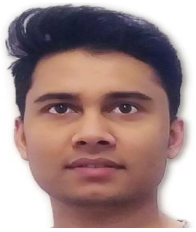
Deep Learning for NLP

TECHNISCHE UNIVERSITÄT DARMSTADT

Lecture 10 – Encoder-Decoder Neural Networks

- **Dr. Steffen Eger**
- **Niraj D Pandey**
- **Wei Zhao**



- Natural Language Learning Group (NLLG)
- Technische Universität Darmstadt

Last lectures

- Learning to represent words
- ConvNets for sentence classification
- RNNs for sequence tagging

Today

1. Auto-encoders
2. Encoder-Decoder architectures
3. Extensions & further applications

- Lecture based on:
  - Chapter 14, www.deeplearningbook.org
  - Stanford tutorials:
    https://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf
    http://ufldl.stanford.edu/wiki/index.php/Stacked_Autoencoders
  - NLP publications (as referenced)

Auto-encoders

- A neural network that is trained to attempt to copy its input to its output

  → I.e., learn the identity function, $F(\boldsymbol{x}) = \boldsymbol{x}$

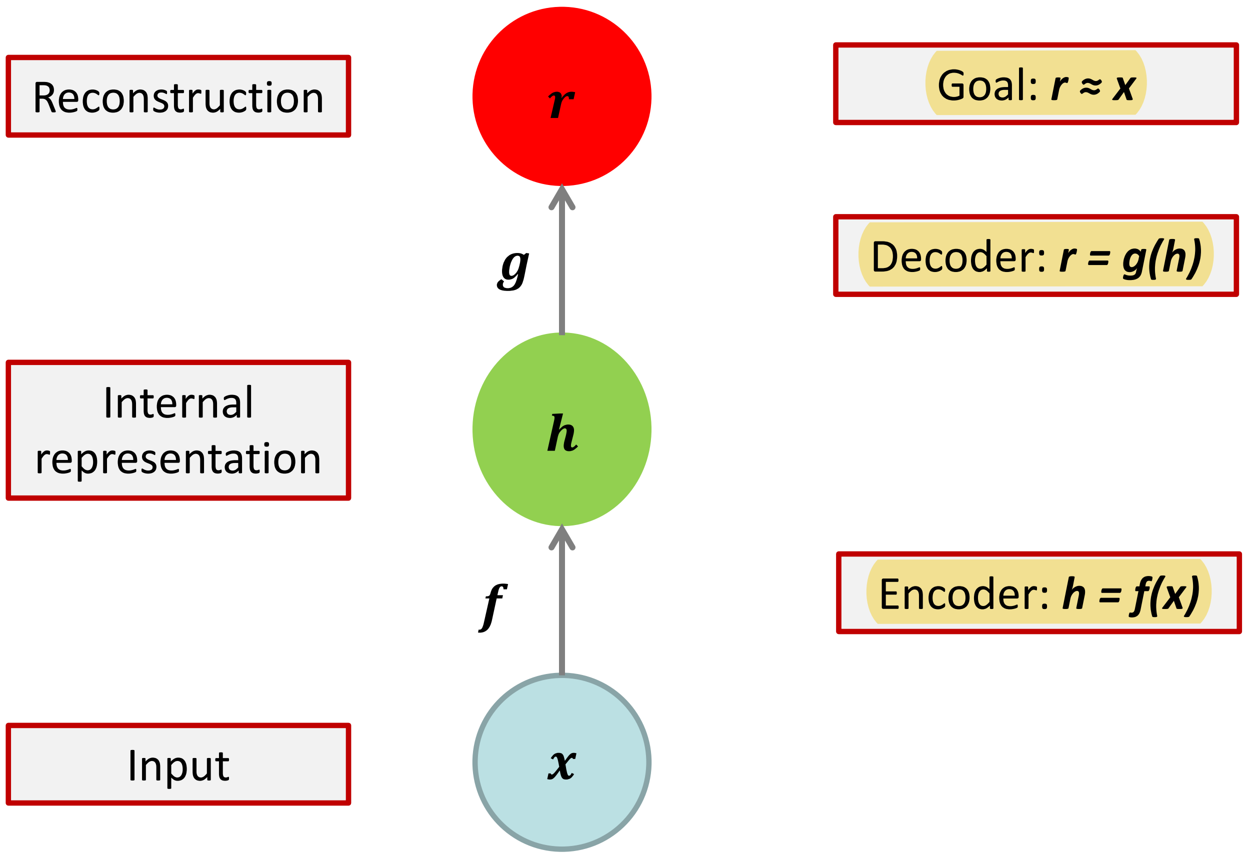
  → That is too easy (under which conditions?)

- Restrict the auto-encoder so that it can only copy approximately
  → need a bottleneck in order to learn successfully

- Traditionally used for dimensionality reduction or representation learning

Architecture

- Simple feed-forward network / MLP



Learning

- An autoencoder may be seen as situated in the **unsupervised** learning scenario

- Alternatively: supervised with self-defined auxiliary task (similarly to how we learn word embeddings)

  - The training data consists of pairs $(\boldsymbol{x}, \boldsymbol{x})$ or variants thereof

Undercomplete auto-encoder

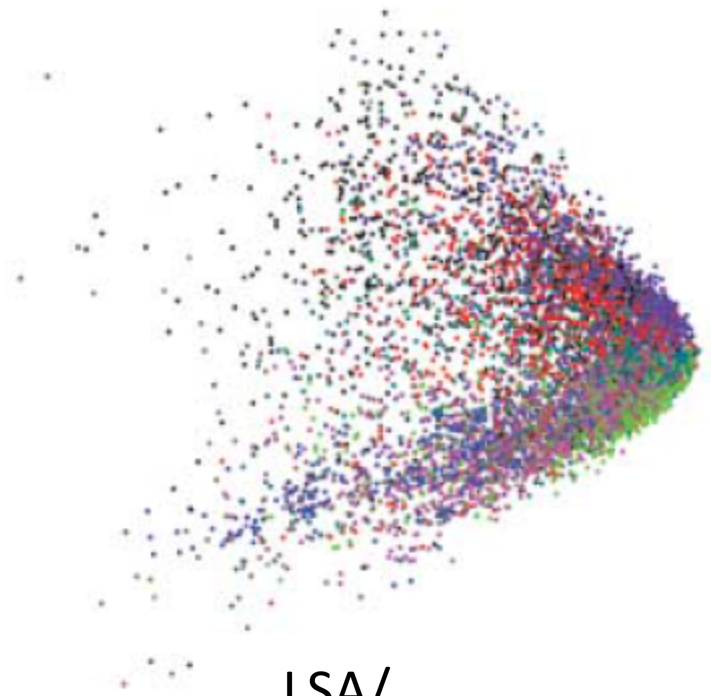
- **Undercomplete**: The hidden dimensionality is smaller than the input dimensionality

- With linear activation functions and square error loss, an undercomplete autoencoder learns [the same as] PCA.

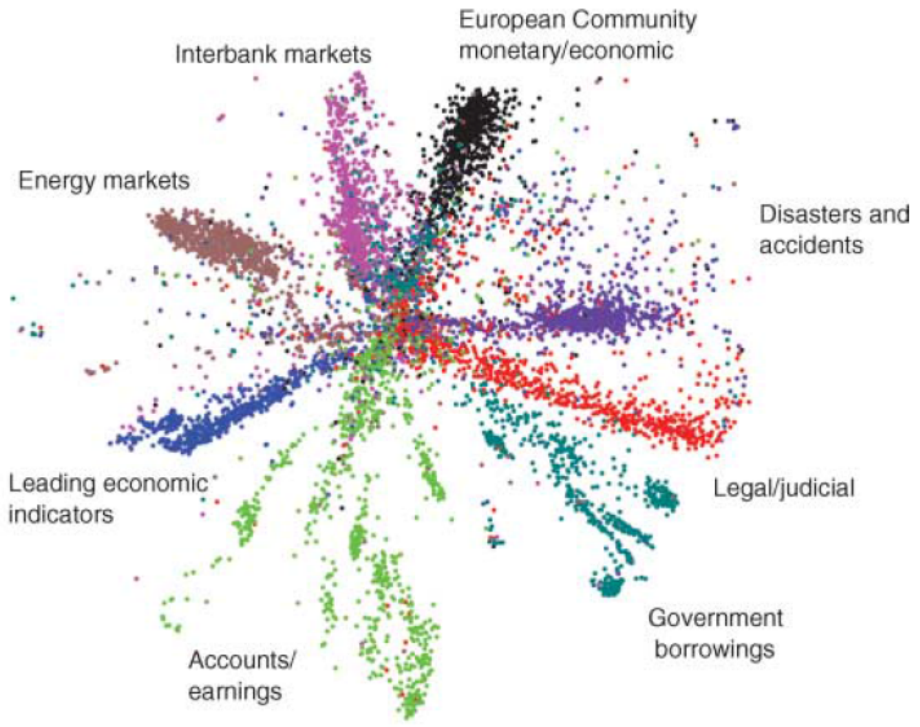
- Auto-encoders with non-linear encoder and decoder functions can be seen as a (more powerful) generalization of PCA.

