University of Zurich[UZH]

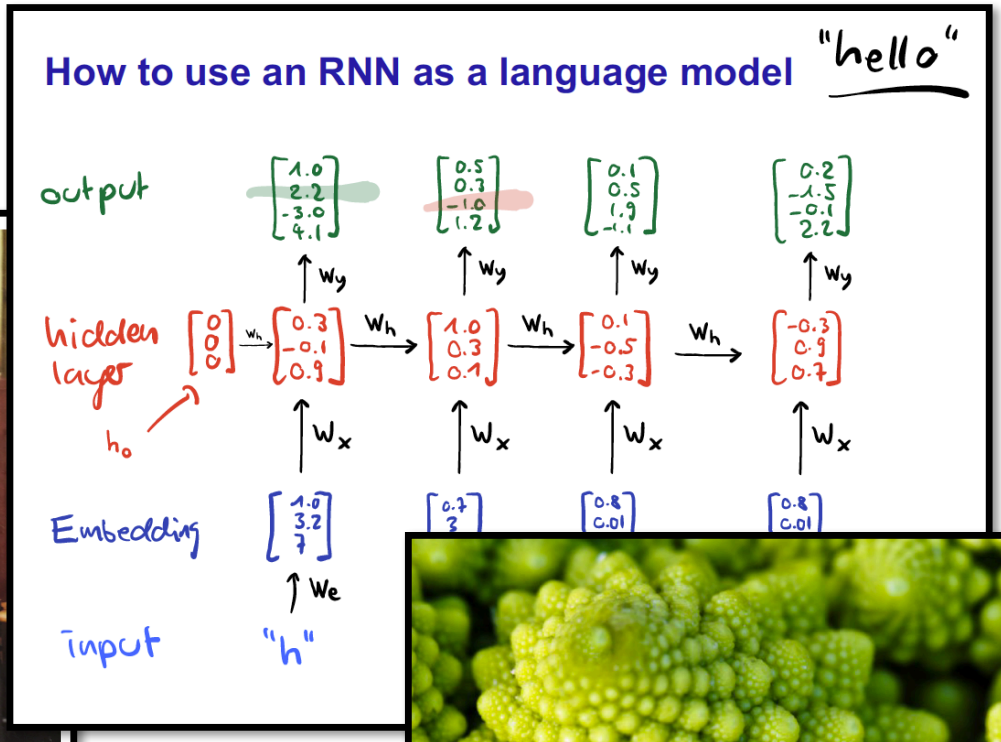
Institute of Computational Linguistics

Machine Translation

10 Encoder-Decoder Models

Mathias Müller

Last time


How to use an RNN as a language model
"hello"
output
hidden layer
Embedding
input
"h"
h_0
W_y
W_h
W_x
W_e

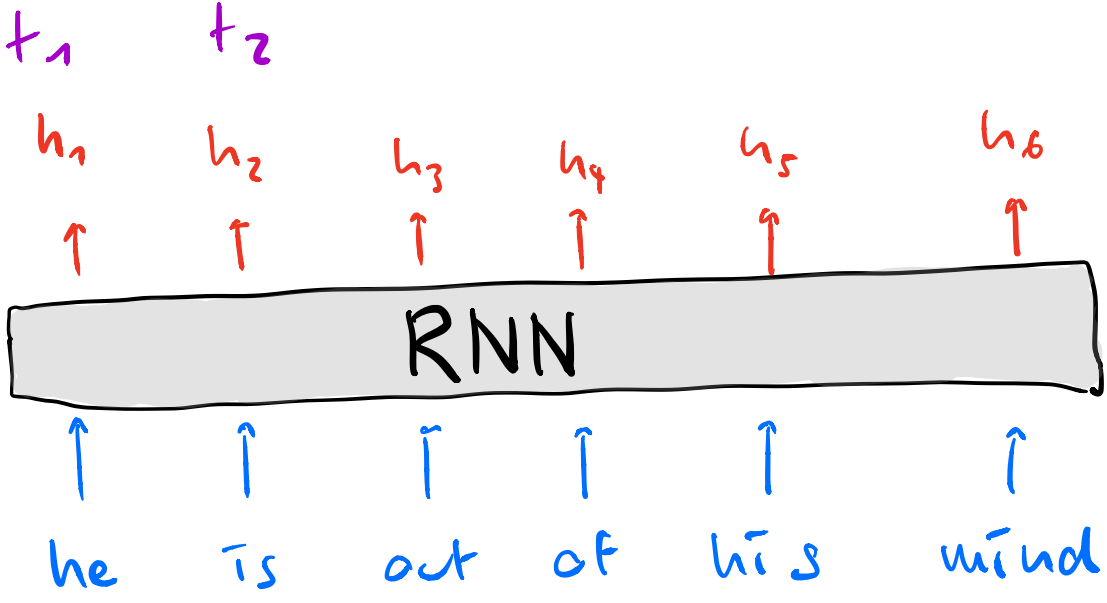
Topics of this lesson

- **Encoder-Decoder Models**: How to use RNNs for machine translation
- **Main weaknesses** of simple encoder-decoder models
- **Daikon**: Our educational NMT tool, written in Tensorflow

