VL Deep Learning for Natural Language Processing

17. Machine Translation

Prof. Dr. Ralf Krestel

AG Information Profiling and Retrieval

Seq2Seq

- A new task: machine translation

is
the main application area
for

- A new network architecture: sequence-to-sequence

gets
improved
by

- A new deep learning technique: attention

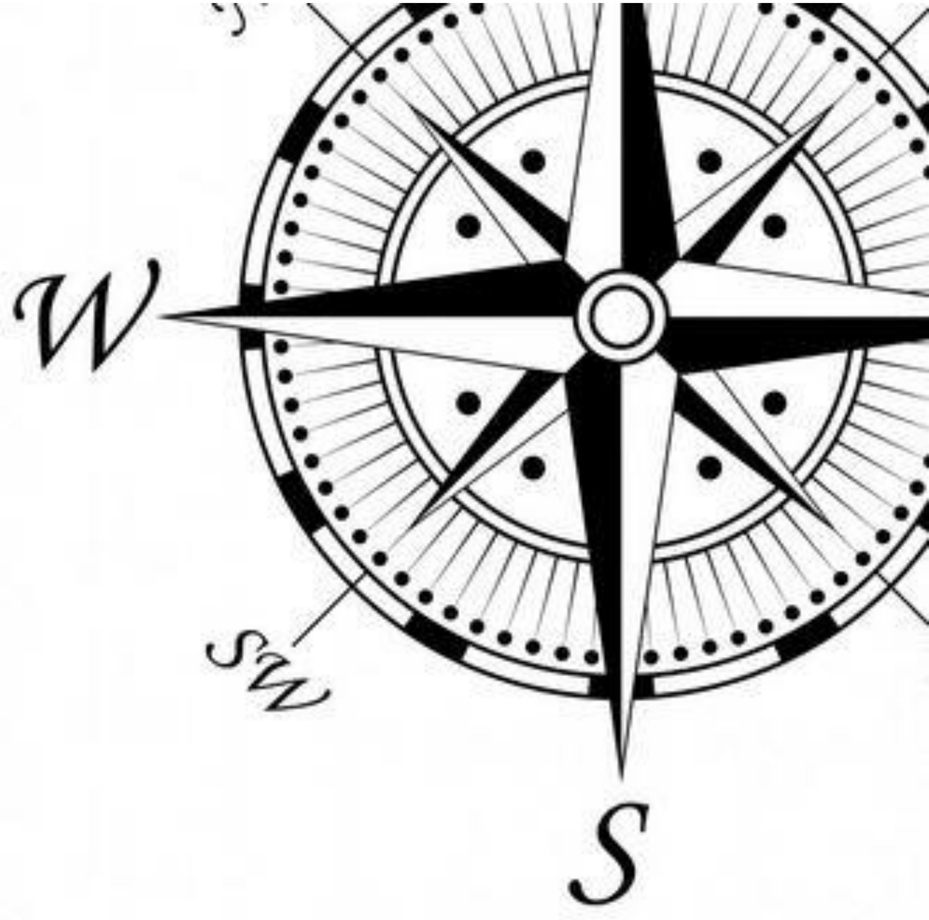
Learning Goals for this Chapter

- Know the (short) history of machine translation
- Know the task and challenges of translation
- Understand seq2seq neural network architectures
- Explain the attention mechanism

- Relevant chapters:
  - S7 (2021): https://www.youtube.com/watch?v=wzfWHP6SXxY

Topics Today

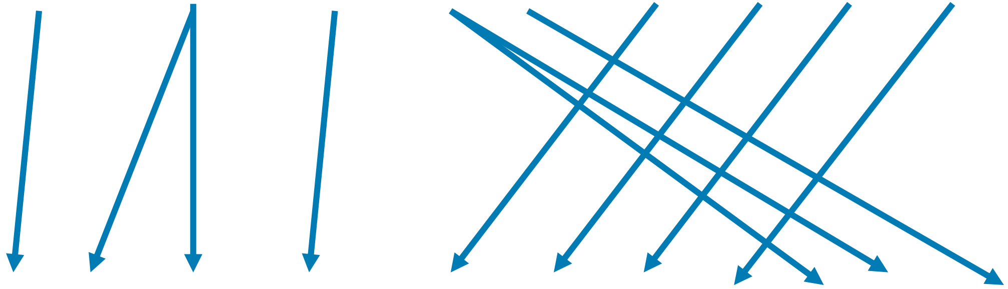
1. **Machine Translation**
2. Sequence-to-Sequence Models
3. Code Example
4. Seq2Seq with Attention



Translation

- The task of machine translation is to translate a sentence x in one language (**source**) into a sentencey in another language (**target**).

x: Ein Männlein steht im Walde ganz still und stumm.



y: A little man stands quite still and silent in the forest.

1950s: First Approaches

- Motivated by the cold war
  - Mainly Russian → English

- **Rule-based** systems
  - Bilingual dictionaries to map Russian word

- Results were quite modest
  - Less money for research

Rene Descartes proposes in 1629 the idea of a universal language, where each symbol of each language is mapped to a universal symbol.

1990–2010: Statistical Machine Translation

- SMT Idea: Learn a probabilistic model from data
- E.g. we want to find the best German sentences y, given the English sentence x

$$\underset{y}{argmax}\, P(y|x)$$

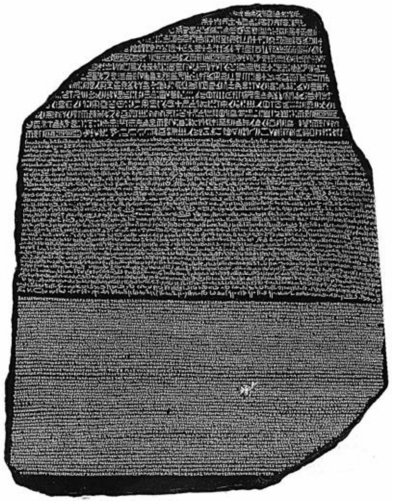
- Using Bayes rule:

$$= \underset{y}{argmax}\, P(x|y)P(y)$$

Translation model:
- **Describes how to translate words and phrases**
- **Learnt using parallel corpora**

Language model:
- **Describes how good German looks like**
- **Learnt using a monolingual corpus**

Statistical Machine Translation (SMT) I



- How to learn the translation model $P(x|y)$?
  - With a large amount of parallel data!

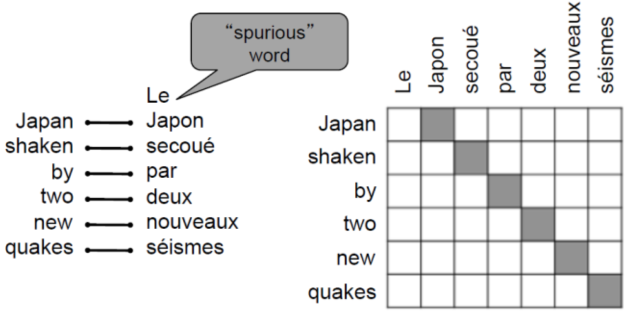
- More specific, we do not want to learn $P(x|y)$, but $P(x, a|y)$, where a is an **alignment**.

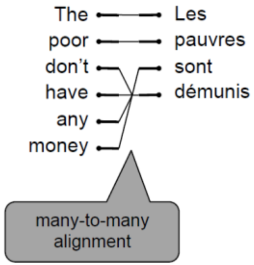
- **Alignment** is the assignment of English words to German words within our sentences x and y.

