Deep Learning

Grundlagen Grundbegriffe Grundübungen

Alfatraining Kurs 2025-08 Dozent: Karsten Flügge

Themen des Tages

Tag 14

Sequence-to-Sequence Verfahren, Encoder-Decoder Architektur Übersetzungen

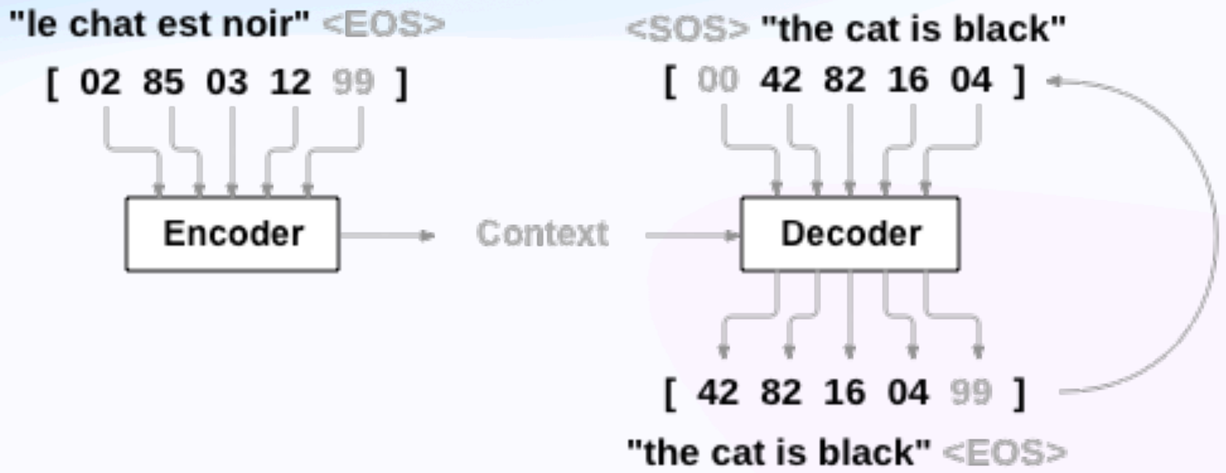
Attention-Schichten,
Transformers

Sequence-to-Sequence

In den früheren Jahren von Maschinen Learning wurden sehr viele manuelle Schritte ausgeführt um die Daten vorzubereiten und nachzubereiten und zwischendrin wurden viele ad-hoc Algorithmen ausgeführt. Dies hat zu teilweise sehr unübersichtlichen Projekten geführt, mit Abhängigkeiten welche auf verschiedenen Systemen schwer erfüllt werden konnten.

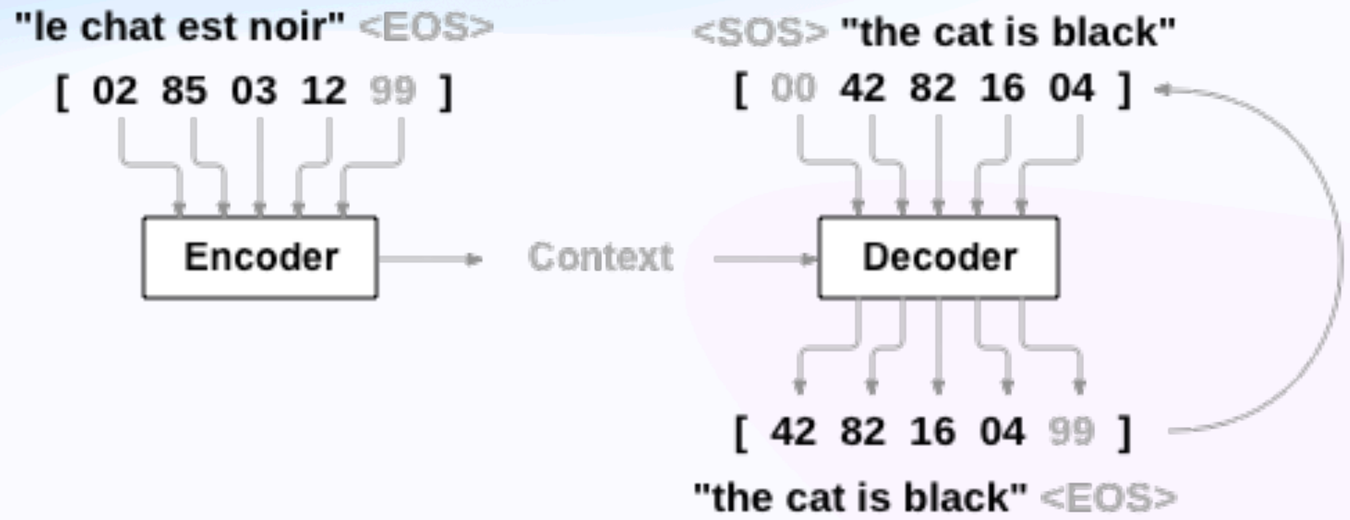
Dank Deep Learning wurde das Versprechen **greifbar**, dass man
beliebige Daten in beliebige Daten umwandeln kann
und die notwendigen Schritte dazwischen vom System automatisch lernen zu lassen.

Recently: OpenAI **voice to voice** chat model. In EINEM Netzwerk.



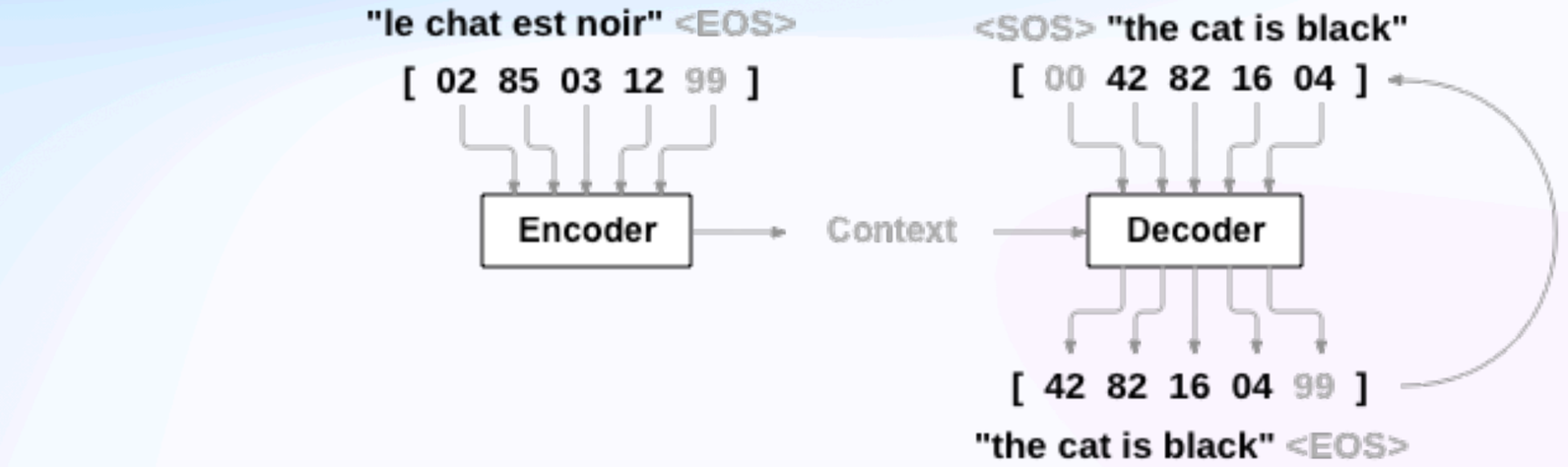
Sequence-to-Sequence

Wie beim **Auto-encoder** trennt man gern in **Encoder** und **Decoder** Komponenten zu besserer Datenerzeugung.
Wie nennt sich die komprimierte Information (Aktivierung des Encoders)?
Feature Vektor … im Latent **Space** (eher allgemeiner)
Context Vektor … im Latent **Space**
Embedding Vektor … im Latent **Space**
Representation Vektor … im Latent **Space**



