Machine Translation

04: Neural Machine Translation

Rico Sennrich

University of Edinburgh

Overview

last lecture

- how do we represent language in neural networks?
- how do we treat language probabilistically (with neural networks)?

today's lecture

- how do we model translation with a neural network?
- how do we generate text from a probabilistic translation model?

Modelling Translation

- Suppose that we have:
  - a source sentence S of length m $(x_1, \dots, x_m)$
  - a target sentence T of length n $(y_1, \dots, y_n)$
- We can express translation as a probabilistic model

$$T^* = \arg\max_T p(T|S)$$

- Expanding using the chain rule gives

$$\begin{aligned} p(T|S) &= p(y_1, \dots, y_n | x_1, \dots, x_m) \\ &= \prod_{i=1}^{n} p(y_i | y_1, \dots, y_{i-1}, x_1, \dots, x_m) \end{aligned}$$

Differences Between Translation and Language Model

- Target-side language model:

$$p(T) = \prod_{i=1}^{n} p(y_i | y_1, \dots, y_{i-1})$$

- Translation model:

$$p(T|S) = \prod_{i=1}^{n} p(y_i | y_1, \dots, y_{i-1}, x_1, \dots, x_m)$$

- We could just treat sentence pair as one long sequence, but:
  - We do not care about $p(S)$
  - We may want different vocabulary, network architecture for source text

Differences Between Translation and Language Model

- Target-side language model:

$$p(T) = \prod_{i=1}^{n} p(y_i|y_1, \ldots, y_{i-1})$$

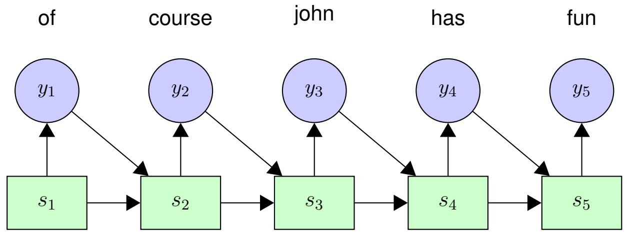
- Translation model:

$$p(T|S) = \prod_{i=1}^{n} p(y_i|y_1, \ldots, y_{i-1}, x_1, \ldots, x_m)$$

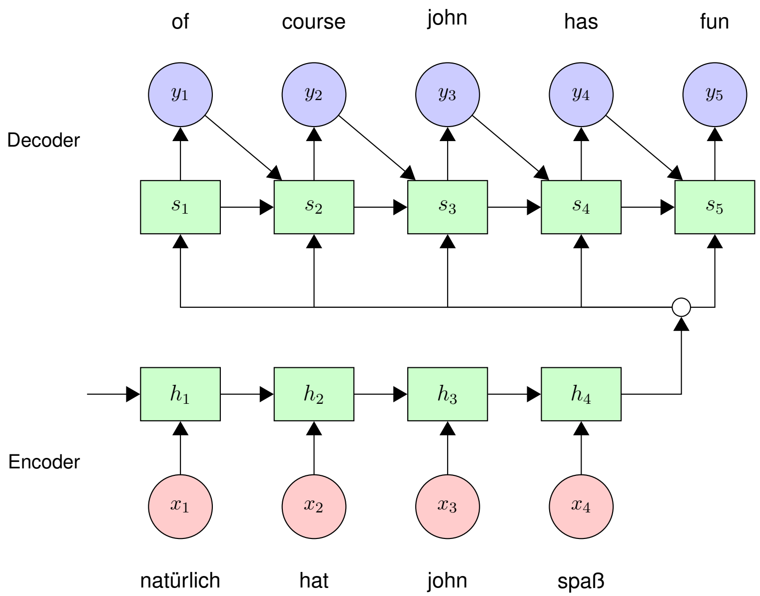
- We could just treat sentence pair as one long sequence, but:
  - We do not care about $p(S)$
  - We may want different vocabulary, network architecture for source text

→ Use separate RNNs for source and target.