Auto-encoders vs. PCA

- Articles from the Reuter corpus were mapped to a 2000 dimensional vector, using the most common word stems



LSA/
PCA



Deep Autoencoder

Hinton et al., Reducing the Dimensionality of Data with Neural Networks

Denoising auto-encoders

- Add some random noise to the input

$$\tilde{\boldsymbol{x}} \leftarrow \mathbf{noise}(\boldsymbol{x})$$

- The auto-encoder should learn to "remove the noise", i.e., reconstruct input from a corrupted version
  - This forces the network to actually learn something even when hidden dimensionality is $\geq$ input dimensionality

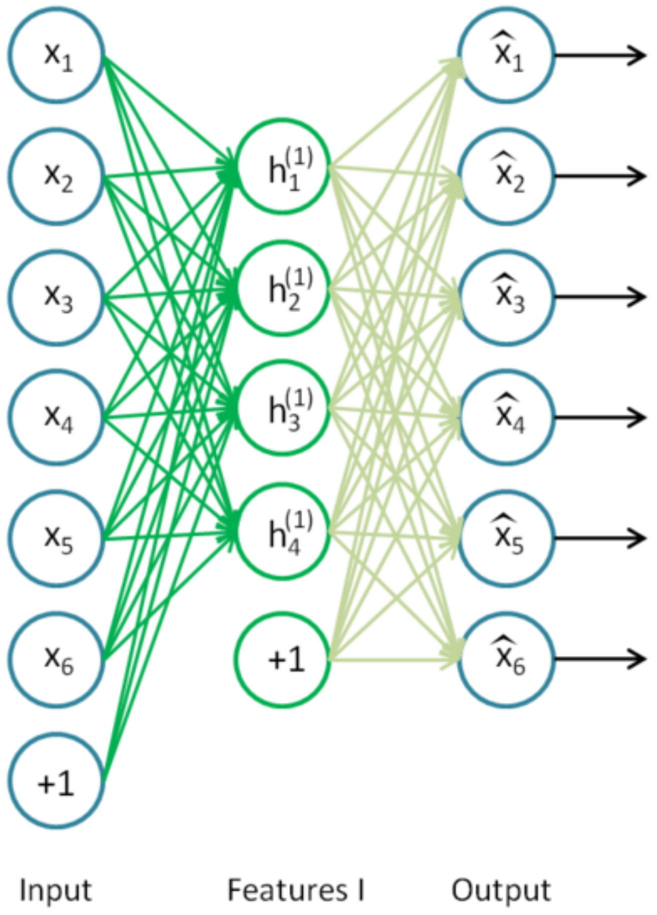
Deep auto-encoders

- Stack multiple hidden layers.
- Train each hidden layer independently as an auto-encoder.

- We will go through an example from http://ufldl.stanford.edu/wiki/index.php/Stacked_Autoencoders

Layerwise training

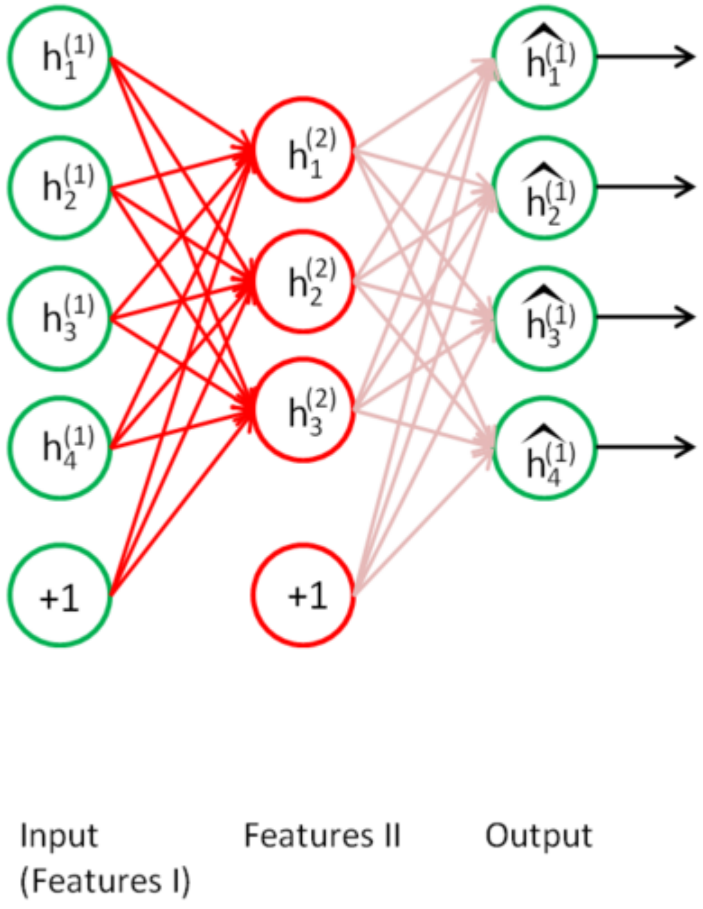
- Step 1: Train a sparse auto-encoder on the input.



http://ufldl.stanford.edu/wiki/index.php/Stacked_Autoencoders

Layerwise training

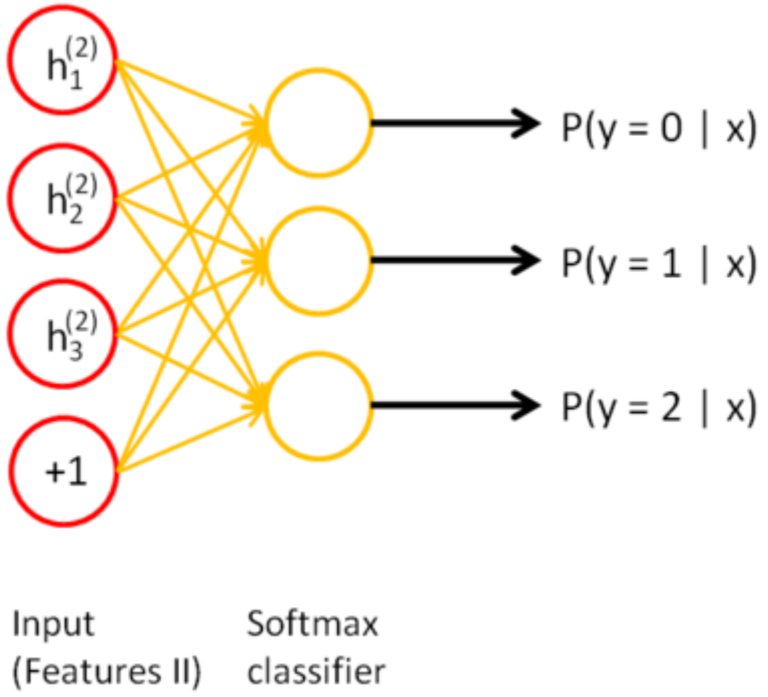
- Step 1: Train a sparse auto-encoder on the input.
- Step 2: Use the hidden layer (the features) of step 1 as input for a second auto-encoder.
- (optional: repeat steps 1 and 2 for more hidden layers)



http://ufldl.stanford.edu/wiki/index.php/Stacked_Autoencoders

Layerwise training

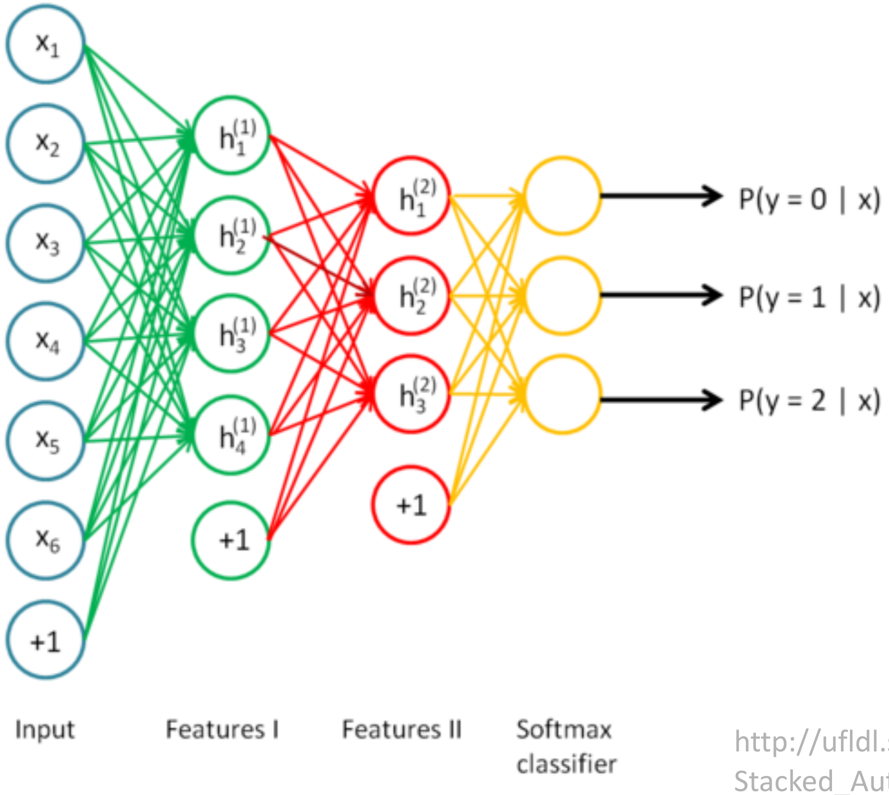
- Step 1: Train a sparse auto-encoder on the input.
- Step 2: Use the hidden layer (the features) of step 1 as input for a second auto-encoder.
- (optional: repeat step 2 for more hidden layers)
- Step 3: calculate a softmax classifier (only) on the last layer



http://ufldl.stanford.edu/wiki/index.php/Stacked_Autoencoders

Stacked auto-encoder

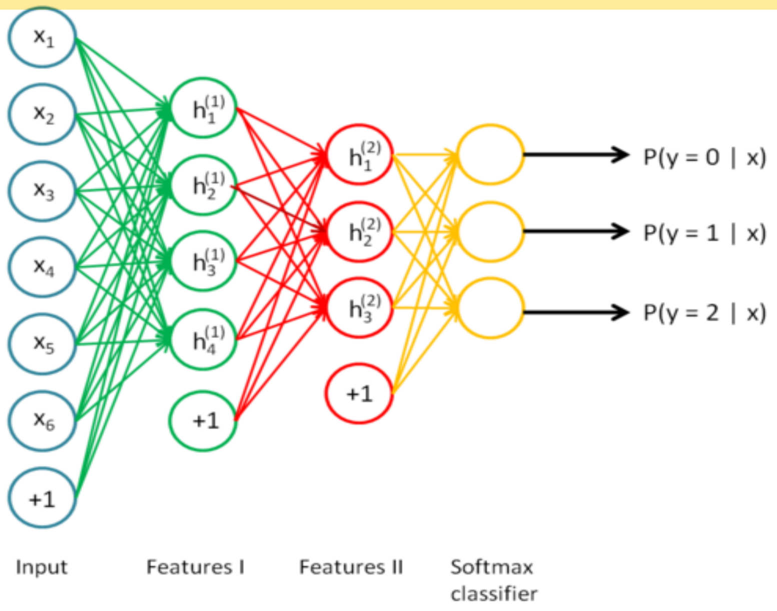
- Stack it all together
- Finetune the complete network.



