МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

кафедра систем штучного інтелекту



**ЗВІТ**

про виконання лабораторної роботи №5

з курсу «Проектування систем глибинного навчання»

на тему LSTM мережі для задачкласифікації з використаннямKeras»

Виконав:

*ст. групи КНСШ-12*

*Карпінський Р.М*

Перевірив:

*Пелешко Д.Д*

ЛЬВІВ – 2021

**Мета:** Виконати задані завдання за темою LSTM мережі для задачкласифікації з використаннямKeras.

**Завдання.**

* Вивчити структуру LSTM та принцип побудови мережі за допомогою Keras.
* Розібрати use case 1, провести експериментальні дослідження при різних параметрах мережі.
* Розібрати use case 2, проаналізувати три випадки генерації мереж з використанням LSTM. Зробити порівняння результатів. Досягти Кращої точності ніж наведено в прикладах, за рахунок переналаштування мереж.
* Модифікувати архітектуру LSTM+CNN додавши до неї Droupout. Перенавчити систему і порівняти результати.

**Виконання роботи:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder

from keras.models import Model

from keras.layers import LSTM, Activation, Dense, Dropout, Input, Embedding

from tensorflow.keras.optimizers import RMSprop

from keras.preprocessing.text import Tokenizer

from keras.preprocessing import sequence

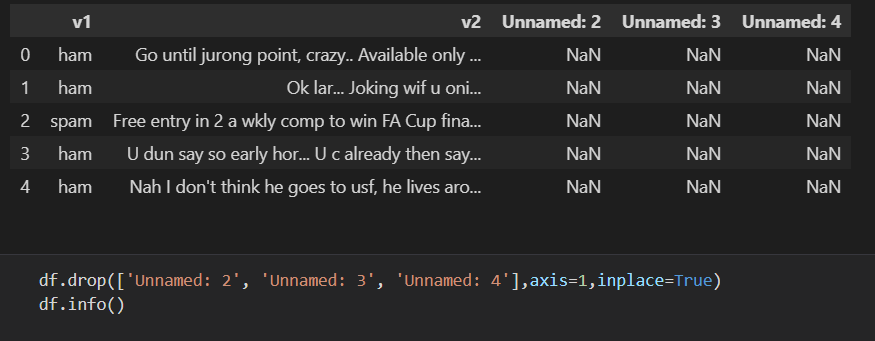
from tensorflow.keras.utils import to\_categorical

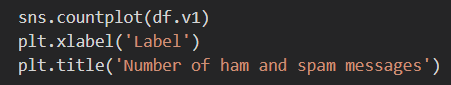
from keras.callbacks import EarlyStopping

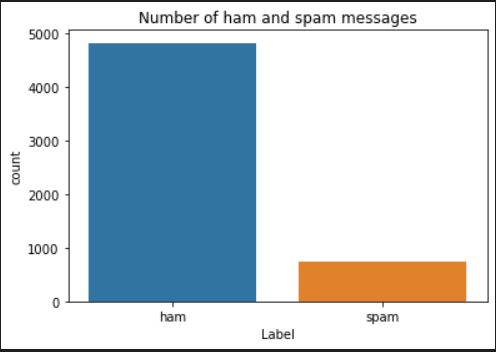
%matplotlib inline

df = pd.read\_csv('spam.csv',delimiter=',',encoding='latin-1')

df.head()







X = df.v2

Y = df.v1

le = LabelEncoder()

Y = le.fit\_transform(Y)

Y = Y.reshape(-1,1)

X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(X,Y,test\_size=0.15)

max\_words = 1000

max\_len = 150

tok = Tokenizer(num\_words=max\_words)

tok.fit\_on\_texts(X\_train)

sequences = tok.texts\_to\_sequences(X\_train)

sequences\_matrix = sequence.pad\_sequences(sequences,maxlen=max\_len)

def RNN():

    inputs = Input(name='inputs',shape=[max\_len])

    layer = Embedding(max\_words,50,input\_length=max\_len)(inputs)

    layer = LSTM(64)(layer)

    layer = Dense(256,name='FC1')(layer)

    layer = Activation('relu')(layer)

    layer = Dropout(0.5)(layer)

    layer = Dense(1,name='out\_layer')(layer)

    layer = Activation('sigmoid')(layer)

