【要点まとめ】
モデルの定義では、Encoderの出力をコンテキストのみDecoderに対して渡している点に注目する必要がある。
これにより、コンテキスト意外は繋がりのない分離したモデル構造となる。 ※実装演習に例を写経

【実装演習】
モデルを定義する

```
NUM_ENC_TOKENS = 1
NUM_DEC_TOKENS = 1
NUM_HIDDEN_PARAMS = 10
NUM_STEPS = 24

tf.keras.backend.clear_session()

e_input = tf.keras.layers.Input(shape=(NUM_STEPS, NUM_ENC_TOKENS),
name='e_input')
_, e_state = tf.keras.layers.SimpleRNN(NUM_HIDDEN_PARAMS,
return_state=True, name='e_rnn')(e_input)

d_input = tf.keras.layers.Input(shape=(NUM_STEPS, NUM_DEC_TOKENS),
name='d_input')
d_rnn = tf.keras.layers.SimpleRNN(NUM_HIDDEN_PARAMS,
return_sequences=True, return_state=True, name='d_rnn')
d_rnn_out, _ = d_rnn(d_input, initial_state=[e_state])

d_dense = tf.keras.layers.Dense(NUM_DEC_TOKENS, activation='linear',
name='d_output')
d_output = d_dense(d_rnn_out)

model_train = tf.keras.models.Model(inputs=[e_input, d_input],
outputs=d_output)
model_train.compile(optimizer='adam', loss='mean_squared_error')

model_train.summary()
```

学習用データを準備する

```
n = len(x) - NUM_STEPS
ex = np.zeros((n, NUM_STEPS))
dx = np.zeros((n, NUM_STEPS))
dy = np.zeros((n, NUM_STEPS))

for i in range(0, n):
    ex[i] = seq_in[i:i + NUM_STEPS]
    dx[i, 1:] = seq_out[i:i + NUM_STEPS - 1]
    dy[i] = seq_out[i: i + NUM_STEPS]

ex = ex.reshape(n, NUM_STEPS, 1)
```

```
dx = dx.reshape(n, NUM_STEPS, 1)
dy = dy.reshape(n, NUM_STEPS, 1)
```

学習を行う

```
BATCH_SIZE = 16
EPOCHS = 80

history = model_train.fit([ex, dx], dy, batch_size=BATCH_SIZE,
epochs=EPOCHS, validation_split=0.2, verbose=False)
```

推論用モデルを定義する

```
model_pred_e = tf.keras.models.Model(inputs=[e_input], outputs=[e_state])

pred_d_input = tf.keras.layers.Input(shape=(1, 1))
pred_d_state_in = tf.keras.layers.Input(shape=(NUM_HIDDEN_PARAMS))

pred_d_output, pred_d_state = d_rnn(pred_d_input, initial_state=
[pred_d_state_in])
pred_d_output = d_dense(pred_d_output)

pred_d_model = tf.keras.Model(inputs=[pred_d_input, pred_d_state_in],
outputs=[pred_d_output, pred_d_state])
```

推論用関数を定義する

```
def predict(input_data):
    state_value = model_pred_e.predict(input_data)
    _dy = np.zeros((1, 1, 1))

    output_data = []

    for i in range(0, NUM_STEPS):
        y_output, state_value = pred_d_model.predict([_dy, state_value])

        output_data.append(y_output[0, 0, 0])
        _dy[0, 0, 0] = y_output

    return output_data
```

推論の実行（に加えて、グラフのプロット）

```
init_points = [0, 24, 49, 74]
```

```
for i in init_points:
    _x = ex[i : i + 1]
    _y = predict(_x)

    if i == 0:
        plt.plot(x[i : i + NUM_STEPS], _y, color="red", label='output')
    else:
        plt.plot(x[i : i + NUM_STEPS], _y, color="red")

    plt.plot(x, seq_out, color = 'blue', linestyle = "dashed", label =
'correct')
    plt.grid()
    plt.legend()
    plt.show()
```

【実装演習考察】

Seq2Seqの理論は動画やテキストで（少しは）理解していたつもりでしたが、いざ実装は自力では不可能でした。モデルの設定からデータの前準備、推論の実行まで写経することで一連の実装の流れを理解できたように思えます。tensorflowやkerasの扱いを身につけるためにも、理屈だけではなく実装も続けていこうと思いました。