4. RNN: Q&A session

CS-E4890: Deep Learning

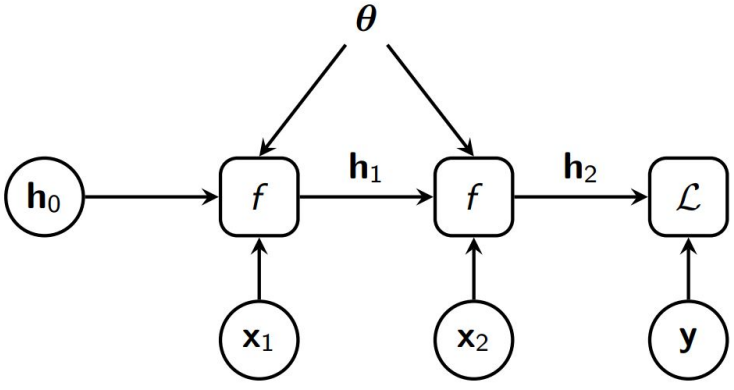
Aalto University
School of Science

Kalle Kujanpää

24.03.2023

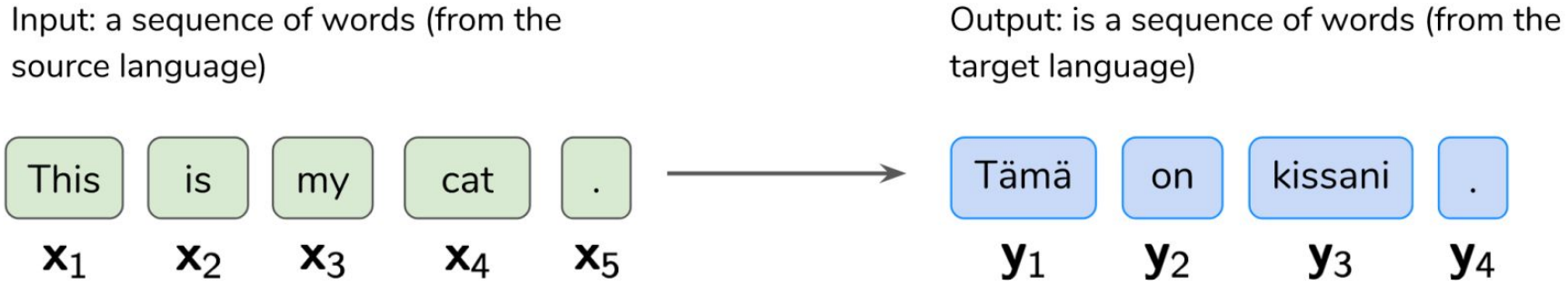
RNNs: Short Recap

- **A family of NNs for handling sequential data with variable length inputs and outputs**
- **Traditionally used for**
  - Natural language processing
  - Speech recognition
  - Time series prediction
  - Reinforcement learning

