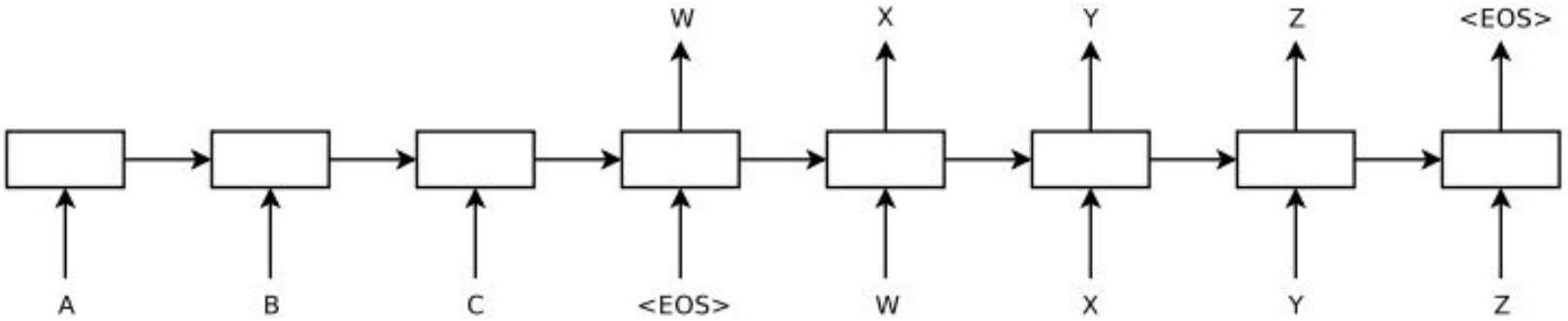
DeepPavlov: Seq2Seq Tutorial

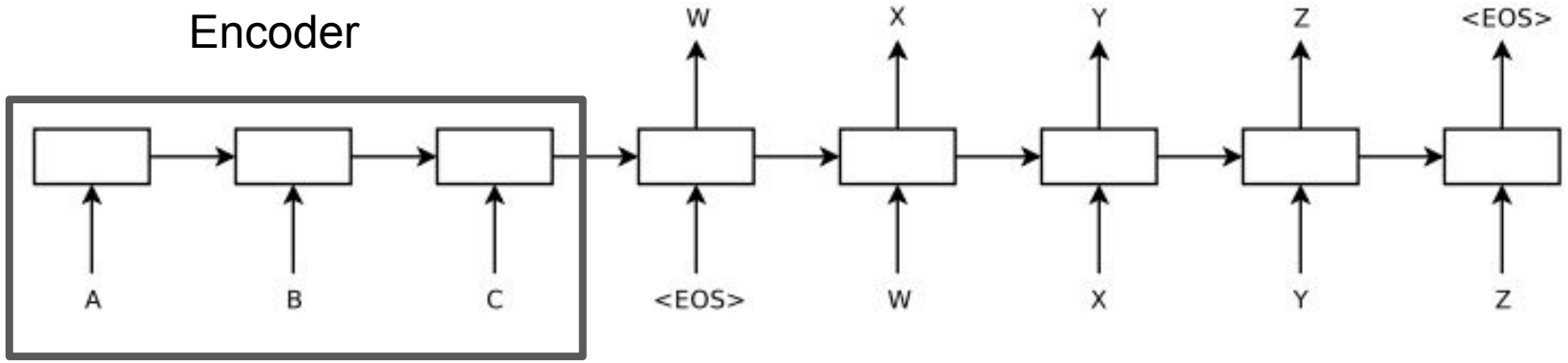
Kuratov Yury

Sequence-to-sequence



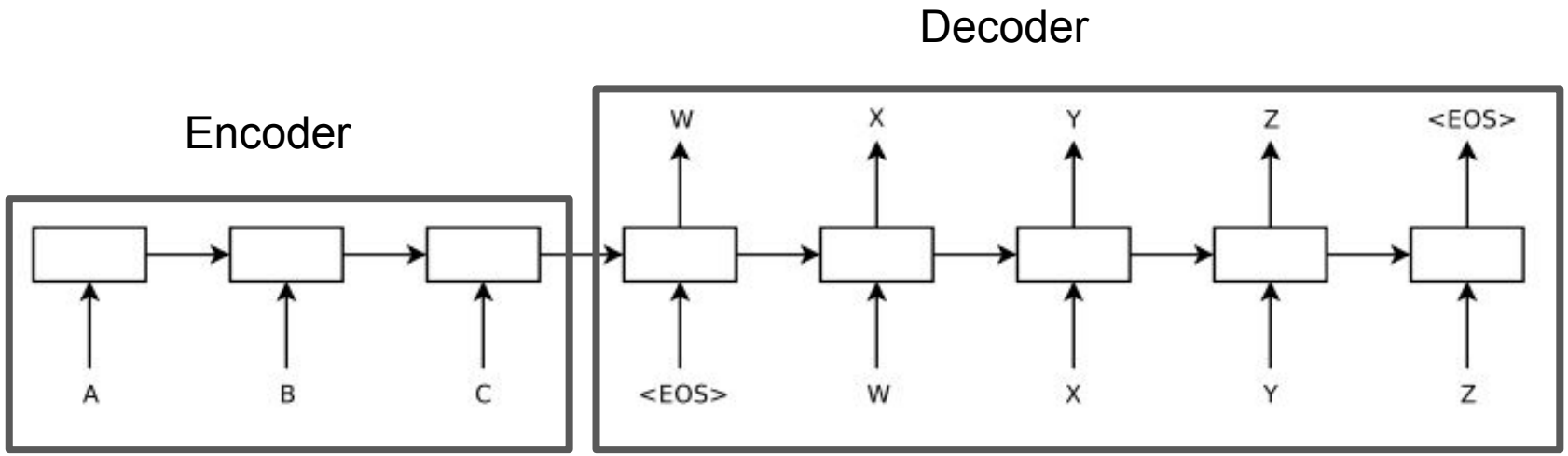
Sequence to Sequence Learning with Neural Networks https://arxiv.org/abs/1409.3215

Sequence-to-sequence



