Maschinelle Sprachverarbeitung mit Large Language Models

Von Bag of Words bis Agents

Tim König, Universität Hildesheim, 27. & 28.2.2025

Ziele

- **Überblick** über die neuere Entwicklung der maschinellen Sprachverarbeitung
- **Logik** der vorgestellten Methoden und wie sie aufeinander aufbauen
- **Keine Detailvorstellung** der einzelnen Algorithmen oder der zugrundeliegenden Mathematik
- **Praktische Anwendung** der Methoden

Tag 1

- Bag of Words & Word Embeddings
- Transformer
- Classification mit Transformer-Modellen
- Model Evaluation

Tag 2

- Classification mit Large Language Models
  - OpenAIs GPT-Modelle
  - Ollama-Framework
- Agents & kombinierte Modelle mit Langchain

Maschinelle Sprachverarbeitung

Annahmen

- Sprache bildet latente Sinnstrukturen ab
- Sprache kann statistisch repräsentiert werden
- Die Modellierung von Sprache wird genauer, je größer unsere Datengrundlage ist

Trends

Begrenzte Compute Power → Mehr Compute

Weniger Daten → Mehr Daten

Mehr Preprocessing → Weniger Preprocessing

Bag of Words

Prinzip

- Zählen von Wörtern in Dokumenten:
  „Welches Wort kommt wie häufig in welchem Dokument vor?“
- Repräsentation idR. als Matrix
- Wortreihenfolge wird ignoriert – alles kommt in einen „Sack“ (*Bag of Words*)

Document Feature Matrix (DFM)

I am fine, am I not?

It is a fine day.

Each day, I have an apple.

...

I	am	fine	not	it	is	a	day	each	have	…
2	2	1	1	0	0	0	0	0	0	…
0	0	1	0	1	1	1	1	0	0	…
1	0	0	0	0	0	0	1	1	1	…
…	…	…	…	…	…	…	…	…	…	…

Methoden

- Auszählen von Worten in Dokumenten (Dictionary Ansätze)
- Gewichtung, z.B. mit TF-IDF (TermFrequency-InverseDocumentFrequency)
- Projektion auf Wortkookkurrenzmatrix
- Matrixoperationen, z.B.:
  - Cosinusähnlichkeit zwischen Dokumenten
  - Dimensionalitätsreduktion, z.B mit Singular Value Decomposition (SVD), für Clustering etc. (auch bekannt als Latent Semantic Analysis, LSA)
  - …

Vor- und Nachteile

Vorteile

- Simpel und intuitiv
- Erlaubt Messung eng definierter Konzepte mittels Dictionaries
- Ausreichend für einfache Klassifikationen, z.B. thematisches Clustering von Dokumenten

Nachteile

- Wortreihenfolge wird ignoriert („Hund beißt Mann“ vs. „Mann beißt Hund“)
- Kontext wird ignoriert (z.B. „Ich arbeite auf der Bank“ vs. „Ich sitze auf der Bank“)
- Sparsity: hoch-dimensionale Matrix mit wenig Informationen => Skaliert schlecht & vergleichsweise langsam
- Kann nicht auf vortrainierte Modelle zurückgreifen (jedes Mal bei null anfangen)

Word Embeddings

Prinzip

You shall know a word by the company it keeps.
- J.R. Firth[1]

- Intuition: Wörter können durch ihren Kontext beschrieben werden

- Distributive Hypothese: *Wörter mit ähnlichen Verteilungen haben ähnliche Bedeutung* (gleicher Kontext, gleicher Bedeutung!)

[1] *Firth, J.R. (1957). "A synopsis of linguistic theory 1930-1955". Studies in Linguistic Analysis: 1–32.* Reprinted in *F.R. Palmer, ed. (1968). Selected Papers of J.R. Firth 1952-1959. London: Longman.*

