

Sequence Modeling: Recurrent and Recursive Networks

Markus Dumke

27th January 2016

Contents

Introduction

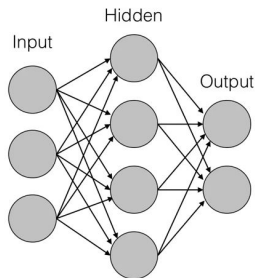
Model architecture

Optimization and Vanishing Gradient Problem

LSTM & other RNN models

Application: Machine Translation

Why RNN's?



- Independence
- Fixed Length

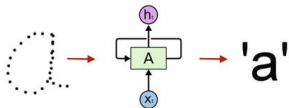
<https://www.nervanasys.com/recurrent-neural-networks/>

$$g_t(x_t, \dots, x_1) = f(x_t, h_{t-1})$$

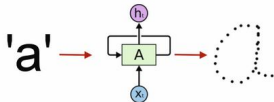
He went to Germany in 2010.
In 2010 he went to Germany.

- sequential data:
texts, speech, time series
- variable length
- long-term dependencies
- memory

Applications

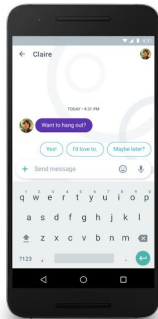


Handwriting recognition



Handwriting generation

Smart reply



<https://greydanus.github.io/2016/08/21/handwriting/>

<https://research.googleblog.com/2016/05/chat-smarter-with-allo.html>

Applications

Image Captioning



"man in black shirt is playing
guitar."

Karpathy and Fei-Fei (2015)

Pixel RNNs



Figure 1. Image completions sampled from a PixelRNN.

Van den Oord et al. (2016)

Applications

- Machine translation
- Sentiment analysis
- Text summaries
- Speech recognition and generation
- Time series
- Deep Reinforcement Learning

Contents

Introduction

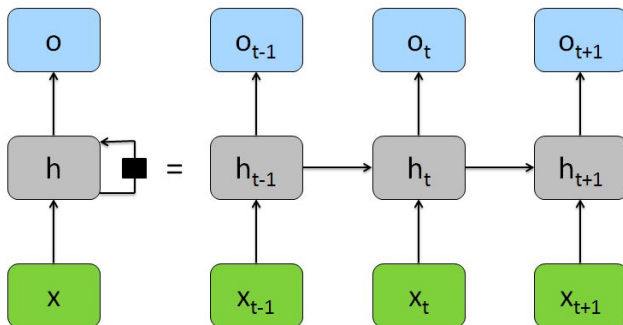
Model architecture

Optimization and Vanishing Gradient Problem

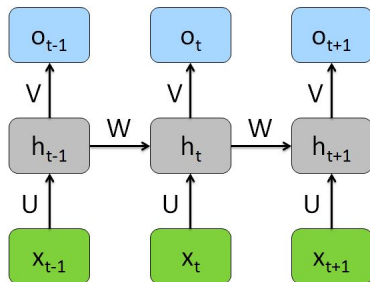
LSTM & other RNN models

Application: Machine Translation

Recurrent Neural Network



Recurrent Neural Network



for $t = 1$ to T :

$$h_t = f(b + W h_{t-1} + U x_t)$$

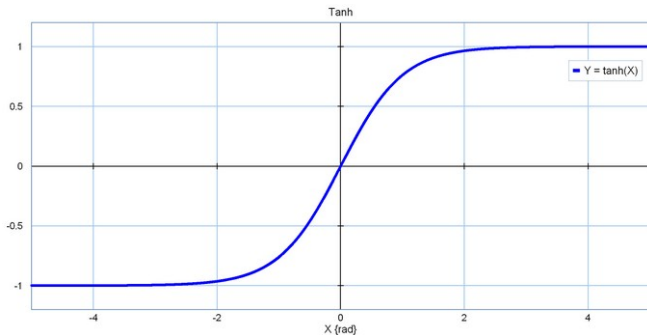
$$o_t = c + V h_t$$

$$\hat{y}_t = \text{softmax}(o_t)$$

$$= \frac{\exp(o_t^{k'})}{\sum_k \exp(o_t^k)} \quad \forall k'$$

Which activation function?

