Week3

CH16 - RNN과 어텐션을 사용한 자연어 처리

16.1 Char-RNN을 사용해 세익스피어 문체 생성하기

16.1.1 훈련 데이터셋 만들기

```
#데이터 로딩
shakespeare_url = "https://raw.githubusercontent.com/karpathy/char-rnn/master/data/tinyshakespeare/input.txt"
filepath = keras.utils.get_file("shakespeare.txt", shakespeare_url)
with open(filepath) as f:
    shakespeare_text = f.read()

#전처리 - 토큰화
tokenizer = keras.preprocessing.text.Tokenizer(char_level=True)
tokenizer.fit_on_texts(shakespeare_text)

#글자 수준 인코딩 결과
>>> tokenizer.texts_to_sequences(["First"])
[[20, 6, 9, 8, 3]]
>>> tokenizer.sequences_to_texts([[20, 6, 9, 8, 3]])
['f i r s t']

max_id = len(tokenizer.word_index) # 고유 글자 개수
dataset_size = tokenizer.document_count # 전체 글자 개수

[encoded] = np.array(tokenizer.texts_to_sequences([shakespeare_text])) - 1
```

16.1.2 순차 데이터셋을 나누는 방법

훈련셋, 테스트셋, 검증셋은 중복되지 않도록 만들어야 함

시계열 → 시간에 따라 분할

```
train_size = dataset_size * 90 // 100
dataset = tf.data.Dataset.from_tensor_slices(encoded[:train_size])
```

16.1.3 순차 데이터를 윈도 여러 개로 자르기

긴 시퀀스 → 작고 많은텍스트 윈도로 변환

TBPTT: 문자열 길이만큼 역전파를 위해 펼침

```
n_steps = 100
window_length = n_steps + 1 # target = 1글자 앞의 input
dataset = dataset.window(window_length, shift=1, drop_remainder=True) #모든 윈도가 동일한 갯수의 글자 포함
```

- 리스트의 리스트와 비슷한 **중첩 데이터셋**을 만듦
- 데이터셋이 아닌 텐서를 기대 → **플랫 데이터셋**으로 변환

```
dataset = dataset.flat_map(lambda window: window.batch(window_length))

batch_size = 32
dataset = dataset.shuffle(10000).batch(batch_size)
dataset = dataset.map(lambda windows: (windows[:, :-1], windows[:, 1:]))
```

- 범주형 입력 특성: 원-핫 벡터 or 임베딩으로 인코딩

```
#원-핫 벡터로 인코딩
dataset = dataset.map(
    lambda X_batch, Y_batch: (tf.one_hot(X_batch, depth=max_id), Y_batch))

#프리페칭
dataset = dataset.prefetch(1)
```

16.1.4 Char-RNN 모델 만들고 훈련하기

```
model = keras.models.Sequential([
    keras.layers.GRU(128, return_sequences=True, input_shape=[None, max_id],
                     #dropout=0.2, recurrent_dropout=0.2), #GPU만 사용
                     dropout=0.2),
    keras.layers.GRU(128, return_sequences=True,
                     #dropout=0.2, recurrent_dropout=0.2), #GPU만 사용
                     dropout=0.2),
    keras.layers.TimeDistributed(keras.layers.Dense(max_id,
                                                    activation="softmax"))
])
model.compile(loss="sparse_categorical_crossentropy", optimizer="adam")
history = model.fit(dataset, epochs=10)
```

16.1.5 Char-RNN 모델 사용하기

```
# 평가를 위한 텍스트를 전처리하는 함수 생성
def preprocess(texts):
    X = np.array(tokenizer.texts_to_sequences(texts)) - 1
    return tf.one_hot(X, max_id)

#모델 사용 및 평가
>>> X_new = preprocess(["How are yo"])
>>> Y_pred = np.argmax(model(X_new), axis=-1)
>>> tokenizer.sequences_to_texts(Y_pred + 1)[0][-1] # 첫 번째 문장 마지막 글자
'u' #How are you
```