Praxis

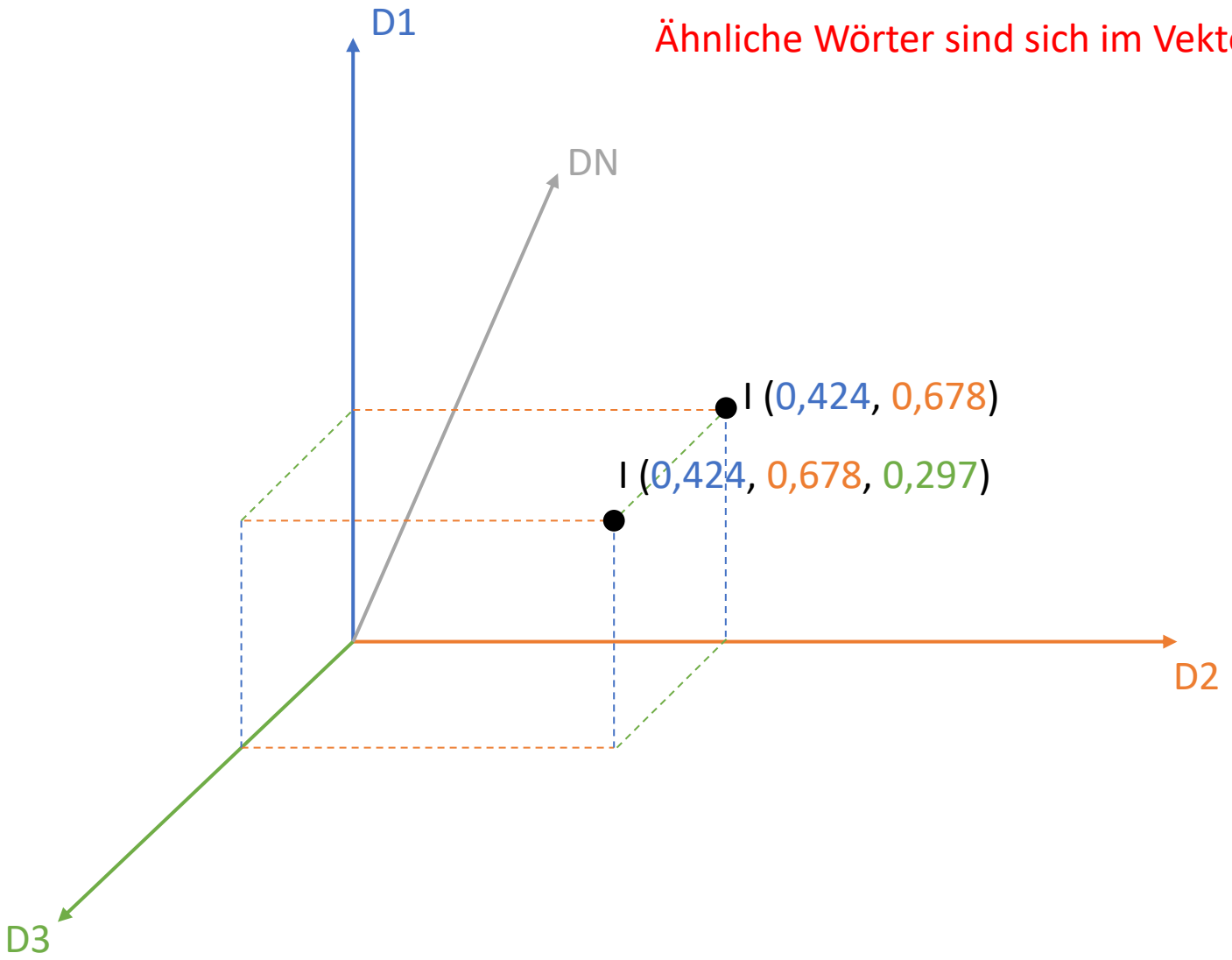
- Repräsentation von Wörtern in N latenten Dimensionen, idR. mittels Vektoren
- Berechnung der Dimensionen mittels verschiedener Techniken (modellabhängig)
- Dense Matrices, d.h. keine “leeren” Dimensionen
- Ähnliche Wörter sind sich im Vektorraum nahe, z.B. haben Synonyme einen (fast) identischen Vektorraum
- Operationen der linearen Algebra können zum Vergleich der Vektoren angewendet werden
- Neben einzelnen Wörtern können auch andere Sinneinheiten, z.B. Sätze, embedded werden

Word Embedding Matrix

	I	am	fine	not	it	is	a	day	each	have	…
D1	0,424	0,124	-0,531	-0,152	0,879	0,512	0,104	-0,164	0,512	0,523	…
D2	0,678	-0,575	0,612	0,698	-0,276	0,987	0,205	0,179	0,087	0,142	…
D3	0,279	0,679	0,125	0,412	0,128	-0,153	0,164	0,635	-0,012	0,652	…
DN	…	…	…	…	…	…	…	…	…	…	…

Vektorraum

	I
D1	0,424
D2	0,678
D3	0,279
DN	…



Ähnliche Wörter sind sich im Vektorraum nahe!

Berechnung der Dimensionen

Zahlreiche unterschiedliche Arten, die Dimensionen und deren Werte zu berechnen, z.B.

- Einfache Dimensionalitätsreduktion der Wortkookkurrenzmatrix (LSA)
- Word2Vec
- GloVe
- BERT
- Nomic
- …

Static Word Embeddings

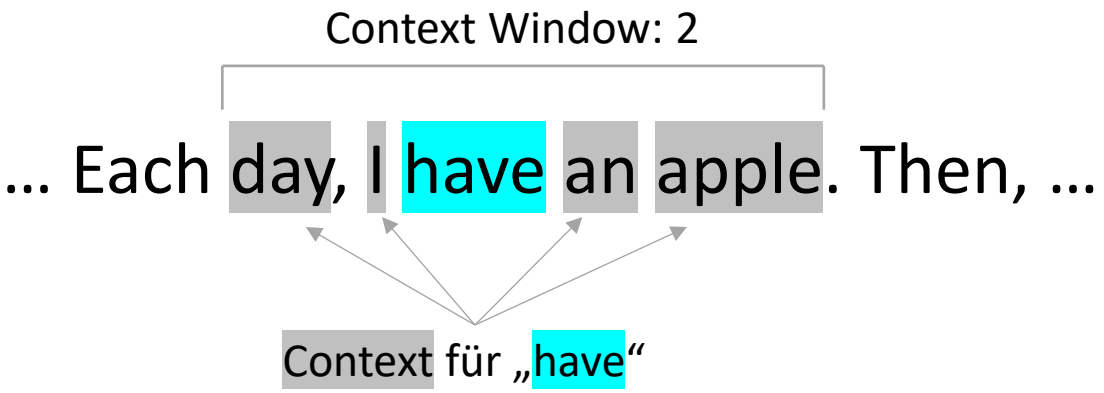
Prinzip

- Ein Vektor pro Wort
- Anzahl Dimensionen variabel
- Höhere Anzahl Dimensionen ermöglicht die Abbildung feinerer Nuancen, z.B. in unterschiedlichen Regionen des Vektorraums
- Vektor eines Worts wird mithilfe der umgebenden Wörter probabilistisch berechnet

