STADS

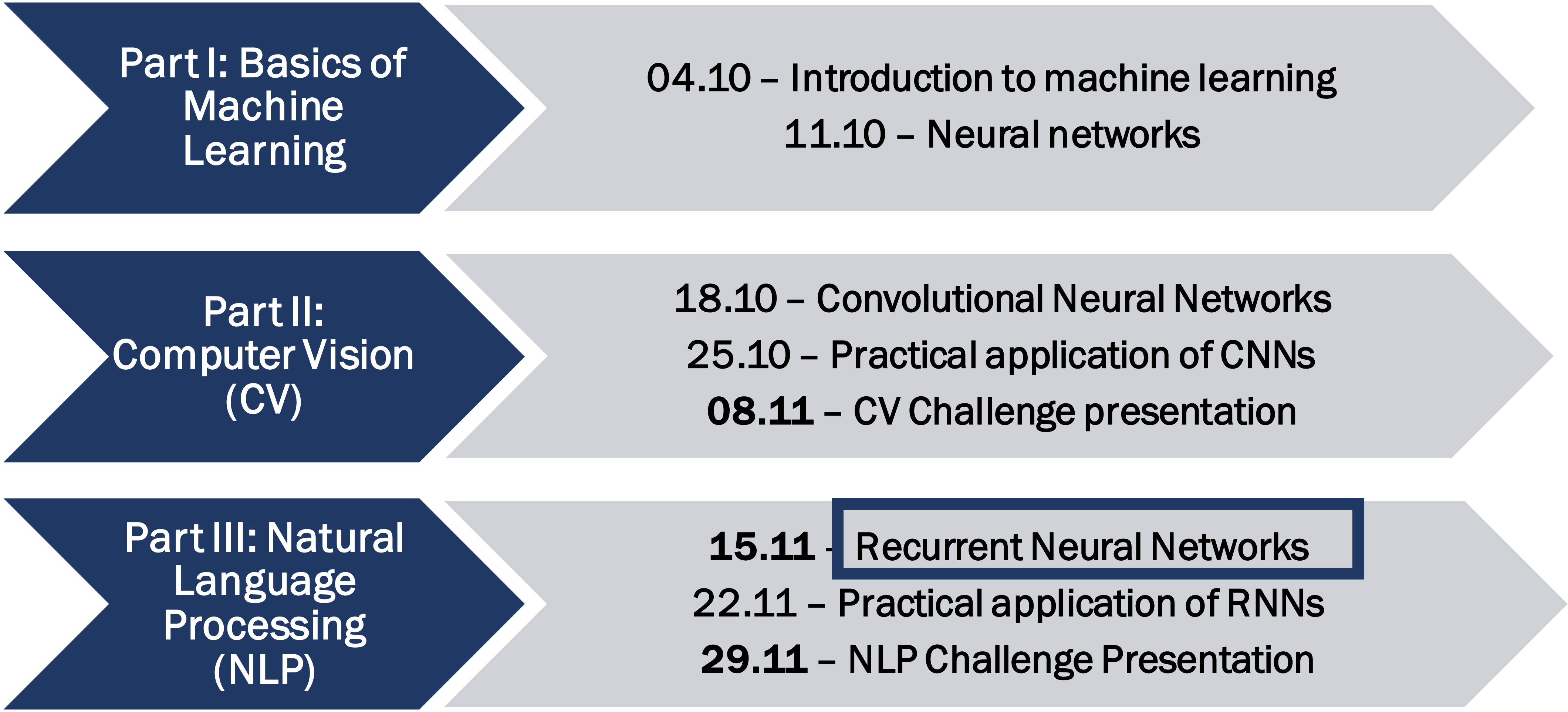
Machine Learning Course
Workshop VI: Natural Language Processing

In **Kooperation** mit

Prof. Dr. Martin Schlather
Lehrstuhl für Stochastik und ihre Anwendungen

Prof. Dr. Leif Döring
Lehrstuhl für Stochastik

Course structure



Part I: Basics of Machine Learning

04.10 – Introduction to machine learning

11.10 – Neural networks

Part II: Computer Vision (CV)

18.10 – Convolutional Neural Networks

25.10 – Practical application of CNNs

08.11 – CV Challenge presentation

Part III: Natural Language Processing (NLP)

15.11 - Recurrent Neural Networks

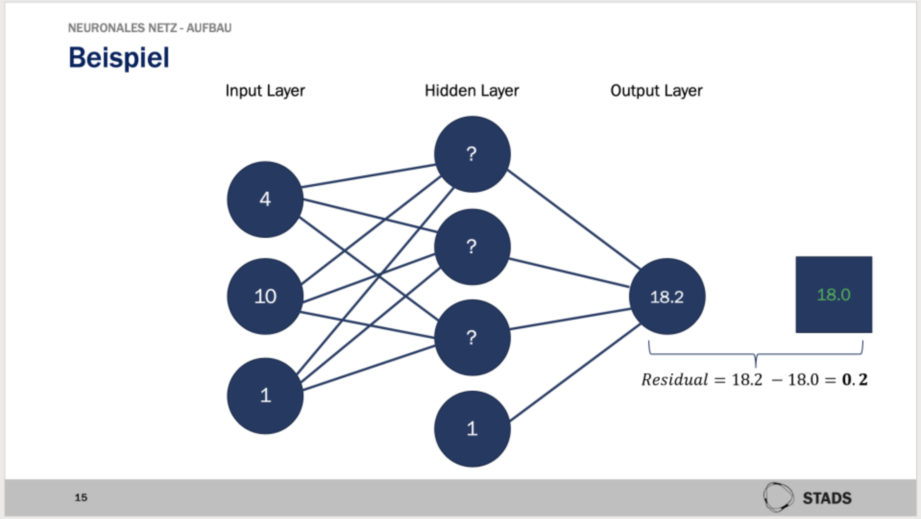
22.11 – Practical application of RNNs

29.11 – NLP Challenge Presentation

Disclaimer

This presentation is based on the course on Sequence Models from DeepLearning.ai, as well as the Deep Learning course by Prof. Gemulla. These are highly recommended for in-depth knowledge.

What happened so far…

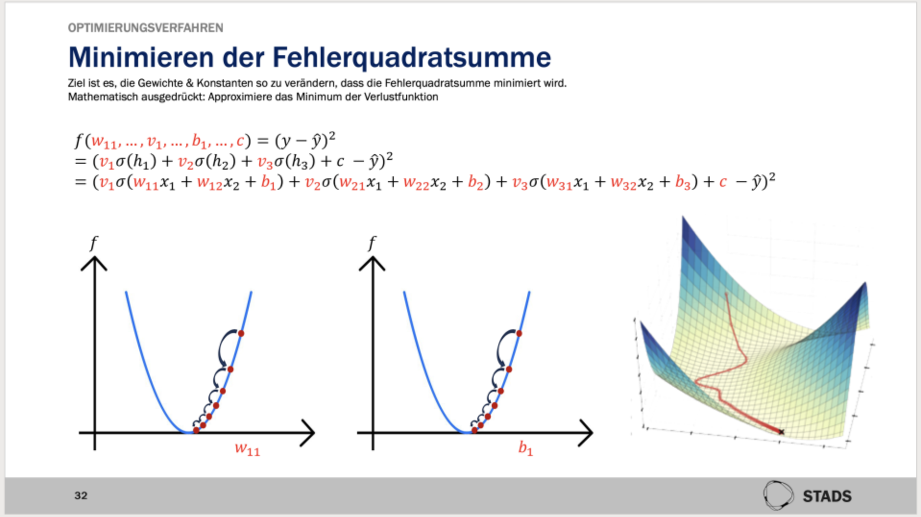


NEURONALES NETZ - AUFBAU

Beispiel

$Residual = 18.2\ - 18.0 = \mathbf{0.2}$

15 STADS



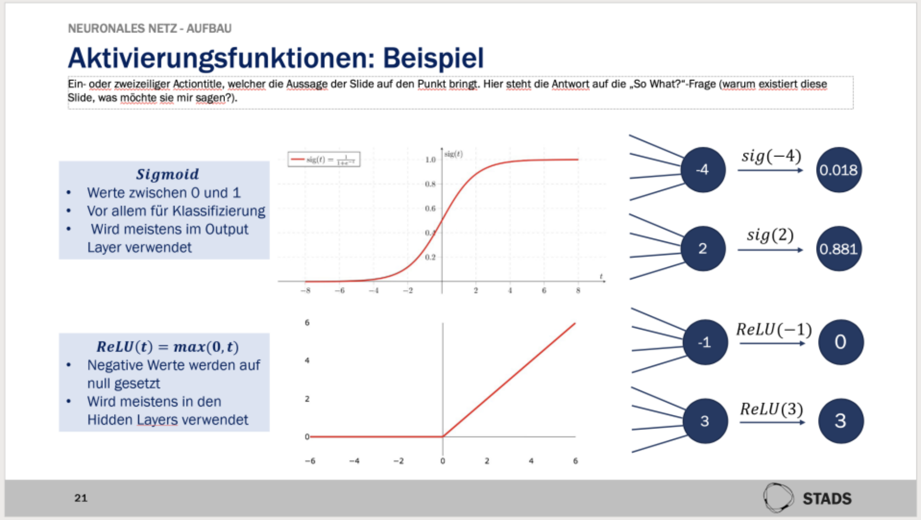
OPTIMIERUNGSVERFAHREN

Minimieren der Fehlerquadratsumme

Ziel ist es, die Gewichte & Konstanten so zu verändern, dass die Fehlerquadratsumme minimiert wird.
Mathematisch ausgedrückt: Approximiere das Minimum der Verlustfunktion

$$f(w_{11},\ldots,v_1,\ldots,b_1,\ldots,c) = (y-\hat{y})^2$$
$$= (v_1\sigma(h_1) + v_2\sigma(h_2) + v_3\sigma(h_3) + c\ - \hat{y})^2$$
$$= (v_1\sigma(w_{11}x_1 + w_{12}x_2 + b_1) + v_2\sigma(w_{21}x_1 + w_{22}x_2 + b_2) + v_3\sigma(w_{31}x_1 + w_{32}x_2 + b_3) + c\ - \hat{y})^2$$

32 STADS



NEURONALES NETZ - AUFBAU

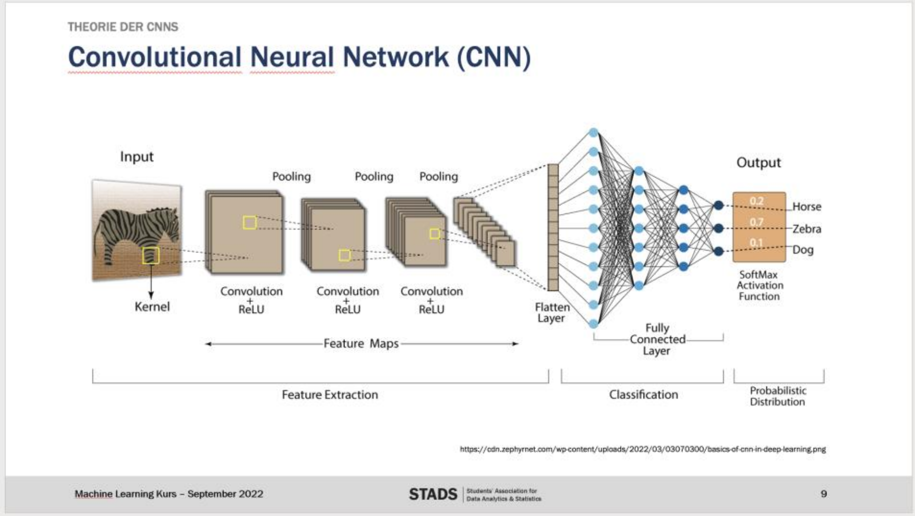
Aktivierungsfunktionen: Beispiel

Ein- oder zweizeiliger Actiontitle, welcher die Aussage der Slide auf den Punkt bringt. Hier steht die Antwort auf die „So What?"-Frage (warum existiert diese Slide, was möchte sie mir sagen?).

