NLP Einführung

KI Labor - Wintersemester 2022

Stefan Käser, Jochen Gietzen, Maximilian Blanck,
Adrian Westermaier, Tim Bossenmaier, Pascal Fecht

Karlsruhe, 04. November 2022

Zeitplan

Datum	Thema	Inhalt	Präsenz
30. Sept..	**Allg.**	**Organisation, Teamfindung, Vorstellung CV**	**Ja**
7. Okt.	Ausfall (DMA Techday)		
14. Okt.	CV	Q&A Sessions	Nein
21. Okt.	CV	Sprintwechsel, Vorstellung Assignment	Ja
28. Okt.	CV	Q&A Sessions	Nein
4. Nov.	**CV / NLP**	**Abgabe CV, Vorstellung NLP**	**Ja**
11. Nov.	NLP	Q&A Sessions	Nein
18. Nov.	NLP	Sprintwechsel, Vorstellung Assignment	Ja
25. Nov.	NLP	Q&A Sessions	Nein
2. Dez.	Ausfall (Winter Plenum)		
9. Dez.	**NLP / RL**	**Abgabe NLP, Vorstellung RL**	**Ja**
16. Dez.	RL	Q&A Sessions	Nein
23. Dez.	RL	Sprintwechsel, Vorstellung Assignment	Ja / Nein
13. Jan.	RL	Q&A Sessions	Nein
20. Jan.	**RL**	**Abgabe RL, Abschluss KI Labor**	**Ja**

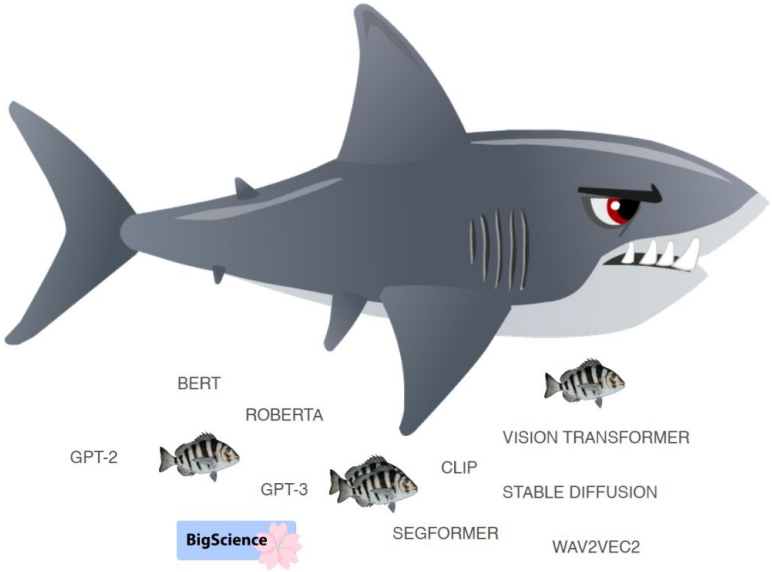
inovex

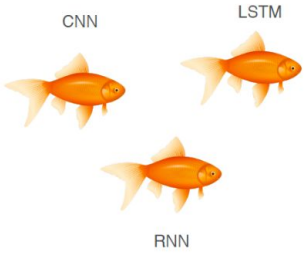
Agenda

- Trends & News in NLP
- Features
  - Count-basierte Ansätze und BoW
  - Word Embeddings (word2vec)
  - Vokabular
- Deep Learning Modelle in NLP
  - Rekurrenz
  - Attention
  - Transformer
- Vorstellung Übungsaufgaben

Trends & News in NLP

It’s all about Transformers



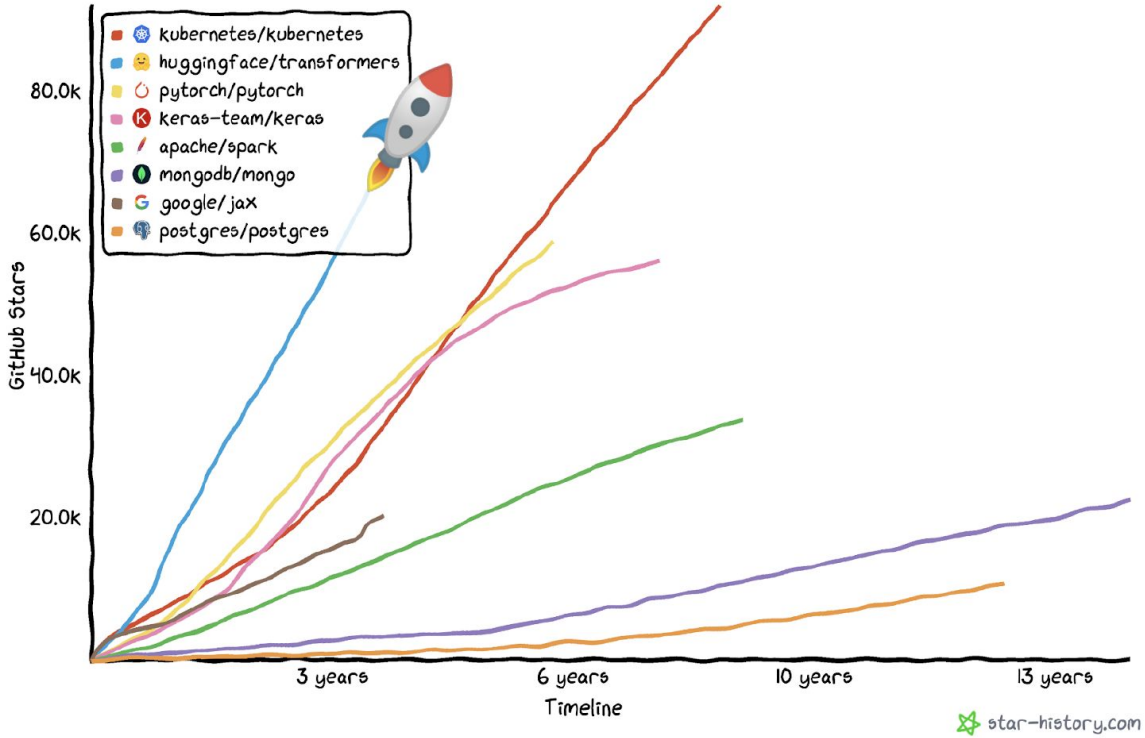


"*Transformers are emerging as a general-purpose architecture for ML*"
https://www.stateof.ai/

RNN and CNN usage down, Transformers usage up!
https://www.kaggle.com/kaggle-survey-2021

https://www.slideshare.net/JulienSIMON5/reinventing-deep-learning-with-hugging-face-transformers

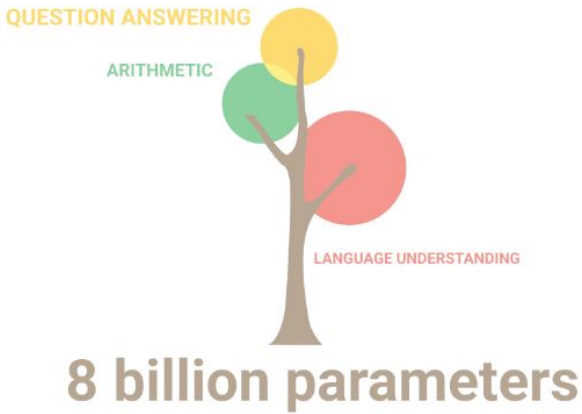
It’s all about Transformers



https://www.slideshare.net/JulienSIMON5/reinventing-deep-learning-with-hugging-face-transformers