Encoder-Decoder for Translation

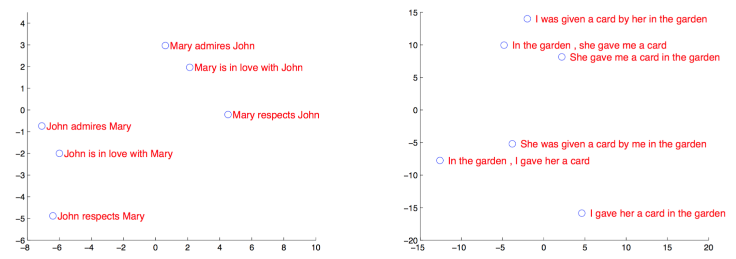


Encoder-Decoder for Translation



Summary vector

- Last encoder hidden-state “summarises” source sentence
- With multilingual training, we can potentially learn language-independent meaning representation

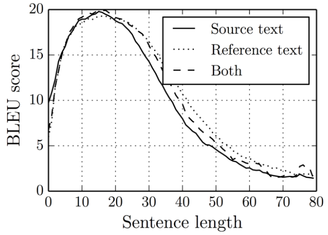


[Sutskever et al., 2014]

Summary vector as information bottleneck

Problem: Sentence Length

- Fixed sized representation degrades as sentence length increases
- Reversing source brings some improvement [Sutskever et al., 2014]

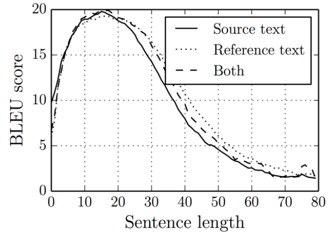


[Cho et al., 2014]

Summary vector as information bottleneck

Problem: Sentence Length

- Fixed sized representation degrades as sentence length increases
- Reversing source brings some improvement [Sutskever et al., 2014]

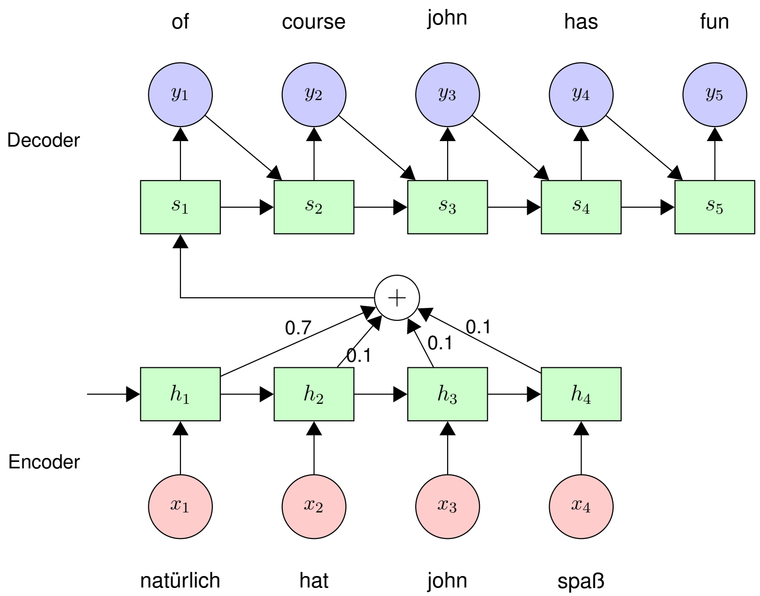


[Cho et al., 2014]