Sequence to Sequence Learning with Neural Networks https://arxiv.org/abs/1409.3215

Sequence-to-sequence



Sequence to Sequence Learning with Neural Networks https://arxiv.org/abs/1409.3215

Modern sequence-to-sequence

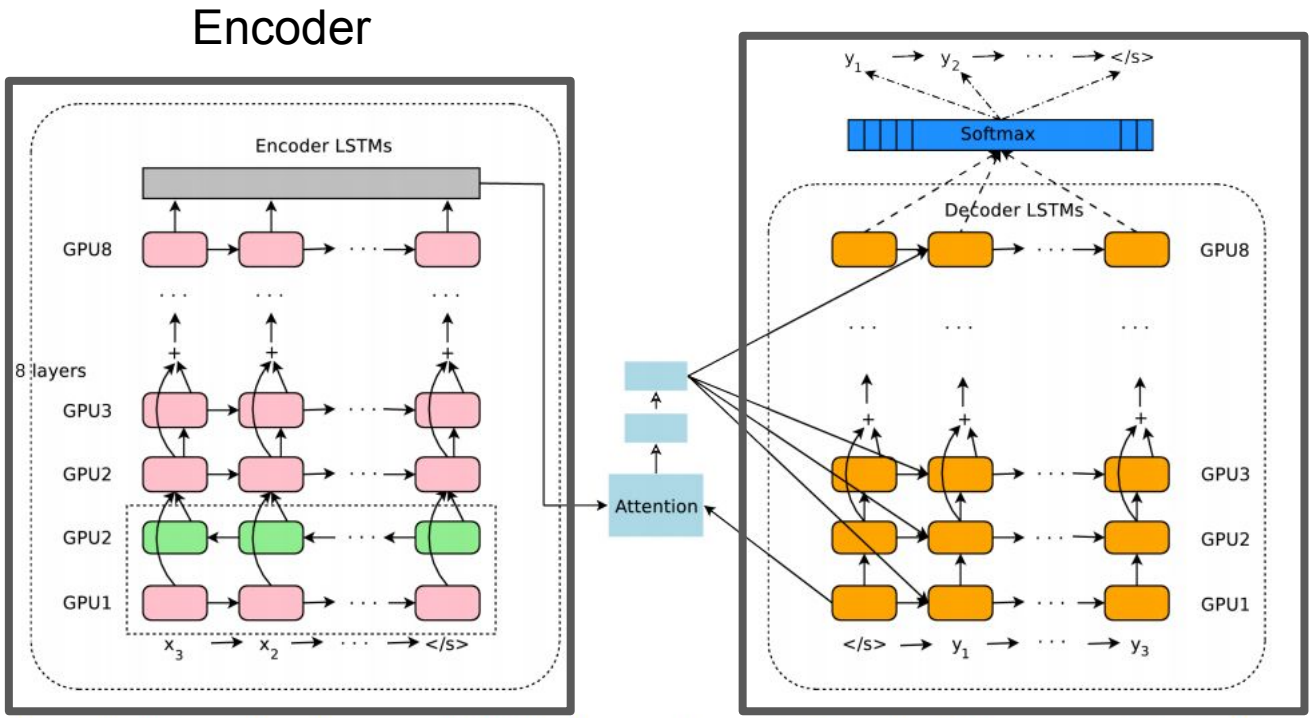
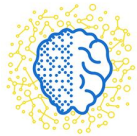