inovex

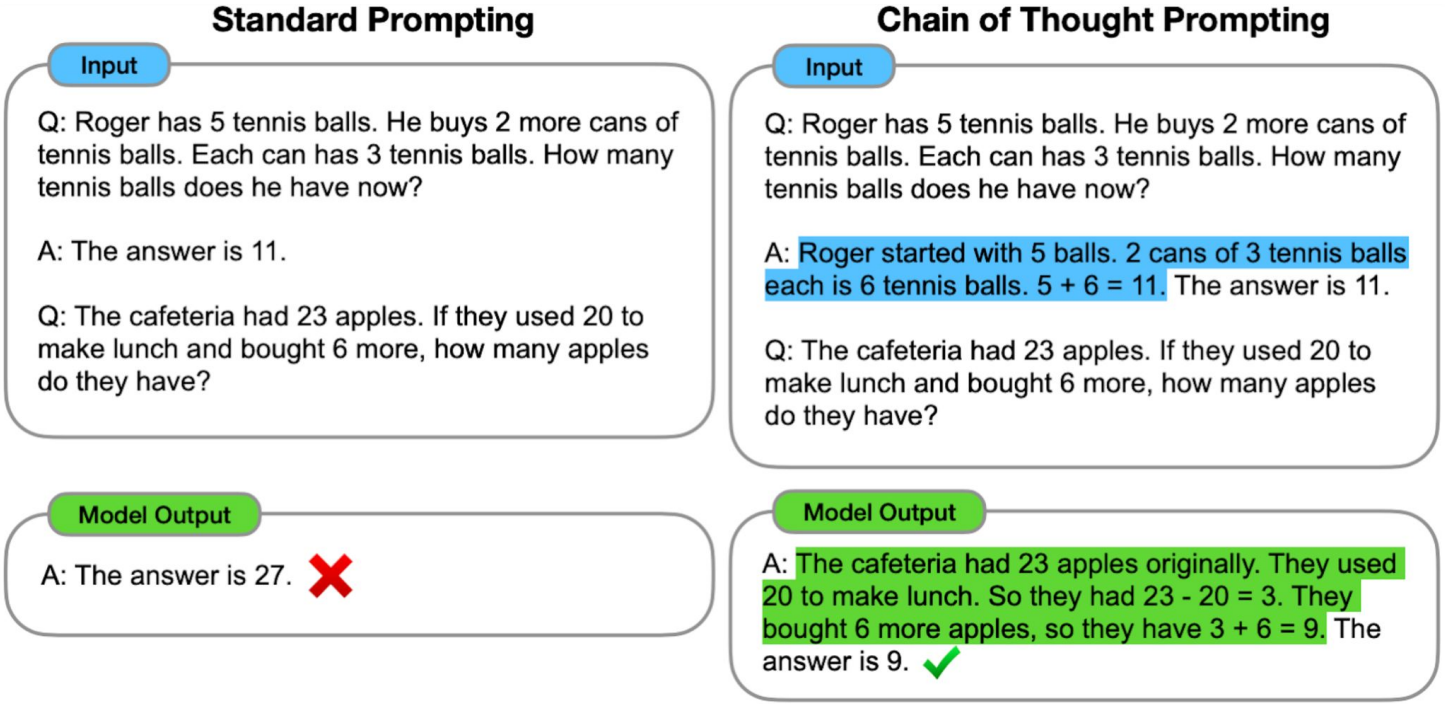
Pathways Language Model (PaLM)



https://ai.googleblog.com/2022/04/pathways-language-model-palm-scaling-to.html?s=09

inovex

Prompting & Chain-of-Thought Prompting

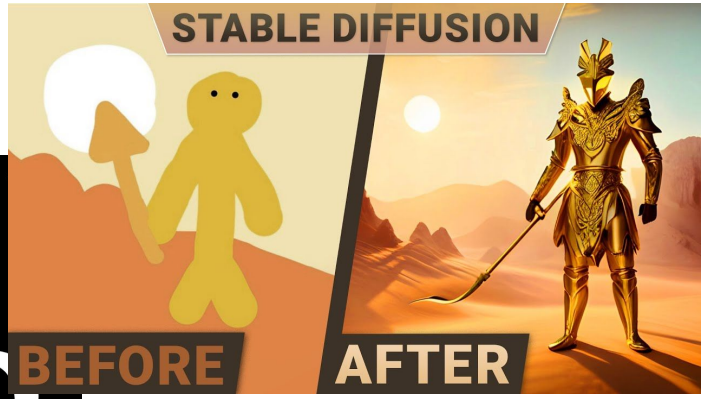


https://ai.googleblog.com/2022/05/language-maodels-perform-reasoning-via.html

inovex

Generative AI

https://openai.com/dall-e-2/



https://huggingface.co/spaces/stabilityai/stable-diffusion

A teddy bear painting a portrait.

https://make-a-video.github.io/?_hsmi=228530653&_hsenc=p2ANqtz--xv8V68o-WyL98JY77cAw8NsyzbbbFtFkjMdOYwZ6PqZZyLMO73NXMWjRWIwEJIy0WENJsUxuIWZz4oxQhX3AV5v-usA

inovex

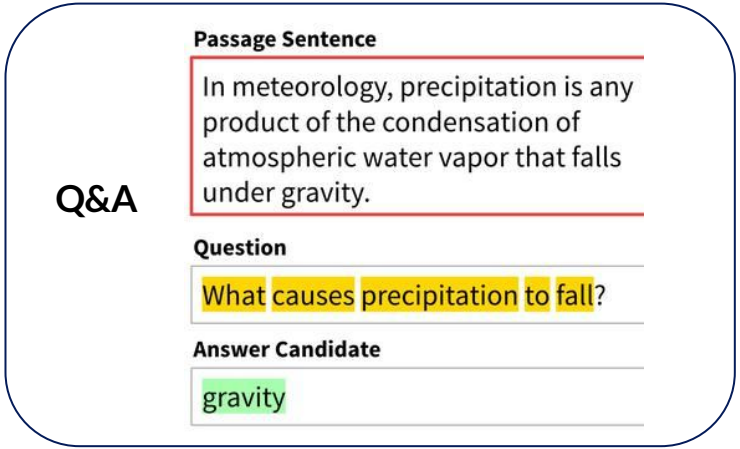
Natural Language Processing

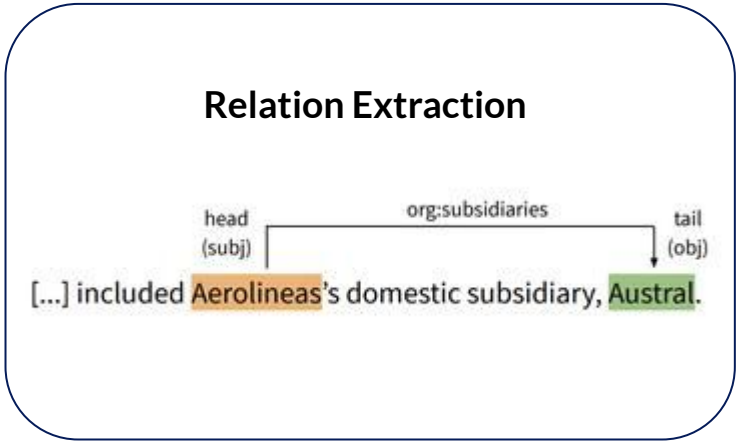
Natural Language Processing . . .

