Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Звіт до лабораторної роботи №7
«Рекурентні нейронні мережі LSTM»
з дисципліни
«Програмні засоби проектування та реалізації нейромережевих систем»

Виконала:
студентка групи ІМ-13
Мартинюк Марія Павлівна
номер у списку: 63

Перевірив:
Шимкович В. М.

Київ 2024

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

Хід роботи:
Спершу було реалізовано додаткову функцію для відображення графіку залежності точності відносно епох навчання:

```
def accuracy_graph_display(history):
    plt.plot(history.history['accuracy'], label='Accuracy',
color='green')
    plt.plot(history.history['val_accuracy'], label='Validation
Accuracy', color='red')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.ylim([0, 1])
    plt.legend(loc='lower right')
    plt.grid(True)
    plt.show()
```

Сама нейронна мережа реалізована у вигляді класу SentimentAnalysisModel, у якому присутні відповідні методи ініціалізації, завантаження рекомендованого завданням датасету, ділення його на тренувальні та тестувальні дані, створення та тренування моделі. Також додатково реалізовано метод predict_sentiment за допомогою якого пізніше було протестовано модель на конкретних реченнях.

```
class SentimentAnalysisModel:
    def __init__(self, vocabulary=20000, length=200):
        self.vocabulary = vocabulary
        self.length = length
        self.tokenizer = None
        self.model = None

       def prepare_data(self, trainX):
        self.tokenizer = Tokenizer(num_words=self.vocabulary)
        self.tokenizer.fit_on_texts(trainX)

    def create_model(self):
```

```
        self.model = models.Sequential([
            Embedding(self.vocabulary, 8),
            LSTM(16),
            Dense(64, activation="relu"),
            Dense(1, activation="sigmoid")
        ])

        self.model.compile(optimizer="RMSprop",
loss="binary_crossentropy", metrics=["accuracy"])
        return self.model

    def train_model(self, trainX, trainY, testX, testY, epochs=10):
        train_t = self.tokenizer.texts_to_sequences(trainX)
        test_t = self.tokenizer.texts_to_sequences(testX)
        train_t = pad_sequences(train_t, maxlen=self.length)
        test_t = pad_sequences(test_t, maxlen=self.length)
        history = self.model.fit(train_t, np.array(trainY),
epochs=epochs, validation_data=(test_t, np.array(testY)))

        return history

    def predict_sentiment(self, sentences):
        for sentence in sentences:
            print(sentence)
            input_seq =
pad_sequences(self.tokenizer.texts_to_sequences([sentence]),
maxlen=self.length)
            prediction = self.model.predict(input_seq, verbose=0)
            prediction_score = prediction[0][0]
            if prediction_score < 0.45:
                sentiment = 'Negative'
            elif prediction_score > 0.85:
                sentiment = 'Positive'
            else:
                sentiment = 'Neutral'
            print(f"Prediction: {sentiment} ({prediction_score})")
```

Головний метод відповідає за створення екземпляру класу нейронної мережі, завантаження даних, та тренування моделі. Також у цьому блоці відбувається виклик функції відображення графіку залежності та

перевірки на додатково визначених реченнях з різним емоційним забарвленням (Positive, Negative, Neutral).

```
def main():
    data = tfds.load("yelp_polarity_reviews", as_supervised=True)
    train_set, test_set = data['train'], data['test']
    trainX, trainY = [], []
    for element in train_set:
        trainX.append(element[0].numpy().decode())
        trainY.append(int(element[1].numpy()))

    testX, testY = [], []
    for element in test_set:
        testX.append(element[0].numpy().decode())
        testY.append(int(element[1].numpy()))

    model = SentimentAnalysisModel()
    model.prepare_data(trainX)
    model.create_model()
    history = model.train_model(trainX, trainY, testX, testY)
    accuracy_graph_display(history)

    model.predict_sentiment(test_sentences)
```

