Natural Language Processing

Using tensorflow

BITTIGER

The Lifelong Learning Platform of Silicon Valley

Copyright Policy

All content included on the Site or third-party platforms as part of the class, such as text, graphics, logos, button icons, images, audio clips, video clips, live streams, digital downloads, data compilations, and software, is the property of BitTiger or its content suppliers and protected by copyright laws.

Any attempt to redistribute or resell will result in the appropriate legal action being taken.

We thank you in advance for respecting our copyrighted content.

Outline

1. Input Shape
2. RNN Feed
3. Model (variables and ops)

Input Shape

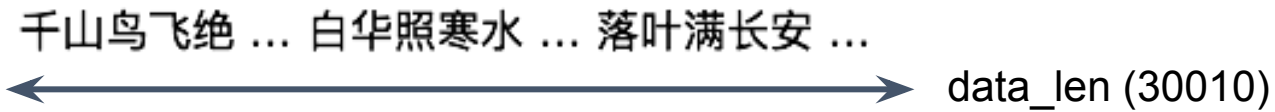
BITTIGER

The Lifelong Learning Platform of Silicon Valley

Shape of Input

BITTIGER

- First, concatenate all characters



- batch_len = data_len // batch_size (300 = 30010 // 100)

Reshape

BITTIGER

```
tf.reshape(raw_data[0:batch_size * batch_len],
                    [batch_size,  batch_len])
```

Drop last 10 words

千山鸟飞绝 … 白华照寒水 … 落叶满长安

reshape

batch_len

batch_size

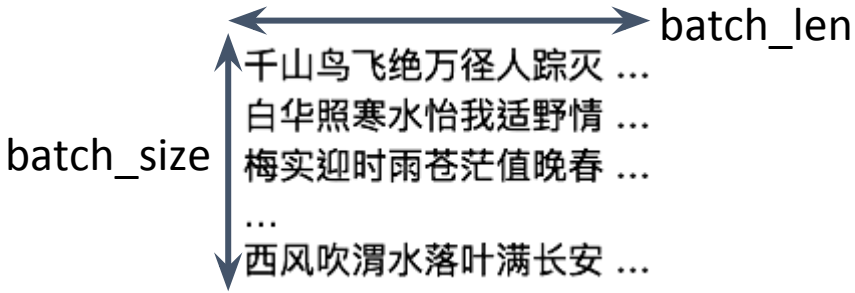
千山鸟飞绝万径人踪灭 …
白华照寒水怡我适野情 …
梅实迎时雨苍茫值晚春 …
…
西风吹渭水落叶满长安 …

Batch

BITTIGER

- `epoch_size = (batch_len - 1) // num_steps`

  59 = (300-1) // 5



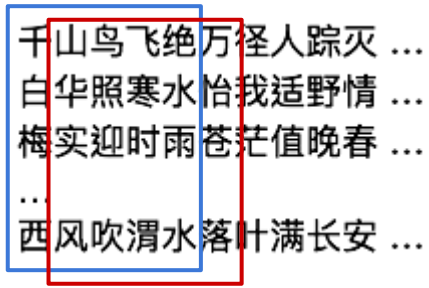

```
i = tf.train.range_input_producer(epoch_size, shuffle=False).dequeue()
i = 0, 1, 2, …, 58
```

Batch

BITTIGER

```
i = 0, 1, 2, …, 58
x = data[:, i*num_steps   :(i+1)*num_steps]
y = data[:, i*num_steps+1:(i+1)*num_steps+1]
```

```
x1 = [0:100, 0*5 : 1*5]
y1 = [0:100, 0*5+1 : 1*5+1]
```

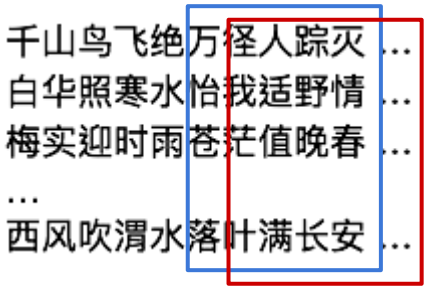


Batch

BITTIGER

```
i = 0, 1, 2, …, 58
x = data[:, i*num_steps   :(i+1)*num_steps]
y = data[:, i*num_steps+1:(i+1)*num_steps+1]
```

```
x2 = [0:100, 1*5 : 2*5]
y2 = [0:100, 1*5+1 : 2*5+1]
```