› **beschäftigt** sich unter anderem mit:

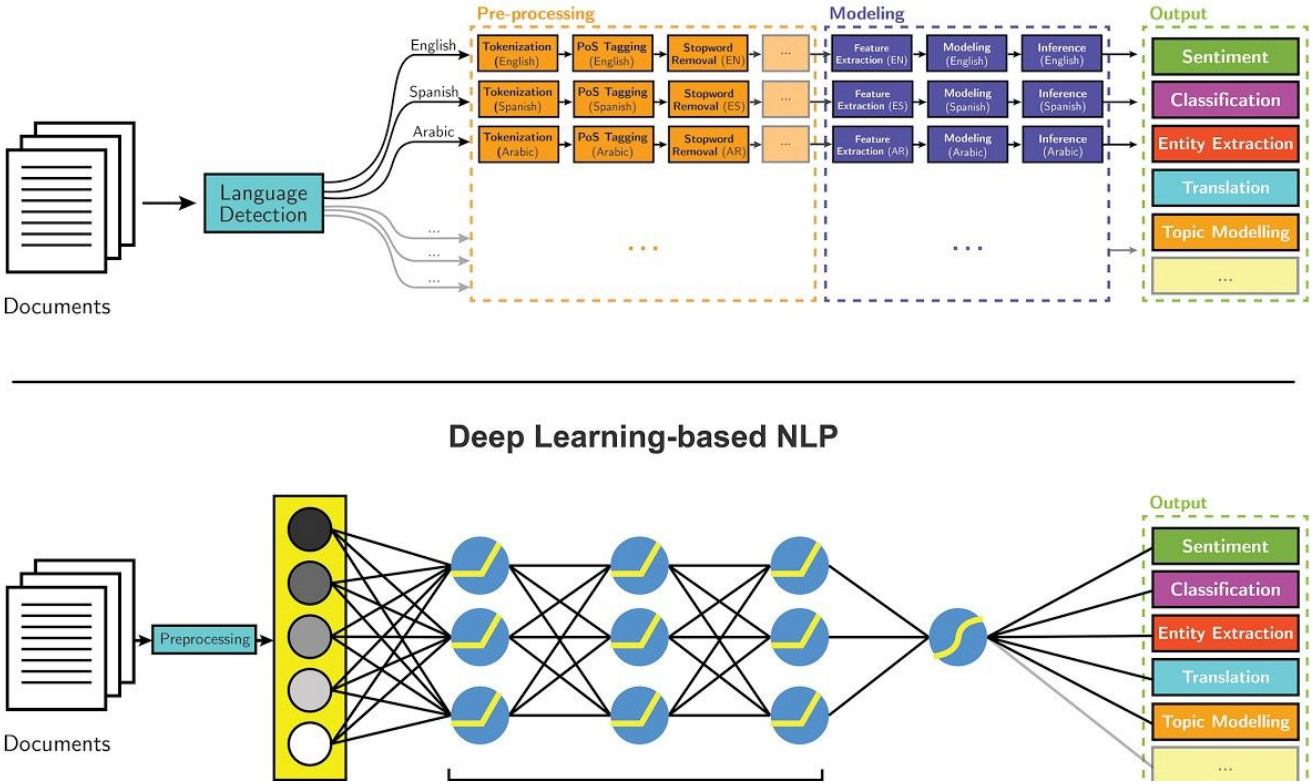
text classification, named entity recognition, machine translation, part of speech tagging, sentiment analysis, question answering, text summarization, text generation, speech recognition, speech to text, text to speech, chatbots, virtual assistants, relation extraction …

→ https://paperswithcode.com/area/natural-language-processing





Klassische vs Deep Learning NLP Pipeline



https://towardsdatascience.com/introduction-to-natural-language-processing-for-text-df845750fb63

Wie wird Sprache im Computer abgebildet?

- Sprache muss irgendwie in den Computer kommen.
  - Wort / Laut ⇒ Zahl(en) (Featurisierung)
- Das Vokabular einer Sprache ist in der Regel sehr umfangreich
  - Dimensionalitätsproblem (?)
- Einige Wörter sind enger verbunden als andere
  - Ähnlichkeit, Abstand ⇒ Maße?

Featurerisierung

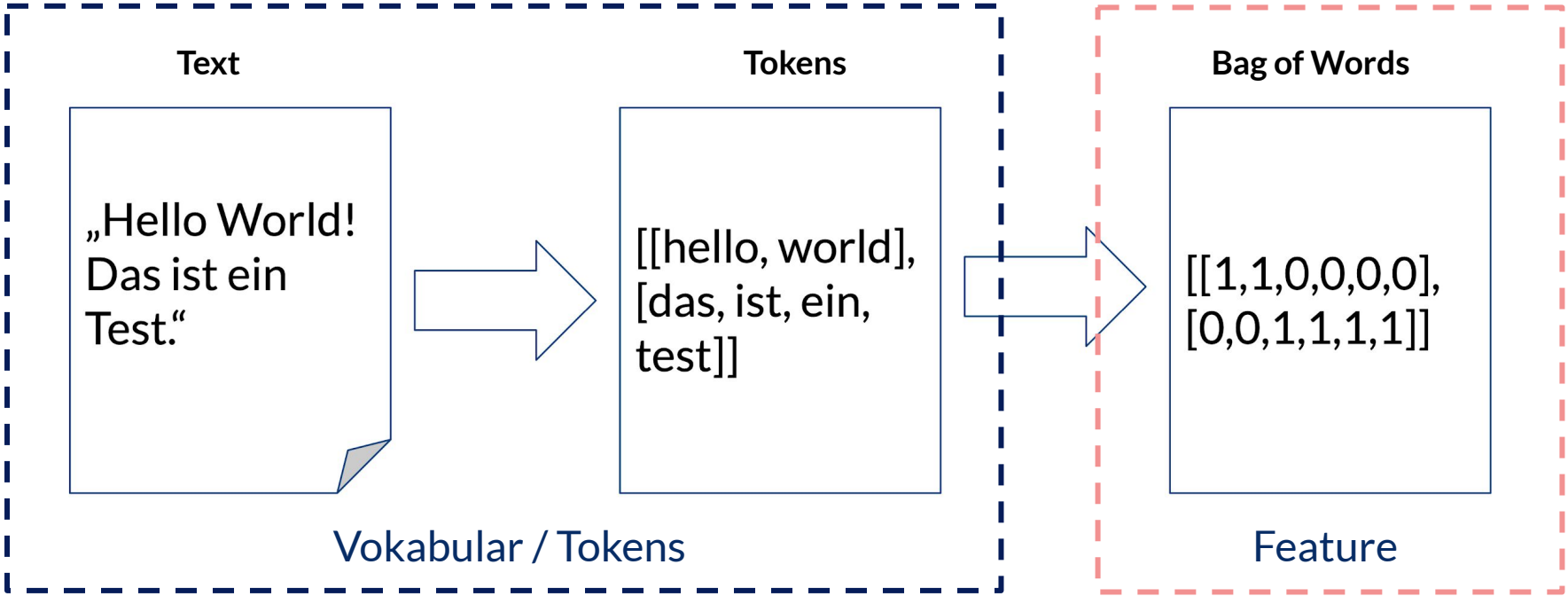
Bag of Words (BoW)

Zähle alle unterschiedlichen Wörter in den Texten, die betrachtet werden:

- Baue einen Vektor mit Vokabularlänge
- Jedes Wort hat einen Index im Vektor
- Merke die Anzahl für jedes Wort und Speichere die Zahl im Vektor
- Feature für einen Text:
  - [0,0,2,1,0,0,...,0,0,1,3,0,0]