Sigmoid
- Werte zwischen 0 und 1
- Vor allem für Klassifizierung
- Wird meistens im Output Layer verwendet

$\boldsymbol{ReLU(t) = max(0,t)}$
- Negative Werte werden auf null gesetzt
- Wird meistens in den Hidden Layers verwendet

21 STADS



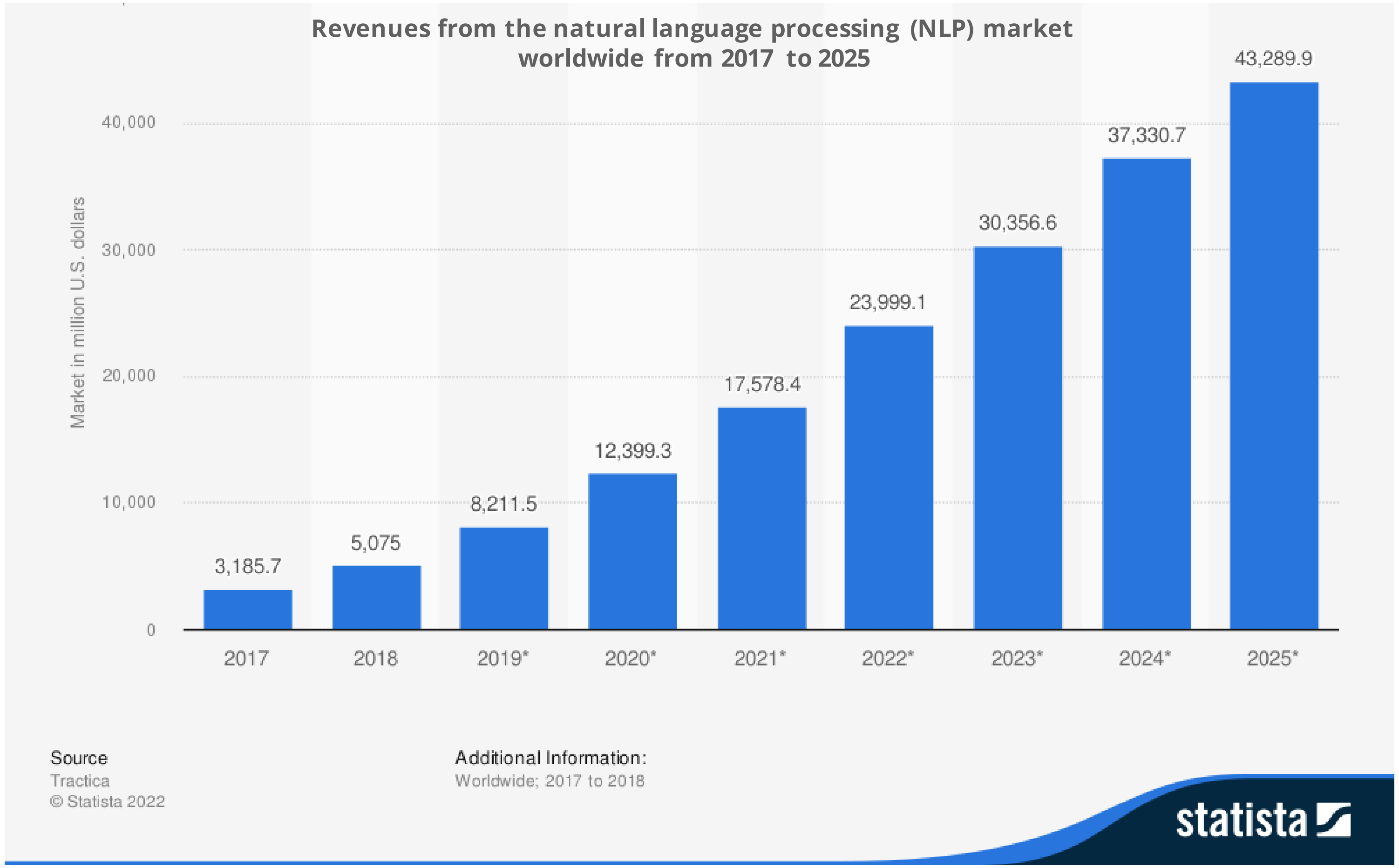
THEORIE DER CNNS

Convolutional Neural Network (CNN)

https://cdn.zephyrnet.com/wp-content/uploads/2022/03/03070300/basics-of-cnn-in-deep-learning.png

Machine Learning Kurs – September 2022 STADS Students' Association for Data Analytics & Statistics 9

43% of global companies are already using NLP to increase efficiency[1]



Revenues from the natural language processing (NLP) market worldwide from 2017 to 2025

Source
Tractica
© Statista 2022

Additional Information:
Worldwide; 2017 to 2018

[1] Quelle (25.10.2022): https://www.reworked.co/information-management/why-companies-are-investing-in-natural-language-processing/

What actually is Natural Language Processing?

Natural Language Processing

Natural language processing aims to develop machines that can understand text or voice input and respond with their own text or voice output - in the same way humans do.[1]

Use cases

1. Speech recognition
2. Sentiment Analysis
3. Named Entity Recognition (NEM)
4. Chat bots

Building blocks

Classic machine learning or deep learning

+

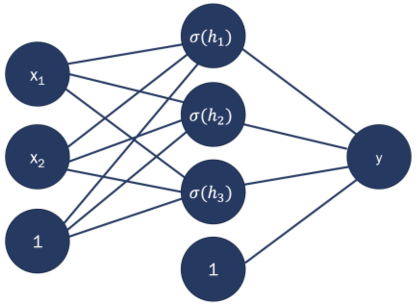
LSTMs, Attention, Embeddings

[1] https://www.ibm.com/cloud/learn/natural-language-processing

Challenges of Natural Language Processing

Variable Input-/Output lengths

- Typically, the number of words in sentences varies
- Therefore, classic neural networks with fixed inputs and outputs are unsuitable



Sequence is important

“The food in the cafeteria is good and not too expensive.” 🙂

“The food in the cafeteria is not good and too expensive.” ☹

Long-term word dependencies

- Words at the beginning of a long sentence can refer to words at the end of the sentence.

“The dinosaurs, which became extinct 66 million years ago, were present on every continent.”

Bidirectional dependencies

- The meaning of a word depends not only on the context before the word, but also the words following it.

„Die Bank hat ein großes Gebäude in Frankfurt.“

The approach: Recurrent Neural Networks

Important and exciting applications of RNNs

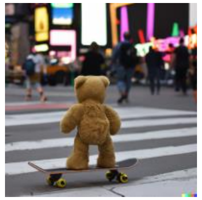
	Input		Output
• Named entity recognition	Tomorrow Nicki Minaj plays in Mannheim	→	Tomorrow [Nicki Minaj]$_{\text{Person}}$ plays in [Mannehim]$_{\text{Ort}}$
• Text classification	The movie was surprisingly bad.	→	★☆☆☆☆
• Machine translation	Das Schiffsverkehr ist eingestellt.	→	Shipping traffic is suspended.
• Text completion	Write a slogan for a dating app.	→	Find your match today!
• Speech recognition		→	Alexa, what's the weather?
• Image generation	a teddy bear on a skateboard in times square	→	
• Video analysis		→	Dog catching frisbee

Image sources: https://cdn.openai.com/papers/dall-e-2.pdf
https://www.clipartmax.com/middle/m2H7G6K9m2K9m2b1_audio-sound-wave-png/
https://giphy.com/gifs/tvland-leah-remini-king-of-queens-kingofqueens-VJwck53yG6y8s2H30g

Agenda

Introduction

Theory of RNNs using NLP as an example

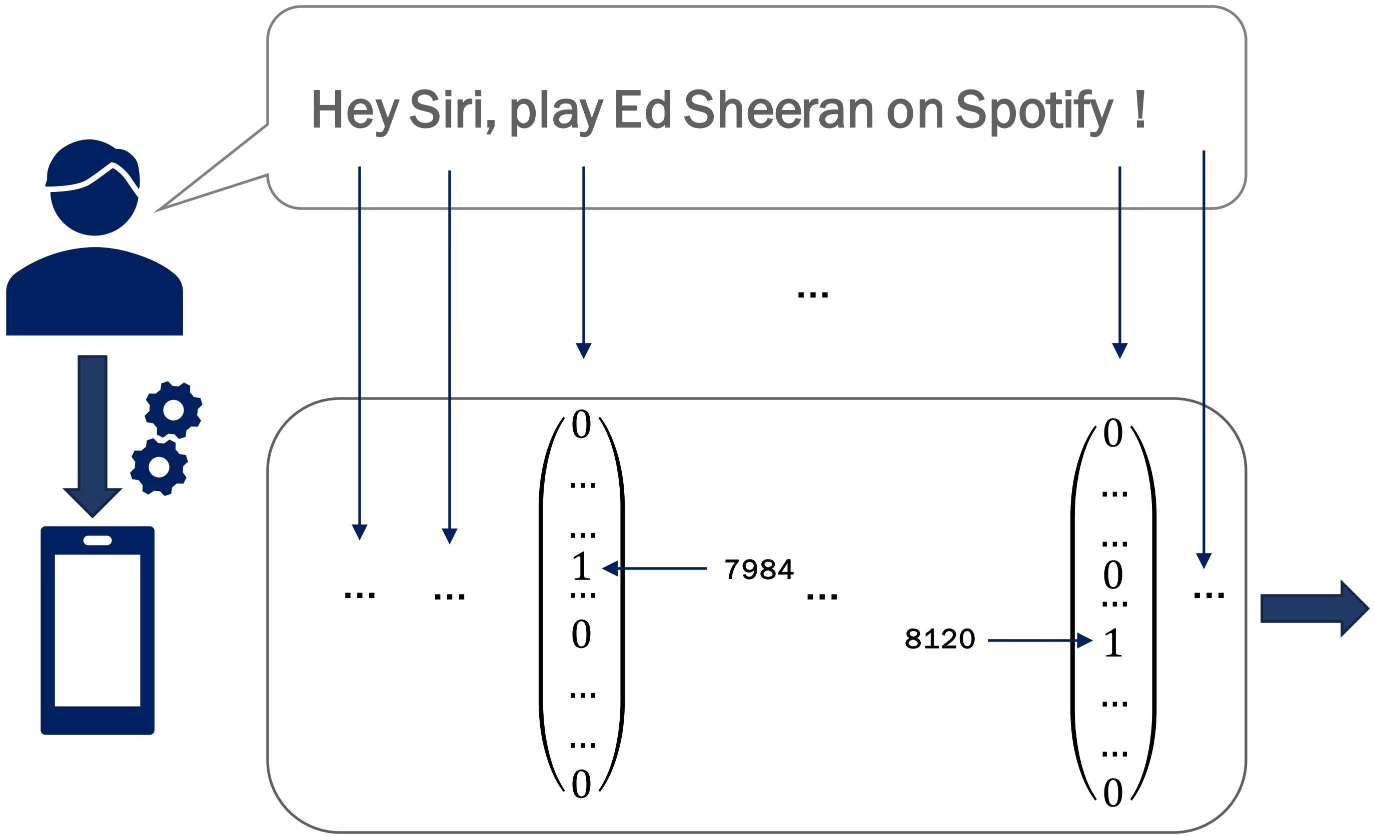
Programming project

How can we encode language?

A “dictionary” allows us to represent words as unit vectors (one-hot encoding):

Dictionary

Aachen	1
…	
…	
play	7984
…	
Spotify	8120
…	
…	
Zyzzyva	14000



Hey Siri, play Ed Sheeran on Spotify !

…
… …
$\begin{pmatrix}0\\ \vdots\\ \vdots\\ 1\\ \vdots\\ 0\\ \vdots\\ \vdots\\ 0\end{pmatrix}$
7984
…
$\begin{pmatrix}0\\ \vdots\\ \vdots\\ 0\\ \vdots\\ 1\\ \vdots\\ \vdots\\ 0\end{pmatrix}$
8120
…

Bildquelle: https://www.billboard.com/wp-content/uploads/2021/09/Ed-Sheeran-2021-vmas-performance-billboard-1548-1631495259.jpg?w=942&h=623&crop=1

Example: Named entity recognition

Alexa, play Ed Sheeran on Apple Music!