Sequence-to-Sequence

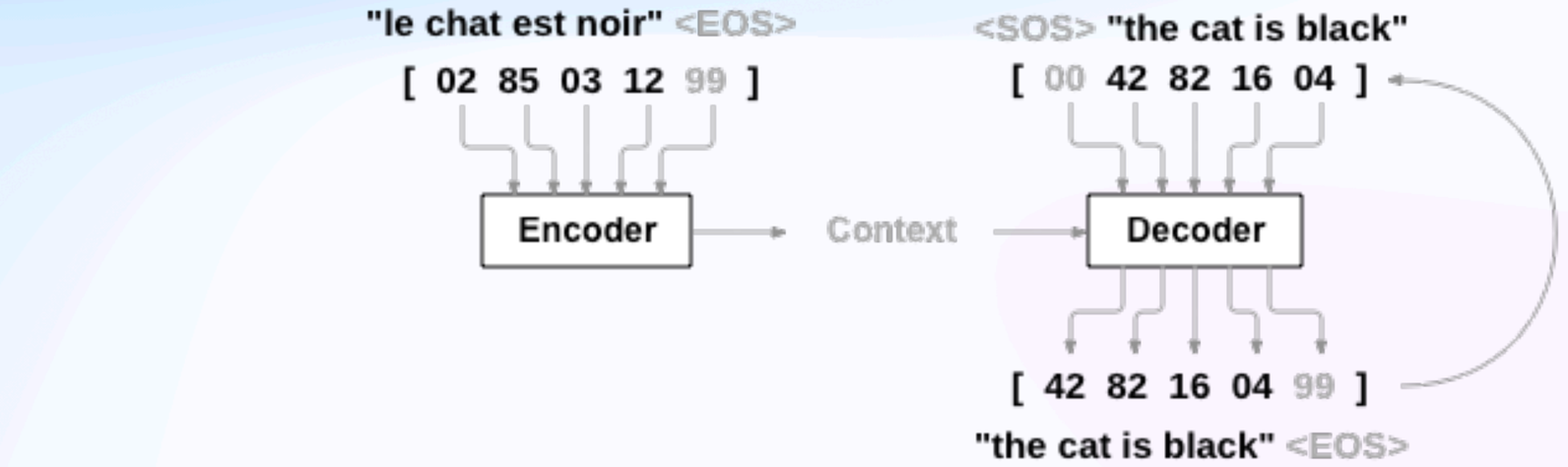
Unsere RNNs sind in der Lage beliebig lange Texte auszuspucken.
Wir können mit einem extra token **EOS** "End of Sequence" mit trainieren, wenn es *fertig* sein soll.



Sequence-to-Sequence

Optional gibt es ein Sondertoken "**SOS**" Start of Sequence, welches dem RNN hilft zu sehen dass es mit einem neuen Satz beginnen soll.

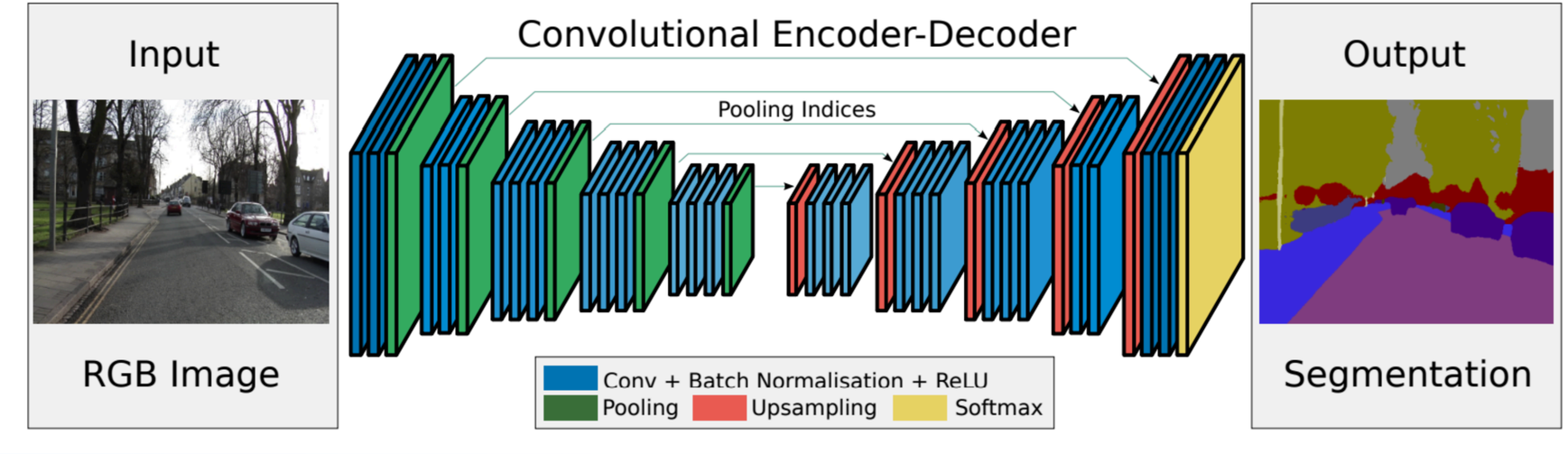
z.B. Erster Buchstabe Groß in Deutschen Sätzen.



Encoder-Decoder Architektur

Umwandeln einer Eingabe in eine Ausgabe von **unterschiedlicher Länge oder Struktur**

Encoder: kodiert Eingabesequenz in eine feste Länge, den **Kontextvektor** oder die **Latent-Representation**
Decoder: dekodiert den Kontextvektor, um eine Ausgabesequenz zu **generieren**
"Auto auf Straße" Text nicht so fein wie Bild, aber sehr ähnlich.
💡**Eingabe und Ausgabe können von ganz unterschiedlicher Natur sein (Text, Bild, Sprache …)!**

