

Künstliche Intelligenz (Sommersemester 2024)

Kapitel 07: Language Models

Prof. Dr. Adrian Ulges



ChatGPT is at capacity right now

Write a rap about the status of ChatGPT.

Yeah, yo, what's up ChatGPT fam
A lot of people trying to jam
But don't worry, we got your back
Just check back soon, we'll get on track

ChatGPT's the place to be
For all your AI chat needs
We're working hard to keep up the pace
So hold tight, we'll be back in this space

So don █

Outline

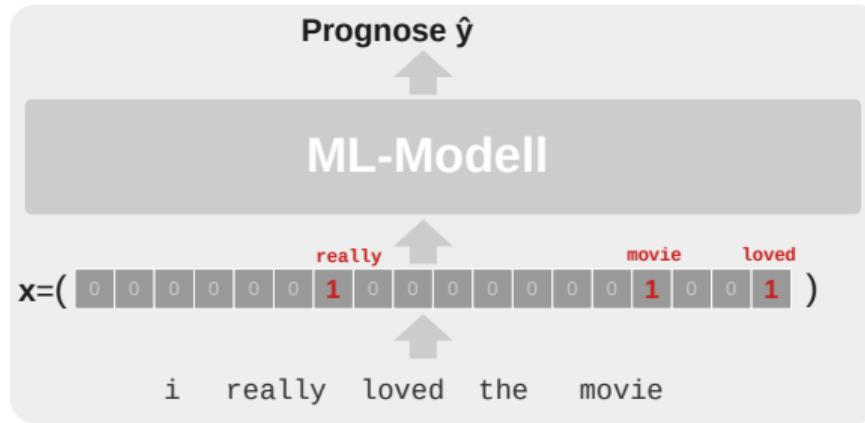


1. Phase 1: Embedding-based Language Models (2013-17)

2. Phase 2: Contextualized Language Models (2018-21)

3. Phase 3: Large Language Models (seit 2022)

Natural Language Processing (NLP)



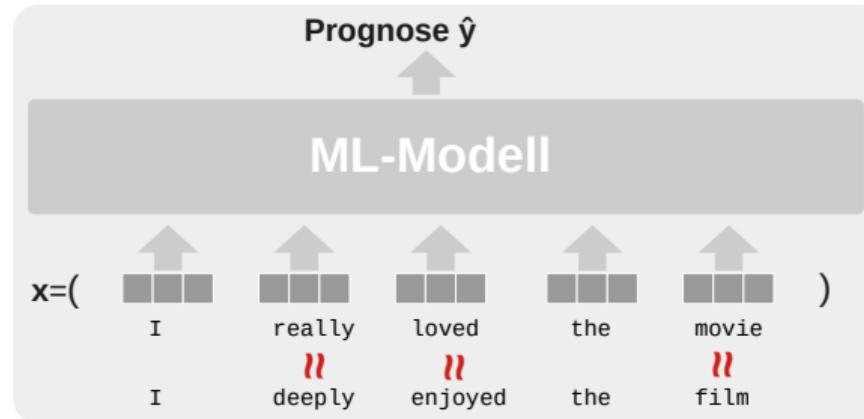
Traditionelle Ansätze

... modellieren Text als Merkmalsvektor x mit **vordefinierten** Merkmalen:

- ▶ Bag-of-Words (*siehe unser News Classifier*)
- ▶ Linguistische Merkmale (*POS-Tags, Suffixe, Großschreibung, ...*)

Nachteil dieses Ansatzes: geringe Robustheit bzgl. Paraphrasierung.

Word Embeddings



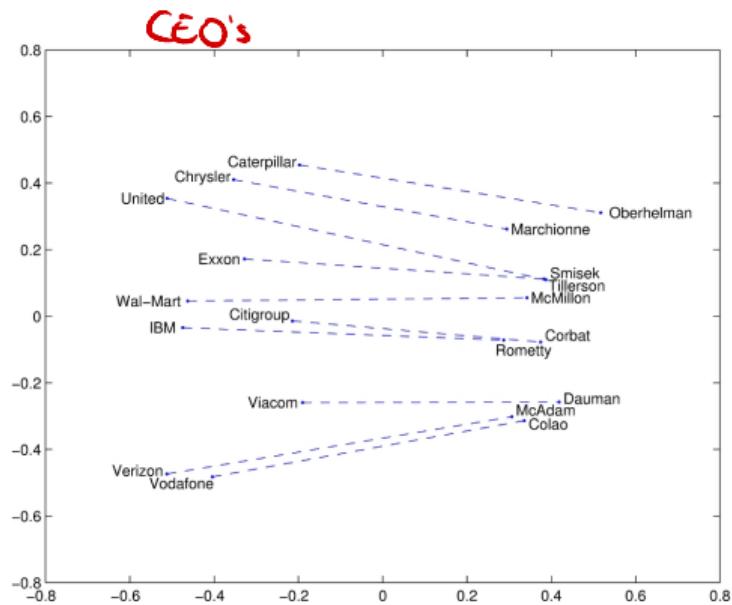
Idee: Repräsentiere jedes Wort mit einem sog. Embedding-Vektor

