훈련과정명	인공지능 자연어처리(NLP) 기업데이터 분석 전문가 양성과정 (A반)
교과목	인공지능 자연어처리 이론 및 실습
실시일	

실기평가 문항

1. 아래 코드는 '이상한 나라의 엘리스' 소설을 읽고 전처리 과정을 수행한 프로그램이다. 비어있는 주석 (comment)란에 적절한 주석 (설명문)을 채워 넣으시오.

```
import nltk
from nltk import pos_tag
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import string

# 소설 alice in wonderland를 읽어온다.
fin = open("alice_in_wonderland.txt", "r")
text = [line for line in fin if len(line) > 0]
fin.close()

# text를 한 라인씩 읽어서 전처리 과정을 수행한다.
pre_proc_text = []
for one_line in text:
    # string.punctuation = '!"#$%&₩'()*+,-./:;<=>?@[₩₩]^_`{|}~' 문자를
    # 공백 문자로 치환한다.
    one_line = "".join([" " if ch in string.punctuation else ch for ch in one_line])

    # text를 토큰으로 분리하고 소문자로 변환한다.
    tokens = [word.lower() for word in nltk.word_tokenize(one_line)]

    # _____(1)_____
    tokens = [tok for tok in tokens if tok not in stopwords.words('english')]

    # _____(2)_____
    tokens = [word for word in tokens if len(word) >= 3]

    # _____(3)_____
    # (cooking --> cook, cookery --> cookeri)
    tokens = [PorterStemmer().stem(word) for word in tokens]

    # _____(4)_____
    tagged_corpus = pos_tag(tokens)
```

실기평가 문항

```
    # WordNetLemmatizer()를 이용하여 단어의 원형을 복원한다.
    # lemmatizer.lemmatize('believes') --> belief
    Noun_tags = ['NN','NNP','NNPS','NNS']
    Verb_tags = ['VB','VBD','VBG','VBN','VBP','VBZ']

    lemmatizer = WordNetLemmatizer()

    # POS tag에 따라 단어의 원형 복원이 달라진다.
    # ('cooking', 'n') --> cooking, ('cooking', 'v') --> cook
    def prat_lemmatize(token, tag):
        if tag in Noun_tags:
            return lemmatizer.lemmatize(token,'n')
        elif tag in Verb_tags:
            return lemmatizer.lemmatize(token,'v')
        else:
            return lemmatizer.lemmatize(token,'n')

    # 전처리가 완료된 문장을 다시 문서로 합친다.
    pre_proc_text.append(" ".join([prat_lemmatize(token,tag) for token,tag in tagge
d_corpus]))
```

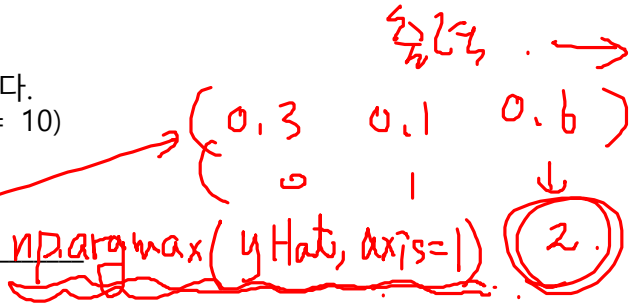
실기평가 문항

2. 아래 코드는 9개 문장으로 구성된 문서 (x_train)를 3개의 카테고리로 (y_train) 분류하는 프로그램이다. 문서를 TFIDF로 변환한 후 딥러닝으로 분류한다. 빈칸을 채워 프로그램을 완성해 보시오.

```
from tensorflow.keras.layers import Input, Dense, Dropout
from tensorflow.keras.models import Model
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np

x_train = ['i open my eyes in the darkness',
           'the sound of my beating heart feel unfamiliar',
           'i face you in the mirror',
           'an afraid of look a long pending question',
           'perhaps, then loving someone else',
           'it is more difficult to love myself',
           'nohestly, let admit what we have to admit',
           'that your standards are more strict',
           'when they are applied to you']

# 3개 카테고리로 분류된 label을 아래와 같이 임의로 설정했다.
y_train = np.array([0, 1, 2, 0, 1, 2, 0, 1, 2]).reshape(9, 1)

# x_train 문서를 TFIDF로 변환한다.
tf_vector = TfidfVectorizer(max_features = 8)
tfidf =_____(1)_____

# 딥러닝 모델을 생성한다.
xInput = Input(batch_shape = (None, 8))      # 입력으로 TFIDF를 받는다.
h = Dense(32, activation = 'relu')(xInput)
h = Dropout(0.5)(h)
yOutput = Dense(3, activation='softmax')(h)  # 출력으로 y_train이 나오게 한다.

model = Model(xInput, yOutput)
model.compile(loss=_____(2)_____, optimizer='adam')
```

