# Neural MT Basics I

David Vilar
dvilar@amazon.com

MT Marathon 2018
3. September 2018

Welcome to the MT Marathon!

# Overview

**Goal:** Introduce the encoder-decoder architecture.

**Roadmap:** What we will see in this lecture:

- Neural language models.
- Word embeddings.
- Recurrent neural networks (including LSTMs).
- Encoder-Decoder architecture.
- Comparison with previous approaches.

# Overview

**Goal:** Introduce the encoder-decoder architecture.

**Roadmap:** What we will see in this lecture:

- Neural language models.
- Word embeddings.
- Recurrent neural networks (including LSTMs).
- Encoder-Decoder architecture.
- Comparison with previous approaches.

**Follow-up:** What you will see in the next lecture:

- Attention.
- Advanced models.

# Introduction

# (Statistical) Machine Translation History

A brief (and simplified) timeline:

**1949** Shannon/Weaver: statistical approach.

**1950-1970** Empirical and statistical language analysis.

**1969** Chomsky: Ban on statistics.

*The notion "probability of a sentence" is an entirely useless one, under any known interpretation of this term.*

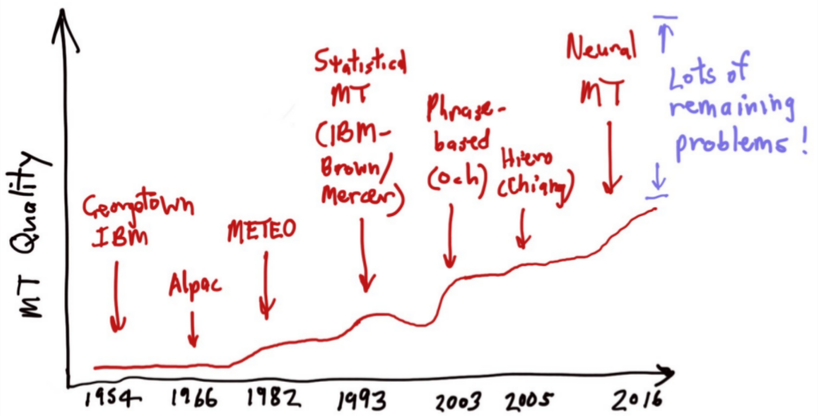
**1970-2000** Hype of artificial and rule-based approaches.

**1989-1995** IBM Reasearch: Statistical translation.

**1995-2014** Phrase-based approaches (and extensions).

**2014-??** Hype of deep learning approaches.

# Progress in MT




From ACL tutorial by Luong, Cho and Manning 2016

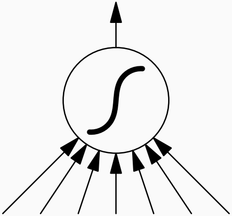
# Preliminaries

# Roadmap

- Neural networks.
- Language models.

# Single Neuron

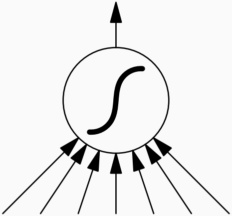
Linear transformation followed by a non linearity:



$$y = f\left(\sum_k w_k x_k + b\right)$$

# Single Neuron

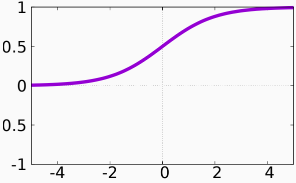
Linear transformation followed by a non linearity:



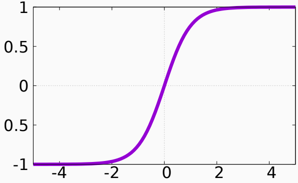
$$y = f\left(\sum_k w_k x_k + b\right)$$

For a whole layer: $\mathbf{y} = f\left(\mathbf{W}\mathbf{x} + \mathbf{b}\right)$

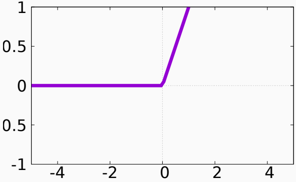
# Non-linear Functions



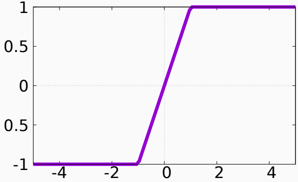

$\sigma(x) = \frac{1}{1+e^{-x}}$




$\tanh(x)$




$\max(0, x)$ (ReLU)




$\text{htanh}(x)$

# Multi-layer FF Networks

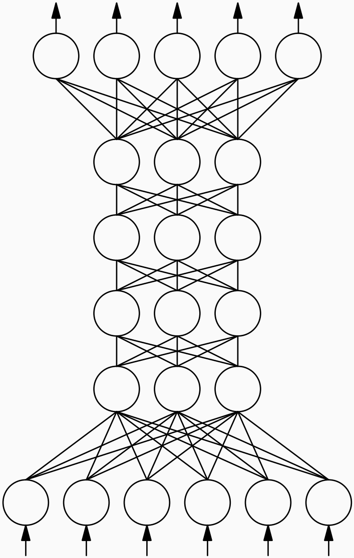
- Several layers, the output of one layer is the input of the next one:

$$\mathbf{y}^{(l)}(\mathbf{x}) = f(\mathbf{W}^{(l)}\mathbf{y}^{(l-1)}(\mathbf{x}) + \mathbf{b}^{(l)})$$

- Output layer usually is a softmax operation:

$$p(Y = i|\mathbf{x}) \equiv \frac{e^{x_i}}{\sum_j e^{x_j}}$$

- Training: error backpropagation.
  - “Smart use of chain rule”



# Language Models

A language model is a probability distribution over sentences:

$$p(w_1 w_2 \ldots w_N) = p(w_1^N)$$

# Language Models

A language model is a probability distribution over sentences:

$$p(w_1 w_2 \dots w_N) = p(w_1^N)$$

It can be decomposed according to the chain rule:

$$p(w_1^N) = \prod_{n=1}^{N} p(w_n | w_1^{n-1})$$

**Note:** mathematical equality.

# $n$-gram Language Models

Until not so long ago...

- $k$-th order Markov *assumption*: $(k+1)$-grams:

$$p(w_1^N) = \prod_{n=1}^{N} p(w_n | w_1^{n-1}) \approx \prod_{n=1}^{N} p(w_n | w_{n-k}^{n-1})$$

- "Big tables" of probabilities.
- ML estimation: relative frequencies.
  - Smoothing for unseen events.

[Kneser and Ney, 1995, Chen and Goodman, 1996]

# Motivation: ASR

Example from the Wall Street Journal 5K task:

| LM | Recognized | errors |
|---|---|---|
| 0-gram | h ih t s eh n uh t ur z n ih g oh sh ee ey t ih ng – – s ey l – – s ur t un aa s eh t s aw n t uh b r oh k ur ih j y ooh n ih t s | 11 |
| | HIT SENATORS NEGOTIATING SALE CERTAIN ASSETS ONTO BROKERAGE UNIT'S | 9 |

# Motivation: ASR

Example from the Wall Street Journal 5K task:

| LM | Recognized | errors |
|---|---|---|
| 1-gram | ih t s s eh n ih t ih z n ih g oh sh ee ey t ih ng – – s ey l – – s ur t un aa s eh t s aw v dh uh b r oh k ur ih j y ooh n ih t | 6 |
| | ITS SENATE IS NEGOTIATING SALE CERTAIN ASSETS OF THE BROKERAGE UNIT | 5 |

# Motivation: ASR

Example from the Wall Street Journal 5K task:

| LM | Recognized | errors |
| --- | --- | --- |
| 2-gram | ih t s eh d ih t ih z n ih g oh sh ee ey t ih ng dh uh s ey l aw v s ur t un aa s eh t s aw v dh uh b r oh k ur ih j y ooh n ih t | 0 |
| | IT SAID IT IS NEGOTIATING THE SALE OF CERTAIN ASSETS OF THE BROKER-AGE UNIT | 0 |

# Motivation: MT

Word choice:

I withdrew money from the bank.

$\Downarrow$

Saqué dinero del *banco*.

[As opposed to “orilla” ($\rightarrow$ riverbank)]

Word order:

> Die italienische Regierung will nicht mehr allein für die Flüchtlinge auf den Schiffen der EU-Mission Sophia verantwortlich sein.

$$\Downarrow$$

> The Italian government wants not any more alone for the refugees aboard the ship of the EU mission Sophia responsible be.

# Motivation: MT

Word order:

Die italienische Regierung will nicht mehr allein für die Flüchtlinge auf den Schiffen der EU-Mission Sophia verantwortlich sein.

$\Downarrow$

The Italian government wants not any more alone for the refugees aboard the ship of the EU mission Sophia responsible be.

$\Downarrow$

The Italian government does not want to be responsible any more for the refugees aboard the ship of the EU mission Sophia.