→ **Problem**: Der Vektor beinhaltet sehr viele “0”en

Text → Tokens → Bag of Words



https://machinelearningmastery.com/gentle-introduction-bag-words-model/

BoW und hand-crafted Features

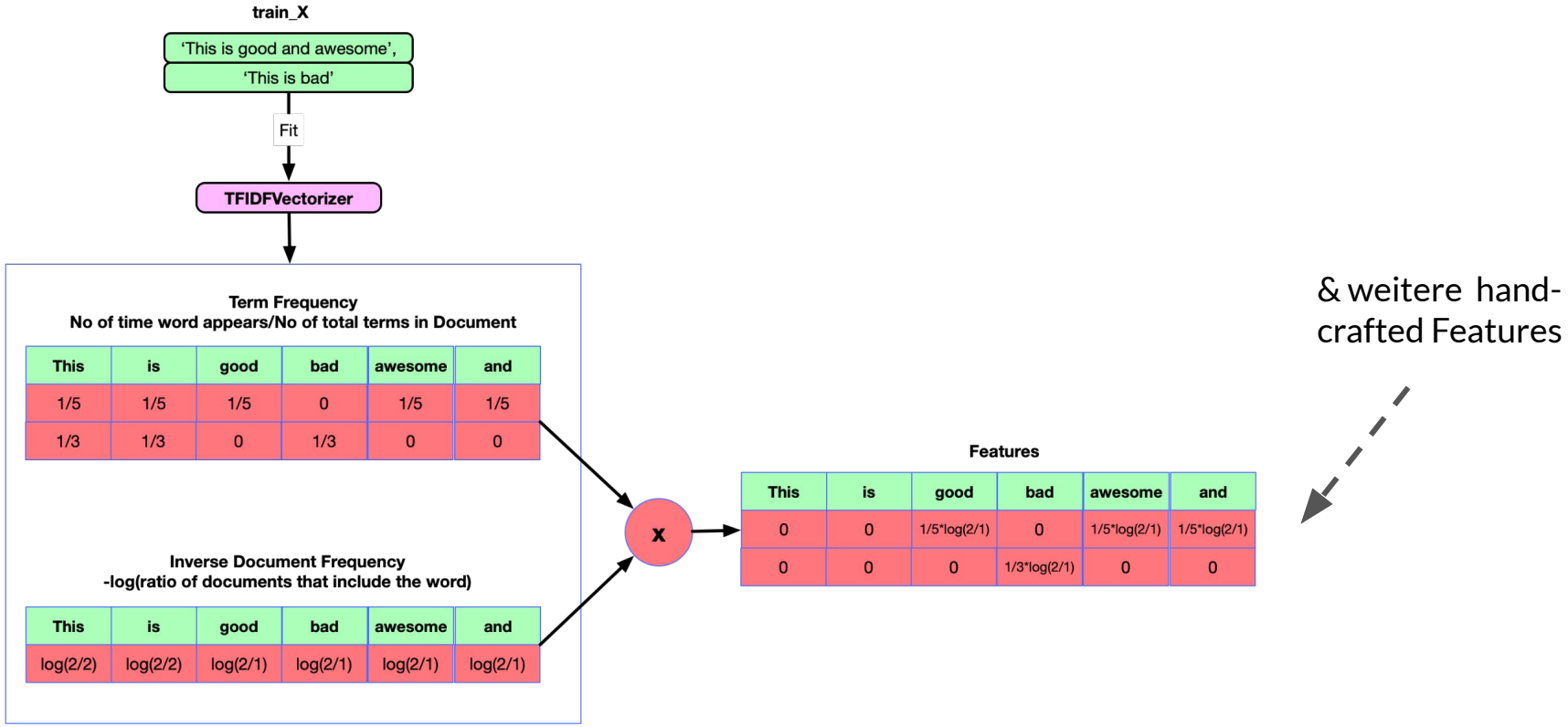
Optimierung von BoW mit **Tf-idf**:

- Reduzierung des Vokabulars mit Heuristik, die wichtige Tokens identifiziert (bspw. auf 2000 Input Features)
- Wörter werden durch ihr “vorkommen” (im Corpus) gewichtet

Hand-Crafted Features, wie:

- Statistiken (Satzlänge, Anzahl unterschiedlicher Wörter)
- Integration von externen Datenquellen (z.B. Wie viele Wörter kommen in einem Schimpfwort Lexikon vor?)
- Named Entities / POS Tags
  - Wird im Text über Personen, Organisationen, … gesprochen?
  - Wie viele Verben, Adjektive, usw. kommen vor?

Beispiel: Featurerisierung mit tf-Idf



https://mlwhiz.com/blog/2019/02/08/deeplearning_nlp_conventional_methods/

Word Embeddings

- Wörter werden als Vektoren in niedrig-dimensionalen Raum abgebildet
- Wörter mit ähnlicher Bedeutung stehen sich Nahe
- Werden auf großem Datensatz trainiert und dann für viele Tasks als grundlegendes “Sprachverständnis” genutzt (Transfer Lernen)
- Nur eine Bedeutung je Wort, unabhängig vom Kontext
  - Die **Bank** war besetzt mit zwei Menschen.
  - Die **Bank** erhöht die Zinsen.

Word2Vec

- “You shall know a word by the company it keeps” (Firth, J. R. 1957:11)
- Word2Vec-Paper: https://arxiv.org/abs/1301.3781

Idee:

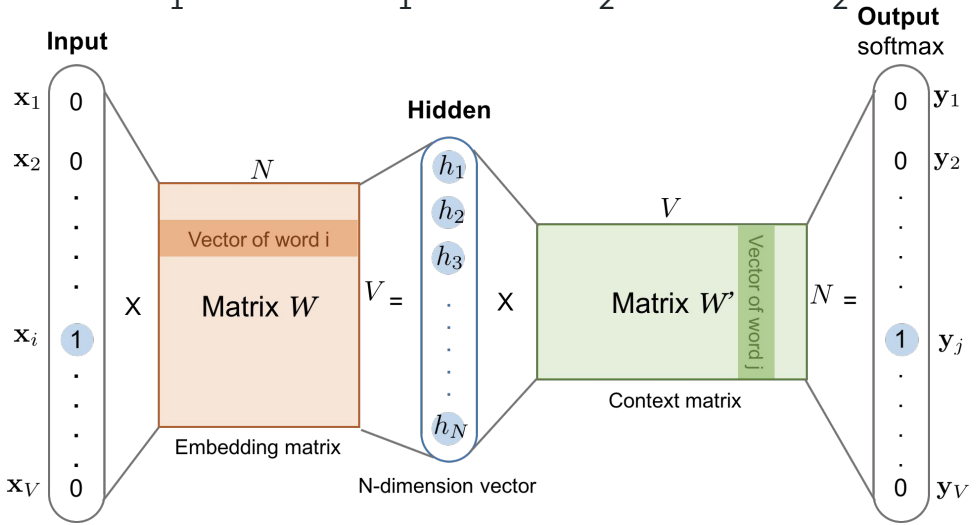
- Die umliegenden Wörter eines Wortes (in einem beliebigen Text) haben Einfluss auf die Bedeutung dieses Wortes.
- Beispielsatz: “*Ein Teddybär* malt *ein Portrait*.”

Word2Vec

- Jedes Wort wird auf einen Vektor der Länge **n** abgebildet:
- Bsp. “Auto” → [0.012, 0.981, -0.271,...]
- n ist im Word2Vec-paper 300, andere Werte möglich
- Wie werden diese Werte erzeugt?
  - 2 Ansätze:
    - Skip-Gram
    - CBoW
  - Sehr gute Erklärung: https://lilianweng.github.io/lil-log/2017/10/15/learning-word-embedding.html

Word2Vec - Skip-Gram (Kontext zum Wort)

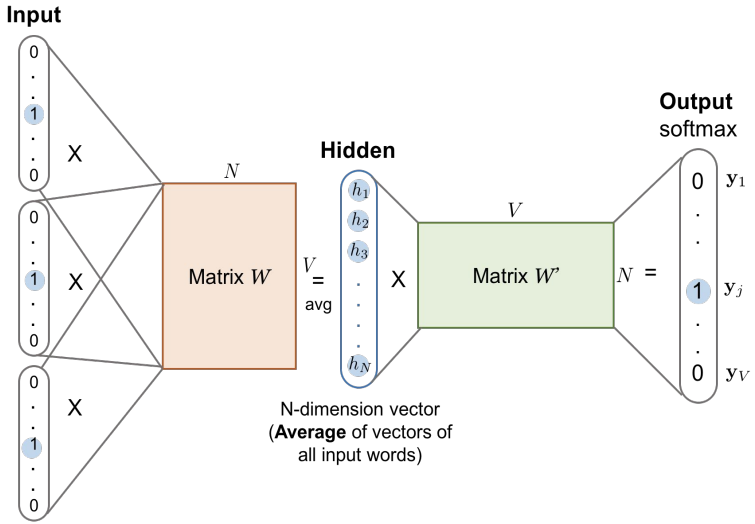
- Gegeben das Wort "**malt**", wollen wir den Kontext vorhersagen
  - "*Ein Teddybär* **malt** *ein Portrait.*" (Fenstergröße 5)
  - Kontext: ["Ein", "Teddybär", "ein", "Portrait"]; Target: ["**malt**"]
  - Trainingsdaten: X_1=["**malt**"], Y_1=["Ein"]; X_2 =["**malt**"], Y_2=["Teddybär"]; ...



https://lilianweng.github.io/lil-log/2017/10/15/learning-word-embedding.html

Word2Vec - Cbow (Wort zum Kontext)