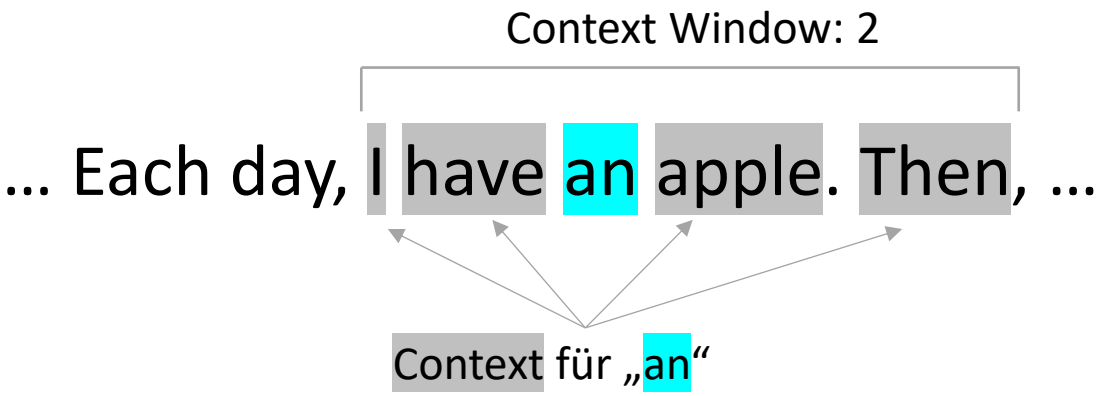
Praxis

- Embeddings können auf Grundlage des zu untersuchenden Corpus neu generiert werden…
- … oder aus einem bestehenden Modell übernommen werden (idR. größere, möglichst ausgewogen kuratierte Corpora)
- Corpuszusammensetzung, Preprocessing (Entfernen von Stopwords etc.), Modellierungsentscheidungen (Context Window, Anzahl Dimensionen) beeinflussen die Ergebnisse u.U. entscheidend!

