''

Sequence-to-Sequence-Learning

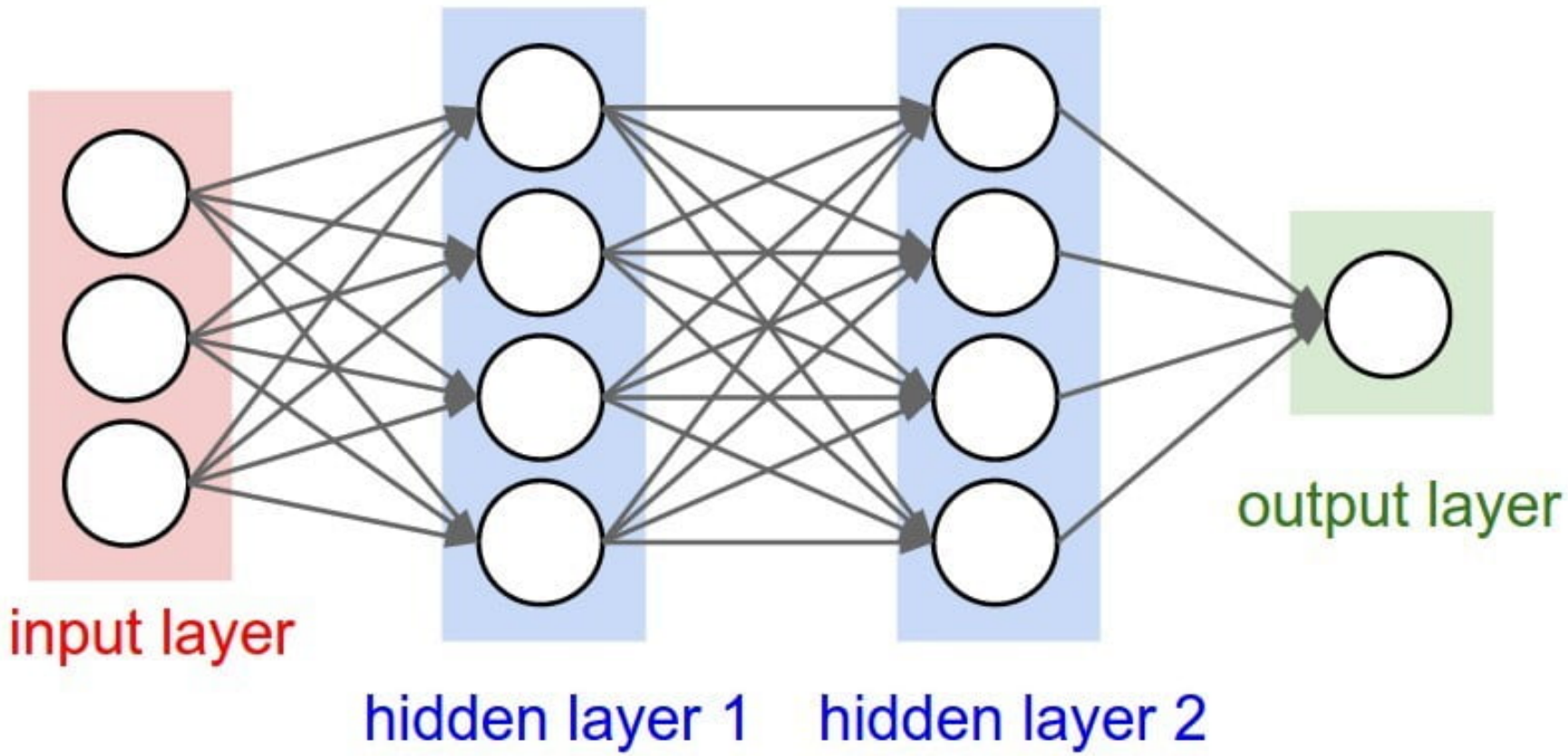
Jens Neuerburg

PS Tools für Computerlinguisten

02.02.2018

Inhalt

Letzte Woche...

