HAM / SPAM CLASSIFICATION USING Bidirectional LSTM

- Junheui Lee (2021020357)

< requirements for project >

For analysis) tensorflow, keras, sklearn

For preprocessing) pandas, numpy, random, nltk

For visualization) plotly, matplotlib, seaborn, wordcloud

1. Introduction

특정한 메시지가 메일함에 도착했을 때, 메시지가 스팸으로 분류될 경우 스팸 메일함으로 보내지게 됩니다. 저는 어떠한 메시지가 스팸인지 아닌지 한 학기 동안 배운 딥러닝으로 잘 구분할 수 있을지에 대한 궁금증이 생겨, UCI Machine Learning Repository에 있는 SMS Spam Collection Dataset을 download 받아 분석을 진행했습니다.

2. EDA, Visualization, Preprocessing

```
spam[spam.duplicated()] ## 중복 데이터
```

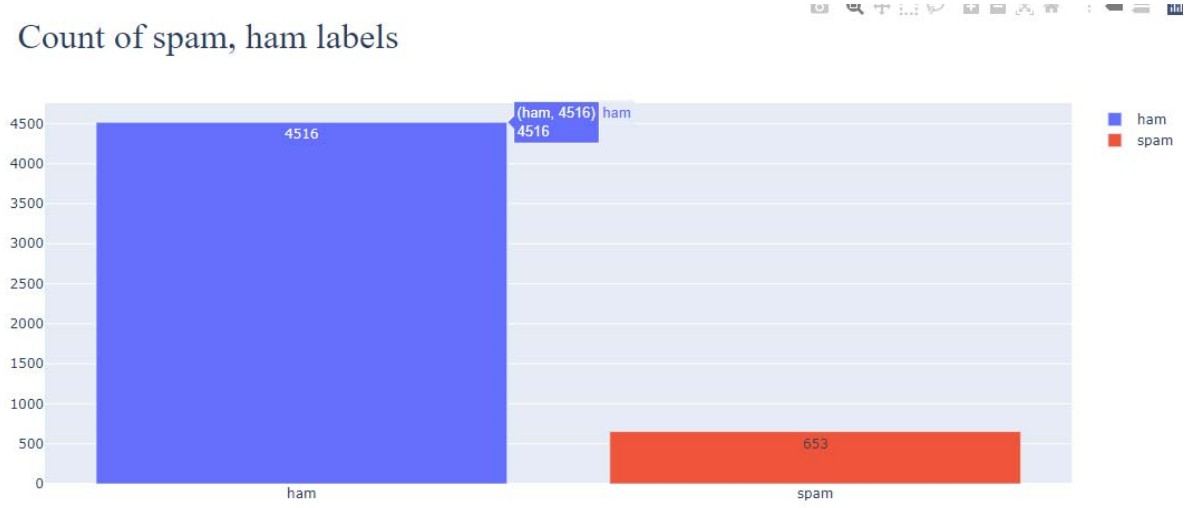
	label	message
102	ham	As per your request 'Melle Melle (Oru Minnamin...
153	ham	As per your request 'Melle Melle (Oru Minnamin...
206	ham	As I entered my cabin my PA said,'' Happy B'd...
222	ham	Sorry, I'll call later
325	ham	No calls..messages..missed calls
...	...	...
5524	spam	You are awarded a SiPix Digital Camera! call 0...
5535	ham	I know you are thinkin malaria. But relax, chi...
5539	ham	Just sleeping..and surfing
5553	ham	Hahaha..use your brain dear
5558	ham	Sorry, I'll call later