Batch

BITTIGER

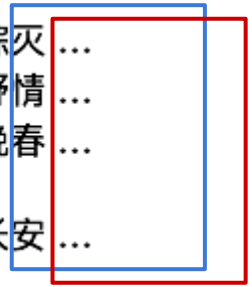
```
i = 0, 1, 2, …, 58
x = data[:, i*num_steps   :(i+1)*num_steps]
y = data[:, i*num_steps+1:(i+1)*num_steps+1]
```

290:295

```
x59 = [0:100, 58*5:59*5]
y59 = [0:100, 58*5+1:59*5+1]
```

291:296

千山鸟飞绝万径人踪灭 …
白华照寒水怡我适野情 …
梅实迎时雨苍茫值晚春 …
…
西风吹渭水落叶满长安 …



- Last part of data that cannot form a batch will be abandoned.

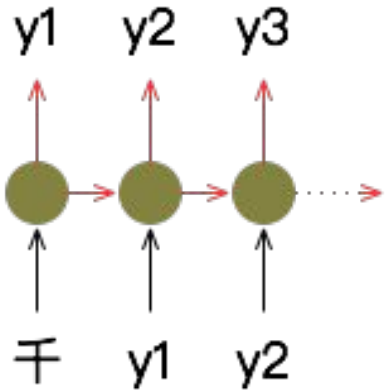
RNN Feed

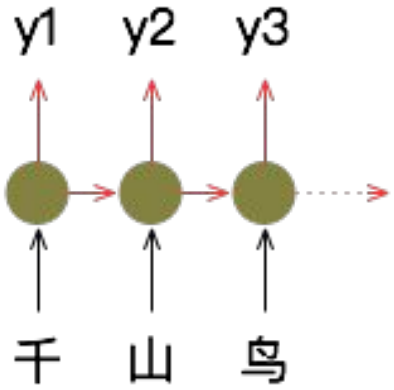
BITTIGER

The Lifelong Learning Platform of Silicon Valley

Feed During Training

BITTIGER

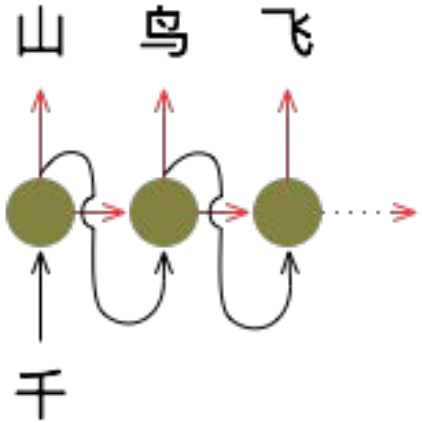




- Less information for the model when predicting > y2 words
- More information
- Simulate the condition where the model can predict the correct word

After Training

BITTIGER



- When the model is well-trained, it should produce correct words on its own

Model: variables and ops

BITTIGER

The Lifelong Learning Platform of Silicon Valley

Variables and ops

BITTIGER

- A layer may contain variables and operations

  variable
  ```
  embedding_weights = tf.get_variable("embedding",
                                      [vocab_size, embedding_size])
  ```

  op
  ```
  embed_inputs =
    tf.nn.embedding_lookup(embedding_weights, input)
  ```

- Or just op, no variable

  ```
  tf.nn.dropout(embed_inputs, ......)
  ```

Can use the same name

BITTIGER

- A layer may contain variables and operations

  variable `embedding_weights = tf.get_variable("embedding", [vocab_size, embedding_size])`

  op `embed_inputs = tf.nn.embedding_lookup`

- Or just op, no variable

  `tf.nn.dropout(embed_inputs, ...`

You can use the same name to connect output of one layer to the input of another layer. (optional)