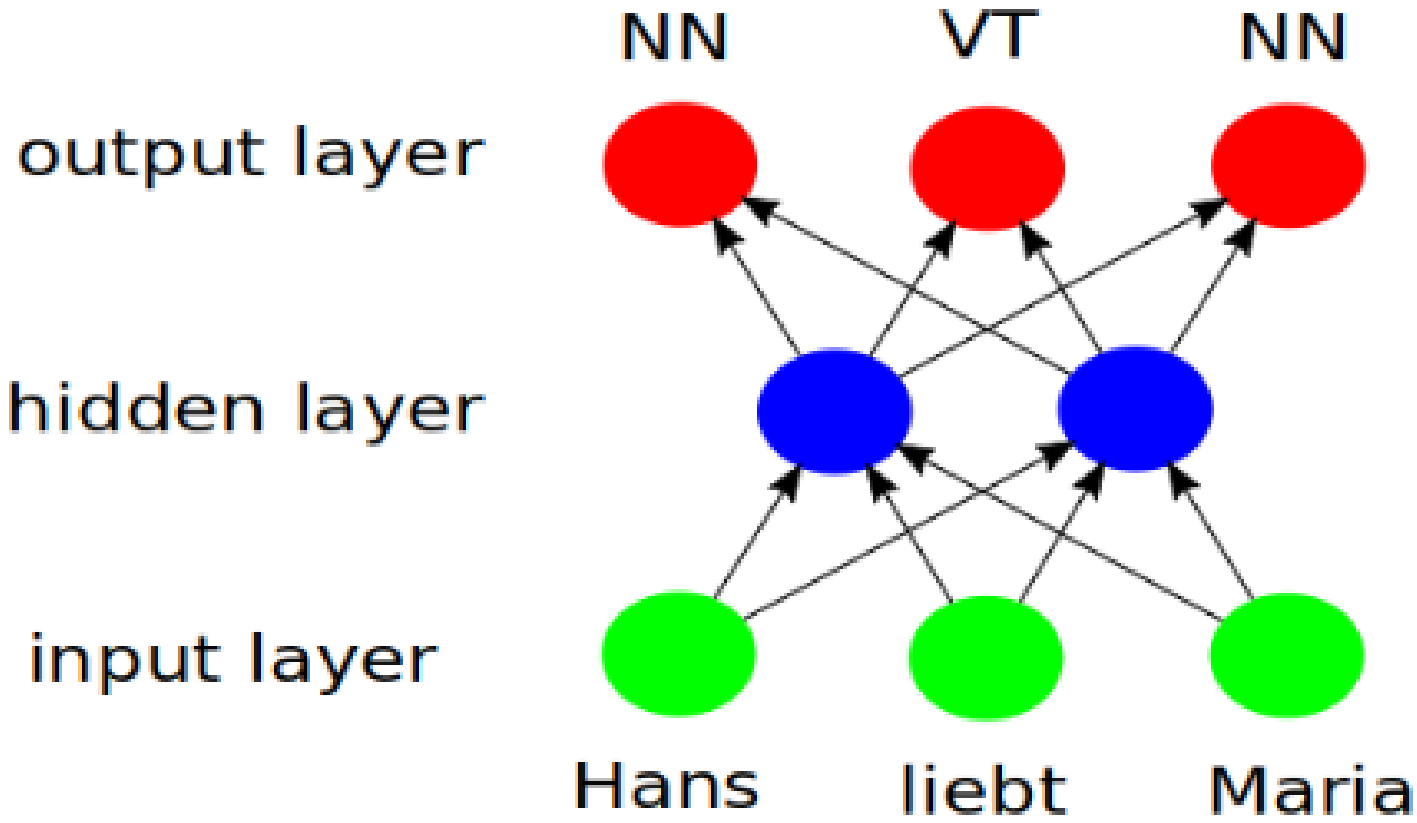


Neuronale Netze in der Computerlinguistik

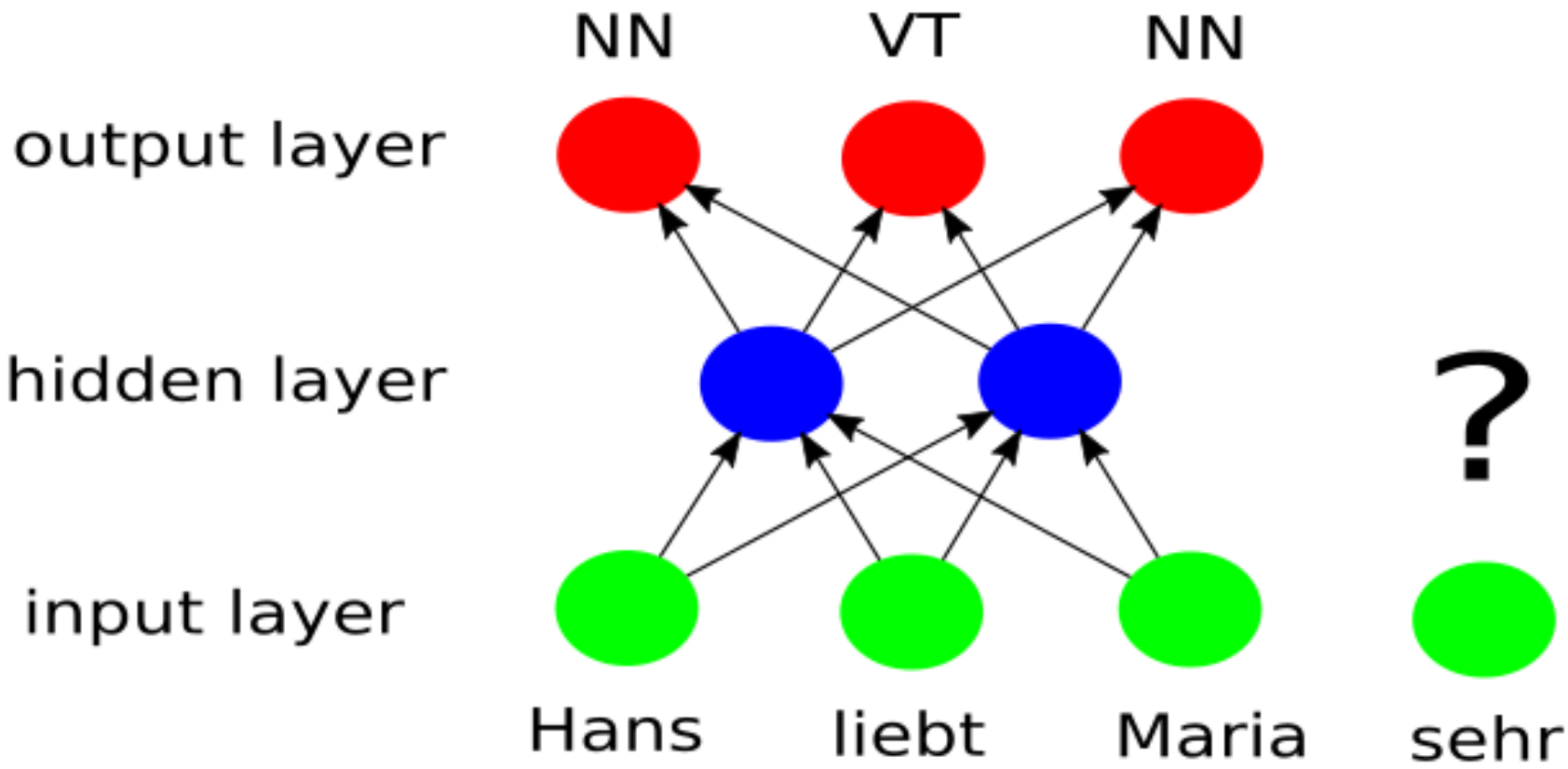
- POS-Tagging
- Spracherkennung
- Maschinelle Übersetzung
- Parsing
- Sentiment Analysis
- ...