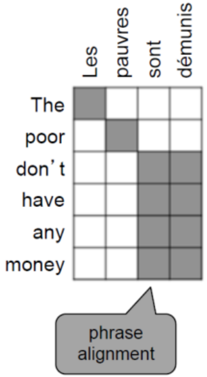
- A couple of factors influence the learning of $P(x, a|y)$:
  - Probabilities of certain alignments
    - Depends on position in sentence as well
  - Probabilities of fertility of certain words
  - …

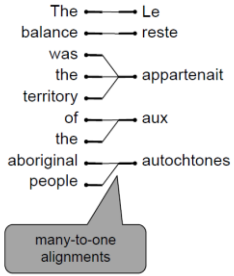
Alignments

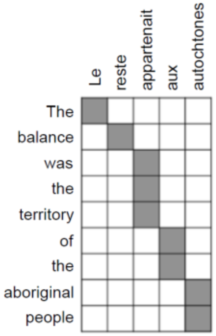
1. Besides 1-to-1 alignments there are other possibilities:
2. 1-to-0 or 0-to-1
   – Some words do not have counterparts in other languages
3. 1-to-many
   – These are fertile words
4. Many-to-1
5. Many-to_many
   – Phrases

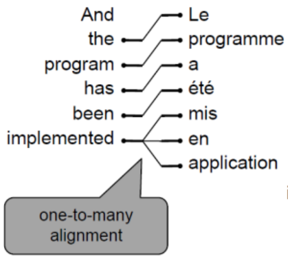


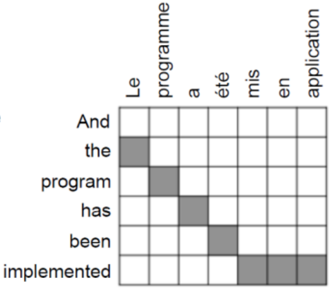










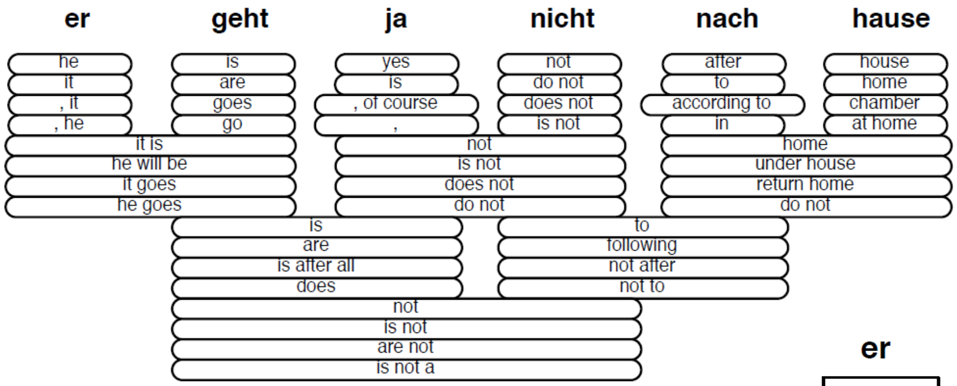


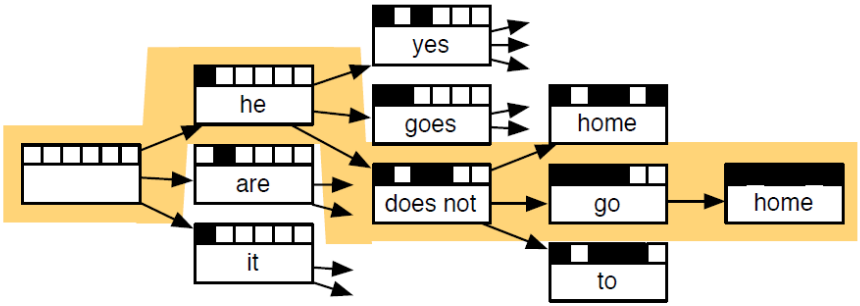
SMT II

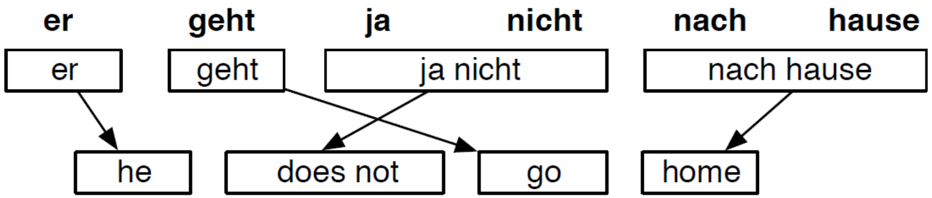
$$\underset{y}{argmax}\, P(x, a|y)P(y)$$

- Methods to compute argmax:
  - Iterate through all possible y to compute the probabilities
    - Way too expensive!
    - Waaaaaaay too expensive!!!!!!!!!!!
  - Heuristic search algorithm that slowly, step-by-step builds up a translation and ignores unlikely translation paths

Heuristic Search







SMT III

- SMT ~~is~~ was a huge research field
  - Own, specialized conferences, challenges, …
- Best SMT-systems are very complex
  - Easy to fill a whole semester!
  - Typically many independent components
  - A lot of feature engineering
    - Depending on involved languages
  - Additional resources needed
    - Equivalent phrases, dictionaries, synonyms, …
    - Need to be created and maintained
  - A lot of manual effort
    - Development and maintainance of whole system
    - For each pair of languages seperately!

2014

Success Story of Neural Machine Translation

- Within 2 years from first experiments to leading state-of-the-art
  - 2014: First paper to Seq2Seq published
  - 2016: Google Translate switches from SMT to NMT

Neural Machine Translation

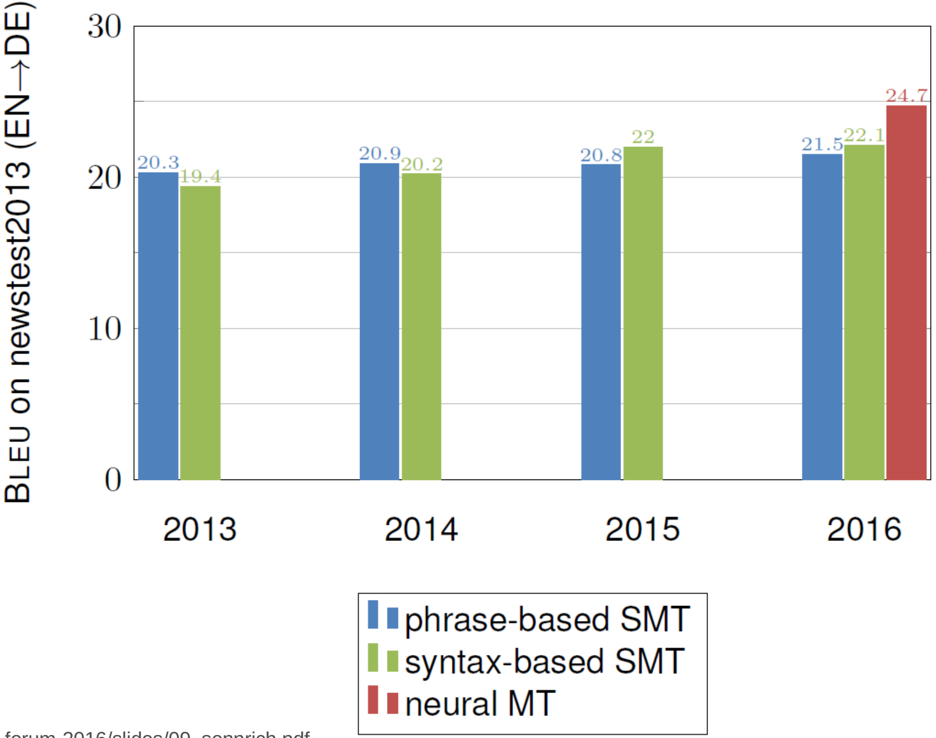
- SMT-Systems have been developed by hundreds of people for multiple decades.

- A handful of researchers only needed a couple of months to evaporate previous results with NMT!

Evaluation

- BLEU (BiLingual Evaluation Understudy)
  - Computes a similarity score between a machine-generated sentence and one or multiple manual generated translations
    - Ngram-Accuracyt (3-4-grams)
    - Penalty for translations that are too short
  - Useful metric but **not perfect**
    - Many tranlations are correct; there is not the one correct translation.
    - A really good translation could receive a low BLEU-score if the n-gram overlap with a human reference translation is low.