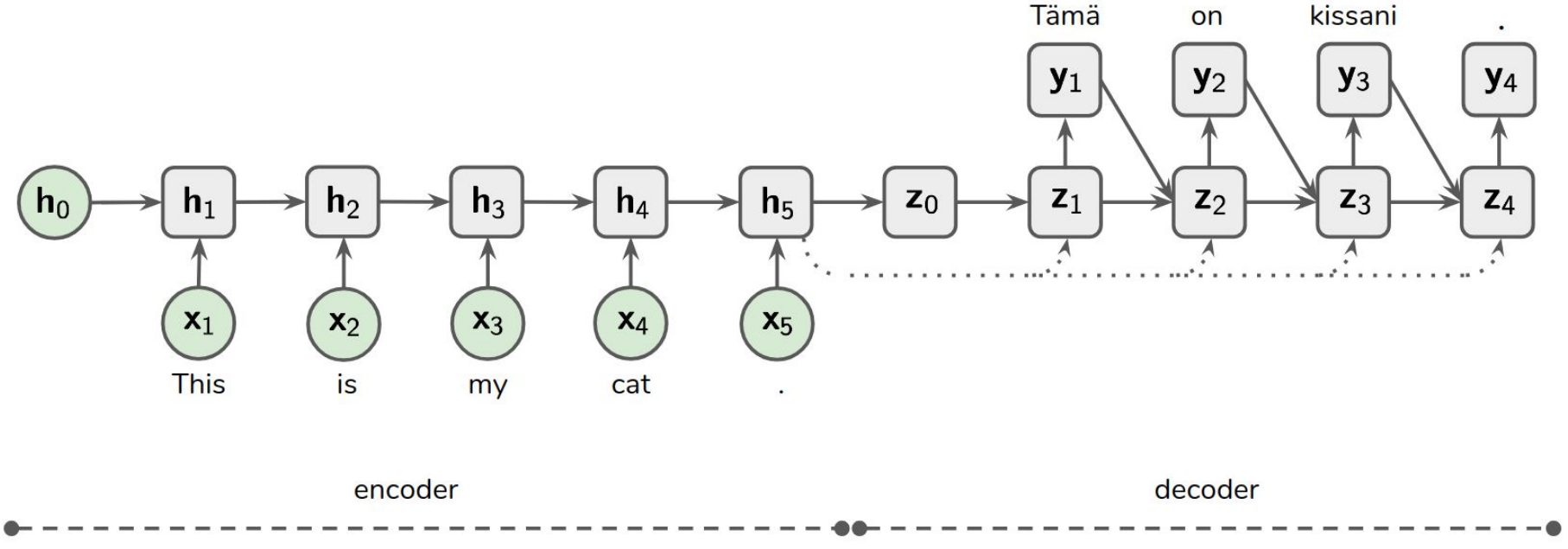


Neural Machine Translation

- **Translate a sentence from source language to target language**
- **A sequence-to-sequence model, where input and output sequences may be of different lengths**