Figure 1: The model architecture of GNMT, Google's Neural Machine Translation system. On the left

Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation
https://arxiv.org/abs/1609.08144

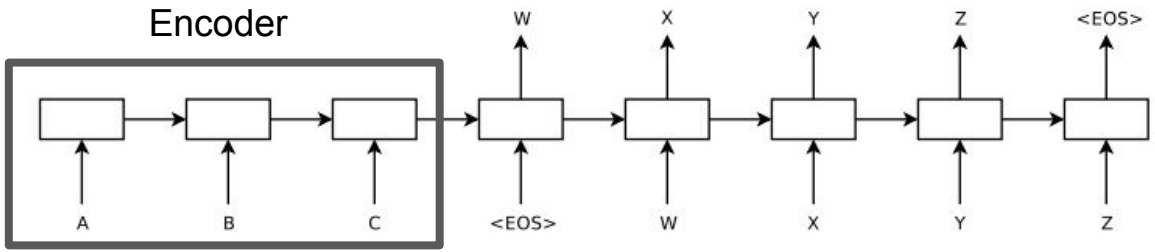
Sequence-to-sequence tutorial structure



- **Implemented:**
  - **DatasetReader** to read the data
  - **DatasetIterator** to generate batches
  - **Vocabulary** to convert words to indexes
  - and some other components for pre- and postprocessing
- **To implement:**
  - **Encoder** to encode input sequence
  - **Decoder** to produce output sequence
  - **Teacher Forcing** to help model during training time
  - **Attention mechanism** to give access to all encoder states during decoding

Sequence-to-sequence: encoder

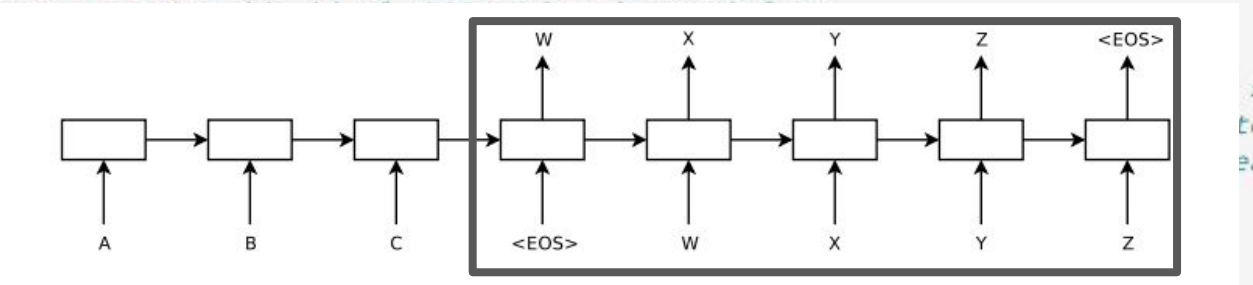
```python
def encoder(inputs, inputs_len, embedding_matrix, cell_size, keep_prob=1.0):
    # inputs: tf.int32 tensor with shape bs x seq_len with token ids
    # inputs_len: tf.int32 tensor with shape bs
    # embedding_matrix: tf.float32 tensor with shape vocab_size x vocab_dim
    # cell_size: hidden size of recurrent cell
    # keep_prob: dropout keep probability

    # YOUR CODE HERE

    return encoder_outputs, encoder_state
```