http://ufldl.stanford.edu/wiki/index.php/Stacked_Autoencoders

Stacked auto-encoder

- Layerwise pre-training led to the first big improvements in DL
- Nowadays it loses importance because researchers have gained better intuition for **initializing** the parameters and **bigger datasets** are becoming available (also we have better **regularization** techniques + better **optimizers**, better **activation fncts**, etc.).



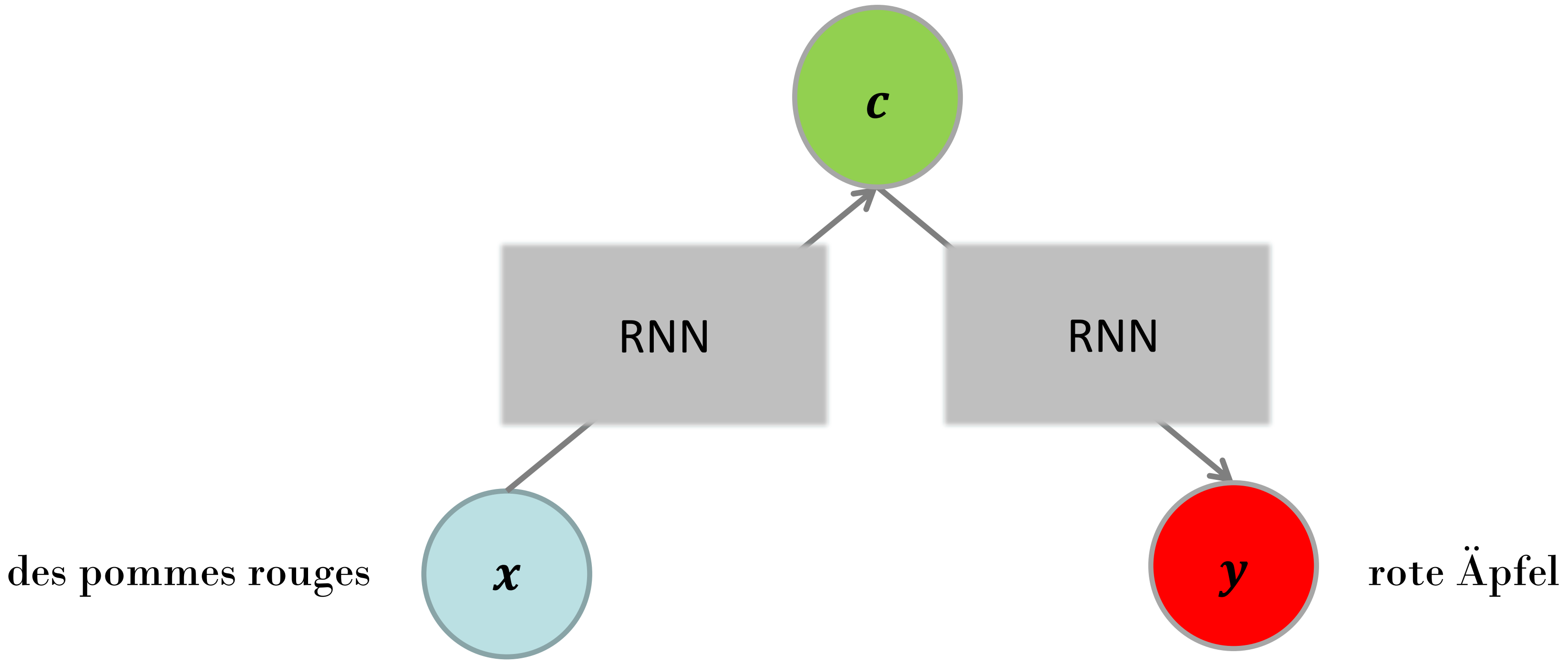
http://ufldl.stanford.edu/wiki/index.php/Stacked_Autoencoders

Agenda

1. Autoencoder
2. **Encoder-Decoder architectures**
   - Noisy-channel model
   - **Machine translation (MT)**
     - Cho, K., Gulcehre, B. V. M. C., Bahdanau, D., Schwenk, F. B. H., & Bengio, Y. (2014): Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. EMNLP.
     - Sutskever et al. 2014, Sequence-to-Sequence Learning with Neural Nets
     - Bahdanau, Dzmitry, Kyunghyun Cho, Yoshua Bengio (2015): Neural Machine Translation by Jointly Learning to Align and Translate. ICLR. http://arxiv.org/abs/1409.0473
3. Extensions & Further applications

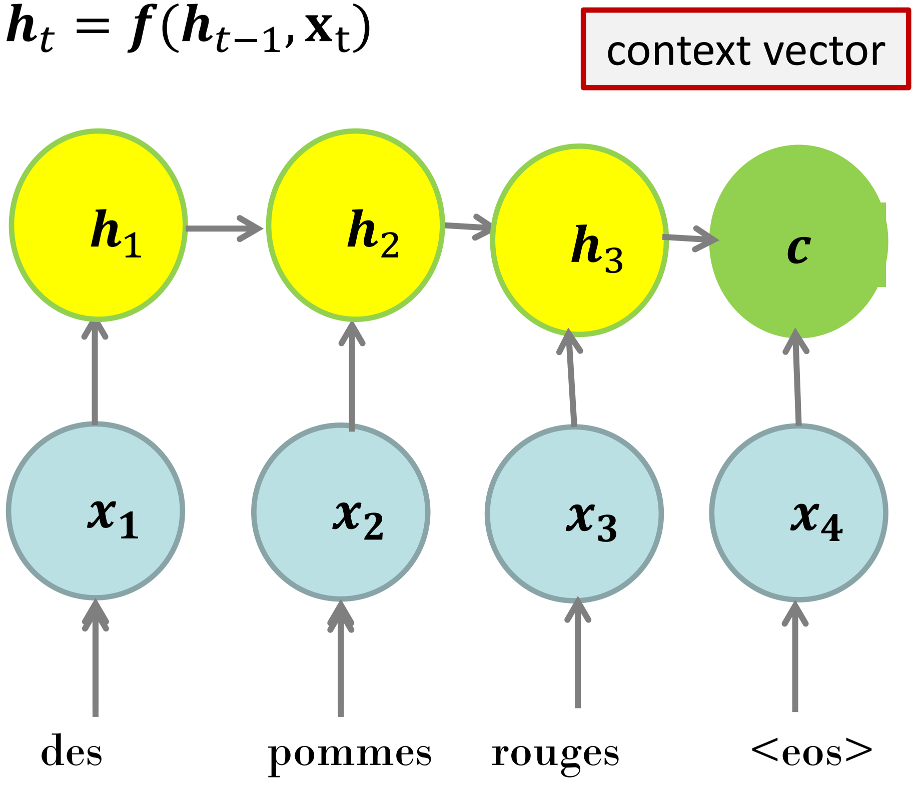
Encoder-Decoder architecture

- Instead of feedforward networks we can also have more complex architectures for the encoder and the decoder e.g. recurrent neural networks or LSTMs