Development over Time: PBSMT, SBSMT, NMT



http://www.meta-net.eu/events/meta-forum-2016/slides/09_sennrich.pdf

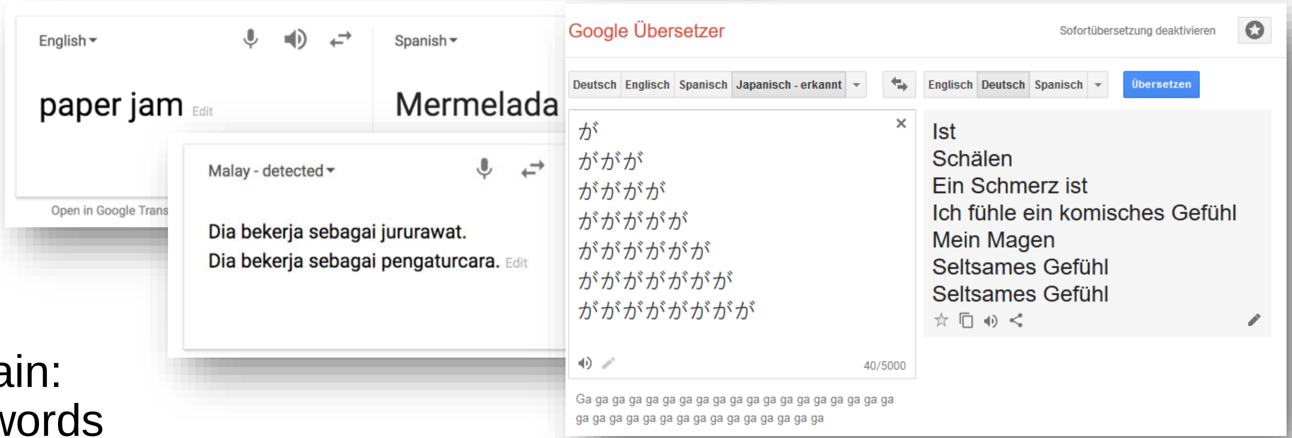
Comparison: NMT vs. SMT

- **Advantage**
  - Better results
    - **Text more fluent**
    - **More context-aware**
    - **Better word and phrase similarity**
      - Distributed representations share strength
  - Only one single neural network
    - **End-to-end optimization**
    - No subcomponents, which need to be optimized individually
  - A lot less manual effort
    - No feature-engineering
    - One method for all language pairs

- **Disadvantage**
  - Less easy to interpret
    - Hard to debug
  - Hard to control
    - No way to encode rules
    - Security issues
  - No explicit use of syntactic, semantic, discourse, anaphora, … structure

Is Machine Translation Solved?

- **No!**



- Many difficulties remain:
  - Out-of-vocabulary words
  - Domain mismatch between training and test data
  - Keep context accross long documents
  - Language pairs with few training examples
  - Common Sense
  - Bias in training data
  - Surprising outputs

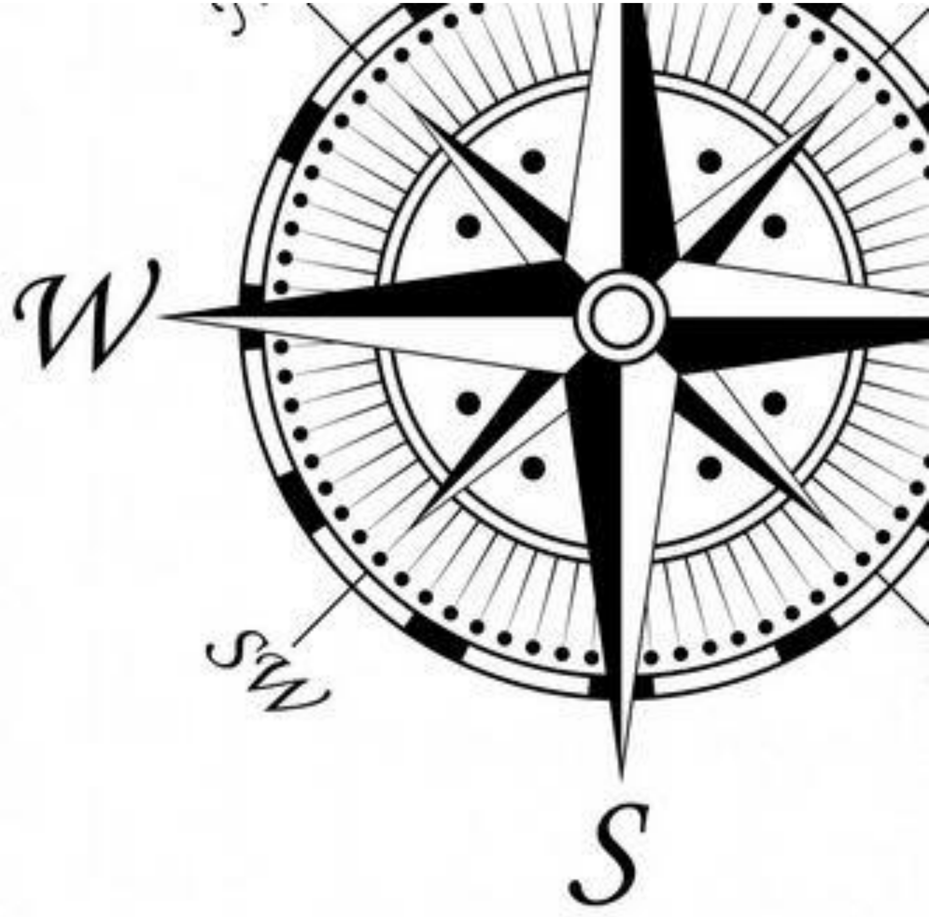
Machine Translation

- How could a neural network architecture look like for machine translation?

Start 5 4 3 2 1 End

Topics Today

1. Machine Translation
2. **Sequence-to-Sequence Models**
3. Code Example
4. Seq2Seq with Attention



1990–2010: Statistical Machine Translation

- SMT Idea: Learn a probabilistic model from data

$$\underset{y}{argmax}\, P(y|x) = \underset{y}{argmax}\, P(x|y)P(y)$$

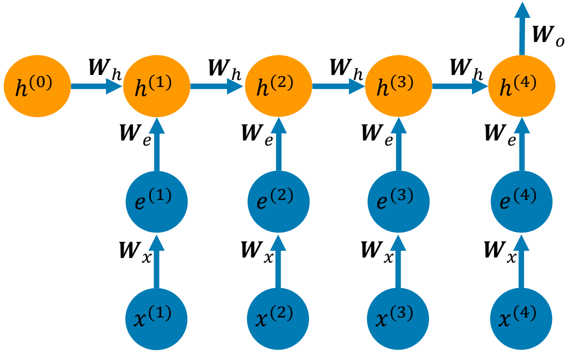
-

Basic Idea

- SMT idea: learn a probabilistic model from data

$$\underset{y}{argmax}\, P(y|x) = \underset{y}{argmax}\, P(x|y)P(y)$$

- NMT idea: translate using a neural network.



- Network type is called **sequence-to-sequence** (Seq2Seq) and consists of **two RNNs**.

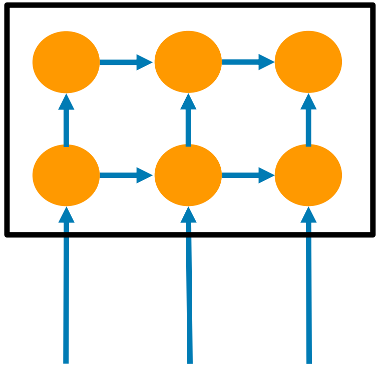
- Models $P(y|x)$ directly
  - Conditional Language Model
    - Models a word in the target sentences given the previous word of the target sentence and the source sentence
    - $P(y|x) = \mathrm{P}(\mathrm{y}_1|\mathrm{x})\mathrm{P}(\mathrm{y}_2|\mathrm{y}_1,\mathrm{x})\mathrm{P}(\mathrm{y}_3|\mathrm{y}_1,\mathrm{y}_2,\mathrm{x}) \ldots \mathrm{P}(\mathrm{y}_\mathrm{T}|\mathrm{y}_1,\ldots,\mathrm{y}_{\mathrm{T}-1},\mathrm{x})$