...aber können neuronale Netze eigentlich alles?

Beispiel: POS-Tagging mit NN



Beispiel: POS-Tagging mit NN



Beispiel Sentiment Analysis

(Fiktive) Restaurantkritik:
„Mein Steak war **zäh wie Leder** und mein Bier **hat geschmeckt wie Klostein**, Ambiente war aber **angenehm** und die Musik war **gut**. “

- Gute oder schlechte Kritik?
- wir erkennen Wörter wie **angenehm** oder **gut**, aber nicht den Kontext

Kontexterkennung?

Tell Me This 20 hours ago (edited)

Human: What do we want!?
Computer: Natural language processing!
Human: When do we want it!?
Computer: When do we want what?

Reply • 203

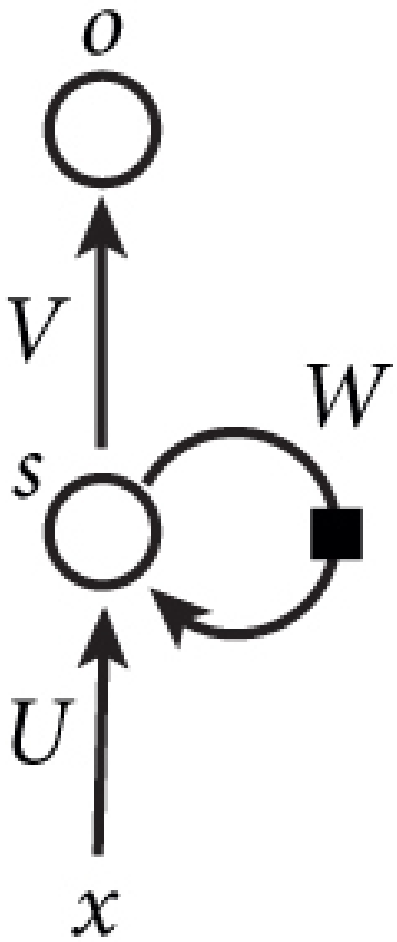
View reply

Was hätten wir gerne?

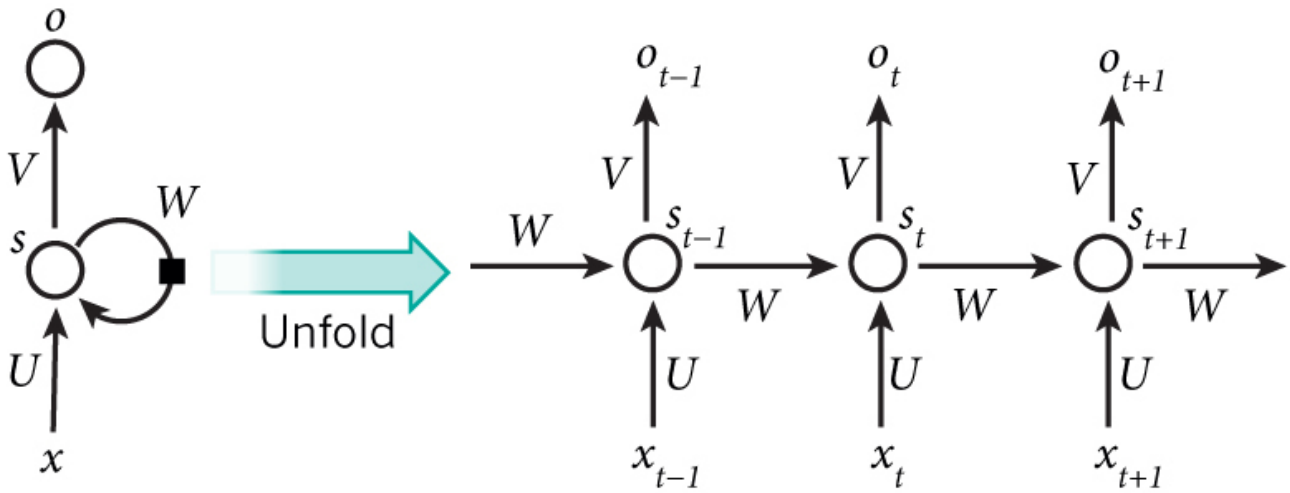
- Ein Netzwerk, das mit unterschiedlich langen Inputs umgehen kann
- Ein Netzwerk, das Kontexte im Input erkennen kann