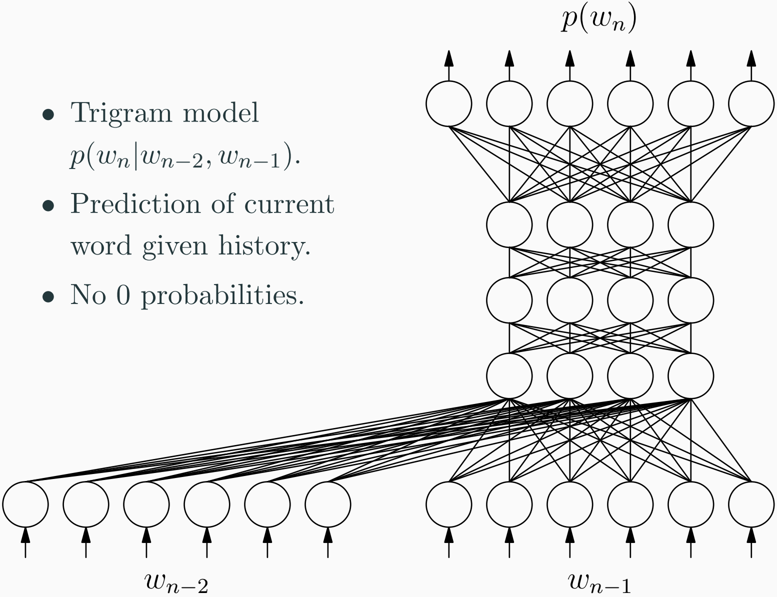
# Neural Language Models

# Roadmap

- $n$-gram *neural* language models.
- How to represent words.
- Dropping the Markov assumption (vanilla RNNs).
- LSTMs.

# Feed-forward LM

- Trigram model $p(w_n|w_{n-2}, w_{n-1})$.
- Prediction of current word given history.
- No 0 probabilities.



# 1-hot encodings

- 1-hot encoding is the "natural" way to encode symbolic information (e.g. words).
- But:
  - The encoding itself is arbitrary (e.g. first appearance of a word in the training text).
  - No useful information can be read from the vector representation.
  - Example:

$$\texttt{the green dog bites the cat}$$

| | |
|---|---|
| *the* | $(1, 0, 0, 0, 0)$ |
| *green* | $(0, 1, 0, 0, 0)$ |
| *dog* | $(0, 0, 1, 0, 0)$ |
| *bites* | $(0, 0, 0, 1, 0)$ |
| *cat* | $(0, 0, 0, 0, 1)$ |

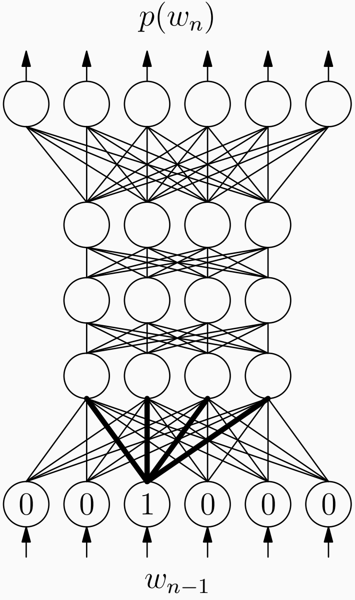
# Word Embeddings

What happens in the first layer of the network?

- Usually simplified form

$$\mathbf{y}^{(1)}(\mathbf{x}) = \mathbf{W}^{(1)}\mathbf{x}$$

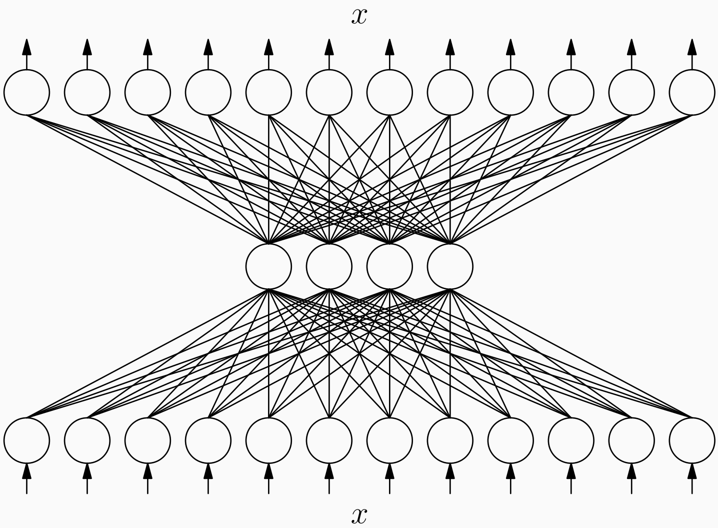
  where $\mathbf{x}$ is a 1-hot vector.
- Multiplication reduces to column lookup.
- Maps words into continuous vectors.



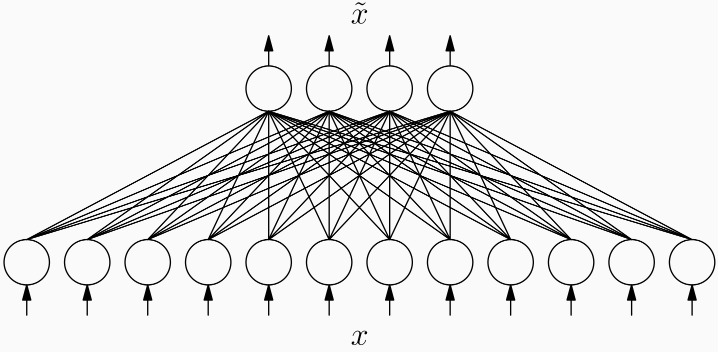
How do these word vectors look like?

- Word embedding: mapping of words (discrete) into a continuous space.
- Arises naturally when dealing with 1-hot encodings.
- Can be trained separately.
  - Active area of research.
  - Big improvements on some tasks.

# Excursion: The most "stupid" network



# Excursion: The most “stupid” network



# Excursion: The most “stupid” network

If the “stupid” network has no errors:

- We mapped an 12-dimensional (sparse?) vector into a 4-dimensional dense vector.

# Excursion: The most “stupid” network

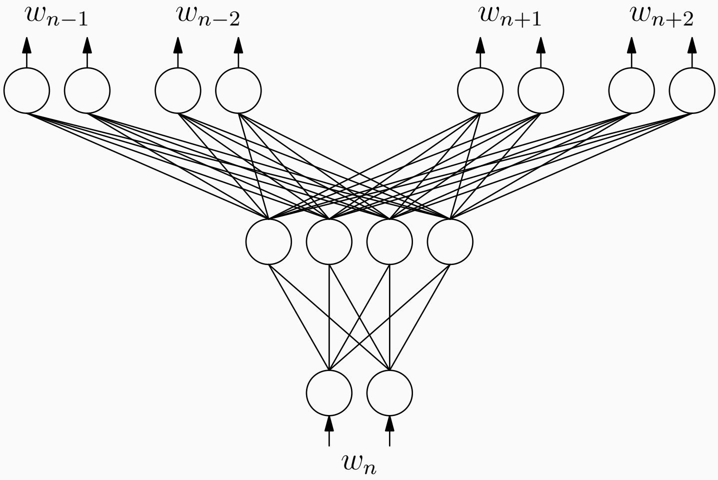
If the “stupid” network has no errors:

- We mapped an 12-dimensional (sparse?) vector into a 4-dimensional dense vector.

However:

- The representation is still arbitrary, as no information about the word themselves is taken into account.

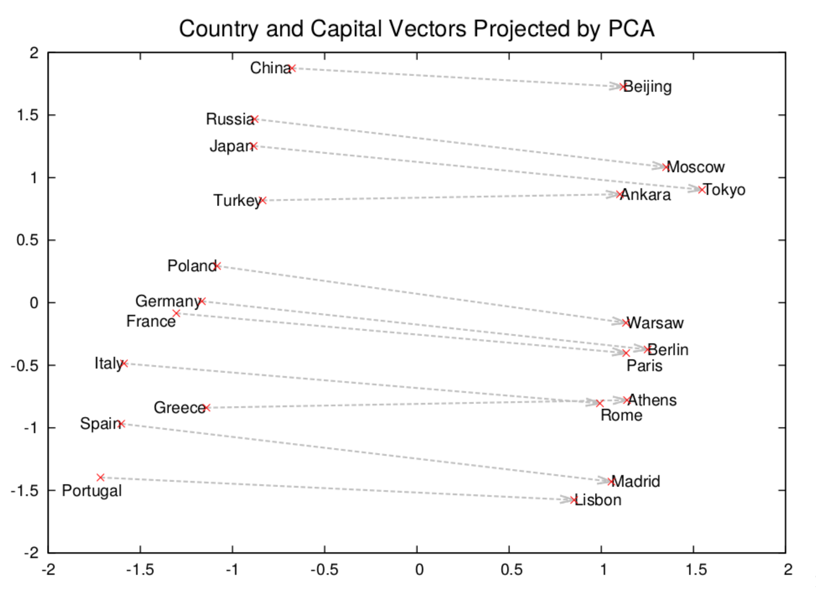
# Excursion: Skip-gram model




[Mikolov et al., 2013]

- Assumption: similar words appear in similar contexts.
- Goal: similar words have similar representations (as they will predict similar contexts).
- Indeed:
  - $\text{vec}(King) - \text{vec}(Man) + \text{vec}(Woman)$ results in a vector that is closest to $Queen$.
  - $\text{vec}(Madrid) - \text{vec}(Spain) + \text{vec}(France)$ results in a vector that is closest to $Paris$.