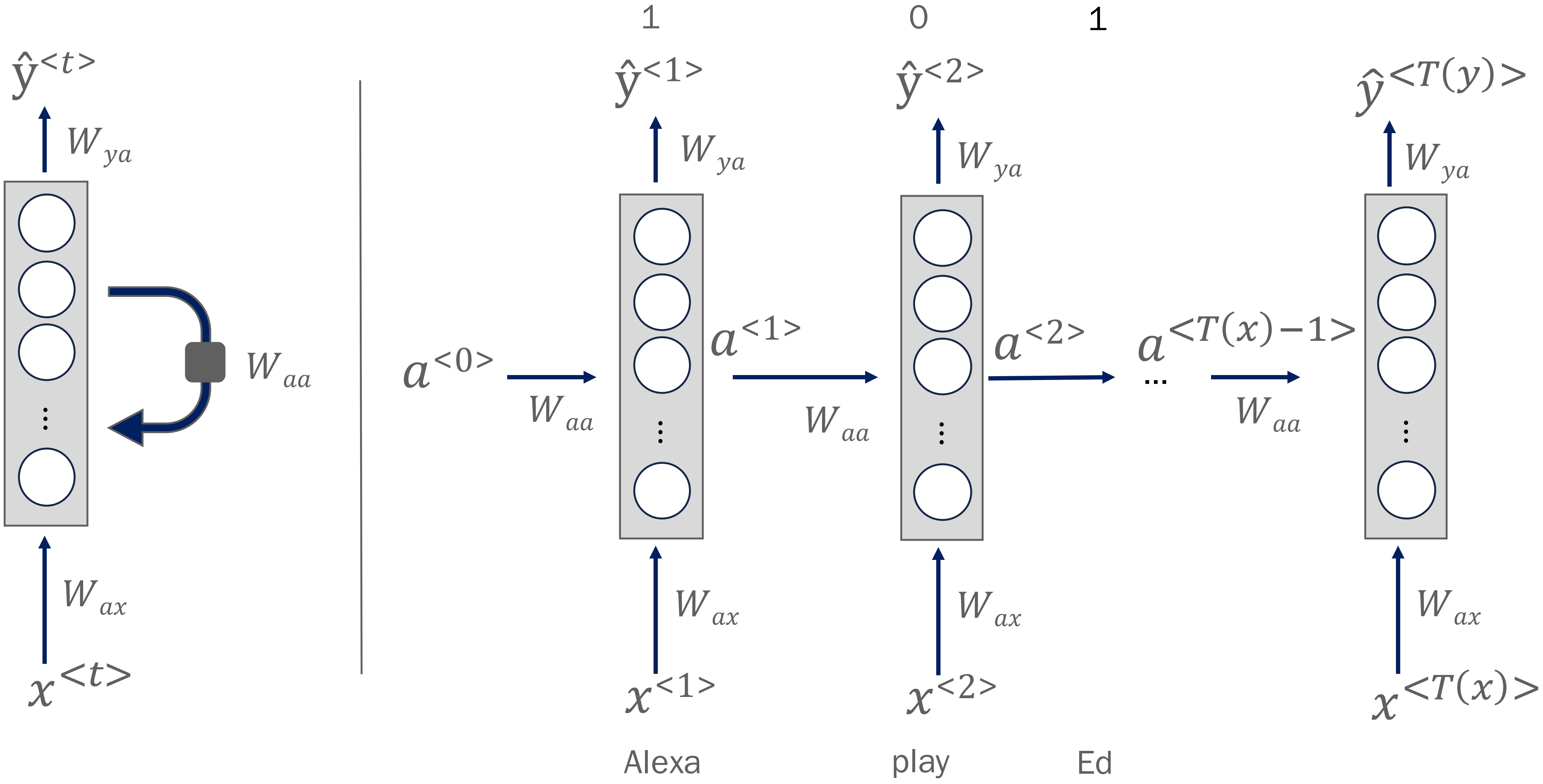
$\boldsymbol{x}$: $x^{<1>}$ $x^{<2>}$ • • • • $x^{<7>}$

$$\begin{pmatrix} 0 \\ 1 \\ \cdots \\ 0 \\ \cdots \\ 0 \\ \cdots \\ \cdots \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ \cdots \\ \cdots \\ 0 \\ \cdots \\ 1 \\ \cdots \\ \cdots \\ 0 \end{pmatrix} \qquad\qquad \begin{pmatrix} 0 \\ \cdots \\ \cdots \\ 1 \\ \cdots \\ 0 \\ \cdots \\ \cdots \\ 0 \end{pmatrix}$$

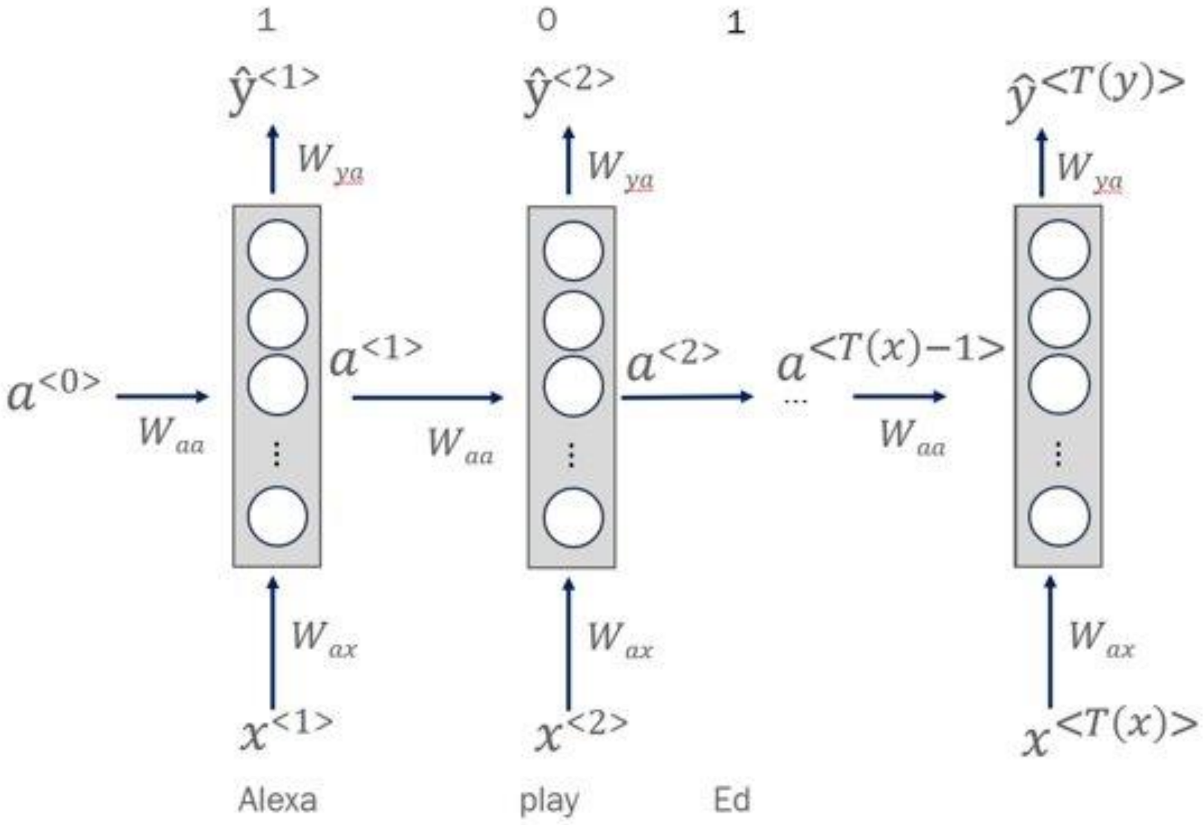
1 0 1 1 0 0 0

$\boldsymbol{y}$: $y^{<1>}$ $y^{<2>}$ • • • • $y^{<7>}$

Architecture of RNNs



Building RNNs



$$a^{<0>} = 0$$

$$a^{<1>} = g_1(W_{aa}\, a^{<0>} + W_{ax}\, x^{<1>} + b_a)$$

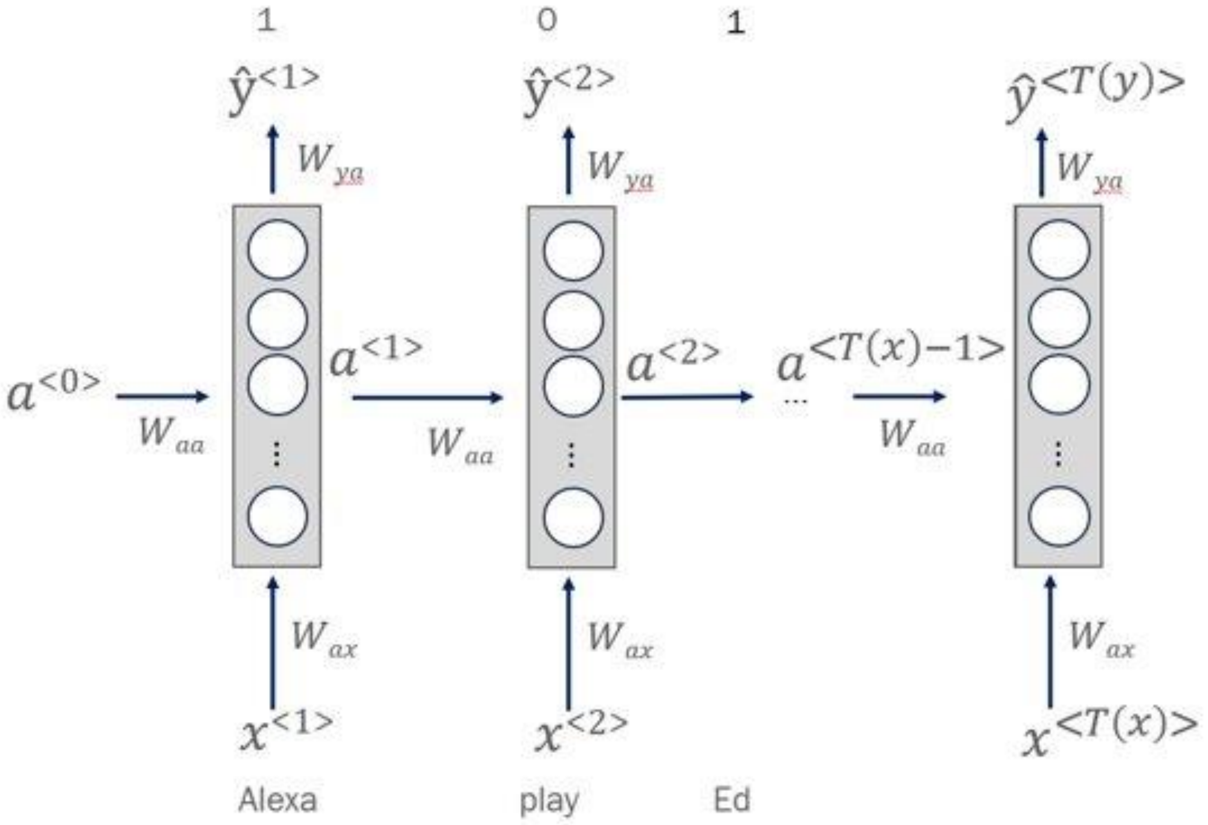
$$\hat{y}^{<1>} = g_2(W_{ya}\, a^{<1>} + b_y)$$

$$\vdots$$

$$a^{<t>} = g(Waa\, a^{<t-1>} + W_{ax}\, x^{<t>} + b_a)$$

$$\hat{y}^{<t>} = g(W_{ya}\, a^{<t>} + b_y)$$

Building RNNs (simplified Notation)



$W_a = [W_{aa} \mid W_{ax}]$

$[a^{t-1}\, x^{<t>}] = \begin{pmatrix} a^{<t\text{-}1>} \\ x^{<t>} \end{pmatrix}$

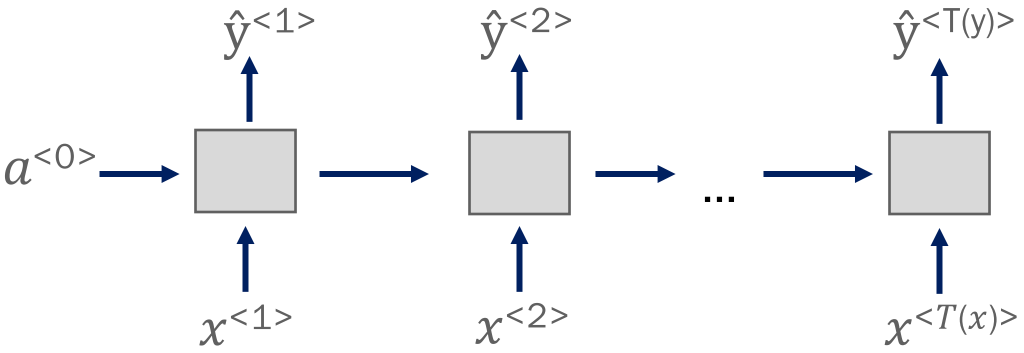
$a^{<t>} = g(W_a\,[a^{<t\text{-}1>}, x^{<t>}] + b_a)$