$$f(x) = \tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



http://www.20sim.com/webhelp/language_reference_functions_tanh.php

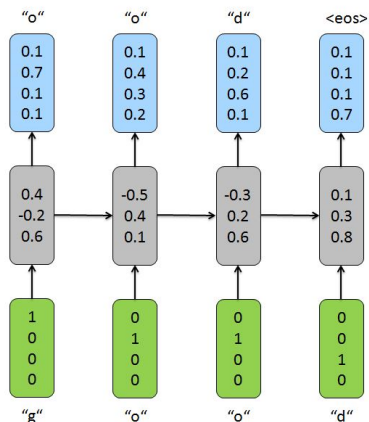
Language Modeling

- Input: word/character encoded as one-hot vector
- Output: Probability distribution over words given previous words

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

- score sentences with their probabilities

Recurrent Neural Network



$$h_t = f\left(\underset{n_h \times 1}{b} + \underset{n_h \times n_h}{W} \underset{n_h \times 1}{h_{t-1}} + \underset{n_h \times n_y}{U} \underset{n_y \times 1}{x_t} \right)$$

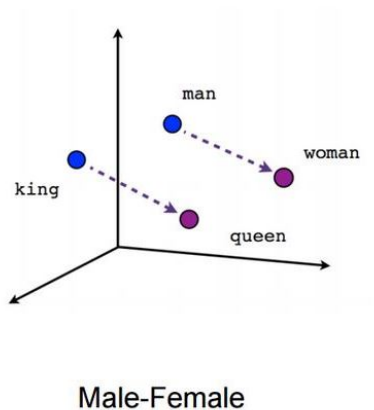
$$o_t = \underset{n_y \times 1}{c} + \underset{n_y \times 1}{V} \underset{n_y \times n_h}{h_{t-1}}$$

Vocabulary size $n_y > 100000$

Word Embeddings (Word2vec)

- Data sparsity

$$\text{man} \rightarrow \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0.35 \\ -0.83 \\ \vdots \\ 0.11 \\ 3.2 \end{bmatrix}$$



<https://www.tensorflow.org/tutorials/word2vec/>

Sampling from an RNN

- Sample from conditional distribution at each time step
- How to generate sequence length?
 - special end symbol
 - Bernoulli output
 - integer value τ

Contents

Introduction

Model architecture

Optimization and Vanishing Gradient Problem

LSTM & other RNN models

Application: Machine Translation

Optimization

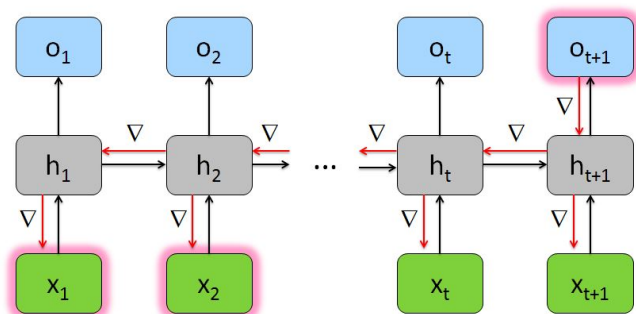
- Forward Propagation:
 - compute hidden states, outputs and loss
 - Loss function, e.g. negative log-likelihood

$$L = \sum_t L_t = \sum_t -\log p_{model}(y_t | x_1, \dots, x_t)$$

- Backward Propagation through time (BPTT):
 - compute gradients
- Stochastic Gradient Descent

Vanishing (and Exploding) Gradient Problem

$$\frac{\partial L}{\partial h_1} = \frac{\partial L}{\partial h_T} \frac{\partial h_T}{\partial h_{T-1}} \cdots \frac{\partial h_2}{\partial h_1}$$

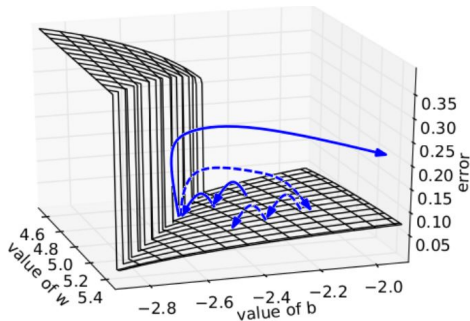


How to deal with exploding gradients?