16.1.6 가짜 셰익스피어 텍스트 생성하기

- `tf.random.categorical` 함수를 사용

  모델이 추정한 확률을 기반으로 다음 글자를 무작위로 선택

- `categorical( )` 함수

  클래스의 로그 확률(=로짓)을 전달하면 랜덤하게 클래스 인덱스 샘플링

  생성된 텍스트의 다양성 제어 → '**온도**' 숫자로 로짓을 나눔

  온도: 원하는 값으로 설정 / 0에 가까울수록 높은 확률을 가진 글자 선택 / 매우 높으면 모든 글자 같은 확률

```
def next_char(text, temperature=1):
    X_new = preprocess([text])
    y_proba = model(X_new)[0, -1:, :]
    rescaled_logits = tf.math.log(y_proba) / temperature
    char_id = tf.random.categorical(rescaled_logits, num_samples=1) + 1
    return tokenizer.sequences_to_texts(char_id.numpy())[0]

#함수 반복 호출
def complete_text(text, n_chars=50, temperature=1):
    for _ in range(n_chars):
        text += next_char(text, temperature)
    return text

#온도 별 결과
>>> print(complete_text("t", temperature=0.2))
the maid in padua for my father is a stood
and so m

>>> print(complete_text("t", temperature=1)) #1일 때 제일 잘 작동함
toke on advised in sobel countryman,
and signior gr

>>> print(complete_text("t", temperature=2))
tpeniomently!
well maze: yet 'pale deficuruli-faeem
```

좋은 모델을 위해서...

- GRU 층과 층의 뉴런 수 늘리기
- 훈련 시간 늘리기
- 규제 추가하기

~~※ 현재 글자인 n_steps보다 긴 패턴을 학습할 수 없음~~

→ LSTM나 GRU도 매우 긴 시퀀스는 다룰 수 ❌

→상태가 있는 RNN 사용 ‼

16.1.7 상태가 있는 RNN

상태가 있는 RNN

- 한 훈련 배치를 처리한 후 마지막 상태를 다음 훈련 배치의 초기 상태로 사용
- 역전파는 짧은 시퀀스 - 모델은 장기간 패턴 학습

```
#순차적이고 겹치지 않는 입력 시퀀스 생성
dataset = tf.data.Dataset.from_tensor_slices(encoded[:train_size])
dataset = dataset.window(window_length, shift=n_steps, drop_remainder=True) #shift=1 대신 n_steps 사용
dataset = dataset.flat_map(lambda window: window.batch(window_length))
dataset = dataset.batch(1)                                                  #shuffle( ) 사용하면 안됨
dataset = dataset.map(lambda windows: (windows[:, :-1], windows[:, 1:]))
dataset = dataset.map(
    lambda X_batch, Y_batch: (tf.one_hot(X_batch, depth=max_id), Y_batch))
dataset = dataset.prefetch(1)

#모델 생성
model = keras.models.Sequential([
    keras.layers.GRU(128, return_sequences=True, stateful=True, #stateful 지정
                     dropout=0.2, recurrent_dropout=0.2,
                     batch_input_shape=[batch_size, None, max_id]), #batch_input_shape 매개변수 지정
    keras.layers.GRU(128, return_sequences=True, stateful=True,
                     dropout=0.2, recurrent_dropout=0.2),
    keras.layers.TimeDistributed(keras.layers.Dense(max_id,
                                                    activation="softmax"))
])

#상태 재설정 - 콜백
class ResetStatesCallback(keras.callbacks.Callback):
    def on_epoch_begin(self, epoch, logs):
        self.model.reset_states()
```