Step-by-step Building

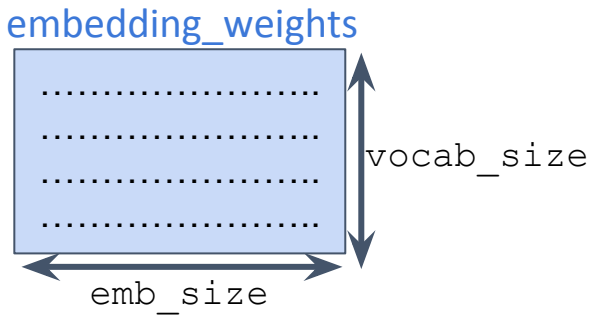
BITTIGER

- We will demonstrate with variables and operations
- Understand how to go from input to output

Building Blocks

BITTIGER

1. Create variable if needed



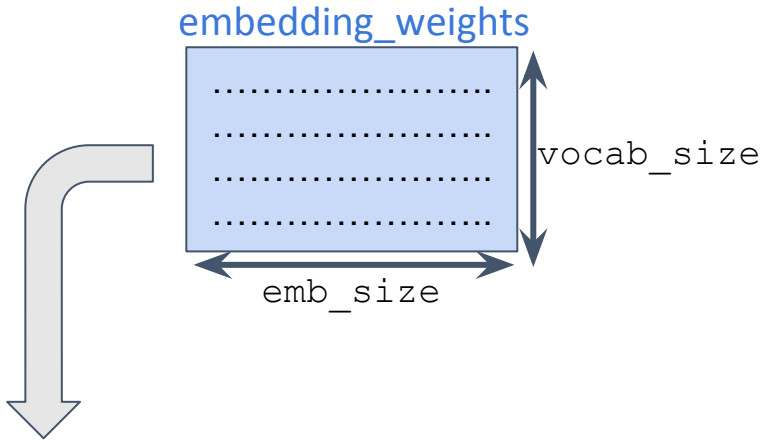
embedding_weights =

```
tf.get_variable("embedding",
        [vocab_size, embedding_size])
```

Building Blocks

BITTIGER

1. Create variable if needed
2. Add op that uses the variable

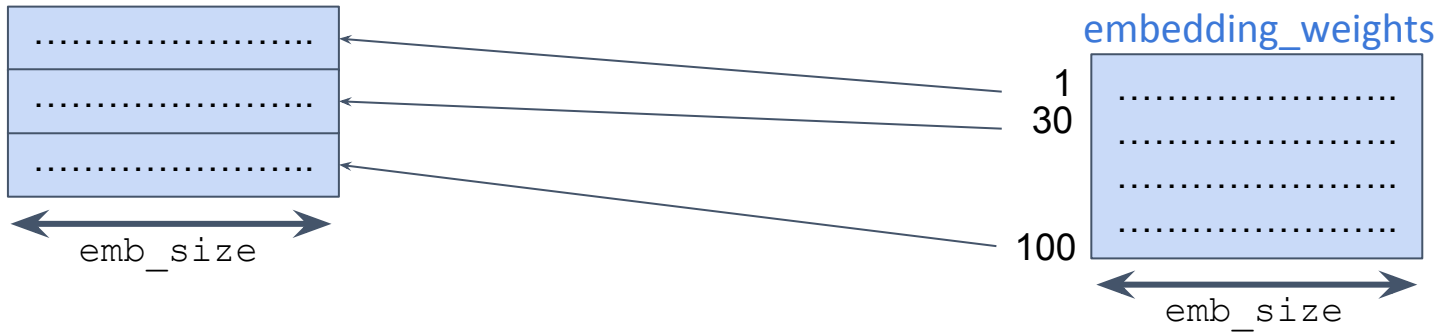



```
embedding_lookup(embedding_weights, input)
```

