bilstm_word2vec_copypaste

August 30, 2019

[1]:
```
import math

import re

from bs4 import BeautifulSoup

import numpy as np
import pandas as pd

import nltk
from nltk.corpus import stopwords
from nltk import word_tokenize

import keras
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Embedding, Bidirectional, LSTM, Dense,␣
↪SpatialDropout1D, Dropout
from keras.callbacks import ReduceLROnPlateau

from sklearn.model_selection import train_test_split

import gensim
```

```
Using TensorFlow backend.
```

[2]:
```
labeled_data = pd.read_csv("data/labeledTrainData.tsv", sep="\t", quoting=3)

test_data = pd.read_csv("data/testData.tsv", sep="\t", quoting=3)
```

[3]:
```
display(
    labeled_data.shape, test_data.shape
)
```

```
(25000, 3)
```

```
(25000, 2)
```

[4]:
```
train_reviews = labeled_data['review']
train_sentiments = labeled_data['sentiment']
```

[5]:
```
all_reviews = train_reviews.append( test_data['review'] )
```

[6]:
```
nltk_stopwords_set = set( stopwords.words('english') )

def preprocess_text(text):
    text = BeautifulSoup(text).get_text()
    text = text.lower()
    text = re.sub(r'[^\w\s]','', text)
    tokens = word_tokenize(text)
    tokens = [t for t in tokens if t.isalpha()]
    tokens = [t for t in tokens if t not in nltk_stopwords_set]

    return tokens
```

[7]:
```
display('preprocessing all_reviews...')
all_reviews = all_reviews.apply(
    lambda x: preprocess_text(x)
)
```

```
'preprocessing all_reviews...'
```

[8]:
```
# Build Word2Vec model to get embedding layer

embedding_vector_size = 152  # faster: "% 4 = 0"

display('training Word2Vec model...')
word2vec_model = gensim.models.Word2Vec(
    sentences=all_reviews,
    size=embedding_vector_size, min_count=1, window=5,
    workers=8
)
```

```
'training Word2Vec model...'
```

[9]:
```
# Tokenize all reviews
# Note: tokenize train AND test data in one go

max_features = 5000

tokenizer = Tokenizer(num_words=max_features)
tokenizer.fit_on_texts(all_reviews)
```

```
all_reviews_seq = tokenizer.texts_to_sequences(all_reviews)
```

[10]:
```
print(
    len(all_reviews_seq),
    all_reviews_seq[0]
)
```

```
50000 [400, 70, 438, 99, 512, 2489, 108, 56, 899, 535, 174, 174, 175, 80, 14,
617, 2346, 120, 96, 9, 470, 3908, 175, 24, 230, 583, 2195, 1146, 77, 4703, 77,
679, 2, 263, 70, 10, 335, 1724, 483, 1108, 3310, 421, 784, 3381, 17, 462, 614,
1321, 16, 1028, 156, 383, 1661, 775, 2355, 4, 549, 70, 608, 67, 241, 98, 516,
150, 1, 343, 7, 45, 22, 343, 183, 9, 217, 657, 679, 2, 123, 322, 398, 129, 4160,
1598, 573, 842, 947, 827, 1092, 1596, 362, 248, 16, 529, 2128, 842, 33, 332, 18,
40, 1321, 436, 175, 3941, 470, 84, 4, 1420, 393, 2260, 116, 2091, 2461, 573, 17,
79, 110, 4573, 259, 1217, 16, 573, 481, 555, 608, 651, 3, 422, 265, 502, 116,
604, 3213, 1119, 738, 253, 1, 18, 4, 3, 566, 66, 28, 18, 643, 140, 236, 97, 614,
3665, 1782, 1, 149, 383, 1661, 245, 3, 889, 18, 43, 1422, 1149, 2195, 13, 554,
99, 379, 13, 20, 40, 18, 164, 390, 4041, 3214, 40, 91, 242, 428, 217, 252, 120,
3, 307, 1459]
```