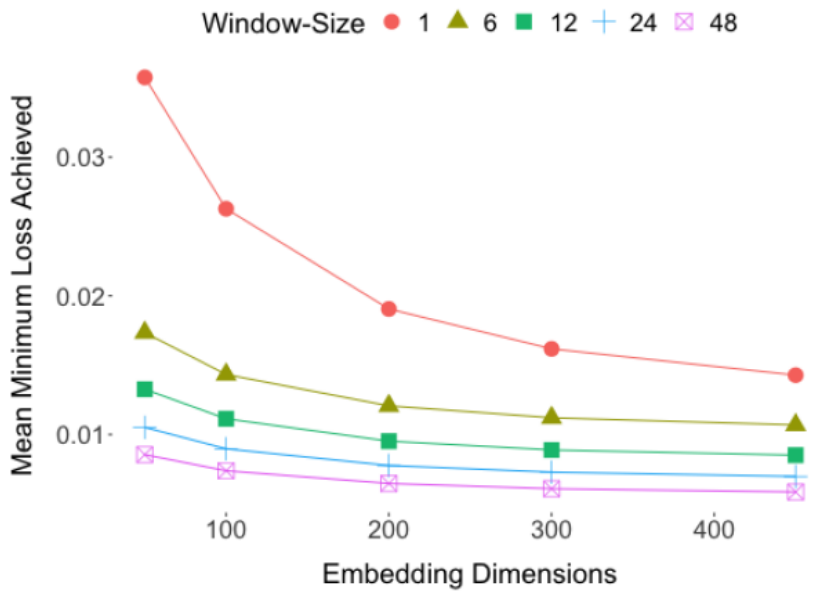
Context Window



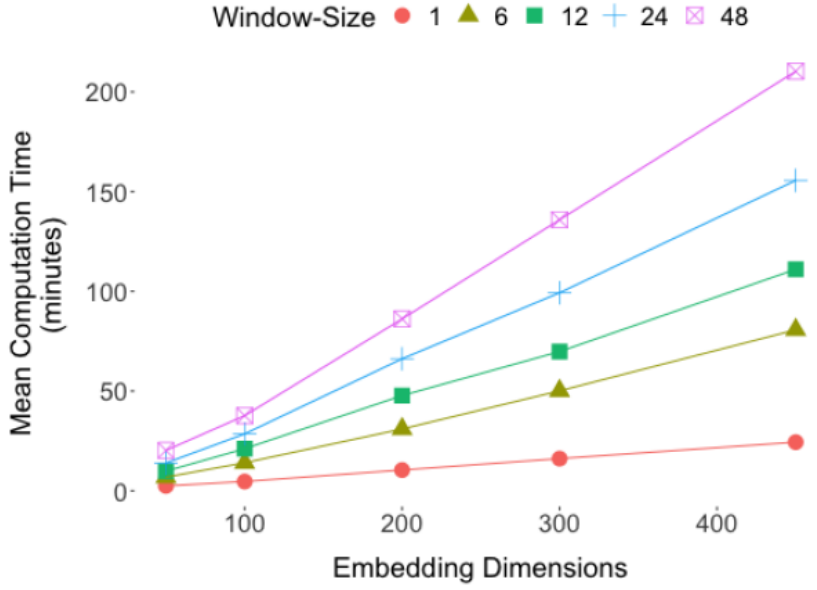
Context Window



Context Window & Embedding Dimensions



(a) Mean Minimum Loss Achieved



(b) Computation Time (minutes)

Rodriguez, Pedro, und Arthur Spirling. 2021. Word Embeddings: What works, what doesn't, and how to tell the difference for applied research. *The Journal of Politics* 2021. https://doi.org/10.1086/715162, Figure 1

Gängige Modelle

- Word2Vec

  Mikolov, Tomas, Kai Chen, Greg Corrado, und Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. https://doi.org/10.48550/arXiv.1301.3781.

- GloVe

  Pennington, Jeffrey, Richard Socher, und Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. https://doi.org/10.3115/v1/D14-1162.

- FastText

  Joulin, Armand et al. 2016. FastText.zip: Compressing text classification models. https://doi.org/10.48550/arXiv.1612.03651.

Vor- und Nachteile

Vorteile

- Semantische Beziehung ähnlicher Worte abgebildet
- Wortbedeutung von Synomymen wird erfasst
- Deutlich besser skalierbar als BoW
- Vortrainierte Modelle sind performativ

Nachteile

- Bias in Modellen aufgrund der Textzusammensetzung in den Trainingsdaten
- Wörter, die nicht im Corpus sind oder entfernt wurden, können nicht abgebildet werden (Ausnahme: Subword Embeddings)
- Polysemie wird nur eingeschränkt berücksichtigt (“Ich arbeite auf der Bank” vs. “Ich sitze auf der Bank”)