Building Blocks

BITTIGER

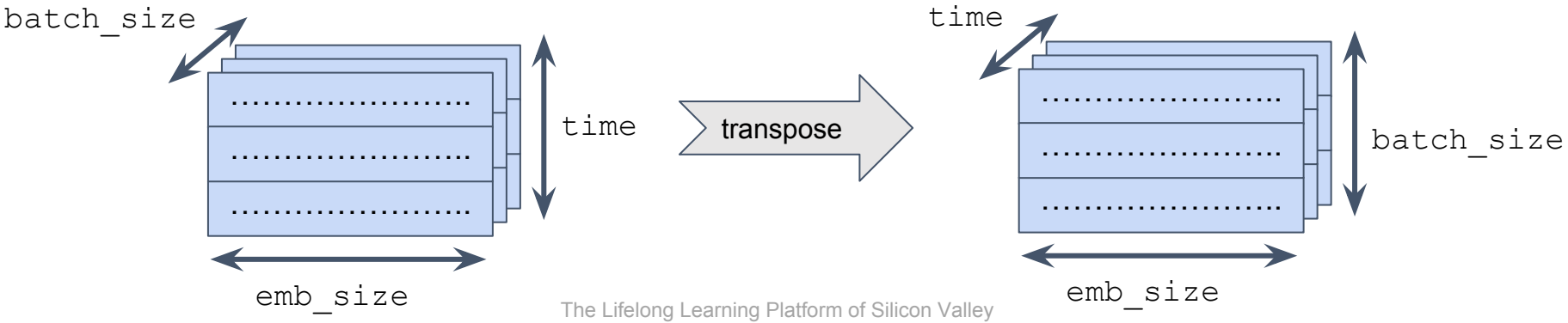
```
embedding_lookup(embedding_weights, [1,30,100])
```



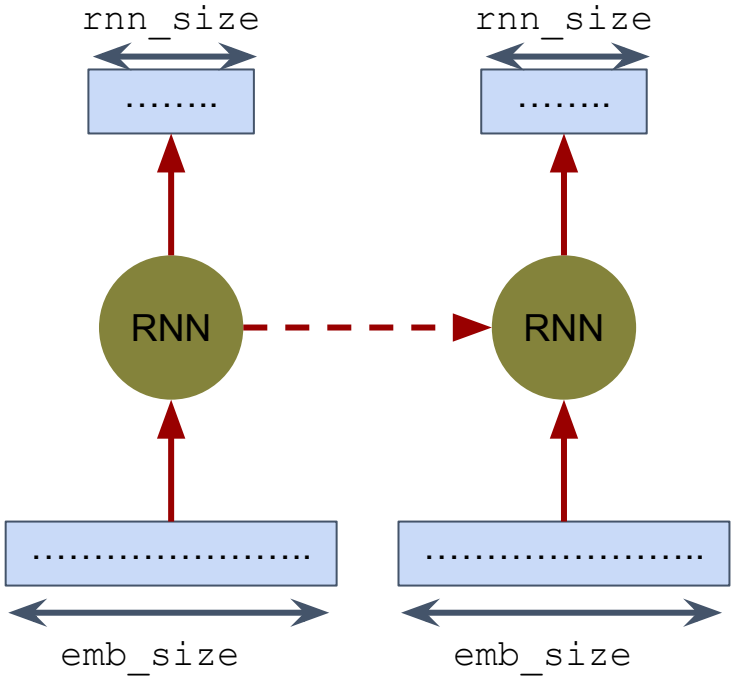
Building Blocks

BITTIGER

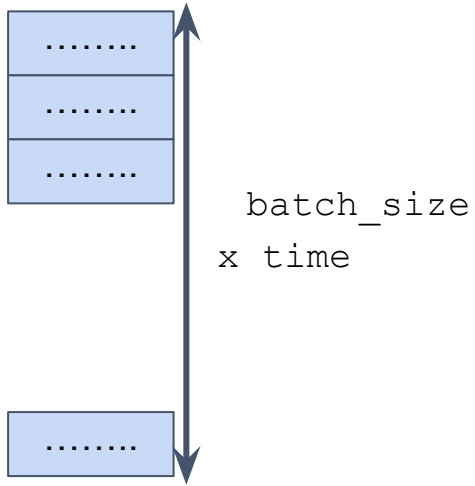
1. Create variable if needed
2. Add op that uses the variable
3. Transpose/reshape if needed before sending to the next layer