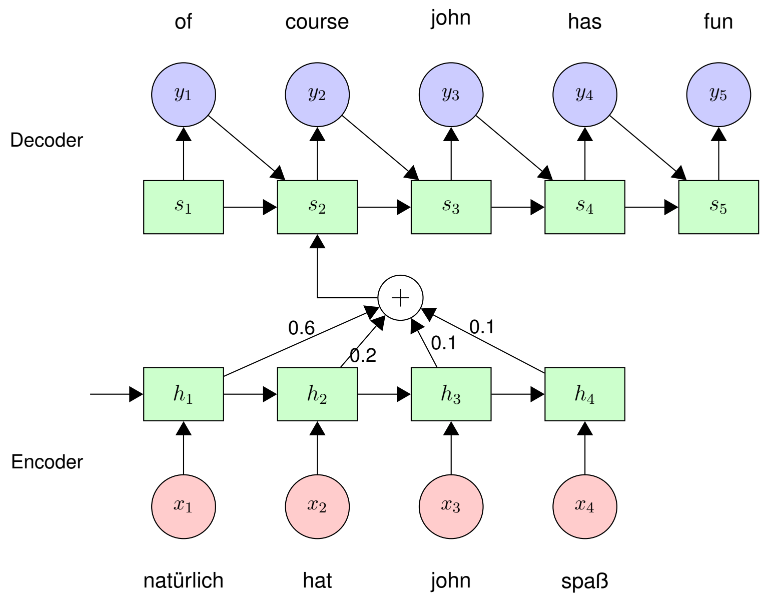
Solution: Attention

- Compute *context vector* as weighted average of source hidden states
- Weights computed by feed-forward network with softmax activation