Sequence-to-sequence: decoder

```
def decoder(encoder_outputs, encoder_state, embedding_matrix, mask,
            cell_size, max_length, y_ph,
            start_token_id=1, keep_prob=1.0,
            teacher_forcing_rate_ph=None,
            use_attention=False, is_train=True):
    # decoder
    # encoder_outputs: tf.float32 tensor with shape bs x seq_len x encoder_cell_size
    # encoder_state: tf.float32 tensor with shape bs x encoder_cell_size
    # embedding_matrix: tf.float32 tensor with shape vocab_size x vocab_dim
    # mask: tf.int32 tensor with shape bs x seq_len with zeros for masked sequence elements
    # cell_size: hidden size of recurrent cell
    # max_length: max length of output sequence
```

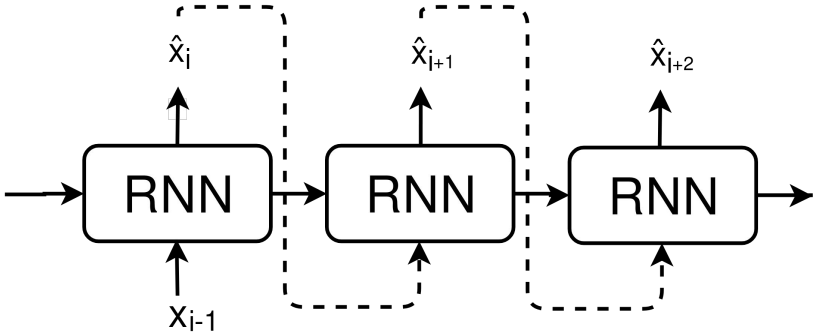
Decoder



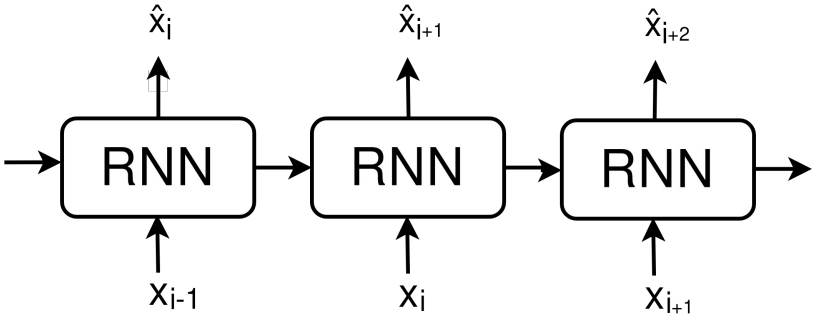

```
    return y_pred_tokens, y_logits
```

Teacher forcing

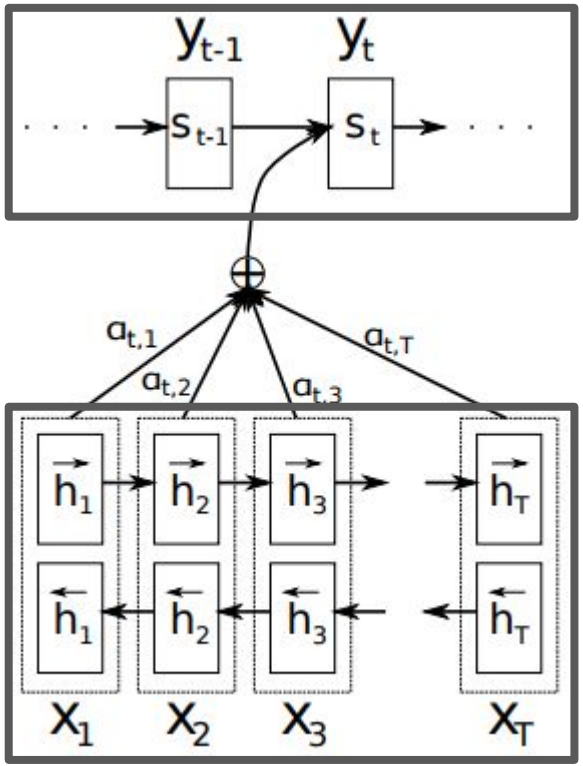
- Using model output



- Teacher forcing



Attention mechanism

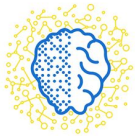


Decoder

Encoder

Neural Machine Translation by Jointly Learning to Align and Translate https://arxiv.org/abs/1409.0473