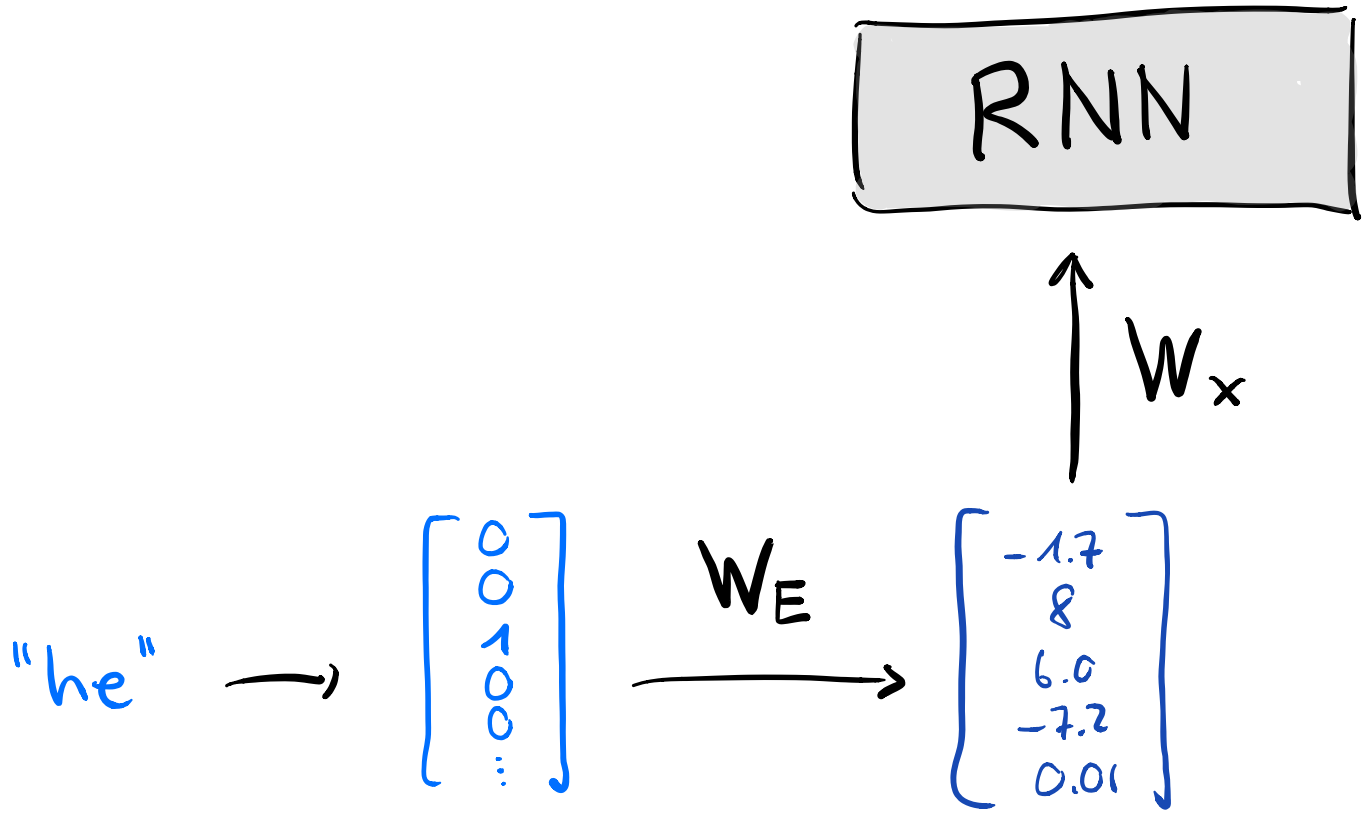
How to use RNNs for machine translation

RNNs high-level review

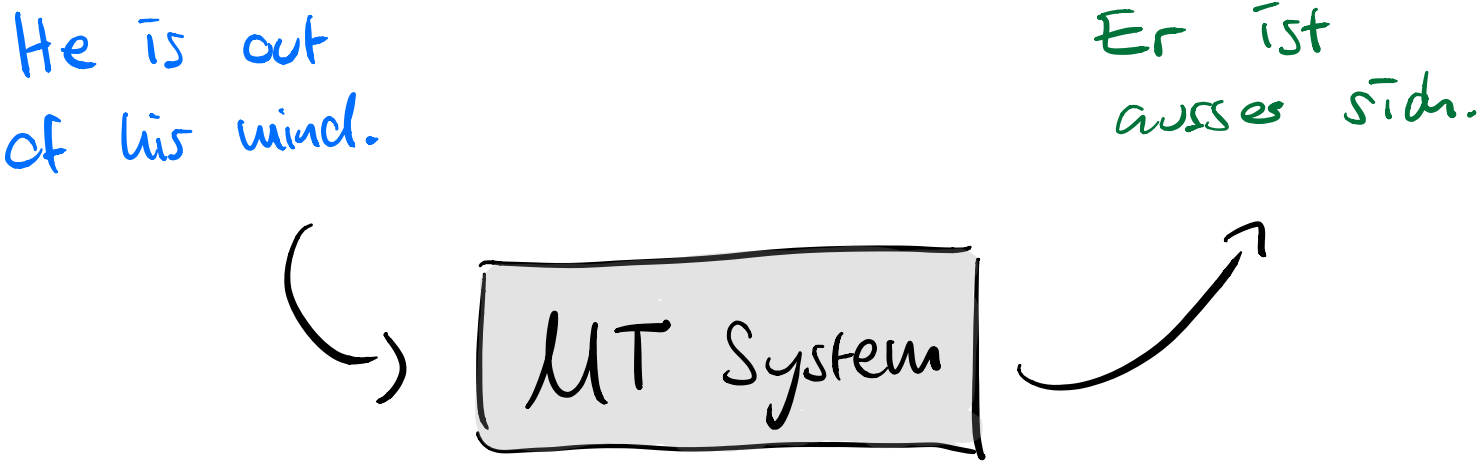
- Read or write sequences, or both
- Compute a new hidden state at each time step