# Excursion: Skip-gram model



Different implementations available

- One of the most well known: `word2vec` by Mikolov et al.

For machine translation:

- Embeddings trained at the same time as the full system.
- Pre-trained embeddings may be used for initialization.
    - Useful for other taks, e.g. NLU.
    - No gains reported for MT.

https://code.google.com/archive/p/word2vec/

# Recap

- Language model:

$$p(w_1^N)$$

- Chain rule: (mathematical equality)

$$p(w_1^N) = \prod_{n=1}^{N} p(w_n | w_1^{n-1})$$

- $k$-th order Markov *assumption*: $(k+1)$-grams

$$p(w_1^N) \approx \prod_{n=1}^{N} p(w_n | w_{n-k}^{n-1})$$

# Recap

Advantage of NNLMs we encountered up to this point:

- FF language models deal with the sparsity problem (by projecting into a continuous space).

# Recap

Advantage of NNLMs we encountered up to this point:

- FF language models deal with the sparsity problem (by projecting into a continuous space).

  . . . but they still are under the Markov chain assumption.

# Recap

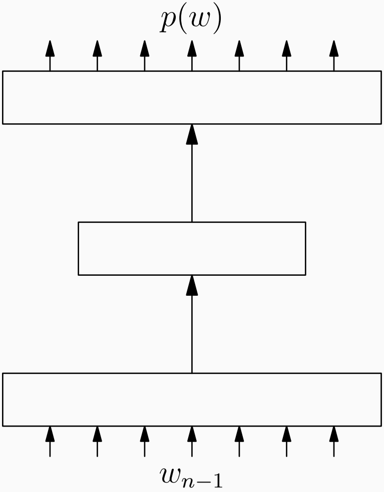
Advantage of NNLMs we encountered up to this point:

- FF language models deal with the sparsity problem (by projecting into a continuous space).

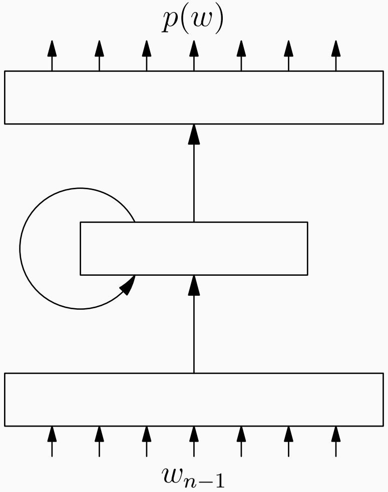
  ... but they still are under the Markov chain assumption.

We would like to be able to take into account the *whole* history.
$\rightarrow$ Let the network remember everything it has seen!

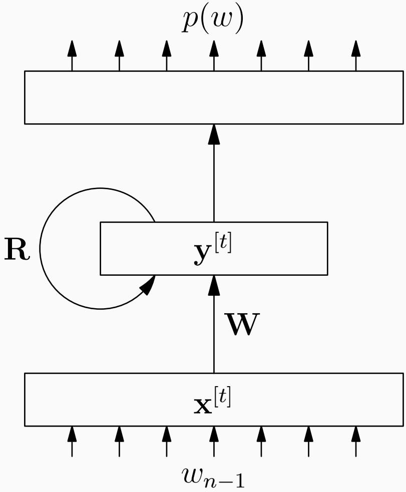
# Recurrent NNs



# Recurrent NNs



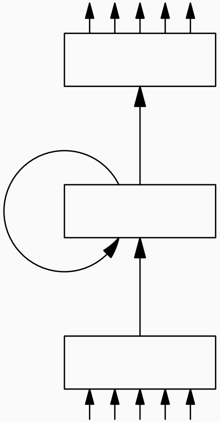
# Recurrent NNs




In Equations: $\mathbf{y}^{[t]} = f(\mathbf{W}\mathbf{x}^{[t]} + \mathbf{R}\mathbf{y}^{[t-1]} + \mathbf{b})$

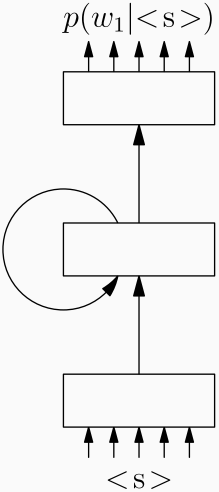
# Recurrent NNs

$p(w_1^4) =$



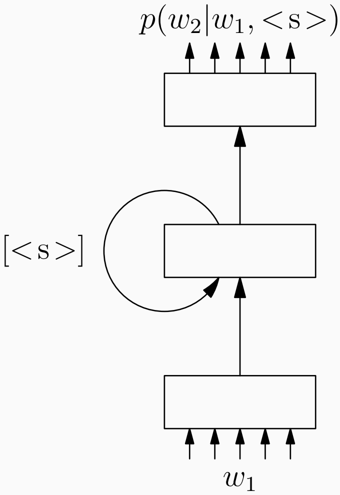
# Recurrent NNs

$p(w_1^4) =$
$p(w_1|< \mathrm{s} >)$



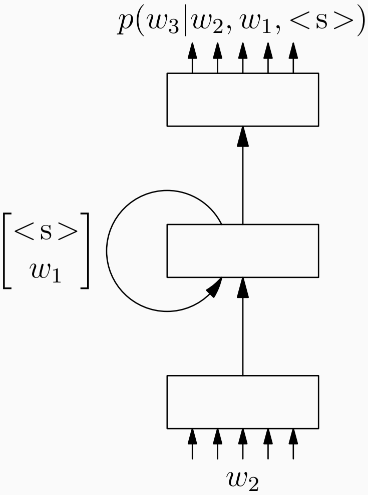
# Recurrent NNs

$p(w_1^4) =$
$p(w_1|< \mathrm{s} >)$
$\times p(w_2|w_1, < \mathrm{s} >)$



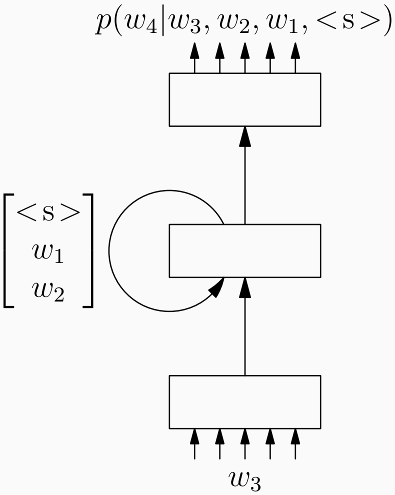
# Recurrent NNs

$p(w_1^4) =$
$p(w_1|< \text{s} >)$
$\times p(w_2|w_1, < \text{s} >)$
$\times p(w_3|w_2, w_1, < \text{s} >)$



# Recurrent NNs

$$
\begin{aligned}
&p(w_1^4) = \\
&p(w_1 | < \mathrm{s} >) \\
&\times p(w_2 | w_1, < \mathrm{s} >) \\
&\times p(w_3 | w_2, w_1, < \mathrm{s} >) \\
&\times p(w_4 | w_3, w_2, w_1, < \mathrm{s} >)
\end{aligned}
$$

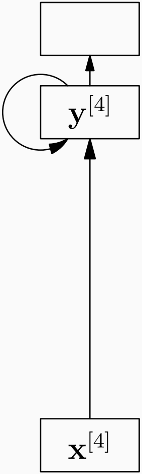


# Backpropagation through time

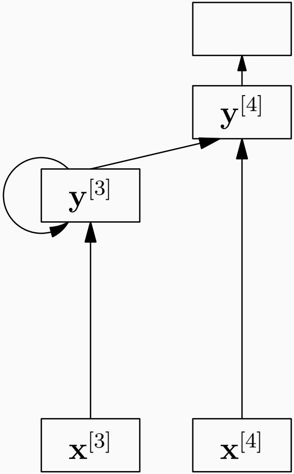
How to train a RNN?

