TaskA_LSTM

November 27, 2024

[37]:

```
import nltk
import keras
import pandas as pd
import numpy as np
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Embedding
from keras.layers import TextVectorization
from keras.models import Sequential
from keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import KFold
from sklearn.model_selection import train_test_split
```

[2]:

```
"""
Download dataset SubtaskA.jsonl from
https://github.com/mbzuai-nlp/M4GT-Bench.
"""
DATA_PATH = "C:/Users/Admin/Downloads/SubtaskA.jsonl"

# initialize dataframe
df = pd.read_json(DATA_PATH, lines=True)
```

[14]:

```
print(df.source.value_counts())
print()
print(df.model.value_counts())
```

```
source
wikihow      36556
reddit       33999
arxiv        33998
wikipedia    31365
peerread     16891
Name: count, dtype: int64

model
human      65177
chatGPT    16892
gpt4       14344
```

```
davinci    14340
bloomz     14332
dolly      14046
cohere     13678
Name: count, dtype: int64
```

[22]:

```
print(df[df.label == 0].model.value_counts())
print()
print(df[df.label == 1].model.value_counts())
```

```
model
human    65177
Name: count, dtype: int64

model
chatGPT    16892
gpt4       14344
davinci    14340
bloomz     14332
dolly      14046
cohere     13678
Name: count, dtype: int64
```

[27]:

```
df[['text', 'label']]
```

[27]:

```
                                                     text  label
0       We consider a system of many polymers in solut…      1
1       We present a catalog of 66 YSOs in the Serpens…      1
2       Spectroscopic Observations of the Intermediate…      1
3       We present a new class of stochastic Lie group…      1
4       ALMA as the ideal probe of the solar chromosph…      1
…                                                    …    …
152804   The main results presented in this dissertati…      0
152805   Fine-grained sketch-based image retrieval (FG…      0
152806   We present the derivation of the NNLO two-par…      0
152807   The principle of optimism in the face of unce…      0
152808   We consider the setting of prediction with ex…      0

[152809 rows x 2 columns]
```

[15]:

```
"""
Pre-process dataframe.
"""
MAX_VOCAB = 10_000
MAX_LENGTH = 200

# init text vectorizer
vectorize_layer = TextVectorization(
    max_tokens=MAX_VOCAB,
    standardize='lower_and_strip_punctuation',
    split='whitespace',
    ngrams=None,
    output_mode='int',
    output_sequence_length=MAX_LENGTH,
    pad_to_max_tokens=False,
    vocabulary=None,
    idf_weights=None,
    sparse=False,
    ragged=False,
    encoding='utf-8',
    name=None,
)

# create vocabulary
vectorize_layer.adapt(df['text'])
vocab = vectorize_layer.get_vocabulary()
```

[34]:

```
# vectorize text data (in subsets for memory constraints)
X = []
y = df['label']

subset_size = df.shape[0] // 100
for i in range(0, df.shape[0], subset_size):
    subset = df['text'][i : i + subset_size]
    X.append(vectorize_layer(subset))

X = np.vstack(X)
print(X.shape, y.shape)
```

```
(152809, 200) (152809,)
```

[39]:

```
"""
LSTM model generator.
"""
EMBEDDING_DIM = 128
N_HIDDEN = 100
OPTIMIZER = 'adam'
N_CLASSES = 2

def get_model(model_path=None):
    if model_path:
        # load existing model
        model = keras.models.load_model(model_path)
    else:
        # create new model
        model = Sequential()
        embeddings = Embedding(
            input_dim=MAX_VOCAB,
            output_dim=EMBEDDING_DIM,
        )
        model.add(embeddings)
        model.add(LSTM(N_HIDDEN, return_sequences=False))
        model.add(Dense(N_CLASSES)) # , activation='softmax'
        model.compile(
            loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
            optimizer=OPTIMIZER,
            metrics=['accuracy']
        )
    return model
```

[41]:

```
"""
Train and evaluate model.
"""
# create new model
model = get_model()

# create data splits
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.15,
    random_state=777,
)

# train the model
model.fit(
    X_train,
    y_train,
    epochs=5,
    batch_size=64
)

# final evaluation of the model
scores = model.evaluate(
    X_test,
    y_test,
    verbose=0
)
accuracy = scores[1]

# report results
print("Accuracy: %.2f%%" % (accuracy * 100))
```

```python
# save model
model.save("models/taskA_lstm.keras")
```

```
Epoch 1/5
2030/2030            370s 181ms/step
- accuracy: 0.7507 - loss: 0.5110
Epoch 2/5
2030/2030            367s 181ms/step
- accuracy: 0.8752 - loss: 0.3056
Epoch 3/5
2030/2030            368s 181ms/step
- accuracy: 0.9035 - loss: 0.2410
Epoch 4/5
2030/2030            366s 180ms/step
- accuracy: 0.9293 - loss: 0.1850
Epoch 5/5
2030/2030            363s 179ms/step
- accuracy: 0.9453 - loss: 0.1446
Accuracy: 88.61%
```

[10]:

```python

```