16.2 감성 분석

- 전처리

    - 공백 처리
        - 다쿠 구도
        - 바이트 페어 인코딩
        - TF.Text 라이브러리

- 전처리를 모델 자체에 포함시키기

```
# 원본 데이터를 텍스트로 적재
import tensorflow_datasets as tfds

datasets, info = tfds.load("imdb_reviews", as_supervised=True, with_info=True)
train_size = info.splits["train"].num_examples
test_size = info.splits["test"].num_examples

#전처리
def preprocess(X_batch, y_batch):
    X_batch = tf.strings.substr(X_batch, 0, 300)
    X_batch = tf.strings.regex_replace(X_batch, rb"<br\s*/?>", b" ")
    X_batch = tf.strings.regex_replace(X_batch, b"[^a-zA-Z']", b" ")
    X_batch = tf.strings.split(X_batch)
    return X_batch.to_tensor(default_value=b"<pad>"), y_batch

#어휘 사전 구축
from collections import Counter

vocabulary = Counter()
for X_batch, y_batch in datasets["train"].batch(32).map(preprocess):
    for review in X_batch:
        vocabulary.update(list(review.numpy()))

#각 단어를 어휘 사전의 인덱스로 변환
#룩업 테이블 생성
words = tf.constant(truncated_vocabulary)
word_ids = tf.range(len(truncated_vocabulary), dtype=tf.int64)
vocab_init = tf.lookup.KeyValueTensorInitializer(words, word_ids)
num_oov_buckets = 1000
table = tf.lookup.StaticVocabularyTable(vocab_init, num_oov_buckets)

#모델링
embed_size = 128
model = keras.models.Sequential([
    keras.layers.Embedding(vocab_size + num_oov_buckets, embed_size,
                           input_shape=[None]),
    keras.layers.GRU(128, return_sequences=True),
    keras.layers.GRU(128),
    keras.layers.Dense(1, activation="sigmoid")
])
model.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accuracy"])
history = model.fit(train_set, epochs=5)
```

16.2.1 마스킹

- 모든 층에서 패딩 토큰 무시

  → 임베딩 층에 `mask_zero=True` 매개변수 추가

- 수동으로 마스킹

```
K = keras.backend
embed_size = 128
inputs = keras.layers.Input(shape=[None])
mask = keras.layers.Lambda(lambda inputs: K.not_equal(inputs, 0))(inputs) #마스킹
z = keras.layers.Embedding(vocab_size + num_oov_buckets, embed_size)(inputs)
z = keras.layers.GRU(128, return_sequences=True)(z, mask=mask)
z = keras.layers.GRU(128)(z, mask=mask)
outputs = keras.layers.Dense(1, activation="sigmoid")(z)
model = keras.models.Model(inputs=[inputs], outputs=[outputs])
model.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accuracy"])
history = model.fit(train_set, epochs=5)
```

16.2.2. 사전훈련된 임베딩 재사용하기

- 사전훈련된 모델 컴포넌트: **모듈**

```
#사전훈련 모델
import tensorflow_hub as hub

model = keras.Sequential([
    hub.KerasLayer("https://tfhub.dev/google/tf2-preview/nnlm-en-dim50/1",
                   dtype=tf.string, input_shape=[], output_shape=[50]),
    keras.layers.Dense(128, activation="relu"),
    keras.layers.Dense(1, activation="sigmoid")
])
model.compile(loss="binary_crossentropy", optimizer="adam",
              metrics=["accuracy"])

#학습시키고자 하는 모델
import tensorflow_datasets as tfds

datasets, info = tfds.load("imdb_reviews", as_supervised=True, with_info=True)
train_size = info.splits["train"].num_examples
batch_size = 32
train_set = datasets["train"].batch(batch_size).prefetch(1) #배치와 프리페치를 제외하고 전처리 필요 X
history = model.fit(train_set, epochs=5) #바로 학습
```

16.3 신경망 기계 번역을 위한 인코더-디코더 네트워크

- 영어를 프랑스어로 번역

  인코더 → 영어

  디코더 → 프랑스어

    SOS: Start-of-Sequence 디코더의 입력을 의미

    EOS: End-of-Sequence

- 영어 문장이 인코더로 주입 시 **문장이 뒤집어 짐**

  디코더가 번역할 첫 번째 단어이기 때문...!

- 텐서플로 애드온으로 인코더-디코더 구현

16.3.1 양방향 RNN

단어를 인코딩하기 전 다음 단어 미리보기

- 동일한 입력에 대해 두 개의 순환 층: **양방향 순환 층**

```
model = keras.models.Sequential([
    keras.layers.GRU(10, return_sequences=True, input_shape=[None, 10]),
    keras.layers.Bidirectional(keras.layers.GRU(10, return_sequences=True)) #양방향 순환 층
])

model.summary()
```

16.3.2 빔 검색

모델의 실수를 고치는 방법: **빔 검색**

- 가능성이 있는 k개의 문장 리스트 유지
- 디코더 마다 문장의 단어를 하나씩 생성 → k개의 문장 생성
- 파라미터 k: **빔 너비**
- 텐서플로 애드온으로 구현 `BeamSearchDecoder`

⇒ 긴 문장에서는 성능이 매우 나쁨... (해결: 어텐션 매커니즘 등장 !)