Train using big parallel corpus

  - No „detour“ via $P(source|target)P(target)$

Encoder

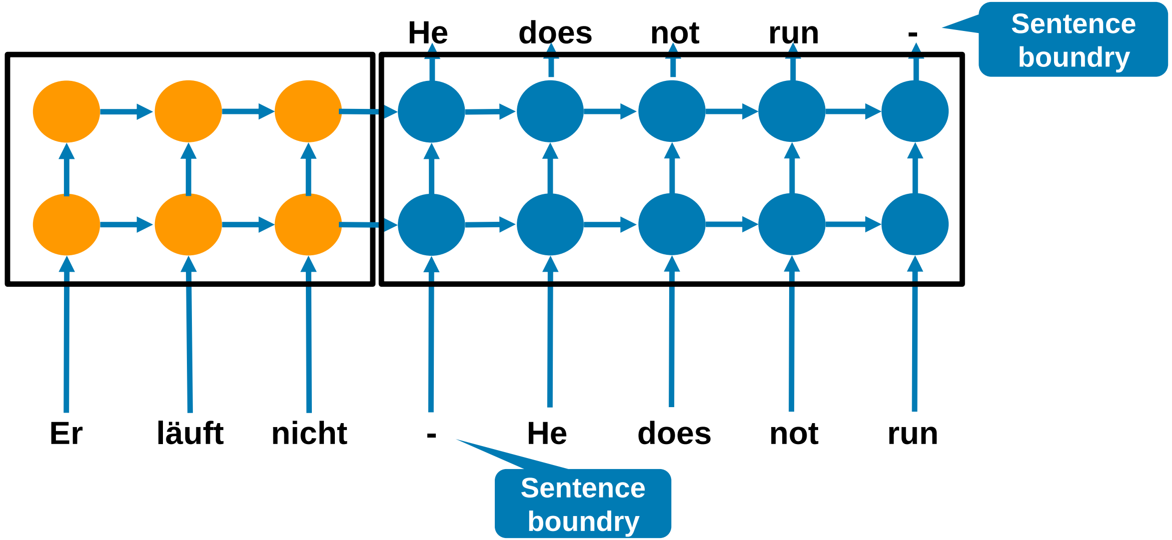
- Encoder: Represents the source target



Er läuft nicht - He does not run

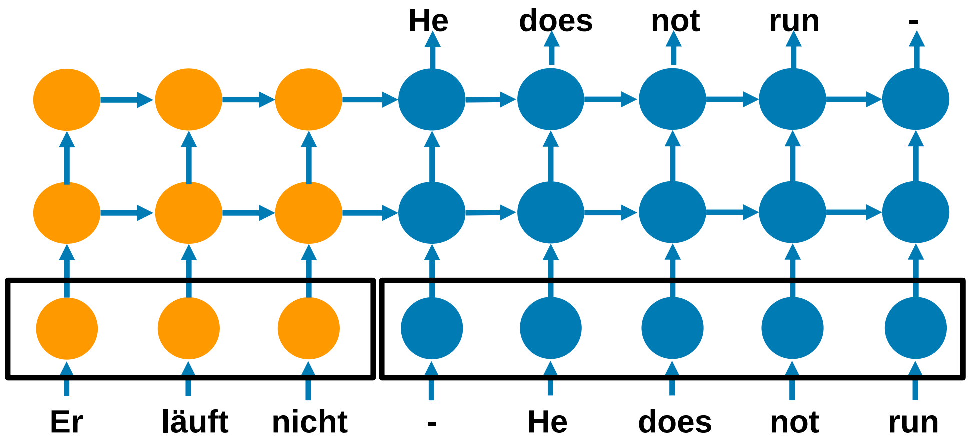
Decoder

- Decoder: Generates the target sentence



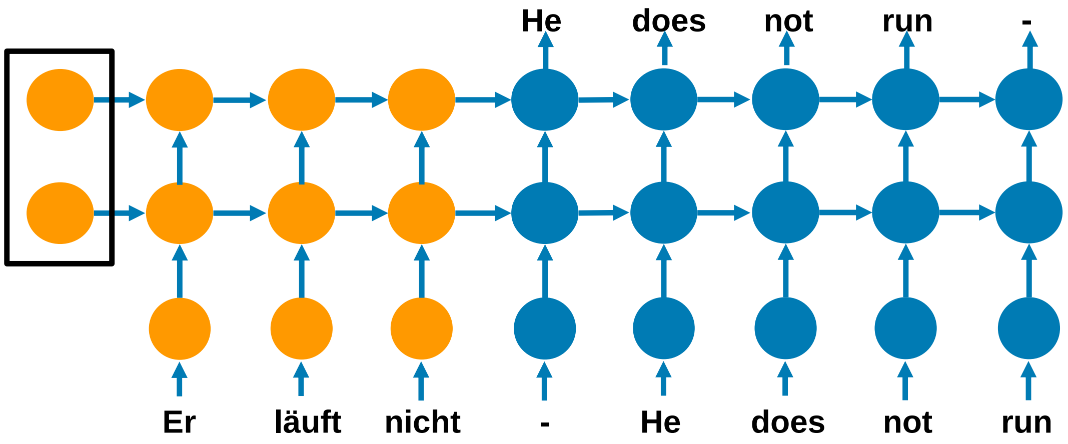
Embeddings

- Source and target embeddings
  - Need to be trained once per language



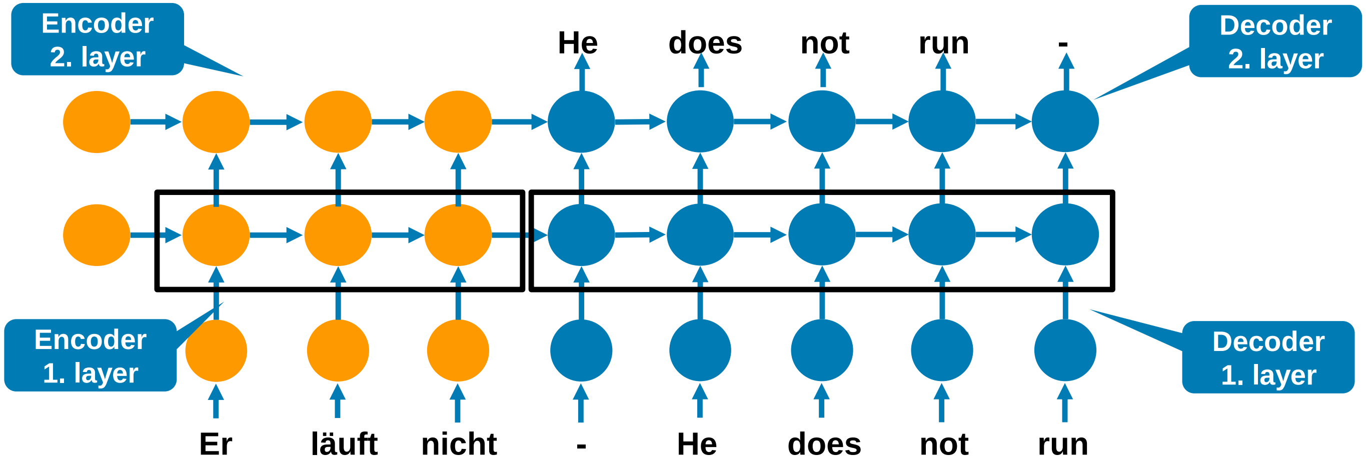
Initial States

- Initial states
  - Often set to 0



Training

- Two layers × two parts

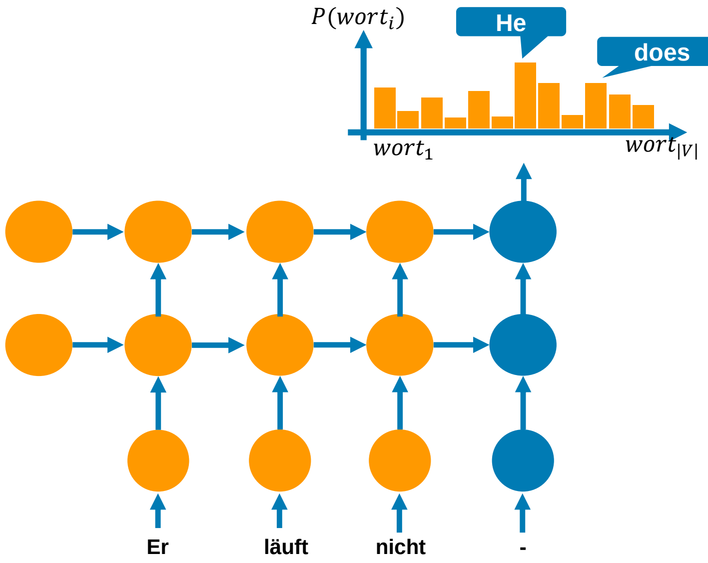


Multi-layer RNNs in Practice

- Multi-layer or stacked RNNs allow the network to compute more complex representations
- The lower RNNs should compute lower-level features and the higher RNNs should compute higher-level features.
- High-performing RNNs are usually multi-layer (but aren't as deep as convolutional or feed-forward networks)
- For example: In a 2017 paper, Britz et al. find that for Neural Machine Translation, 2 to 4 layers is best for the encoder RNN, and 4 layers is best for the decoder RNN
- Often 2 layers is a lot better than 1, and 3 might be a little better than 2
- Usually, skip-connections/dense-connections are needed to train deeper RNNs (e.g., 8 layers)
- Transformer-based networks (e.g., BERT) are usually deeper, like 12 or 24 layers.
  - they have a lot of skipping-like connections