Text input high-level review

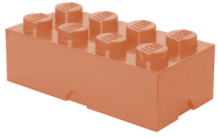


Machine translation: problem definition

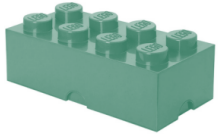


How to build machine translation with blocks we already know?

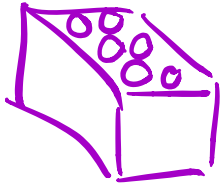
FFNNs



Embeddings



RNNs



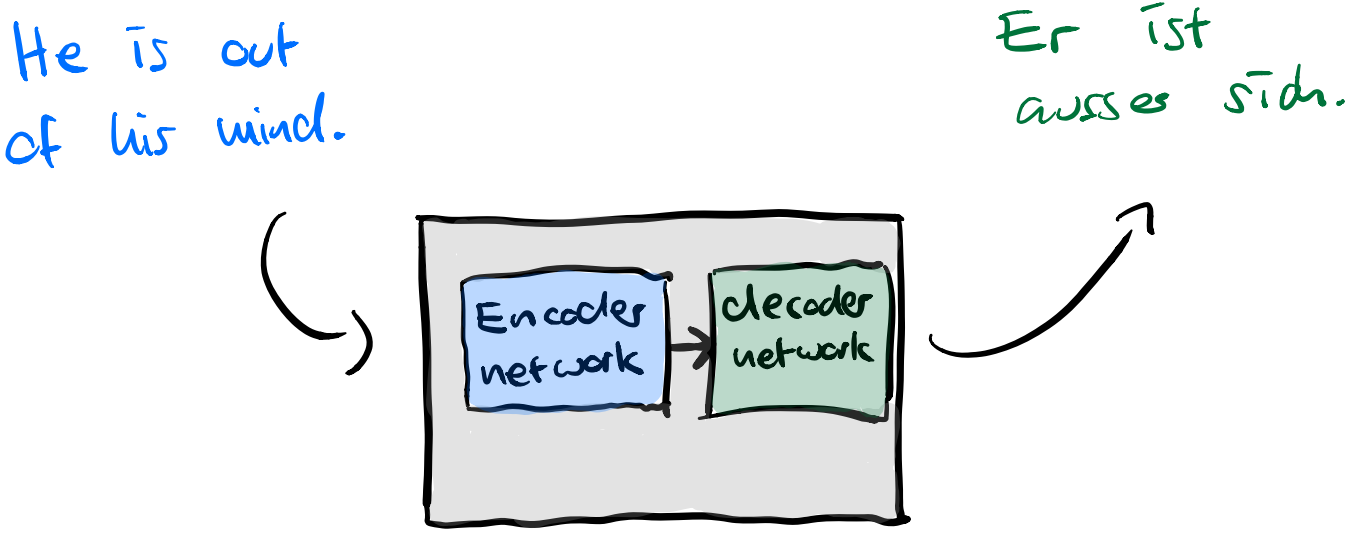
parallel data

https://bit.ly/2T0GDeK

Encoder-decoder Approach

Use 2 RNNs:

