딥 러닝을 이용한 자연어 처리 입문

12. 태깅 작업 (Tagging Task)

12-01 케라스를 이용한 태깅 작업 개요

시퀀스 레이블링

입력 시퀀스에 대해 각 위치마다 레이블을 할당하는 작업

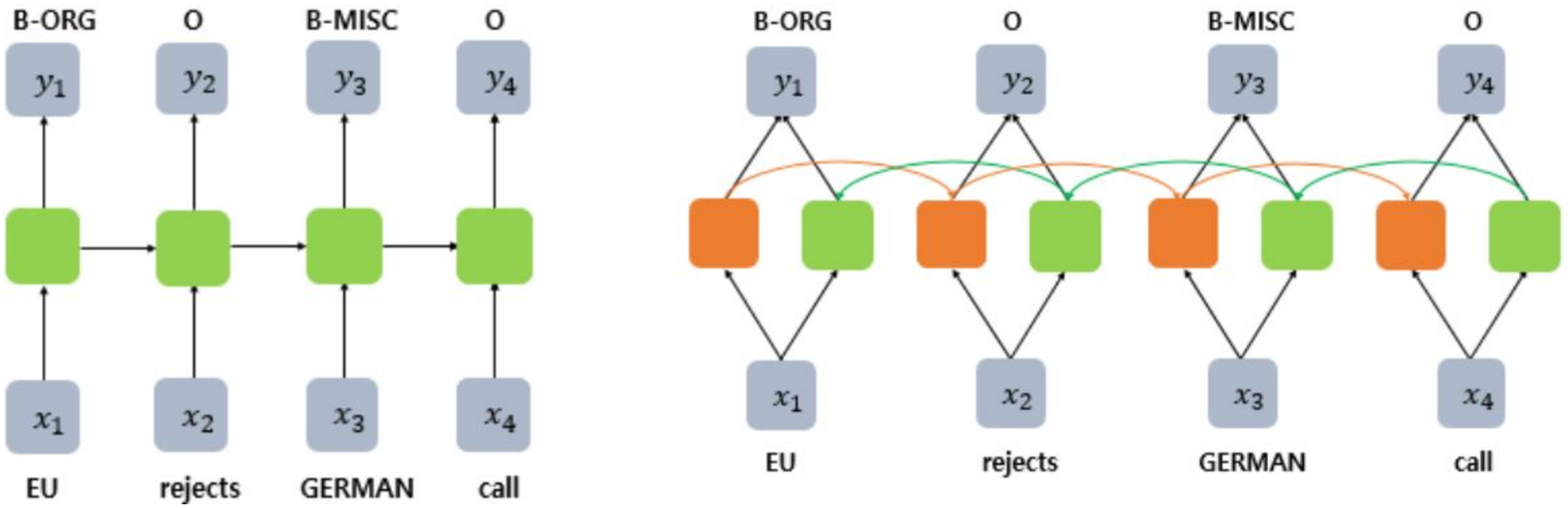
태깅 작업

주어진 텍스트의 단어에 대해 품사 태그나 개체명과 같은 레이블을 할당하는 작업

ex. "I love cats"
"I"는 대명사, "love"는 동사, "cats"는 명사

12-01 케라스를 이용한 태깅 작업 개요

RNN의 다-대-다(Many-to-Many) 문제



양방향 LSTM(Bidirectional LSTM)

```
model.add(Bidirectional(LSTM(hidden_units, return_sequences=True)))
```

12-02 양방향 LSTM를 이용한 품사 태깅

양방향 LSTM(Bi-directional LSTM)으로 POS Tagger 만들기

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, InputLayer, Bidirectional, TimeDistributed, Embedding
from tensorflow.keras.optimizers import Adam

embedding_dim = 128
hidden_units = 128

model = Sequential()
model.add(Embedding(vocab_size, embedding_dim, mask_zero=True))
model.add(Bidirectional(LSTM(hidden_units, return_sequences=True)))
model.add(TimeDistributed(Dense(tag_size, activation=('softmax'))))

model.compile(loss='sparse_categorical_crossentropy', optimizer=Adam(0.001), metrics=['accuracy'])
model.fit(X_train, y_train, batch_size=128, epochs=7, validation_data=(X_test, y_test))
```

12-02 양방향 LSTM를 이용한 품사 태깅

양방향 LSTM(Bi-directional LSTM)으로 POS Tagger 만들기

```
print("\n 테스트 정확도: %.4f" % (model.evaluate(X_test, y_test)[1]))
```

```
25/25 [==============================] - 0s 6ms/step - loss: 0.0720 - accuracy: 0.9016
테스트 정확도: 0.9016
```