403 rows × 2 columns

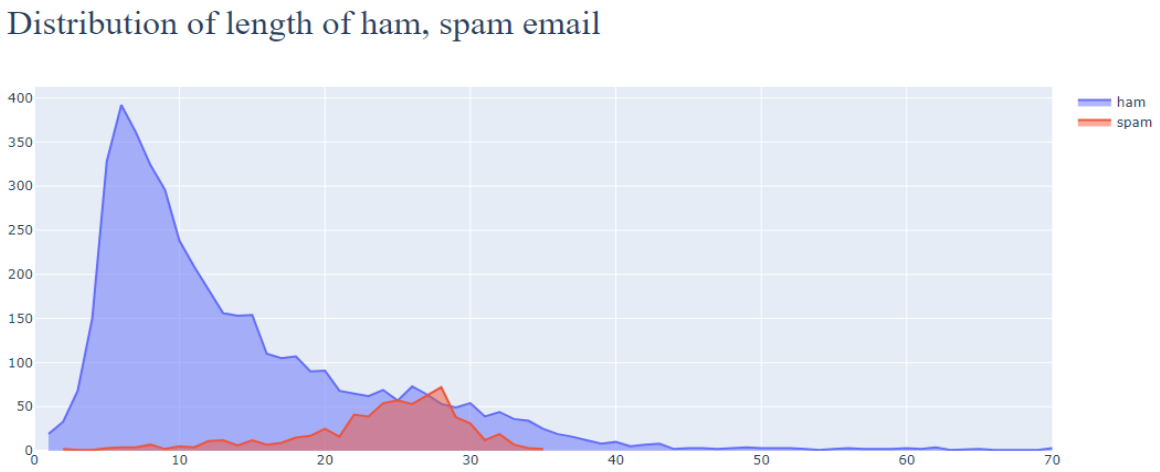
spam data는 5572개의 메시지가 있었고, 각각의 메시지는 spam인지 아닌지에 대한 label이 달려 있습니다. 그 중 403개의 메시지가 동일한 내용의 메시지임을 확인해, 중복되는 메시지를 제거했습니다.

```
spam.groupby('label').describe().T
```

label		ham	spam
message	count	4516	653
	unique	4516	653
	top	I'm leaving my house now...	Ur ringtone service has changed! 25 Free credi...
	freq	1	1



중복 메시지를 제거한 결과 전체 데이터는 5572개에서 5169개가 되었고, HAM인 데이터는 4516개, SPAM인 데이터는 653개임을 확인했습니다.



SPAM인 메시지는 HAM인 메시지에 비해 비교적 긴 길이를 가지는 것을 알 수 있었습니다. 이를 통해 SPAM인 메시지와 아닌 메시지는 명확한 차이를 가지고 있는 것을 알 수 있었습니다.

```
spam['message']
```

```
0       Go until jurong point, crazy.. Available only ...
1                           Ok lar... Joking wif u oni...
2       Free entry in 2 a wkly comp to win FA Cup fina...
3       U dun say so early hor... U c already then say...
4       Nah I don't think he goes to usf, he lives aro...
                              ...
5567    This is the 2nd time we have tried 2 contact u...
5568                 Will Ì_ b going to esplanade fr home?
5569    Pity, * was in mood for that. So...any other s...
5570    The guy did some bitching but I acted like i'd...
5571                           Rofl. Its true to its name
Name: message, Length: 5169, dtype: object
```

SPAM data의 메시지에는 분석에 불필요한 요소들이 많이 포함되어 있는 것을 확인할 수 있었습니다. 그래서 핵심적인 단어들만 추출하여 메시지 분석이 용이하도록 전처리를 진행했습니다.

```
def clean_corpus(corpus):
    # make text to lower text
    corpus = str(corpus).lower()
    # remove text in square brackets
    corpus = re.sub('\[.*?\]', '', corpus)
    # remove homepage type
    corpus = re.sub('https?://\S+|www\.\S+', '', corpus)
    # remove continus '.'
    corpus = re.sub('<.*?>+', '', corpus)
    # remove special characters
    corpus = re.sub('[%s]' % re.escape(string.punctuation), '', corpus)
    # remove enter
    corpus = re.sub('\n', '', corpus)
    # remove words containing digits
    corpus = re.sub('\w*\d\w*', '', corpus)

    return corpus
```

```
stop_words = stopwords.words('english')
my_stop_words = ['u', 'im']
stop_words = stop_words + my_stop_words

def remove_stopwords(corpus):
    corpus = ' '.join(word for word in corpus.split(' ') if word not in stop_words)
    return corpus
```

```
snow_stemmer = nltk.SnowballStemmer("english")

def stemming_corpus(corpus):
    corpus = ' '.join(snow_stemmer.stem(word) for word in corpus.split(' '))
    return corpus
```

```python
def preprocessing_corpus(corpus):
    # Clean puntuation, urls, and so on
    corpus = clean_corpus(corpus)
    # Remove stopwords
    corpus = remove_stopwords(corpus)
    # Stemm all the words in the sentence
    corpus = stemming_corpus(corpus)

    return corpus
```

```python
spam['message_clean'] = spam['message'].apply(preprocessing_corpus)
```

```python
spam.head()
```

	label	message	message_len	message_clean
0	ham	Go until jurong point, crazy.. Available only ...	20	go jurong point crazi avail bugi n great world...
1	ham	Ok lar... Joking wif u oni...	6	ok lar joke wif oni
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	28	free entri wkli comp win fa cup final tkts m...
3	ham	U dun say so early hor... U c already then say...	11	dun say earli hor c alreadi say
4	ham	Nah I don't think he goes to usf, he lives aro...	13	nah dont think goe usf live around though

전처리의 과정은 다음과 같습니다.