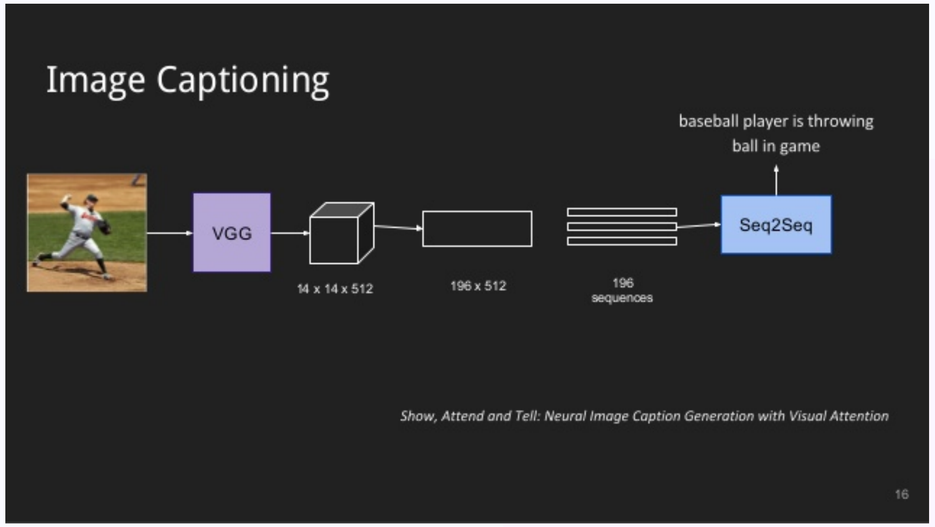


Encoder-Decoder Anwendungen

Beispiele und Anwendungen

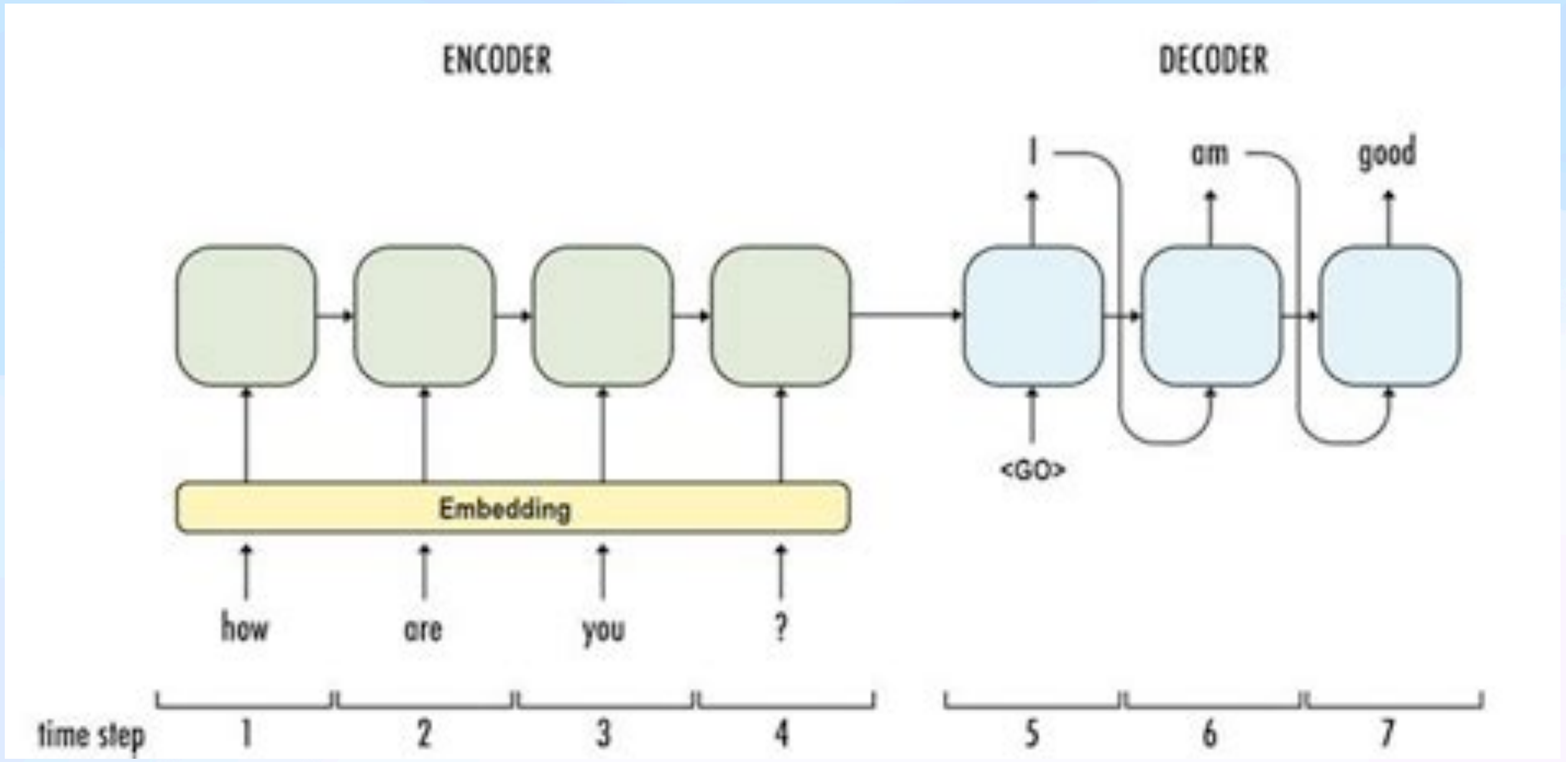
Maschinelle Übersetzung:
von einer Sprache in eine andere.
Bild-Unterschriften:
Textbeschreibung zu einem Bild / Video
Text-zu-Sprache-Konvertierung
Sprache-zu-Text-Konvertierung

Textzusammenfassung
Fragen-Beantwortung
ChatBots



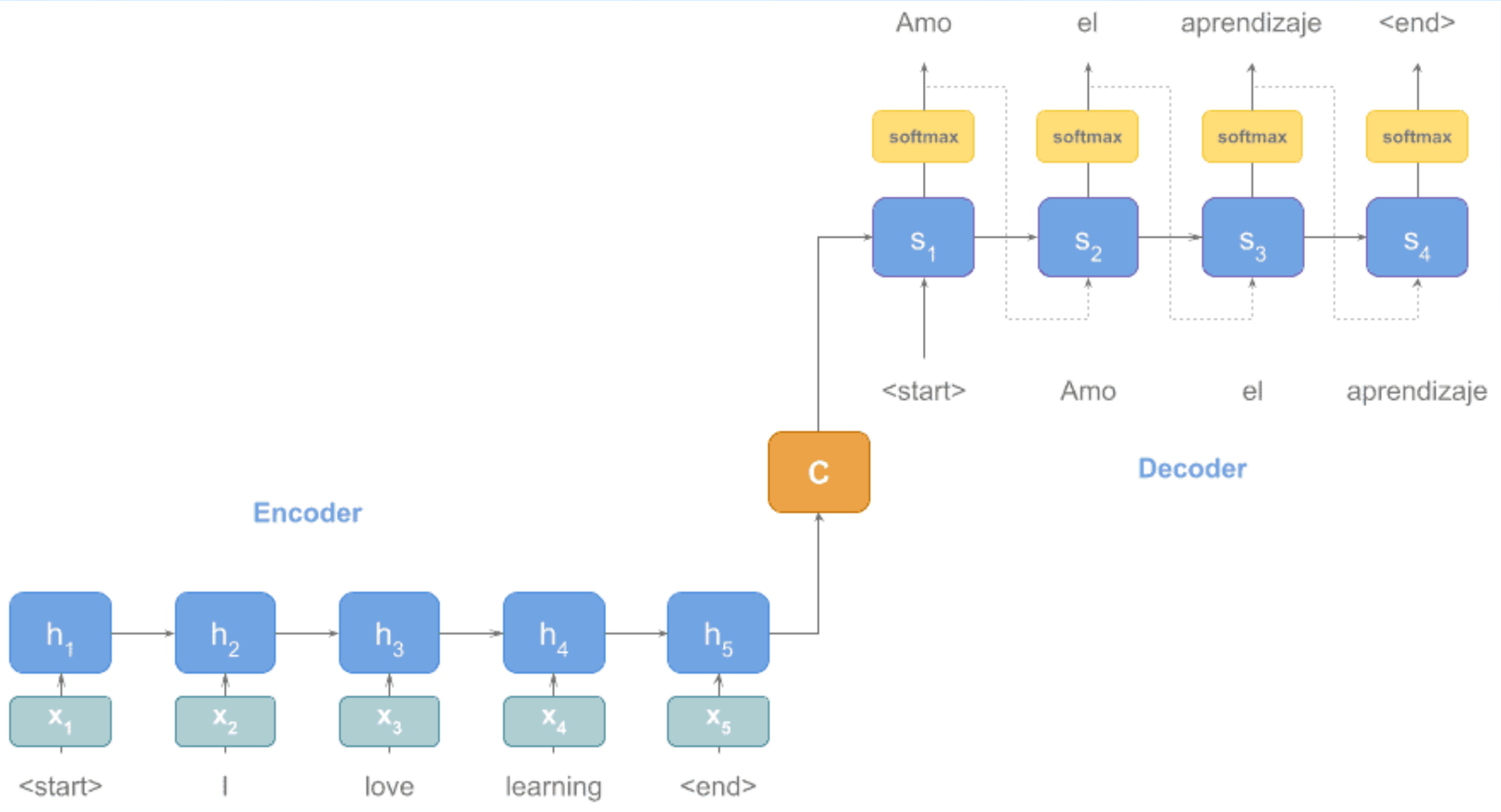
Encoder-Decoder

Erweiterung des Auto-Encoder Prinzips, bei welchem der Output nichtmehr gleich dem input ist. Unterschied zu normalem 'mapping' von Deep Neural Networks?