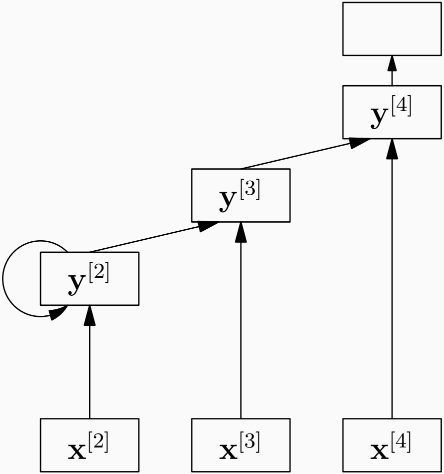
- Use backpropagation.
- Unfold recurrent connections through time.
- Results in a wide network, backpropagation can be used.
- Use chain rule not only for layers, but also for time steps.

# Backpropagation through time

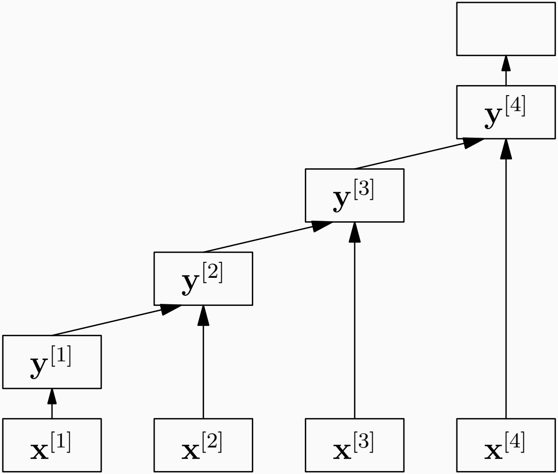


# Backpropagation through time



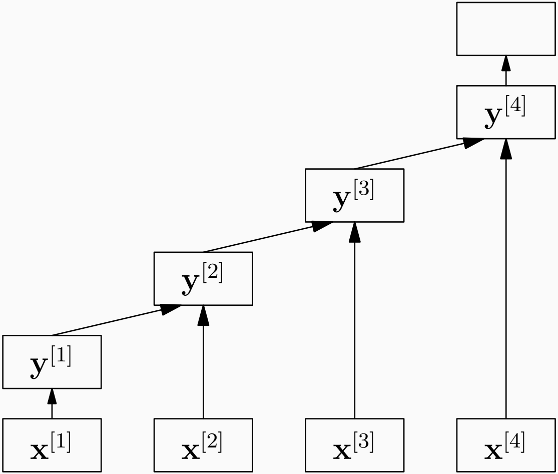

$\mathbf{y}^{[4]}$
$\mathbf{y}^{[3]}$
$\mathbf{y}^{[2]}$
$\mathbf{x}^{[2]}$
$\mathbf{x}^{[3]}$
$\mathbf{x}^{[4]}$

# Backpropagation through time



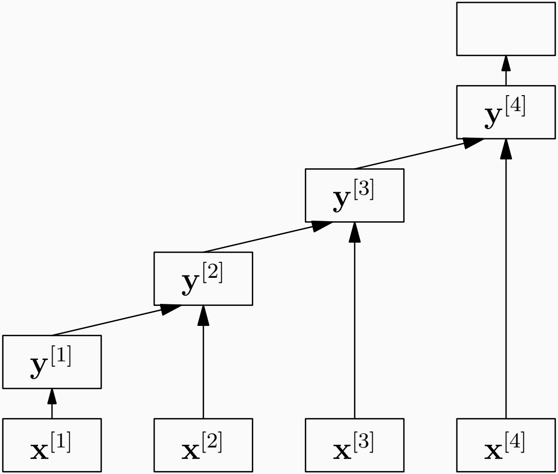
# Backpropagation through time

$$\frac{\partial \mathcal{L}}{\partial \theta} =$$



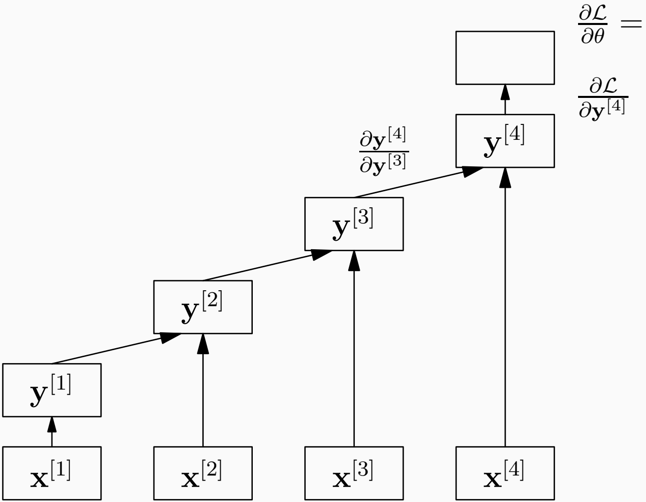
# Backpropagation through time

$$\frac{\partial \mathcal{L}}{\partial \theta} =$$

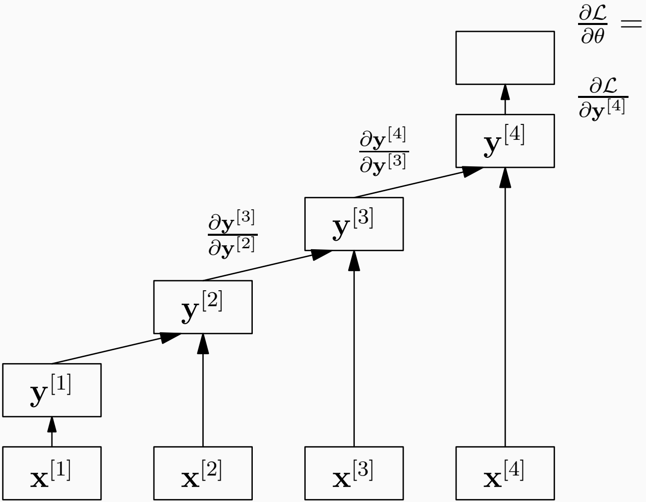



$$\frac{\partial \mathcal{L}}{\partial \mathbf{y}^{[4]}}$$

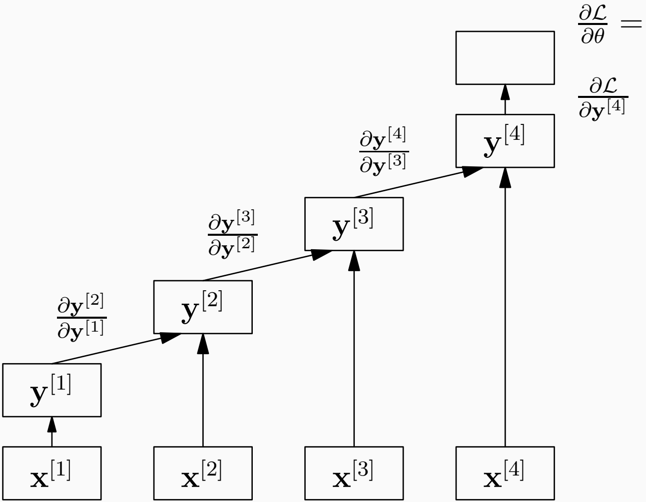
# Backpropagation through time



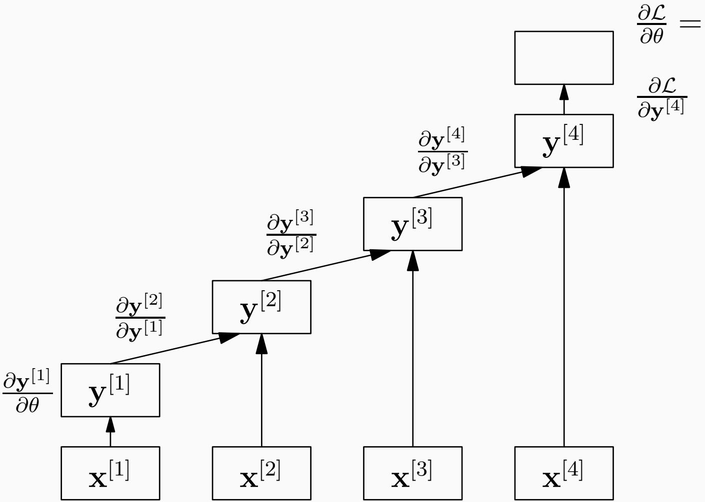
# Backpropagation through time



# Backpropagation through time



# Backpropagation through time



# Exploding and vanishing gradient

Observation: sometimes the gradient "misbehaves".

# Exploding and vanishing gradient

Observation: sometimes the gradient "misbehaves".

- Sometimes *vanishes* (norm $\approx 0$).

# Exploding and vanishing gradient

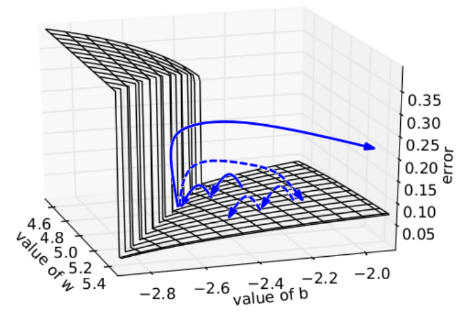
Observation: sometimes the gradient "misbehaves".