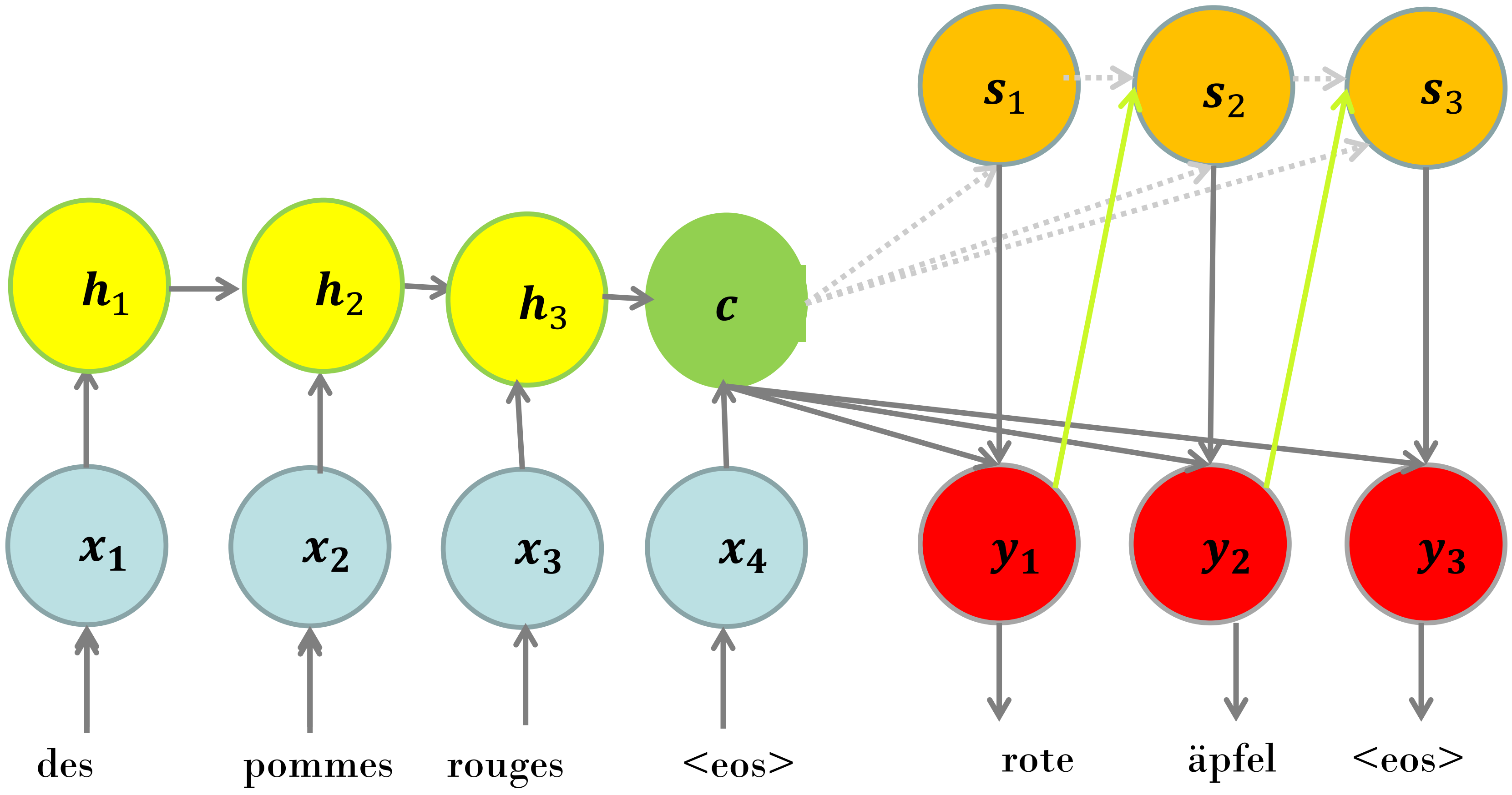
RNN Encoder

- Read in the whole input sequence.
- Learn a context vector $\boldsymbol{c}$
- $\boldsymbol{h}_t = \boldsymbol{f}(\boldsymbol{h}_{t-1}, \mathbf{x}_\mathrm{t})$



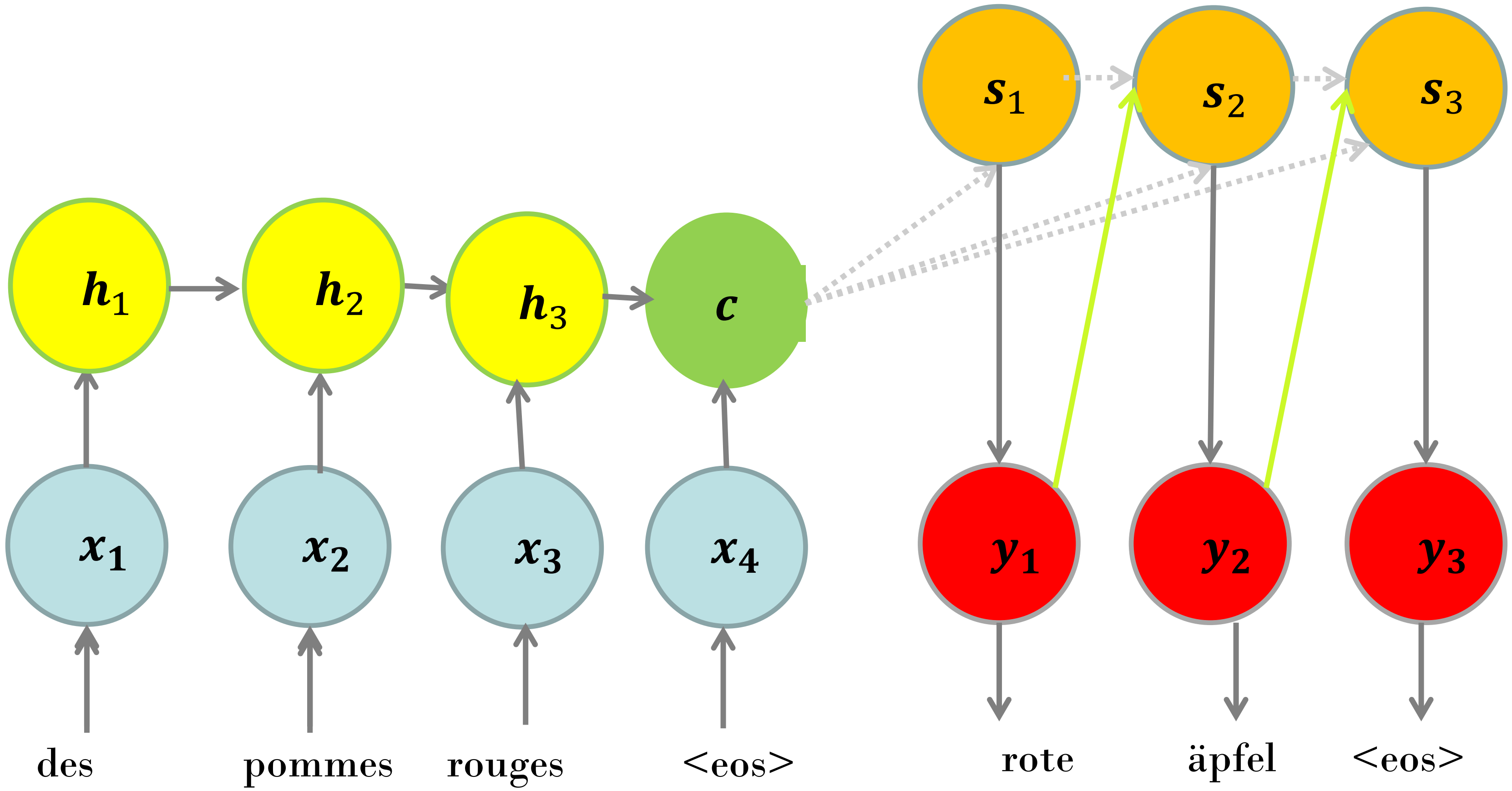
RNN Encoder-Decoder architecture

- Predict one output word at a time.