- Im Kontext “Ein”, “Teddybär”, ”ein”, “Portrait”, wollen wir “**malt**” vorhersagen.
  - “*Ein Teddybär* malt *ein Portrait*.” (Fenstergröße 5)
  - Kontext: [“Ein”, “*Teddybär*”, ”*ein*”, “*Portrait*”]; Target: [“malt”]
  - Trainingsdaten: X= [“Ein”, “*Teddybär*”, ”*ein*”, “*Portrait*”], Y=[“malt”]



https://lilianweng.github.io/lil-log/2017/10/15/learning-word-embedding.html

Word Embeddings, Abstände, Gensim

- Mit Word Embeddings lassen sich nun Abstände berechnen
  - Bsp.: “Ente” und “Gans” haben einen geringeren Abstand als “Ente” und “Auto”, da sie in *Texten* öfters mit den gleichen Kontext Wörtern vorkommen
- Gensim (https://radimrehurek.com/gensim/) ist ein Python package, das das Handling von Wortvektoren vereinfacht.
- Beispiel:

```
from gensim.models import Word2Vec
model = Word2Vec.load("GoogleNews-vectors-negative300.bin'")
model.similarity('germany', 'france')
```

Vokabular

Vokabular: Herausforderungen

- Wie viele unterschiedliche Wörter soll mein Modell als Eingabe erhalten?
- Nicht alle Wörter können im Vokabular abgebildet werden.
- Sehr seltene Wörter gehen unter.
- Wörter (“der”, “ein”, …), die häufig vorkommen, aber wenig Inhalt enthalten, können starke Features werden.
- Wie gehen wir mit Noise (z. B. falscher Rechtschreibung) um?
- Was passiert, wenn wir zur Inferenz Wörter erhalten, die nicht Teil des Vokabulars sind?

Tokenisierung

“Wörter” → einfacher Ansatz: Trenne an “ “ (Leerzeichen):

```
re.compile(r'\W*\s+').split("Das Auto fährt")
```

→ [“Das”, Auto”, “fährt”]

Aber, was ist mit:

- “H-Milch”, “Auto-*n*bahn“, “Dr. Musterdoktor”
- Sätze → Trenne an “.” Punkt?
- Zahlen? Daten? Uhrzeiten?

Tokenisierung

Sind Wörter die richtige Granularität für Tokens?

- Subwords
  - Nicht im Vokabular enthaltene Token werden in Sub-Wörter geteilt.
- Characters
  - Jedes Char hat ein Encoding
  - Nachteil: Die Bedeutung von Wörtern geht verloren.
- Byte-Pair Encodings → Erklärung
  - Mix aus Subword und Character Level Encoding

Tagging

“Sehr seltene Wörter gehen unter.”

Beispiel: Jede IBAN ist ein eigenes Token

⇒ Kein relevantes Feature für Modell

```
1 if re.match('^DE(?:\\s*[0-9a-zA-Z]\\s*){20}$', TOKEN):
2     return "<IBAN>"
```

⇒ Modell erkennt alle (deutschen) IBANs als ein einziges Feature.

Stoppwörter

“Wörter (“der”, “ein”, …), die häufig vorkommen, aber wenig Inhalt enthalten, können starke Features werden.”

- Stoppwörter entfernen

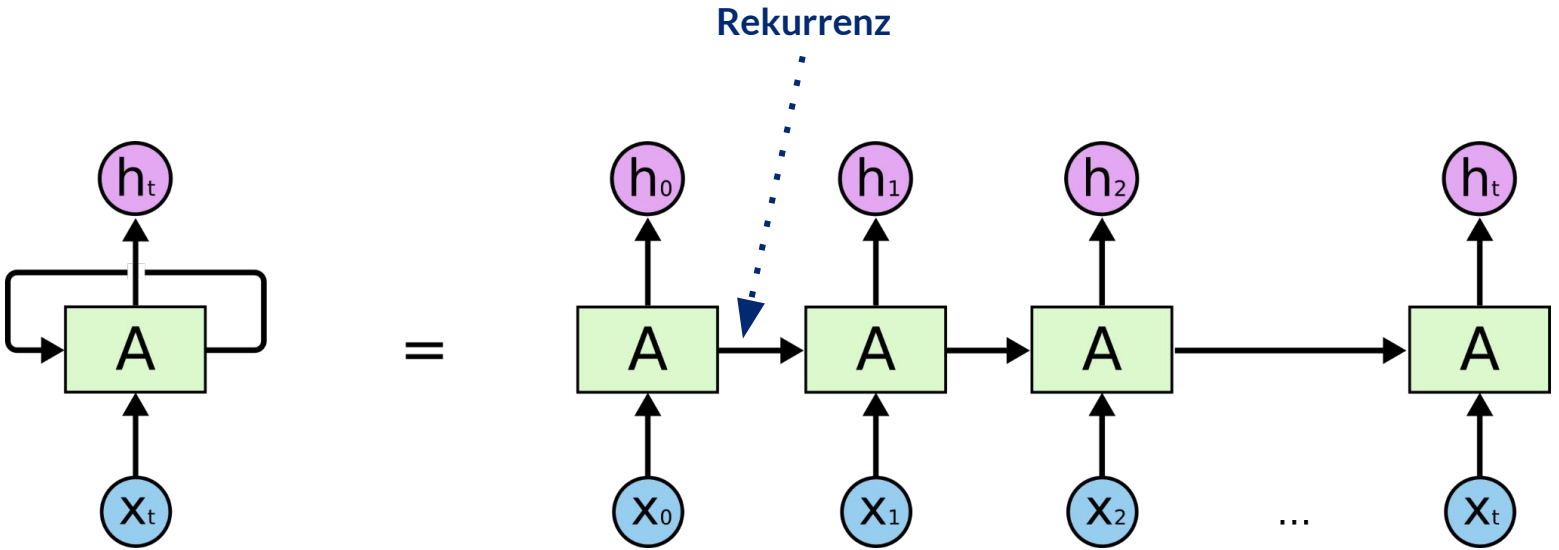
```
1 from nltk.corpus import stopwords
2 stop_words = set(stopwords.words('english'))
3 tokens = [w for w in tokens if not w in stop_words]
```

- Stemming / Lemmatisierung: Hauses, Häuser, Hauses, … ⇒ Haus
- Und viele weitere …