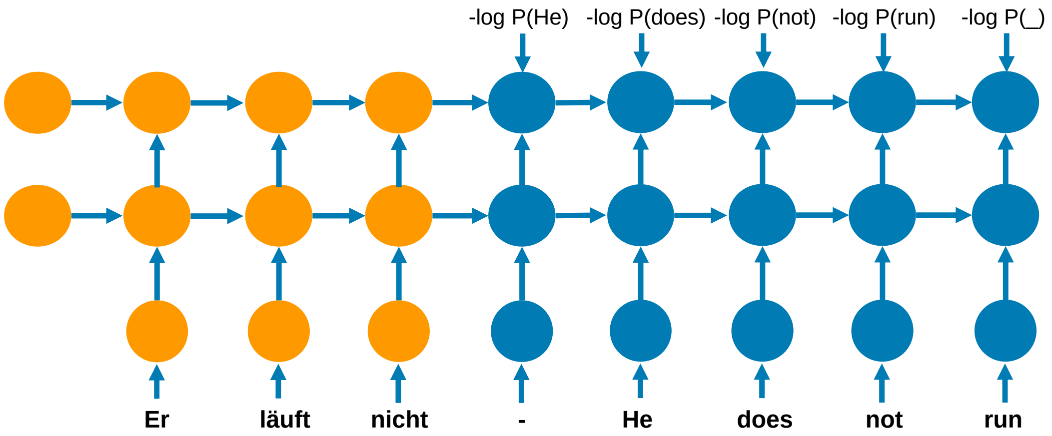
From Vectors to Words

- Hidden State → Scores → Probabilitis



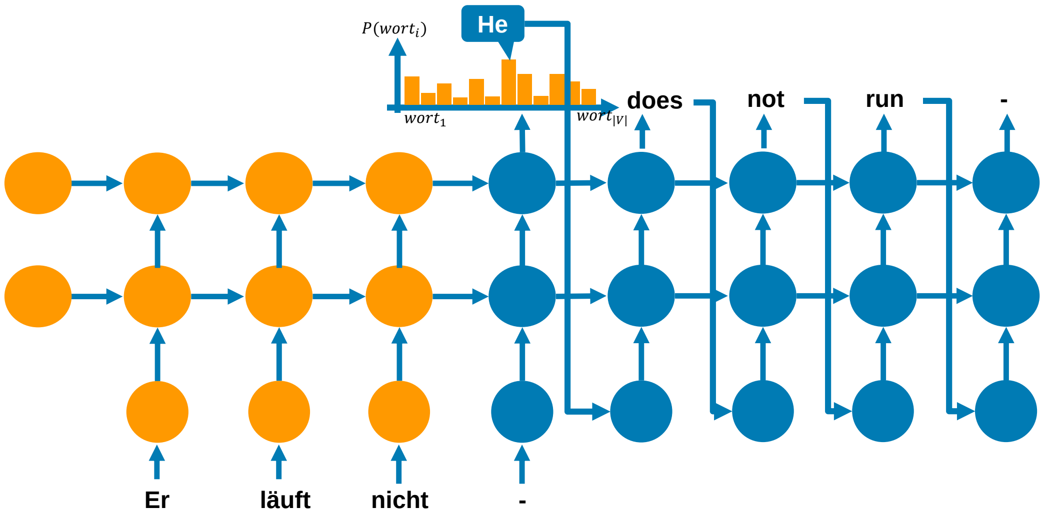
Loss Function

- Find the maximum of $P(target|source)$
- Sum-up the individual losses
- Backpropagation through time



Testing

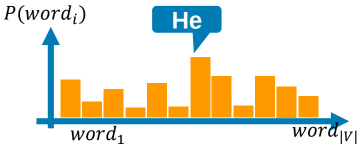
- During training we know the correct translation from a parallel corpus.
- During testing we only have the source sentences.
  - Take the word with the highest probability (greedy)



Can we do Better than Greedy?

- Greedy: decoder takes at every step the word with the highest probability (argmax) to generate the target sentence.

- Problem?



- Greedy decoding cannot take back decisions once made.
  - Les pauvres sont démunis (the poor don‘t have any money)
    - The…
    - The poor…
    - The poor are…

- Better method: beam search

Beam Search

- Idealy we want to maximize

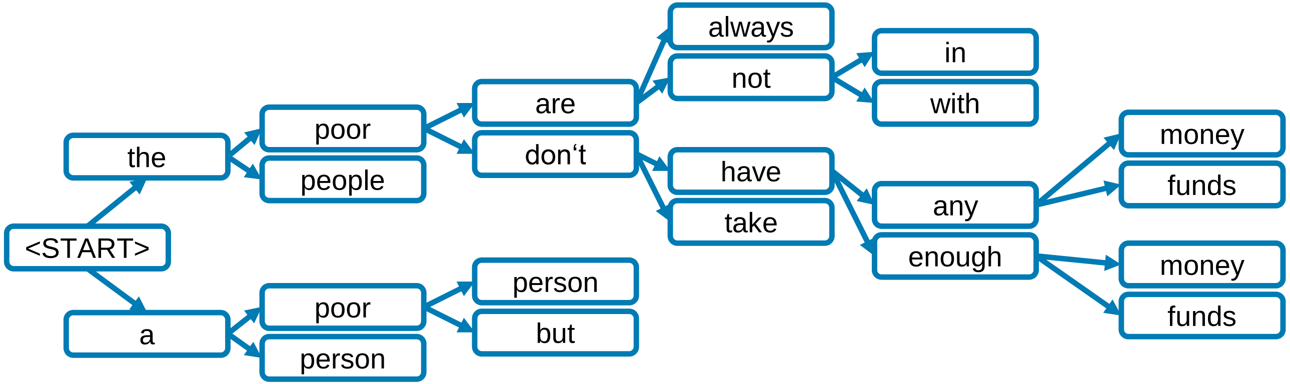
$$P(y|x) = \mathrm{P}(\mathrm{y}_1|\mathrm{x})\mathrm{P}(\mathrm{y}_2|\mathrm{y}_1, \mathrm{x})\mathrm{P}(\mathrm{y}_3|\mathrm{y}_1, \mathrm{y}_2, \mathrm{x}) \ldots \mathrm{P}(\mathrm{y}_\mathrm{T}|\mathrm{y}_1, \ldots, \mathrm{y}_{\mathrm{T}-1}, \mathrm{x})$$

- We could iterate through all words
  - Complexity $O(V^T)$ with V being the vocabulary size and T the length of the target sentence
  - Way too expensive!!!!!

- Beam search: At each step store the *k* most probable partial translations
  - *k* is the beam size (~5-10)
  - Finds not necessarily the optimal solution
  - But very efficient