Масив попередньо підготовлених речень виглядає наступним чином:

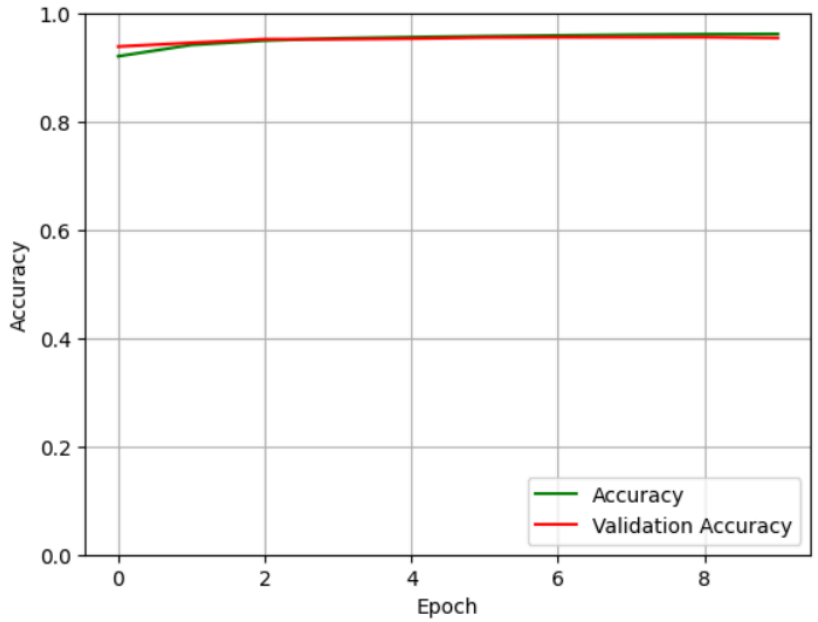
```
test_sentences = [
    "The packaging was impressive, and the item arrived in perfect condition.", # +
    "The characters were well-defined, but lacked depth and complexity.",       # +-
    "The application keeps crashing, it's very frustrating to use.",            # -
    "The play was fantastic, the actors gave a stellar performance.",           # +
    "The plot was straightforward, with no unexpected twists."                  # +-
]
```

Результати навчання:

По завершенню навчання була досягнута точність 95.51%, що є досить гарним результатом.

```
Epoch 9/10
17500/17500 [==============================] - 166s 9ms/step - loss: 0.1055 - accuracy: 0.9619 - val_loss: 0.1153 - val_accuracy: 0.9564
Epoch 10/10
17500/17500 [==============================] - 166s 10ms/step - loss: 0.1045 - accuracy: 0.9623 - val_loss: 0.1292 - val_accuracy: 0.9551
```

Графік залежності точності відносно епох навчання:



Результати додаткового тестування на низці речень різного емоційного забарвлення наступні:

```
The packaging was impressive, and the item arrived in perfect condition.
Prediction: Neutral (0.8145988583564758)
The characters were well-defined, but lacked depth and complexity.
Prediction: Negative (0.11872754245996475)
The application keeps crashing, it's very frustrating to use.
Prediction: Negative (0.04729093611240387)
The play was fantastic, the actors gave a stellar performance.
Prediction: Positive (0.9942723512649536)
The plot was straightforward, with no unexpected twists.
Prediction: Neutral (0.7160633206367493)
```

Практично усі випадки визначені правильно.

Висновок:

Як результат виконання лабораторної роботи було створено рекурентну нейронну мережу LSTM для розпізнавання емоційного забарвлення тексту. Окрім цього для наочності було виведено результати навчання у вигляді графіку з відображення зміни точності відносно епох навчання. Додатково модель було протестовано на іншому наборі даних для перевірки якості

роботи моделі в реальних умовах. За отриманими результатами можна зробити висновок, що мережа, пройшовши 10 епох, навчилась класифікувати текст із точністю 95.51%.