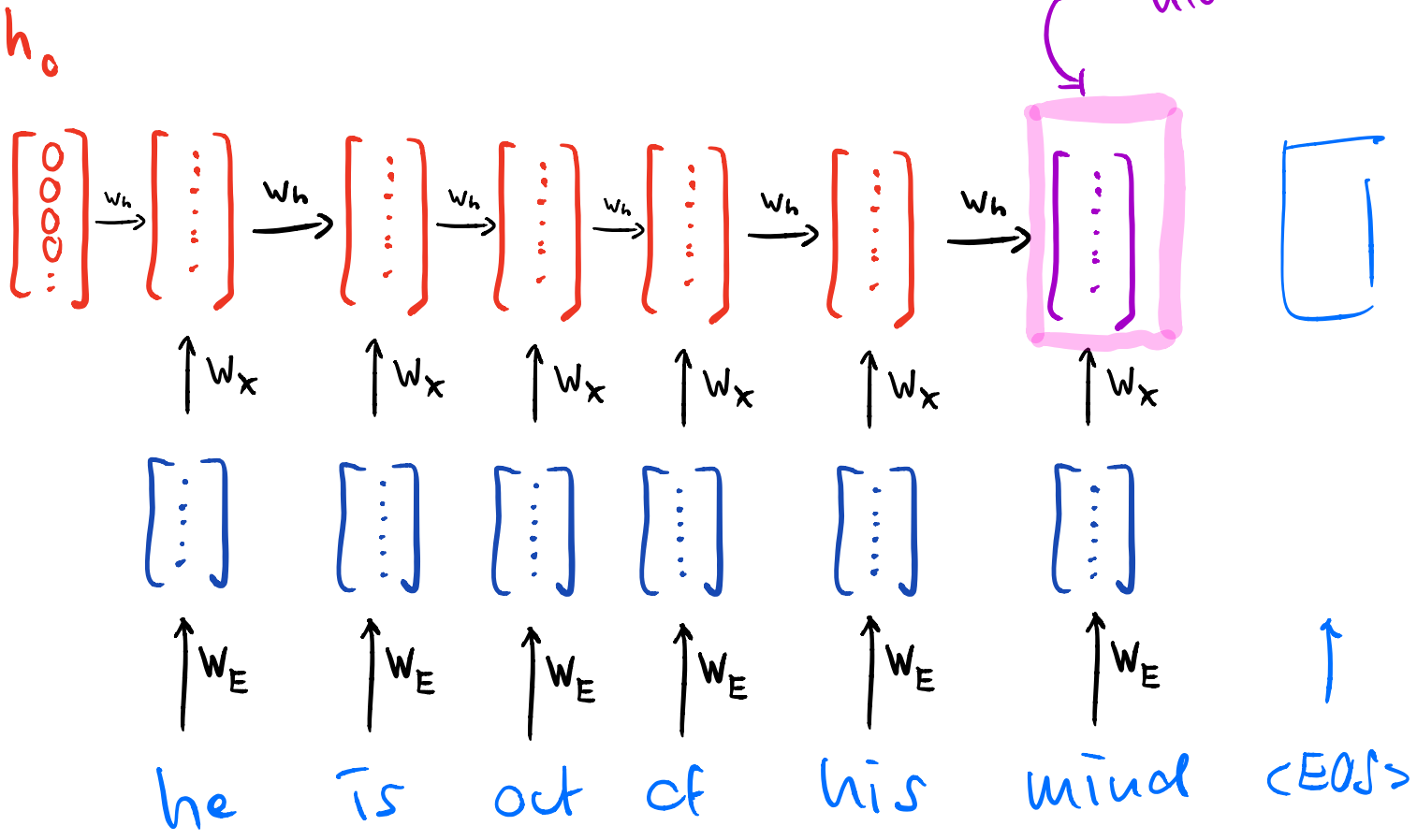
- **Encoder** summarizes the source
- **Decoder** produces the target

