으로 시간에 따라 나눔(예. 2000~2012 훈련, 2013~2015 검증, 2016~2018 테스트)

    ➔ Stationary 가정: 과거(훈련)에서 학습하는 패턴이 미래(검증, 테스트)에도 등장(주가 예측이 어려운 이유)
  - 텍스트 데이터의 경우, 두 세트 사이에 데이터가 걸치지 않고 완전히 분리되어야 함
- 셰익스피어 텍스트의 처음 90%를 훈련 세트로 사용
- tf.data.Dataset.from_tensor_slices 를 사용해 tf.data.Dataset 객체 생성

```
dataset_size = tokenizer.document_count
train_size   = dataset_size * 90 // 100
print(f'전체길이: {dataset_size}, 훈련길이: {train_size}')
```

[12] ✓ 0.1s

```
전체길이: 1115394, 훈련길이: 1003854
```

```
import tensorflow as tf

dataset = tf.data.Dataset.from_tensor_slices(encoded[: train_size])
dataset
```

[13] ✓ 0.2s

```
<TensorSliceDataset shapes: (), types: tf.int32>
```