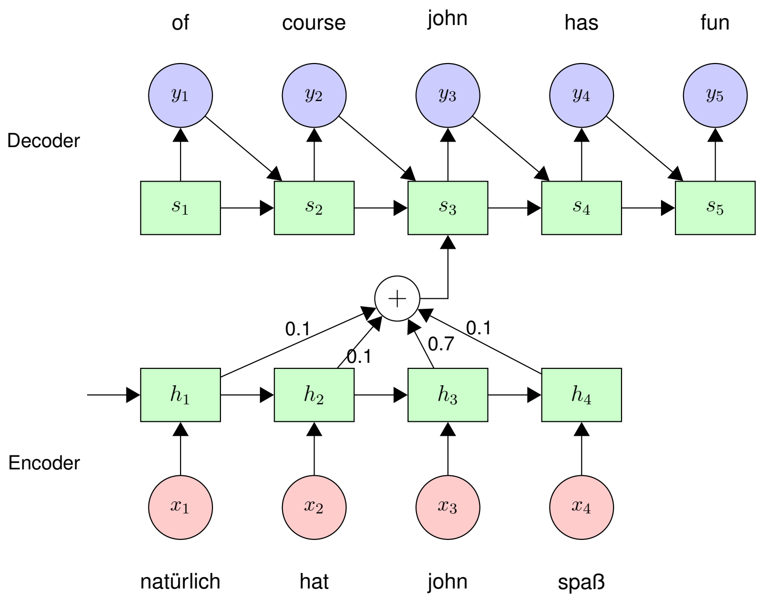
Encoder-Decoder with Attention



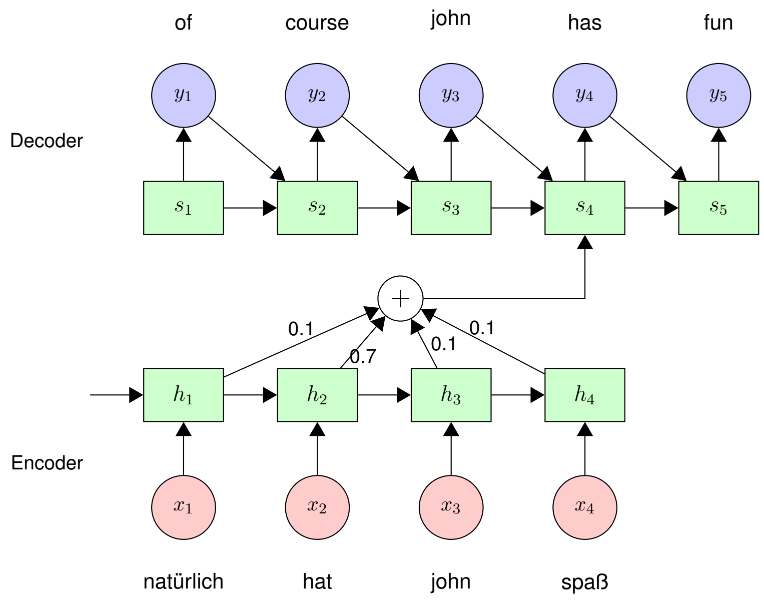
Encoder-Decoder with Attention



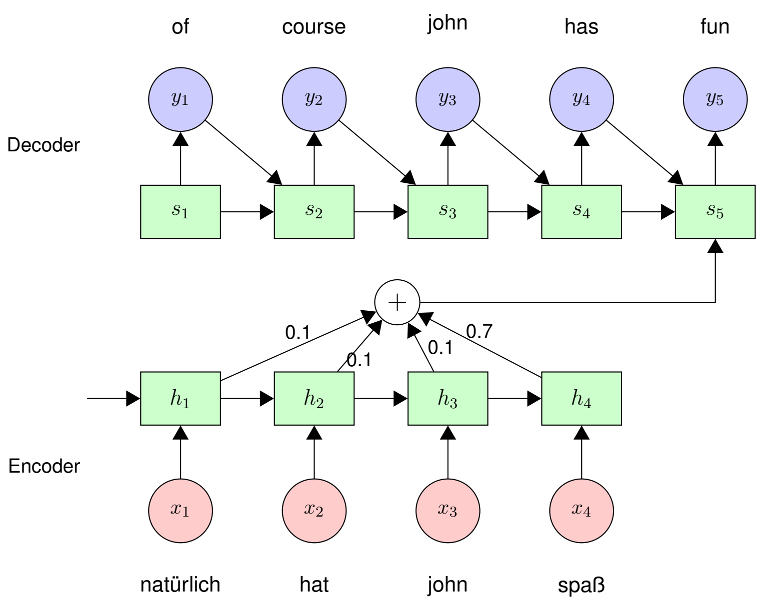
Encoder-Decoder with Attention



Encoder-Decoder with Attention



Encoder-Decoder with Attention



Attentional encoder-decoder: Maths

simplifications of model by [Bahdanau et al., 2015] (for illustration)

- plain RNN instead of GRU
- simpler output layer
- we do not show bias terms
- decoder follows *Look, Update, Generate* strategy [Sennrich et al., 2017]
- Details in `https://github.com/amunmt/amunmt/blob/master/contrib/notebooks/dl4mt.ipynb`

notation

- W, U, E, C, V are weight matrices (of different dimensionality)
  - E one-hot to embedding (e.g. $50000 \cdot 512$)
  - W embedding to hidden (e.g. $512 \cdot 1024$)
  - U hidden to hidden (e.g. $1024 \cdot 1024$)
  - C context (2x hidden) to hidden (e.g. $2048 \cdot 1024$)
  - V_o hidden to one-hot (e.g. $1024 \cdot 50000$)
- separate weight matrices for encoder and decoder (e.g. E_x and E_y)
- input X of length T_x; output Y of length T_y

Attentional encoder-decoder: Maths

encoder

$$\overrightarrow{h}_j = \begin{cases} 0, & \text{, if } j = 0 \\ \tanh(\overrightarrow{W}_x E_x x_j + \overrightarrow{U}_x h_{j-1}) & \text{, if } j > 0 \end{cases}$$

$$\overleftarrow{h}_j = \begin{cases} 0, & \text{, if } j = T_x + 1 \\ \tanh(\overleftarrow{W}_x E_x x_j + \overleftarrow{U}_x h_{j+1}) & \text{, if } j \le T_x \end{cases}$$