https://www.nltk.org/, https://spacy.io/

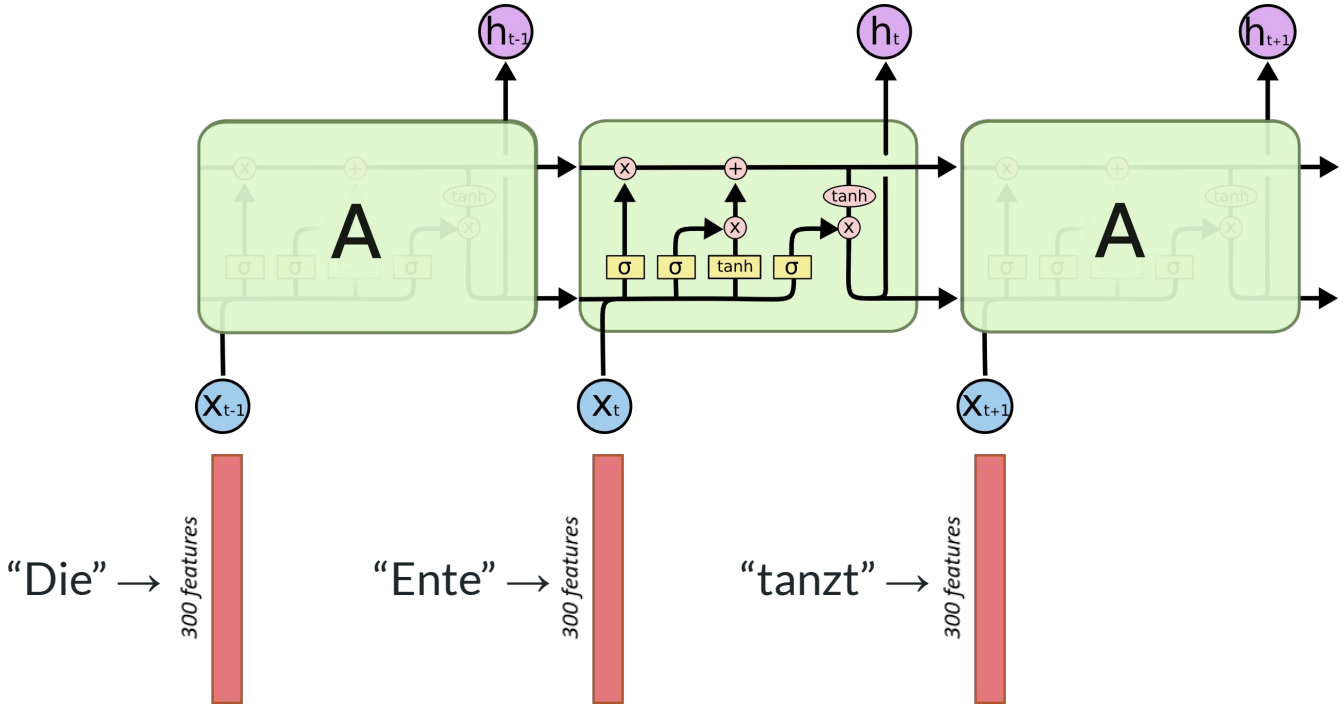
Modelle

LSTM - Long Short Term Memory



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

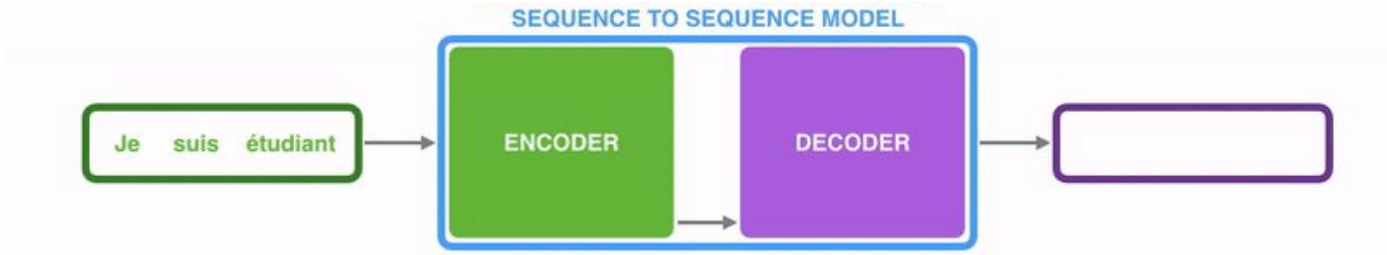
Rekurrenz: LSTM - Long Short Term Memory



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

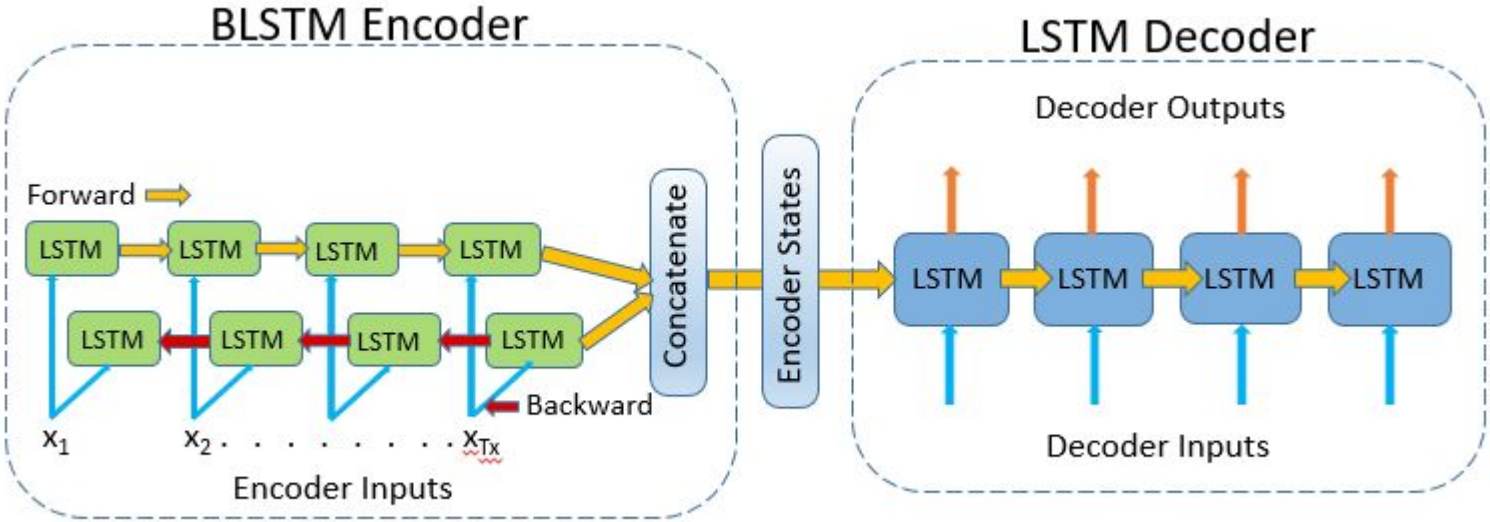
Sequence2Sequence Modelle

- Aufteilung in Encoder und Decoder Teile
  - Encoder: Liest Text ein und encodiert diesen
  - Decoder: Generiert Wort für Wort
- Prominentes Beispiel: Maschinelles Übersetzen



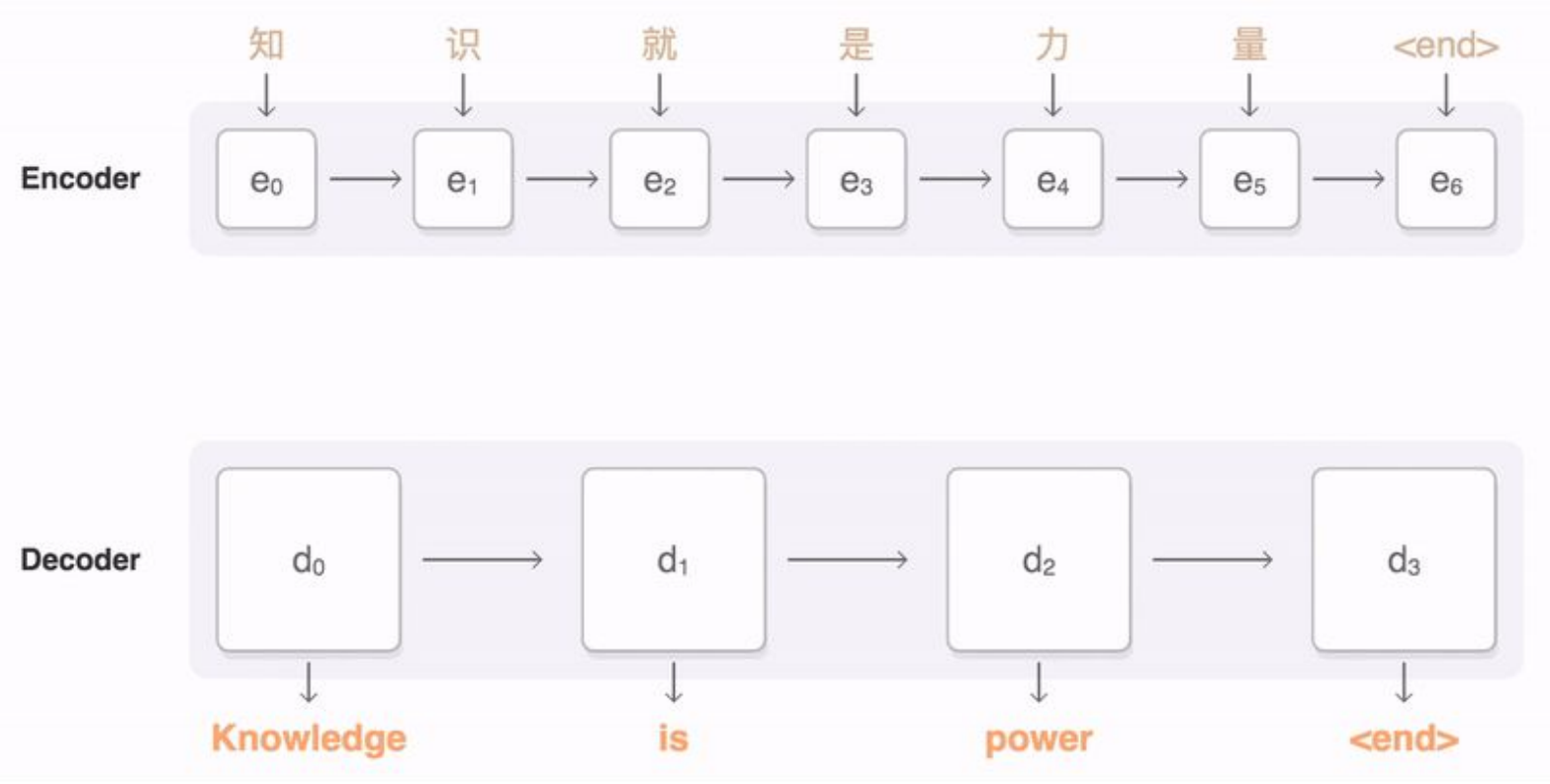
https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Seq2Seq Modelle mit Rekurrenz