[11]:
```
# Pad

# Calculate maxlen for a document => AVG
# Note: reduce() could be used to summarize length of array

len_sum = 0
for doc in all_reviews_seq:
    len_sum += len(doc)

avg_doc_len = math.ceil(len_sum / len(all_reviews_seq))

display(avg_doc_len)

# Apply padding
all_reviews_pad = pad_sequences(
    all_reviews_seq,
    maxlen=avg_doc_len
)

display(all_reviews_pad.shape)
```

```
95
```

```
(50000, 95)
```

[13]:
```
# Weights for embedded layer

embedding_l_weights = np.zeros(shape=(
    len(tokenizer.word_index) + 1,  # i starts from 1 in the next for-loop
    embedding_vector_size
))

for word, idx in tokenizer.word_index.items():
    vector_i = word2vec_model.wv[word]
    if vector_i is not None:
        embedding_l_weights[idx] = vector_i
```

[55]:
```
# BiLSTM RNN

model = Sequential()

model.add(Embedding(
    input_dim=len(tokenizer.word_index) + 1,
    output_dim = embedding_vector_size,

    input_length=avg_doc_len,

    weights=[embedding_l_weights]
))
model.add(Bidirectional(LSTM(128, dropout=0.25, recurrent_dropout=0.1)))
model.add(Dense(10))
model.add(Dropout(0.3))
model.add(Dense(1, activation='sigmoid'))
```

[56]:
```
# Keras: Reduce learning rate when a metric has stopped improving.

learning_rate_reduction = ReduceLROnPlateau(
    monitor='val_acc',
    patience=2, factor=0.5, min_lr=0.0001,
    verbose=1
)
```

[57]:
```
# Get data to train the model
X = all_reviews_pad[:25000, :]
X = X.reshape(-1, avg_doc_len)
y = train_sentiments

X_tr, X_val, y_tr, y_val = train_test_split(
    X, y,
    test_size=0.15, shuffle=True
)
```

[58]:
```
# Compile Keras model
```

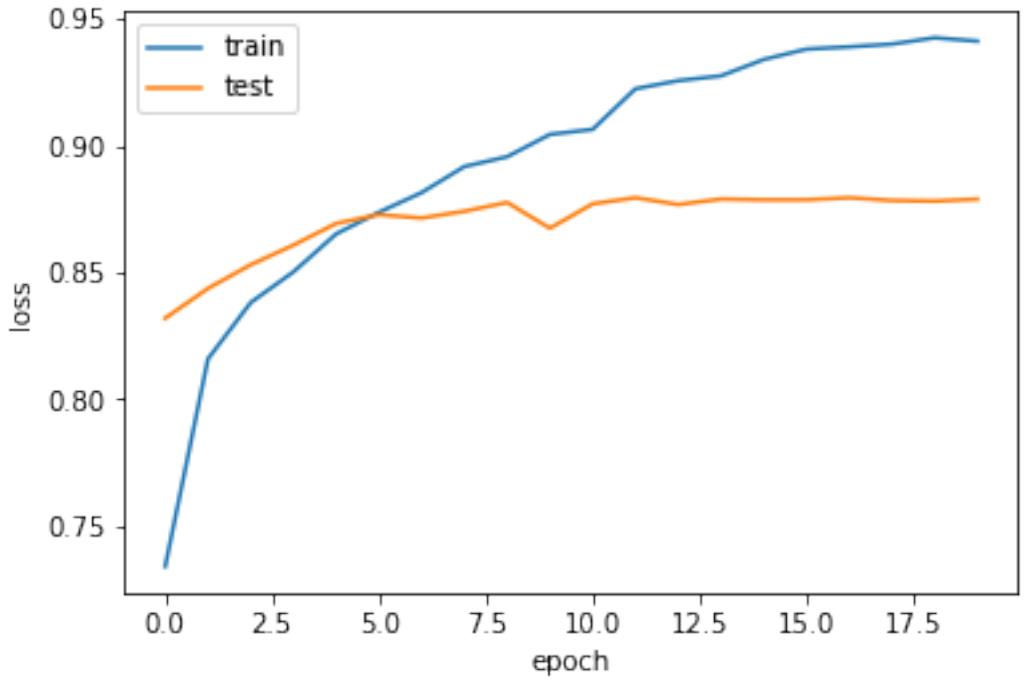
```
model.compile(
    optimizer='RMSprop',
    loss='binary_crossentropy',
    metrics=['acc']
)
```