Gradient Clipping

if $\|\nabla W\| > threshold$:

$$\nabla W \leftarrow \frac{threshold}{\|\nabla W\|} \nabla W$$



Pascanu, Mikolov and Bengio (2013)

Contents

Introduction

Model architecture

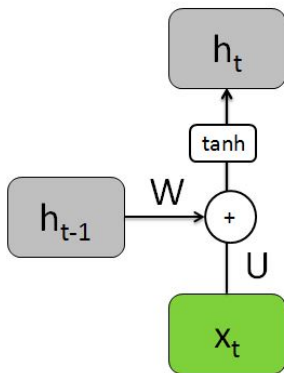
Optimization and Vanishing Gradient Problem

LSTM & other RNN models

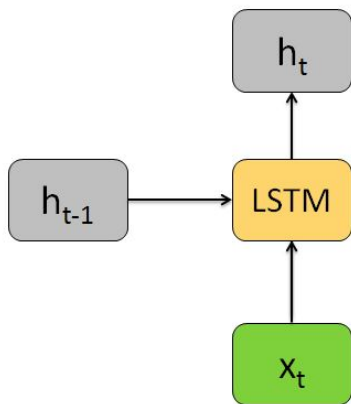
Application: Machine Translation

Vanilla RNN

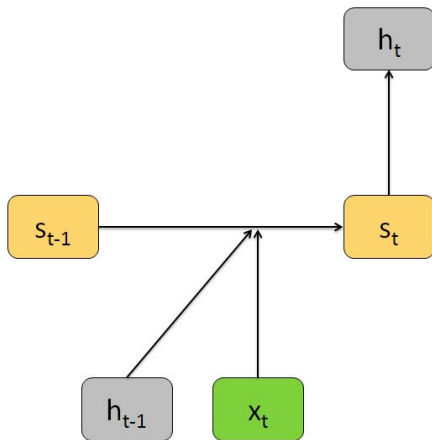
$$h_t = \tanh(b + W h_{t-1} + U x_t)$$



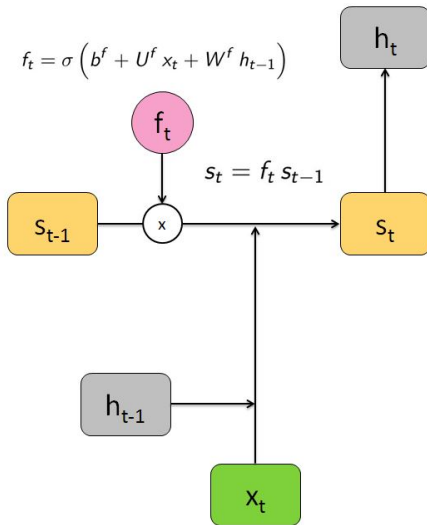
LSTM



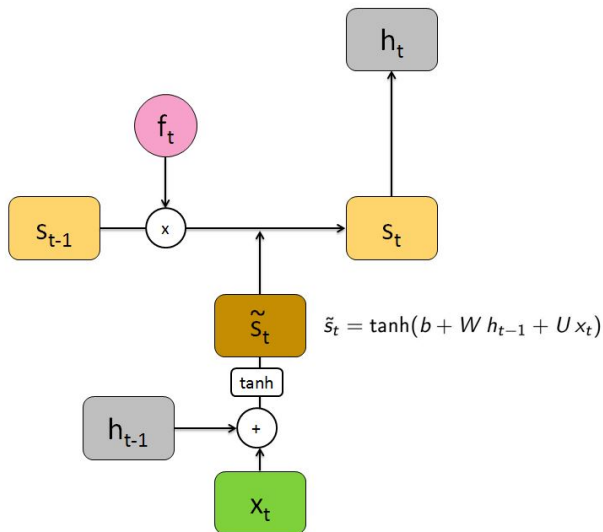
LSTM



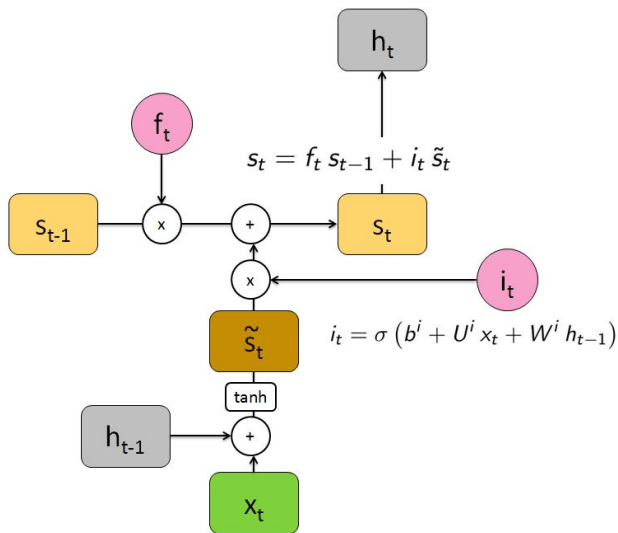
LSTM



