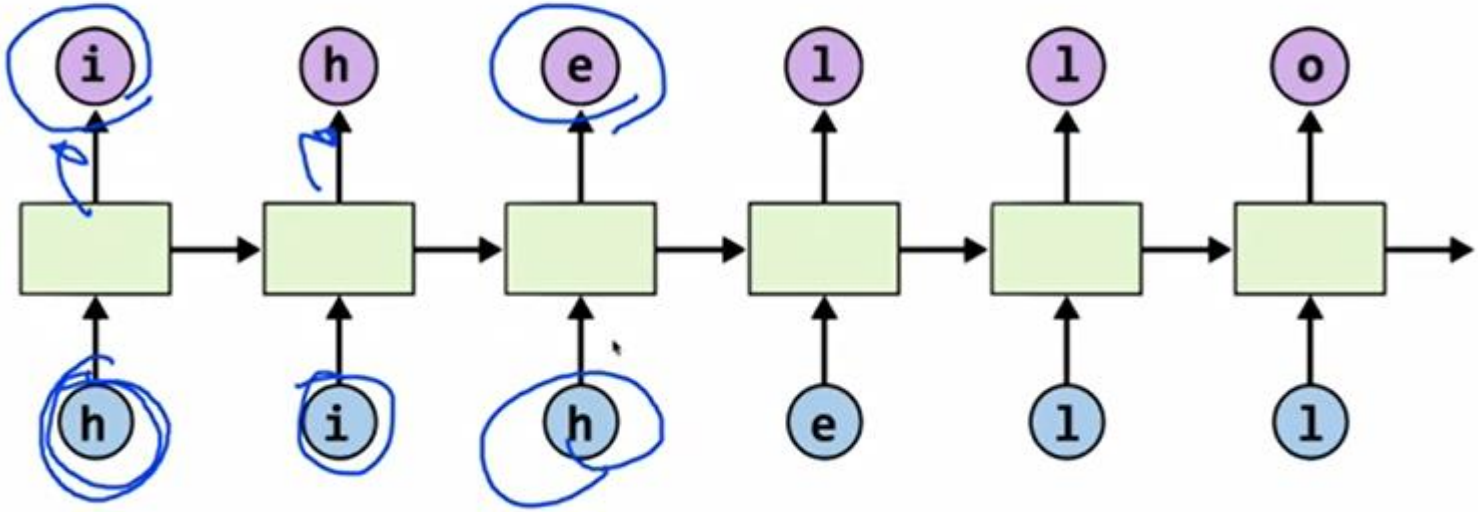
YouTube

ML lab12-2: RNN - Hi Hello Training

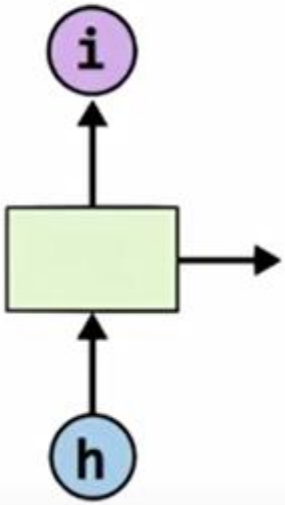
Teach RNN 'hihello'



One-hot encoding

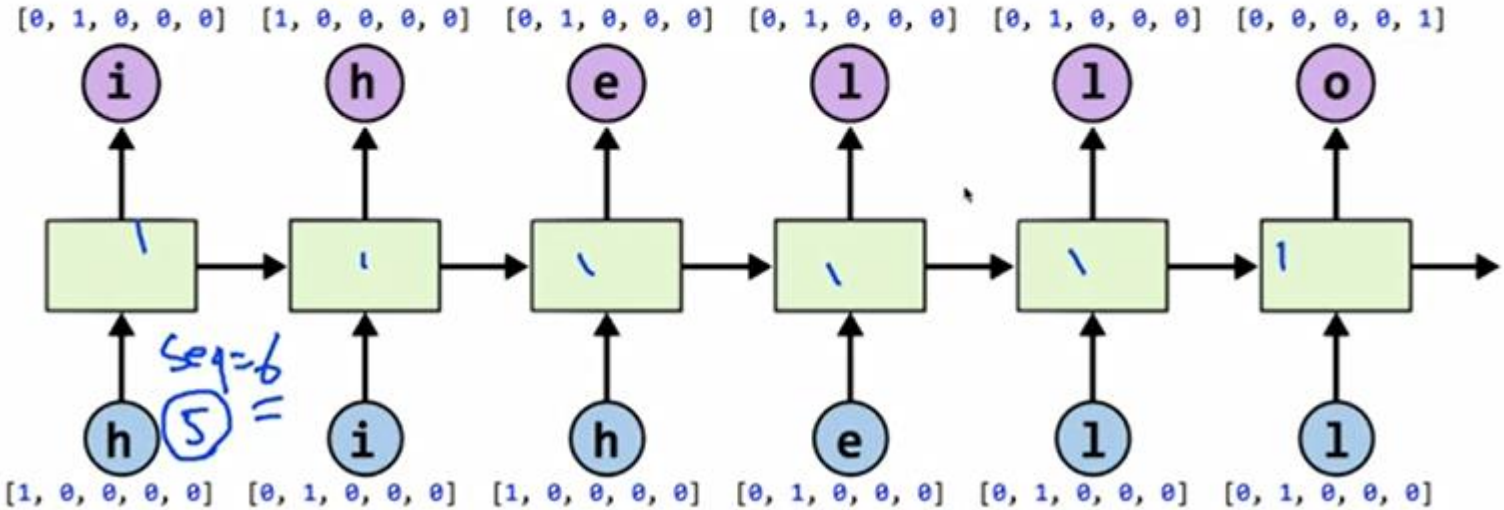
- text: 'hihello'
- unique chars (vocabulary, voc):
  h, i, e, l, o
- voc index: (dic)
  h:0, i:1, e:2, l:3, o:4

```
[1, 0, 0, 0, 0],   # h 0
[0, 1, 0, 0, 0],   # i 1
[0, 0, 1, 0, 0],   # e 2
[0, 0, 1, 0, 0],   # l 3
[0, 0, 0, 1, 0],   # o 4
```



