딥 러닝을 이용한 자연어 처리 입문

10장 RNN을 이용한 텍스트 분류

발표자 : 김성윤

목차

1. 텍스트 분류 – 개념 & 종류

- 텍스트를 입력 받아 텍스트가 어떤 종류의 범주에 속하는지를 구분하는 작업
- 분류 범주에 따라 **‘이진 분류’** or **‘다중 클래스 분류’**
- 예) 긍·부정 리뷰를 분류하는 **‘감성 분석’**, 텍스트의 의도를 분류하는 **‘의도 분석’**

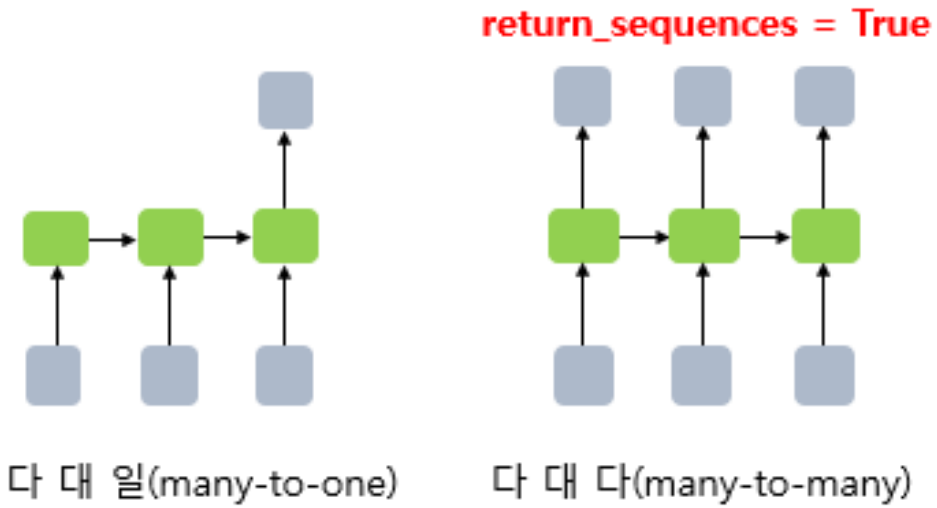
1. 텍스트 분류 – RNN 분류 코드

```
model.add(SimpleRNN(hidden_units, input_shape=(timesteps, input_dim)))
```

- hidden_units = RNN의 출력의 크기 = 은닉 상태의 크기
- timesteps = 시점의 수 = 각 문서에서의 단어 수
- input_dim = 입력의 크기 = 임베딩 벡터의 차원

1. 텍스트 분류 – RNN 분류 코드

- 텍스트 분류는 RNN의 다-대-일 문제에 속함
- 모든 시점에 대해서 입력을 받지만 최종 시점의 RNN 셀만이 은닉 상태를 출력
- 이후, 출력층으로 가서 활성화 함수를 통해 정답을 고르는 문제



2. 텍스트 분류 실습 – 공통적인 전처리 과정

- 데이터 불러오기
- 데이터 개수 확인
- Null 값 확인, 중복 데이터 제거
- 데이터 비율 확인
- 정수 인코딩
- 단어 등장 빈도 수 확인
- 단어 길이 제한(패딩)

2. 텍스트 분류 실습 – RNN

- 이진 분류 문제이므로 시그모이드 함수 사용
- 예) 스팸 메일 분류

```
from tensorflow.keras.layers import SimpleRNN, Embedding, Dense
from tensorflow.keras.models import Sequential

embedding_dim = 32
hidden_units = 32

model = Sequential()
model.add(Embedding(vocab_size, embedding_dim))
model.add(SimpleRNN(hidden_units))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
history = model.fit(X_train_padded, y_train, epochs=4, batch_size=64, validation_split=0.2)
```

2. 텍스트 분류 실습 – LSTM 사용

- 다중 클래스 분류 문제이므로 소프트맥스 함수 사용
- 예) 로이터 뉴스 분류, 네이버 영화 리뷰 감성 분류

```
embedding_dim = 128
hidden_units = 128
num_classes = 46

model = Sequential()
model.add(Embedding(vocab_size, embedding_dim))
model.add(LSTM(hidden_units))
model.add(Dense(num_classes, activation='softmax'))

es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=4)
mc = ModelCheckpoint('best_model.h5', monitor='val_acc', mode='max', verbose=1, save_best_only=True)

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['acc'])
history = model.fit(X_train, y_train, batch_size=128, epochs=30, callbacks=[es, mc], validation_data=(X_test, y_tes
t))
```

