

LSTM cho bài toán sinh từ có giao diện

1. Các thư viện import:

```
import tensorflow as tf
import numpy as np
import collections
from keras.layers import LSTM, Dense
```

- **tensorflow** và **keras**: sử dụng để xây dựng mô hình deep learning với LSTM và các lớp Dense.
- **numpy**: hỗ trợ thao tác với mảng và ma trận.
- **collections**: giúp đếm số lần xuất hiện của các từ.

2. Đọc dữ liệu từ file (**read_data**):

```
def read_data(fname):
    with open(fname) as f:
        content = f.readlines()
    content = [x.strip() for x in content]
    words = []
    for line in content:
        words.extend(line.split())
    return np.array(words)
```

- Hàm **read_data(fname)** mở và đọc từng dòng trong tệp văn bản, loại bỏ khoảng trắng và lưu trữ các từ vào một mảng **words**.
- Sau khi đọc, **words** chứa tất cả các từ trong văn bản dưới dạng mảng **numpy**.

3. Xây dựng từ điển từ và mã hóa từ (**build_dataset**):

```
def build_dataset(words):
    count = collections.Counter(words).most_common()
    word2id = {}
    for word, freq in count:
        word2id[word] = len(word2id)
    id2word = dict(zip(word2id.values(), word2id.keys()))
    return word2id, id2word
```

- Hàm `build_dataset(words)` tạo từ điển `word2id`, ánh xạ mỗi từ vào một chỉ số duy nhất.
- Từ điển `id2word` cũng được tạo để có thể tra cứu từ từ chỉ số.
- `vocab_size`: kích thước từ điển (số lượng từ duy nhất trong dữ liệu).

4. Tạo tập dữ liệu huấn luyện:

```
X, Y = [], []
for i in range(timestep, len(data)):
    X.append([w2i[data[k]] for k in range(i-timestep, i)])
    Y.append(w2i[data[i]])

encoded_data = [w2i[x] for x in data]
X = encoded_data[:-1]
Y = encoded_data[timestep:]
train_data = tf.keras.preprocessing.timeseries_dataset_from_array(
    X, Y, sequence_length=timestep, sampling_rate=1
)
```

- Dùng `timestep` là số từ phía trước cần dùng để dự đoán từ tiếp theo.
- Với mỗi từ `data[i]`, mảng `X` chứa các chỉ số của các từ từ `i-timestep` đến `i-1`, còn `Y` chứa chỉ số từ `i`.
- Tạo `train_data` với `tf.keras.preprocessing.timeseries_dataset_from_array`, trong đó `X` là dữ liệu đầu vào, `Y` là nhãn để huấn luyện mô hình.

5. Xây dựng mô hình LSTM:

```

model = tf.keras.Sequential()
model.add(LSTM(512, return_sequences=True,
input_shape=(timestep, 1)))
model.add(LSTM(512, return_sequences=False))
model.add(Dense(vocab_size))
model.summary()

```

- Mô hình LSTM được xây dựng với hai lớp LSTM 512 đơn vị, trong đó lớp đầu tiên có `return_sequences=True` để giữ lại các chuỗi trung gian cho lớp LSTM tiếp theo.
- Lớp `Dense` với số đầu ra là `vocab_size` nhằm dự đoán từ tiếp theo dựa trên toàn bộ từ vựng.

Model: "sequential_2"

Layer (type)	Output Shape	Param #
lstm_4 (LSTM)	(None, 3, 512)	1,052,672
lstm_5 (LSTM)	(None, 512)	2,099,200
dense_2 (Dense)	(None, 91)	46,683

Total params: 3,198,555 (12.20 MB)
 Trainable params: 3,198,555 (12.20 MB)
 Non-trainable params: 0 (0.00 B)

6. Biên dịch và huấn luyện mô hình:

```

model.compile(optimizer='adam',
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
metrics=['accuracy'])
model.fit(train_data, epochs=500)

```

- Mô hình được biên dịch với `adam` làm optimizer và `SparseCategoricalCrossentropy` làm hàm loss để phân loại các từ tiếp theo.
- Huấn luyện mô hình trong 500 epoch trên tập `train_data`.

```

1/1 ————— 0s 264ms/step - accuracy: 1.0000 - loss: 0.0191
Epoch 490/500
1/1 ————— 0s 265ms/step - accuracy: 1.0000 - loss: 0.0190
Epoch 491/500
1/1 ————— 0s 277ms/step - accuracy: 1.0000 - loss: 0.0189
Epoch 492/500
1/1 ————— 0s 273ms/step - accuracy: 1.0000 - loss: 0.0188
Epoch 493/500
1/1 ————— 0s 261ms/step - accuracy: 1.0000 - loss: 0.0187
Epoch 494/500
1/1 ————— 0s 280ms/step - accuracy: 1.0000 - loss: 0.0186
Epoch 495/500
1/1 ————— 0s 279ms/step - accuracy: 1.0000 - loss: 0.0185
Epoch 496/500
1/1 ————— 0s 268ms/step - accuracy: 1.0000 - loss: 0.0184
Epoch 497/500
1/1 ————— 0s 300ms/step - accuracy: 1.0000 - loss: 0.0183
Epoch 498/500
1/1 ————— 0s 266ms/step - accuracy: 1.0000 - loss: 0.0182
Epoch 499/500
1/1 ————— 0s 276ms/step - accuracy: 1.0000 - loss: 0.0181
Epoch 500/500
1/1 ————— 0s 263ms/step - accuracy: 1.0000 - loss: 0.0180

```

7. Dự đoán từ tiếp theo có giao diện:

```

import streamlit as st
import numpy as np
import tensorflow as tf

```

`streamlit` cho giao diện, `numpy` cho xử lý mảng, và `tensorflow`.

```

def encode(sent):
    return np.array([[w2i[w] for w in sent.split() if w in w2i]])

```

Chuyển đổi chuỗi từ thành mảng NumPy chứa chỉ số từ theo từ điển `w2i`.

```

def generate_sequence(start_words, num_words=10):
    current_words = encode(start_words)
    generated_words = start_words.split()

    for _ in range(num_words):
        pred = model.predict(current_words)
        next_word_id = np.argmax(pred)
        next_word = i2w[next_word_id]
        generated_words.append(next_word)
        current_words = np.append(current_words[:, 1:], [[next_word_id]], axis=1)

    return " ".join(generated_words)

st.write("Nhập vào 3 từ và nhấn nút để sinh ra 10 từ tiếp theo.")

```

Sinh ra chuỗi từ dựa trên 3 từ đầu vào

Vòng lặp: Dự đoán từ tiếp theo 10 lần và cập nhật `current_words` sau mỗi lần dự đoán.

```
input_text = st.text_input("Nhập 3 từ:", " ")

if st.button("Sinh từ tiếp theo"):
    if len(input_text.split()) == 3:
        output_text = generate_sequence(input_text)
        st.write("Chuỗi từ sinh ra:", output_text)
    else:
        st.write("Vui lòng nhập chính xác 3 từ.")
```

Nhập 3 từ và nhấn nút để sinh từ tiếp theo. Nếu không đúng 3 từ, hiển thị thông báo lỗi.

Nhập vào 3 từ và nhấn nút để sinh ra 10 từ tiếp theo.

Nhập 3 từ:

Aesop is considered

Sinh từ tiếp theo

Chuỗi từ sinh ra: Aesop is considered the author of many famous fables worldwide, passed down orally