- Sometimes *vanishes* (norm $\approx 0$).
- Sometimes *explodes* (norm $\to \infty$).

# Exploding and vanishing gradient

Observation: sometimes the gradient "misbehaves".

- Sometimes *vanishes* (norm $\approx 0$).
- Sometimes *explodes* (norm $\to \infty$).



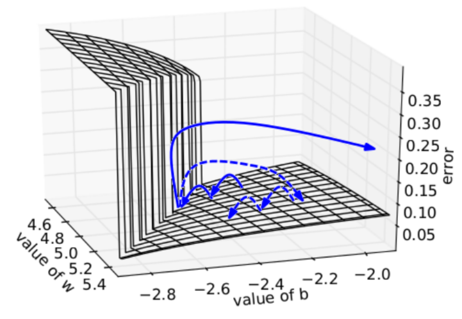
# Exploding and vanishing gradient

What to do?

- Exploding gradient: clip the gradient (divide by the norm). [Full vector or element-wise]

What to do?

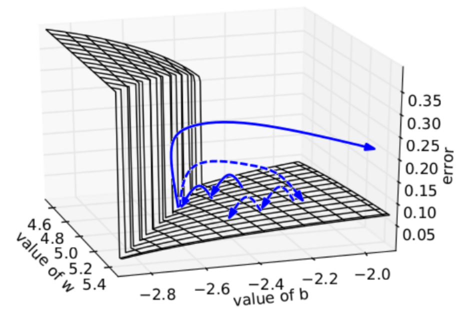
- Exploding gradient: clip the gradient (divide by the norm). [Full vector or element-wise]



# Exploding and vanishing gradient

What to do?

- Exploding gradient: clip the gradient (divide by the norm). [Full vector or element-wise]




- Vanishing gradient: No easy solution.

# Exploding and vanishing gradient

Why does this happen?

Sequence of length $T$, $\mathbf{y}^{[t]} = f(\mathbf{W}\mathbf{x}^{[t]} + \mathbf{R}\mathbf{y}^{[t-1]} + \mathbf{b})$.

# Exploding and vanishing gradient

Why does this happen?

Sequence of length $T$, $\mathbf{y}^{[t]} = f(\mathbf{W}\mathbf{x}^{[t]} + \mathbf{R}\mathbf{y}^{[t-1]} + \mathbf{b})$.

Derivative of the loss function $\mathcal{L}$:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \sum_{1 \leq t_2 \leq T} \frac{\partial \mathcal{L}^{[t_2]}}{\partial \theta} = \sum_{1 \leq t_2 \leq T} \sum_{1 \leq t_1 \leq t_2} \frac{\partial \mathcal{L}^{[t_2]}}{\partial \mathbf{y}^{[t_2]}} \frac{\partial \mathbf{y}^{[t_2]}}{\partial \mathbf{y}^{[t_1]}} \frac{\partial \mathbf{y}^{[t_1]}}{\partial \theta}$$

$$\frac{\partial \mathbf{y}^{[t_2]}}{\partial \mathbf{y}^{[t_1]}} = \prod_{t_1 < t \leq t_2} \frac{\partial \mathbf{y}^{[t]}}{\partial \mathbf{y}^{[t-1]}}$$

# Exploding and vanishing gradient

$$\frac{\partial \mathbf{y}^{[t_2]}}{\partial \mathbf{y}^{[t_1]}} = \prod_{t_1 < t \leq t_2} \frac{\partial \mathbf{y}^{[t]}}{\partial \mathbf{y}^{[t-1]}}$$

# Exploding and vanishing gradient

$$\frac{\partial \mathbf{y}^{[t_2]}}{\partial \mathbf{y}^{[t_1]}} = \prod_{t_1 < t \leq t_2} \frac{\partial \mathbf{y}^{[t]}}{\partial \mathbf{y}^{[t-1]}}$$

It can be shown:

$$\left\| \frac{\partial \mathbf{y}^{[t]}}{\partial \mathbf{y}^{[t-1]}} \right\| \leq \|\mathbf{R}^T\| \left\| \operatorname{diag}\left( f'(\mathbf{R}\mathbf{y}^{[t-1]}) \right) \right\| \leq \gamma \sigma_{\max}$$

with

- $\gamma$ a maximal bound for $f'(\mathbf{R}\mathbf{y}^{[t-1]})$.
    - e.g. $|\tanh'(x)| \leq 1$; $|\sigma'(x)| \leq \frac{1}{4}$.
- $\sigma_{\max}$ the largest singluar value of $\mathbf{R}^T$.

[Pascanu et al., 2013] and previous work

# LSTMs: Intuition

- RNNs blindly pass information from one state to the other.
- LSTMs include mechanisms for

# LSTMs: Intuition

- RNNs blindly pass information from one state to the other.
- LSTMs include mechanisms for
  - Ignoring the input.

# LSTMs: Intuition

- RNNs blindly pass information from one state to the other.
- LSTMs include mechanisms for
  - Ignoring the input.
  - Suppressing the “current” output.

# LSTMs: Intuition

- RNNs blindly pass information from one state to the other.
- LSTMs include mechanisms for
  - Ignoring the input.
  - Suppressing the "current" output.
  - Forgetting the history.

# RNN units

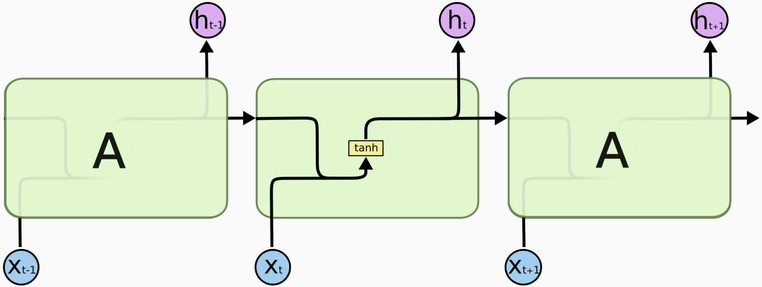



Diagram: `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`

# LSTM units

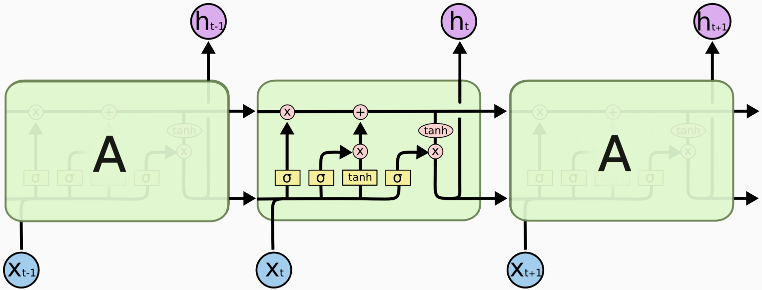



Diagram: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

# LSTM Equations

Compute a "candidate value", similar to RNNs:

$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{y}_{t-1} + \mathbf{b}_c)$$

[Hochreiter and Schmidhuber, 1997]

# LSTM Equations

Compute a "candidate value", similar to RNNs:

$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{y}_{t-1} + \mathbf{b}_c)$$

Input gate: control the influence of the current input.

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{y}_{t-1} + \mathbf{b}_i)$$

[Hochreiter and Schmidhuber, 1997]

# LSTM Equations

Compute a "candidate value", similar to RNNs:

$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{y}_{t-1} + \mathbf{b}_c)$$

Input gate: control the influence of the current input.

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{y}_{t-1} + \mathbf{b}_i)$$

Forget gate: control the influence of the history.

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{y}_{t-1} + \mathbf{b}_f)$$

[Hochreiter and Schmidhuber, 1997]

# LSTM Equations

Memory cell state: combination of new and old state.

$$\mathbf{C}_t = \mathbf{i}_t \odot \tilde{\mathbf{C}}_t + \mathbf{f}_t \odot \mathbf{C}_{t-1}$$

[Hochreiter and Schmidhuber, 1997]

# LSTM Equations

Memory cell state: combination of new and old state.

$$\mathbf{C}_t = \mathbf{i}_t \odot \tilde{\mathbf{C}}_t + \mathbf{f}_t \odot \mathbf{C}_{t-1}$$

Output gate: how much we want to output to the exterior.

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{y}_{t-1} + \mathbf{b}_o)$$

[Hochreiter and Schmidhuber, 1997]

# LSTM Equations

Memory cell state: combination of new and old state.

$$\mathbf{C}_t = \mathbf{i}_t \odot \tilde{\mathbf{C}}_t + \mathbf{f}_t \odot \mathbf{C}_{t-1}$$

Output gate: how much we want to output to the exterior.