Die Antwort: Rekurrente Neuronale Netze

Vereinfacht: Neuronale Netze mit Schleifen



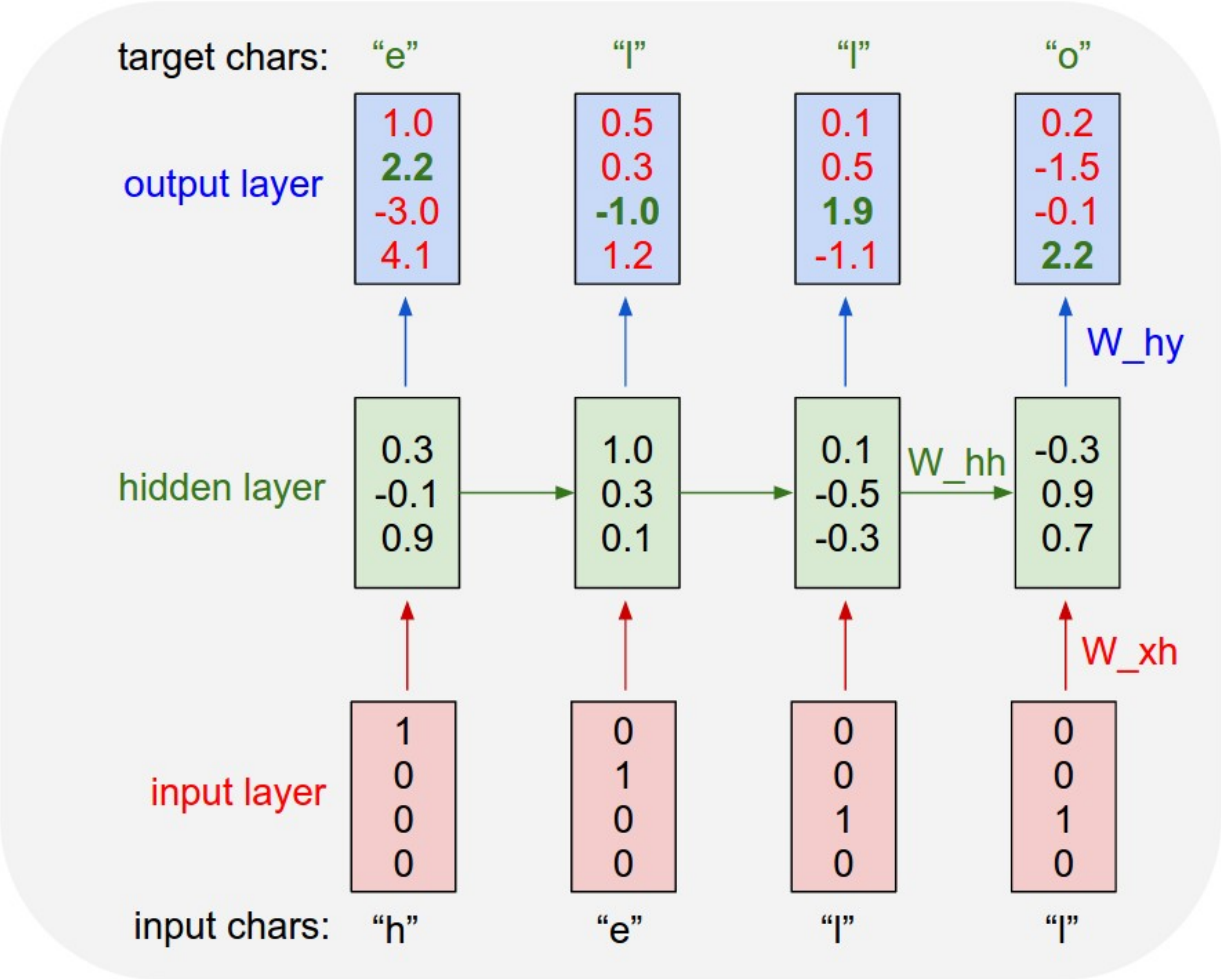
Rekurrente Neuronale Netze



$$s_t = f(Ux_t + Ws_{t-1}) \tag{1}$$

$$o_t = softmax(Vs_t) \tag{2}$$

Unter der Haube: Beispiel Textgenerierung



Ausprobieren: Textgenerator

Ähnlich wie letzte Woche, benutzt Tensorflow statt PyTorch

- In Docker: /tools/charnn
- Training mit Standardeinstellungen:
  python - -data_dir /data/yourdata
- Trainingsdaten: jede beliebige .txt - Datei!
- Textdatei aus dem Web benutzen:
  mkdir data/mycorpus
  wget https://www.meinlink.txt
  mv meintext.txt input.txt
- zum samplen nach dem Training:
  python sample.py - -save_dir save
- Parameter anzeigen lassen:
  python train.py - -help

Was haben wir jetzt?

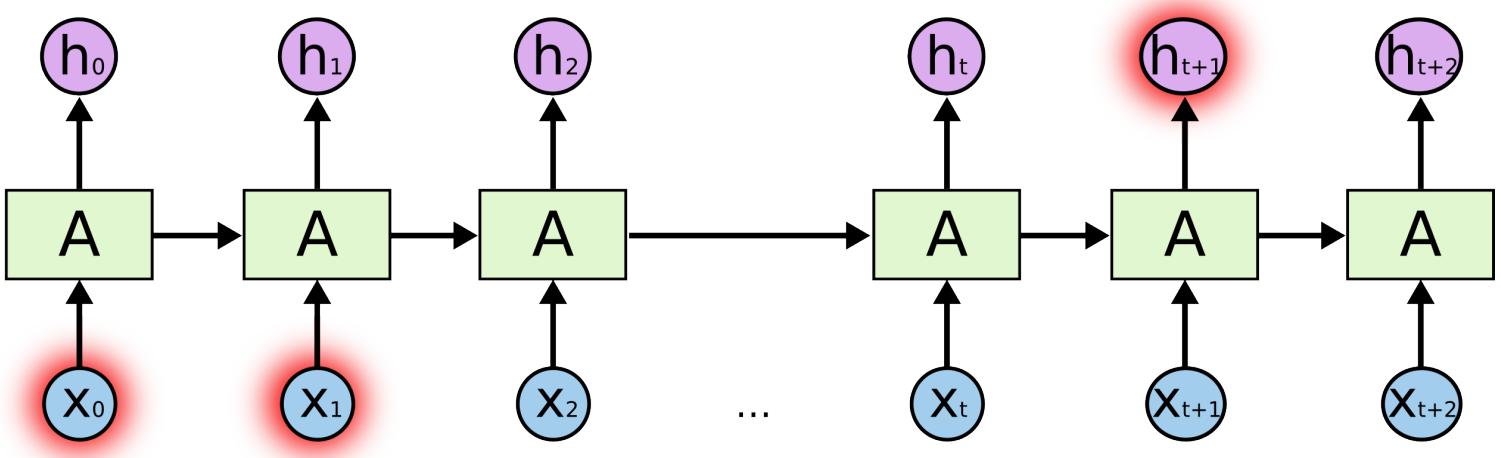
- Ein Netzwerk, dem wir unterschiedlich lange Inputs geben können...
- Ein Netzwerk, das sich Inputs merken kann...

...haben wir jetzt das perfekte neuronale Netz?