Sequence-to-sequence model

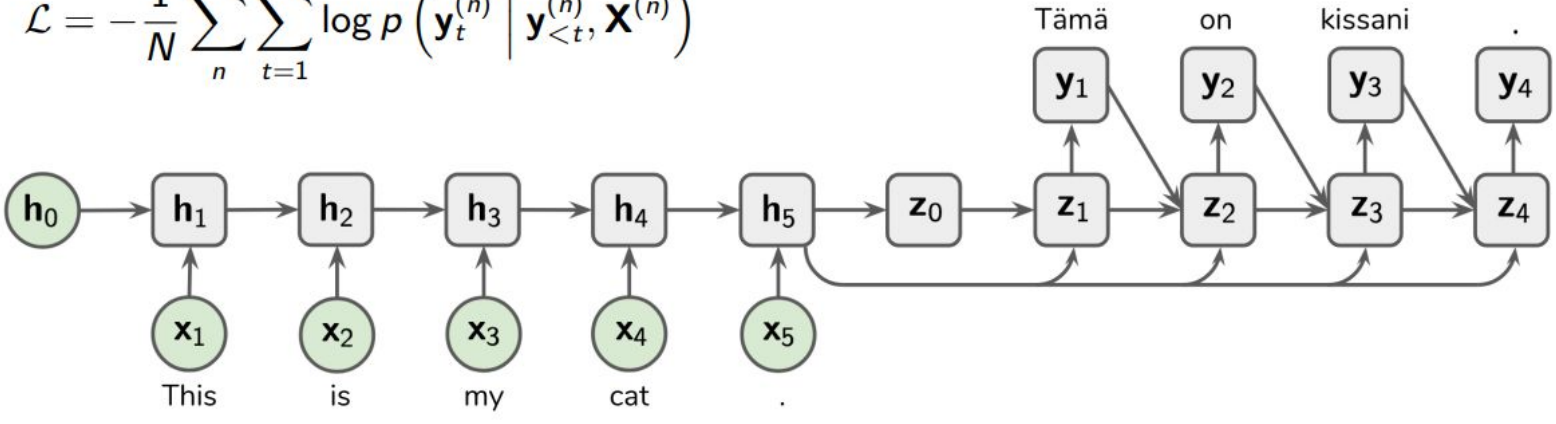


Training the Model

- Minimize the negative log-likelihood of the output sequence

□ A categorical distribution over the words at each step

$$\mathcal{L} = -\frac{1}{N} \sum_{n} \sum_{t=1}^{T_n} \log p\left(\mathbf{y}_t^{(n)} \,\middle|\, \mathbf{y}_{<t}^{(n)}, \mathbf{X}^{(n)}\right)$$



RNNs in PyTorch (from lecture)

- There are two way to build a computational graph with RNNs in PyTorch.
- In simple cases, the whole sequence can be processed with one call:

```
h = torch.zeros(...)
h = rnn.forward(x, h)
```

- In more difficult cases, you need to build a graph with a for-loop:

```
h = torch.zeros(...)
for x_t in x:
    h = rnn.forward(x_t, h)
```

- The initial states of RNNs are often initialized with zeros.

Word Embeddings (from lecture)

- A simple word representation is one-hot vector. Word i is represented with vector $\mathbf{z}$ such that $z_i = 1$, $z_{j \neq i} = 0$.
- Better representaion:
  - represent each word i as a vector $\mathbf{w}_i$
  - treat all vectors $\mathbf{w}_i$ as model parameters and tune them in the training procedure
  - this is equivalent to $\mathbf{Wz}$ where $\mathbf{W}$ is a matrix of word embeddings (word vectors $\mathbf{w}_i$ in its columns).
- This is implemented in `torch.nn.Embedding(num_embeddings, embedding_dim)`
  - `num_embeddings` is the size of the dictionary
  - `embedding_dim` is the size of each embedding vector $\mathbf{w}_i$

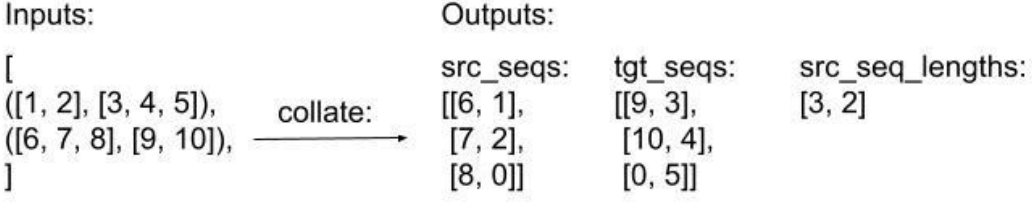
04_rnn: Dataset

- **French English translation**
- **Each word is represented by an integer index**
  - 4489 French words
  - 2925 English words
  - 8682 Sentences

```
Source sentence: "vous etes depourvues d ambition . EOS"
Sentence as tensor of word indices:
tensor([ 118,  215, 1542,  234, 1209,    5,    1])
Target sentence: "you re unambitious . EOS"
Sentence as tensor of word indices:
tensor([130,  78, 669,   4,   1])
```