Teach RNN 'hihello'

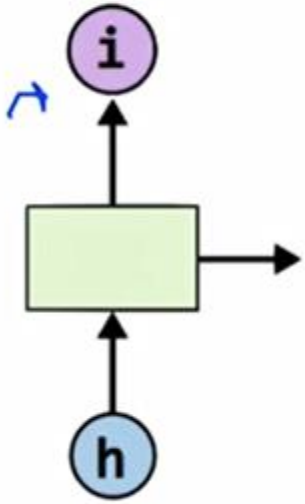
```
[1, 0, 0, 0, 0],   # h 0
[0, 1, 0, 0, 0],   # i 1
[0, 0, 1, 0, 0],   # e 2
[0, 0, 0, 1, 0],   # l 3
[0, 0, 0, 0, 1],   # o 4
```



Creating rnn cell

```
# RNN model
rnn_cell = rnn_cell.BasicRNNCell(rnn_size)

rnn_cell = rnn_cell. BasicLSTMCell(rnn_size)
rnn_cell = rnn_cell. GRUCell(rnn_size)
```

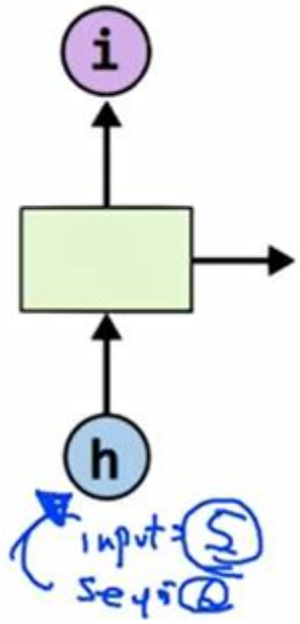


Execute RNN

hidden_rnn_size

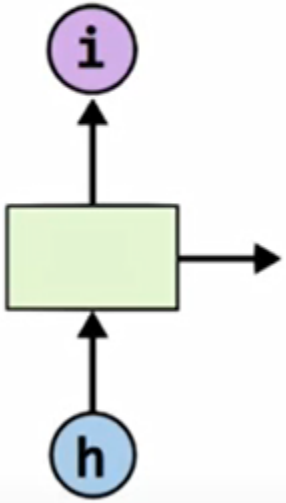
```
# RNN model
rnn_cell = rnn_cell.BasicRNNCell(rnn_size)

outputs, _states = tf.nn.dynamic_rnn(
                    rnn_cell,
                    x,
                    initial_state=initial_state,
                    dtype=tf.float32)
```



RNN parameters

```
hidden_size = 5        # output from the LSTM
input_dim = 5          # one-hot size
batch_size = 1         # one sentence
sequence_length = 6    # |ihello| == 6
```



Data creation

dic

```
idx2char = ['h', 'i', 'e', 'l', 'o'] # h=0, i=1, e=2, l=3, o=4
x_data = [[0, 1, 0, 2, 3, 3]]        # hihell
x_one_hot = [[[1, 0, 0, 0, 0],       # h 0
              [0, 1, 0, 0, 0],       # i 1
              [1, 0, 0, 0, 0],       # h 0
              [0, 0, 1, 0, 0],       # e 2
              [0, 0, 0, 1, 0],       # l 3
              [0, 0, 0, 1, 0]]]      # l 3
```

time

```
y_data = [[1, 0, 2, 3, 3, 4]]        # ihello
X = tf.placeholder(tf.float32,
        [None, sequence_length, input_dim]) # X one-hot
Y = tf.placeholder(tf.int32, [None, sequence_length])  # Y label
```