Building Blocks



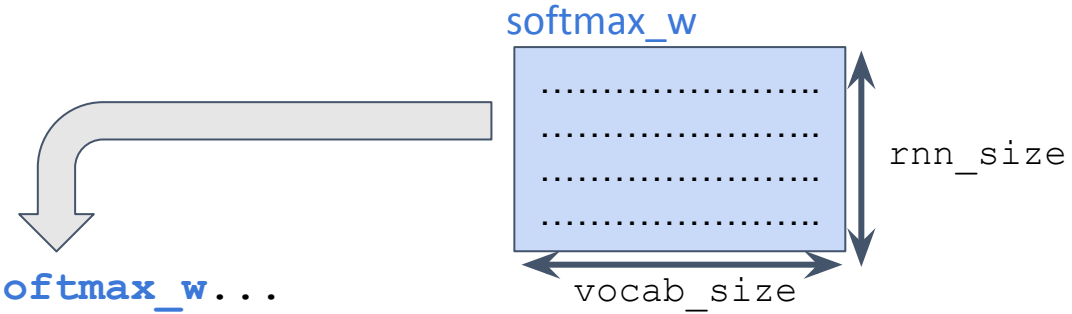
- Merge 1 and 2 dimension of RNN layer output



Building Blocks

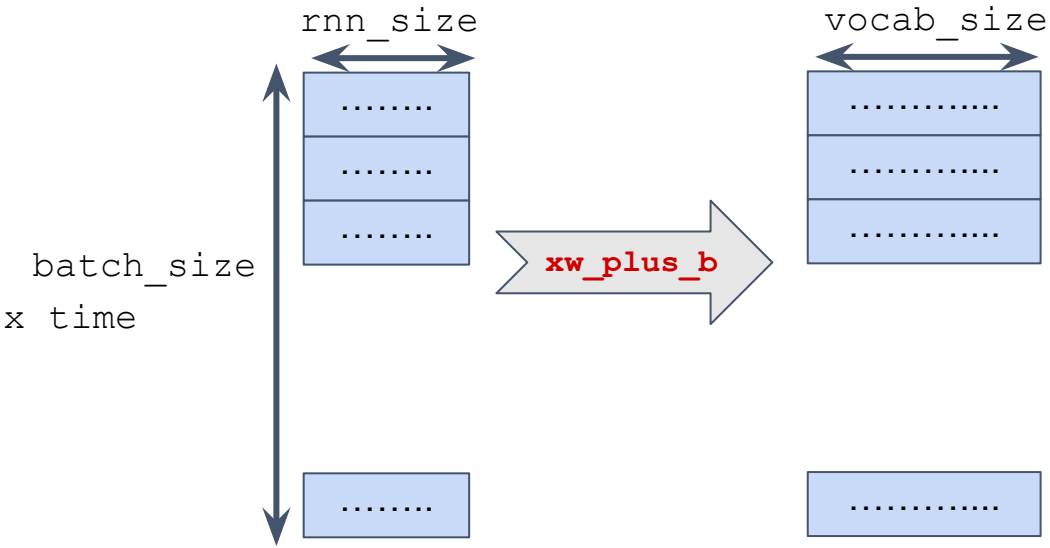
BITTIGER

1. Create variable if needed
2. Add op that uses the variable
3. Transpose/reshape if needed before sending to the next layer

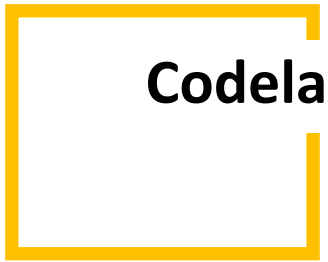



```
xw_plus_b(rnn_output,softmax_w...
```