Coding time…

Contextual Word Embeddings

aka dynamische Embeddings, Contextualized Embeddings

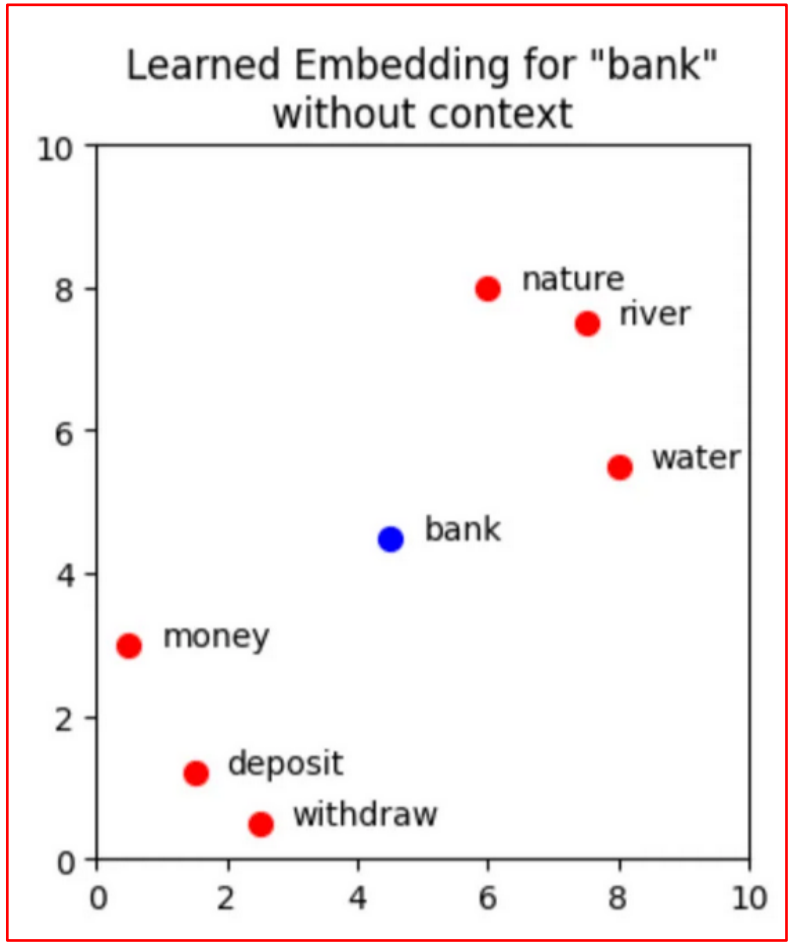
Prinzip

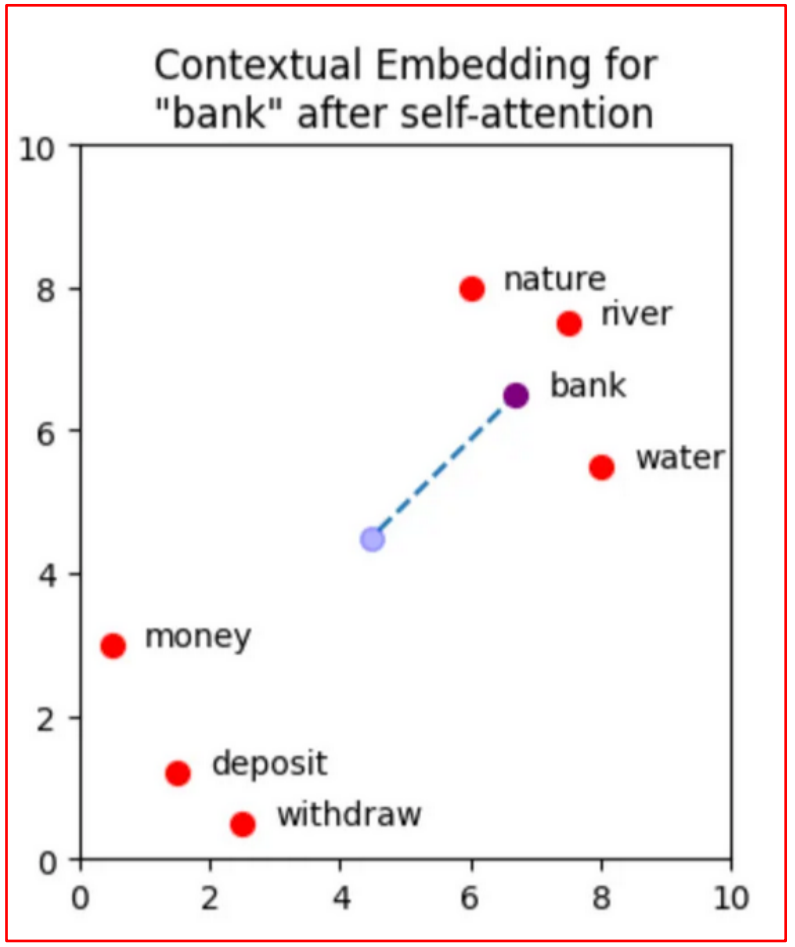
- Embeddings werden für alle Wörter *abhängig* vom Kontext adaptiert

- Weiterentwicklung des Prinzips der Kontextabhängigkeit: Bedeutung des Worts nicht statisch aus dem gesamten Kontext abgeleitet, sondern die Embeddings selbst sind kontextabhängig!

Prinzip

Statische Emebddings





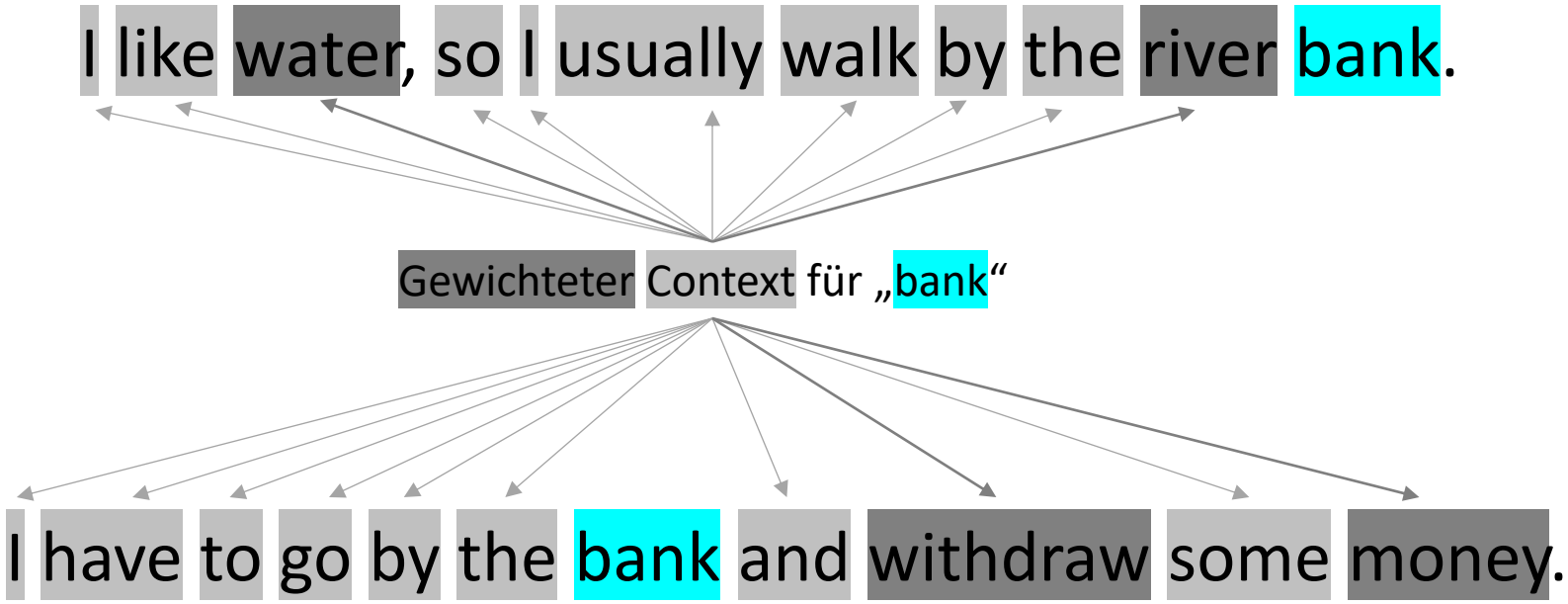
Embedding im Kontext „Natur“

https://towardsdatascience.com/contextual-transformer-embeddings-using-self-attention-explained-with-diagrams-and-python-code-d7a9f0f4d94e/