[59]:
```
# Fit Keras model

history = model.fit(
    X_tr, y_tr,
    epochs=10,
    batch_size=500,
    callbacks=[learning_rate_reduction],
    validation_data=(X_val, y_val)
)
```

```
Train on 21250 samples, validate on 3750 samples
Epoch 1/20
21250/21250 [==============================] - 19s 910us/step - loss: 0.5337 -
acc: 0.7342 - val_loss: 0.3869 - val_acc: 0.8320
Epoch 2/20
21250/21250 [==============================] - 16s 768us/step - loss: 0.4223 -
acc: 0.8160 - val_loss: 0.3736 - val_acc: 0.8437
Epoch 3/20
21250/21250 [==============================] - 16s 769us/step - loss: 0.3766 -
acc: 0.8382 - val_loss: 0.3701 - val_acc: 0.8531
Epoch 4/20
21250/21250 [==============================] - 16s 769us/step - loss: 0.3549 -
acc: 0.8503 - val_loss: 0.3231 - val_acc: 0.8608
Epoch 5/20
21250/21250 [==============================] - 16s 770us/step - loss: 0.3274 -
acc: 0.8652 - val_loss: 0.3214 - val_acc: 0.8693
Epoch 6/20
21250/21250 [==============================] - 16s 768us/step - loss: 0.3042 -
acc: 0.8737 - val_loss: 0.3012 - val_acc: 0.8728
Epoch 7/20
21250/21250 [==============================] - 16s 769us/step - loss: 0.2926 -
acc: 0.8815 - val_loss: 0.3016 - val_acc: 0.8715
Epoch 8/20
21250/21250 [==============================] - 16s 769us/step - loss: 0.2742 -
acc: 0.8918 - val_loss: 0.3124 - val_acc: 0.8741
Epoch 9/20
21250/21250 [==============================] - 17s 787us/step - loss: 0.2602 -
acc: 0.8956 - val_loss: 0.3072 - val_acc: 0.8776
Epoch 10/20
21250/21250 [==============================] - 17s 777us/step - loss: 0.2388 -
acc: 0.9044 - val_loss: 0.3353 - val_acc: 0.8675
Epoch 11/20
```

```
21250/21250 [==============================] - 17s 798us/step - loss: 0.2309 -
acc: 0.9064 - val_loss: 0.3061 - val_acc: 0.8771

Epoch 00011: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
Epoch 12/20
21250/21250 [==============================] - 17s 797us/step - loss: 0.2021 -
acc: 0.9224 - val_loss: 0.3259 - val_acc: 0.8795
Epoch 13/20
21250/21250 [==============================] - 17s 801us/step - loss: 0.1930 -
acc: 0.9256 - val_loss: 0.3237 - val_acc: 0.8768
Epoch 14/20
21250/21250 [==============================] - 17s 788us/step - loss: 0.1866 -
acc: 0.9275 - val_loss: 0.3361 - val_acc: 0.8789

Epoch 00014: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
Epoch 15/20
21250/21250 [==============================] - 16s 774us/step - loss: 0.1723 -
acc: 0.9339 - val_loss: 0.3478 - val_acc: 0.8787
Epoch 16/20
21250/21250 [==============================] - 16s 770us/step - loss: 0.1664 -
acc: 0.9380 - val_loss: 0.3419 - val_acc: 0.8787

Epoch 00016: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
Epoch 17/20
21250/21250 [==============================] - 16s 770us/step - loss: 0.1619 -
acc: 0.9389 - val_loss: 0.3402 - val_acc: 0.8795
Epoch 18/20
21250/21250 [==============================] - 16s 770us/step - loss: 0.1604 -
acc: 0.9400 - val_loss: 0.3451 - val_acc: 0.8784

Epoch 00018: ReduceLROnPlateau reducing learning rate to 0.0001.
Epoch 19/20
21250/21250 [==============================] - 17s 800us/step - loss: 0.1523 -
acc: 0.9425 - val_loss: 0.3521 - val_acc: 0.8781
Epoch 20/20
21250/21250 [==============================] - 16s 776us/step - loss: 0.1547 -
acc: 0.9412 - val_loss: 0.3502 - val_acc: 0.8789
```