Beam Search: Example

- Beam size = 2



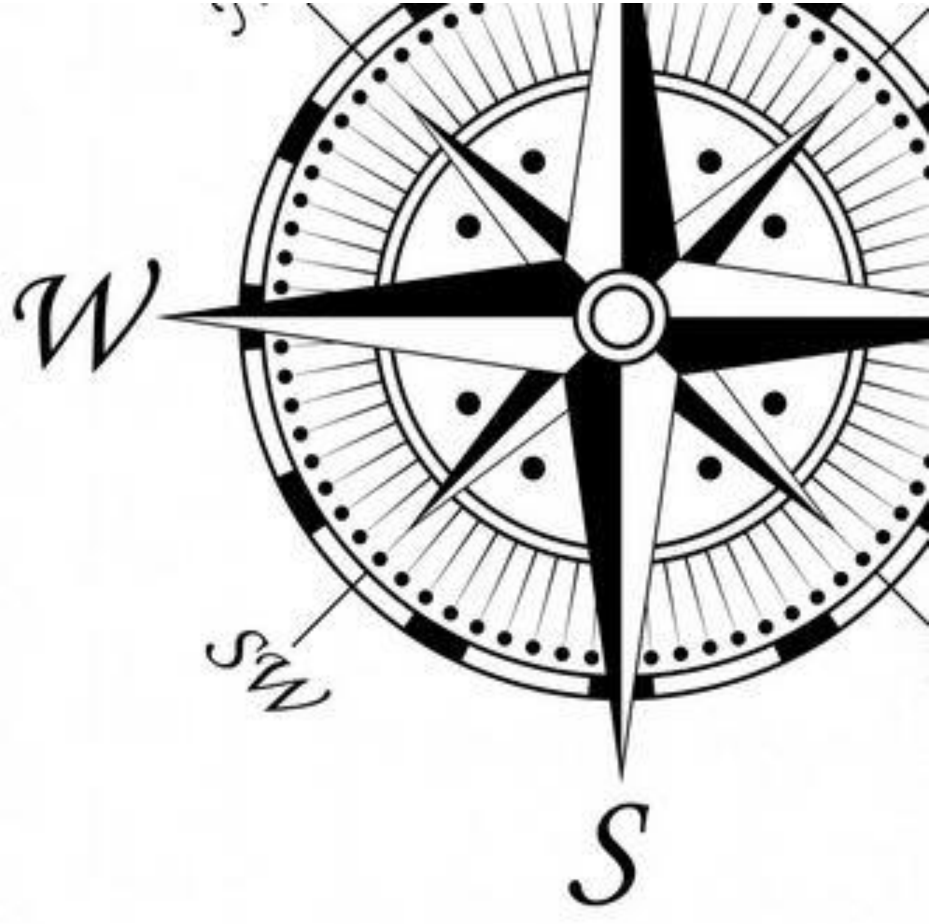
Seq2Seq Applications

- Not only machine translation
- Many text mining problems can be phrased as sequence-to-sequence problem
  - Summarization
    - Long text → short text
  - Dialog systems
    - Previous utterance → next utterance
  - Parsing
    - Input text → parse tree as sequence
  - Code generation
    - Natural language → Python code
  - Text simplification
    - Normal (complicated) text → easy-to-read text

	Encoder	Decoder
Sutskever et al. 2014	Deep LSTM	Deep LSTM
Cho et al., 2014 Bahdanau et al., 2015 Jean et al., 2015	(Bidirectional) GRU	GRU
Kalchbrenner & Blunsom, 2013	CNN	(Inverse CNN) RNN

Topics Today

1. Machine Translation
2. Sequence-to-Sequence Models
3. **Code Example**
4. Seq2Seq with Attention



Preprocessing I

```
from keras.models import Model
from keras.layers import Input, LSTM, Dense
import numpy as np
lines = open('fra.txt', encoding='utf-8').read().split('\n')
eng_sent = []
fra_sent = []
eng_chars = set()
fra_chars = set()
nb_samples = 10000
for line in range(nb_samples):
        eng_line = str(lines[line]).split('\t')[0]
        fra_line = '\t' + str(lines[line]).split('\t')[1] + '\n,
        eng_sent.append(eng_line)
        fra_sent.append(fra_line)
for ch in eng_line:
        if (ch not in eng_chars):
            eng_chars.add(ch)
for ch in fra_line:
        if (ch not in fra_chars):
            fra_chars.add(ch)
```

https://github.com/kmsravindra/ML-AI-experiments/raw/master/AI/Neural%20Machine%20Translation/fra.txt

\t to mark sentence start and end

Preprocessing II

```
fra_chars = sorted(list(fra_chars))
eng_chars = sorted(list(eng_chars))
eng_index_to_char_dict = {}
eng_char_to_index_dict = {}
for k, v in enumerate(eng_chars):
    eng_index_to_char_dict[k] = v
    eng_char_to_index_dict[v] = k
fra_index_to_char_dict = {}
fra_char_to_index_dict = {}
for k, v in enumerate(fra_chars):
    fra_index_to_char_dict[k] = v
    fra_char_to_index_dict[v] = k
max_len_eng_sent = max([len(line) for line in eng_sentences])
max_len_fra_sent = max([len(line) for line in fra_sentences])
>16
>59
```

Preprocessing III

```
tokenized_eng_sentences = np.zeros(shape =
(nb_samples,max_len_eng_sent,len(eng_chars)), dtype='float32')
tokenized_fra_sentences = np.zeros(shape =
(nb_samples,max_len_fra_sent,len(fra_chars)), dtype='float32')
target_data = np.zeros((nb_samples, max_len_fra_sent,
len(fra_chars)),dtype='float32')

for i in range(nb_samples):
    for k,ch in enumerate(eng_sent[i]):
        tokenized_eng_sentences[i,k,eng_char_to_index_dict[ch]] = 1
    for k,ch in enumerate(fra_sent[i]):
        tokenized_fra_sentences[i,k,fra_char_to_index_dict[ch]] = 1
        if k > 0:
            target_data[i,k-1,fra_char_to_index_dict[ch]] = 1
```

Seq2Seq Model Architecture

```
#Encoder
encoder_input = Input(shape=(None,len(eng_chars)))
encoder_LSTM = LSTM(256,return_state = True)
encoder_outputs, encoder_h, encoder_c = encoder_LSTM (encoder_input)
encoder_states = [encoder_h, encoder_c]
# Decoder
decoder_input = Input(shape=(None,len(fra_chars)))
decoder_LSTM = LSTM(256,return_sequences=True, return_state = True)
decoder_out, _ , _ = decoder_LSTM(decoder_input, initial_state=encoder_states)
decoder_dense = Dense(len(fra_chars),activation='softmax')
decoder_out = decoder_dense (decoder_out)
model = Model(inputs=[encoder_input, decoder_input],outputs=[decoder_out])
# Trainieren
model.compile(optimizer='rmsprop', loss='categorical_crossentropy')
model.fit(x=[tokenized_eng_sentences,tokenized_fra_sentences],
          y=target_data,
          batch_size=64,
          epochs=50,
          validation_split=0.2)
```

Model for Testing

```
# Inference models for testing
encoder_model_inf = Model(encoder_input, encoder_states)
decoder_state_input_h = Input(shape=(256,))
decoder_state_input_c = Input(shape=(256,))
decoder_input_states = [decoder_state_input_h, decoder_state_input_c]
decoder_out, decoder_h, decoder_c = decoder_LSTM(decoder_input,
                                          initial_state = decoder_input_states)
decoder_states = [decoder_h , decoder_c]
decoder_out = decoder_dense(decoder_out)
decoder_model_inf = Model(inputs=[decoder_input] + decoder_input_states,
                          outputs=[decoder_out] + decoder_states )
```

Sentence Translation

```
def decode_seq(inp_seq):
    states_val = encoder_model_inf.predict(inp_seq)
    target_seq = np.zeros((1, 1, len(fra_chars)))
    target_seq[0, 0, fra_char_to_index_dict['\t']] = 1
    translated_sent = ''
    stop_condition = False
    while not stop_condition:
        decoder_out, decoder_h, decoder_c = decoder_model_inf.predict(
                                                 x=[target_seq] + states_val)
        max_val_index = np.argmax(decoder_out[0,-1,:])
        sampled_fra_char = fra_index_to_char_dict[max_val_index]
        translated_sent += sampled_fra_char
        if ( (sampled_fra_char == '\n') or (len(translated_sent) >
max_len_fra_sent)) :
            stop_condition = True
        target_seq = np.zeros((1, 1, len(fra_chars)))
        target_seq[0, 0, max_val_index] = 1
        states_val = [decoder_h, decoder_c]
    return translated_sent
```