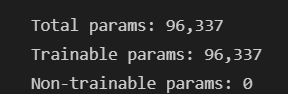
    model = Model(inputs=inputs,outputs=layer)

    return model

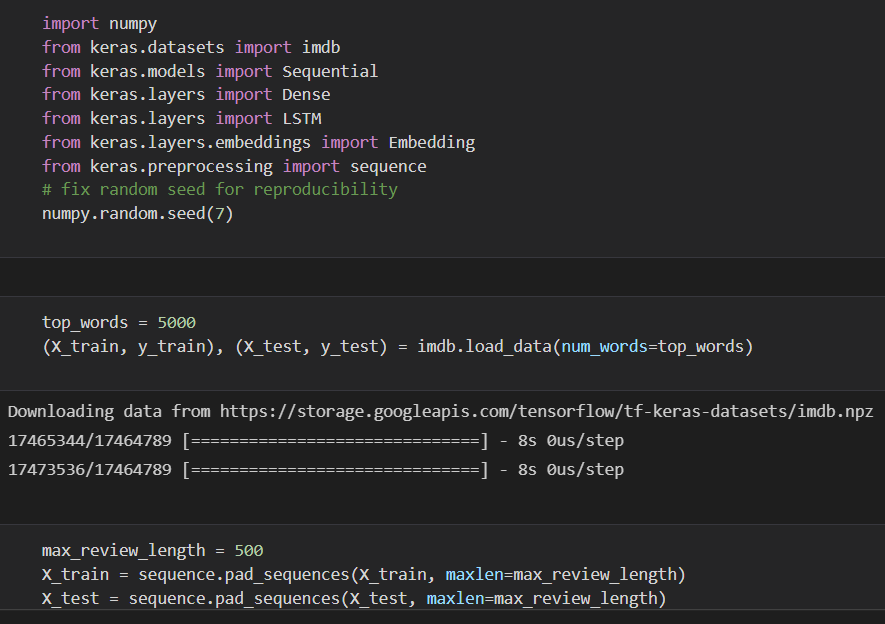
model = RNN()

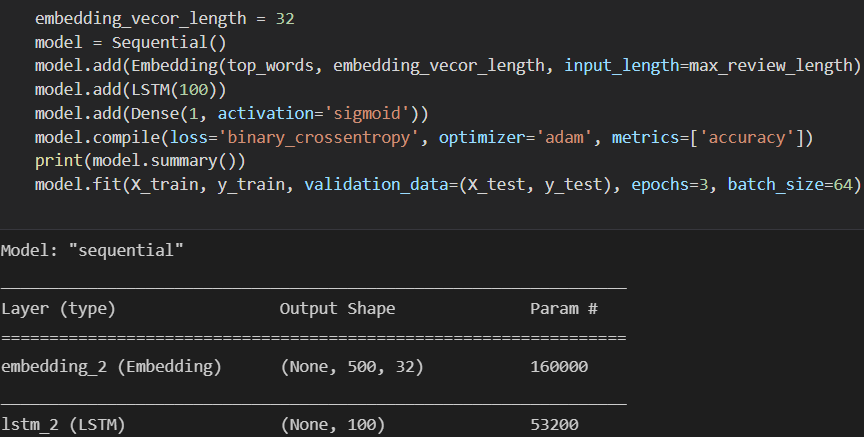
model.summary()

model.compile(loss='binary\_crossentropy',optimizer=RMSprop(),metrics=['accuracy'])



print('Test set\n Loss: {:0.3f}\n Accuracy: {:0.3f}'.format(accr[0],accr[1]))





import numpy

from keras.datasets import imdb

from keras.models import Sequential

from keras.layers import Dense

from keras.layers import LSTM

from keras.layers import Dropout

from keras.layers.embeddings import Embedding

from keras.preprocessing import sequence

# fix random seed for reproducibility

numpy.random.seed(7)

# load the dataset but only keep the top n words, zero the rest

top\_words = 5000

(X\_train, y\_train), (X\_test, y\_test) = imdb.load\_data(num\_words=top\_words) # truncate and pad input sequences

max\_review\_length = 500

X\_train = sequence.pad\_sequences(X\_train, maxlen=max\_review\_length)

X\_test = sequence.pad\_sequences(X\_test, maxlen=max\_review\_length)

# create the model

embedding\_vecor\_length = 32

model = Sequential()

model.add(Embedding(top\_words, embedding\_vecor\_length, input\_length=max\_review\_length))

model.add(Dropout(0.2))

model.add(LSTM(100))

model.add(Dropout(0.2))

model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])