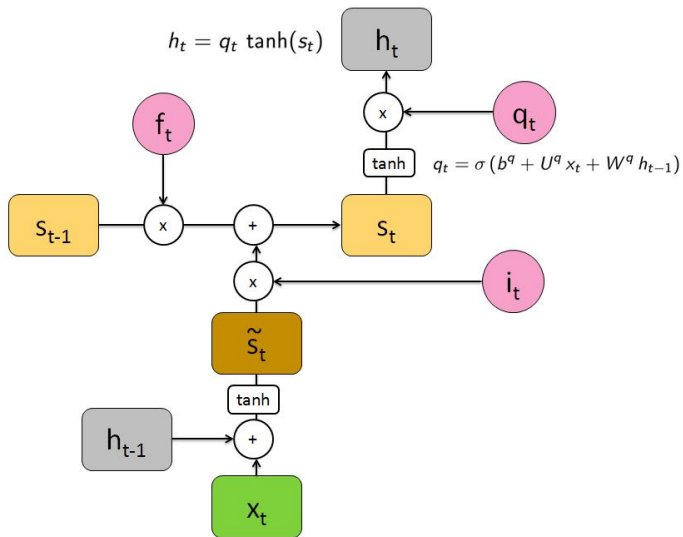
LSTM



LSTM



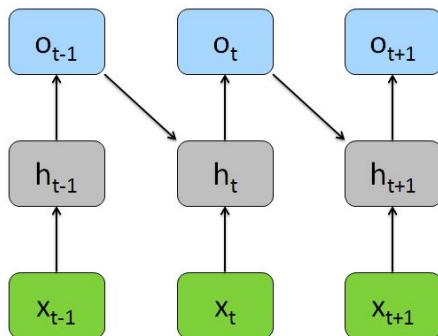
LSTM



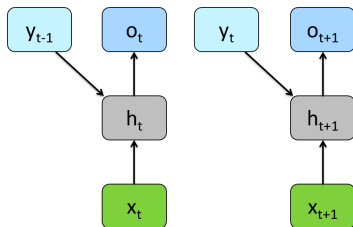
LSTM in R (mxnet)

```
1 | model <- mx.lstm(X.train, X.val,  
2 |                 ctx = mx.cpu(),  
3 |                 num.round = 100,  
4 |                 num.lstm.layer = 1,  
5 |                 seq.len = 32,  
6 |                 num.hidden = 20,  
7 |                 num.label = 100,  
8 |                 batch.size = 32,  
9 |                 initializer = mx.init.uniform(0.1),  
10 |                learning.rate = 0.1,  
11 |                dropout = 0,  
12 |                clip_gradient = 1)
```