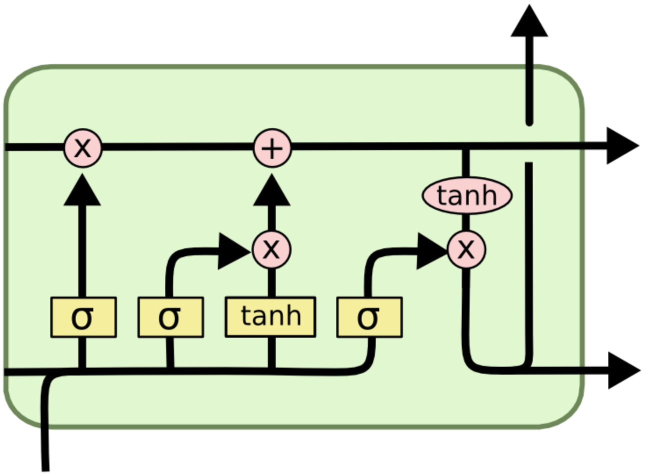
$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{y}_{t-1} + \mathbf{b}_o)$$

Output of the cell:

$$\mathbf{y}_t = \mathbf{o}_t \odot \tanh(\mathbf{C}_t)$$

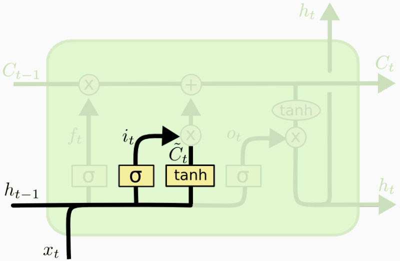
[Hochreiter and Schmidhuber, 1997]

# LSTM Visualization



Compute a "candidate value", similar to RNNs

Input gate: control the influence of the current output

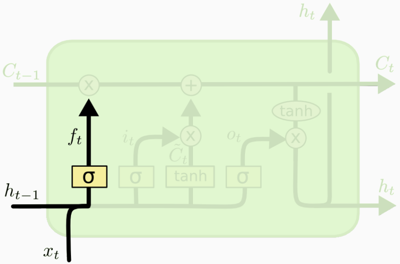



$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{y}_{t-1} + \mathbf{b}_c)$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_i x_t + \mathbf{U}_i \mathbf{y}_{t-1} + \mathbf{b}_i)$$

# LSTM Visualization

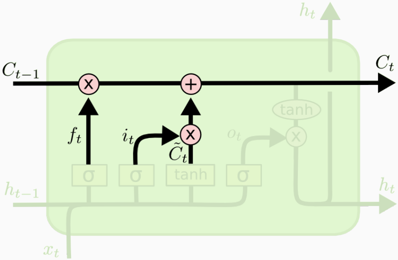
Forget gate: control the influence of the history




$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{y}_{t-1} + \mathbf{b}_f)$$

# LSTM Visualization

Memory cell state: combination of new and old state

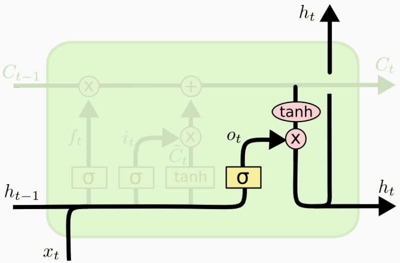



$$\mathbf{C}_t = \mathbf{i}_t \odot \tilde{\mathbf{C}}_t + \mathbf{f}_t \odot \mathbf{C}_{t-1}$$

# LSTM Visualization

Output gate: how much we want to output to the exterior

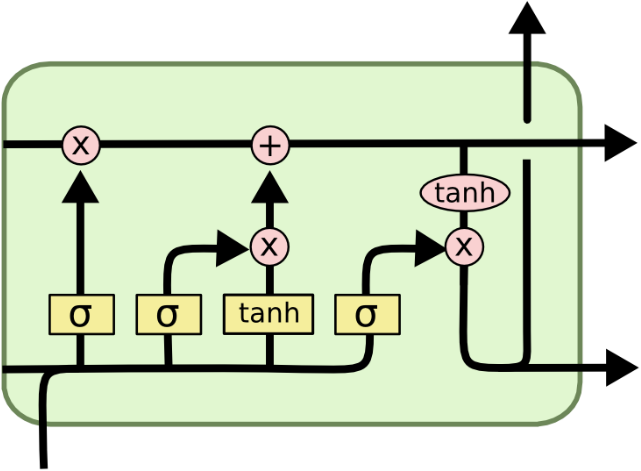
Output of the cell




$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{y}_{t-1} + \mathbf{b}_o)$$

$$\mathbf{y}_t = \mathbf{o}_t \odot \tanh(\mathbf{C}_t)$$

# LSTM Visualization



# LSTMs: additional remarks

- LSTMs solve the vanishing gradient problem, but the gradient can still explode.
    - Use gradient clipping.

# LSTMs: additional remarks

- LSTMs solve the vanishing gradient problem, but the gradient can still explode.
    - Use gradient clipping.
- Different variants of LSTMs. Basic idea is similar, but
    - Different gates.
    - Different parametrization of the gates.
    - Pay attention when reading the literature.

# GRUs

Gated Recurrent Units:

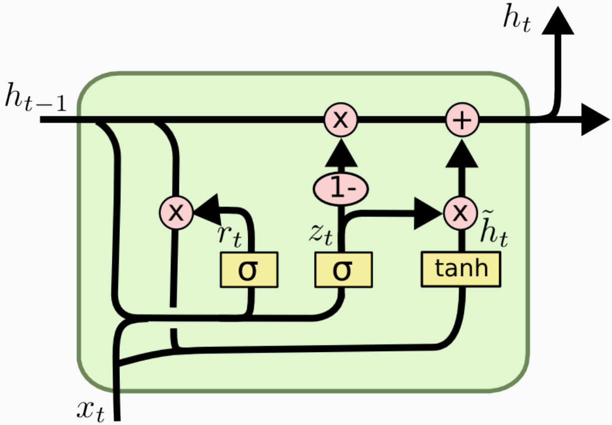
- Combine forget and input gates into an "update gate".
- Suppress output gate.
- Add a "reset gate".

Simpler than LSTMs (less parameters) and similar performance.

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1} + \mathbf{b}_z)$$
$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1} + \mathbf{b}_r)$$
$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W} \mathbf{x}_t + \mathbf{U}(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b})$$
$$\mathbf{h}_t = \mathbf{z}_t \odot \tilde{\mathbf{h}}_t + (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1}$$

[Cho et al., 2014b]

# GRUs Visualization



# Neural Machine Translation

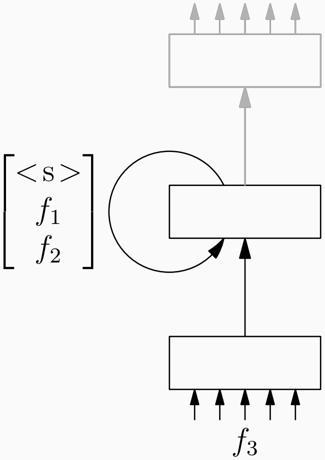
The fundamental equation for machine translation

$$\hat{e}_1^I = \operatorname*{argmax}_{e_1^I} \left\{ p(e_1^I | \boldsymbol{f_1^J}) \right\}$$

is basically a language model **expanded with source information**.

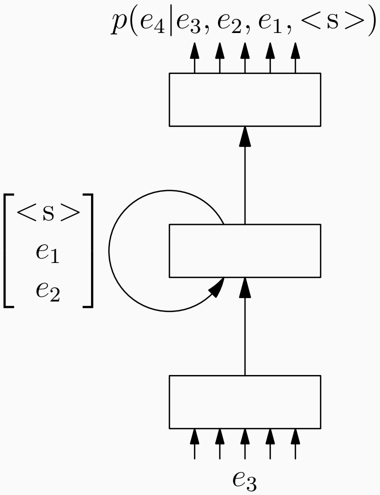
# Encoder

RNNs give us a way to represent the input.

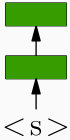


# Decoder

RNNs give us a way to generate the output.



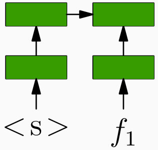
# Encoder-Decoder Architecture




The **encoder** creates a representation of the input sentence.

[Sutskever et al., 2014, Cho et al., 2014b]

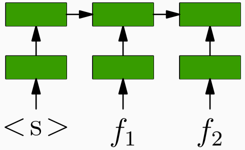
# Encoder-Decoder Architecture




The **encoder** creates a representation of the input sentence.

[Sutskever et al., 2014, Cho et al., 2014b]

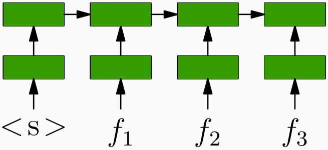
# Encoder-Decoder Architecture




The **encoder** creates a representation of the input sentence.

[Sutskever et al., 2014, Cho et al., 2014b]

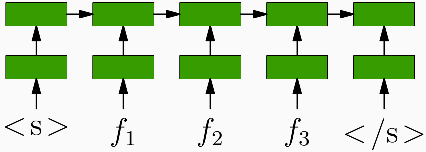
# Encoder-Decoder Architecture




The **encoder** creates a representation of the input sentence.