```
index_to_word = src_tokenizer.index_word
index_to_tag = tar_tokenizer.index_word

i = 10 # 확인하고 싶은 테스트용 샘플의 인덱스.
y_predicted = model.predict(np.array([X_test[i]])) # 입력한 테스트용 샘플에 대해서 예측값 y를 리턴
y_predicted = np.argmax(y_predicted, axis=-1) # 확률 벡터를 정수 레이블로 변환.

print("{:15}|{:5}|{}".format("단어", "실제값", "예측값"))
print(35 * "-")

for word, tag, pred in zip(X_test[i], y_test[i], y_predicted[0]):
    if word != 0: # PAD값은 제외함.
        print("{:17}: {:7} {}".format(index_to_word[word], index_to_tag[tag].upper(), index_to_tag[pred].uppe
r()))
```

```
단어              |실제값  |예측값
-----------------------------------
in              : IN      IN
addition        : NN      NN
,               : ,       ,
buick           : NNP     NNP
is              : VBZ     VBZ
a               : DT      DT
relatively      : RB      RB
respected       : VBN     VBN
nameplate       : NN      NN
among           : IN      IN
american        : NNP     NNP
express         : NNP     NNP
card            : NN      NN
holders         : NNS     NNS
,               : ,       ,
says            : VBZ     VBZ
0               : -NONE-  -NONE-
*t*-1           : -NONE-  -NONE-
an              : DT      DT
american        : NNP     NNP
express         : NNP     NNP
spokeswoman     : NN      NN
.               : .       .
```

12-03 개체명 인식(Named Entity Recognition)

1. 개체명 인식(Named Entity Recognition)이란?

문장에서 특정한 유형의 단어를 식별하여 그것이 어떤 유형인지를 알려주는 것

"유정이는 2018년에 골드만삭스에 입사했다"

사람(person): "유정이"
조직(organization): "골드만삭스"
시간(time): "2018년"

12-03 개체명 인식(Named Entity Recognition)

2. NLTK를 이용한 개체명 인식(Named Entity Recognition using NTLK)

```
from nltk import word_tokenize, pos_tag, ne_chunk

sentence = "James is working at Disney in London"
# 토큰화 후 품사 태깅
tokenized_sentence = pos_tag(word_tokenize(sentence))
print(tokenized_sentence)
```

```
[('James', 'NNP'), ('is', 'VBZ'), ('working', 'VBG'), ('at', 'IN'), ('Disney', 'NNP'), ('in', 'IN'), ('Londo
n', 'NNP')]
```

```
# 개체명 인식
ner_sentence = ne_chunk(tokenized_sentence)
print(ner_sentence)
```

```
(S
  (PERSON James/NNP)
  is/VBZ
  working/VBG
  at/IN
  (ORGANIZATION Disney/NNP)
  in/IN
  (GPE London/NNP))
```

12-04 개체명 인식의 BIO 표현 이해하기

1. BIO 표현

개체명이 시작되는 부분을 "B"로 표시,
개체명의 내부 부분을 "I"로 표시,
개체명이 아닌 부분은 "O"로 표시

```
해 B-movie
리 I-movie
포 I-movie
터 I-movie
보 O
러 O
메 B-theater
가 I-theater
박 I-theater
스 I-theater
가 O
자 O
```

12-04 개체명 인식의 BIO 표현 이해하기

2. 개체명 인식 데이터 이해하기

[단어] [품사 태깅] [청크 태깅] [개체명 태깅]

```
EU NNP B-NP B-ORG
rejects VBZ B-VP O
German JJ B-NP B-MISC
call NN I-NP O
to TO B-VP O
boycott VB I-VP O
British JJ B-NP B-MISC
lamb NN I-NP O
. . O O

Peter NNP B-NP B-PER
Blackburn NNP I-NP I-PER
```

12-04 개체명 인식의 BIO 표현 이해하기

3. 양방향 LSTM(Bi-directional LSTM)으로 개체명 인식기

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding, LSTM, Bidirectional, TimeDistributed
from tensorflow.keras.optimizers import Adam

embedding_dim = 128
hidden_units = 128

model = Sequential()
model.add(Embedding(input_dim=vocab_size, output_dim=embedding_dim, input_length=max_len, mask_zero=True))
model.add(Bidirectional(LSTM(hidden_units, return_sequences=True)))
model.add(TimeDistributed(Dense(tag_size, activation='softmax')))

model.compile(loss='categorical_crossentropy', optimizer=Adam(0.001), metrics=['accuracy'])
model.fit(X_train, y_train, batch_size=128, epochs=8, validation_data=(X_test, y_test))
```

```
print("\n 테스트 정확도: %.4f" % (model.evaluate(X_test, y_test)[1]))
```

```
2809/2809 [==============================] - 9s 3ms/step
테스트 정확도: 0.9573
```