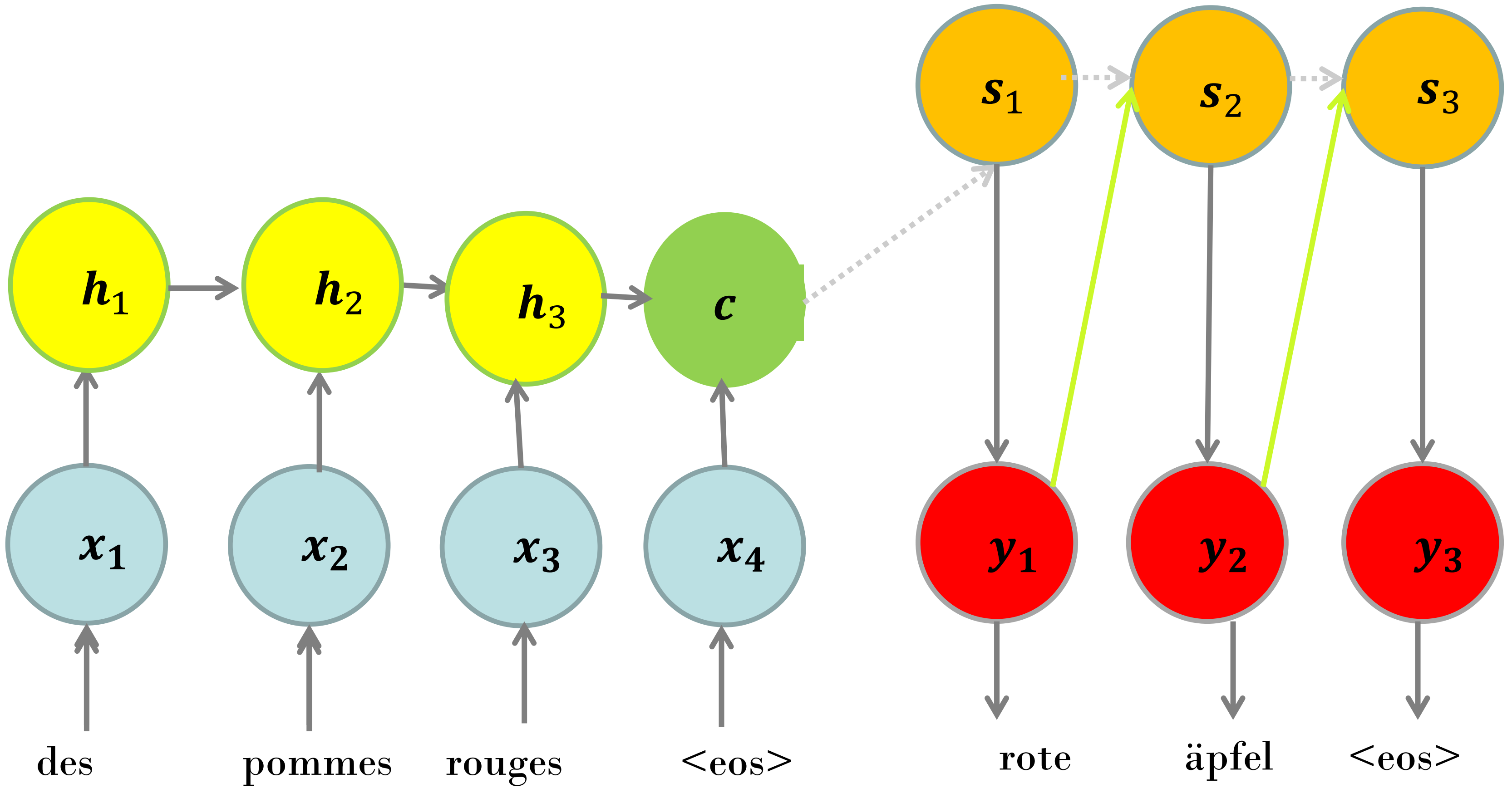
RNN Encoder-Decoder architecture

- Different variants



RNN Encoder-Decoder architecture

- Different variants



Encoder-Decoder architecture

- Cho et al. used this architecture for
  - translating phrases
  - **scoring phrases from another MT system**
- Problems
  - Long range dependencies

  → Problems especially with long input sequences
  - Context vector ***c*** has a fixed length
  - Input size is variable

  → Problems especially with long input sequences

Encoder-Decoder architecture:

- Summary of basic idea:
  - Two RNNs (could be other architectures as well, like Transformer)
  - The one encodes the input sentence (can think of it as a sentence embedding)
  - The other decodes the “sentence embedding” (i.e., the context vector retrieved by the encoder)
    - The decoder is like a language model
    - It typically chooses the **max** element at each time step
      - Rather than sampling probabilistically → since we want the best output given the fixed input

Encoder-Decoder architecture

- Sutskever et al. 2014, Sequence-to-Sequence Learning with Neural Nets

- Their core contribution is to **reverse** the *input sequence* during training and testing

    - The cat sat on the mat → mat the on sat cat The

Encoder-Decoder architecture

- Sutskever et al. 2014, Sequence-to-Sequence Learning with Neural Nets

- Their core contribution is to **reverse** the *input sequence* during training and testing

  - The cat sat on the mat → mat the on sat cat The

- On average, the dependencies have the same length whether we reverse or not

Encoder-Decoder architecture

- Sutskever et al. 2014, Sequence-to-Sequence Learning with Neural Nets

- The cat sat on the mat vs. Die Katze saß auf der Matte
  - Dist(The,Die) = 6
  - Dist(cat,Katze) = 6 ....

- mat the on sat cat The vs. Die Katze saß auf der Matte
  - Dist(The,Die) = 1
  - Dist(cat,Katze) = 2 ....
  - Dist(mat,Matte) = 12

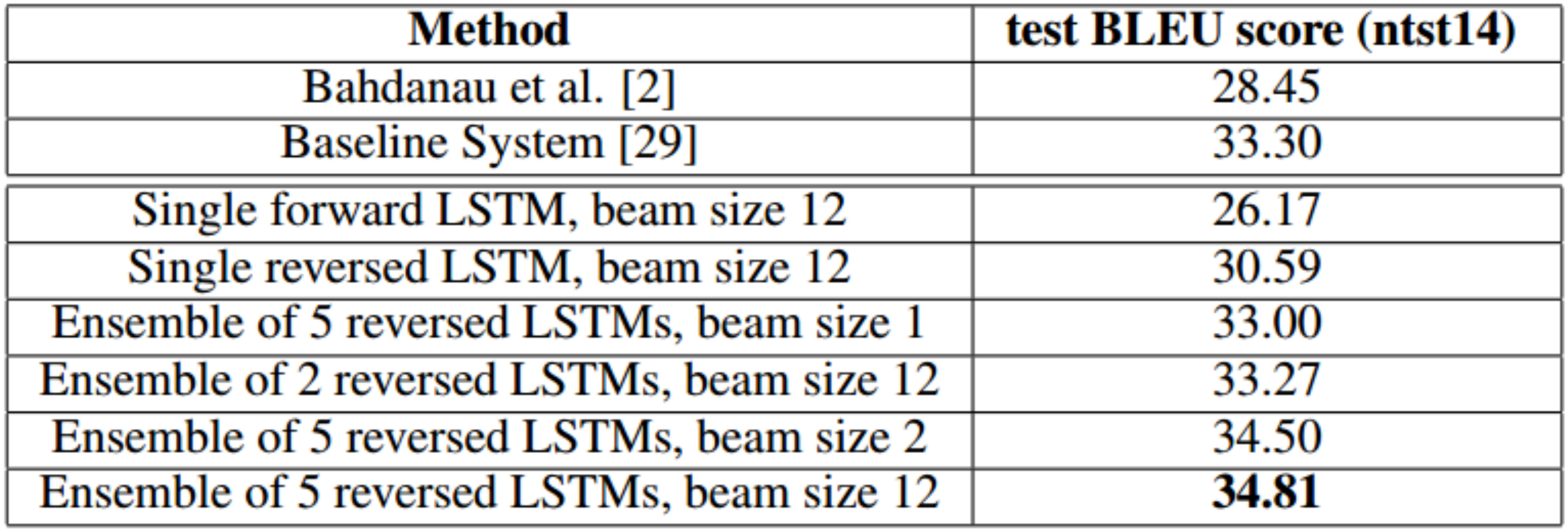
Encoder-Decoder architecture

- Sutskever et al. 2014, Sequence-to-Sequence Learning with Neural Nets

- While the effect is not entirely clear, Sutskever et al. speculate that
  - Minimal time lag is now greatly reduce (not the average one)
  - This way, it's easier to "establish communication" between source and target sentences

Encoder-Decoder architecture

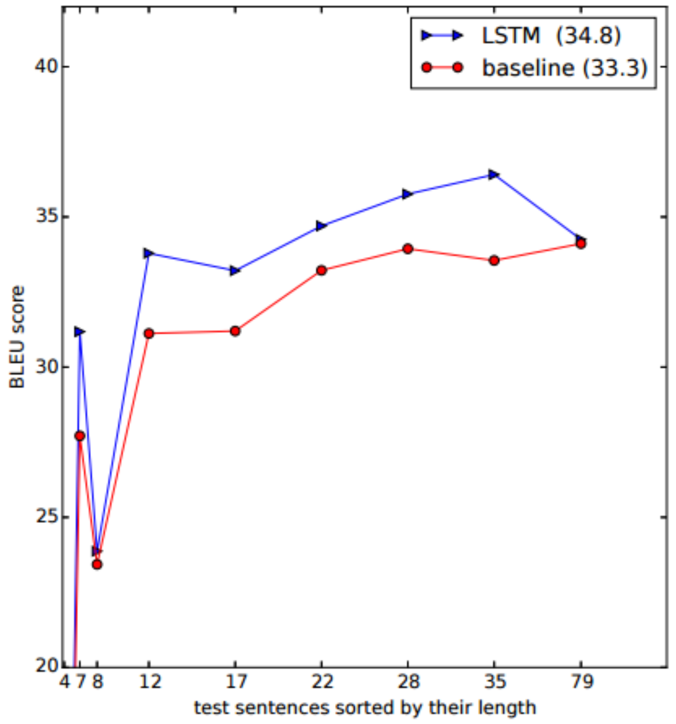
- Sutskever et al. 2014, Sequence-to-Sequence Learning with Neural Nets

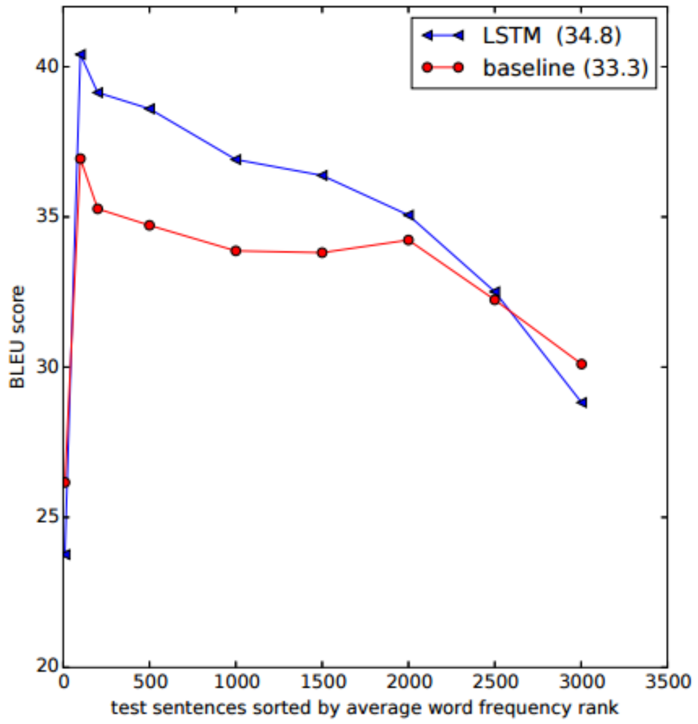


Method	test BLEU score (ntst14)
Bahdanau et al. [2]	28.45
Baseline System [29]	33.30
Single forward LSTM, beam size 12	26.17
Single reversed LSTM, beam size 12	30.59
Ensemble of 5 reversed LSTMs, beam size 1	33.00
Ensemble of 2 reversed LSTMs, beam size 12	33.27
Ensemble of 5 reversed LSTMs, beam size 2	34.50
Ensemble of 5 reversed LSTMs, beam size 12	**34.81**

