Week12

Created By	DongGu Kim
Last Edited	@Jun 05, 2020 11:29 AM
Property	
Tags	

LSTM 모델링

SVM, LGBM, LSTM 모델 성능 비교

AWS EC2 웹 배포 [진행 중]

---

1. LSTM 모델링

1. Tokenizer (텍스트를 정수형 데이터타입으로)

```
from keras.preprocessing.text import Tokenizer

# tokenizer
max_words = 10000
tokenizer = Tokenizer(num_words = max_words)
tokenizer.fit_on_texts(X_train)
X_train = tokenizer.texts_to_sequences(X_train)
X_test = tokenizer.texts_to_sequences(X_test)


# 텍스트 데이터 특징 확인
import matplotlib.pyplot as plt

print("문자의 최대 길이 :" , max(len(l) for l in X_train))
print("문자의 평균 길이 : ", sum(map(len, X_train))/ len(X_train))
plt.hist([len(s) for s in X_train], bins=50)
plt.xlabel('length of Data')
plt.ylabel('number of Data')
plt.show()

print("문자의 최대 길이 : ", max(len(l) for l in X_test))
print("문자의 평균 길이 : ", sum(map(len, X_test))/ len(X_test))
plt.hist([len(s) for s in X_test], bins=50)
plt.xlabel('length of Data')
plt.ylabel('number of Data')
plt.show()
```

2. 문자메시지 최대길이 설정

```
from keras.layers import Embedding, Dense, LSTM
from keras.models import Sequential
from keras.preprocessing.sequence import pad_sequences
```

```
max_len = 264

X_train = pad_sequences(X_train, maxlen=max_len)
X_test = pad_sequences(X_test, maxlen=max_len)
```