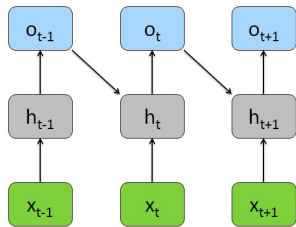
RNN with output recurrence



Teacher Forcing

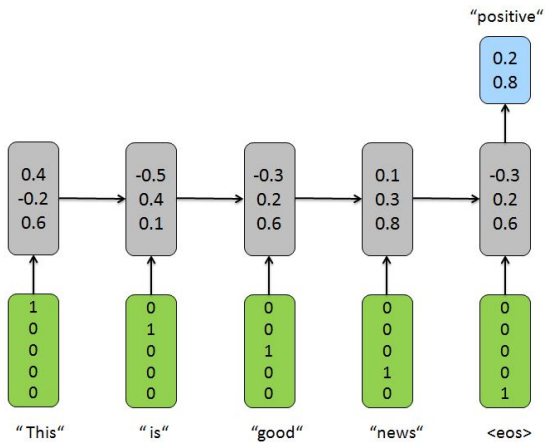


At train time

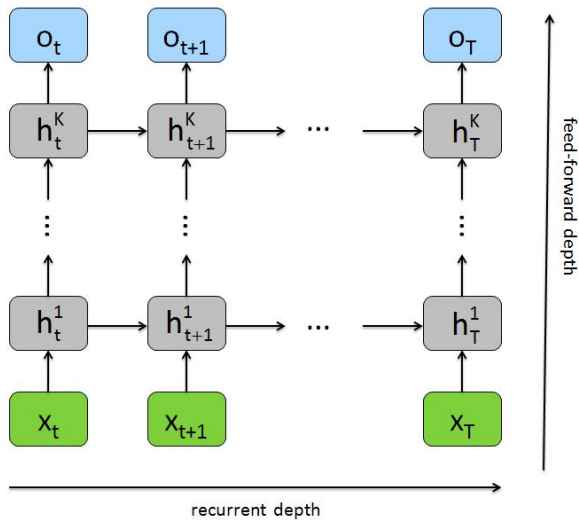


At test time

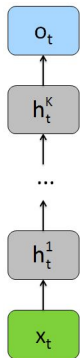
One-output RNN



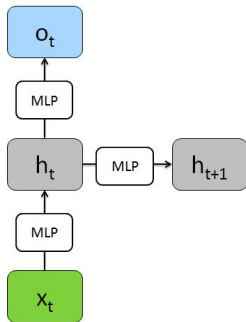
Deep RNNs



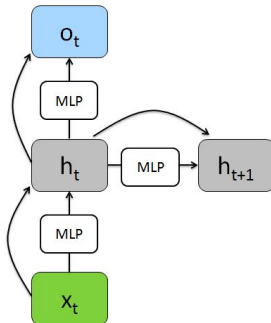
Deep RNNs



Stacked layers

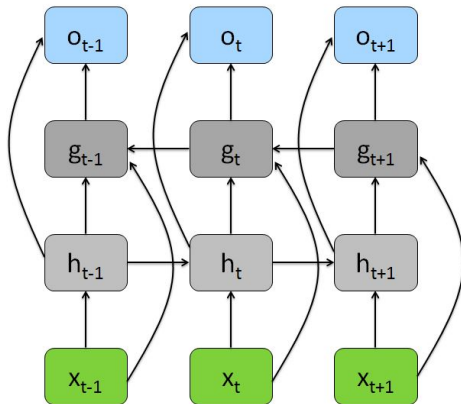


Deep computations

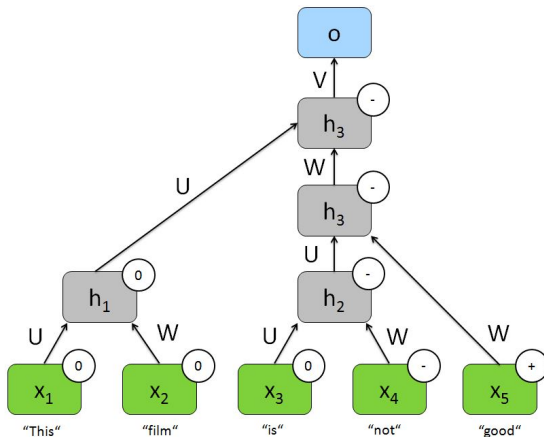


Skip connections

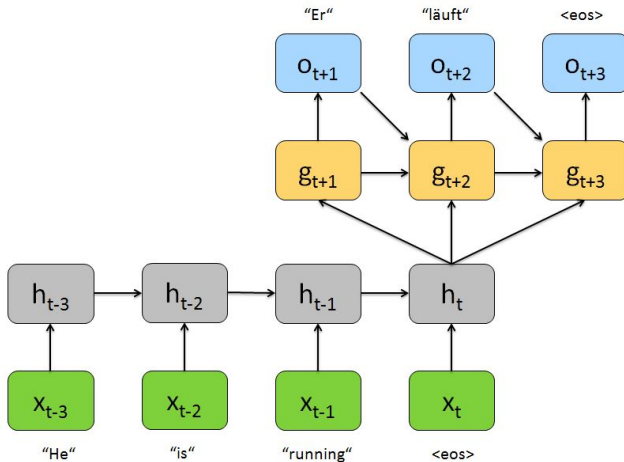
Bidirectional RNN



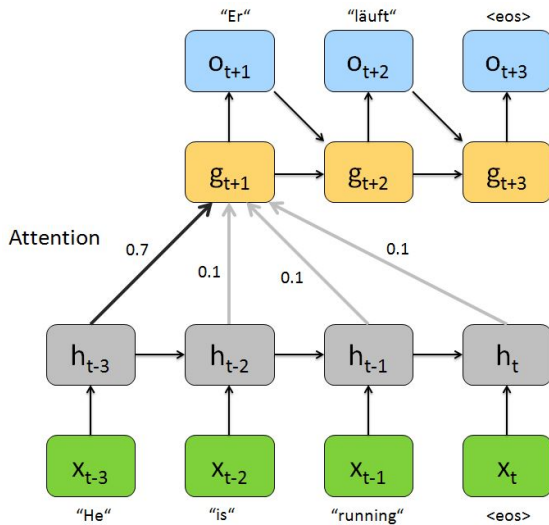
Recursive Neural Network



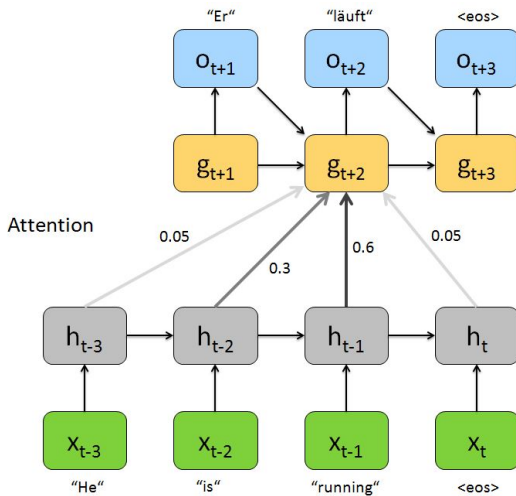
Encoder-Decoder Network



Attention



Attention



Contents

Introduction