$\hat{y}^{<t>} = g(W_{ya}\,a^{<t>} + b_y)$

Example of RNN architecture

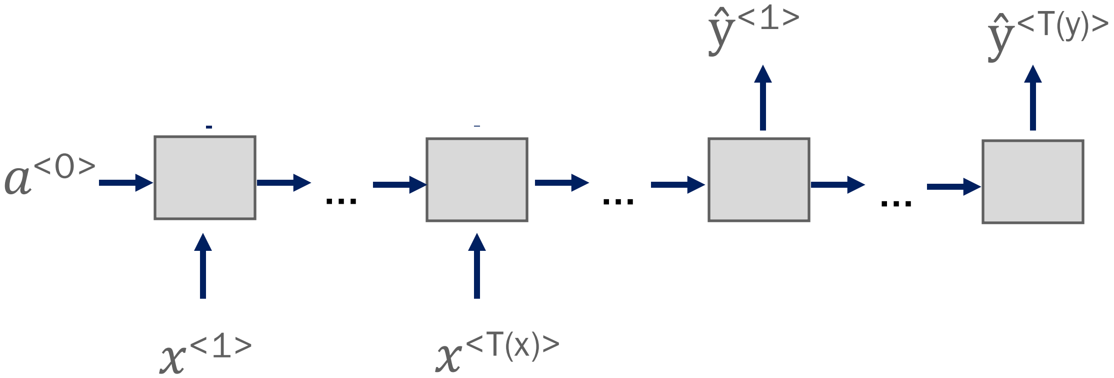
Many-to-many $(T(x) = T(y))$

Named-Entity-Recognition, Speech-to-Text



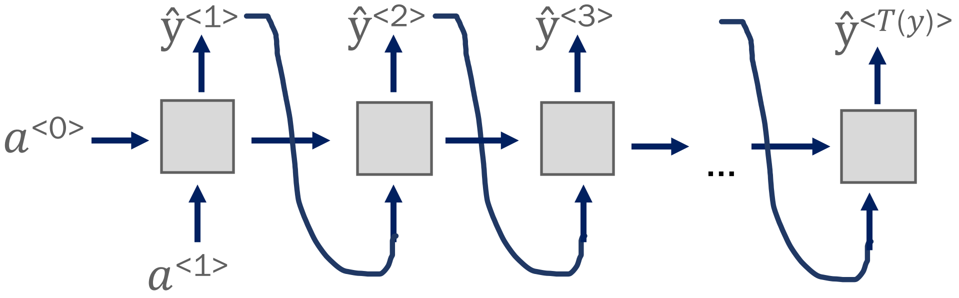
Many-to-many $(T(x) \neq T(y))$

Text-translation



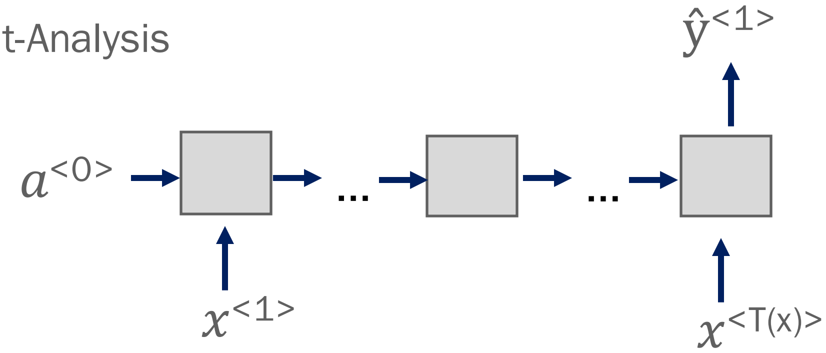
One-to-many

Music generation, Language modelling



Many-to-one

Sentiment-Analysis

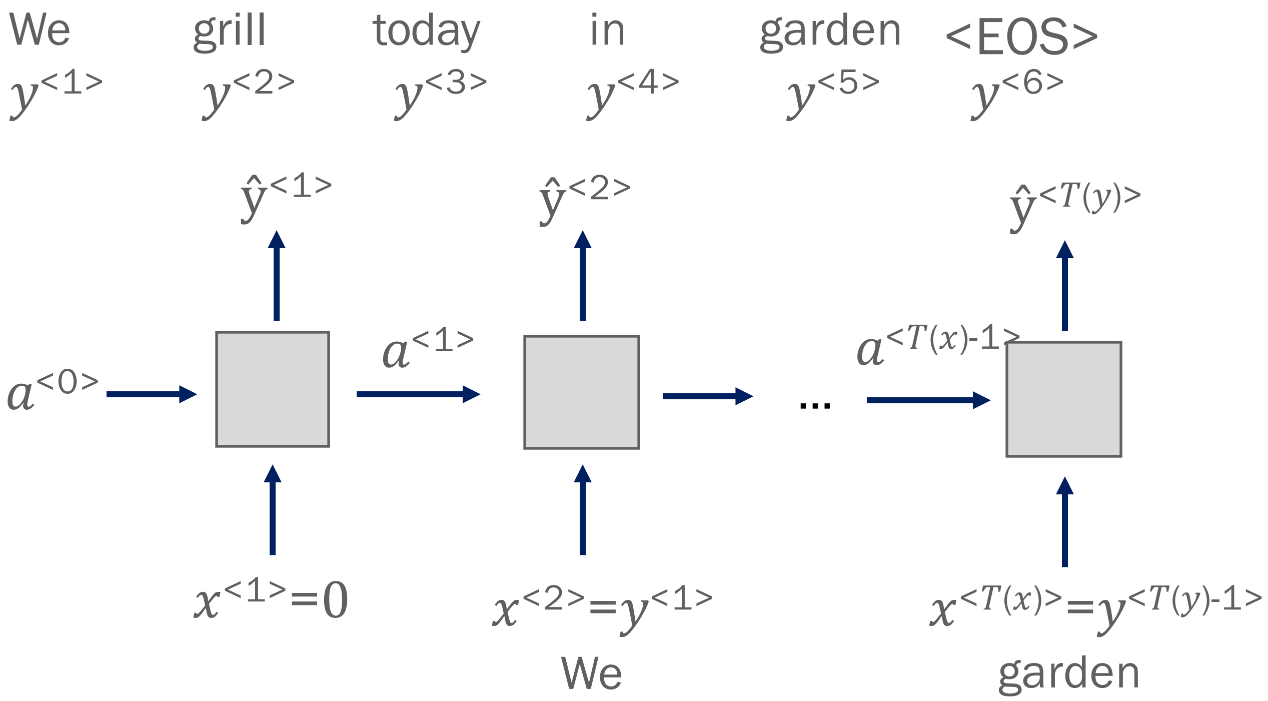


Language Models

Distribution across sequences of words

- Predicting the next word/letter in a text
- Basis for many other NLP applications (text classification, question answering, speech recognition), examples: BERT, GPT-3,

<u>Training data</u>: large dataset of texts (e.g. Wikipedia articles).

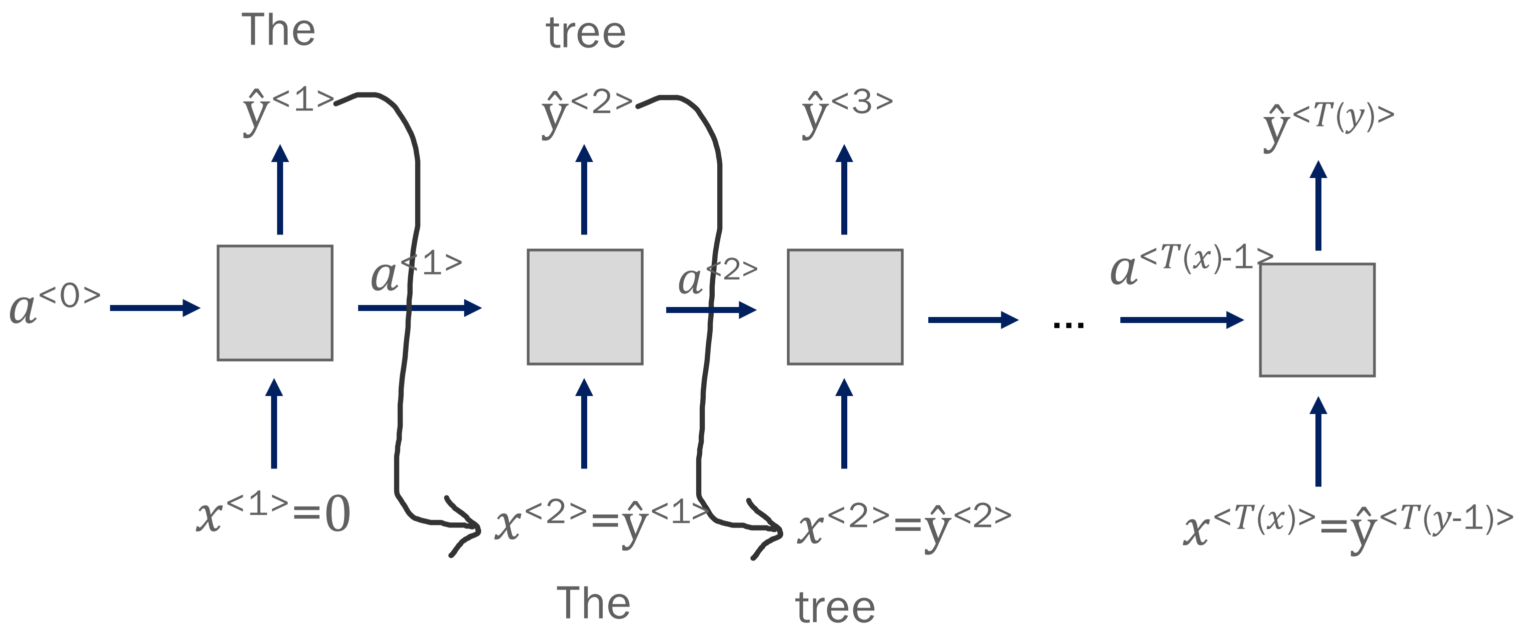


- $\hat{y}^{<1>}$ - Softmax over dictionary
  → modeled
  $(p(aachen), \dots, p(spielen), \dots p(wir), \dots)$
- $\hat{y}^{<2>} \rightarrow p(\cdot | wir)$
- $\hat{y}^{<3>} \rightarrow p(\cdot | \text{“}wir\ grillen\text{“})$

- $p\left(y^{<1>}, \cdots, y^{<T(y)>}\right) = p(y^{<1>}) p(y^{<2>} | y^{<1>}) * \cdots * p(y^{<T(y)>} | \, y^{<1>}, \cdots, y^{<T(y)-1>})$

- Loss:
  $$\mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -\sum_i y_i^{<t>} \log\left(\hat{y}_i^{<t>}\right)$$
  $$\mathcal{L} = \sum_t \mathcal{L}^{<t>} (\hat{y}^{<t>}, y^{<t>}).$$

Generating new Sequences



- Generated words are recursively used as the next input
- Sampling the words according to softmax output (probability vectors) using vocabulary