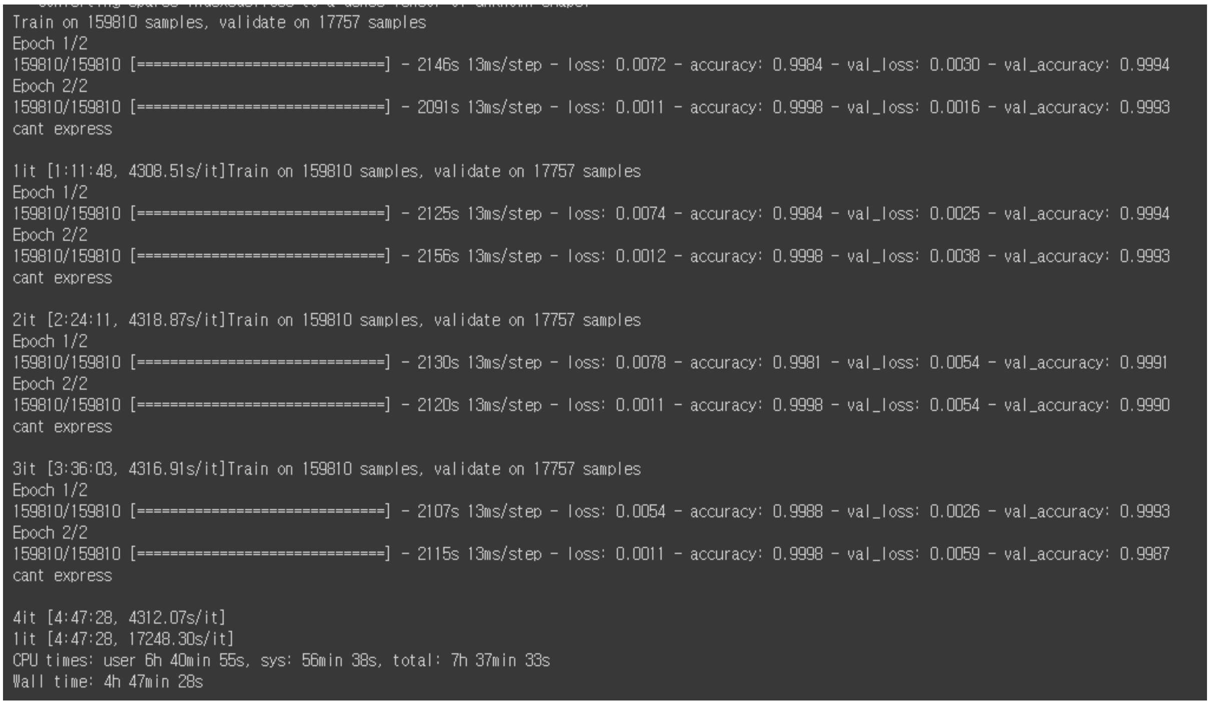
3. class label one hot encoding

- smishing: 0 1
- normal : 1 0

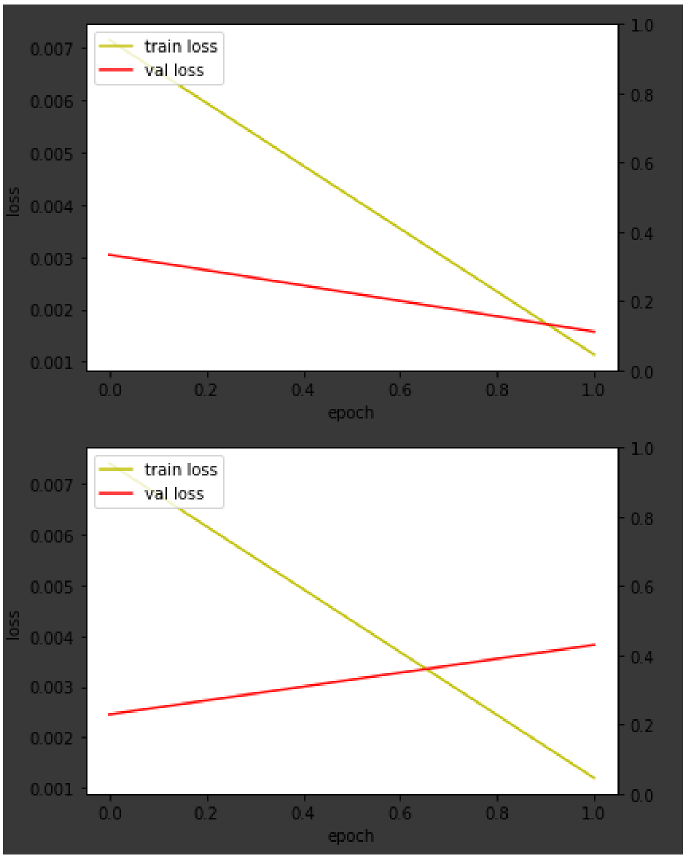
```
YY_train = pd.DataFrame(Y_train)
YY_train['smishing0'] = 1

for idx in YY_train.index:
    if YY_train['smishing'][idx] == 1:
        YY_train['smishing0'][idx] = 0

YY_train = YY_train[['smishing0', 'smishing']]
```