[Sutskever et al., 2014, Cho et al., 2014b]

# Encoder-Decoder Architecture

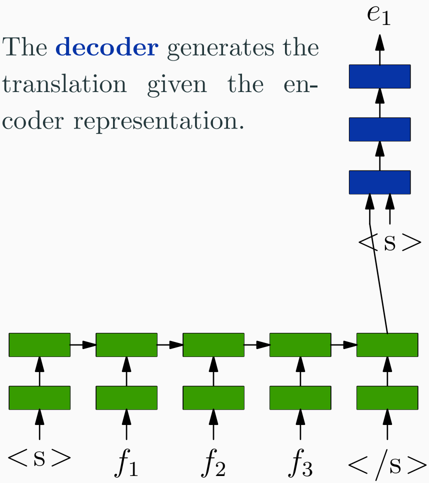



The **encoder** creates a representation of the input sentence.

[Sutskever et al., 2014, Cho et al., 2014b]

# Encoder-Decoder Architecture

The **decoder** generates the translation given the encoder representation.

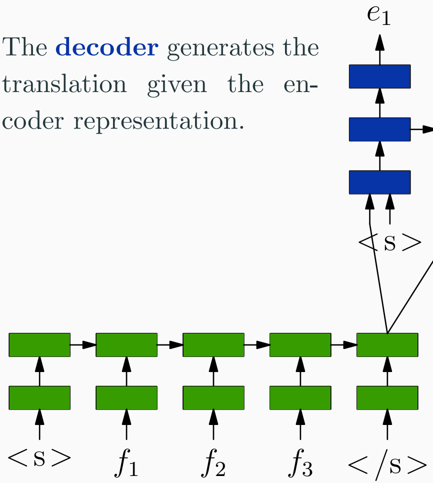



The **encoder** creates a representation of the input sentence.

[Sutskever et al., 2014, Cho et al., 2014b]

# Encoder-Decoder Architecture

The **decoder** generates the translation given the encoder representation.

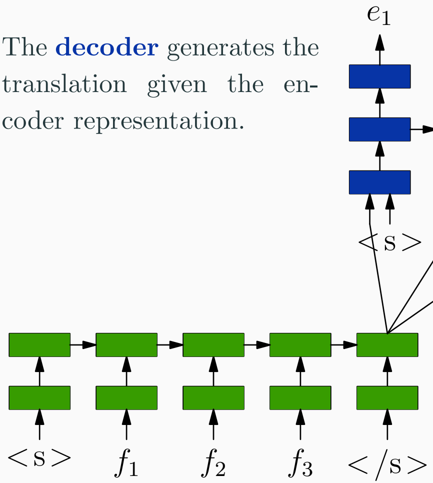



The **encoder** creates a representation of the input sentence.

[Sutskever et al., 2014, Cho et al., 2014b]

# Encoder-Decoder Architecture

The **decoder** generates the translation given the encoder representation.

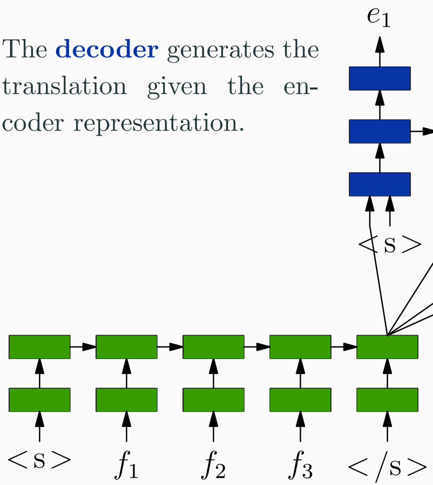



The **encoder** creates a representation of the input sentence.

[Sutskever et al., 2014, Cho et al., 2014b]

# Encoder-Decoder Architecture

The **decoder** generates the translation given the encoder representation.

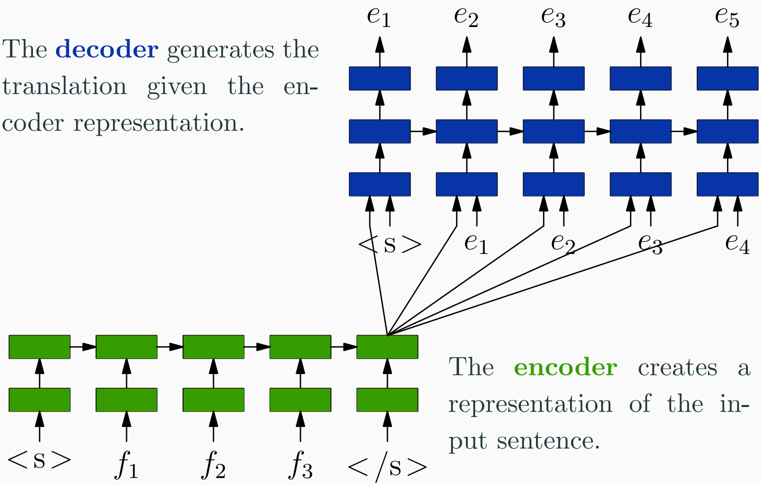



The **encoder** creates a representation of the input sentence.

[Sutskever et al., 2014, Cho et al., 2014b]

# Encoder-Decoder Architecture

The **decoder** generates the translation given the encoder representation.

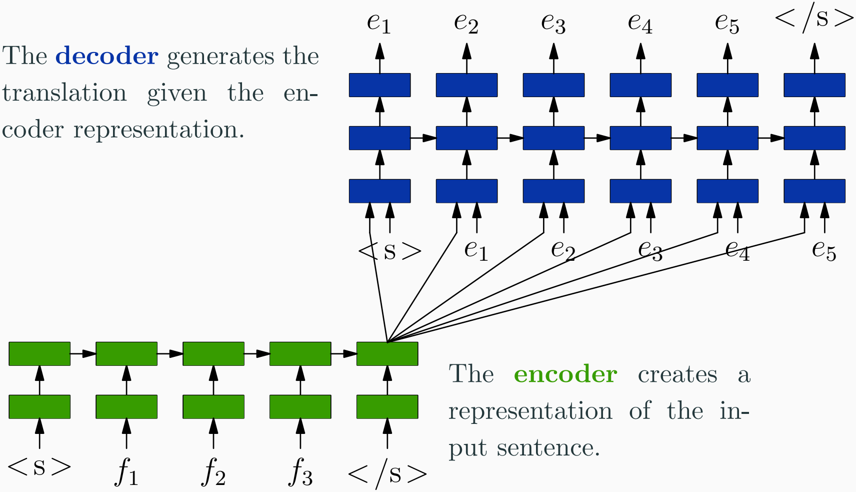



The **encoder** creates a representation of the input sentence.

[Sutskever et al., 2014, Cho et al., 2014b]

# Encoder-Decoder Architecture

The **decoder** generates the translation given the encoder representation.

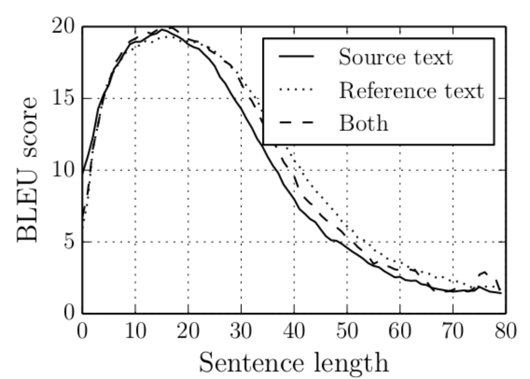



The **encoder** creates a representation of the input sentence.

[Sutskever et al., 2014, Cho et al., 2014b]

# NMT: Issues

A fixed representation lenth may not be enough.




Solution: Include an attention mechanism (next lecture).

[Cho et al., 2014a, Bahdanau et al., 2014]

The encoder-decoder allows for great flexibility, e.g.

# Generalizations

The encoder-decoder allows for great flexibility, e.g.

- General sequence-to-sequence tasks.

# Generalizations

The encoder-decoder allows for great flexibility, e.g.

- General sequence-to-sequence tasks.
- Image based encoder $\rightarrow$ Image captioning system.

# Generalizations

The encoder-decoder allows for great flexibility, e.g.

- General sequence-to-sequence tasks.
- Image based encoder $\rightarrow$ Image captioning system.
- Acoustic based encoder $\rightarrow$ Speech translation system.

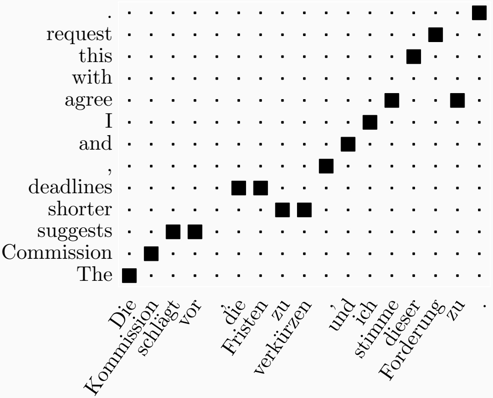
# Generalizations

The encoder-decoder allows for great flexibility, e.g.