Dort sind **Bedeutungsaspekte** des Wortes gespeichert:

- ▶ Bedeutungssähliche Worte haben ähnliche Embeddings
- ▶ Beziehungen zwischen Wörtern lassen sich aus den Vektoren ableiten.

Hierdurch **generalisiert** das ML-Modell besser auf Paraphrasierungen.

Word Embeddings: Illustration



- ▶ Links: Semantisch/syntaktisch ähnliche Worte haben ähnliche Embeddings.
- ▶ Rechts: Beziehungen zwischen Wörtern zeigen sich im Embedding-Raum, z.B. als Verschiebungen.
- ▶ In Embeddings ist also ein grundlegendes Sprachverständnis gespeichert.

Wenn gute Embeddings \rightarrow leicht Dokumente zu klassifizieren

Word Embeddings

Formalisierung

- ▶ Für jedes **Wort/Token** t unseres Vokabulars \mathcal{V} speichern wir ein d -dimensionales Embedding (*mit* $d \approx 100\text{--}1000$).
- ▶ Alle Embeddings werden in einer **Embedding-Matrix** $E \in \mathbb{R}^{\#\mathcal{V} \times d}$ gespeichert.

Schlüssefrage: Lernen?

Woher kommen die Werte in der Embedding-Matrix, d.h. das “Sprachverständnis” ?

- ▶ Wir benötigen einen Ansatz, um Sprache von **großen Textkorpora** zu lernen.
- ▶ Dieser Ansatz muss von **ungelabelten** Daten lernen können.
- ▶ Wir trainieren **selbstüberwacht**, mit **Sprachmodellierung** als Hilfs-Task (engl. “auxiliary task”).



Ansatz: Sprachmodellierung

Beispiel für Sprachmodellierung: Was wissen wir über “cobbler”?

A piece of **cobbler** is on the table. [...] Everybody likes **cobbler**. [...] Too much **cobbler** makes you fat. [...] One makes **cobbler** out of fruit and dough.

Definition (Sprachmodell)

Sei t das Auftreten eines Worts in einem Textcorpus, und seien t'_1, t'_2, \dots, t'_k der “Kontext”, d.h. das Auftreten anderer Worte in der Umgebung von t . Dann schätzt ein **Sprachmodell** (engl. “language model”) die **Wahrscheinlichkeit** $P(t|t'_1, t'_2, \dots, t'_k)$.

Beispiel

Too much ? makes you fat.

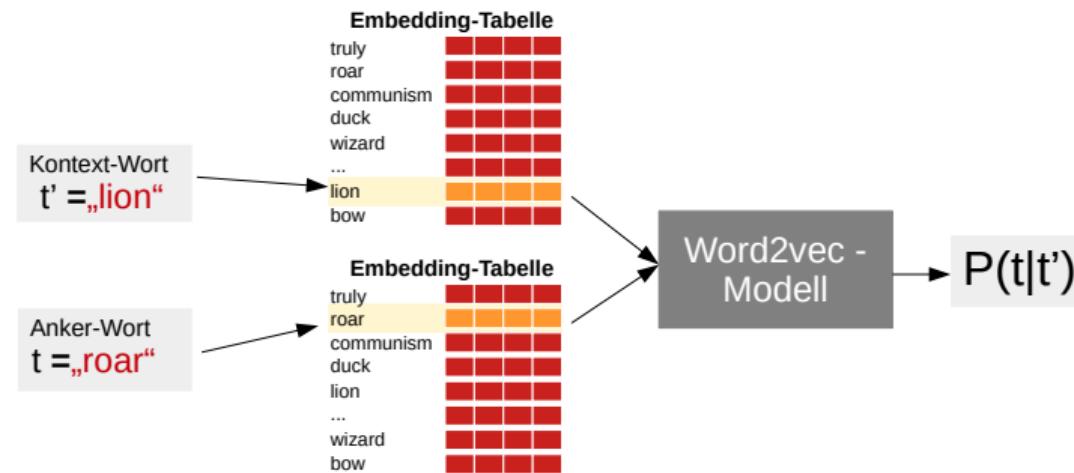
- “pizza”, “cobbler” sind wahrscheinlich.
- “have”, “wind” sind unwahrscheinlich.