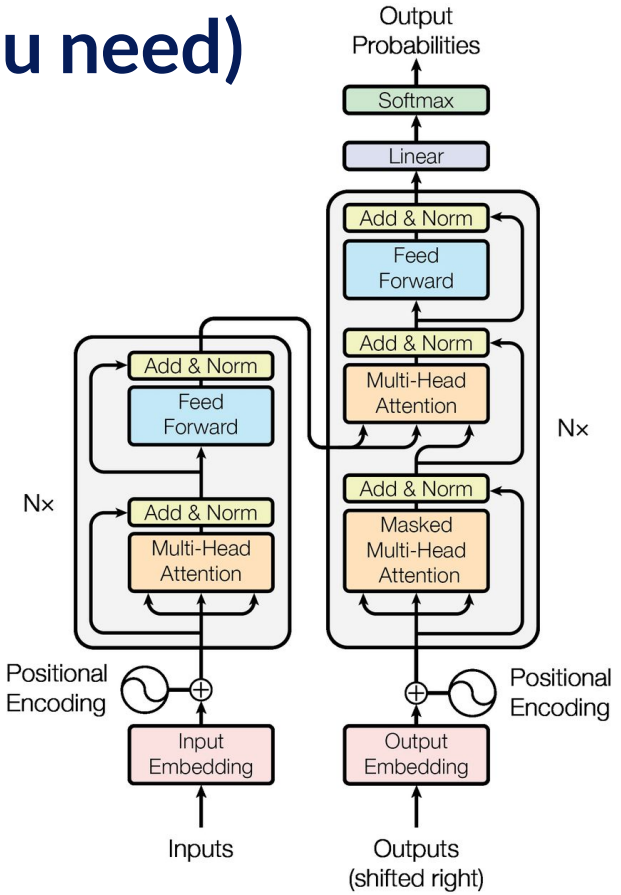
- **Training**: Der Decoder wird mit dem Übersetzungstext der Zielsprache trainiert
- **Inferenz**: Start Token von Zielsprache und Encoder State wird übergeben

Attention


知
识
就
是
力
量
<end>
Encoder
e0
e1
e2
e3
e4
e5
e6
Decoder
d0
d1
d2
d3
Knowledge
is
power
<end>

Transformer (Attention is all you need)

- **Keine Rekurrenz**: Statt Wort-für-Wort werden Sequenzen (Wörter, Sätze) parallel verarbeitet
- **Self-Attention**: Wichtigkeit von Wörtern im Satz wird gelernt
- **Positional Encodings** kombiniert mit Embeddings um Position im Satz zu behalten.