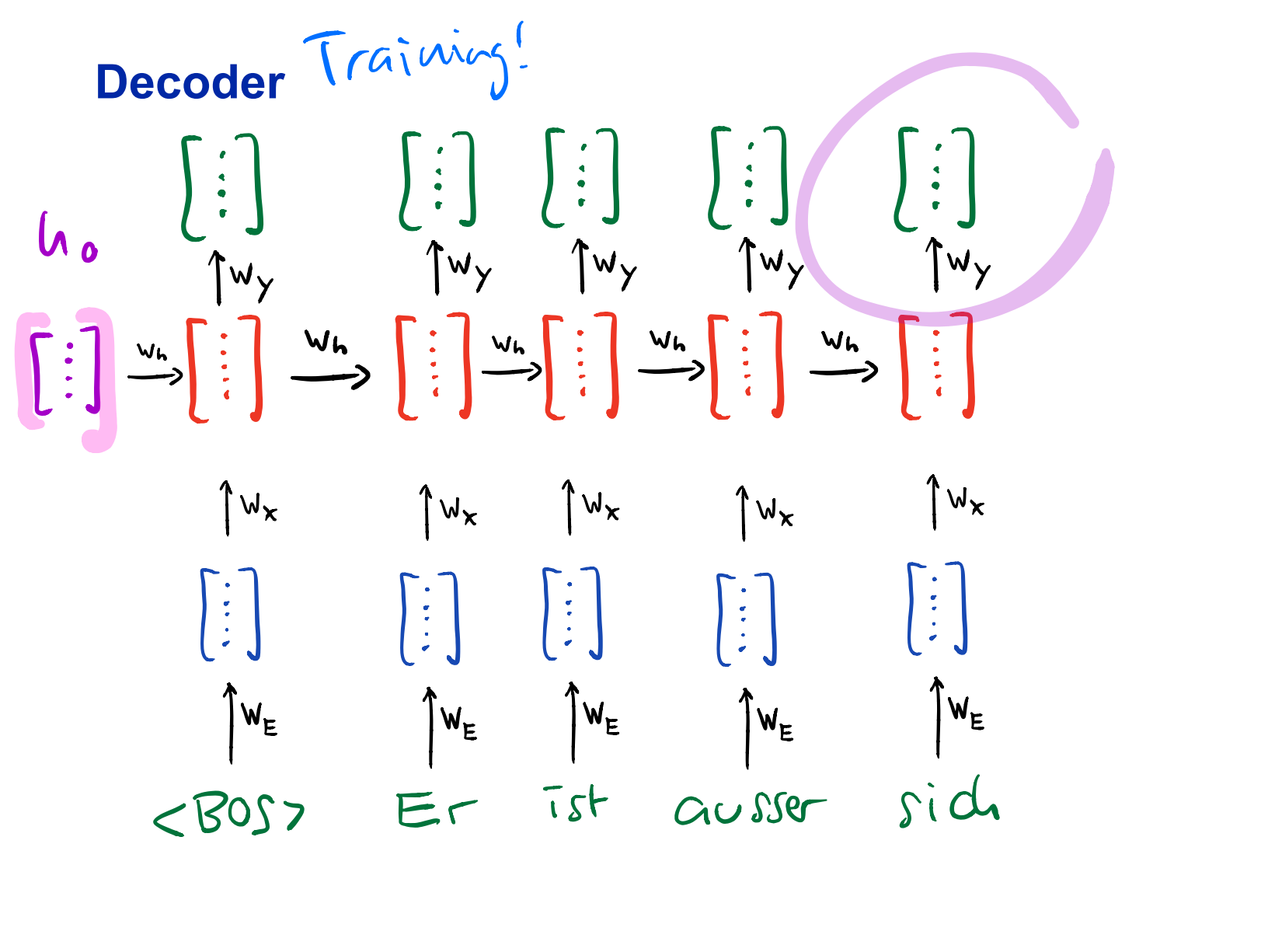


Encoder

1024
256 512

last encoder hidden state




Decoder
Training!
h_0
W_y
W_h
W_x
W_E
<BOS>
Er
ist
ausser
sich

Summary Endoder-Decoder Model

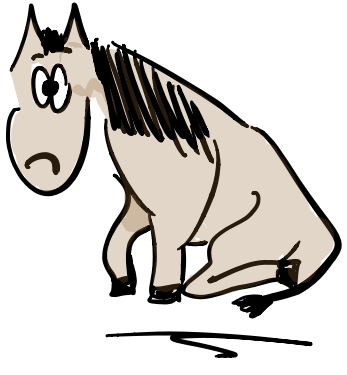
- Key idea: use two separate RNNs,
  - **Encoder** to summarize ("encode") the source sentence
  - **Decoder** to generate the target sentence
- Use last hidden state of encoder to initialize hidden state of decoder

Main Weaknesses of Encoder-Decoder Models

Sometimes NMT results are sad

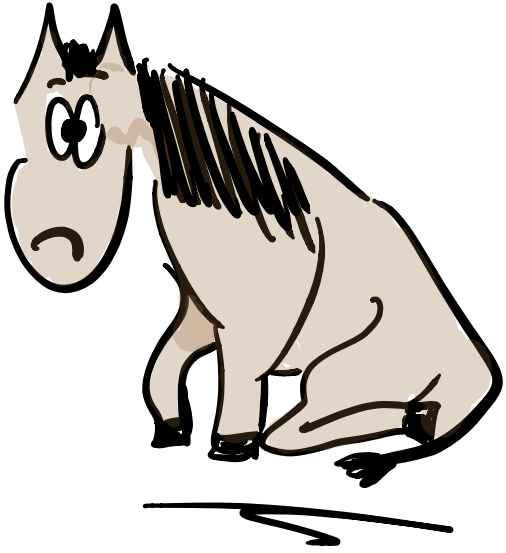
WorldViews

In Ukraine, Google translates Russia as 'Mordor' and top diplomat's name as 'sad little horse'



Two main reasons Enc-Dec results are sad

- Long sentences
- Open, infinite vocabulary

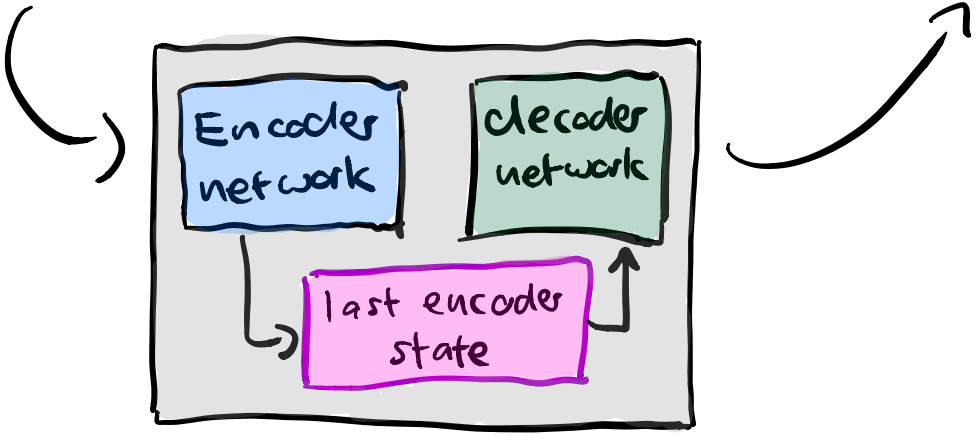


“Curse of sentence length”

Remember: last encoder state is the only information available to the decoder

He is out of his mind.