우선, 다음과 같은 과정을 통해 분류에 유의하지 않은 문자들을 제거했습니다.

I) 대문자와 소문자가 섞여 있어 소문자로 통일해 주었습니다.

II) 대괄호 안에 있는 문자들을 제거해 주었습니다.

III) https:와 같은 홈페이지를 제거해 주었습니다.

IV) 연속된 '.' 및 특수문자, enter를 제거해 주었습니다.

V) 숫자를 포함한 문자를 제거했습니다.

다음에, 문장에 많이 사용되지만 의미가 없는 stopwords를 제거했습니다. stopwords를 제거함으로써, 메시지의 핵심 문장들을 모델이 더 잘 파악할 수 있도록 했습니다.

마지막으로, stemming을 진행했습니다. 사람들이 사용하는 언어에서 단어는 문법적 요소나 맥락에 따라 다양한 형태로 변화하는 것을 알 수 있습니다. 미래 / 현재 / 과거 시제가 존재하는 것을 예시로 들 수 있습니다. stemming을 진행함으로써 변화된 단어의 원형만을 찾아줄 수 있고, 단어의 원형만 사용해서 전처리 전에 비해 메시지의 복잡함을 줄여주는 메시지 정규화의 효과를 얻을 수 있었습니다.

전처리를 진행한 결과, 전처리를 마친 메시지가 기존의 메시지에 비해 분석에 용이한 형태가 된 것을 message_clean에서 확인할 수 있었습니다.

< Figure : Wordcloud of HAM, SPAM data >

전처리를 완료한 메시지로 wordcloud 시각화를 한 결과, HAM 메시지에서는 일상적인 go, come, ok 등의 단어가 주로 사용되나 SPAM 메시지에서는 free, call, text 등의 상업적인 단어가 사용되는 것을 알 수 있었습니다. 처음 wordcloud 과정에서 im, u가 상위 단어에 올라오는 것을 확인하였고, 이는 stopwords로 분류되는 것이 맞다고 생각해 stopwords에 추가했습니다.

```
from sklearn.preprocessing import LabelEncoder

LE = LabelEncoder()
LE.fit(spam['label'])

spam['label_encoded'] = LE.transform(spam['label'])
```

SPAM, HAM을 분류하는 딥러닝 분석을 시작하기 전에, 모델의 마지막 dense layer에는 sigmoid 활성 함수를 사용한 후 binary cross-entropy 손실 함수를 사용하기 위해 HAM을 0, SPAM을 1로 encoding 했습니다.

3. Modeling

GLOVE Word Embedding

```
texts = spam['message_clean']
target = spam['label_encoded']
```

```
word_tokenizer = Tokenizer() # keras tokenizer
word_tokenizer.fit_on_texts(texts)

vocab_length = len(word_tokenizer.word_index) + 1
vocab_length
```

```
def embed_corpus(corpus):
    return word_tokenizer.texts_to_sequences(corpus)

longest_train = max(texts, key=lambda sentence: len(word_tokenize(sentence)))
maxlen = len(word_tokenize(longest_train))

train_padded_sentences = pad_sequences(
    embed_corpus(texts),
    maxlen,
    padding='post'
)
```

```
embeddings_dictionary = dict()
embedding_dim = 100

with open('glove.6B.100d.txt', encoding='UTF8') as f:
    for line in f.readlines():
        records = line.split()
        word = records[0]
        vector_dimensions = np.asarray(records[1:], dtype='float32')
        embeddings_dictionary[word] = vector_dimensions

f.close()

print(len(embeddings_dictionary))
```

```
embedding_matrix = np.zeros((vocab_length, embedding_dim))

for word, index in word_tokenizer.word_index.items():
    embedding_vector = embeddings_dictionary.get(word)
    if embedding_vector is not None:
        embedding_matrix[index] = embedding_vector
```

전체 corpus에서 단어들의 동시 발생 빈도를 기반으로 단어의 의미를 살려 vector화 시켜주는 GLOVE embedding을 사용하기 위해 메시지를 tokenizing 시킨 후, 400000 개의 단어로 학습된 GLOVE pre-trained weights를 가져와 적용했습니다.