Building Blocks



- Represent probability of word



Codelab

BITTIGER

The Lifelong Learning Platform of Silicon Valley

Attention

BITTIGER

The Lifelong Learning Platform of Silicon Valley

Concept

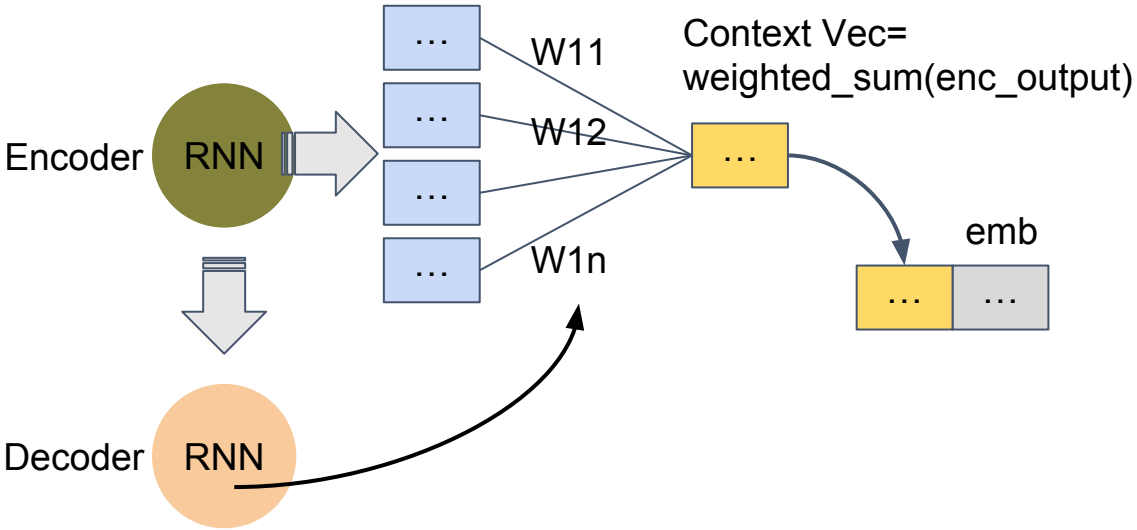
BITTIGER

1. Calculate weight over encoder outputs

   Using current decoder memory
2. Weighted sum of encoder outputs, “context vector”
3. Merge context with decoder output (word)
4. Send to decoder for next decoding step
5. Back to 1.