Er ist ausser sich.



Curse of sentence length

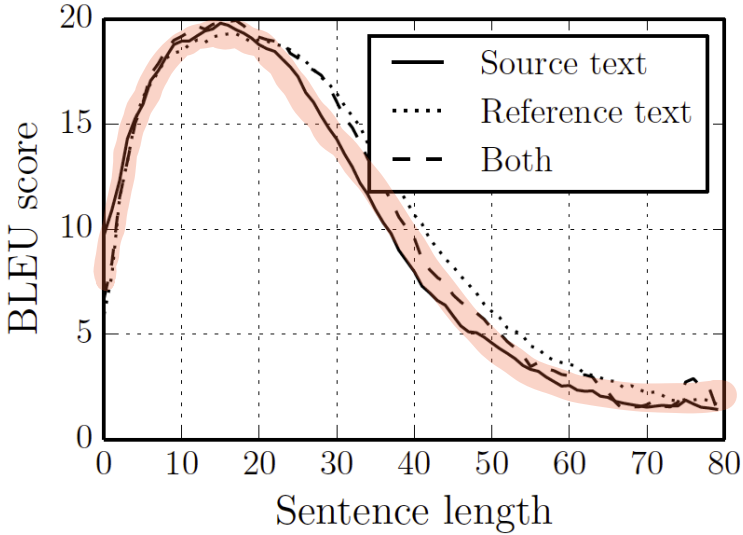
You can't cram the meaning of a single $&!#* sentence into a single $!#&* vector!

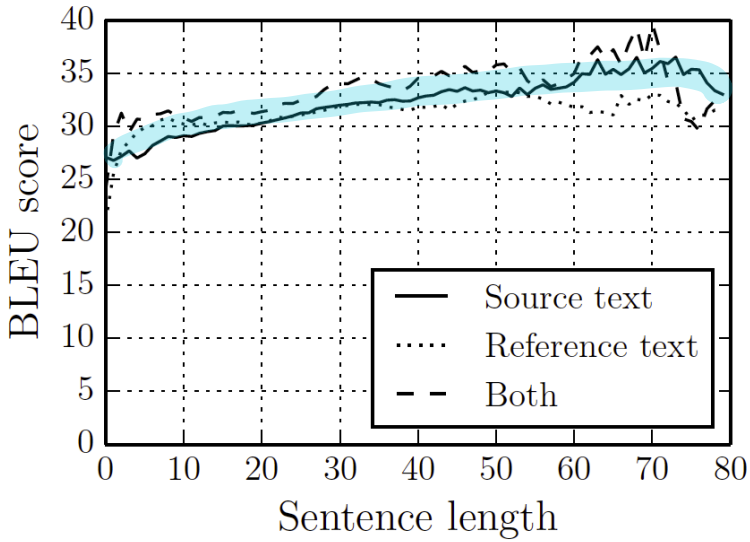
"You can't." →

"You can't cram the meaning..." →

Curse of sentence length

Cho et al. 2014 * In 2014





NMT*

SMT

Vocabulary size

~ 1m = E: 256m

Important observations about vocabulary:

- Vocabulary must be decided at training time, and be quite small

  Common sizes : 50k - 100k

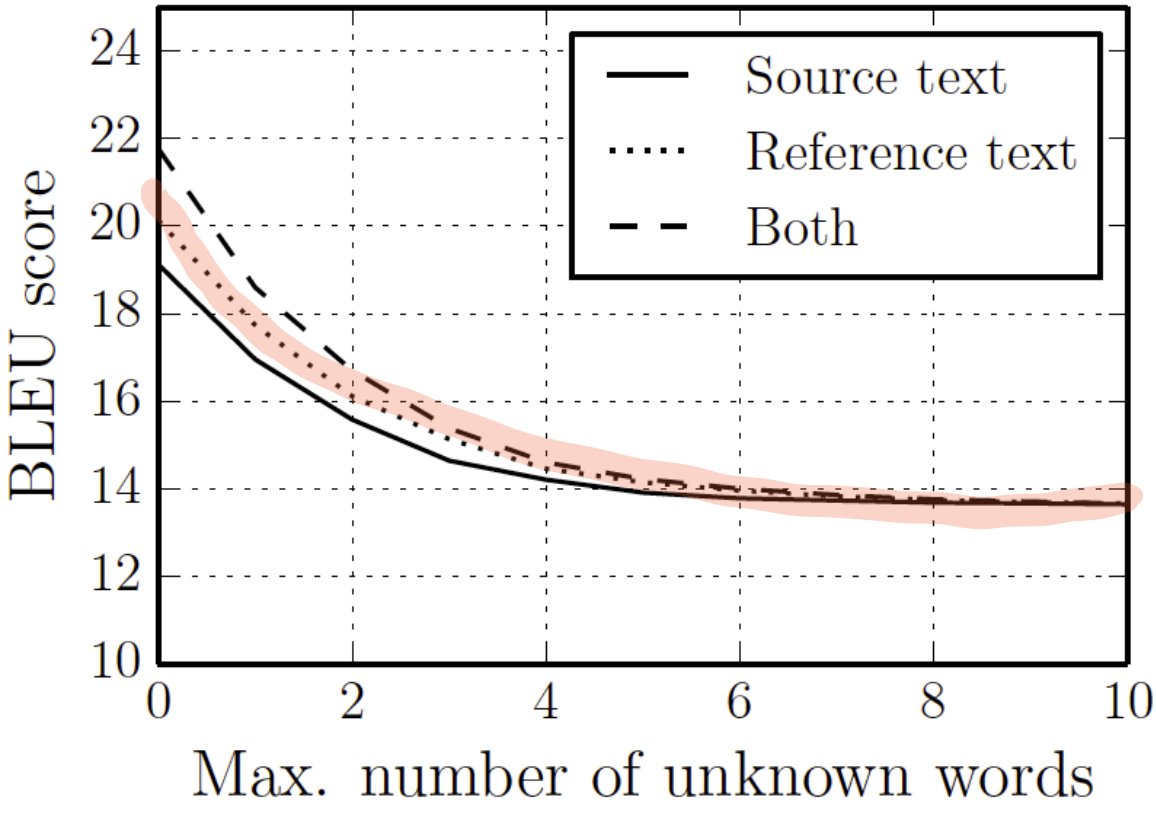
- At test time, the system will encounter even more out-of-vocabulary words