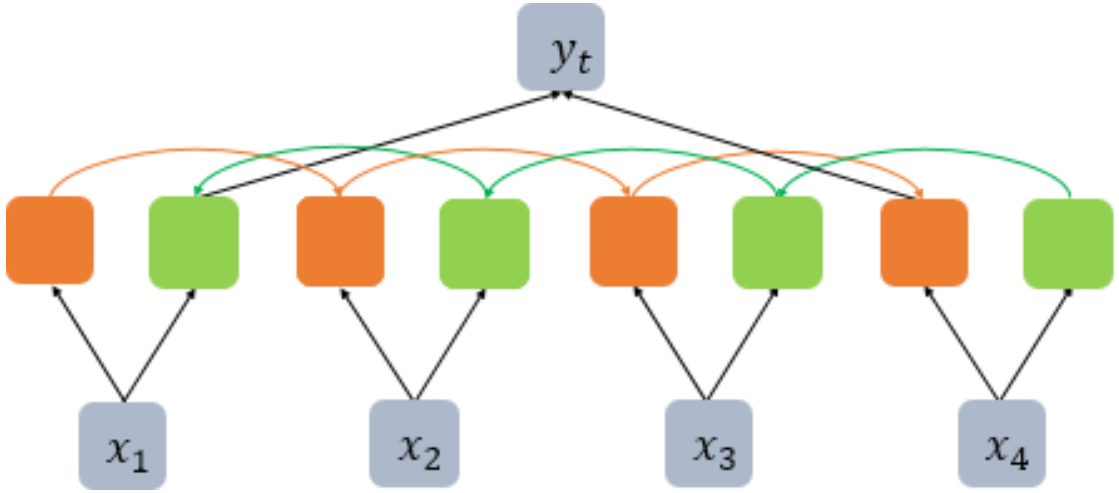
2. 텍스트 분류 실습 – GRU

- 이진 분류 문제이므로 시그모이드 함수 사용
- 예) IMDB 리뷰 감성 분류, 네이버 쇼핑 리뷰 감성 분류

```
embedding_dim = 100
hidden_units = 128

model = Sequential()
model.add(Embedding(vocab_size, embedding_dim))
model.add(GRU(hidden_units))
model.add(Dense(1, activation='sigmoid'))

es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=4)
mc = ModelCheckpoint('GRU_model.h5', monitor='val_acc', mode='max', verbose=1, save_best_only=True)

model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
history = model.fit(X_train, y_train, epochs=15, callbacks=[es, mc], batch_size=64, validation_split=0.2)
```

2. 텍스트 분류 실습 – BiLSTM

- 두 개의 독립적인 LSTM 아키텍처를 함께 사용하는 구조
- 순방향 LSTM은 마지막 시점의 은닉 상태를 반환
- 역방향 LSTM은 첫번째 시점의 은닉 상태를 반환



2. 텍스트 분류 실습 – BiLSTM

- 예) 한국어 스팀 리뷰 감성 분류

```
embedding_dim = 100
hidden_units = 128

model = Sequential()
model.add(Embedding(vocab_size, embedding_dim))
model.add(Bidirectional(LSTM(hidden_units))) # Bidirectional LSTM을 사용
model.add(Dense(1, activation='sigmoid'))

es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=4)
mc = ModelCheckpoint('best_model.h5', monitor='val_acc', mode='max', verbose=1, save_best_only=True)

model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
history = model.fit(X_train, y_train, epochs=15, callbacks=[es, mc], batch_size=256, validation_split=0.2)
```

3. 나이브 베이즈 분류기 – 베이즈 정리

P(정상 메일 | 입력 텍스트) = P(w_1 | 정상 메일) × P(w_2 | 정상 메일) × P(w_3 | 정상 메일) × P(정상 메일)
P(스팸 메일 | 입력 텍스트) = P(w_1 | 스팸 메일) × P(w_2 | 스팸 메일) × P(w_3 | 스팸 메일) × P(스팸 메일)

- 두 확률 중 더 큰 확률을 따름.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- 베이즈 정리 공식

3. 나이브 베이즈 분류기 – 실습

```
dtmvector = CountVectorizer()
X_train_dtm = dtmvector.fit_transform(newsdata.data)
```

DTM 행렬

```
tfidf_transformer = TfidfTransformer()
tfidfv = tfidf_transformer.fit_transform(X_train_dtm)
```

TF-IDF 행렬

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

```
mod = MultinomialNB()
mod.fit(tfidfv, newsdata.target)
```

사이킷런의 나이브 베이즈 모델 사용
(alpha = 1.0 : 라플라스 스무딩 적용)