Model architecture

Optimization and Vanishing Gradient Problem

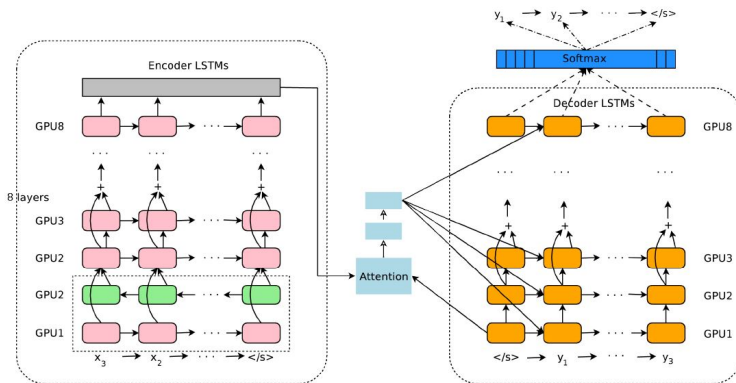
LSTM & other RNN models

Application: Machine Translation

Machine Translation

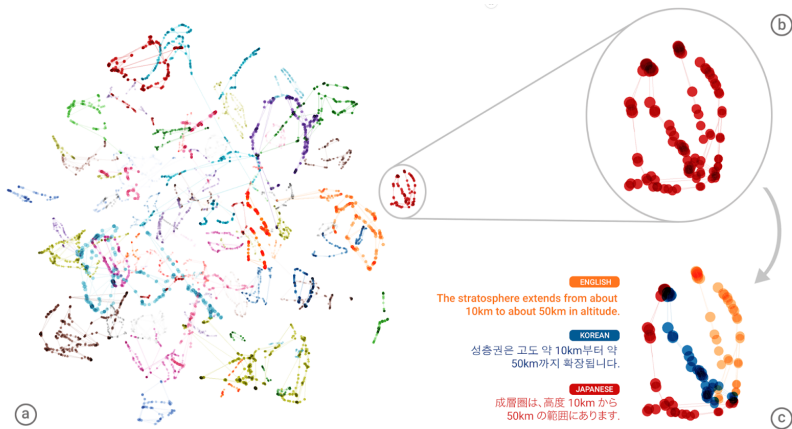
- last decades: phrase-based systems
- neural networks as part of phrase-based systems
- Encoder-decoder RNNs:
 - Sutskever et al. (2014), Bahdanau et al. (2015)
- Google's Neural Machine Translation (September/November 2016)

Google's Neural Machine Translation System



Wu et al. (2016): Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation

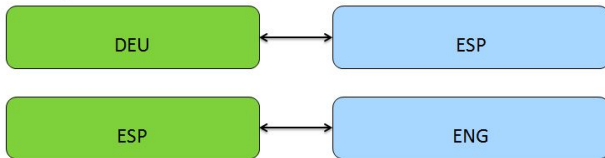
Language Embeddings



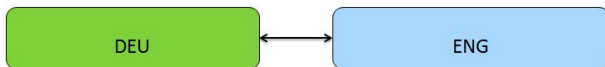
Johnson et al. (2016): Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation

Zero-shot Translation