Example Translation

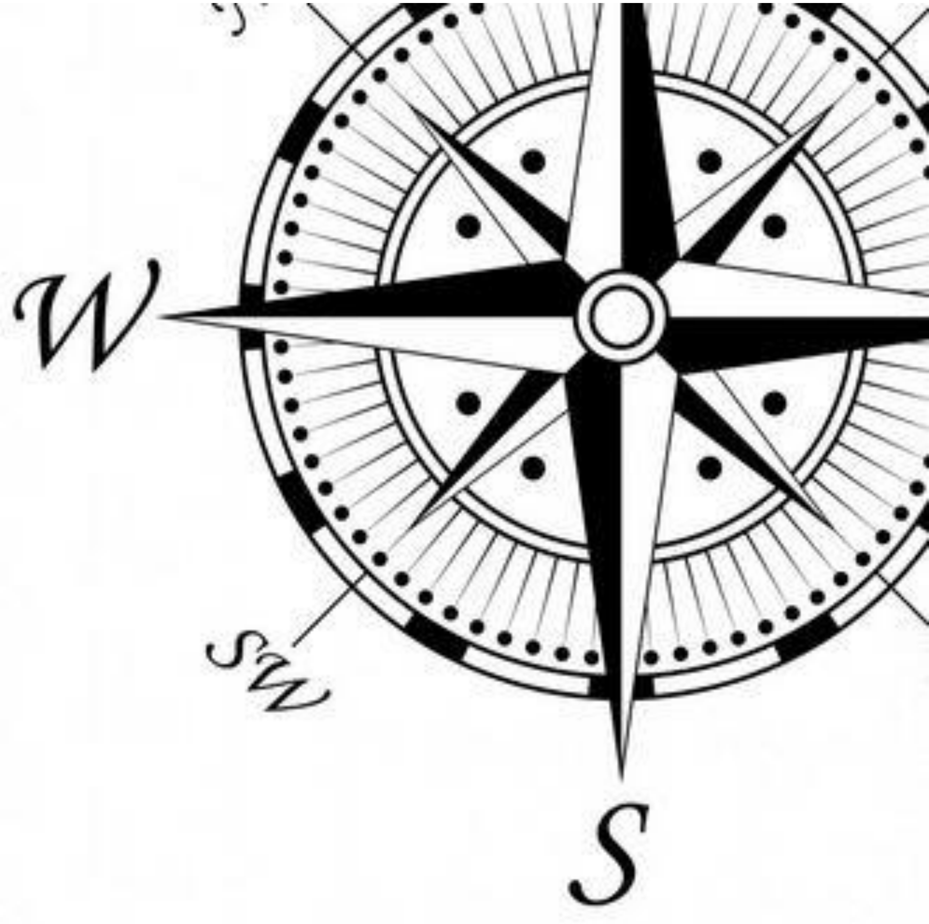
```
for seq_index in range(10):
    inp_seq = tokenized_eng_sentences[seq_index:seq_index+1]
    translated_sent = decode_seq(inp_seq)
    print('-')
    print('Input sentence:', eng_sent[seq_index])
    print('Decoded sentence:', translated_sent)
```

Hyperparameters

- Choice greatly affects the quality of your model.
- In short, take parameters from papers/tutorials, and perform grid search around them (try many combinations of parameters).
  - **RNN variants**: LSTMs have a different (much better) recurrent equation.
  - **Hidden state sizes**: larger: more memory! Requires more data.
  - **Embedding sizes**: more representation power! Requires more data.
  - **Learning rate**: the step size you take in learning your parameters! Start this "large", and cut it in half when your training stops improving development set Performance.
  - **Regularization**: "dropout" prevents overfitting by making each node in your hidden state unavailable for an observation with a given probability. Try some values around .2 to .3.
  - **Batch size**: the number of observations to group together before performing a parameter update step. Larger batches: less fine-grained training, many more observations per minute, especially on GPU.

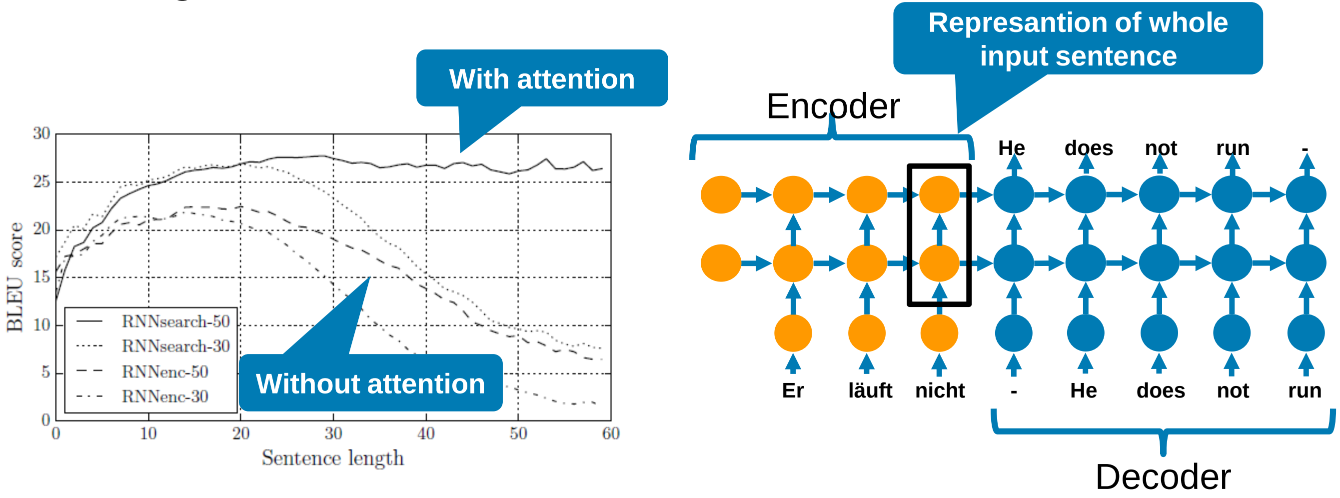
Topics Today

1. Machine Translation
2. Sequence-to-Sequence Models
3. Code Example
4. **Seq2Seq with Attention**



Information Bottlenecks

- The complete source sentence is stored in two vectors.
  - Bottleneck problem
- In particular longer source sentences are then hard to translate

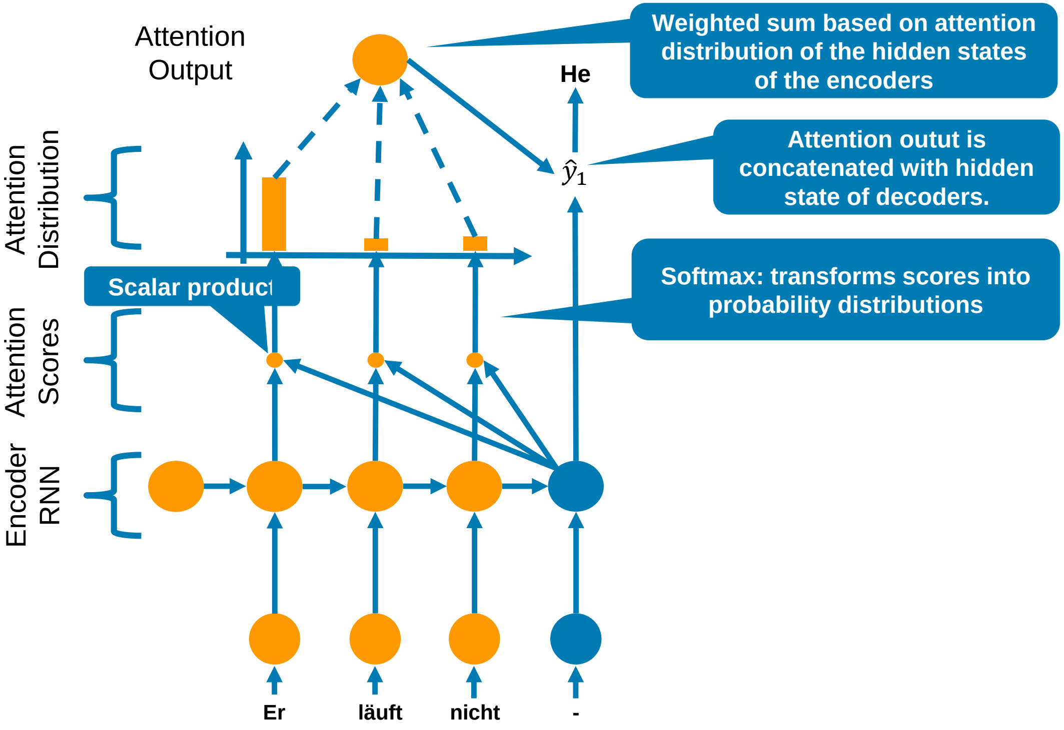


Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." *arXiv preprint arXiv:1409.0473* (2014).