Attention mechanism



```python
def dot_attention(memory, state, mask, scope="dot_attention"):
    # inputs: bs x seq_len x hidden_dim
    # state: bs x hidden_dim
    # mask: bs x seq_len

    # YOUR CODE HERE

    return att, att_weights
```

The context vector c_i is, then, computed as a weighted sum of these annotations h_i:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j. \tag{5}$$

The weight α_{ij} of each annotation h_j is computed by

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})},$$

where

$$e_{ij} = a(s_{i-1}, h_j)$$

Neural Machine Translation by Jointly Learning to Align and Translate https://arxiv.org/abs/1409.0473

Let’s put it all together!

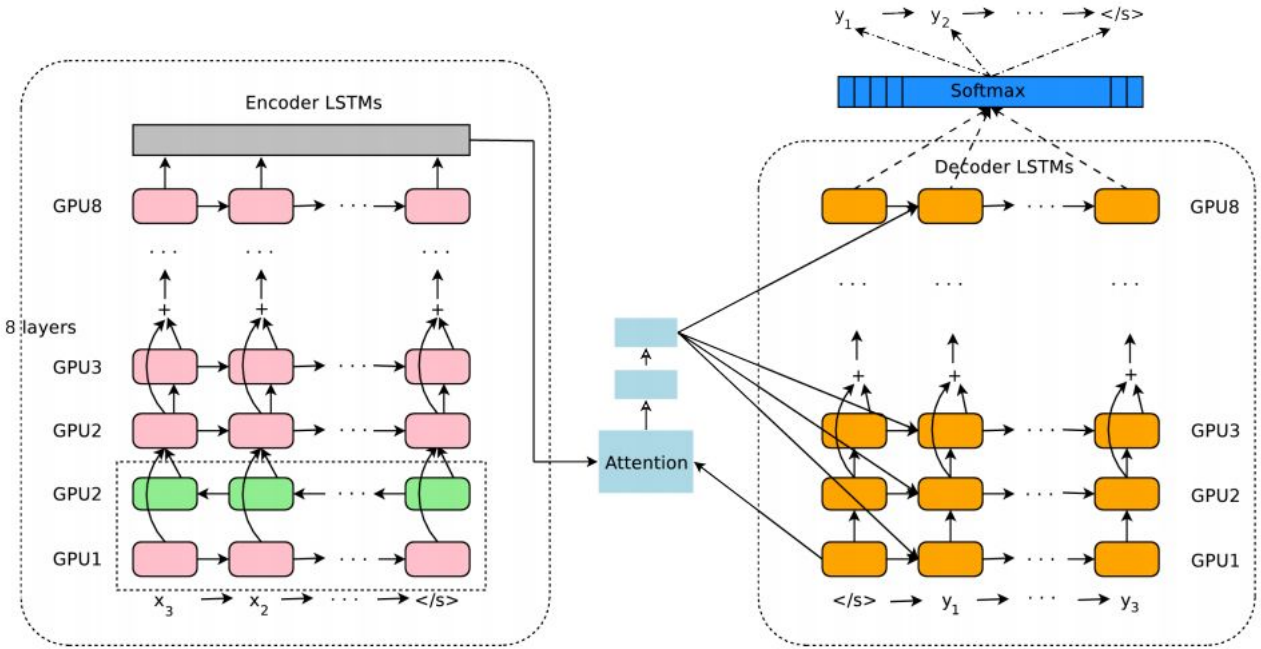


Figure 1: The model architecture of GNMT, Google’s Neural Machine Translation system. On the left

Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation
https://arxiv.org/abs/1609.08144