$$h_j = (\overrightarrow{h}_j, \overleftarrow{h}_j)$$

Attentional encoder-decoder: Maths

decoder

$$s_i = \begin{cases} \tanh(W_s \overleftarrow{h}_i), & \text{, if } i = 0 \\ \tanh(W_y E_y y_{i-1} + U_y s_{i-1} + C c_i) & \text{, if } i > 0 \end{cases}$$

$$t_i = \tanh(U_o s_i + W_o E_y y_{i-1} + C_o c_i)$$

$$y_i = \text{softmax}(V_o t_i)$$

attention model

$$e_{ij} = v_a^\top \tanh(W_a s_{i-1} + U_a h_j)$$

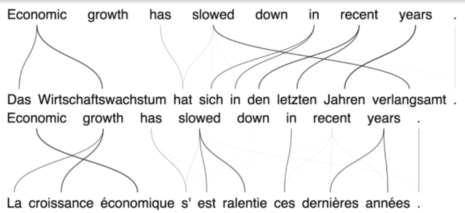
$$\alpha_{ij} = \text{softmax}(e_{ij})$$

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

Attention model

attention model

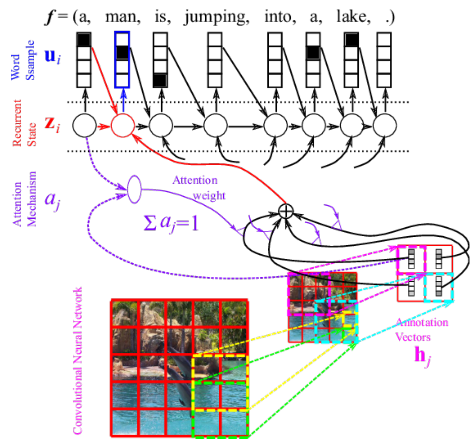
- side effect: we obtain alignment between source and target sentence
- information can also flow along recurrent connections, so there is no guarantee that attention corresponds to alignment
- applications:
  - visualisation
  - replace unknown words with back-off dictionary [Jean et al., 2015]
  - ...



Kyunghyun Cho
http://devblogs.nvidia.com/parallelforall/introduction-neural-machine-translation-gpus-part-3/

Attention model

attention model also works with images:



[Cho et al., 2015]

Attention model

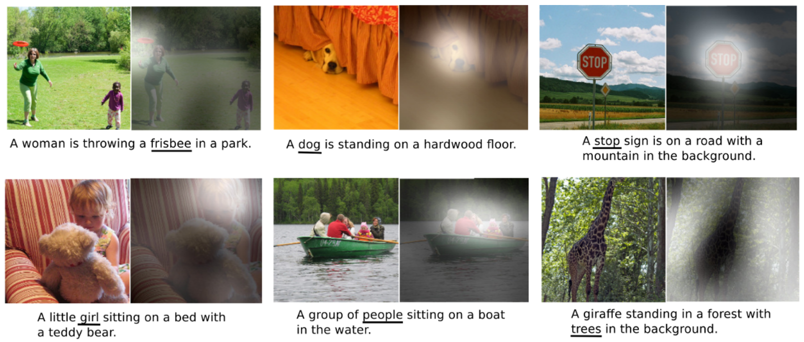


Fig. 5. Examples of the attention-based model attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word) [22]

[Cho et al., 2015]

Application of Encoder-Decoder Model

Scoring (a translation)

p(La, croissance, économique, s'est, ralentie, ces, dernières, années, . | Economic, growth, has, slowed, down, in, recent, year, .) = ?