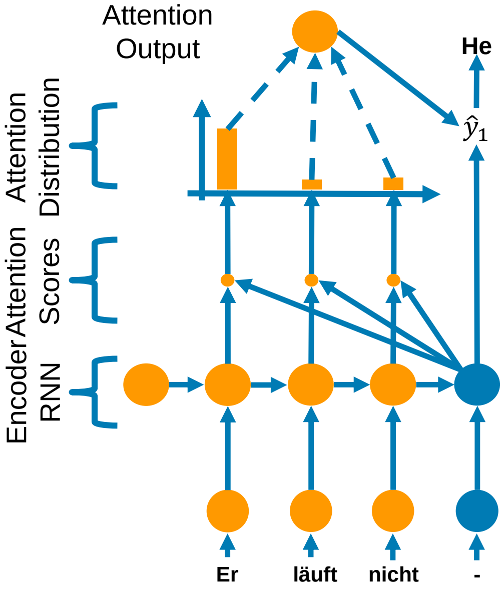
Attention Mechanism I

Basic idea:
- At each step of the decoding process, the focus of the network is only on a particular part of the input sequence.

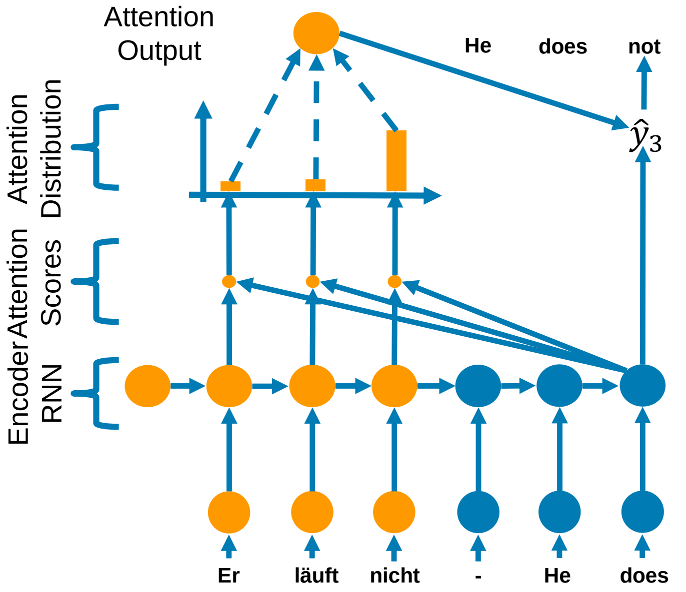
- Learn translation and alignment at the same time!



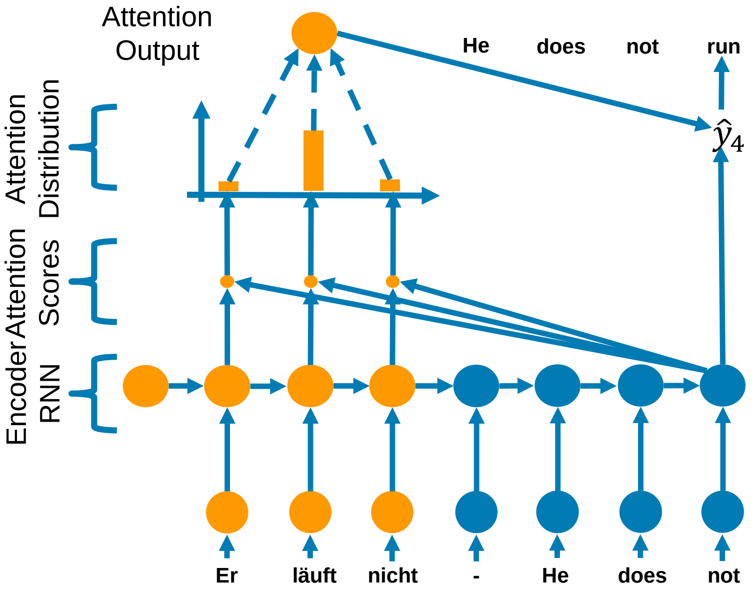
Attention Mechanism II



Attention Mechanism III



Attention Mechanism IV



Seq2Seq Equations

- Hidden States of the encoders: $\boldsymbol{h}^{(1)}, \dots, \boldsymbol{h}^{(N)} \in R^h$
- At time t the hidden state of the decoders is $\boldsymbol{s}^{(t)} \in R^h$
- Attention score at time t:

$$\mathbf{e}^{(\mathrm{t})} = \left[\left(\boldsymbol{s}^{(t)}\right)^{\mathrm{T}} \mathbf{h}^{(\mathrm{t})}, \dots, \left(\boldsymbol{s}^{(t)}\right)^{\mathrm{T}} \mathbf{h}^{(\mathrm{N})}\right] \in R^N$$

- To get the attention distribution $\alpha^{(t)}$ for step t we take the softamx:

$$\alpha^{(t)} = sofmax\left(\boldsymbol{e}^{(t)}\right) \in R^N$$

- From this we compute the attention output $\boldsymbol{a}^{(t)}$ as weighted sum:

$$\boldsymbol{a}^{(t)} = \sum_{i=1}^{N} \alpha_i^{(t)} \boldsymbol{h}^{(i)} \in R^h$$

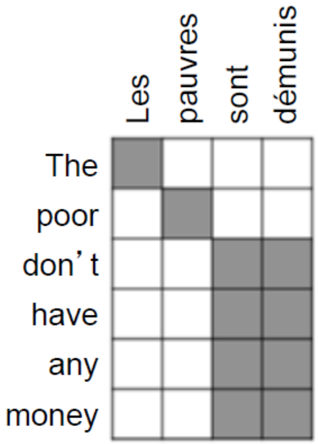
- At last we concatenate the attention output $\boldsymbol{a}^{(t)}$ with the hidden state of the decoders:

$$\left[\boldsymbol{a}^{(t)} \boldsymbol{s}^{(t)}\right] \in R^{2h}$$

- The remaining part is identical to sequence-to-sequence models without attention

Why is Attention so Successful?

- Attention results in **significant performance improvement for NMT**
  - It is very useful for the decoder to focus on certain parts of the input.
- Attention **solves the bottleneck problem**
  - The bottleneck is avoided since the decoder can directly inspect the hidden states of the encoders.
- Attention also **helps with the vanishing gradient problem**
  - There exists a shortcut to states that are far away.
- Attention **enables the interpretation/explanation of results (a little)**
  - We can see the focus of the decoders by looking at the attention distribution.
  - We get the **alignment for free**
  - Although we never trained alignment explicitly
  - The network has learnt alignment on its own



Example Translation

Source Sentence	Orlando Bloom and ***Miranda Kerr*** still love each other

Example Translation

Source Sentence	Orlando Bloom and ***Miranda Kerr*** still love each other
Manual Translation	Orlando Bloom und **Miranda Kerr** lieben sich noch immer
Seq2seq with Attention	Orlando Bloom und **Miranda Kerr** lieben einander noch immer
Seq2seq without Attention	Orlando Bloom und **Lucas Miranda** lieben einander noch immer

Source Sentence	We ' re pleased the FAA recognizes that an enjoyable passenger experience is ***not incompatible*** with safety and security , said Roger Dow , CEO of the U.S. Travel Association
Manuel Translation	Wir freuen uns , dass die FAA erkennt , dass ein angenehmes Passagiererlebnis nicht **im Widerspruch zur Sicherheit steht** , sagte Roger Dow , CEO der U.S. Travel Association
Seq2seq with Attention	Wir freuen uns , dass die FAA anerkennt , dass ein angenehmes ist nicht mit Sicherheit und Sicherheit **unvereinbar** ist , sagte Roger Dow , CEO der US - die
Seq2seq without Attention	Wir freuen uns über die <unk> , dass ein <unk> <unk> mit Sicherheit nicht **vereinbar** ist mit Sicherheit und Sicherheit , sagte Roger Cameron , CEO der US - <unk>

Learning Goals for this Chapter

- Know the (short) history of machine translation
- Know the task and challenges of translation
- Understand seq2seq neural network architectures
- Explain the attention mechanism

- Relevant chapters:
  – S7 (2021): https://www.youtube.com/watch?v=wzfWHP6SXxY

Literature

- *Sequence-to-sequence Models* by John Hewitt & Reno Kriz
  - https://nlp.stanford.edu/~johnhew/public/14-seq2seq.pdf
- *Machine Translation and Sequence-to-sequence Models: A Tutorial* by Graham Neubig
  - https://arxiv.org/pdf/1703.01619.pdf
- *On the Properties of Neural Machine Translation: Encoder–Decoder Approaches* by Cho et al. In Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8). 2014.
  - https://arxiv.org/pdf/1409.1259.pdf
- Luong, M. T., Pham, H., & Manning, C. D. (2015, September). Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 1412-1421).
- Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, *27*.