why?

"<unk>"

Out-of-vocabulary words

Cho et al. 2014



Summary Main Weaknesses

- **Sentence length**: encoder has to summarize entire sentences to a fixed-size vector, no matter how long they are

- **Vocabulary**: vocabulary must be fixed at training time, and be rather small

Both lead to low translation quality!

Another vegetable: Daikon

- Educational Encoder-Decoder NMT tool written in Tensorflow
- Serves as starting point for Exercise 5

Main actions in daikon

Train

```
daikon train --source source.txt --target target.txt
```

Translate

```
echo "Here is a sample input text" | daikon translate
```

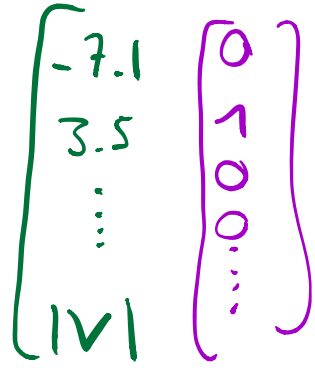
Score

```
daikon score --source source.txt --target target.txt
```

Computation graph

```
def define_computation_graph(source_vocab_size: int, target_vocab_size: int, batch_size: int):

    tf.reset_default_graph()

    # Placeholders for inputs and outputs
    encoder_inputs = tf.placeholder(shape=(batch_size, None), dtype=tf.int32, name='encoder_inputs')

    decoder_targets = tf.placeholder(shape=(batch_size, None), dtype=tf.int32, name='decoder_targets')
    decoder_inputs = tf.placeholder(shape=(batch_size, None), dtype=tf.int32, name='decoder_inputs')

    with tf.variable_scope("Embeddings"):
        source_embedding = tf.get_variable('source_embedding', [source_vocab_size, C.EMBEDDING_SIZE])
        target_embedding = tf.get_variable('target_embedding', [source_vocab_size, C.EMBEDDING_SIZE])

        encoder_inputs_embedded = tf.nn.embedding_lookup(source_embedding, encoder_inputs)
        decoder_inputs_embedded = tf.nn.embedding_lookup(target_embedding, decoder_inputs)
```

Computation graph

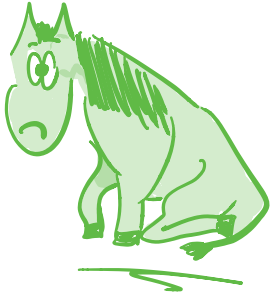



```
with tf.variable_scope("Encoder"):
    encoder_cell = tf.contrib.rnn.LSTMCell(C.HIDDEN_SIZE)
    initial_state = encoder_cell.zero_state(batch_size, tf.float32)

    encoder_outputs, encoder_final_state = tf.nn.dynamic_rnn(encoder_cell,
                                                             encoder_inputs_embedded,
                                                             initial_state=initial_state,
                                                             dtype=tf.float32)

with tf.variable_scope("Decoder"):
    decoder_cell = tf.contrib.rnn.LSTMCell(C.HIDDEN_SIZE)
    decoder_outputs, decoder_final_state = tf.nn.dynamic_rnn(decoder_cell,
                                                             decoder_inputs_embedded,
                                                             initial_state=encoder_final_state,
                                                             dtype=tf.float32)

with tf.variable_scope("Logits"):
    decoder_logits = tf.contrib.layers.linear(decoder_outputs, target_vocab_size)
```