Decoding (a source sentence)

Generate the most probable translation of a source sentence

$$y^* = \text{argmax}_y\, p(y|\text{Economic, growth, has, slowed, down, in, recent, year, .})$$

Decoding

exact search

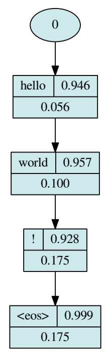
- generate every possible sentence T in target language
- compute score $p(T|S)$ for each
- pick best one

- intractable: $|\text{vocab}|^N$ translations for output length N
  $\rightarrow$ we need approximative search strategy

Decoding

approximative search/1: greedy search

- at each time step, compute probability distribution $P(y_i|S, y_{<i})$
- select y_i according to some heuristic:
  - sampling: sample from $P(y_i|S, y_{<i})$
  - greedy search: pick $\text{argmax}_y\, p(y_i|S, y_{<i})$
- continue until we generate *<eos>*



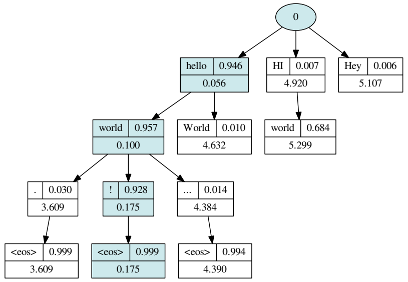
- efficient, but suboptimal

Decoding

approximative search/2: **beam search**

- maintain list of K hypotheses (beam)
- at each time step, expand each hypothesis k: $p(y_i^k|S, y_{<i}^k)$
- select K hypotheses with highest total probability:

$$\prod_i p(y_i^k|S, y_{<i}^k)$$



$K = 3$

- relatively efficient . . . beam expansion parallelisable
- currently default search strategy in neural machine translation
- small beam ($K \approx 10$) offers good speed-quality trade-off

Ensembles

- combine decision of multiple classifiers by voting
- ensemble will reduce error if these conditions are met:
  - base classifiers are accurate
  - base classifiers are diverse (make different errors)

Ensembles in NMT

- vote at each time step to explore same search space (better than decoding with one, reranking n-best list with others)
- voting mechanism: typically average (log-)probability

$$\log P(y_i|S, y_{<i}) = \frac{\sum_{m=1}^{M} \log P_m(y_i|S, y_{<i})}{M}$$

- requirements for voting at each time step:
  - same output vocabulary
  - same factorization of Y
  - but: internal network architecture may be different
- we still use reranking in some situations
  example: combine left-to-right decoding and right-to-left decoding

Further Reading

Required Reading

- Koehn, 13.5

Optional Reading

- Sequence to Sequence Learning with Neural Networks. (Sutskever, Vinyals, Le) : `https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf`
- Neural Machine Translation by Jointly Learning to Align and Translate. (Bahdanau, Cho, Bengio) : `https://arxiv.org/pdf/1409.0473.pdf`

Bibliography I

Bahdanau, D., Cho, K., and Bengio, Y. (2015).
Neural Machine Translation by Jointly Learning to Align and Translate.
In Proceedings of the International Conference on Learning Representations (ICLR).

Cho, K., Courville, A., and Bengio, Y. (2015).
Describing Multimedia Content using Attention-based Encoder-Decoder Networks.

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014).
On the Properties of Neural Machine Translation: Encoder–Decoder Approaches.
In Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Jean, S., Cho, K., Memisevic, R., and Bengio, Y. (2015).
On Using Very Large Target Vocabulary for Neural Machine Translation.
In
Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference o
pages 1–10, Beijing, China. Association for Computational Linguistics.

Junczys-Dowmunt, M. and Grundkiewicz, R. (2016).
Log-linear Combinations of Monolingual and Bilingual Neural Machine Translation Models for Automatic Post-Editing.
In Proceedings of the First Conference on Machine Translation, pages 751–758, Berlin, Germany. Association for Computational Linguistics.

Sennrich, R., Firat, O., Cho, K., Birch, A., Haddow, B., Hitschler, J., Junczys-Dowmunt, M., Läubli, S., Miceli Barone, A. V., Mokry, J., and Nadejde, M. (2017).
Nematus: a Toolkit for Neural Machine Translation.
In
Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational L
pages 65–68, Valencia, Spain.

Bibliography II

Sutskever, I., Vinyals, O., and Le, Q. V. (2014).
Sequence to Sequence Learning with Neural Networks.
In
Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014,
pages 3104–3112, Montreal, Quebec, Canada.