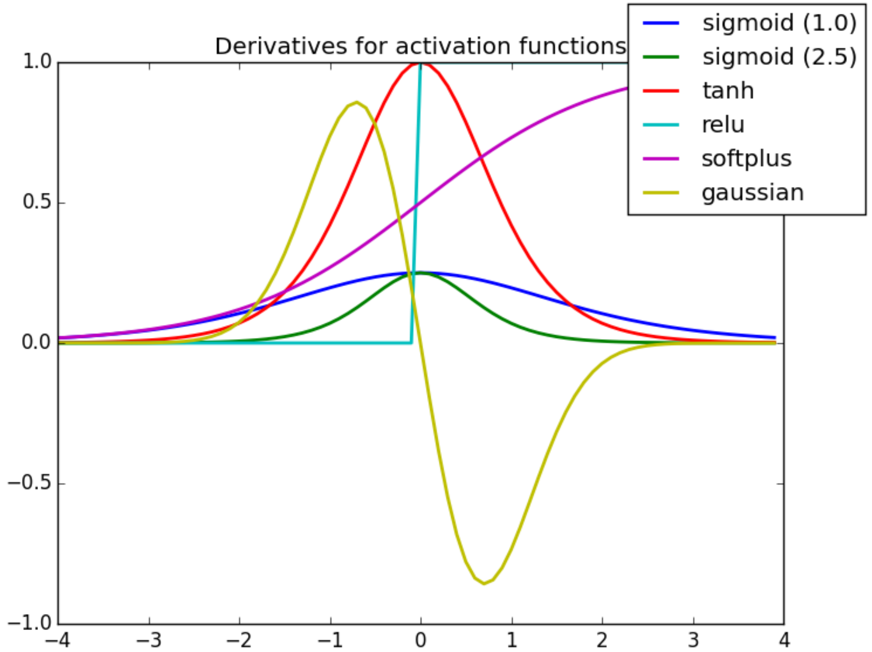
Vanishing Gradients

$$\frac{\partial \mathcal{L}}{\partial W} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}_t}{\partial W} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}_t}{\partial y_t} \cdot \frac{\partial y_t}{\partial a_t} \cdot \frac{\partial a_t}{\partial a_{t-1}} \cdot \ldots \cdot \frac{\partial a_2}{\partial a_1} \cdot \frac{\partial a_1}{\partial W} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}_t}{\partial y_t} \cdot \frac{\partial y_t}{\partial a_t} \cdot \underbrace{\prod_{k=1}^{t-1} \frac{\partial a_{k+1}}{\partial a_k}} \cdot \frac{\partial a_1}{\partial W}$$

$$a_t = g(W_{aa} \cdot a^{t-1} + W_{ax} \cdot x^t + b_a)$$

$$\frac{\partial a_t}{\partial a_{t-1}} = g'(W_{aa} \cdot a_{t-1} + W_{ax} \cdot x_t + b_a) \cdot \frac{\partial}{\partial a_{t-1}} [W_{aa} \cdot a_{t-1} + W_{ax} \cdot x_t]$$

$$= \underbrace{g'(\ldots)}_{< 1} \cdot W_{aa}$$



Bildquelle (23.10.22):
https://miro.medium.com/max/1400/1*n1HFBpwv21FCAzGjmWt1sg.png

Problem: Since the gradient decreases with the number of words, long-term word dependencies are not taken into account when updating the parameters! But this is necessary for language models.

Source: https://medium.datadriveninvestor.com/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577

Vanishing Gradients

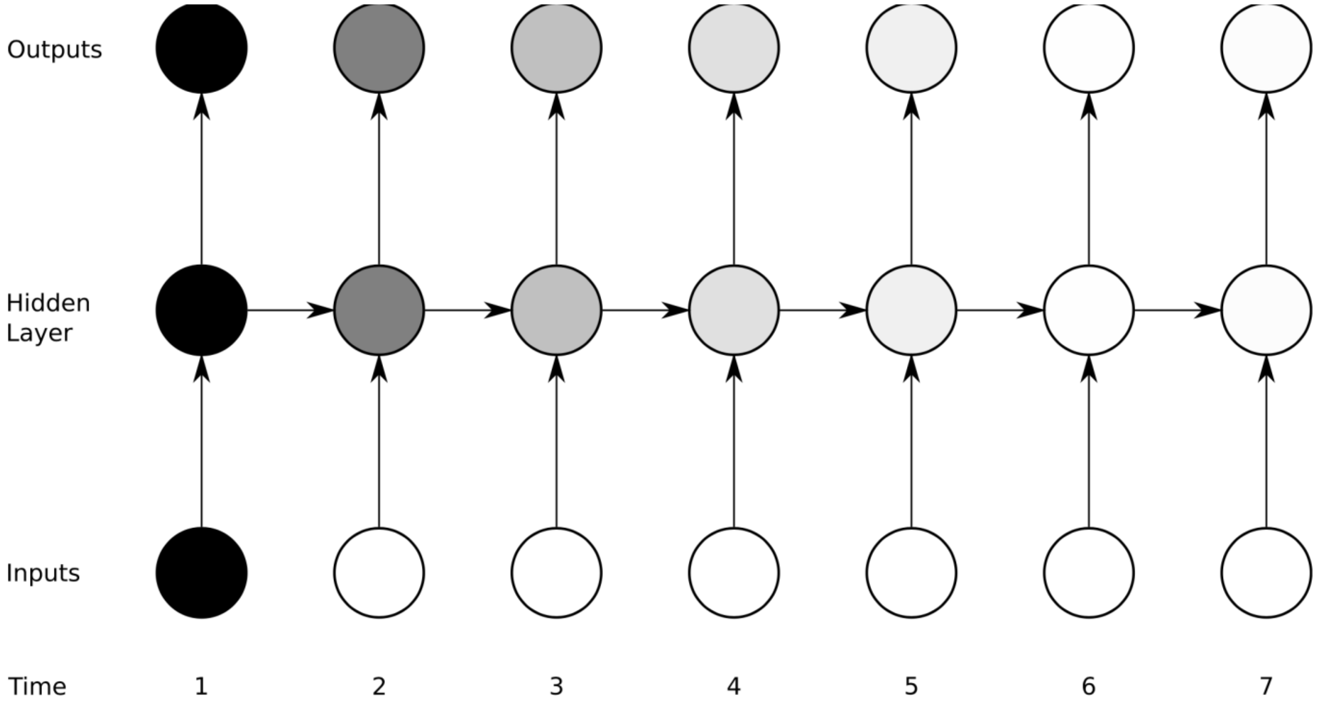


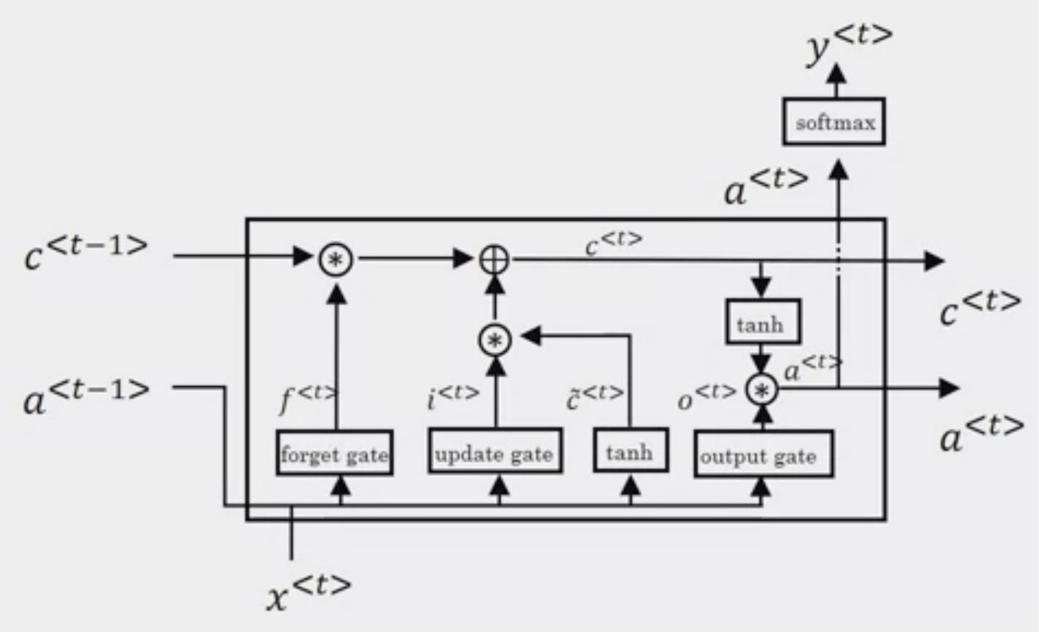
Fig. 10. Illustration of the vanishing gradient problem. The diagram represents a recurrent network unrolled in time. The units are shaded according to how sensitive they are to the input at time 1 (where black is high and white is low). As can be seen, the influence of the first input decays exponentially over time.

LSTMs can solve this problem!

Quelle (23.10.2022):
https://www.researchgate.net/publication/24213728_A_Novel_Connectionist_System_for_Unconstrained_Handwriting_Recognition

Long short-term memory cells (LSTM)

- Solution to the Vanishing Gradient Problem
- Introduction of a “memory cell” c<t> (same dimension as activation a<t>)
- “Gating mechanism” decides when memory cell is updated



$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

update ⟶ $\Gamma_u = \sigma(W_u[\, a^{<t-1>}, x^{<t>}] + b_u)$

forget ⟶ $\Gamma_f = \sigma(W_f[\, a^{<t-1>}, x^{<t>}] + b_f)$

output ⟶ $\Gamma_o = \sigma(W_o[\, a^{<t-1>}, x^{<t>}] + b_o)$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

$$a^{<t>} = \Gamma_o * \tanh c^{<t>}$$

*: Element-by-element multiplication of the vectors

Intuition: Sigmoid function σ to a small area either ≈ 0 or ≈ 1

⟶ Memory cell remains almost constant when $\Gamma_u \approx 0$ and $\Gamma_f \approx 1$

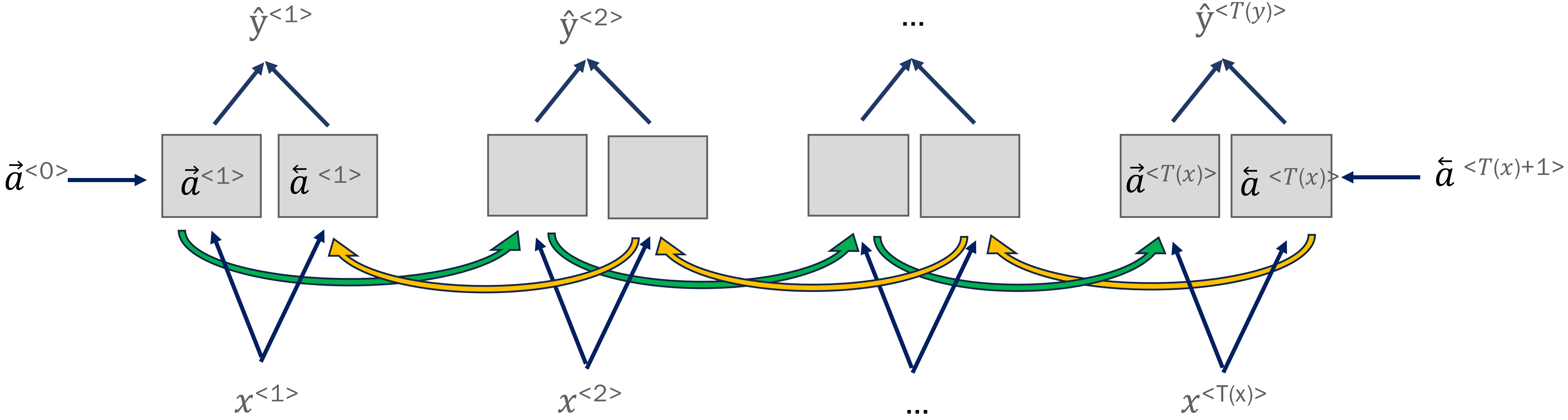
⟶ preserves sensitivity of the outputs to inputs from much earlier time steps

Bildquelle: DeepLearning.ai

Bidirectional RNN‘s

An apple a day is, keeps the doctor away. → 🍎

Apple devices use a lot of Machine Learning. → Brand/company name $\hat{y}^{<t>} = g(W_{ya}[\vec{a}^{<t>}, \overleftarrow{a}^{<t>}] + b_y)$



- Hidden units are also connected in the backward direction → enables use of information from the future
- Not suitable for real-time applications

Word Embeddings

so far: Representation of every word from a dictionary of size V as a one-hot vector

$$\text{Frau} \triangleq \begin{pmatrix} 0 \\ \ldots \\ 0 \\ 1 \\ 0 \\ \ldots \\ \ldots \\ \ldots \\ 0 \end{pmatrix} = e_{26923}$$

Mann	$\triangleq e_{51034}$
Königin	$\triangleq e_{34263}$
König	$\triangleq e_{34003}$
super	$\triangleq e_{86831}$
toll	$\triangleq e_{89239}$