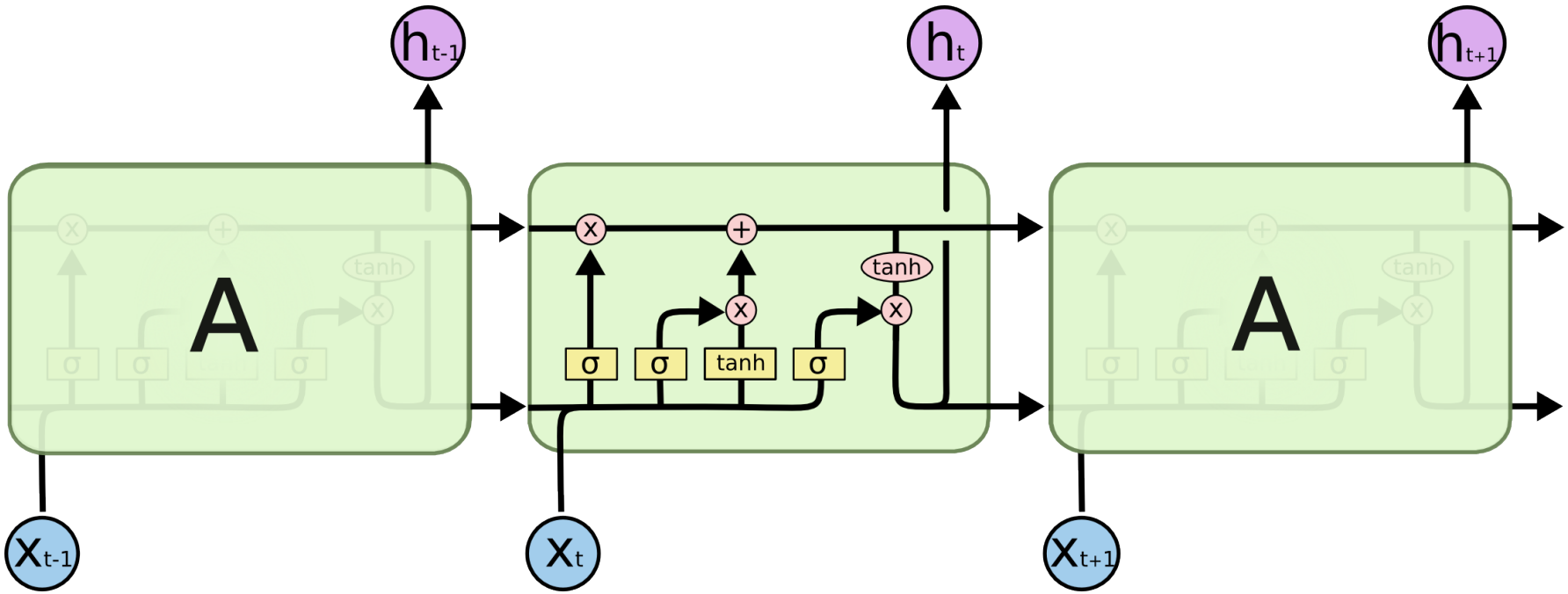
Backpropagation Through Time

Bei der Fehlerrückführung reicht es nicht, wie bei bisherigen Netzen, den Gradienten zum aktuellen Output zu berechnen, sondern wir brauchen den Gradienten von jedem Output zu jedem Zeitschritt...

Vanishing Gradient Problem:



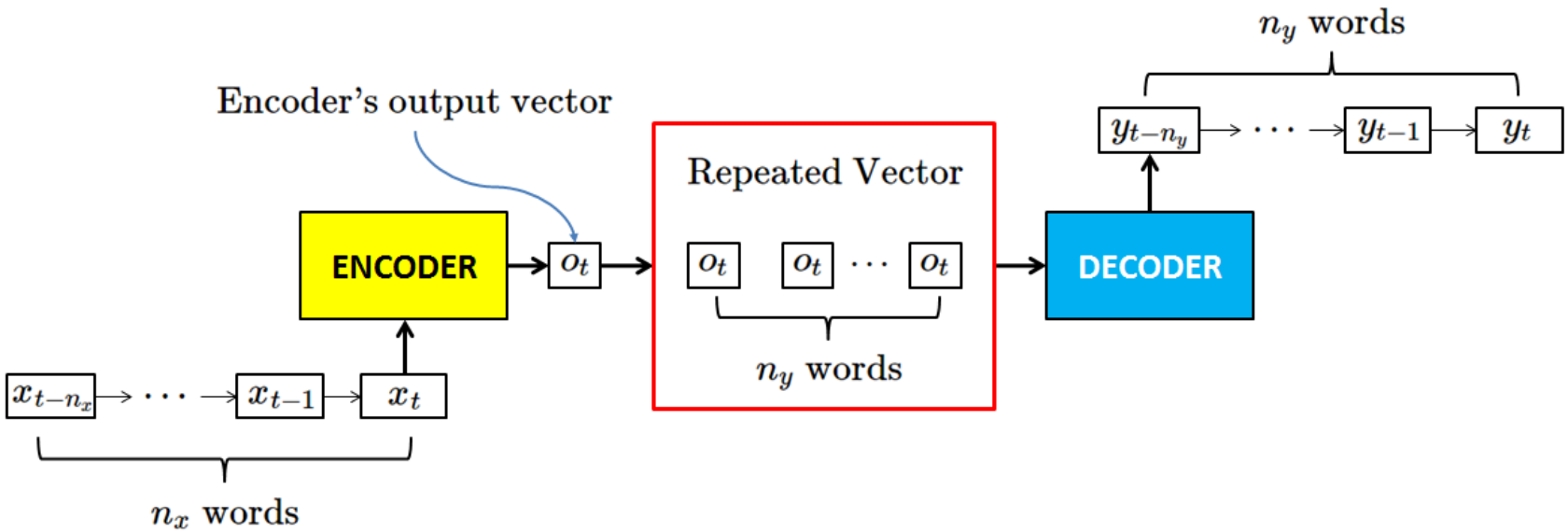
Long Short-Term Memory Networks (LSTMs)