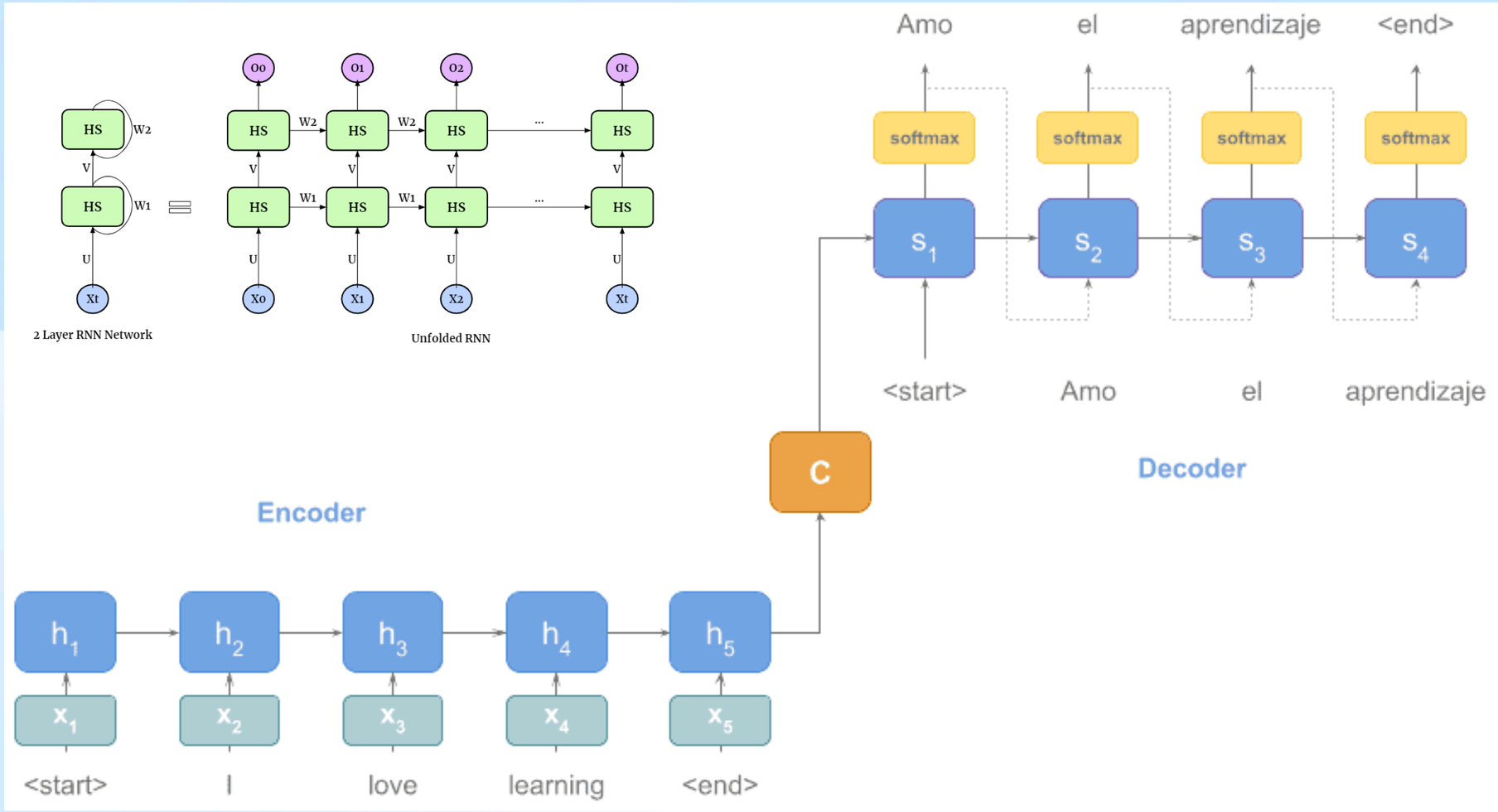
Encoder-Decoder

Ähnlich zum Auto-Encoder ist der **zentrale Layer** ein **bottleneck** welcher alle vorigen Informationen vom Input komprimiert beinhalten soll.
Hier heißt dieser **Embedding** Vector auch **Context C**