Solution: feature vectors / word embeddings

- Representation of words using 300 dimensions n vector emb$_j$
- Embedding Matrix $E \in \mathbb{R}^{300 \times V}$
- $emb_i = E \cdot e_i$
- Embedding Layer implements efficient mapping
- Enables Transfer Learning

Problem:

- $\left\| e_i - e_j \right\|_2 = 1 \; \forall \, i \neq j$
- $\dim(e_i) = V$, typically V >>10000 (word2vec $V \approx$ 3M)

We want:

$e_{\text{Mann}} - e_{\text{Frau}} \approx e_{\text{König}} - e_{\text{Königin}}$

$e_{\text{super}} \approx e_{\text{toll}}$

Classification

"The film was awesome"

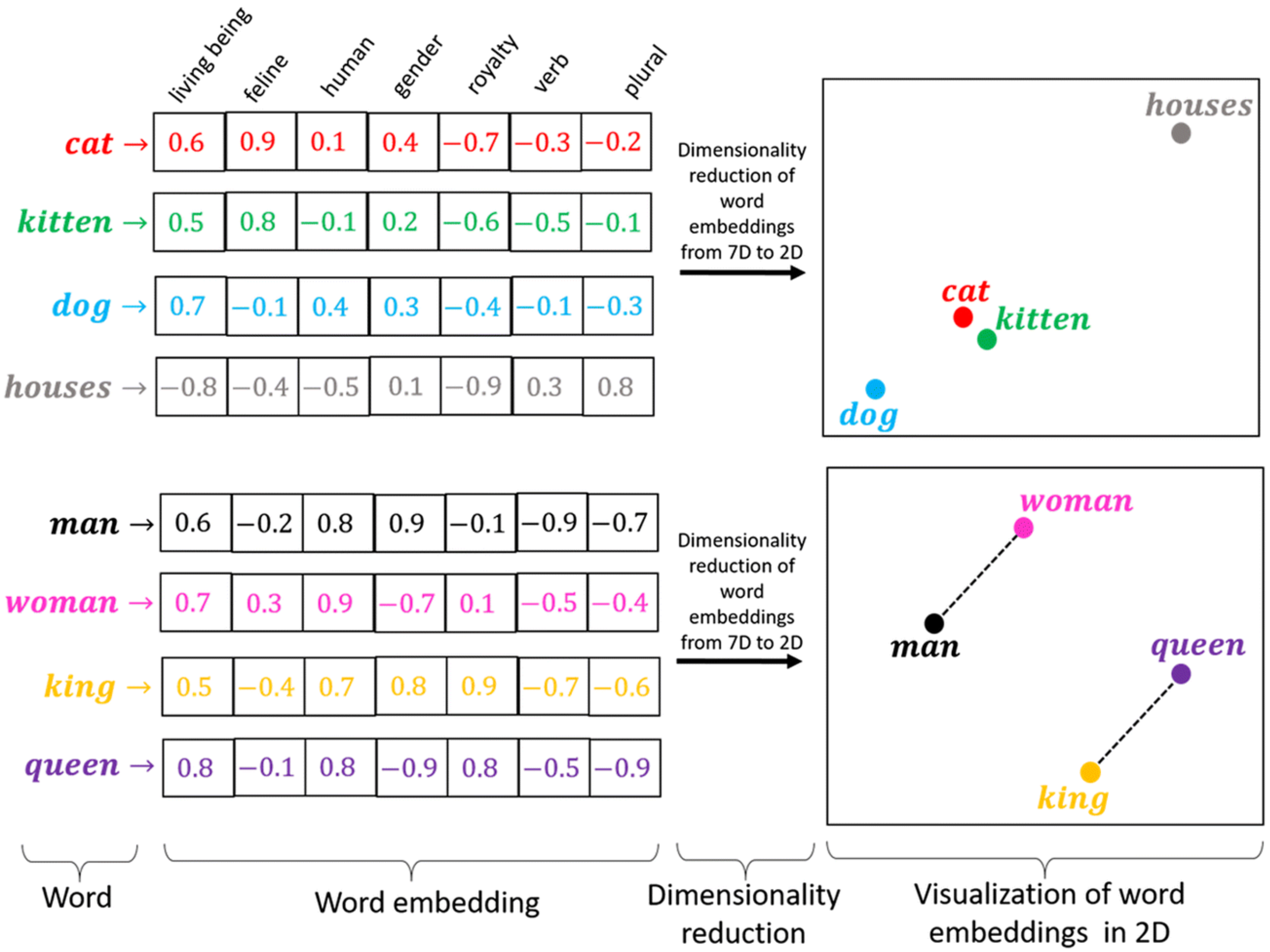
"The film was super"

NER

"Thomas is a dentist"

"Lisa is an optalmologist"

Word Embeddings – Visualisation



	living being	feline	human	gender	royalty	verb	plural
cat →	0.6	0.9	0.1	0.4	−0.7	−0.3	−0.2
kitten →	0.5	0.8	−0.1	0.2	−0.6	−0.5	−0.1
dog →	0.7	−0.1	0.4	0.3	−0.4	−0.1	−0.3
houses →	−0.8	−0.4	−0.5	0.1	−0.9	0.3	0.8

man →	0.6	−0.2	0.8	0.9	−0.1	−0.9	−0.7
woman →	0.7	0.3	0.9	−0.7	0.1	−0.5	−0.4
king →	0.5	−0.4	0.7	0.8	0.9	−0.7	−0.6
queen →	0.8	−0.1	0.8	−0.9	0.8	−0.5	−0.9

Dimensionality reduction of word embeddings from 7D to 2D
houses
cat
kitten
dog
Dimensionality reduction of word embeddings from 7D to 2D
woman
man
queen
king
Word
Word embedding
Dimensionality reduction
Visualization of word embeddings in 2D

t-sne algorithm to plot high-dimensional data in 2 or 3D

Word embeddings fulfill the conditions such as:

$\text{Emb}_{\text{king}}=\text{argmax}_{x\in\text{emb}}(x, \text{emb}_{\text{man}} - \text{emb}_{\text{woman}} + \text{emb}_{\text{king}}$

Transfer learning:

- Using pre-trained word vectors on huge datasets as an embedding layer
- e.g. Word2Vec, GLoVe, ELMo
- Generalization regarding words that possibly did not appear in the small training data set

Bildquelle (23.10.2022): https://medium.com/@hari4om/word-embedding-d816f643140

Lerning Word Embeddings

Skip-grams

I	cook	Lasagna	with	fresh	tomatos.
e_{3849}	e_{4123}	e_{4476}	e_{5134}	e_{2490}	e_{8123}

Sample context/target pairs:

context	target
Lasagna	cook
Lasagna	fresh
Lasagna	tomatos

Example: context=„Lasagna“ target=„fresh“
Model:

$$e_{context} \to E \to emb_{context} \to softmax(V) \to \hat{y} \in \mathbb{R}^V$$

$$\text{loss: } \mathcal{L}(\hat{y}, e_{target})$$

Problem: softmax over V classes is computationally very heavy.

From classification over V
to V binary classifications

Negative Sampling

context	Wort	target
Lasagna	cook	1
Lasagna	house	0
Lasagna	queen	0
Lasagna	car	0

neg. samples

$$e_{context} \to E \to emb_{context}$$

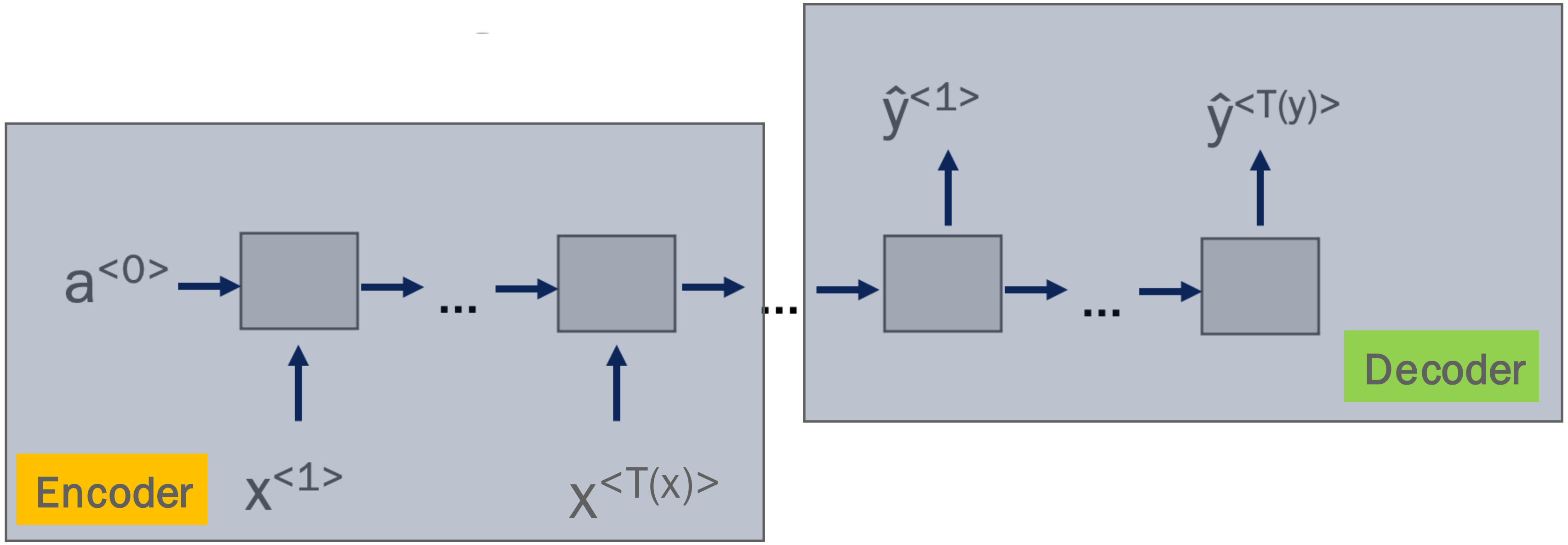
→ Sigmoid("cooks")
→ Sigmoid("house")
...
→ Sigmoid("car")

Sequence to Sequence Models

Input: ”Jana visits her parents in Aachen.”

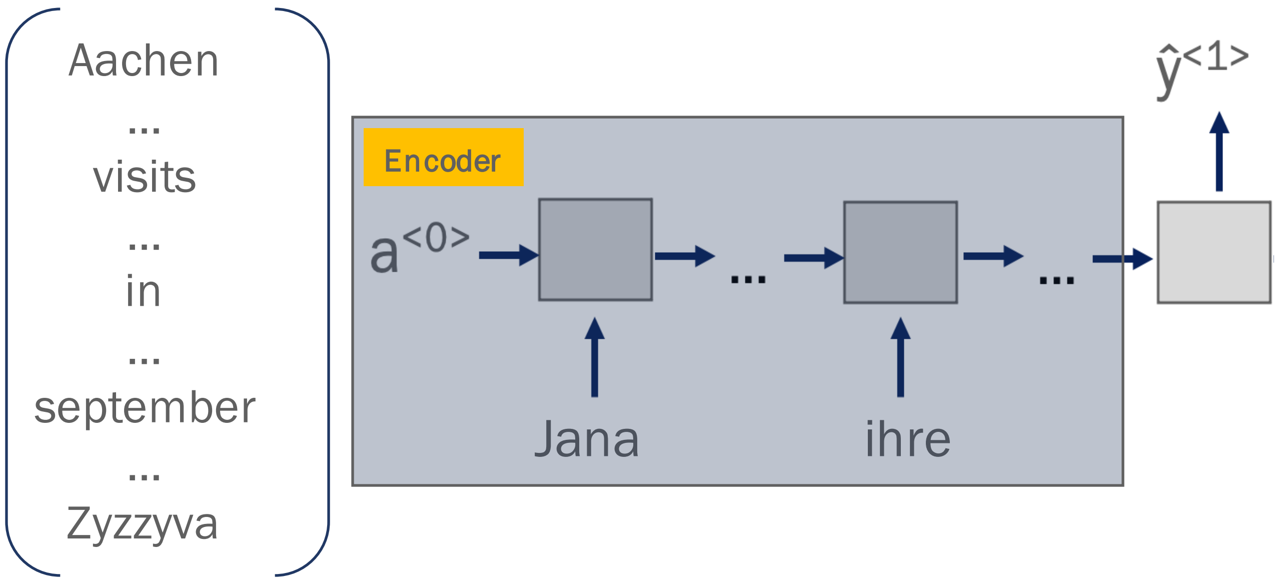
↕

Output: ”Jana besucht ihre Eltern in Aachen.”