16.4 어텐션 매커니즘

- 적절한 단어에 디코더가 초점을 맞추는 기술
- 단기 기억의 제한성에 적은 영향
- **구조**
  - 인코더의 모든 출력을 디코더로 전송

    - 디코더는 모든 인코더 출력의 가중치 합을 계산 → 집중할 단어 결정

      (가중치가 클 수록 주의를 기울여야하는 단어임)

- **가중치는 어디서 오는 것일까?**
    - 정렬 모델(=어텐션 층)
    - `TimeDistributed` 클래스를 적용한 `Dense`층으로 시작
    - 하나의 뉴런으로 구성
    - 인코더의 모든 출력을 입력으로 받아 디코더의 이전 은닉 상태를 연결
    - 각 인코더 출력에 대한 점수를 출력
    - 점수는 각 출력이 디코더의 은닉 상태와 얼마나 잘 맞는지 측정
    - 소프트맥스층 통과 후 최종 가중치 (디코더 타임 스텝에 대한 모든 가중치의 합은 1)

⇒ **바흐나우 어텐션 or 연결 어텐션 (= 덧셈 어텐션)**

루옹 어테션 or 곱셈 어텐션

- 인코더의 출력 하나와 디코더의 이전 은닉층 상태 사이의 유사도 측정
- 두 벡터 사이의 점곱

16.4.1 비주얼 어텐션

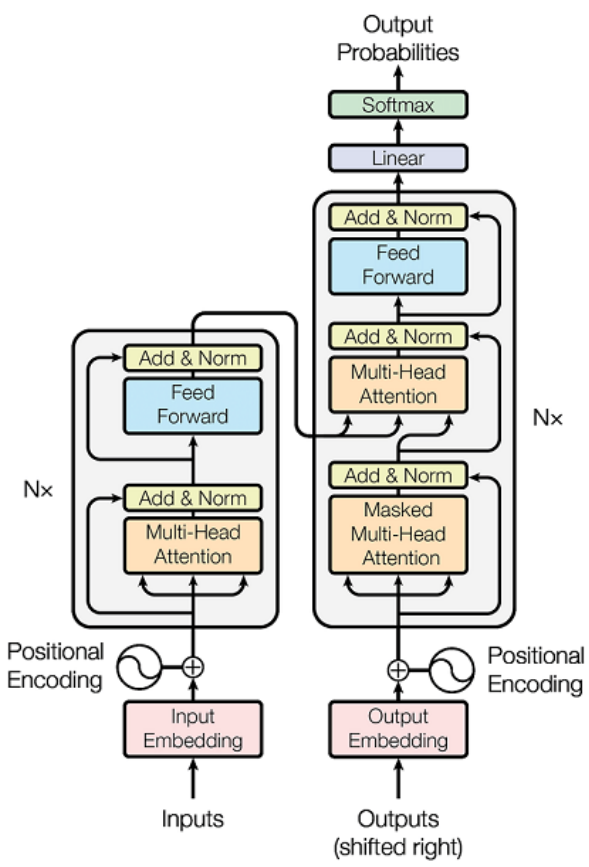
이미지 캡션 생성

- 합성곱 신경망 -이미지 처리 → 특성 맵 출력
- 어텐션 메커니즘 - RNN → 한번에 한 단어
- 초점 생성

→ 장점) **설명 가능성**: 모델이 어떤 출력을 만들도록 이끄는 것이 무엇인지 이해 쉬움

16.4.2 트렌스포머 구조: 어텐션이 필요한 전부다

- **트렌스포머 구조**
    - 왼쪽 인코더
    - 오른쪽 디코더
    - **멀티-헤드 어텐션**: 각 단어와 동일한 문장에 있는 다른 단어의 관계 인코딩 (= 셀프 어텐션)
    - **마스크드 멀티-헤드 어텐션**: 멀티-헤드 어텐션과 동일, 다만 각 단어는 이전에 등장한 단어에만 집중해 디코딩



