Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №7

з дисципліни «Програмні засоби проектування та реалізаціїї нейромережевих систем»

Тема: " Рекурентні нейронні мережі LSTM "

Виконав:	Перевірив:
студент групи ІП-93	Шимкович Володимир
Домінський Валентин	Миколайович
Олексійович	

Київ 2022

Зміст:

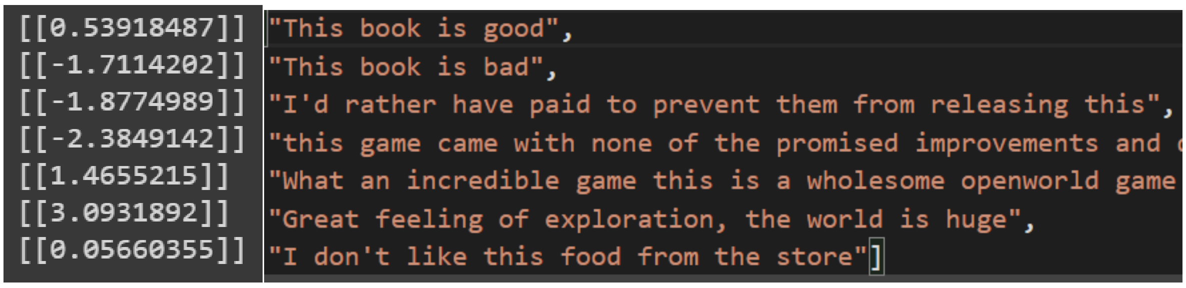
Мета:

Написати програму, що реалізує рекурентну нейронну мережу LSTM для розпізнавання емоційного забарвлення тексту, використати датасет Yelp Dataset

Вихідний код

```
import tensorflow as tf
import tensorflow_datasets as tfds

dataset_name = 'yelp_polarity_reviews/subwords8k'
text_feature = 'text'
encoder_subwords = 50
delimiter = '----------'
example = "the park is nice and quiet" # 1, .., 13, .., 3, ..
examples_are_correct = "examples are correct"
examples_are_not_correct = "examples are not correct"
activation_type = 'relu'
learning_rate = 1e-4
metrics_type = 'accuracy'
model_name = 'lab7.h5'
model_weights_name = "lab7_weights.h5"

"""# Load Dataset"""

(train_dataset, test_dataset), dataset_info = tfds.load(name=dataset_name,
                                          split=(tfds.Split.TRAIN, tfds.Split.TEST),
                                          with_info=True,
                                          as_supervised=True)
encoder = dataset_info.features[text_feature].encoder

print(dataset_info.splits)
print(delimiter)
print(encoder.vocab_size)
print(delimiter)
print(encoder.subwords[:encoder_subwords])

example_ids = encoder.encode(example)
print(example_ids)

example_from_ids = encoder.decode(example_ids)
print(example_from_ids)

if (example == example_from_ids):
  print(examples_are_correct)
else:
  print(examples_are_not_correct)

"""# Training and Validation"""

buffer_size = 800
batch_size = 50

train_data = train_dataset.shuffle(buffer_size).padded_batch(batch_size = batch_size,
padded_shapes = ([None],[]))
test_data = test_dataset.shuffle(buffer_size).padded_batch(batch_size = batch_size,
padded_shapes = ([None],[]))

"""# Model Definition"""

# 1) Word Embeddings = trandsforms integer = [[4], [20]] -> [[0.25, 0.1], [0.6, -0.2]]
# 2) Bi-directional layer =  LSTMs have been one-way models, also
  # called unidirectional ones. In other words, sequences such as
  # tokens (i.e. words) are read in a left-to-right or right-to-left fashion.
  # This does not necessarily reflect good practice, as more recent Transformer
  # based approaches like BERT suggest. In fact, bidirectionality - or processing
  # the input in a left-to-right and a right-to-left fashion,
  # can improve the performance of your Machine Learning model.
# 3) Dense Layer = Just your regular densely-connected NN layer
# 4) Binary Output
model = tf.keras.Sequential([tf.keras.layers.Embedding(encoder.vocab_size,
batch_size),
                             tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(units =
64)),
                             tf.keras.layers.Dense(units = 64, activation =
activation_type),
                             tf.keras.layers.Dense(units = 1)
])

# BinaryCrossEntropy = two label classes
model.compile(optimizer = tf.keras.optimizers.Adam(learning_rate = learning_rate),
              loss = tf.keras.losses.BinaryCrossentropy(from_logits = True),
              metrics = [metrics_type])

"""# Model Training & Saving"""

epochs = 5
validation_cycles = 10

# workers = maximum number of processes to spin up when using process-based threading
hist = model.fit(train_data,
                 epochs = epochs,
                 validation_data = test_data,
                 validation_steps = validation_cycles,
                 workers = 8)

model.save(model_name)
model.save_weights(model_weights_name)

"""# Trained Model Performance Evaluation"""

test_loss, test_acc = model.evaluate(test_data)

print('Accuracy:', test_acc)
print('Loss:', test_loss)

"""## Model Evaluation

If the prediction is >= 0.5, it is positive else it is negative.
"""

def predict(text):
    encoded = encoder.encode(text)
    encoded = tf.cast(encoded, tf.float32)
    return (model.predict(tf.expand_dims(encoded, 0)))

example_texts = ["This book is good",
                 "This book is bad",
                 "I'd rather have paid to prevent them from releasing this",
                 "this game came with none of the promised improvements and didn't
even fix the old bugs",
                 "What an incredible game this is a wholesome openworld game I dont
understand why some of the idiots are writing emotional review how could people
without rational judgment write a review?",
                 "Great feeling of exploration, the world is huge",
                 "I don’t like this food from the store"]

for text in example_texts:
  print(predict(text))
```

Результат роботи:



```
[[0.53918487]]
[[-1.7114202]]
[[-1.8774989]]
[[-2.3849142]]
[[1.4655215]]
[[3.0931892]]
[[0.05660355]]
```

```
"This book is good",
"This book is bad",
"I'd rather have paid to prevent them from releasing this",
"this game came with none of the promised improvements and
"What an incredible game this is a wholesome openworld game
"Great feeling of exploration, the world is huge",
"I don't like this food from the store"]
```

Висновки:

Я дізнався більше інформації про LSTM мережі, чим вони відрізняються від RNN та коли їх варто використовувати