Beispiel: Übersetzung

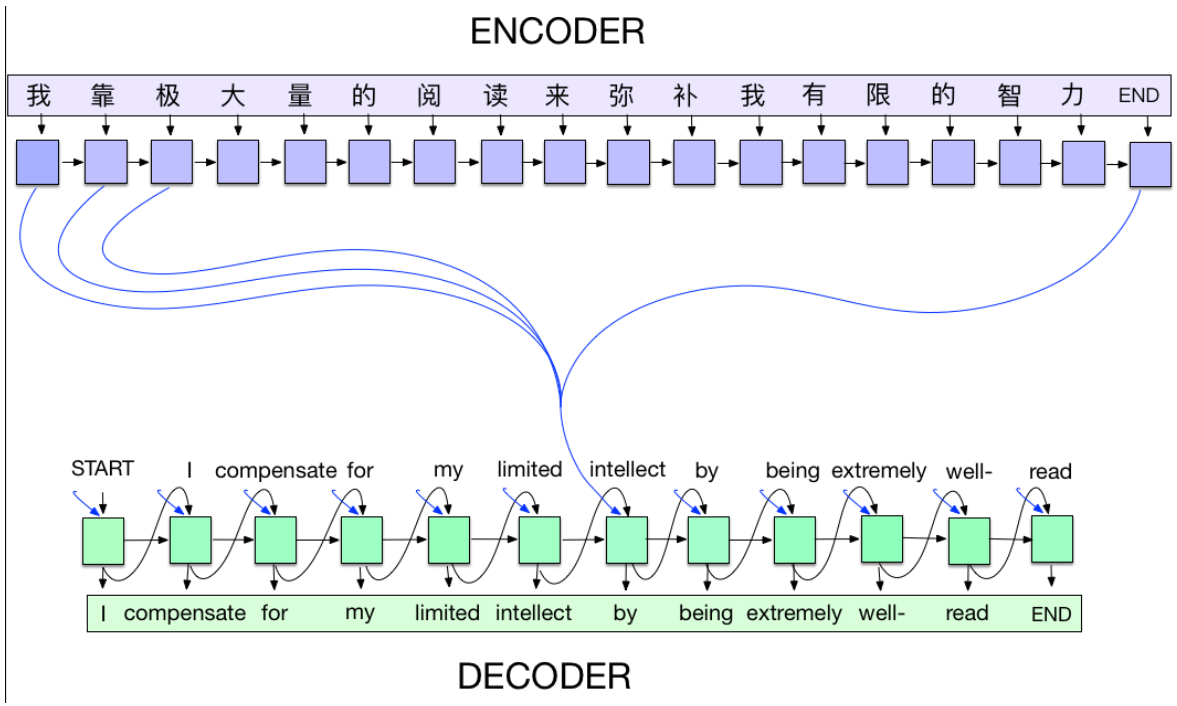
Problem: Unsere Outputs sind nicht gleich lang...
Was können wir tun, damit wir Output erzeugen können, der nicht notwendigerweise gleich lang wie unser Input ist?

Sequence-To-Sequence-Architektur



Tool: Google seq2seq

- Encoder-Decoder framework
- basiert auf Tensorflow
- benutzbar für verschiedene Anwendungen: Übersetzung, Textzusammenfassung...



Übung: google seq2seq

in Docker: tools/seq2seq
Was tun wir damit?

- wir trainieren unseren eigenen „Übersetzer“

Trainingsdaten:

- Paralleler Korpus, d.h. eine Textdatei in der Ausgangssprache, eine Datei in der Zielsprache, mit jeweils einem Satz pro Zeile

Warum eigentlich „Übersetzer“?

Übung: „Maschinelle Übersetzung“

Übung: „Maschinelle Übersetzung“

Testdatensatz generieren:

DATA_TYPE=reverse ./bin/data/toy.sh

Environment-Variablen setzen:

```
export VOCAB_SOURCE=${HOME}/nmt_data/toy_reverse/train/vocab.sources.txt
export VOCAB_TARGET=${HOME}/nmt_data/toy_reverse/train/vocab.targets.txt
export TRAIN_SOURCES=${HOME}/nmt_data/toy_reverse/train/sources.txt
export TRAIN_TARGETS=${HOME}/nmt_data/toy_reverse/train/targets.txt
export DEV_SOURCES=${HOME}/nmt_data/toy_reverse/dev/sources.txt
export DEV_TARGETS=${HOME}/nmt_data/toy_reverse/dev/targets.txt

export DEV_TARGETS_REF=${HOME}/nmt_data/toy_reverse/dev/targets.txt
export TRAIN_STEPS=1000
```

Training

```
export MODEL_DIR=${TMPDIR:-/tmp}/nmt_tutorial
mkdir -p $MODEL_DIR

python -m bin.train \
  --config_paths="
      ./example_configs/nmt_small.yml,
      ./example_configs/train_seq2seq.yml,
      ./example_configs/text_metrics_bpe.yml" \
  --model_params "
      vocab_source: $VOCAB_SOURCE
      vocab_target: $VOCAB_TARGET" \
  --input_pipeline_train "
    class: ParallelTextInputPipeline
    params:
      source_files:
        - $TRAIN_SOURCES
      target_files:
        - $TRAIN_TARGETS" \
  --input_pipeline_dev "
    class: ParallelTextInputPipeline
    params:
       source_files:
        - $DEV_SOURCES
       target_files:
        - $DEV_TARGETS" \
  --batch_size 32 \
  --train_steps $TRAIN_STEPS \
  --output_dir $MODEL_DIR
```