실기평가 문항

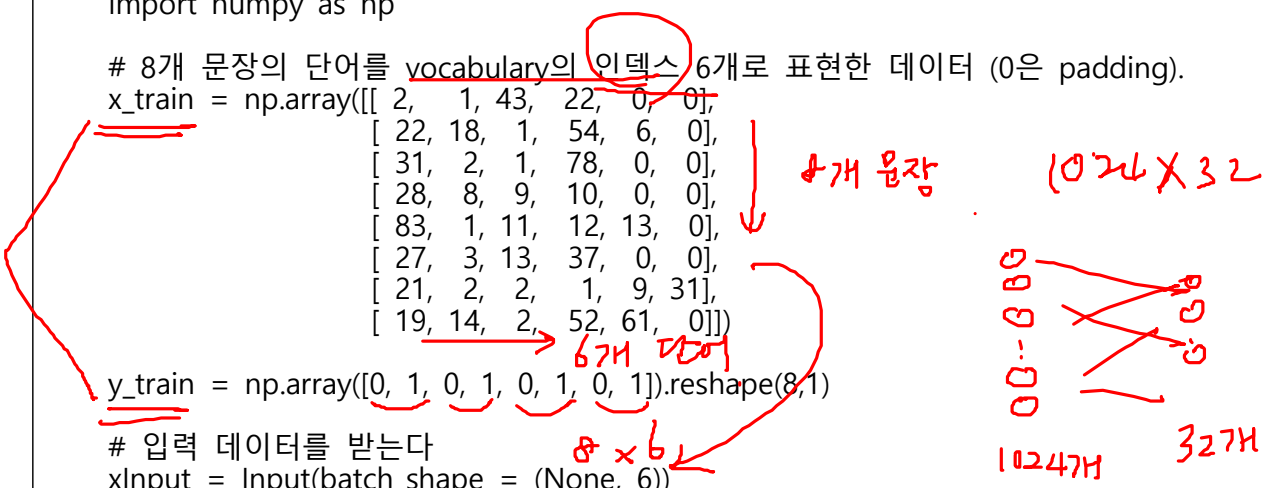
```
# 3개 카테고리로 분류하기 위해 학습한다.
model.fit(tfidf.toarray(), y_train, epochs = 10)

# train 데이터로 정확도를 측정한다.
yHat = model.predict(tfidf.toarray())
y_pred = _____(3)_____
y_pred = y_pred.reshape(9, 1)
accuracy = (y_pred == y_train).mean()
print('정확도 = ', accuracy)
```



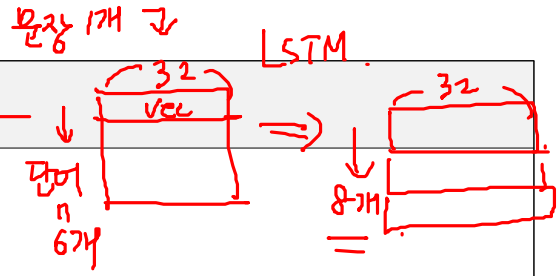
실기평가 문항

3. 아래 코드는 문서를 분류 (이진분류)하는 Keras 코드를 간단히 표현한 것이다. model.summary() 결과의 빈 칸을 채워 보시오.

```
from tensorflow.keras.layers import Input, Embedding, Dropout, LSTM, Dense
from tensorflow.keras.models import Model
import numpy as np

# 8개 문장의 단어를 vocabulary의 인덱스 6개로 표현한 데이터 (0은 padding).
x_train = np.array([[ 2,  1, 43, 22,  0,  0],
                    [ 22, 18,  1, 54,  6,  0],
                    [ 31,  2,  1, 78,  0,  0],
                    [ 28,  8,  9, 10,  0,  0],
                    [ 83,  1, 11, 12, 13,  0],
                    [ 27,  3, 13, 37,  0,  0],
                    [ 21,  2,  2,  1,  9, 31],
                    [ 19, 14,  2, 52, 61,  0]])
y_train = np.array([0, 1, 0, 1, 0, 1, 0, 1]).reshape(8,1)

# 입력 데이터를 받는다
xInput = Input(batch_shape = (None, 6))

# 워드 임베딩 레이어. vocabulary 크기 = 1024, 워드 벡터 = 32 개
embedding = Embedding(1024, 32)(xInput)
embedding = Dropout(0.5)(embedding)

# 워드 임베딩 레이어의 출력 값을 LSTM에 입력한다. (hidden 뉴런 개수 = 32)
xLstm = LSTM(32)(embedding)

# LSTM의 hidden 출력을 받아 1개의 값으로 출력한다. y_train이 학습되도록 한다.
xOutput = Dense(1, activation='sigmoid')(xLstm)

# 모델을 만든다
model = Model(xInput, xOutput)
model.compile(loss='binary_crossentropy', optimizer='adam')
model.summary()

# 모델을 학습한다.
model.fit(x_train, y_train, epochs=50, batch_size=1)
```