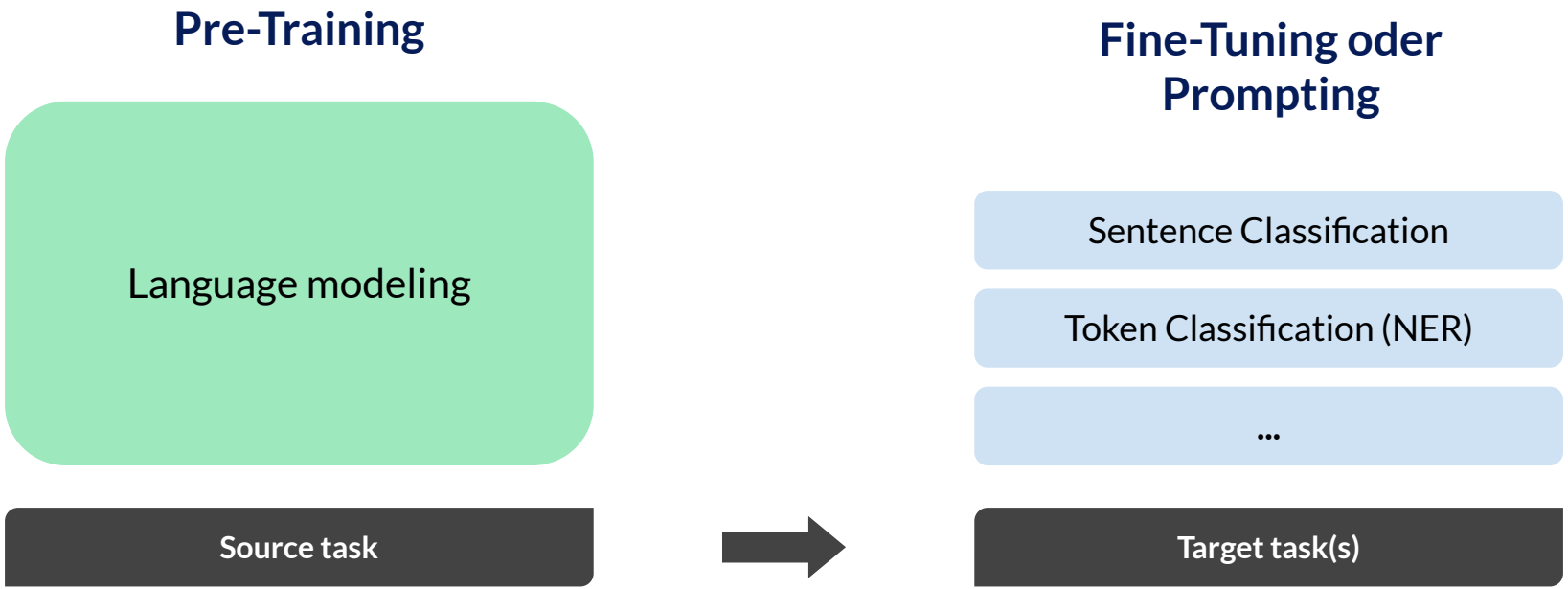


Pre-Trained Language Models

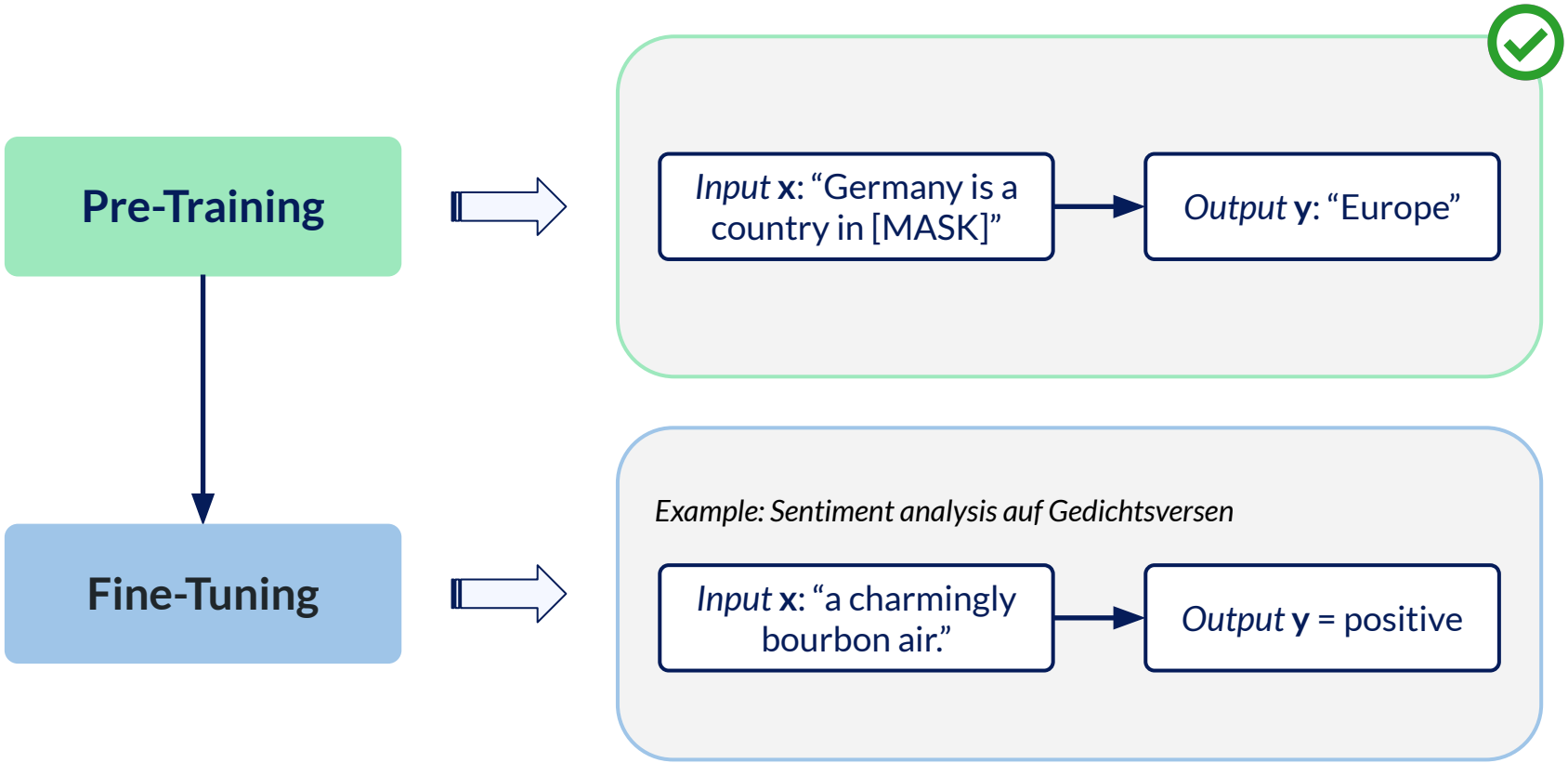
Pre-Trained Language Models (PLM)

- Language Model (LM): Was ist das nächste Wort im Kontext?
  - the weather was *[MASK]* ⇒ [MASK] = (0.5 hot, 0.3 cold, …)
- Semi-supervised learning Task (keine Labels nötig)
  - Auf sehr großen Datenmengen trainiert
- **Transformer** skaliert auf Milliarden Parameter
- Large LM enkodieren generelles Sprachverständnis

Warum “Pre-Trained”?



Fine-Tuning: Sentiment Analysis



In der nächsten Vorlesung

Im zweiten Teil von NLP…

- Unterschiedliche Typen von Language Models
- **Fine-Tuning** Methoden
- Prompting

In den Übungsaufgaben:

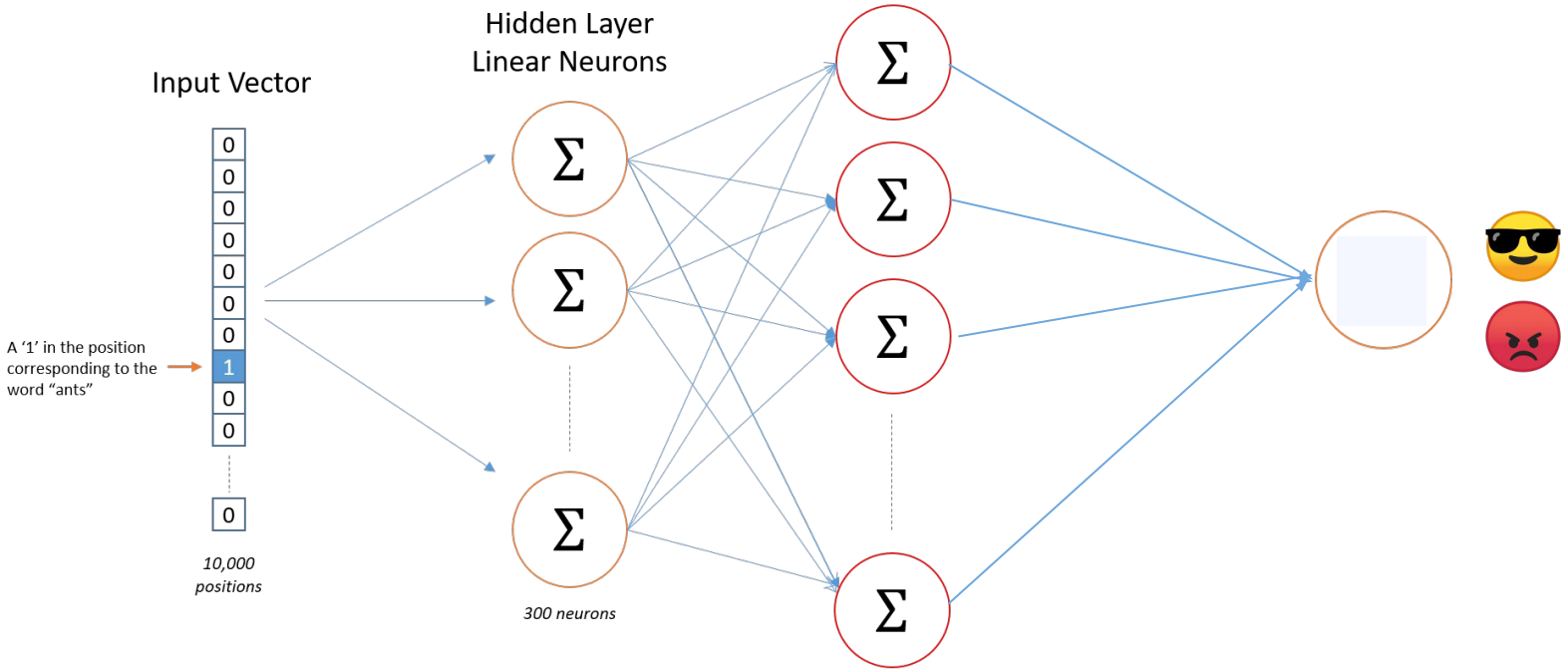
- Erste Berührungspunkte mit Transformers und huggingface transformers
- “Einfaches” Fine-Tuning

Klassifikation
Sentiment Analyse

Sentiment Analyse

- Sentiment Analyse = Stimmung eines Textes erkennen
- Herausforderungen wie:
  - “XY ist *blöd*” vs. “XY wäre *blöd* gewesen, diese Chance nicht zu nutzen.”
- Häufig verwendet für Reviews (Produkt-Reviews, Restaurantbesuche, …)
- In Übungsaufgabe 2:
  - Datensatz: Sentiment140 (16 Mio Tweets)
  - Target Label klassifiziert durch enthaltene Emojis
  - Ziel: Sentiment hinsichtlich Produkte, Marken etc. aus den Tweets herausfinden.

Binäre Sentiment Analyse mit BoW



basierend auf http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

inovex

Language Models


colab

Vielen Dank

inovex GmbH
Ludwig-Erhard-Allee 6
76131 Karlsruhe