Encoder-Decoder architecture

- Sutskever et al. 2014, Sequence-to-Sequence Learning with Neural Nets





Problems

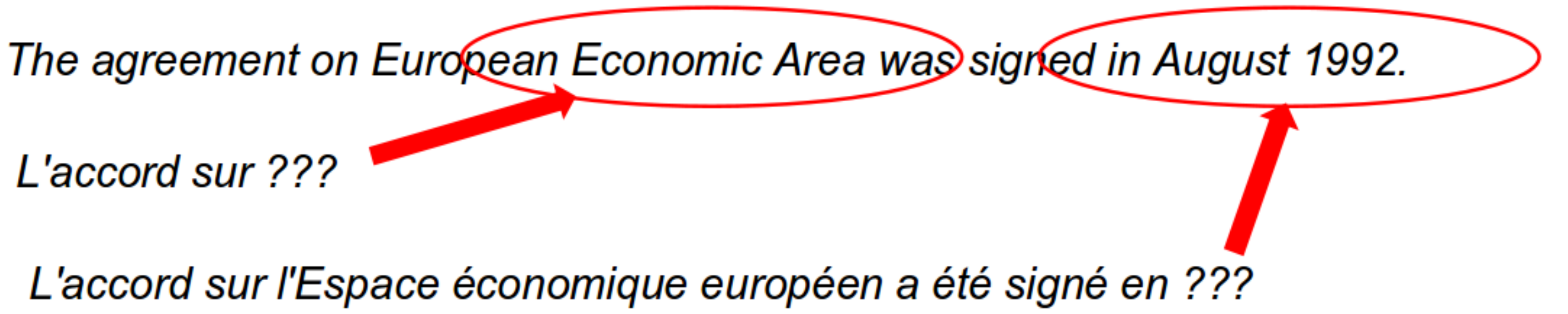
- A major problem with the solutions so far is the global vector $\boldsymbol{c}$:
  - All the input is encoded in a single (big) vector that summarizes the whole sentence
  - However, when we translate a word, it typically only depends on a *local* neighborhood in the source sentence, not on the complete source sentence

Die Katze saß auf der Matte: depends only on cat

Die Katze saß auf der Matte: depends only on The and cat

Problems & an Idea

- Only pay attention to the elements of the input that are relevant for producing the **current** output.


The agreement on European Economic Area was signed in August 1992.
L'accord sur ???
L'accord sur l'Espace économique européen a été signé en ???

Problems & an Idea

TECHNISCHE UNIVERSITÄT DARMSTADT

- A major problem with all presented solutions so far is the global vector $\boldsymbol{c}$:
  - Ideally, we would make use of some sort of **alignment** information during training and testing
  - That's the idea of Bahdanau, Dzmitry, Kyunghyun Cho, Yoshua Bengio (2015): Neural Machine Translation by Jointly Learning to Align and Translate. ICLR. http://arxiv.org/abs/1409.0473

"The decoder decides parts of the source sentence to pay attention to. By letting the decoder have an attention mechanism, we relieve the encoder from the burden of having to encode all information in the source sentence into a fixed- length vector. "

http://www.iclr.cc/lib/exe/fetch.php?media=iclr2015:bahdanau-iclr2015.pdf

Joint Alignment and Translation

$$\boldsymbol{y}_t = \mathbf{g}(\boldsymbol{s}_t, \boldsymbol{y}_{t-1})$$

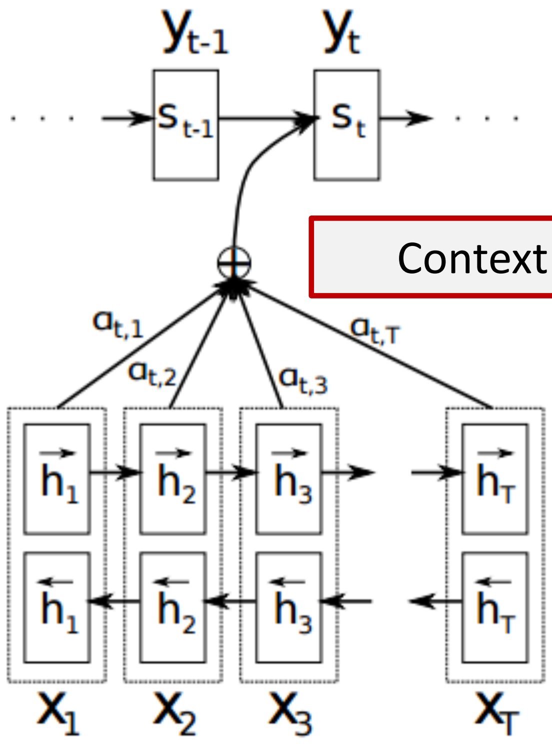
$$\boldsymbol{s}_t = \boldsymbol{f}(\boldsymbol{s}_{t-1}, \boldsymbol{y}_{t-1}, \boldsymbol{c}_t)$$

"It should be noted that unlike the existing encoder–decoder approach, here the probability is conditioned on a distinct context vector ***c***$_t$ *for each target word y*$_t$*."*

Learning the context vector

Generate t-th target word y_t give source sentence $(x_1, \ldots, x_T)$

Concat the two hidden states of the bidirectional RNN



y_{t-1}
y_t
s_{t-1}
s_t
$\alpha_{t,1}$
$\alpha_{t,2}$
$\alpha_{t,3}$
$\alpha_{t,T}$
$\overrightarrow{h}_1$ $\overrightarrow{h}_2$ $\overrightarrow{h}_3$ $\overrightarrow{h}_T$
$\overleftarrow{h}_1$ $\overleftarrow{h}_2$ $\overleftarrow{h}_3$ $\overleftarrow{h}_T$
x_1 x_2 x_3 x_T

Context vector $\boldsymbol{c}_t$

http://arxiv.org/pdf/1409.0473v7.pdf

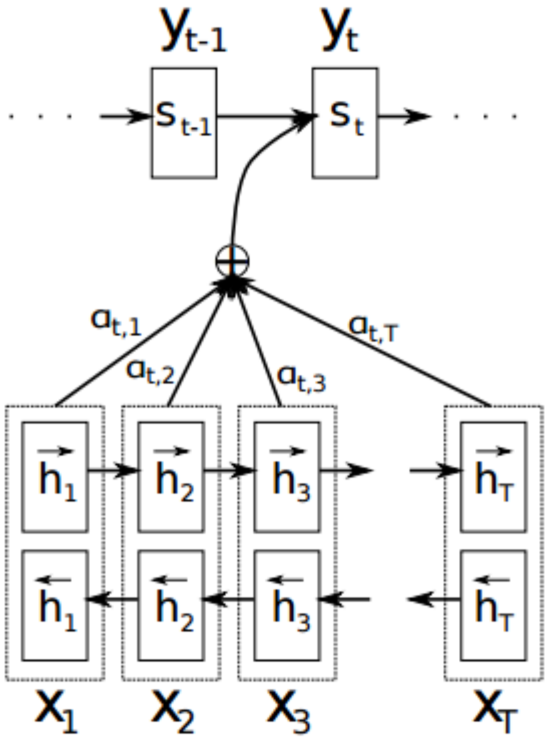
Context vector:

$$\boldsymbol{c}_t = \sum_{j=1}^{T} \alpha_{tj} \boldsymbol{h}_j$$

The context vector varies depending on the relation α_{tj} between the current output y_t and the input words x_j.

Learning the context vector with attention

What is α_{tj}?
Has become popular as "attention".



http://arxiv.org/pdf/1409.0473v7.pdf

Context vector:

$$\boldsymbol{c}_t = \sum_{j=1}^{T} \alpha_{tj} \boldsymbol{h}_j$$

How is the attention vector computed?

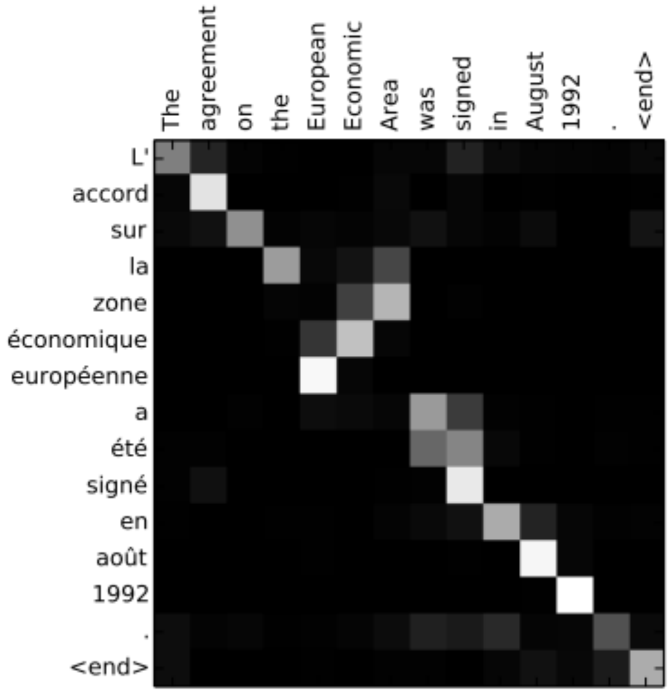
- $\alpha_{tj} = \frac{\exp(e_{tj})}{\sum_k \exp(e_{tk})}$
  - where $e_{tj} = a(\boldsymbol{s}_{t-1}, \boldsymbol{h}_j)$ indicates how likely target word t (which we want to produce at the current time step) is aligned to input word j
  - "We parametrize the alignment model a as a feedforward network which is **jointly** trained with the other components of the proposed system"
  - α_{tj} (and e_{tj}) are also termed **soft alignments**

Illustration: Soft alignments

- Soft alignments: probability distribution over tokens that a current token aligns to
- Hard alignments: “degenerate prob. distribution” with 0/1 values (NB may still align to **many** tokens)

The English words

- **Hard alignment** example:
  - L’ → (1,0,0,0,0,…)
  - été → (0,0,0,1,1,0,0,….)