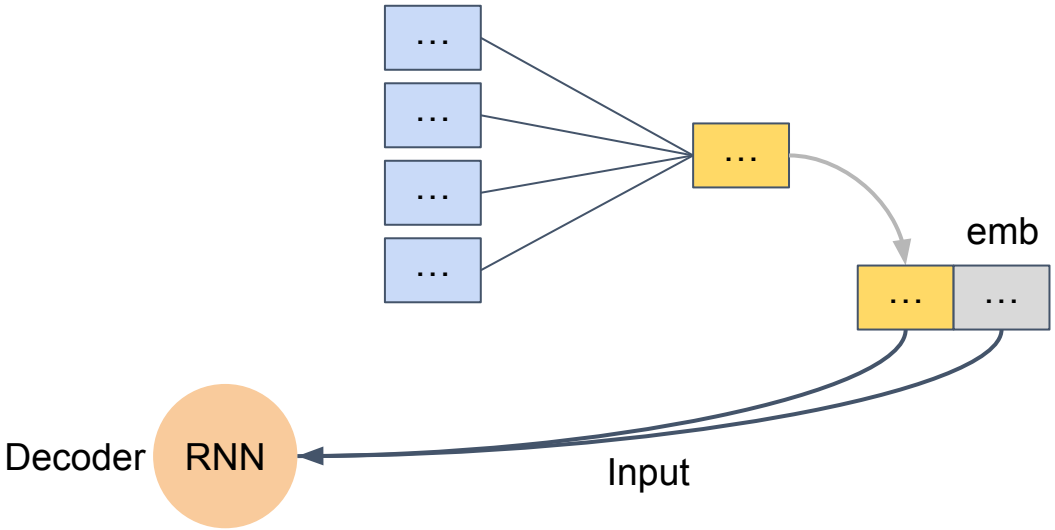
Attention

BITTIGER



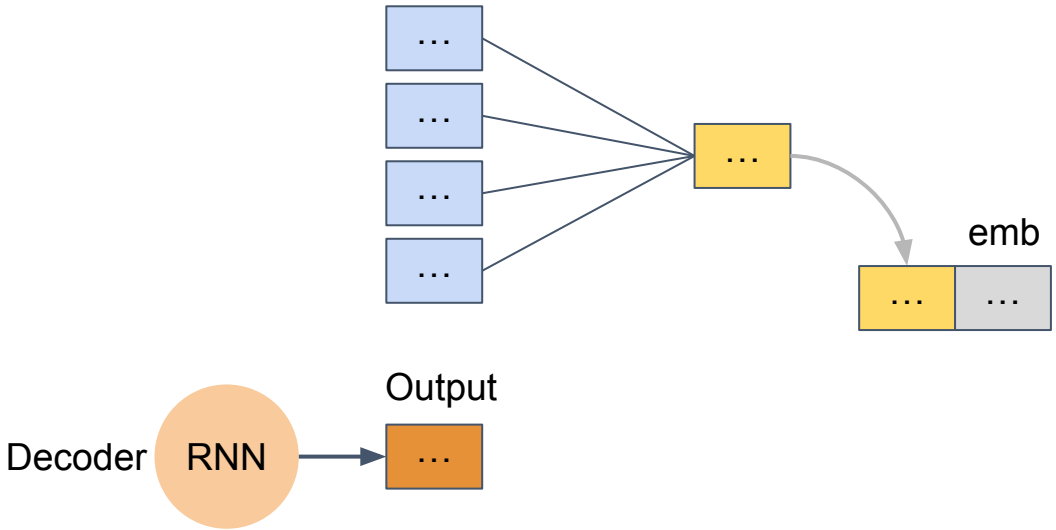
Attention

BITTIGER



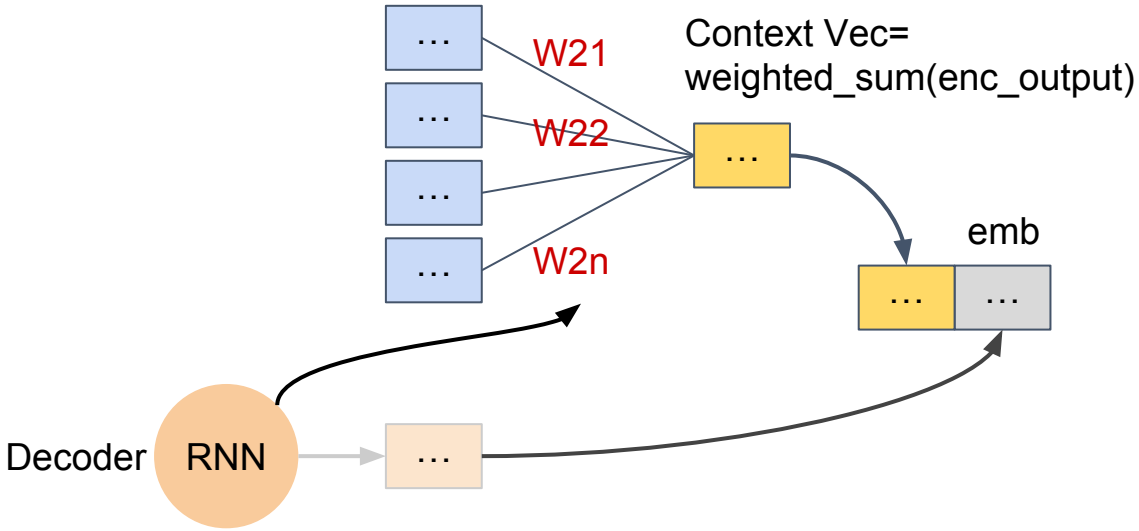
Attention

BITTIGER



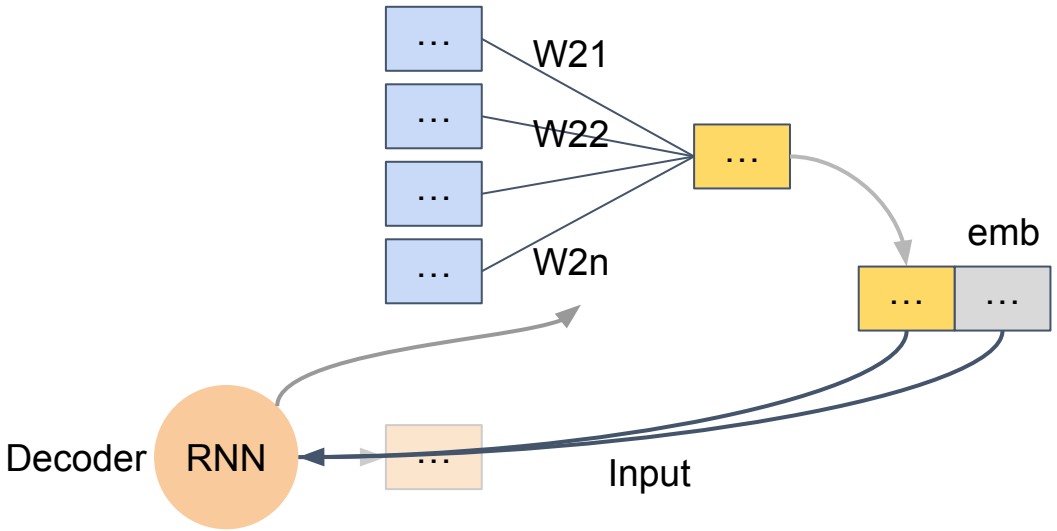
Attention

BITTIGER



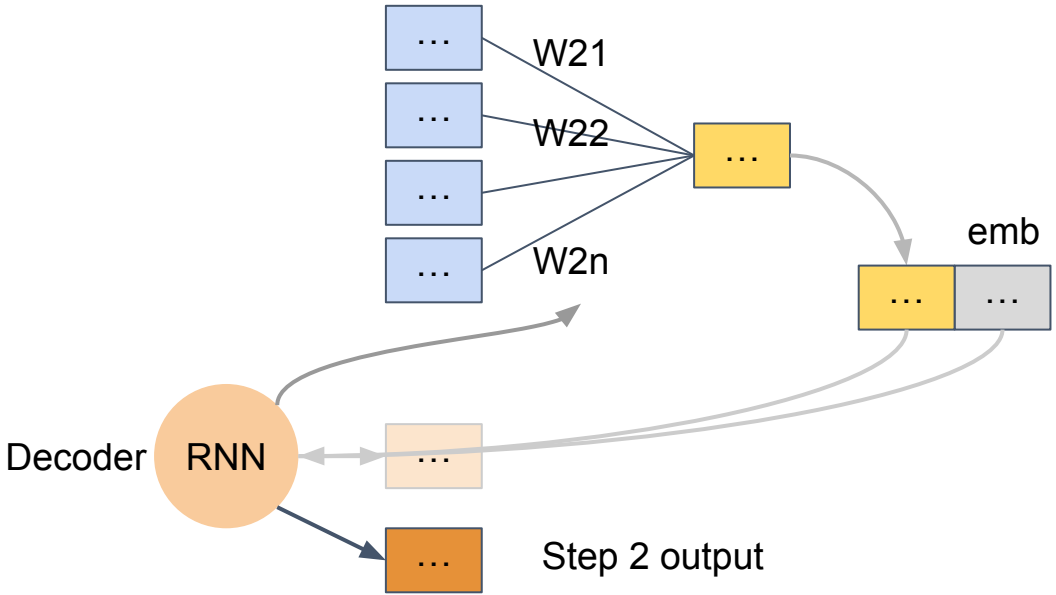
Attention

BITTIGER



Attention

BITTIGER



Definition

BITTIGER

1. Calculate weight over encoder outputs

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_1}\boldsymbol{h}_t + \boldsymbol{W_2}\bar{\boldsymbol{h}}_s\right)$$

2. Normalize weights

$$\alpha_{ts} = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'=1}^{S} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)}$$

3. Sum to get “context vector”

$$\boldsymbol{c}_t = \sum_s \alpha_{ts} \bar{\boldsymbol{h}}_s$$