- General sequence-to-sequence tasks.
- Image based encoder $\rightarrow$ Image captioning system.
- Acoustic based encoder $\rightarrow$ Speech translation system.
- Combination of different encoders $\rightarrow$ Multimodal translation.

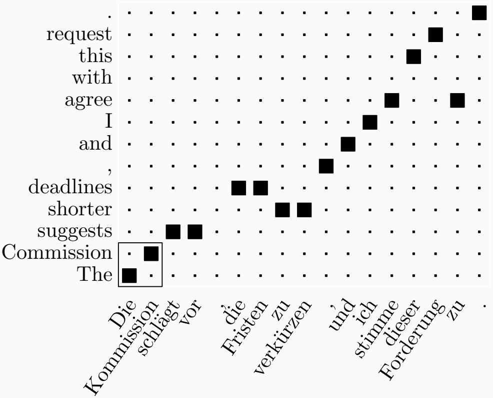
# Historical Perspective

Introduce the concept of aligment.



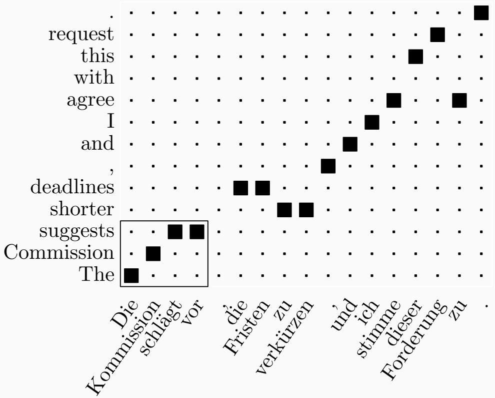

[Brown et al., 1993]

Extract phrases from word alignments.




[Koehn et al., 2003, Och and Ney, 2004]

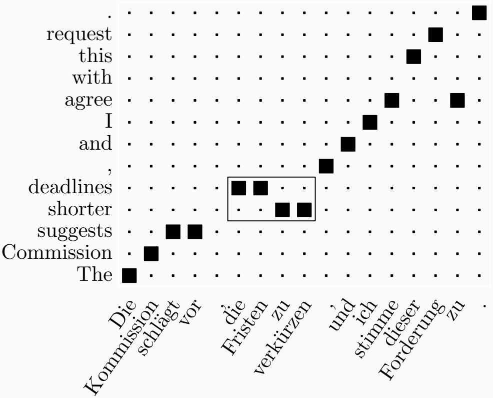
Extract phrases from word alignments.




[Koehn et al., 2003, Och and Ney, 2004]

# From Words to Phrases

Extract phrases from word alignments.




[Koehn et al., 2003, Och and Ney, 2004]

# Log-linear Models

Model the translation probability directly:

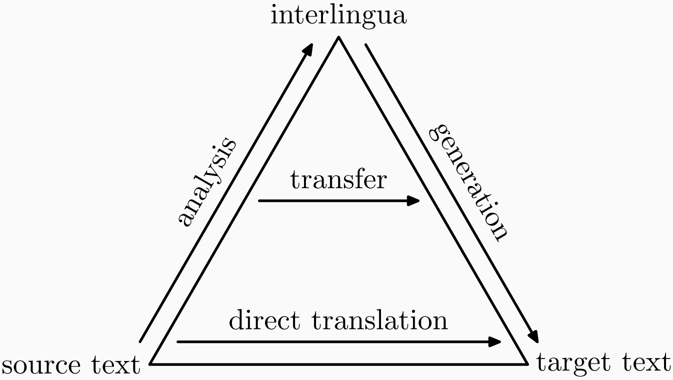
$$p(e_1^I|f_1^J) = \frac{\exp\left(\sum_k \lambda_k f_k(f_1^J, e_1^I)\right)}{\sum_{\hat{e}_1^I} \exp\left(\sum_k \lambda_k f_k(f_1^J, \hat{e}_1^I)\right)}$$

Widely used models:

- Phrase-based models (s2t, t2s).
- Target language model.
- Reordering model.
- Word-based models (s2t, t2s).
- Length heuristics.
- ...

[Och and Ney, 2002]

# Pyramid of Translation Approaches




[Vauquois, 1968]

# Analysis

**PBMT**

**NMT**

# Analysis

**PBMT**

− Local context

**NMT**

+ Global context

# Analysis

**PBMT**

− Local context

∘ Independent models

**NMT**

\+ Global context

\+ Global optimization

# Analysis

**PBMT**

- − Local context
- ∘ Independent models
- − LM one of many models

**NMT**

- + Global context
- + Global optimization
- + Generation guided by LM

## Analysis

**PBMT**

- − Local context
- ∘ Independent models
- − LM one of many models
- + Coverage constraints

**NMT**

- + Global context
- + Global optimization
- + Generation guided by LM
- − Over-/under-generation

# Analysis

**PBMT**

- − Local context
- ∘ Independent models
- − LM one of many models
- + Coverage constraints
- + Model introspection

**NMT**

- + Global context
- + Global optimization
- + Generation guided by LM
- − Over-/under-generation
- − "Black box" approach

# Analysis

**PBMT**

− Local context

∘ Independent models

− LM one of many models

+ Coverage constraints

+ Model introspection

− Model size

**NMT**

+ Global context

+ Global optimization

+ Generation guided by LM

− Over-/under-generation

− “Black box” approach

+ Model size

# Analysis

**PBMT**

- − Local context
- ∘ Independent models
- − LM one of many models
- + Coverage constraints
- + Model introspection
- − Model size

**NMT**

- + Global context
- + Global optimization
- + Generation guided by LM
- − Over-/under-generation
- − "Black box" approach
- + Model size
- − Misspellings/new words

# Implementation

# Deep Learning Toolkits

Efficient algebra (using GPUs) and auto-differentiation.

- MXNet
- Tensorflow
- PyTorch
- Dynet
- [Keras]
- ...

# NMT Toolkits

Implementation of NMT models:

- Sockeye
- OpenNMT
- Marian
- Nematus
- NeuralMonkey
- Tensor2Tensor
- FairSeq
- ...

# Conclusions

# Conclusions

- Introduced the encoder-decoder architecture.
    - The model presented here does *not* achieve SOA.
    - But is the base for more advanced models.
- NN allow for integrated modelling and end-to-end training.
- Word embeddings allow to take advantage of word similarities.

# The End

# References I


Bahdanau, D., Cho, K., and Bengio, Y. (2014).
**Neural machine translation by jointly learning to align and translate.**
*ArXiv e-prints*, abs/1409.0473.

Brown, P. E., Pietra, S. A. D., Pietra, V. J. D., and Mercer, R. L. (1993).
**The mathematics of statistical machine translation: Parameter estimation.**
*Computational Linguistics, Volume 19, Number 2, June 1993, Special Issue on Using Large Corpora: II.*

# References II


Chen, S. F. and Goodman, J. (1996).
**An empirical study of smoothing techniques for language modeling.**
In *34th Annual Meeting of the Association for Computational Linguistics.*

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014a).
**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. Association for Computational Linguistics.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014b).
**Learning phrase representations using rnn encoder–decoder for statistical machine translation.**
In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics.

Hochreiter, S. and Schmidhuber, J. (1997).
**Long short-term memory.**
*Neural computation*, 9(8):1735–1780.

# References IV


Kneser, R. and Ney, H. (1995).
**Improved backing-off for m-gram language modeling.**
In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 181–184, Detroit, Michigan, USA.

Koehn, P., Och, F. J., and Marcu, D. (2003).
**Statistical phrase-based translation.**
In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics.*

# References V


Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013).
**Efficient estimation of word representations in vector space.**
*ArXiv e-prints*, abs/1301.3781.

Och, F. J. and Ney, H. (2002).
**Discriminative training and maximum entropy models for statistical machine translation.**
In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics.*

# References VI


Och, F. J. and Ney, H. (2004).
**The alignment template approach to statistical machine translation.**
*Computational Linguistics, Volume 30, Number 4, December 2004.*

Pascanu, R., Mikolov, T., and Bengio, Y. (2013).
**On the difficulty of training recurrent neural networks.**
In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1310–1318, Atlanta, Georgia, USA. PMLR.

# References VII


Sutskever, I., Vinyals, O., and Le, Q. V. (2014).
**Sequence to sequence learning with neural networks.**
*ArXiv e-prints*, abs/1409.3215.

Vauquois, B. (1968).
**A survey of formal grammars and algorithms for recognition and transformation in machine translation.**
In *IFIP Congress-68*, pages 254–260, Edinburgh.