Beam Search (B=3)

1. Step

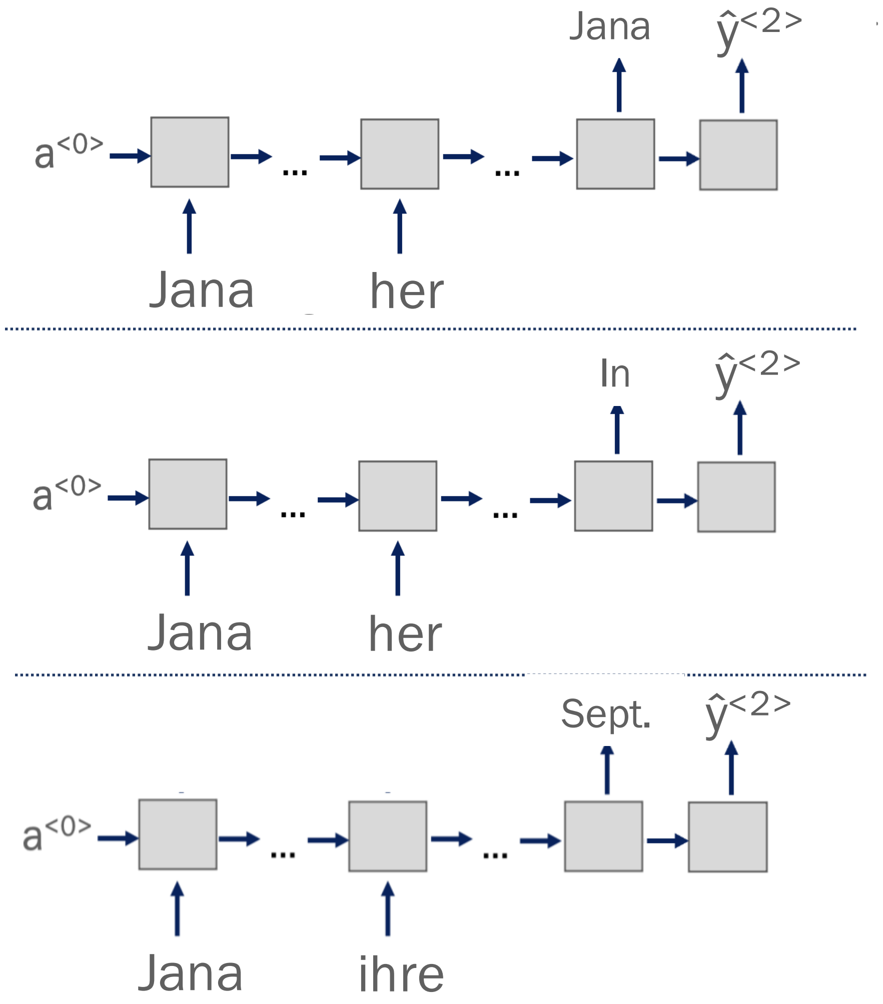


1. Determine conditional probabilities of all vocabulary words $p(\hat{y}^{<1>} \mid x)$
2. Choose **B=3** probability outputs.
3. In this case, e.g. Jana, in, September

Beam Search (B=3)

2. Step

$$\begin{pmatrix} \text{Aachen} \\ \dots \\ \text{visits} \\ \dots \\ \text{in} \\ \dots \\ \text{september} \\ \dots \\ \text{Zyzzyva} \end{pmatrix}$$



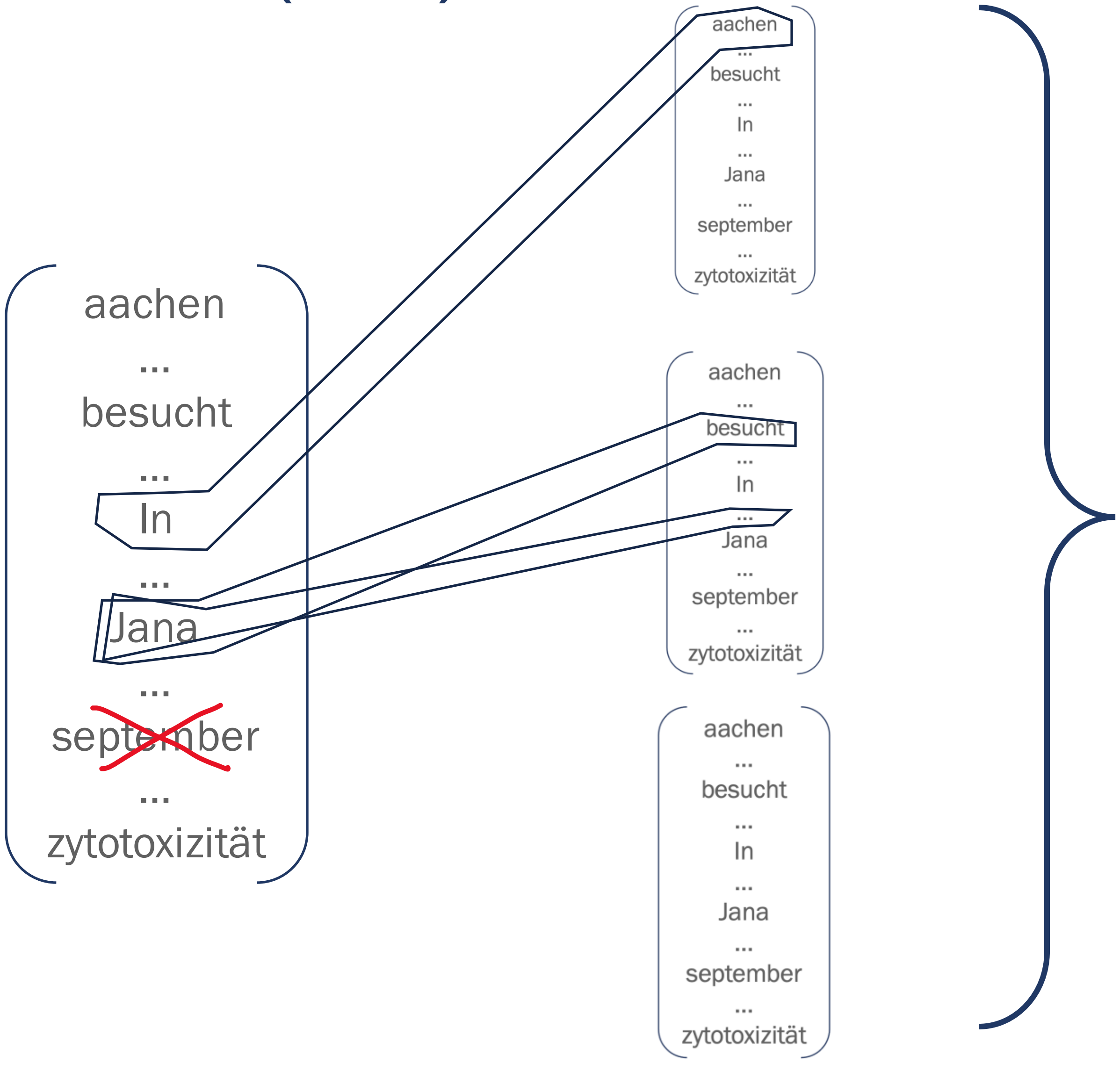
1. Determine conditional probabilities in each case
   $p(\,\hat{y}^{<2>} \mid x, \hat{y}^{<1>} = Jana),$
   $p(\,\hat{y}^{<2>} \mid x, \hat{y}^{<1>} = In),$
   $p(\,\hat{y}^{<2>} \mid x, \hat{y}^{<1>} = September)$

2. Calculate
   $p(\,\hat{y}^{<1>},\, \hat{y}^{<2>} \mid x)$
   $= p(\,\hat{y}^{<1>} \mid x) \cdot p(\,\hat{y}^{<2>} \mid x, \hat{y}^{<1>})$
   and choose $(\hat{y}^{<1>},\, \hat{y}^{<2>})$
   with the highest probability

3. We are again given 3 options with which we can continue in the next step

Beam Search (B=3)

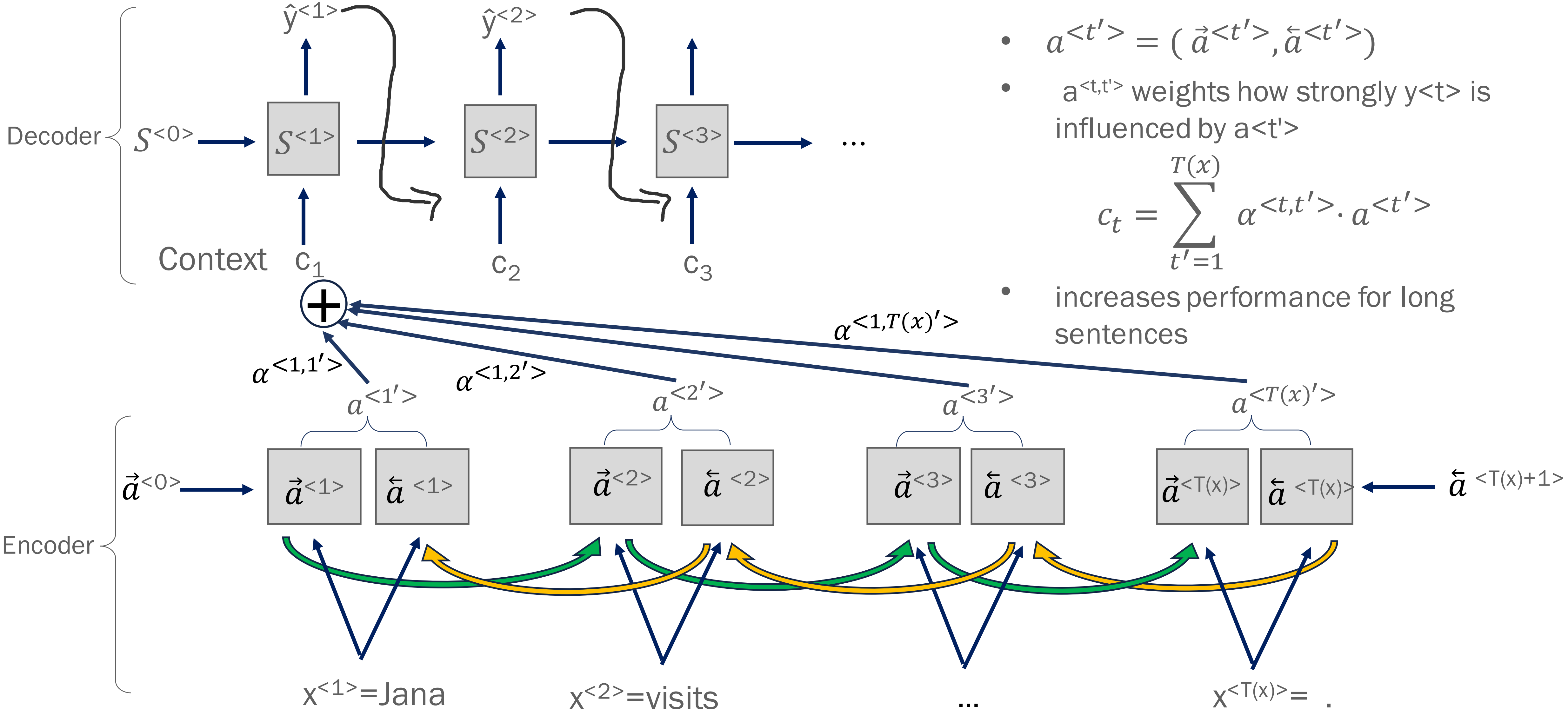
2. Step



In this way we iteratively obtain the conditional probability we are looking for

$$p(\, \hat{y}^{<1>},\ \hat{y}^{<2>}, \dots ,\ \hat{y}^{<T(y)>} \mid x)$$

Attention Models – which Inputs are relevant?