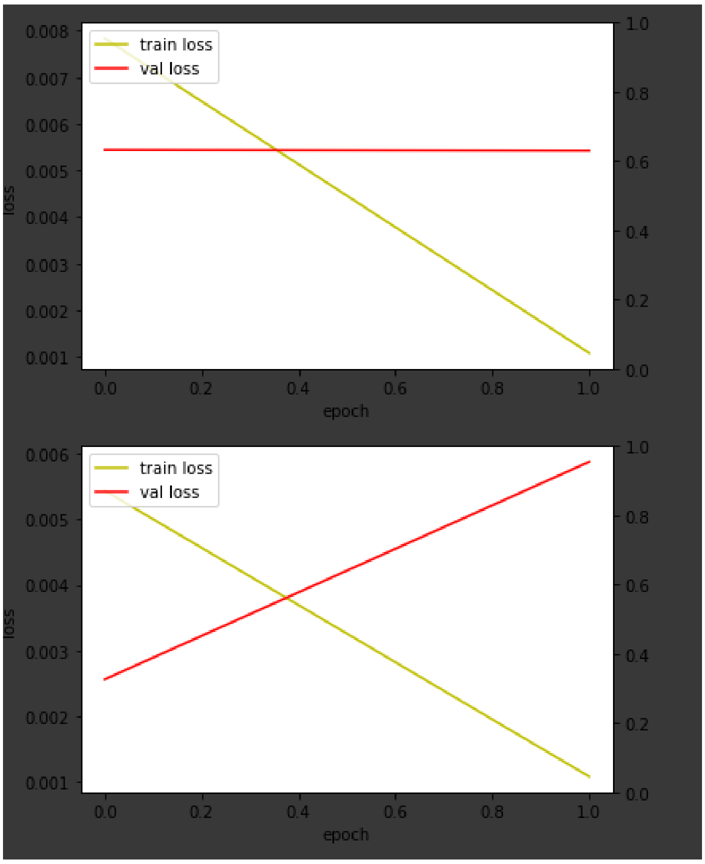
4. LSTM modeling (K-Fold 교차검증 이용)

```
%%time
# 4FOLD, 3SEED ENSEMBLE
# 총 12개의 모델을 평균내어 예측한다

lucky_seed=[1996, 8, 25]


# enumerate: 인덱스와 값을 둘다 반복시킬 때 사용
for num,rs in tqdm(enumerate(lucky_seed)):

    kfold = KFold(n_splits=4, random_state = rs, shuffle = True)

    # numpy.zeros((row,col))
    # row*col size 영행렬 생성
    # train.shape[0],198 -> trainset 41400개, target값:198개
    cv=np.zeros((len(X_train),2))

    for n, (train_idx, validation_idx) in tqdm(enumerate(kfold.split(X_train))):

        x_train, x_validation = X_train[train_idx], X_train[validation_idx]
        y_train, y_validation = YY_train.loc[train_idx], YY_train.loc[validation_idx]

        lstm_model = Sequential()
        lstm_model.add(Embedding(max_words, 100))

        lstm_model.add(LSTM(128))
        lstm_model.add(Dense(2, activation='sigmoid'))
        lstm_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

        try:
            lstm_history = lstm_model.fit(x_train, y_train, epochs=2, batch_size=32, validation_split=0.1)

            fig, loss_ax = plt.subplots()
            acc_ax = loss_ax.twinx()

            loss_ax.plot(lstm_history.history['loss'], 'y', label='train loss')
            loss_ax.plot(lstm_history.history['val_loss'], 'r', label='val loss')
            loss_ax.set_xlabel('epoch')
            loss_ax.set_ylabel('loss')
            loss_ax.legend(loc='upper left')
            acc_ax.plot(lstm_history.history['acc'], 'b', label='train acc')
            acc_ax.plot(lstm_history.history['val_acc'], 'g', label='val acc')
            acc_ax.set_ylabel('accuracy')
            acc_ax.legend(loc='upper left')

            plt.show()
        except:
            print('cant express')

        lstm_model.save("lstm_models/%s_%s_lstm_model.h5"%(n, rs))
        # 모델결과 저장 lib
        try:
            joblib.dump(lstm_model, 'lstm_models2/%s_fold_model_%s.pkl'%(n,rs))
        except:
            pass
        # numpy.zeros((row,col))로 만들어주었던 영행렬: cv
        # data object에 X_validation 예측 값을 넣어줌
        # CROSS-VALIDATION , EVALUATE CV
        try:
            cv[validation_idx,:] = lstm_model.predict(x_validation)
        except:
            pass
```



```
Train on 159810 samples, validate on 17757 samples
Epoch 1/2
159810/159810 [==============================] - 2146s 13ms/step - loss: 0.0072 - accuracy: 0.9984 - val_loss: 0.0030 - val_accuracy: 0.9994
Epoch 2/2
159810/159810 [==============================] - 2091s 13ms/step - loss: 0.0011 - accuracy: 0.9998 - val_loss: 0.0016 - val_accuracy: 0.9993
cant express

1it [1:11:48, 4308.51s/it]Train on 159810 samples, validate on 17757 samples
Epoch 1/2
159810/159810 [==============================] - 2125s 13ms/step - loss: 0.0074 - accuracy: 0.9984 - val_loss: 0.0025 - val_accuracy: 0.9994
Epoch 2/2
159810/159810 [==============================] - 2156s 13ms/step - loss: 0.0012 - accuracy: 0.9998 - val_loss: 0.0038 - val_accuracy: 0.9993
cant express

2it [2:24:11, 4318.87s/it]Train on 159810 samples, validate on 17757 samples
Epoch 1/2
159810/159810 [==============================] - 2130s 13ms/step - loss: 0.0078 - accuracy: 0.9981 - val_loss: 0.0054 - val_accuracy: 0.9991
Epoch 2/2
159810/159810 [==============================] - 2120s 13ms/step - loss: 0.0011 - accuracy: 0.9998 - val_loss: 0.0054 - val_accuracy: 0.9990
cant express

3it [3:36:03, 4316.91s/it]Train on 159810 samples, validate on 17757 samples
Epoch 1/2
159810/159810 [==============================] - 2107s 13ms/step - loss: 0.0054 - accuracy: 0.9988 - val_loss: 0.0026 - val_accuracy: 0.9993
Epoch 2/2
159810/159810 [==============================] - 2115s 13ms/step - loss: 0.0011 - accuracy: 0.9998 - val_loss: 0.0059 - val_accuracy: 0.9987
cant express

4it [4:47:28, 4312.07s/it]
1it [4:47:28, 17248.30s/it]
CPU times: user 6h 40min 55s, sys: 56min 38s, total: 7h 37min 33s
Wall time: 4h 47min 28s
```


train loss
val loss
loss
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.0
0.2
0.4
0.6
0.8
1.0
epoch
train loss
val loss
loss
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.0
0.2
0.4
0.6
0.8
1.0
epoch