Feed to RNN

```
x_one_hot = [[[1, 0, 0, 0, 0],   # h 0
              [0, 1, 0, 0, 0],   # i 1
              [1, 0, 0, 0, 0],   # h 0
              [0, 0, 1, 0, 0],   # e 2
              [0, 0, 0, 1, 0],   # l 3
              [0, 0, 0, 1, 0]]]  # l 3

y_data = [[1, 0, 2, 3, 3, 4]]    # ihello
```

one hot

batch 6

input-dim 5

```
X = tf.placeholder(
    tf.float32, [None, sequence_length, hidden_size])  # X one-hot
Y = tf.placeholder(tf.int32, [None, sequence_length])  # Y label

cell = tf.contrib.rnn.BasicLSTMCell(num_units=hidden_size,
state_is_tuple=True)
initial_state = cell.zero_state(batch_size, tf.float32)
outputs, _states = tf.nn.dynamic_rnn(
    cell, X, initial_state=initial_state, dtype=tf.float32)
```

5

Cost: sequence_loss

```
# [batch_size, sequence_length]
y_data = tf.constant([[1, 1, 1]])

# [batch_size, sequence_length, emb_dim ]
prediction = tf.constant([[[0.2, 0.7], [0.6, 0.2], [0.2, 0.9]]], dtype=tf.float32)

# [batch_size * sequence_length]
weights = tf.constant([[1, 1, 1]], dtype=tf.float32)

sequence_loss = tf.contrib.seq2seq.sequence_loss(logits=prediction, targets=y_data, weights=weight
s)
sess.run(tf.global_variables_initializer())
print("Loss: ", sequence_loss.eval())
```

```
Loss:  0.596759
```

Cost: sequence_loss

```
# [batch_size, sequence_length]
y_data = tf.constant([[1, 1, 1]])

# [batch_size, sequence_length, emb_dim ]
prediction1 = tf.constant([[[0.3, 0.7], [0.3, 0.7], [0.3, 0.7]]],
dtype=tf.float32)
prediction2 = tf.constant([[[0.1, 0.9], [0.1, 0.9], [0.1, 0.9]]],
dtype=tf.float32)

# [batch_size * sequence_length]
weights = tf.constant([[1, 1, 1]], dtype=tf.float32)

sequence_loss1 = tf.contrib.seq2seq.sequence_loss(prediction1, y_data,
weights)
sequence_loss2 = tf.contrib.seq2seq.sequence_loss(prediction2, y_data,
weights)

sess.run(tf.global_variables_initializer())
print("Loss1: ", sequence_loss1.eval(),
        "Loss2: ", sequence_loss2.eval())
```

time

```
Loss1: 0.513015
Loss2: 0.371101
```

Cost: sequence_loss

```
outputs, _states = tf.nn.dynamic_rnn(
   cell, X, initial_state=initial_state, dtype=tf.float32)
weights = tf.ones([batch_size, sequence_length])

sequence_loss = tf.contrib.seq2seq.sequence_loss(
                    logits=outputs, targets=Y, weights=weights)
loss = tf.reduce_mean(sequence_loss)
train = tf.train.AdamOptimizer(learning_rate=0.1).minimize(loss)
```

Training

0, 2, 3

```
prediction = tf.argmax(outputs, axis=2)

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for i in range(2000):
        l, _ = sess.run([loss, train], feed_dict={X: x_one_hot, Y: y_data})
        result = sess.run(prediction, feed_dict={X: x_one_hot})
        print(i, "loss:", l, "prediction: ", result, "true Y: ", y_data)

        # print char using dic
        result_str = [idx2char[c] for c in np.squeeze(result)]
        print("\tPrediction str: ", ''.join(result_str))
```

Results

```
prediction = tf.argmax(outputs, axis=2)

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for i in range(2000):
        l, _ = sess.run([loss, train], feed_dict={X: x_one_hot, Y: y_data})
        result = sess.run(prediction, feed_dict={X: x_one_hot})
        print(i, "loss:", l, "prediction: ", result, "true Y: ", y_data)

        # print char using dic
        result_str = [idx2char[c] for c in np.squeeze(result)]
        print("\tPrediction str: ", ''.join(result_str))
```

Step

```
0 loss: 1.55474 prediction:  [[3 3 3 3 4 4]] true Y:  [[1, 0, 2, 3, 3, 4]] Prediction str:  lllloo
1 loss: 1.55081 prediction:  [[3 3 3 3 4 4]] true Y:  [[1, 0, 2, 3, 3, 4]] Prediction str:  lllloo
2 loss: 1.54704 prediction:  [[3 3 3 3 4 4]] true Y:  [[1, 0, 2, 3, 3, 4]] Prediction str:  lllloo
3 loss: 1.54342 prediction:  [[3 3 3 3 4 4]] true Y:  [[1, 0, 2, 3, 3, 4]] Prediction str:  lllloo
...
1998 loss: 0.75305 prediction:  [[1 0 2 3 3 4]] true Y:  [[1, 0, 2, 3, 3, 4]] Prediction str:  ihello
1999 loss: 0.752973 prediction:  [[1 0 2 3 3 4]] true Y:  [[1, 0, 2, 3, 3, 4]] Prediction str:  ihello
```