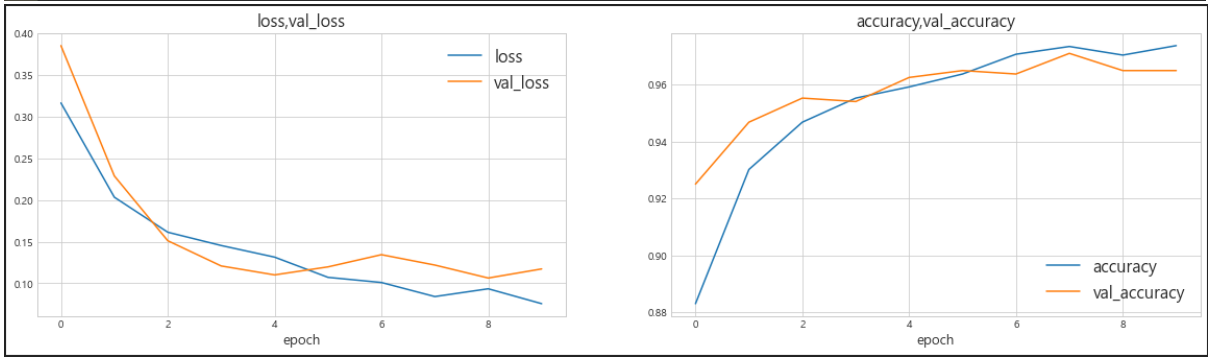
```python
model = Sequential()

model.add(Embedding(
    input_dim=embedding_matrix.shape[0],
    output_dim=embedding_matrix.shape[1],
    weights = [embedding_matrix],
    input_length=maxlen
))

model.add(Bidirectional(LSTM(maxlen, return_sequences = True, recurrent_dropout=0.2)))

model.add(GlobalMaxPool1D())
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(maxlen, activation = "relu"))
model.add(Dropout(0.5))
model.add(Dense(maxlen, activation = "relu"))
model.add(Dropout(0.5))
model.add(Dense(1, activation = 'sigmoid')) # binary classificiation
```

```python
# for freezing glove weights
model.layers[0].set_weights([embedding_matrix])
model.layers[0].trainable=False
```

SPAM / HAM 분류를 위해 Bidirectional LSTM model을 구축했습니다.

과적합을 막기 위해 GlobalMaxPooling, BatchNormaliztion, dropout, recurrent_dropout을 적용했으며, GLOVE weights의 활용을 위해, 해당 weights 부분을 non-trainable하게 설정했습니다. 마지막으로 SPAM / HAM 분류를 위해 sigmoid 활성함수를 사용했습니다.

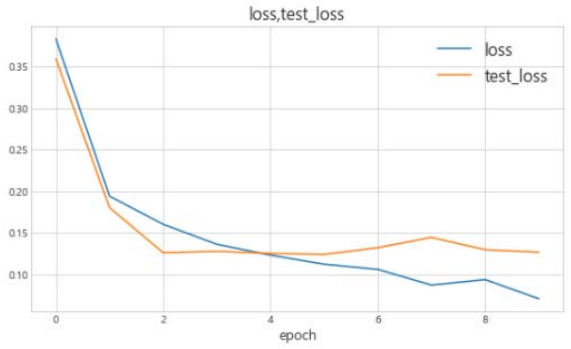
```python
model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])

checkpoint = ModelCheckpoint('model_bidirectional_lstm.h5', monitor = 'val_loss', verbose = 1, save_best_only = True)
reduce_lr = ReduceLROnPlateau( monitor = 'val_loss', factor = 0.2, verbose = 1, patience = 2, min_lr = 0.001)

history = model.fit(
    partial_x_train, partial_y_train,
    epochs = 10, batch_size = 32,
    validation_data = (x_val, y_val),
    verbose = 1,
    callbacks = [reduce_lr, checkpoint]
)
```

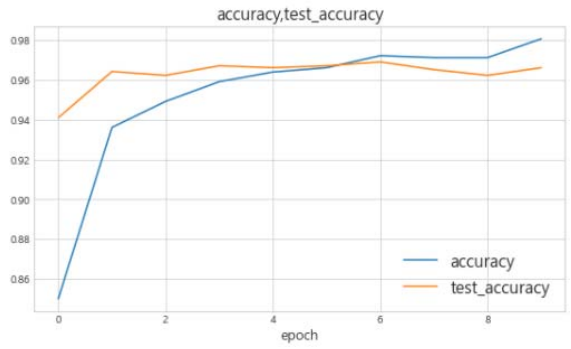


안정적인 결과 복원을 위해 seed를 42로 고정하고 모델링을 진행했습니다. ReducedLROnPlateau scheduler를 통해 validation loss의 개선 정도에 따라 learning rate를 조정하여 최적의 모델 학습이 진행될 수 있도록 했습니다. 10 epoch, 32 batch size를 사용해 모델을 적합했고, learning curve를 통해 epoch 경과에 따라 안정적인 metric의 향상이 있음을 확인할 수 있었습니다.