[63]:
```
import matplotlib.pyplot as plt
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



[64]:
```
y_test_pred = model.predict(X_val)
```

[65]:
```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_val, y_test_pred, average = 'weighted')
```

[65]:
```
0.948706547191803
```

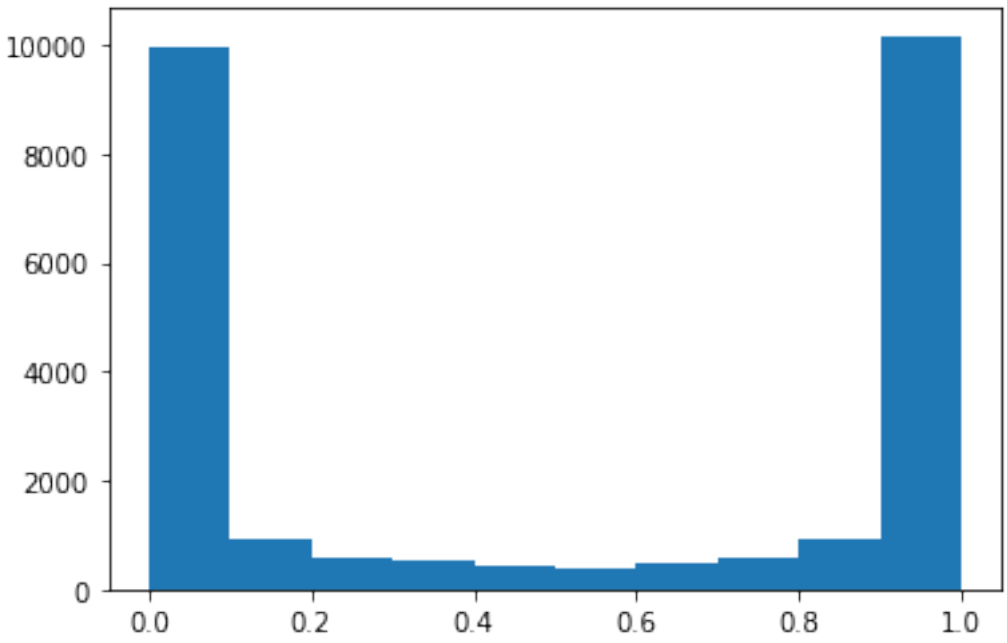
[66]:
```
#predicting test_data
y_pred = model.predict(
    all_reviews_pad[25000:, :]
)
```

[67]:
```
display(y_pred.shape, test_data.shape)
```

```
(25000, 1)


(25000, 2)
```

[69]:
```
plt.hist(y_pred)
```

[69]:
```
(array([ 9982.,   917.,   572.,   514.,   437.,   385.,   498.,   603.,
          920., 10172.]),
 array([2.1457672e-06, 1.0000192e-01, 2.0000169e-01, 3.0000147e-01,
        4.0000123e-01, 5.0000101e-01, 6.0000080e-01, 7.0000058e-01,
        8.0000031e-01, 9.0000010e-01, 9.9999988e-01], dtype=float32),
 <a list of 10 Patch objects>)
```



[72]:
```
pred_median = np.median(y_pred)

submission_predictions = [
    1 if v > pred_median
    else 0
    for v in y_pred
]
```

[76]:
```
corrected_ids = test_data['id'].str.replace('"', '')

submission = pd.DataFrame({
    'id': corrected_ids,
    'sentiment': submission_predictions
})
submission.to_csv('submission.csv', index=False)
```