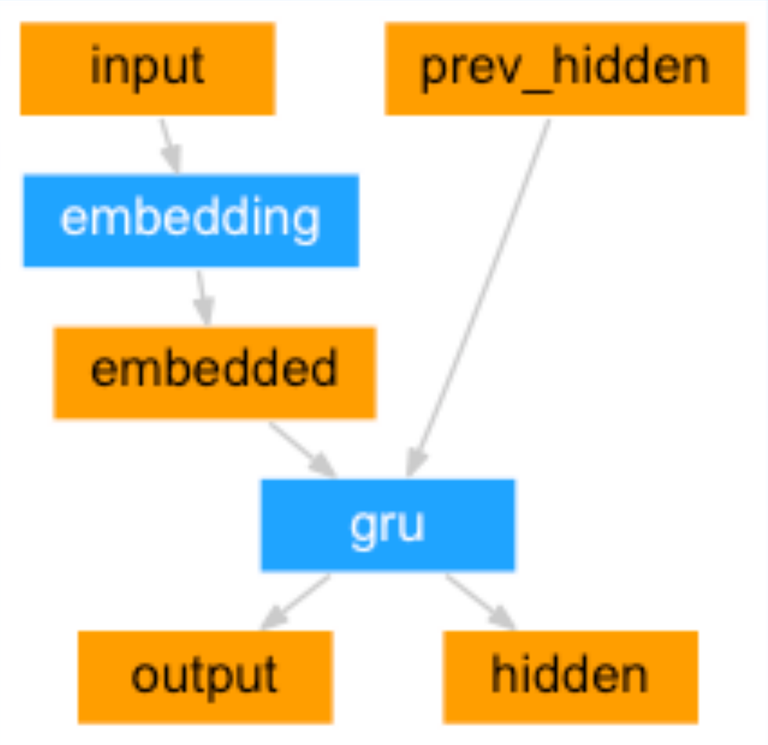
Encoder-Decoder

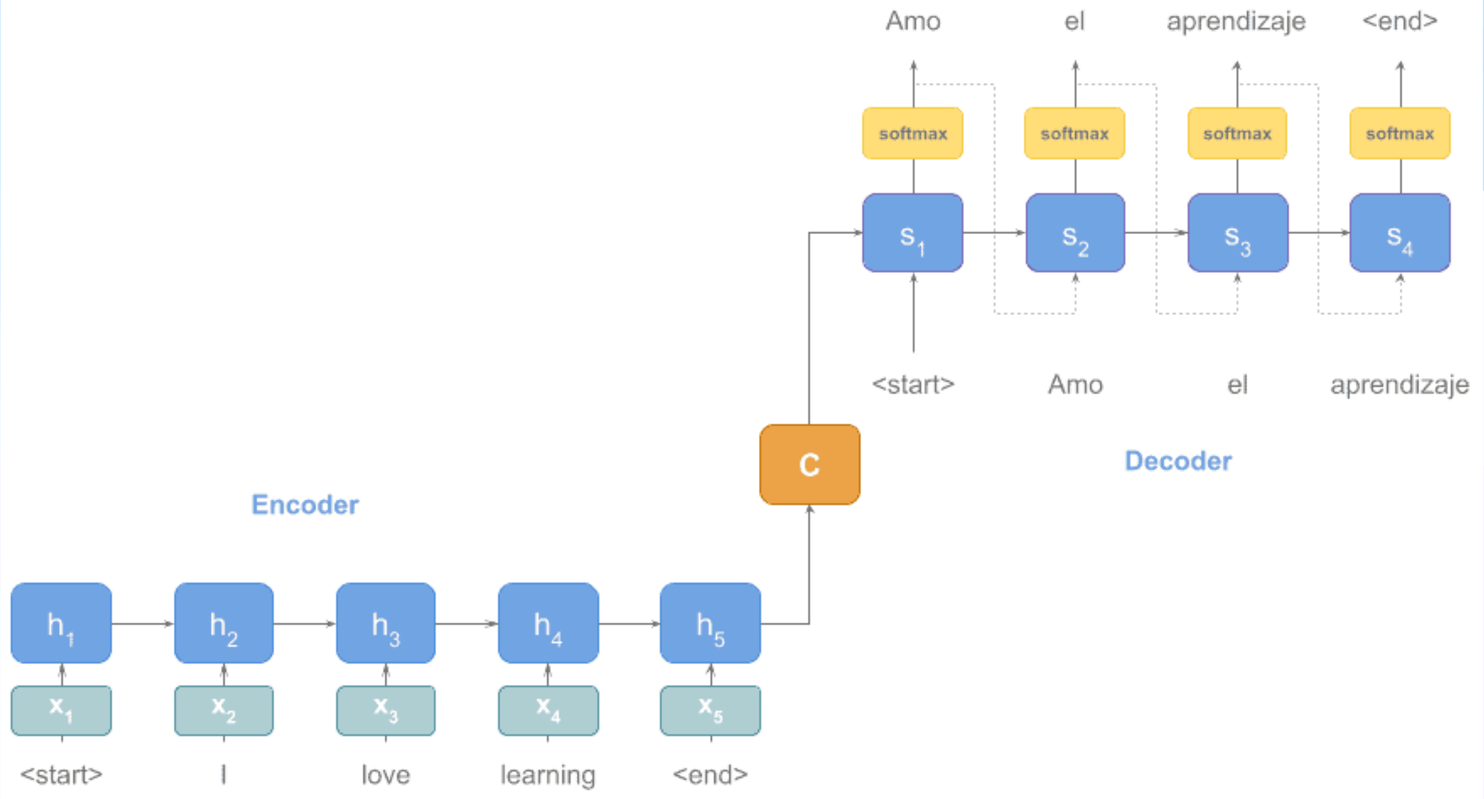
Der **Encoder** ist ein **RNN Block**, dessen **output "context"** an den **Decoder übergeben** wird, also der *letzte hidden state vom Encoder* ist *erster hidden state vom Decoder*:



Encoder-Decoder

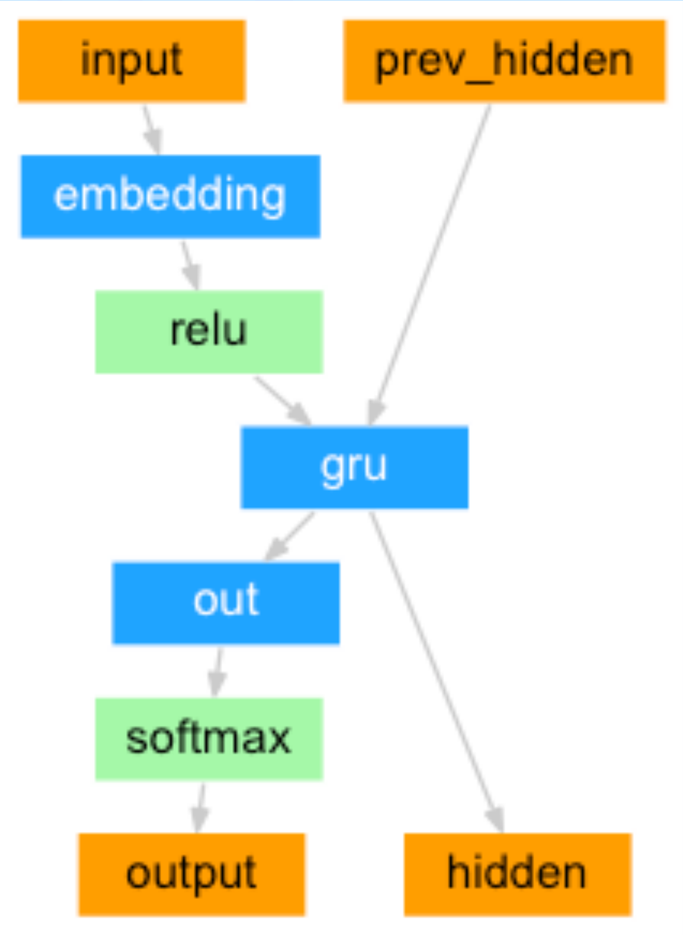
Der **Encoder** baut einen **hidden state** auf welcher alle vorigen Worte '**embedded**'.
Um alle Informationen des ganzen Satzes aufnehmen zu können,
muss der **hidden vector recht groß** sein, weswegen man oft *mehrere RNN(**GRU**) layer stacked*.
(Der output vom RNN wird beim Evaluieren verworfen)