Das Word2vec-Modell

Einfache Sprachmodelle auf der Basis von Word Embeddings waren 2013-17 populär (u.a. **Word2vec** [7], **FastText** [2], **GloVe** [8]).

Hier: Word2vec

- ▶ Bei Word2vec besteht der **Kontext aus einem einzigen Wort**.
- ▶ Gegeben **ein Kontextwort t'** , sage das **Ankerwort t** voraus.

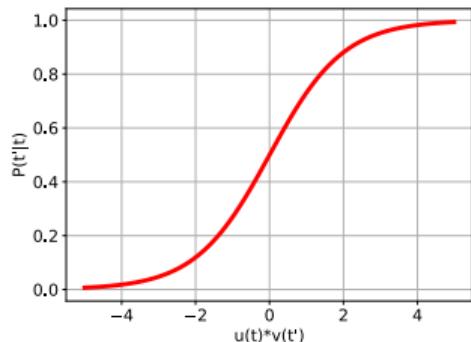
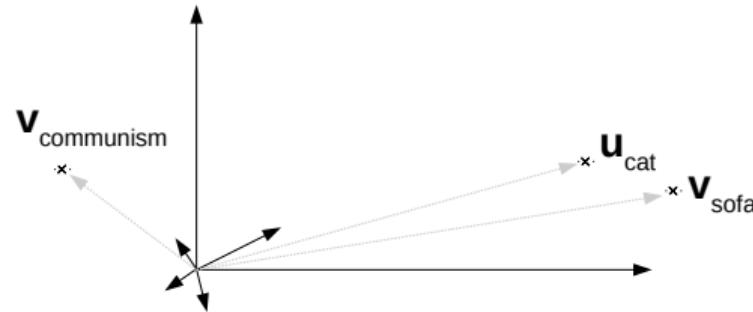


Das Word2vec-Modell [6]

- Wir verwenden zwei Embedding-Tabellen U, V , so dass jedes Wort zwei Embeddings $\mathbf{u}_t, \mathbf{v}_t \in \mathbb{R}^{300}$ besitzt.
- \mathbf{u}_t modelliert die Rolle von t als Kontextwort, \mathbf{v}_t als Ankerwort.

$$P(t|t') := \sigma(\mathbf{u}_{t'} \cdot \mathbf{v}_t) = \frac{1}{1 + \exp(-\mathbf{u}_{t'} \cdot \mathbf{v}_t)}$$

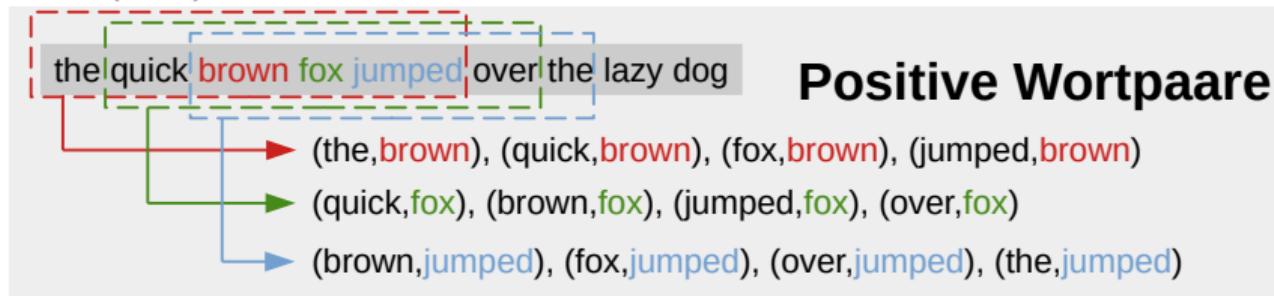
- σ ist die Sigmoid-Aktivierungsfunktion.
- Die Wahrscheinlichkeit $P(t|t')$ steigt mit dem Skalarprodukt der Embeddings:



Word2vec: Training

Als nächstes **lernen** wir die Embedding-Matrizen U, V !

- Gegeben eine Kontextbreite W (z.B. $W=2$), ziehe zufällig **lokale Wortpaare** mit Abstand $\leq W$ aus dem Corpus.
- Jedes Paar (t', t) wird zu einem **Trainingssample**.