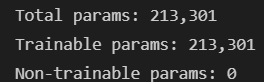
print(model.summary())

model.fit(X\_train, y\_train, epochs=3, batch\_size=64)

# Final evaluation of the model

scores = model.evaluate(X\_test, y\_test, verbose=0)

print("Accuracy: %.2f%%" % (scores[1]\*100))



model = Sequential()

model.add(Embedding(top\_words, embedding\_vecor\_length, input\_length=max\_review\_length))

model.add(LSTM(100, dropout=0.2, recurrent\_dropout=0.2))

model.add(Dense(1, activation='sigmoid'))

import numpy

from keras.datasets import imdb

from keras.models import Sequential

from keras.layers import Dense

from keras.layers import LSTM

from keras.layers.embeddings import Embedding

from keras.preprocessing import sequence

# fix random seed for reproducibility

numpy.random.seed(7)

# load the dataset but only keep the top n words, zero the rest

top\_words = 5000

(X\_train, y\_train), (X\_test, y\_test) = imdb.load\_data(num\_words=top\_words) # truncate and pad input sequences

max\_review\_length = 500

X\_train = sequence.pad\_sequences(X\_train, maxlen=max\_review\_length)

X\_test = sequence.pad\_sequences(X\_test, maxlen=max\_review\_length)

# create the model

embedding\_vecor\_length = 32

model = Sequential()

model.add(Embedding(top\_words, embedding\_vecor\_length, input\_length=max\_review\_length))

model.add(LSTM(100, dropout=0.2, recurrent\_dropout=0.2))

model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])

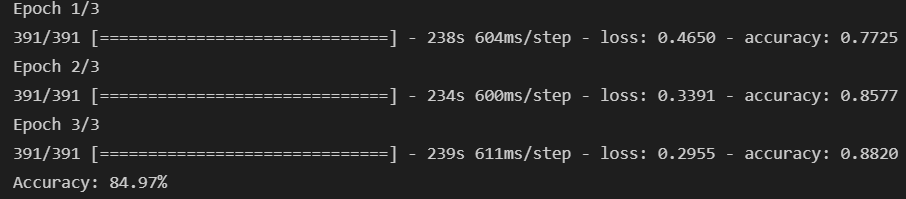
print(model.summary())

model.fit(X\_train, y\_train, epochs=3, batch\_size=64)

# Final evaluation of the model

scores = model.evaluate(X\_test, y\_test, verbose=0)

print("Accuracy: %.2f%%" % (scores[1]\*100))



import numpy

from keras.datasets import imdb

from keras.models import Sequential

from keras.layers import Dense

from keras.layers import LSTM

from keras.layers.convolutional import Conv1D

from keras.layers.convolutional import MaxPooling1D

from keras.layers.embeddings import Embedding

from keras.preprocessing import sequence

# fix random seed for reproducibility

numpy.random.seed(7)

# load the dataset but only keep the top n words, zero the rest

top\_words = 5000

(X\_train, y\_train), (X\_test, y\_test) = imdb.load\_data(num\_words=top\_words) # truncate and pad input sequences

max\_review\_length = 500

X\_train = sequence.pad\_sequences(X\_train, maxlen=max\_review\_length)

X\_test = sequence.pad\_sequences(X\_test, maxlen=max\_review\_length)

# create the model

embedding\_vecor\_length = 32

model = Sequential()

model.add(Embedding(top\_words, embedding\_vecor\_length, input\_length=max\_review\_length))

model.add(Conv1D(filters=32, kernel\_size=3, padding='same', activation='relu'))

model.add(MaxPooling1D(pool\_size=2))

model.add(LSTM(100))

model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])

print(model.summary())

model.fit(X\_train, y\_train, epochs=3, batch\_size=64)

# Final evaluation of the model

scores = model.evaluate(X\_test, y\_test, verbose=0)

print("Accuracy: %.2f%%" % (scores[1]\*100))

**Висновок:** На даній лабораторній роботі, виконав поставлені завдання а саме: вивчив структуру LSTM та принцип побудови мереж, розібрав use case 1, провести експериментальні дослідження, розібрав use case 2, проаналізувати три випадки генерації мереж з використанням LSTM, модифікував архітектуру LSTM+CNN додавши до неї Droupout