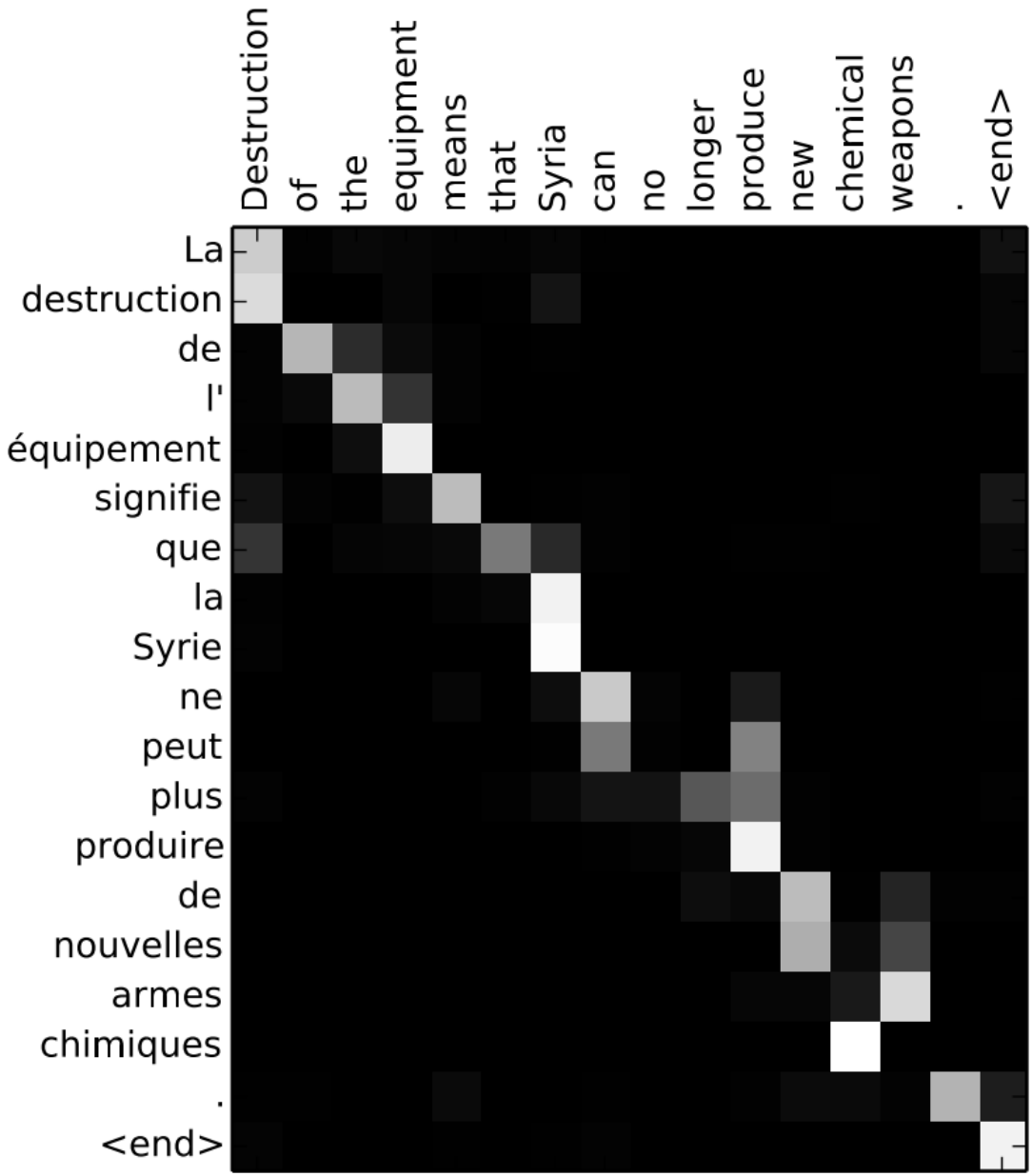
Eine Textdatei „übersetzen“:

```
export PRED_DIR=${MODEL_DIR}/pred
mkdir -p ${PRED_DIR}

python -m bin.infer \
  --tasks "
    - class: DecodeText" \
  --model_dir $MODEL_DIR \
  --input_pipeline "
    class: ParallelTextInputPipeline
    params:
      source_files:
        - $DEV_SOURCES" \
  >  ${PRED_DIR}/predictions.txt
```

zum Testen: eigene Zahlensequenz in sources.txt schreiben

Attention


Destruction
of
the
equipment
means
that
Syria
can
no
longer
produce
new
chemical
weapons
.
<end>
La
destruction
de
l'
équipement
signifie
que
la
Syrie
ne
peut
plus
produire
de
nouvelles
armes
chimiques
.
<end>

Ein echter Shakespeare?

'his bit nong maredet; Lave god in you, weal whot woed!
RONIRUS: I saude, somy nold, Thang! to gie's shanhilffile, Parse my chore, for us us your: And way whoque; a geistlanbit! What whom wouver, feppeanter fally I spack win shall groprees froge'd.
MORCESTER: No, see; how. Bord: In sain many.
MEMENWI: Madlavet in of the had Are and shall diad: Whis a deart, And by?
MAPULO: If I pentile tise thon even, and low With mide: his too?
ROPEN: O. I umaration anty preak. may suse No gound, I heaver'