„Attention is all you need“
Attention-Mechanismus in aller Kürze

- Embeddings werden auf Grundlage der Wörter im Kontext angepasst
- Ähnlichere Wörter (*dot-product similarity*) werden höher gewichtet
- Wörter, welche wenig zum Kontext beitragen (z.B. Füllwörter) werden (idealerweise) weniger stark gewichtet
- Teilberechnungen für unterschiedliche Embedding Dimensionen werden parallel durchgeführt, dadurch sehr performant und nuancierter (*multi-head attention*)

Attention


I like water, so I usually walk by the river bank.
Gewichteter Context für „bank“
I have to go by the bank and withdraw some money.

Transformer

- Parallelisierbar

  Performancegewinn, ermöglicht die Verarbeitung deutlich größerer Datenmengen als zuvor (und damit bessere Embeddings!)

- Encoder/Decoder

  Architektur ist in Encoder und Decoder aufgeteilt, dadurch flexibel

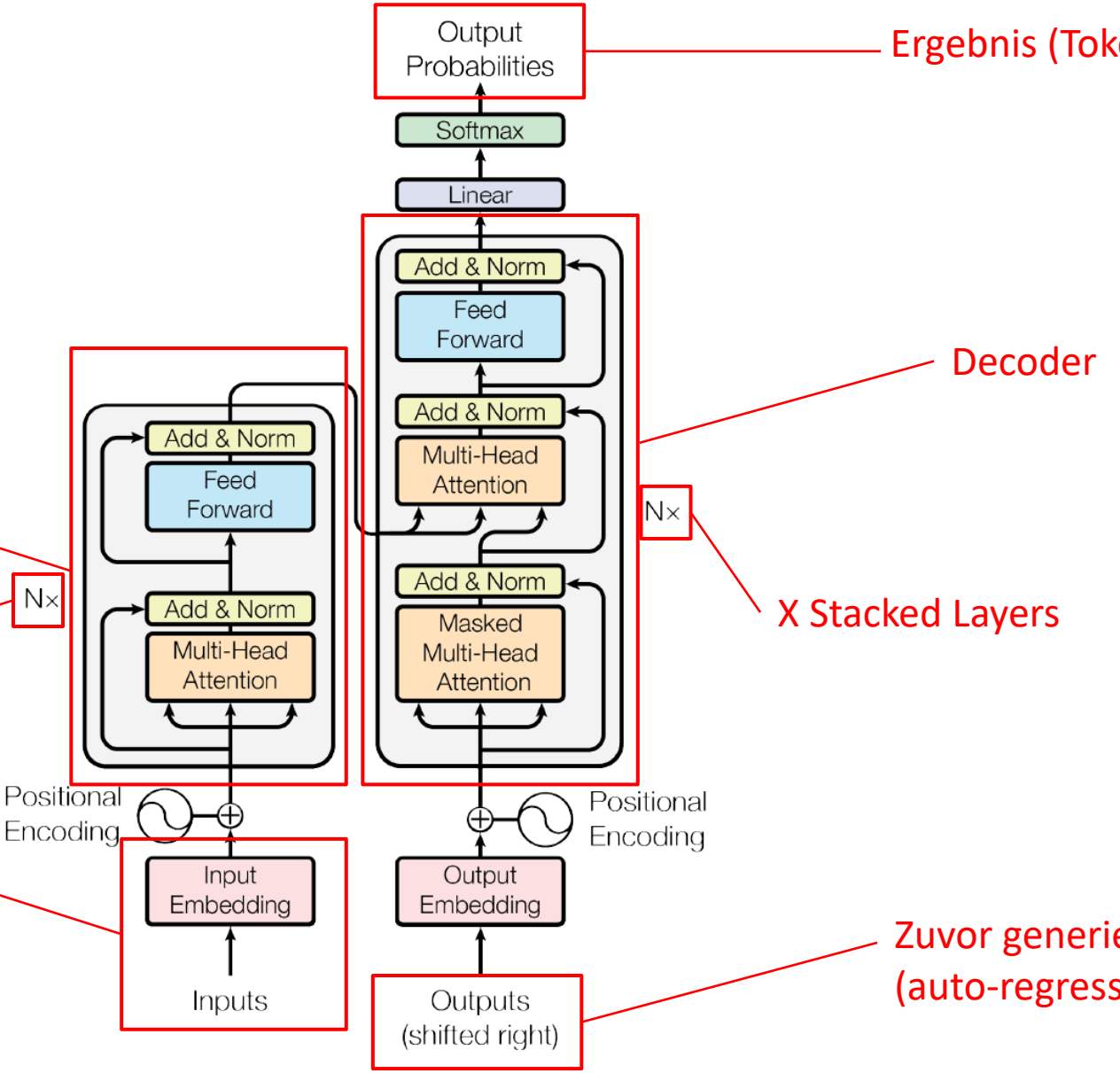
- Output Generation

  Decoder ermöglicht die *Generierung* von Text, z.B. in Chatbots

- Self-supervised learning

  „Textverständnis” wird erzeugt, indem dem Modell beigebracht wird, Wörter in einem Text vorherzusagen (Adaption des Supervised Learning-Ansatzes)