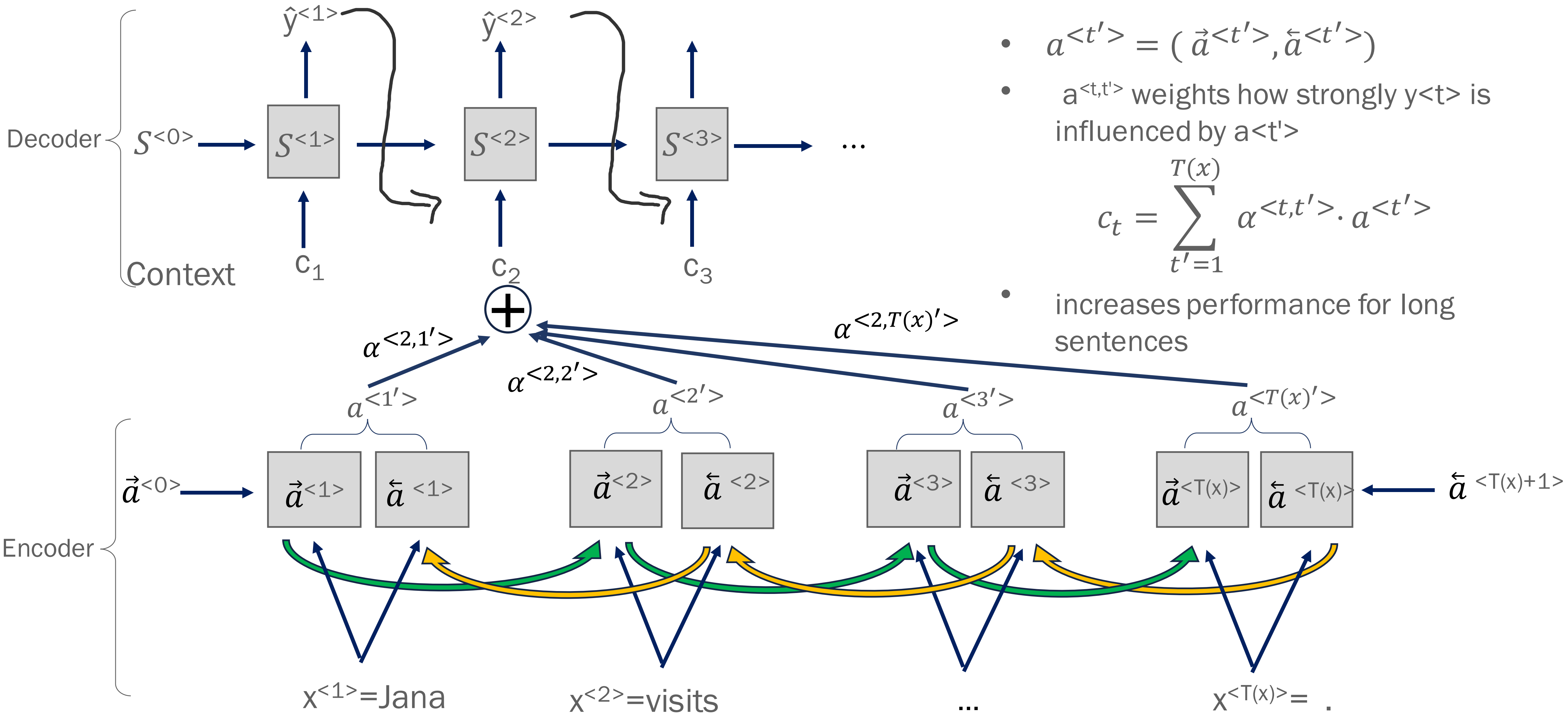


- $a^{<t'>} = (\vec{a}^{<t'>}, \overleftarrow{a}^{<t'>})$
- $a^{<t,t'>}$ weights how strongly y<t> is influenced by a<t'>

$$c_t = \sum_{t'=1}^{T(x)} \alpha^{<t,t'>} \cdot a^{<t'>}$$

- increases performance for long sentences

Attention Models – which Inputs are relevant?



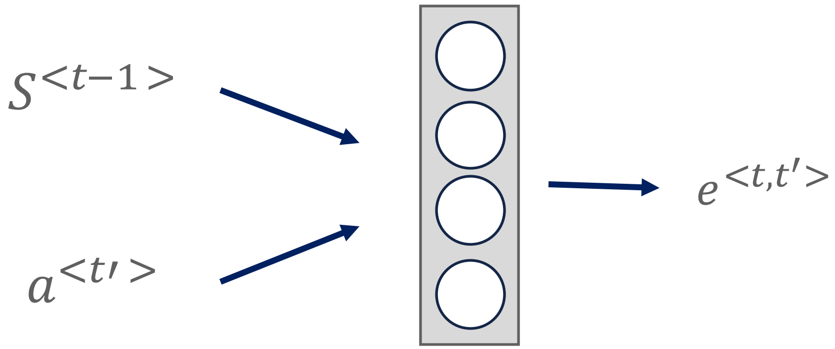
- $a^{<t'>} = (\vec{a}^{<t'>}, \overleftarrow{a}^{<t'>})$
- $a^{<t,t'>}$ weights how strongly y<t> is influenced by a<t'>

$$c_t = \sum_{t'=1}^{T(x)} \alpha^{<t,t'>} \cdot a^{<t'>}$$

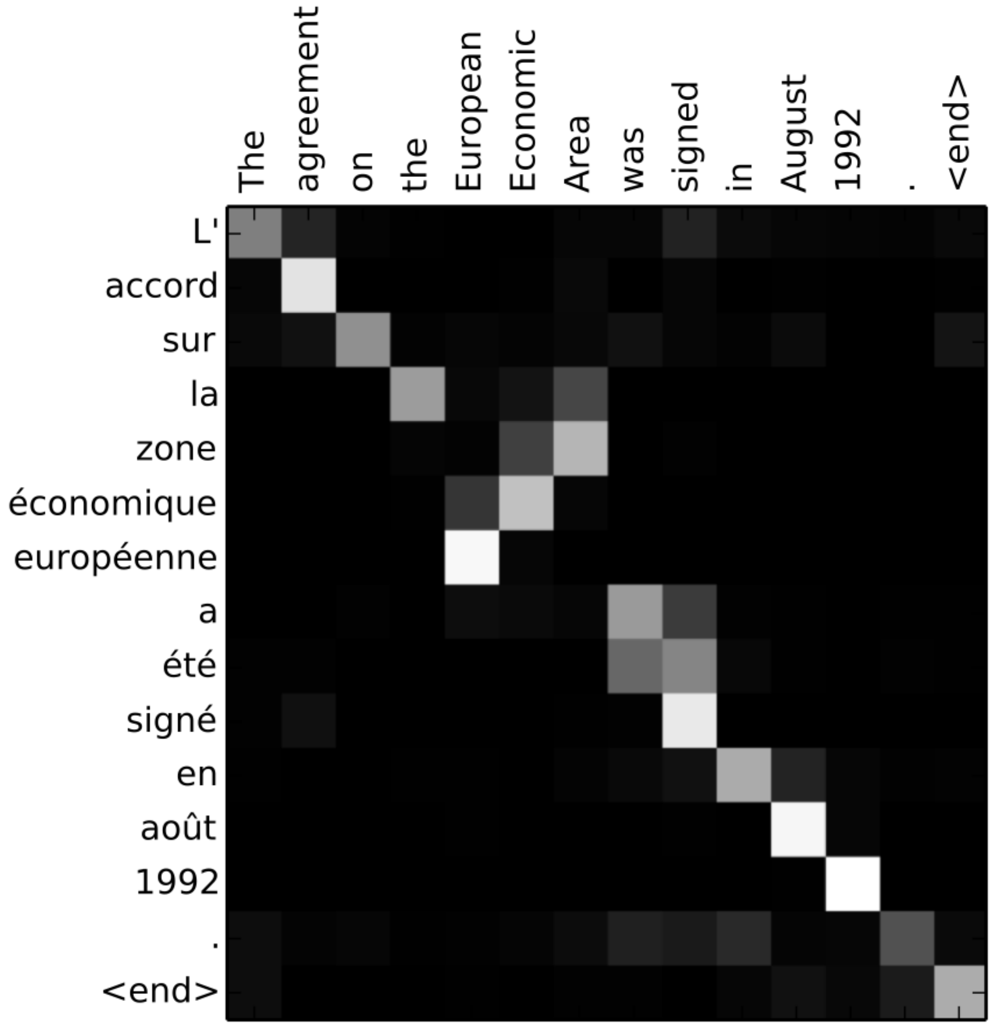
- increases performance for long sentences

Attention Modelle: Berechnung der $\alpha^{<t,t'>}$

$$\alpha^{<t,t'>} = \frac{\exp\left(e^{<t,t'>}\right)}{\sum_{t'=1}^{T(x)} \exp\left(e^{<t,t'>}\right)}$$ (Softmax)



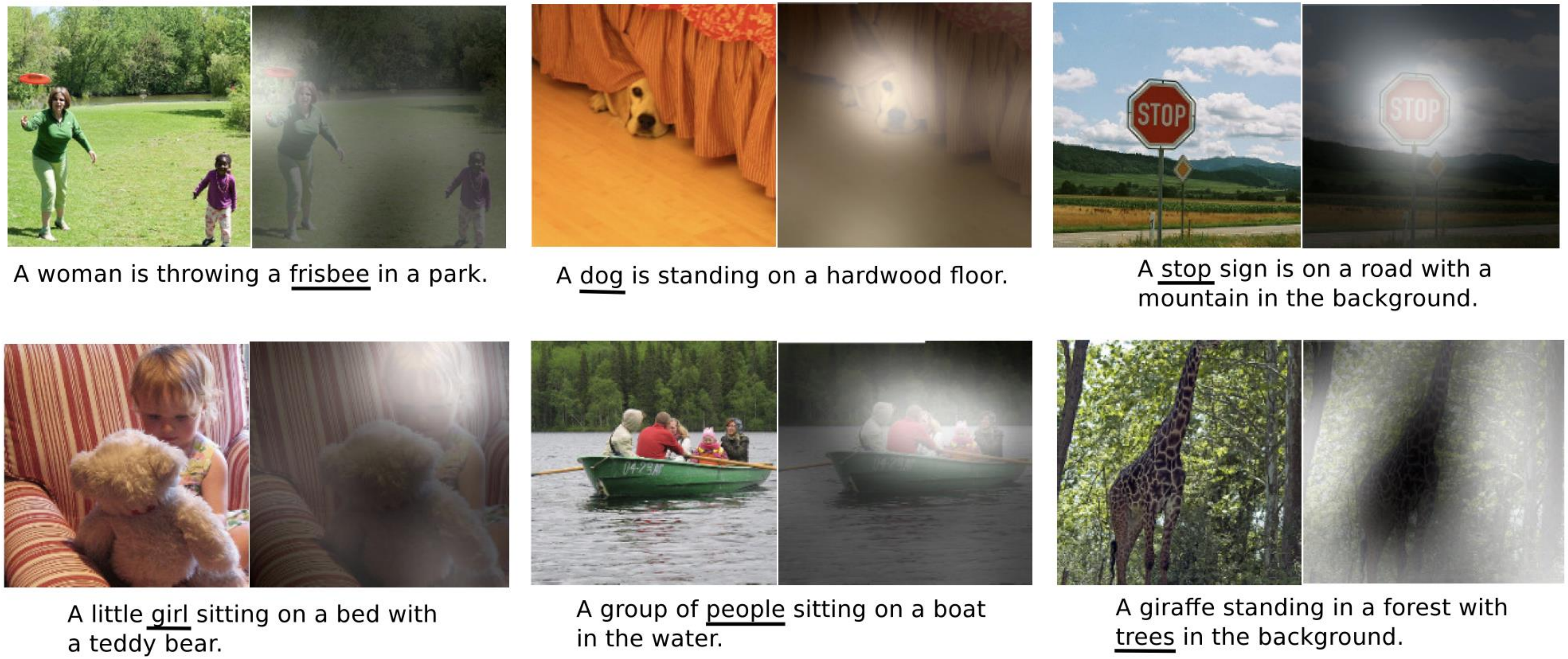
→ α's and thus the context depend on the activations of the encoder as well as the activations/hidden states of the decoder at the previous time step



Bildquelle: https://arxiv.org/pdf/1409.0473.pdf

Attention for Image Captioning

Figure 3. Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)



A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

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

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

A giraffe standing in a forest with trees in the background.

Bildquelle:https://arxiv.org/pdf/1502.03044.pdf

Agenda

- Einleitung
- Theorie der RNNs am Beispiel von NLP
- Programm