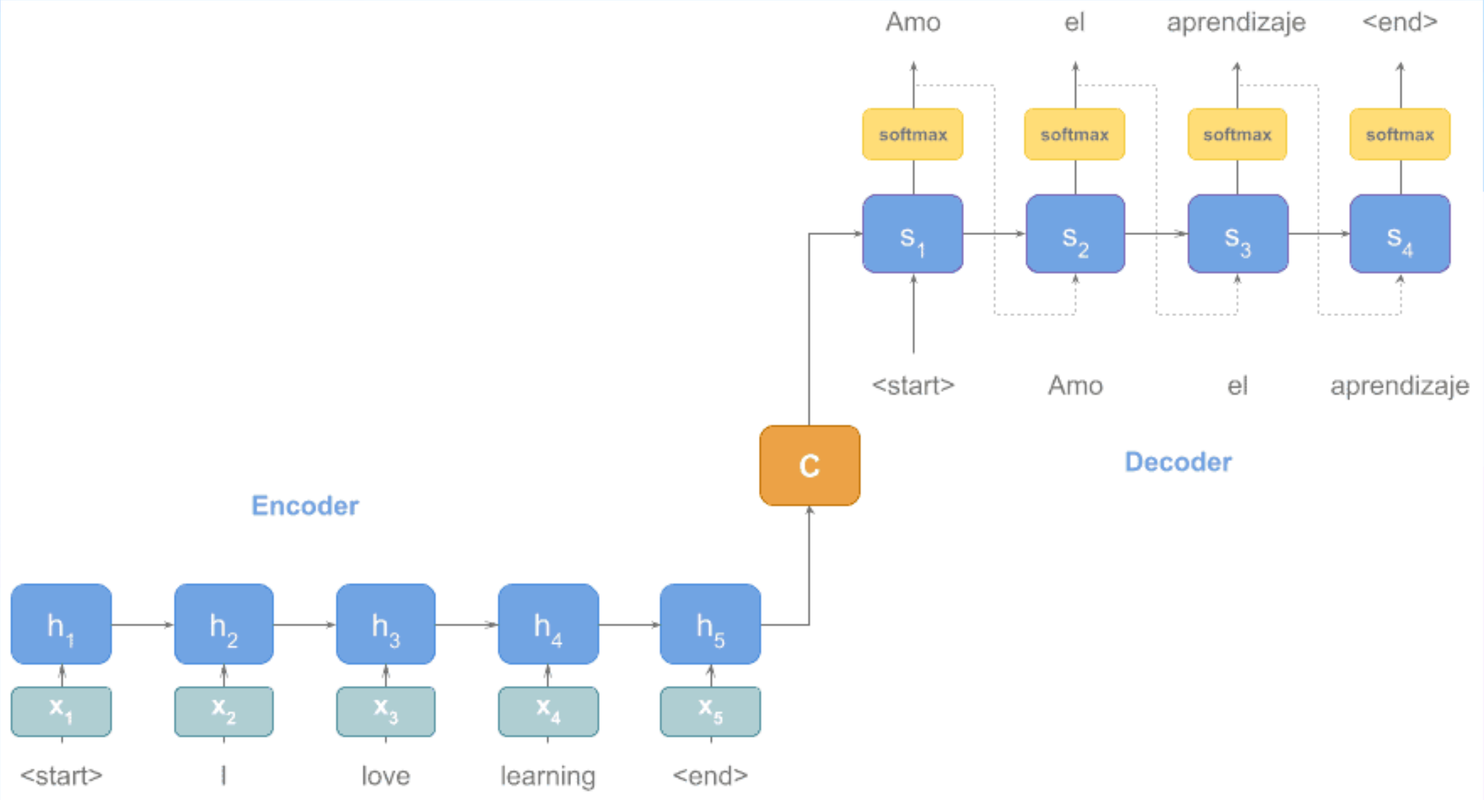




Encoder-Decoder

Der **Decoder** versucht stets das **nächste Wort** vorher zu sagen,
basierend auf dem letzten Wort und dem **Context** (Historie von Input und vorigen outputs)

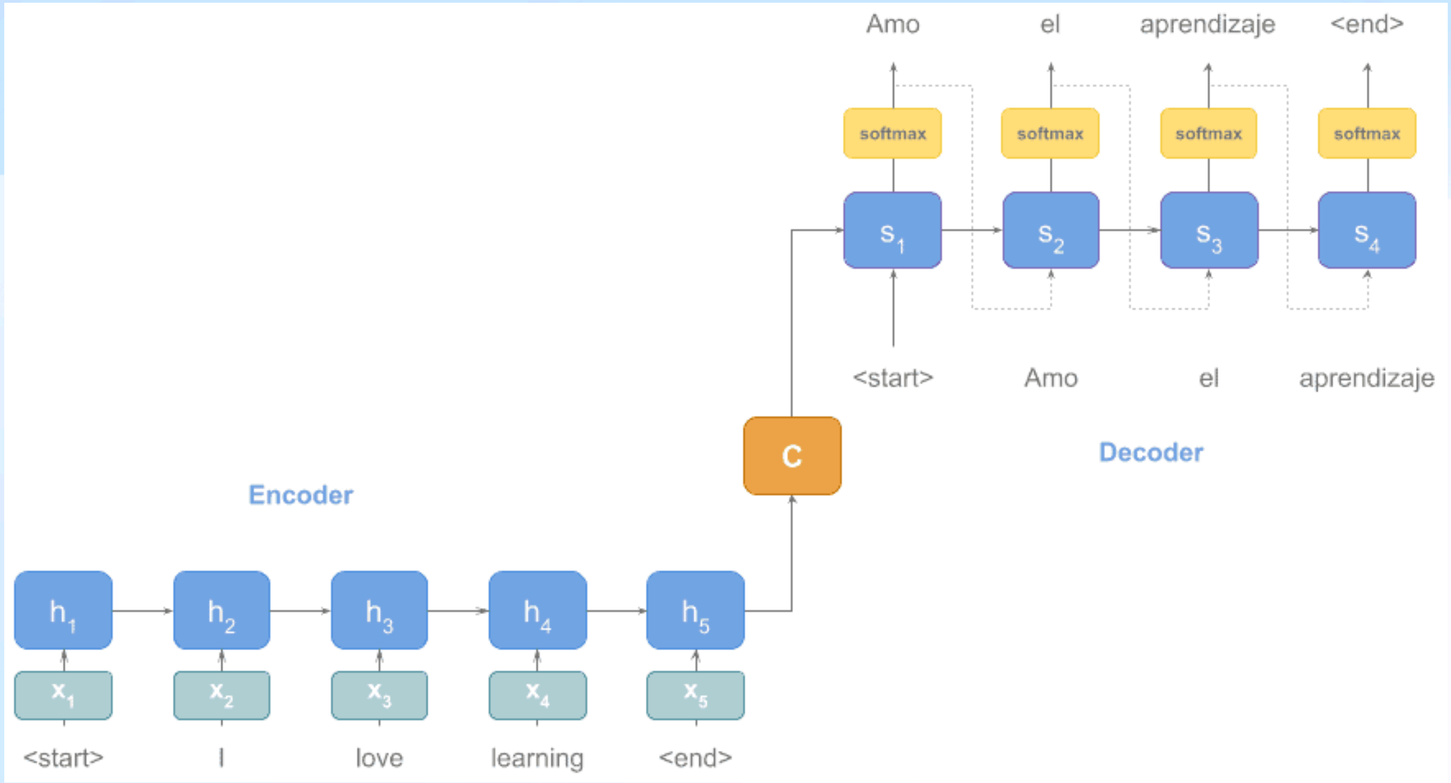




Encoder-Decoder

Anmerkung: man sieht schon, dass es sehr verschwenderisch wäre, verschiedene GRU Gewichte für verschiedene steps t zu trainieren, wenn eigentlich alle das selbe leisten:

decoder(context, current word) = next word



Teacher forcing

“Teacher forcing” trick: damit das System schneller lernt werden beim Training dem Decoder die **korrekten Worte als Input** mitgegeben, **statt die vom Decoder geratenen.**