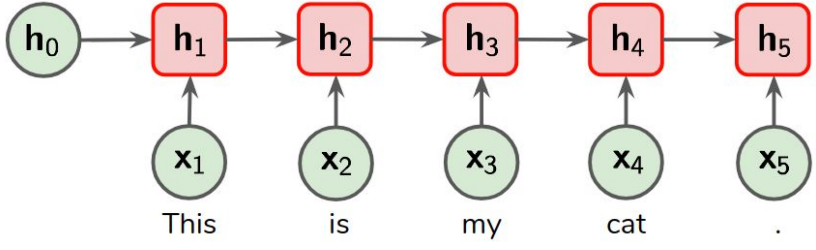
04_rnn: 1st task – collate

```
def collate(list_of_samples):
    """Merges a list of samples to form a mini-batch.

    Args:
      list_of_samples is a list of tuples (src_seq, tgt_seq):
          src_seq is of shape (src_seq_length,)
          tgt_seq is of shape (tgt_seq_length,)

    Returns:
      src_seqs of shape (max_src_seq_length, batch_size): Tensor of padded source sequences.
          The sequences should be sorted by length in a decreasing order, that is src_seqs[:,0] should be
          the longest sequence, and src_seqs[:,-1] should be the shortest.
      src_seq_lengths: List of lengths of source sequences.
      tgt_seqs of shape (max_tgt_seq_length, batch_size): Tensor of padded target sequences.
    """
```



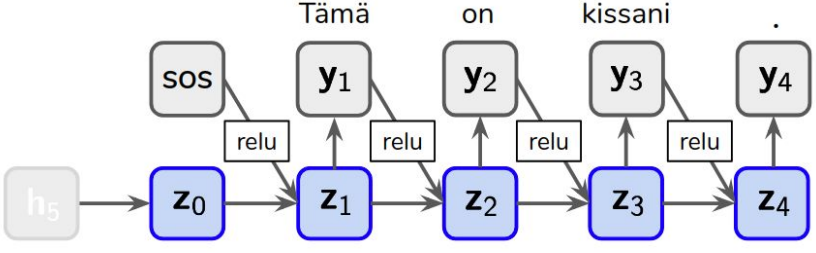
04_rnn: 2nd task – encoder

```
def forward(self, pad_seqs, seq_lengths, hidden):
    """
    Args:
      pad_seqs of shape (max_seq_length, batch_size): Padded source sequences.
      seq_lengths: List of sequence lengths.
      hidden of shape (1, batch_size, hidden_size): Initial states of the GRU.

    Returns:
      outputs of shape (max_seq_length, batch_size, hidden_size): Padded outputs of GRU at every step.
      hidden of shape (1, batch_size, hidden_size): Updated states of the GRU.
    """
```



04_rnn: 3rd task – decoder

```
def forward(self, hidden, pad_tgt_seqs=None, teacher_forcing=False):
    """
    Args:
      hidden of shape (1, batch_size, hidden_size): States of the GRU.
      pad_tgt_seqs of shape (max_out_seq_length, batch_size): Tensor of words (word indices) of the
          target sentence. If None, the output sequence is generated by feeding the decoder's outputs
          (teacher_forcing has to be False).
      teacher_forcing (bool): Whether to use teacher forcing or not.

    Returns:
      outputs of shape (max_out_seq_length, batch_size, tgt_dictionary_size): Tensor of log-probabilities
          of words in the target language.
      hidden of shape (1, batch_size, hidden_size): New states of the GRU.
```



04_rnn: 4th task – training loop

- **Loss function: implement log-probabilities**
- **Can be run on a CPU, GPU is faster**
  - I would recommend staying on Jupyter
- **When computing the loss, ignore the padded values**

$$L = -\frac{1}{N} \sum_{n} \sum_{t=1}^{T_n} \log p\left(\mathbf{y}_t^{(n)} \,\middle|\, \mathbf{y}_{<t}^{(n)}, \mathbf{X}^{(n)}\right)$$

04_rnn: 5th task – translation

```python
def translate(encoder, decoder, pad_src_seqs, src_seq_lengths):
    """Translate sequences from the source language to the target language using the trained model.

    Args:
      encoder (Encoder): Trained encoder.
      decoder (Decoder): Trained decoder.
      pad_src_seqs of shape (max_src_seq_length, batch_size): Padded source sequences.
      src_seq_lengths: List of source sequence lengths.

    Returns:
      out_seqs of shape (MAX_LENGTH, batch_size): LongTensor of word indices of the output sequences.
    """
    # YOUR CODE HERE
    raise NotImplementedError()
```