Transformer



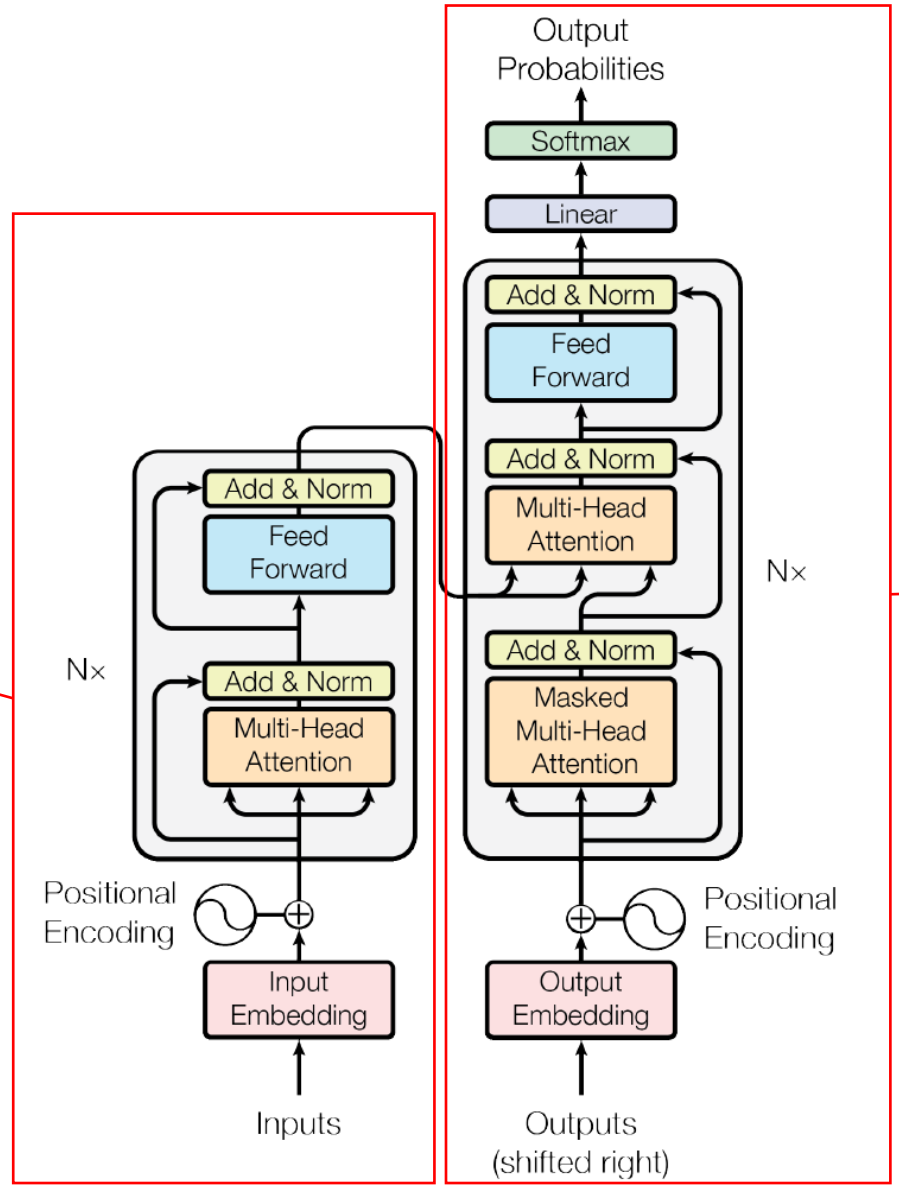
Vaswani, Ashish et al. 2017. Attention Is All You Need. Long Beach, CA, USA. https://doi.org/10.48550/arXiv.1706.03762, Figure 1

Encoder & Decoder

- Encoder: Generiert Embeddings aus Input
  - Keine Generation von Output, nur „verstehen“ des Input
  - Typische Tasks: Classification, Named Entity Recognition
  - Modelle: z.B. BERT, RoBERTa
- Decoder: Generiert Output aus Embeddings des Inputs
  - Generiert Output Wort für Wort auf Grundlage vorheriger Wörter
  - Typische Tasks: Chatbots, Summarisation
  - Modelle: z.B. GPT, Llama
- Encoder + Decoder:
  - Ursprüngliche Transformer Architektur, aber für viele Tasks nicht nötig
  - Typische Tasks: Übersetzung

Transformer

BERT, RoBERTa
(Encoder-only)



GPT, Llama
(Decoder-only)

BERT

- Masked out Training: Das Modell lernt, verstecke Wörter in einer Wortsequenz vorherzusagen – dadurch “Verständnis” in Embeddings