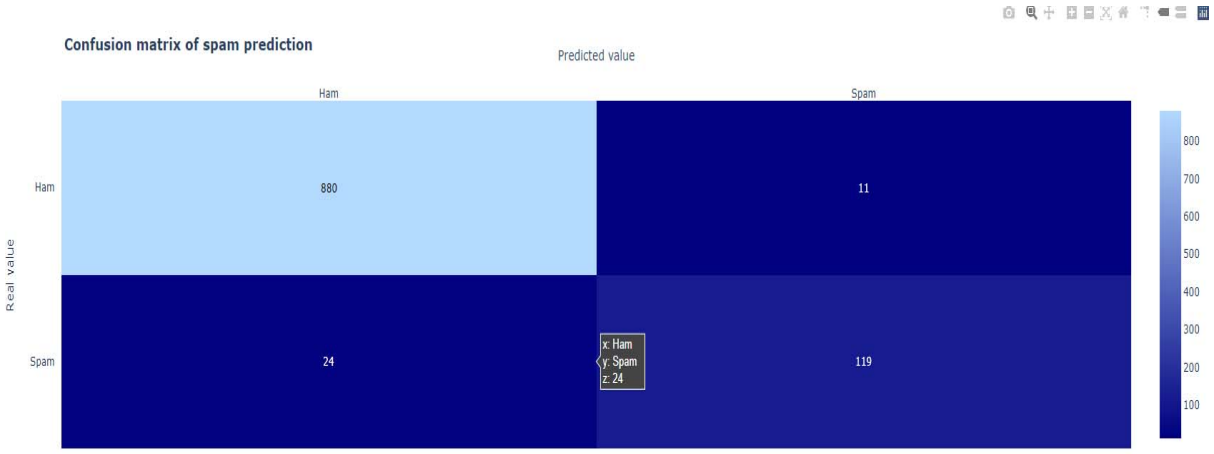
```python
model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])

checkpoint = ModelCheckpoint('model_bidirectional_lstm.h5', monitor = 'val_loss', verbose = 1, save_best_only = True)
reduce_lr = ReduceLROnPlateau( monitor = 'val_loss', factor = 0.2, verbose = 1, patience = 2, min_lr = 0.001)

history = model.fit(
    x_train, y_train,
    epochs = 10, batch_size = 32,
    validation_data = (x_test, y_test),
    verbose = 1,
    callbacks = [reduce_lr, checkpoint]
)
```

```
Epoch 9/10
4135/4135 [==============================] - 18s 4ms/step - loss: 0.0935 - accuracy: 0.9712 - val_loss: 0.1294 - val_accuracy: 0.9623

Epoch 00009: val_loss did not improve from 0.12379
Epoch 10/10
4135/4135 [==============================] - 18s 4ms/step - loss: 0.0704 - accuracy: 0.9807 - val_loss: 0.1264 - val_accuracy: 0.9662

Epoch 00010: ReduceLROnPlateau reducing learning rate to 0.001.

Epoch 00010: val_loss did not improve from 0.12379
```





Validation set을 통해 hyperparameter tuning이 완료된 모델을 앞과 동일한 10 epoch, 32 batch size를 사용하여 전체 train set에 적합하였습니다. 그 결과 training accuracy는 0.9807, test accuracy는 0.9662로 미세한 차이를 보였고, learning curve를 통해 최종 모델에서 overfitting이 완화된 것을 확인할 수 있었습니다.




```
from sklearn.metrics import zero_one_loss
misclassification_rate = zero_one_loss(y_test, y_pred)
misclassification_rate
```

```
0.03384912959381048
```

test set의 design matrix로 SPAM/HAM classification prediction을 한 결과, 모델이 대부분의 classification을 올바르게 한 것을 confusion matrix를 통해 확인할 수 있었습니다. 모델의 misclassification rate는 3.3%로 준수한 성능을 보였습니다.