Soft alignment

Agenda

1. Autoencoder
2. Encoder-Decoder architectures
   - Noisy-channel model
   - Machine translation (MT)
     - Cho, K., Gulcehre, B. V. M. C., Bahdanau, D., Schwenk, F. B. H., & Bengio, Y. (2014): Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. EMNLP.
     - Sutskever et al. 2014, Sequence-to-Sequence Learning with Neural Nets
     - Bahdanau, Dzmitry, Kyunghyun Cho, Yoshua Bengio (2015): Neural Machine Translation by Jointly Learning to Align and Translate. ICLR. http://arxiv.org/abs/1409.0473
3. **Extensions & Further applications**

Variants & Extensions

- In the following we survey (few) further extensions of the encoder-decoder approach
- Or other tasks (other than MT) to which they have been applied

Seq2Seq tasks

- Encoder-Decoder models are extremely general

  - Cannot only translate from English to German
  - But also many other “Sequence-to-Sequence” tasks (Seq2Seq) such as

    - Lemmatization, Grapheme-to-Phoneme conversion, Spelling correction, etc.

  - Of course, you could also POS tag with them, etc.

Seq2Seq tasks

- Encoder-Decoder models are extremely general
    - But also many other “Sequence-to-Sequence” tasks (Seq2Seq) such as
        - Lemmatization, Grapheme-to-Phoneme conversion, Spelling correction, etc.

TASK	EXAMPLE
Lemmatization	g e s p i e l t → s p i e l e n
G2P	s c h u h → S u:
Spelling correction	I _ l v o e _ u → I _ l o v e _ y o u

Encoder-Decoder architectures

- Chung et al. (2016), A character-level decoder without explicit segmentation for neural machine translation

  - Try out **character-based** MT with encoder-decoder models

  - Advantages of character-based approaches:
    - Can (better) predict OOV and rare words
    - Can better deal with morphological variants:
      - run, runs, running, ran

  - Disadvantages: state-space may explode, long range deps

Encoder-Decoder architectures

- Chung et al. (2016), A character-level decoder without explicit segmentation for neural machine translation

  - Show that char-level encoder-decoder systems can:
    - Generate long, coherent sentences
    - Soft-aligns well between a source word and its target characters

Encoder-Decoder architectures

- Chung et al. (2016), A character-level decoder without explicit segmentation for neural machine translation

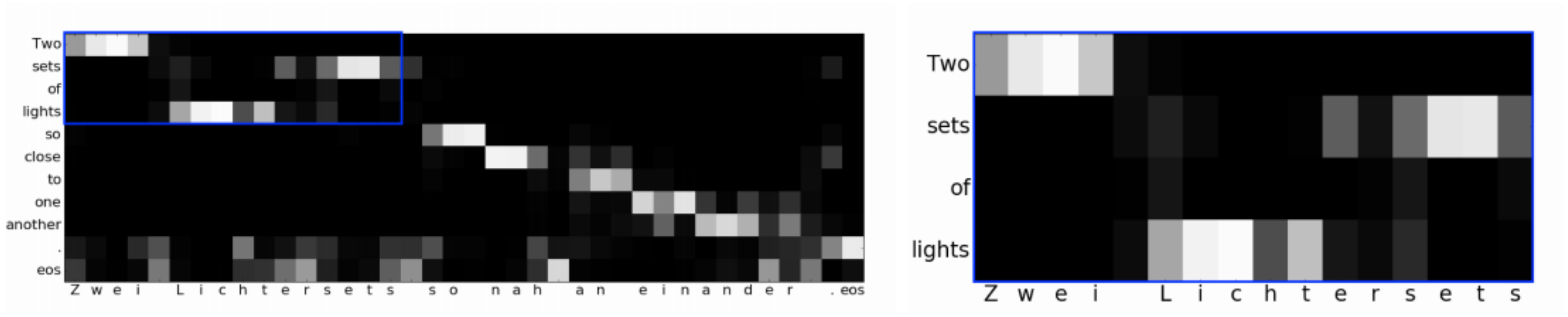
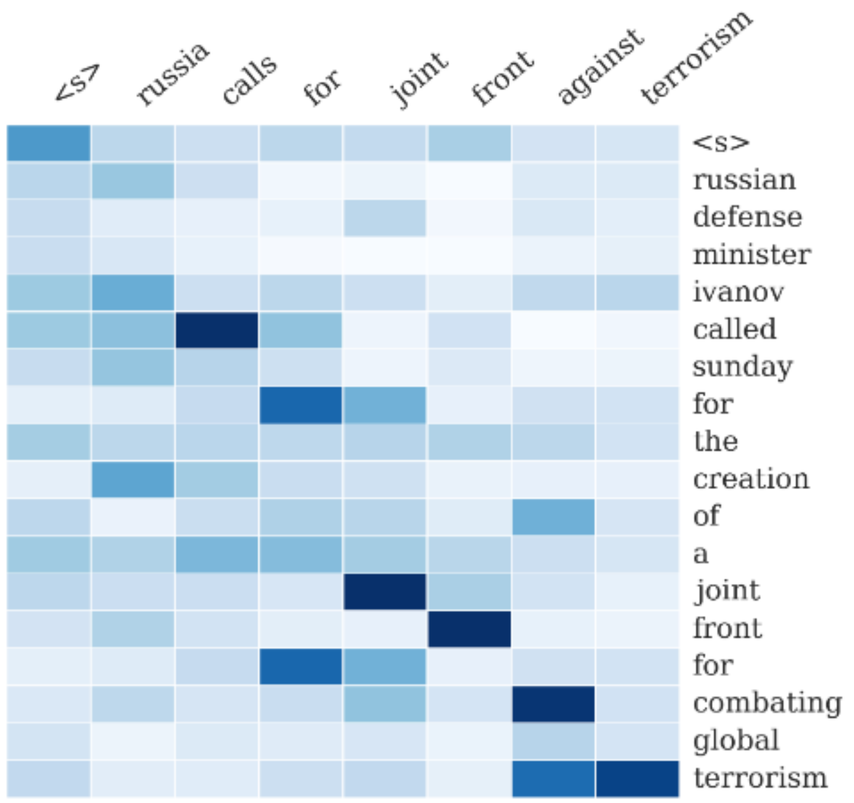


Figure 3: Alignment matrix of a test example from En-De using the BPE→Char (bi-scale) model.

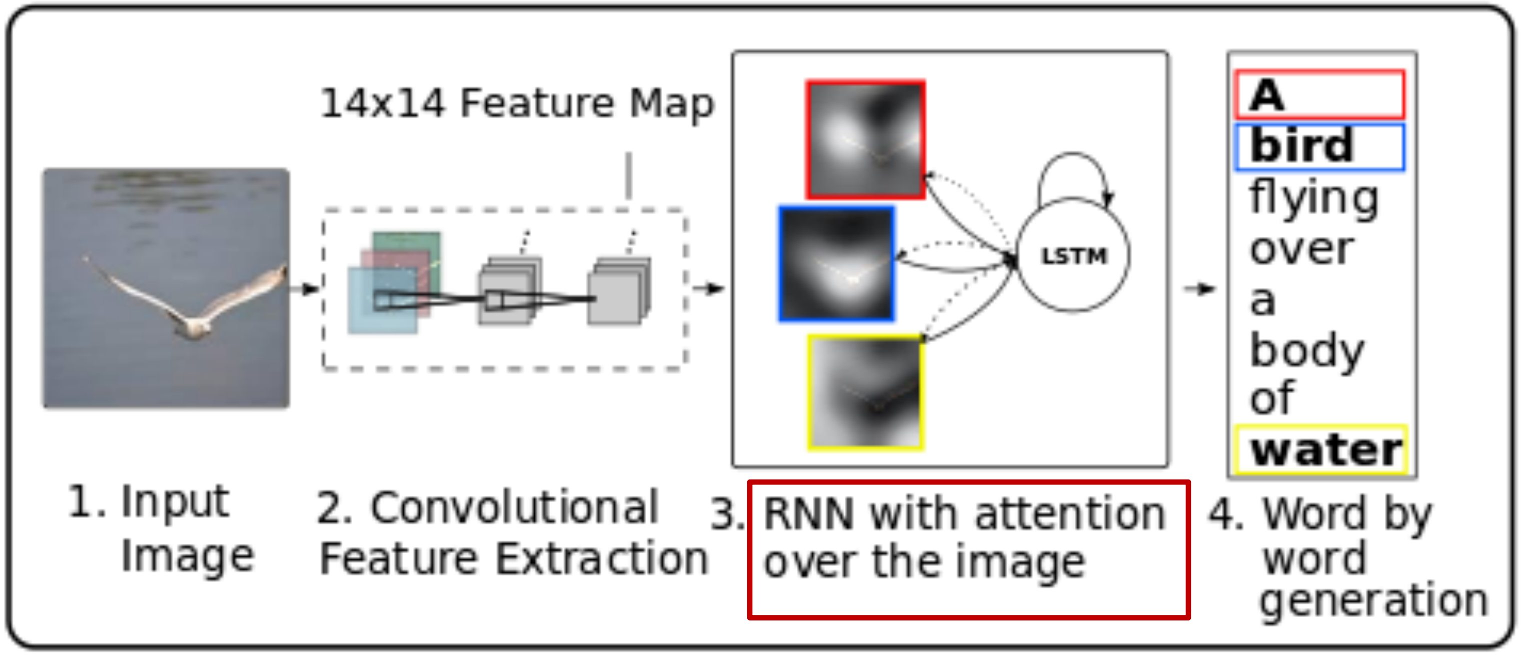
Sequence-to-sequence tasks: Summarization