- Diese Paare werden mit dem Label $y=1$ versehen ("positive Wortpaare").
- Wir ziehen außerdem zufällige **negative Wortpaare** (mit $y=0$) aus dem Vokabular.

Negative Wortpaare

(cat,communism), (grief, sofa), (chicken, refuse), (donor, withdraw)
(reduce, sustain), (lost,labor), (wash,charismatic), (pupil,rally)



Das Word2vec-Modell: Training

Wir leiten die Lernformeln mit **stochastischem Gradientenabstieg** her:

Das Word2vec-Modell: Training





Textcorpus

superman is a hero .
superman saves the world .
lexluthor is *superman* 's nemesis .
lexluthor is a supervillain .
lexluthor wants to conquer the world .
xavier is a hero .
xavier saves the world .
magneto is *xavier* 's nemesis .
magneto is a supervillain .
magneto wants to conquer the world .

Outline



1. Phase 1: Embedding-based Language Models (2013-17)

2. Phase 2: Contextualized Language Models (2018-21)

3. Phase 3: Large Language Models (seit 2022)

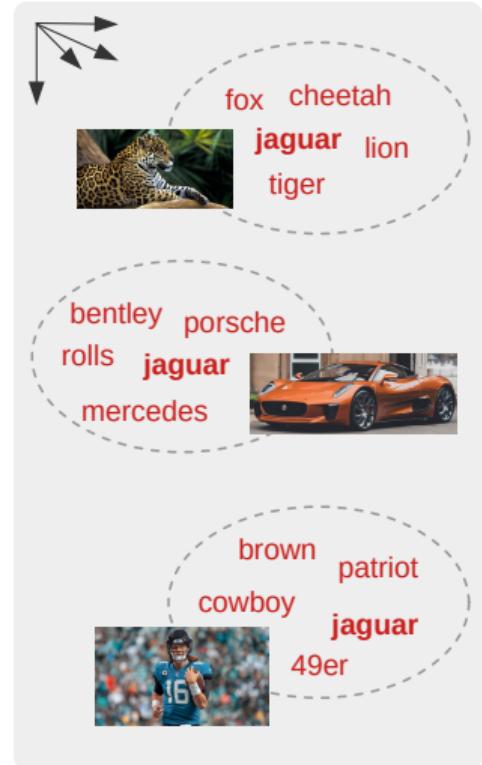


Word Embeddings: Nachteil

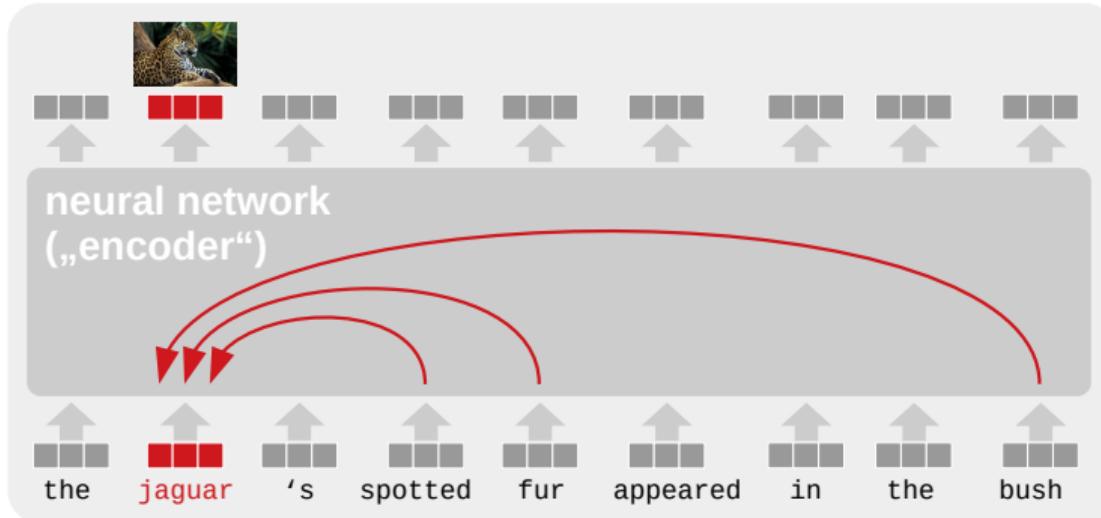
Wörter haben **verschiedene Bedeutungen**, abhängig von ihrem **Kontext**. Wie häufig tritt dieses Problem auf...?
→ sehr häufig 😞

Beispiele

- ▶ Homonyme ("jaguar", "dog", ...)
- ▶ Redewendungen ("unter die Lupe nehmen")
- ▶ Koreferenzen ("... she ...")
- ▶ ...