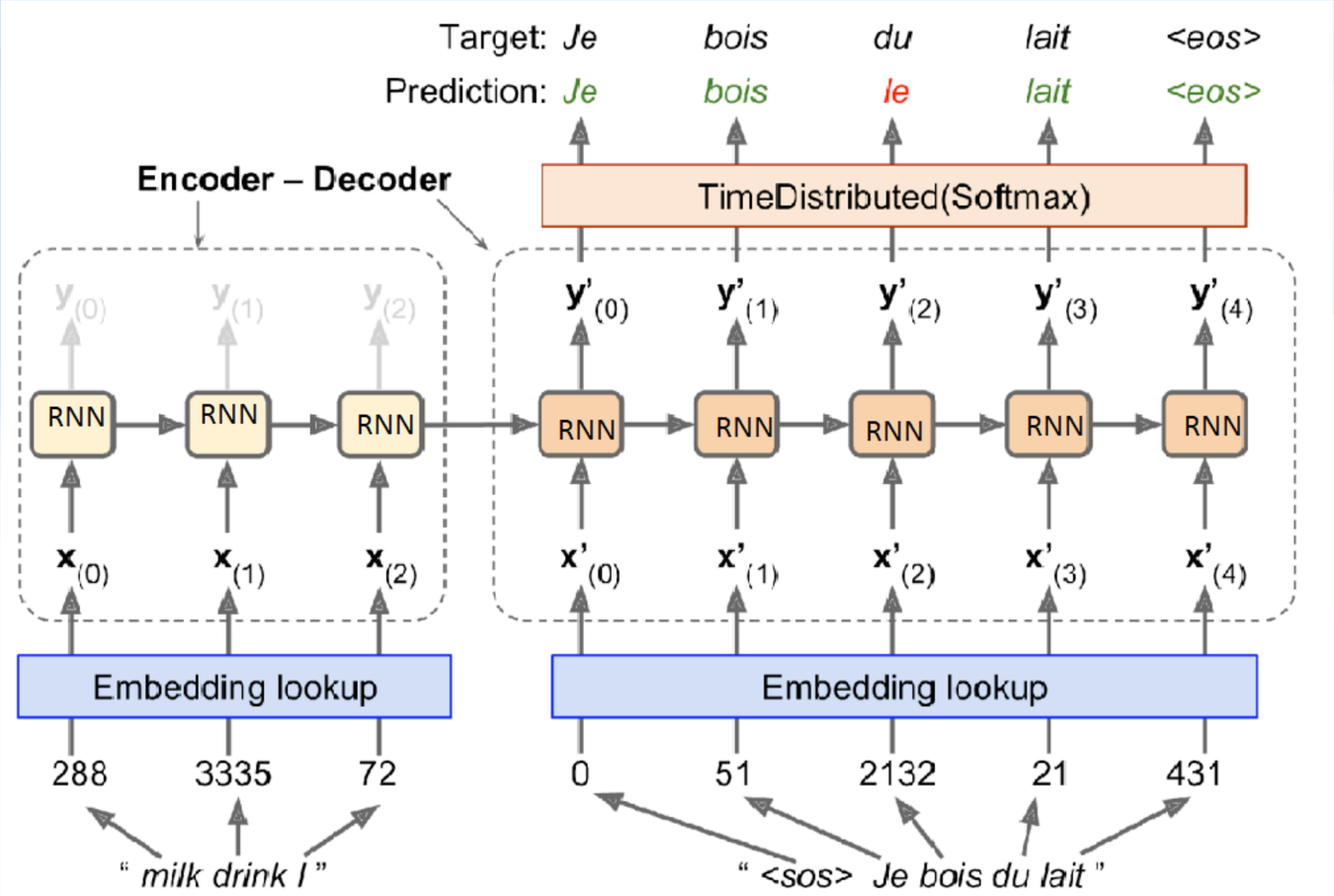


Fig 2. A simple machine translation

Encoder-Decoder

Vorteile:

- Flexibel(?): Kann variable Eingabe- und Ausgabelängen verarbeiten.
- Leistungsstark: Sehr gut für viele Sequenzverarbeitungsaufgaben.
- Architektur komplett innerhalb des Deep Learning Frameworks

Nachteile:

- trotzdem kann die Architektur recht komplex werden
- Lange Trainingszeiten, insbesondere bei langen Sequenzen.
- Schwierigkeiten bei der Handhabung sehr langer Sequenzen ohne Attention!

Aktuelle Entwicklungen: Transformers und Attention Mechanismen haben die Leistungsfähigkeit von Seq2Seq-Modellen signifikant verbessert und sind heute der Standard. Aufbau und Training sind aber fast gleich.

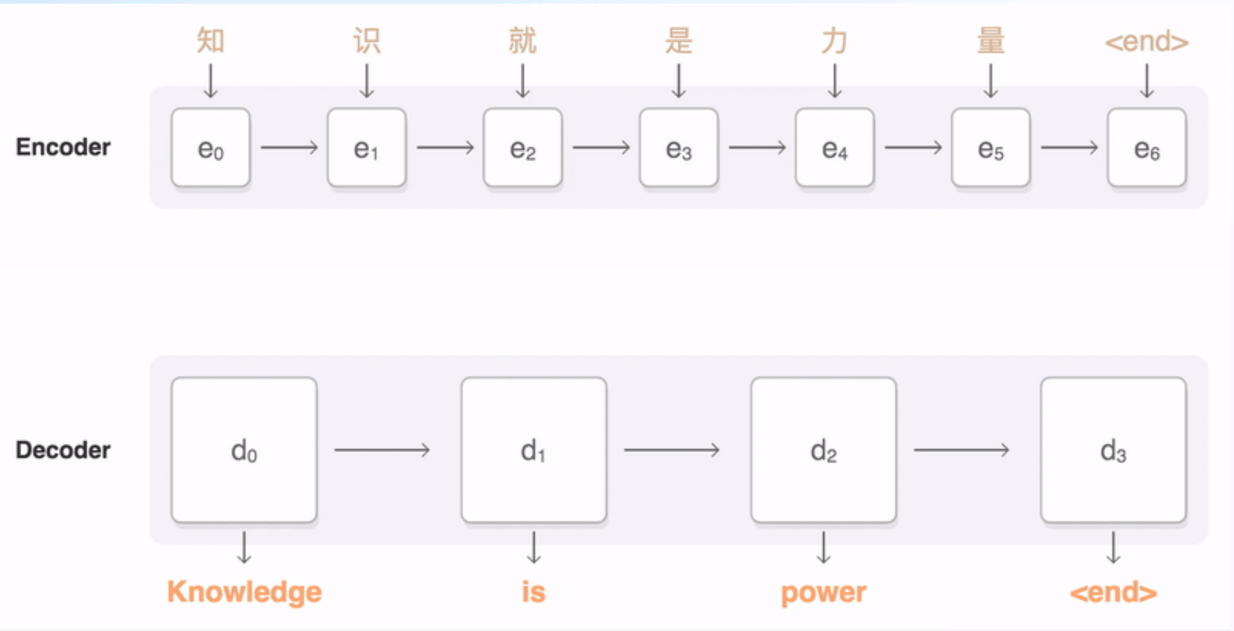
Encoder-Decoder

Aktuelle Entwicklungen:

Bidirektionale Encoder: Verwenden sowohl die Vergangenheit als auch die Zukunft der Sequenz zur Kodierung, was zu einer reicheren Repräsentation führt.

Transformers und Attention Mechanismen haben die Leistungsfähigkeit von Seq2Seq-Modellen signifikant verbessert und sind heute oft der Standard.

Tag 15 schon heute durchnehmen?



Hello World

Text to MNIST

"eins" -> embedding vector -> 1
"one" -> embedding vector -> 1 per conditional GAN
…
wie bekommt man alle Synonyme für eins …? Egal, geht nur ums Konzept der Kopplung / TL.

Lose Kopplung: Embedder NICHT mit trainieren. Trainiere nur mapping von gegebenem Vector zu Bild.

Text-Encoding:
Verwende einen Encoder (z.B. Transformer), um die Texteingabe ("eins", "one") in einen Latent-Vektor zu kodieren.

Bildgenerierung:
Verwende einen Decoder, der diesen Latent-Vektor nimmt und ein Bild (28x28 Pixel, wie im MNIST-Datensatz) erzeugt.

💡 Dies könnte durch einen CNN-basierten Decoder oder einen Generator in einem GAN erfolgen.

Auto Regression

Autoregression im Kontext von Sequenzvorhersage bedeutet, dass das Modell die nächste Ausgabe ausschließlich aus bereits erzeugten oder beobachteten Elementen der Sequenz berechnet.

- **Kausalität:** Nur vergangene Werte fließen in die Schätzung von x_t ein, nicht zukünftige.
- **Training:** Typischerweise “teacher forcing”: Man füttert das Modell mit den echten $x_{<t}$ nicht den eigenen Vorhersagen.
- **Inference:** Autoregressiv generiert; die Vorhersage von x_t wird wieder als Input für $t+1$ genutzt.