12-04 개체명 인식의 BIO 표현 이해하기

3. 양방향 LSTM(Bi-directional LSTM)으로 개체명 인식기

```
i = 10 # 확인하고 싶은 테스트용 샘플의 인덱스.

# 입력한 테스트용 샘플에 대해서 예측 y를 리턴
y_predicted = model.predict(np.array([X_test[i]]))

# 확률 벡터를 정수 레이블로 변경.
y_predicted = np.argmax(y_predicted, axis=-1)

# 원-핫 벡터를 정수 인코딩으로 변경.
labels = np.argmax(y_test[i], -1)

print("{:15}|{:5}|{}".format("단어", "실제값", "예측값"))
print(35 * "-")

for word, tag, pred in zip(X_test[i], labels, y_predicted[0]):
    if word != 0: # PAD값은 제외함.
        print("{:17}: {:7} {}".format(index_to_word[word], index_to_ner[tag].upper(), index_to_ner[pred].uppe
r()))
```

```
단어              |실제값   |예측값
-----------------------------------
sarah             : B-PER   B-PER
brady             : I-PER   I-PER
,                 : O       O
whose             : O       O
republican        : B-MISC  B-MISC
husband           : O       O
was               : O       O
OOV               : O       O
OOV               : O       O
in                : O       O
an                : O       O
OOV               : O       O
attempt           : O       O
on                : O       O
president         : O       O
ronald            : B-PER   B-PER
reagan            : I-PER   I-PER
,                 : O       O
took              : O       O
centre            : O       O
stage             : O       O
at                : O       O
the               : O       O
```

12-05 BiLSTM을 이용한 개체명 인식

양방향 LSTM을 이용한 개체명 인식 - F1-score

모델이 어떠한 개체도 맞추지 못하고 전부 'O'로 예측한 경우

```
예측값 : ['O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',
'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O']
```

```
accuracy = hit/len(labels) # 정답 개수를 총 개수로 나눈다.
print("정확도: {:.1%}".format(accuracy))
```

정확도: 74.4%

$$\text{정밀도} = \frac{TP}{TP+FP} = \text{특정 개체라고 예측한 경우 중에서 실제 특정 개체로 판명되어 예측이 일치한 비율}$$

$$\text{재현률} = \frac{TP}{TP+FN} = \text{전체 특정 개체 중에서 실제 특정 개체라고 정답을 맞춘 비율}$$

$$f1\ score = 2 \times \frac{\text{정밀도} \times \text{재현률}}{\text{정밀도} + \text{재현률}}$$

12-05 BiLSTM을 이용한 개체명 인식

양방향 LSTM을 이용한 개체명 인식 - F1-score

```
labels = ['B-PER', 'I-PER', 'O', 'O', 'B-MISC', 'O','O','O','O','O','O','O','O','O','O','O','B-PER','I-PE
R','O','O','O','O','O','O','B-MISC','I-MISC','I-MISC','O','O','O','O','O','O','B-PER','I-PE
R','O','O','O','O','O']
predicted = ['B-PER', 'I-PER', 'O', 'O', 'B-MISC', 'O','O','O','O','O','O','O','O','O','O','O','B-PER','I-PE
R','O','O','O','O','O','O','O','O','O','O','O','O','O','O','O','O','O','O','O','O','O','O','O','O','O']

print(classification_report([labels], [predicted]))
```

```
              precision    recall  f1-score   support

        MISC       1.00      0.50      0.67         2
         PER       1.00      0.67      0.80         3

   micro avg       1.00      0.60      0.75         5
   macro avg       1.00      0.58      0.73         5
weighted avg       1.00      0.60      0.75         5
```

12-05 BiLSTM을 이용한 개체명 인식

양방향 LSTM을 이용한 개체명 인식 - F1-score

```
def sequences_to_tag(sequences):
    result = []
    # 전체 시퀀스로부터 시퀀스를 하나씩 꺼낸다.
    for sequence in sequences:
        word_sequence = []
        # 시퀀스로부터 확률 벡터 또는 원-핫 벡터를 하나씩 꺼낸다.
        for pred in sequence:
            # 정수로 변환. 예를 들어 pred가 [0, 0, 1, 0 ,0]라면 1의 인덱스인 2를 리턴한다.
            pred_index = np.argmax(pred)
            # index_to_ner을 사용하여 정수를 태깅 정보로 변환. 'PAD'는 'O'로 변경.
            word_sequence.append(index_to_ner[pred_index].replace("PAD", "O"))
        result.append(word_sequence)
    return result
```

12-06 BiLSTM-CRF를 이용한 개체명 인식

CRF(Conditional Random Field)