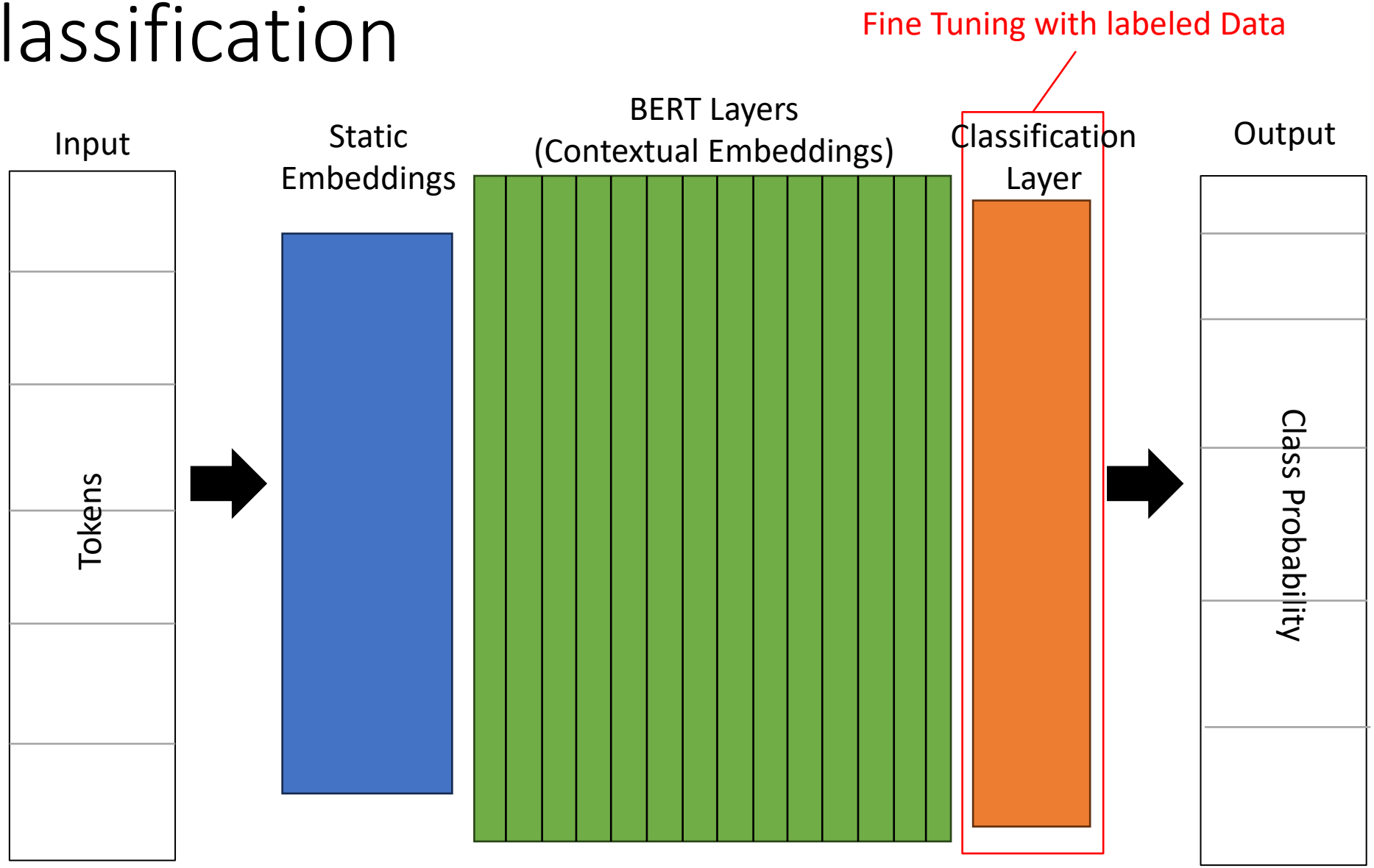
BERT is a MASKED model often used in NLP.

- Zahlreiche Weiterentwicklungen und Adaptionen des Models, z.B. RoBERTa
- Relativ leicht für unterschiedliche Tasks zu trainieren & immernoch eine sehr weit verbreitete Modellarchitektur
- Die Stärke von BERT liegt in seiner ***Anpassungsfähigkeit*** – nicht in seinen out-of-the-box Fähigkeiten. Gut trainierte BERT-Modelle können performanter (und günstiger) sein als LLMs!

Classification

- Bei BERT und co: Supervised Machine Learning Task
- *Transfer Learning*: Baut auf dem *Pre-Training* für Textverständnis auf
- *Supervised Machine Learning*: Das Modell lernt, mittels annotierten Daten latente Kategorien aus Daten vorherzusagen
- Das BERT Modell lernt, die dynamischen Embeddings mit Labeln (z.B. Themen oder Sentiment) zu assoziieren
- Dies geschieht in dem *Classification Layer* – d.h. nur dieser muss trainiert werden, während die anderen Layer aus dem Grundmodell übernommen werden (effizient!)

Classification



Coding time…

Vorbereitung für Tag 2

- OpenAI Account erstellen und Credit aufladen (ca. 5€)
- Ollama installieren