- **Beispiele: GPT**

Encoder-Decoder

https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html

https://colab.research.google.com/github/bastings/annotated_encoder_decoder/blob/master/annotated_encoder_decoder.ipynb

https://colab.research.google.com/github/patrickvonplaten/notebooks/blob/master/Encoder_Decoder_Model.ipynb

Encoder-Decoder

tag14/seq2seq.py

Translation for I am a student. : Ich bin Studentin.
all_input_ids.shape (220909, 5)

(1 1%) 0.7419
(['Ich', 'bin', 'kein', 'Buch.', '<EOS>'], None) (20 25%) 0.1995
…
(['Ich', 'bin', 'Student.', '<EOS>'], None)
(21 26%) 0.1940
(['Ich', 'bin', 'ein', 'Perfektionist.', '<EOS>'], None)
(22 27%) 0.1895
(['Ich', 'bin', 'völlig', 'erschöpft.', '<EOS>'], None)
(23 28%) 0.1850
(['Ich', 'bin', 'Studentin.', '<EOS>'], None)
(25 31%) 0.1770
(['Ich', 'bin', 'Student.', '<EOS>'], None)
(26 32%) 0.1736
(['Ich', 'bin', 'heute', 'beschäftigt.', '<EOS>'], None)
(27 33%) 0.1705
(['Ich', 'bin', 'Studentin.', '<EOS>'], None)
(28 35%) 0.1672

Encoder-Decoder

seq2seq_**byte**_attention.py

Ziel: Sage Zeichen für Zeichen die Übersetzung von "I am a student." auf Deutsch voraus
Statt 200000 tokens (Wörter) haben wir nur 256 tokens (bytes) in unserem Vokabular
das MultiheadAttention sollte aber die Korrelationen zwischen den Zeichen erkennen können!

Seq2Seq

Wie kann man das Problem der begrenzten input/output Sequenzen mildern?

a) **Segmentierung: Zerschneide Text in Phrasen.** Meist OK, aber Kontext geht verloren
b) Vorigen Zustand "**Context**" in nächste Übersetzung **weiter reichen**! als h0

Streaming Attention: Modelle wie das Transformer-XL implementieren ein “segment-level recurrence” mit einem **Zustand**, der zwischen den Segmenten übertragen wird. Dies ermöglicht eine effektive Verarbeitung von Datenströmen ohne feste maximale Sequenzlänge, indem Informationen aus vorherigen Segmenten genutzt werden.

Aufgabe (an mich?): **Streaming** Implementieren

Seq2Seq

Mächtigkeit von byte to byte encoder-decoders:

- **ALLE Daten in alle Daten umwandeln (z.B. source.c => a.out)**
- **Riesige Trainingsmenge (alle Daten)**
- **Keine out-of-vocabulary Probleme**
- **Fehlschreibungen ±** (erkannt aber auch reproduziert?)
- **GPT ist relativ schlecht in Mathe.**
- **Einfachst umzusetzen**

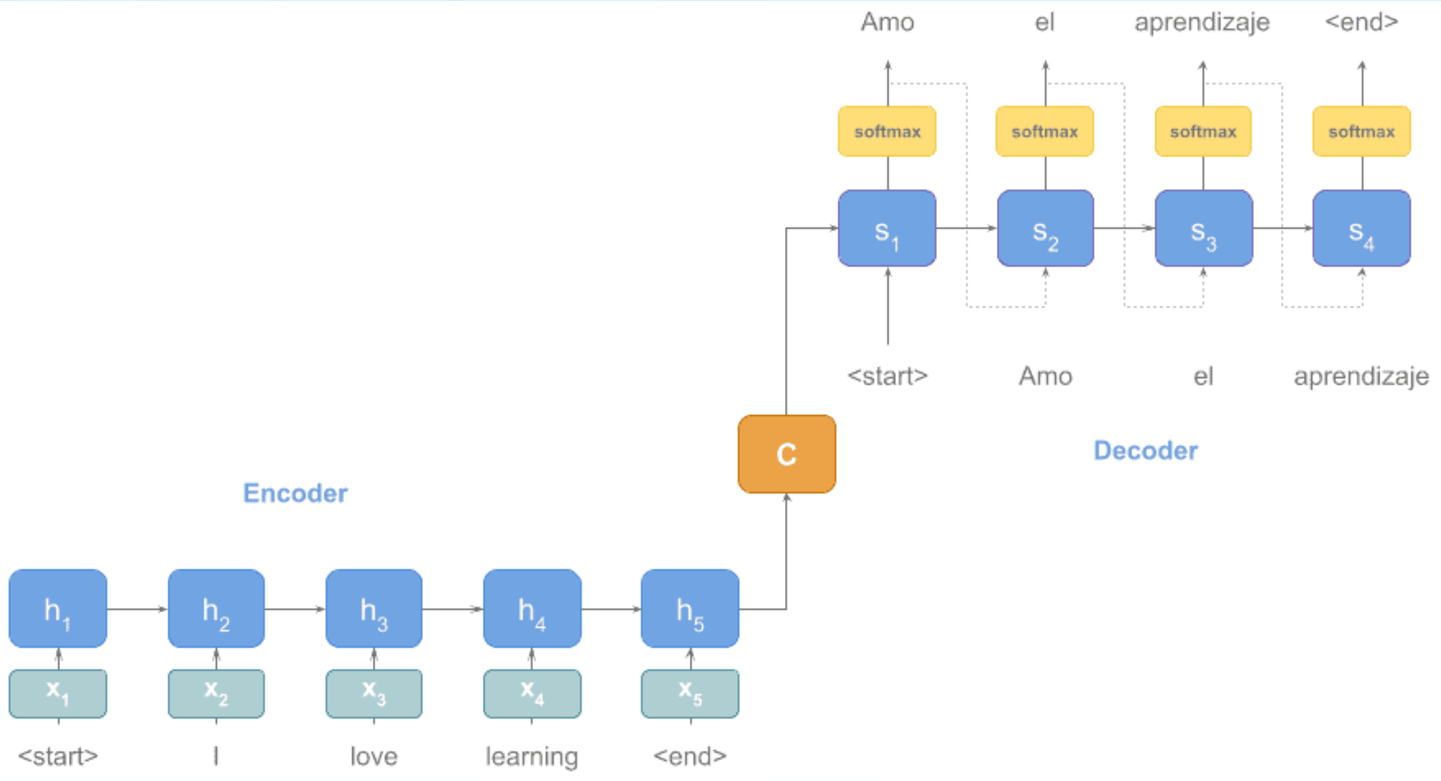
Wünschenswert: Meta channel / Meta Start/Stop Tokens, z.B. :

- **dies ist ein Bild**
- **dies ist englischer Text**
- **Zeilen-Nummern / Datei**
- **Source Qualität ('reddit, not trustworthy') …**

Selbst dieses einfache Modell ist unnötig kompliziert: => Char-RNN / Transformer

Synonyme

Context ≈ Embedding ≈ State ≈ Bottleneck (layer) ≈ internal Representation ≈ Latent Vector
Immer: dense compressed information



Populär gemacht von Koryphäe Ilya Sutskever neu: Konkurrenz zu OpenAI

Synonyme

Wie nennt sich die komprimierte Information (Aktivierung des Encoders)?
Feature Vektor … im Latent **Space** (eher allgemeiner)
Context Vektor … im Latent **Space**
Embedding Vektor … im Latent **Space**
Representation Vektor … im Latent **Space**

#Features ≈ hidden_size