- Alexander M. Rush, Sumit Chopra, Jason Weston (2015):
  A Neural Attention Model for Abstractive Sentence Summarization
  http://www.aclweb.org/anthology/D15-1044



Multi-modal attention: Image Caption Generation

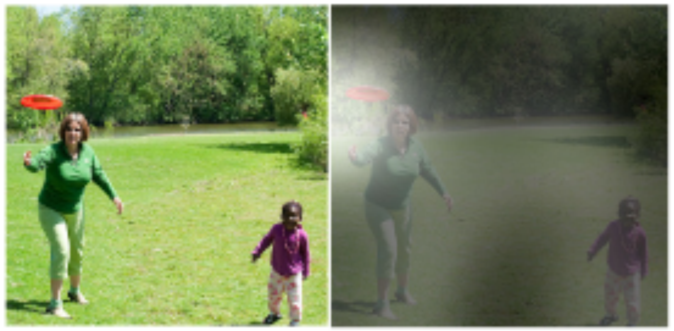
- Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, Yoshua Bengio (2016): **Show, Attend and Tell: Neural Image Caption Generation with Visual Attention**, http://arxiv.org/abs/1502.03044

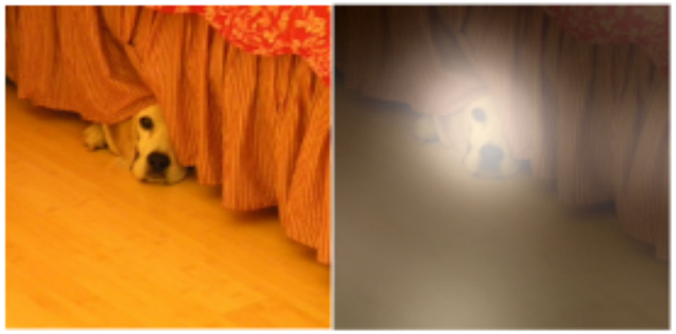

14x14 Feature Map
LSTM
A
bird
flying
over
a
body
of
water
1. Input Image
2. Convolutional Feature Extraction
3. RNN with attention over the image
4. Word by word generation

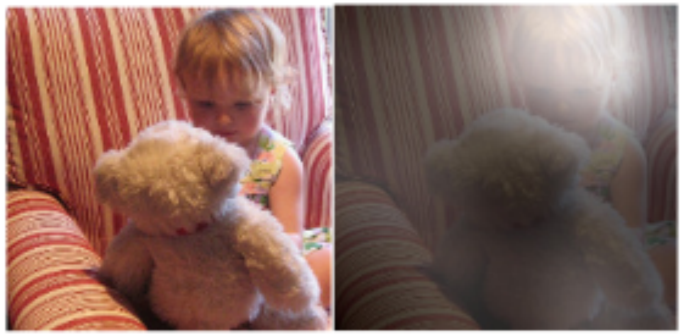
Attention-based word sequence generation

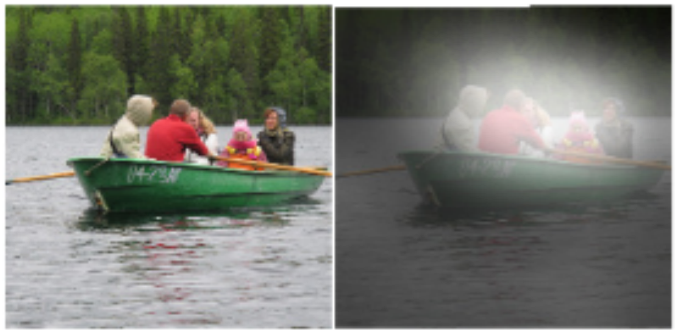
TECHNISCHE UNIVERSITÄT DARMSTADT

- White indicates the attention placed on the picture for generating the underlined word.


A woman is throwing a frisbee in a park.

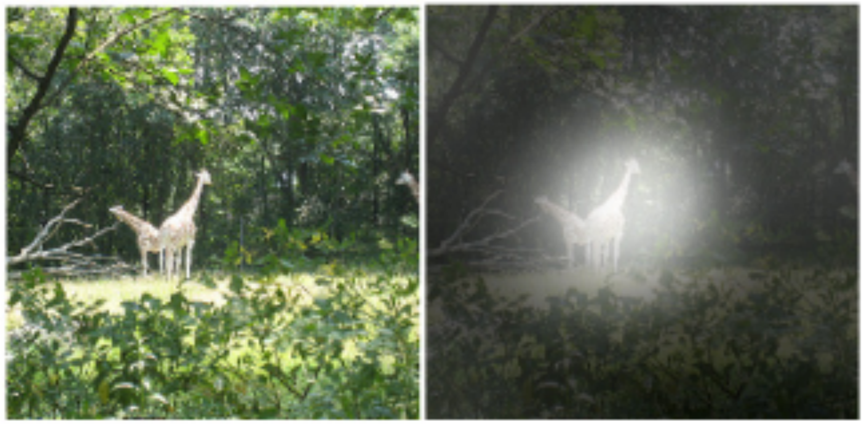

A dog is standing on a hardwood floor.

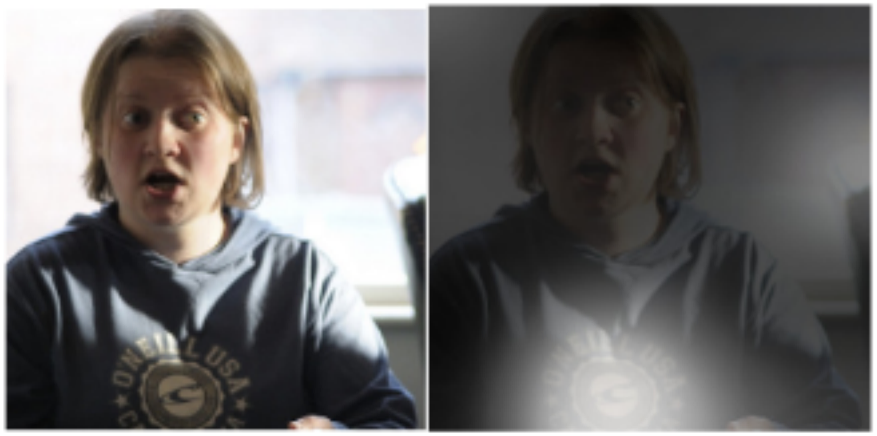

A little girl sitting on a bed with a teddy bear.


A group of people sitting on a boat in the water.

Attention values provide insights to the model

- The attention information also helps for the error analysis.


A large white bird standing in a forest.


A woman holding a clock in her hand.

Summary

- Auto-Encoder
    - Learn to “copy” the input to the output with restrictions.
    - Used to be very valuable for feature learning, currently importance seems to be decreasing.
- Encoder-Decoder
    - Comparable to the noisy channel model.
    - Use one RNN for encoding and one for decoding.
    - Sequence-to-sequence tasks especially if the input and output sequence do not have a 1-1 alignment.
    - **Attention-based approaches**
        - Learn an attention vector that indicates which part of the input should be in the focus.
        - **Soft vs. hard alignments**
- Further extensions to this framework & more applications

References (1)

- Cho, K., Gulcehre, B. V. M. C., Bahdanau, D., Schwenk, F. B. H., & Bengio, Y. (2014): Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. EMNLP.
- Bahdanau, Dzmitry, Kyunghyun Cho, Yoshua Bengio (2015): Neural Machine Translation by Jointly Learning to Align and Translate. ICLR. http://arxiv.org/abs/1409.0473
- Minh-Thang Luong, Hieu Pham, Christopher D. Manning (2015): Effective Approaches to Attention-based Neural Machine Translation, EMNLP.
- Li Dong, Mirella Lapata (2016): Language to Logical Form with Neural Attention, http://arxiv.org/abs/1601.01280
- Alexander M. Rush, Sumit Chopra, Jason Weston (2015): A Neural Attention Model for Abstractive Sentence Summarization, ACL, http://www.aclweb.org/anthology/D15-1044
- Tai, Kai Sheng, Richard Socher, and Christopher D. Manning (2015). Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 1556–1566.
- Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, Yoshua Bengio (2016): Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, http://arxiv.org/abs/1502.03044

TECHNISCHE UNIVERSITÄT DARMSTADT

References (2)

- Faruqui et al. (2016), Morphological inflection generation using character sequence to sequence learning.
- Schnober, Eger, Do Dinh, and Gurevych (2016). Still not there? Comparing traditional sequence-to-sequence models to encoder-decoder neural networks on monotone string translation tasks
- Aharoni and Goldberg (2017). Morphological inflection generation with hard monotonic attention.
- Chung, Cho, and Bengio (2016). A character-level decoder without explicit segmentation for neural machine translation.
- Sutskever et al. (2014), Sequence to sequence learning with neural networks