Training:



Test:



Details Machine Translation

Main challenges:

- speed
- handling of rare words
- not translating all words (coverage)

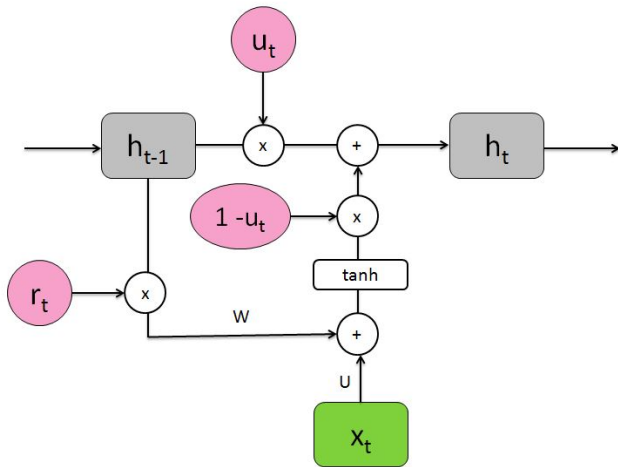
Solutions:

- GPU training
- sub-word units (wordpieces)
- coverage penalty
- length-normalization

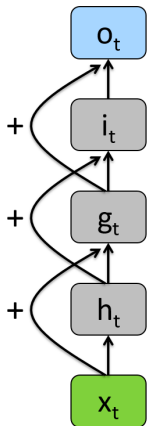
How to deal with vanishing gradients?

- Regularization $\nabla_{h_t} L \approx (\nabla_{h_t} L) \frac{\partial h_t}{\partial h_{t-1}}$
- skip-connections over time
- Leaky units $\mu = \alpha \mu_{t-1} + (1 - \alpha) \nu_t$
- remove short-term connections
- Explicit Memory
- **LSTM**, GRU and other gated RNNs

GRU



Residual Networks (Res-Nets)



- training of very deep models possible
- like an ensemble of shallow architectures