Summary overall

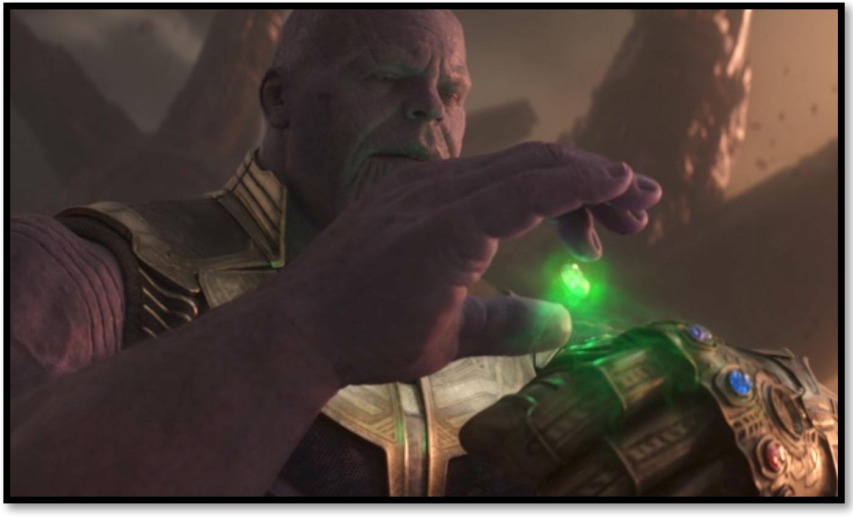
- **Encoder-Decoder Models**: use RNNs for machine translation

- **Main problems**:
  - Long sentences
  - Open vocabulary



- **Daikon**: simple in-house NMT tool

Exercise 5: time management



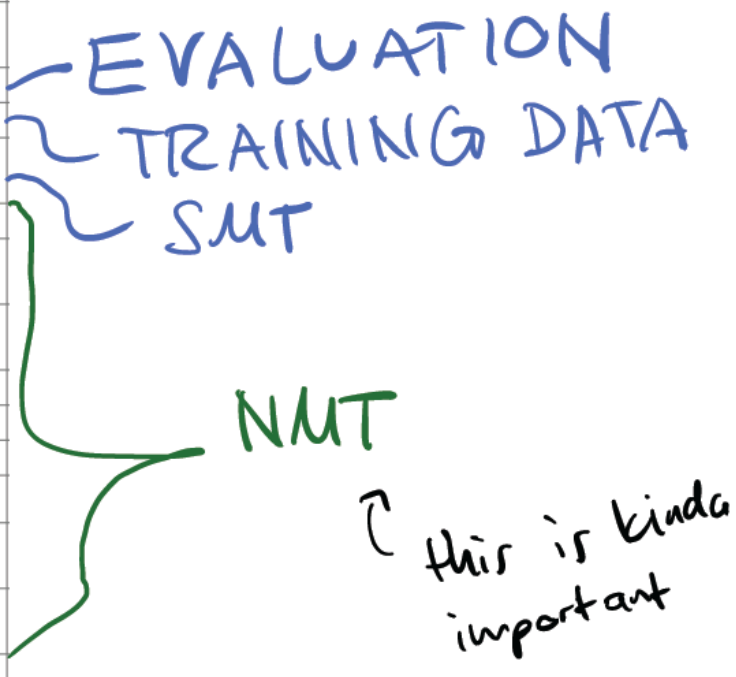
Further reading / recommended links

- Daikon: https://github.com/zurichnlp/daikon
- Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le (2014). Sequence to Sequence Learning with Neural Networks.
- Cho, Kyunghyun, Bart van Merriënboer, Dzmitry Bahdanau and Yoshua Bengio (2014). On the Properties of Neural Machine Translation: Encoder-Decoder Approaches.
- NVIDIA Blog by Kyunghyun Cho: https://devblogs.nvidia.com/introduction-neural-machine-translation-with-gpus/
- Koehn Book NMT Chapter (on OLAT)

- **Do not get hung up on attention until two weeks from now!**

Next time

Termin	Thema
19.02.	Einführung; regelbasierte vs. datengetriebene Modelle
26.02.	Evaluation
05.03.	Trainingsdaten, Vor- und Nachverarbeitung
12.03.	N-Gramm-Sprachmodelle, statistische Maschinelle Übersetzung
19.03.	Grundlagen Lineare Algebra und Analysis, Numpy
26.03.	Lineare Modelle: lineare Regression, logistische Regression
02.04.	Neuronale Netzwerke: MLPs, Backpropagation, Gradient Descent
09.04.	Word Embeddings, Recurrent neural networks
16.04.	Tensorflow und Google Cloud Platform
30.04.	Encoder-Decoder-Modell
07.05.	Decoding-Strategien
14.05.	Attention-Mechanismus, bidirektionales Encoding, Byte Pair Encoding
21.05.	Maschinelle Übersetzung in der Praxis (Anwendungen)
28.05.	Zusammenfassung, Q&A Prüfung
Eventuell: Gastvortrag Prof. Artem Sokolov	
04.06., Raum TBA, 16:15 bis 18:00 Uhr	
Prüfung (schriftlich)	
18.06., AND-2-48, 16.15 bis 18:00 Uhr	



Advance notice: exam questions

- On May 28, we will have an exam Q&A
- Until May 28, please post on OLAT:

  Exam question that would be fair in your opinion

- We will discuss exactly those questions that day.