실기평가 문항



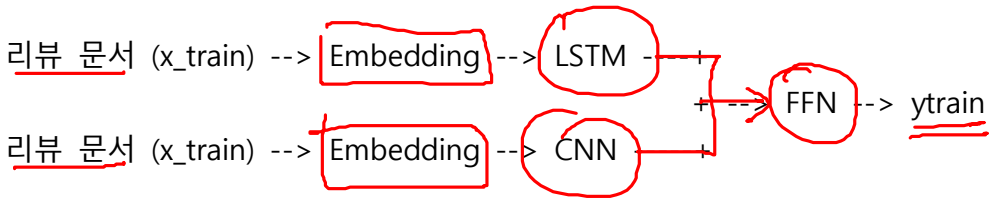
Model: "model"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 6)	0
embedding (Embedding)	(None, 6, 32)	(1024×32)
dropout (Dropout)	(위와 동일)	0
lstm (LSTM)	(None, 32.)	(✓)
dense (Dense)	(None, 1)	33

Total params: 41,121
Trainable params: 41,121
Non-trainable params: 0

실기평가 문항

4. 아래 구조를 갖는 네트워크로 IMDB 데이터의 리뷰(review) (혹은 영화 후기)를 학습 하려고 한다. 빈 칸을 채워 프로그램을 완성하시오.




```
from tensorflow.keras.preprocessing import sequence
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Embedding, Concatenate, LSTM, Conv1D
from tensorflow.keras.layers import Input, Dense, GlobalMaxPooling1D
from tensorflow.keras.datasets import imdb

# IMDB 데이터를 읽어온다. 리뷰 문서인 x_train에는 vocabulary의 단어 인덱스가
# 들어있고, y_train은 0, 1 (0 : positive, 1: negative)이 들어있다.
# 예시 : x_train[0] = [1, 14, 22, 16, 43, 530, 973, ...], y_train[0] = 1
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=6000)

# 1개 리뷰 문서의 단어 개수를 400으로 맞춘다. 400개 보다 작으면
# padding = 0을 추가하고, 400개 보다 크면 뒷 부분을 자른다.
x_train = sequence.pad_sequences(x_train, maxlen=400)

# x_train을 Embedding + LSTM 네트워크로 입력한다.
lstmInput = Input(batch_shape = (None, 400))
lstmEmb = Embedding(6000, 60)(lstmInput)
lstmOut = LSTM(64)(lstmEmb)                    # lstmEmb의 shape = (None, 64)

# x_train을 Embedding + CNN 네트워크로 입력한다.
# Embedding layer는 LSTM과 별도로 사용한다.
cnnInput = Input(batch_shape = (None, 400))
cnnEmb = Embedding(6000, 60)(cnnInput)
convOut = Conv1D(260, 3, activation='relu', strides=1)(cnnEmb)
convOut = GlobalMaxPooling1D()(convOut)  # convOut의 shape = (None, 260)

# LSTM 출력과 CNN 출력을 합친다 (concatenate). LSTM의 출력이 64이므로,
# CNN의 출력도 64로 맞춘다. 합친 결과의 길이는 128이다.
convOut = _____(1)_____
concatOut = _____(2)_____

# 합쳐진 결과를 1개로 줄여서 (linear projection), y_train이 나오도록 한다.
yOutput = Dense(1, activation = 'sigmoid')(concatOut)

# 모델을 생성한다.
model = Model(_____(3)_____)
model.compile(loss='binary_crossentropy', optimizer='adam')

# 학습한다. batch size = 256, epochs = 1000
hist = model.fit(_____(4)_____)
```

문항별 모범답안

1.

(1) 불용어를 제거한다.
(2) 길이가 3 이하인 단어를 제거한다.
(3) 토큰의 어간을 추출한다. 접미사를 제거한다
(4) 토큰의 품사를 태깅한다.

2.

(1) tf_vector.fit_transform(x_train)

(2) 'sparse_categorical_crossentropy'

(3) np.argmax(yHat,axis=1)

3.

Model: "model"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 6)]	0
embedding (Embedding)	(None, 6, 32)	32768
dropout (Dropout)	(None, 6, 32)	0
lstm (LSTM)	(None, 32)	8320
dense (Dense)	(None, 1)	33

Total params: 41,121
Trainable params: 41,121
Non-trainable params: 0

4.

(1) convOut = Dense(64)(convOut)
(2) concatOut = Concatenate()([lstmOut, convOut])
(3) model = Model([lstmInput, cnnInput], yOutput)
(4) hist = model.fit([x_train, x_train], y_train, batch_size=256, epochs=1000)