<위치 인코딩>

- 문장 안에 있는 단어의 위치를 인코딩한 밀집 벡터

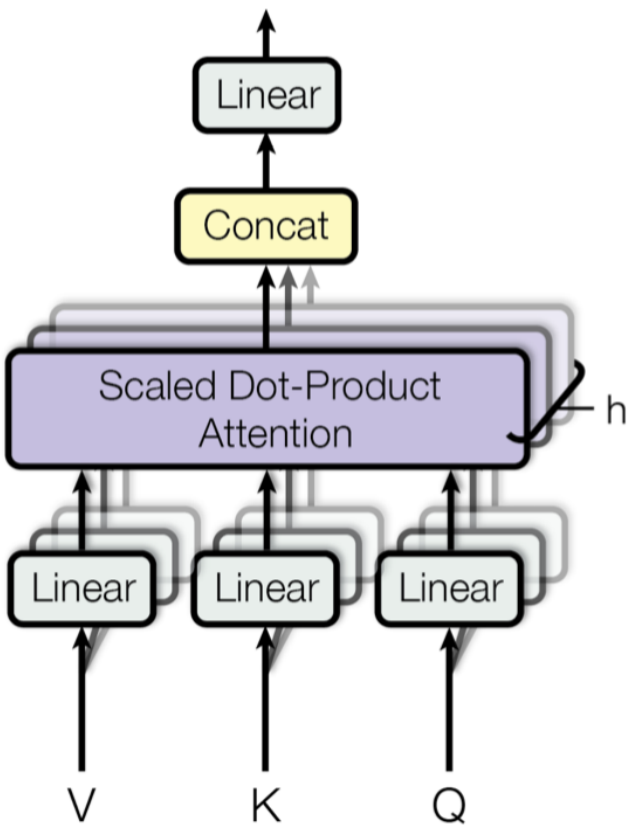
```
#위치 인코딩 생성
class PositionalEncoding(keras.layers.Layer):
    def __init__(self, max_steps, max_dims, dtype=tf.float32, **kwargs):
        super().__init__(dtype=dtype, **kwargs)
        if max_dims % 2 == 1: max_dims += 1 # max_dims는 짝수여야 합니다.
        p, i = np.meshgrid(np.arange(max_steps), np.arange(max_dims // 2))
        pos_emb = np.empty((1, max_steps, max_dims))
        pos_emb[0, :, ::2] = np.sin(p / 10000**(2 * i / max_dims)).T
        pos_emb[0, :, 1::2] = np.cos(p / 10000**(2 * i / max_dims)).T
        self.positional_embedding = tf.constant(pos_emb.astype(self.dtype))
    def call(self, inputs):
        shape = tf.shape(inputs)
        return inputs + self.positional_embedding[:, :shape[-2], :shape[-1]]

#트렌스포머의 첫 번째 층 생성
embed_size = 512; max_steps = 500; vocab_size = 10000
encoder_inputs = keras.layers.Input(shape=[None], dtype=np.int32)
decoder_inputs = keras.layers.Input(shape=[None], dtype=np.int32)
embeddings = keras.layers.Embedding(vocab_size, embed_size)
encoder_embeddings = embeddings(encoder_inputs)
decoder_embeddings = embeddings(decoder_inputs)
positional_encoding = PositionalEncoding(max_steps, max_dims=embed_size)
encoder_in = positional_encoding(encoder_embeddings)
decoder_in = positional_encoding(decoder_embeddings)
```

<멀티-헤드 어텐션>

- **스케일드 점-곱 어텐션**
  - 인코더: 딕셔너리 생성 - 디코더: 키 값 찾기
  - 룩업에 사용할 키(=쿼리)와 딕셔너리에 있는 키 사이의 유사도 계산
  - 유사도 점수 합산 후 1이 되는 가중치로 변환 (소프트맥스 함수 사용)
  - 키와 쿼리가 비슷할 수록 1에 가까워짐

→ 멀티-헤드 어텐션 : 스케일드 점-곱 어텐션층의 묶음



16.5 언어 모델 분야의 최근 혁신

- ELMo
- ULMFiT

- GPT-2
- BERT
  - MLM: 마스크드 언어 모델
  - NSP: 다음 문장 예측

16.6 연습문제