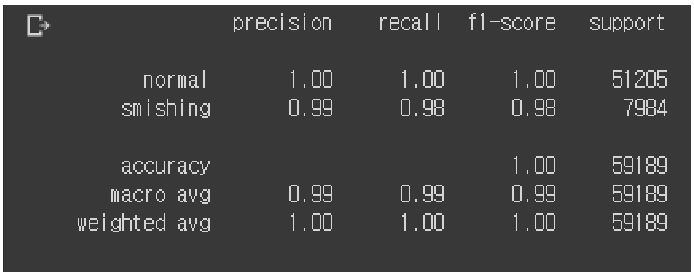

train loss
val loss
loss
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.0
0.2
0.4
0.6
0.8
1.0
epoch
train loss
val loss
loss
0.006
0.005
0.004
0.003
0.002
0.001
0.0
0.2
0.4
0.6
0.8
1.0
epoch

5. LSTM prediction

```
# MODEL LOAD & TEST PREDICT
# 12 MODELS 평균 사용
%%time
models = os.listdir('lstm_models/')
models_list = [x for x in models if x.endswith(".h5")]

from keras.models import load_model
models = os.listdir('lstm_models/')
models_list = [x for x in models if x.endswith(".h5")]

# 모델결과가 잘 나왔는지 check
# assert: 좌항과 우항의 값이 같으면 정상 작동, 다르면 오류 발생
assert len(models_list) ==12


temp_predictions = np.zeros((X_test.shape[0],2))

# 12개 모델을 반복시켜서 결과산출 -> 12로 나눠서 평균값 계산
for model in tqdm(models_list):
    model = load_model('lstm_models/'+model)
    predict_proba = model.predict(X_test)
    temp_predictions += predict_proba/12


submission = pd.DataFrame(data=np.zeros((X_test.shape[0],2)))
submission.index = Y_test.index
submission.index.name = 'id'
submission+=temp_predictions

submission = submission.sort_index()
submission = submission.groupby('id').mean()
submission.set_index('id', inplace = True)
submission.to_csv('lstm_submission.csv', index=True)
```

6. Classification Report

```
from sklearn.metrics import classification_report
result = classification_report(Y_test, submission['pred'], target_names=['normal','smishing'])
print(result)
```



```
              precision    recall  f1-score   support

      normal       1.00      1.00      1.00     51205
    smishing       0.99      0.98      0.98      7984

    accuracy                           1.00     59189
   macro avg       0.99      0.99      0.99     59189
weighted avg       1.00      1.00      1.00     59189
```

2. SVM, LGBM, LSTM 예측 모델 성능 비교

모델 성능 비교

모델	일반	스미싱	모델링	예측 시간	False Record
SVM	1.00	0.99	15초	1초	80개
LGBM	1.00	0.99	3분	3초	202개
LSTM	1.00	0.98	72분	1분	283개

모델 별 False Record 비교

Name	False Record	False positive	False negative
SVM	80개	1개	79
LGBM	202개	6개	196개
LSTM	283개	117개	166개

모델간 False Record 비교

비교 모델	갯수	id
SVM, LGBM, LSTM 모두 틀린 Record	37	[256005, 287239, 257551, 292368, 275522, 255064, 259673, 282717, 275060, 287888, 258711, 293527, 257704, 259761, 260286, 283839, 255179, 280781, 258259, 259805, 250099, 255764, 258851, 270633, 257893, 259432, 289651, 258933, 253852, 258462, 257966, 257463, 260036, 258012, 256990, 242668, 255484]
SVM, LGBM 모두 틀린 Record	38	[256005, 287239, 257551, 292368, 275522, 255064, 259673, 282717, 275060, 287888, 258711, 293527, 257704, 259761, 260286, 283839, 255179, 280781, 258259, 259805, 250099, 255764, 258851, 270633, 257893, 259432, 289651, 258933, 253852, 258462, 257966, 243634, 257463, 260036, 258012, 256990, 242668, 255484]
SVM, LSTM 모두 틀린 Record	68	[256005, 287239, 275982, 257551, 292368, 283677, 280609, 272932, 255546, 255040, 275522, 255064, 259673, 282717, 275060, 287888, 258711, 293527, 257704, 259761, 260286, 283839, 255168, 255179, 280781, 258259, 259805, 266982, 260332, 260335, 250099, 256760, 288523, 255764, 257813, 258851, 270633, 259884, 256819, 262471, 256338, 258386, 259932, 290652, 257893, 259432, 289651, 258933, 255353, 259985, 255894, 260502, 253852, 258462, 257966, 267695, 261553, 257463, 260036, 261578, 258012, 256990, 242668, 279533, 257519, 270326, 288757, 255484]

> 💡 **SVM이 헷갈려하는 Record → LGBM or LSTM 어떤 것으로 다시 예측할 지**

3. Amzaon EC2 웹 배포 [진행 중]

[완료]

- 계정 생성 및 EC2 인스턴스 생성
- virtualbox를 이용한 ubuntu server 기본환경 설정
- github를 이용한 소스코드 복사
- 가상환경 설치 및 필요 라이브러리 설치

[진행 중]

- nginx를 이용한 퍼블릭 배포