Ansatz: Kontextualisierung von Word Embeddings

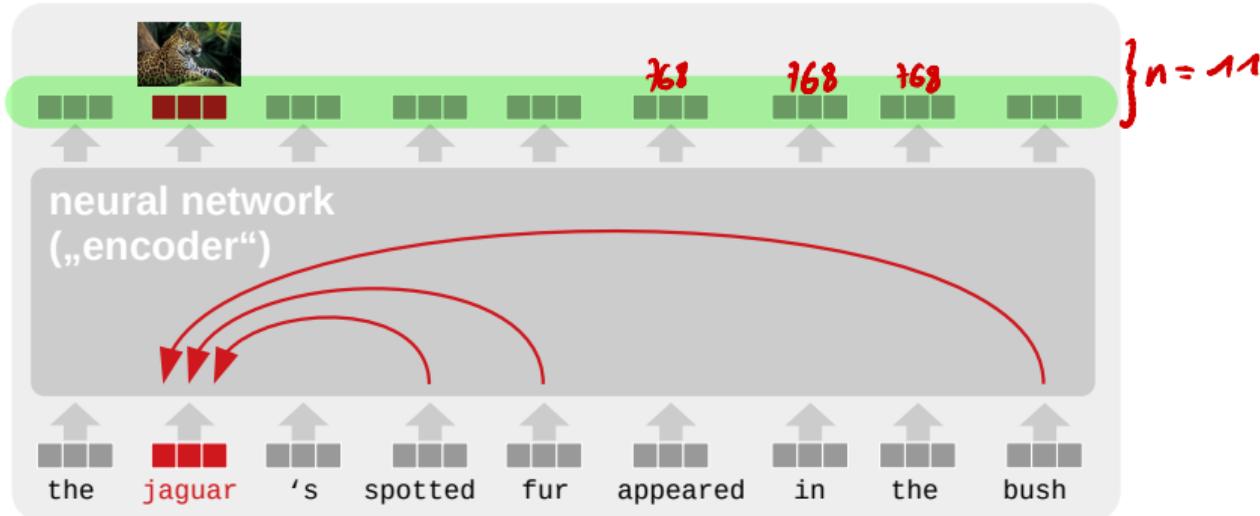


- ▶ Eingabe: Ein kompletter **Kontext / Satz / Textblock**.
- ▶ Worte werden zunächst durch **Word Embeddings** (unten) repräsentiert.
- ▶ Diese Word Embeddings werden durch ein neuronales Netz (sog. "Encoder") **kontextualisiert**. Der Kontext wirkt auf jedes Embedding ein und **disambiguier**t es.

Ansatz: Kontextualisierung von Word Embeddings



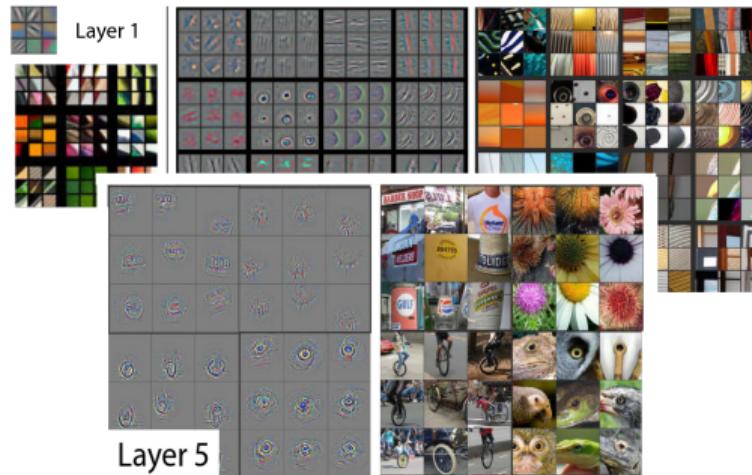
[1,11,768]
batchsize



- ▶ Sehr aktiv beforscht seit 2018 [9, 12, 10, 3, 4, 5, 11], viele Arbeiten verwenden die sogenannte **Transformer**-Architektur basierend auf **Attention**.
- ▶ Transformer **lernen** sowohl ihre Word-Embedding-Matrix als auch die Parameter ihres Encoders mittels Language Modeling (*kommt gleich...*).

Transformer sind tiefe Netze Bild: [13]

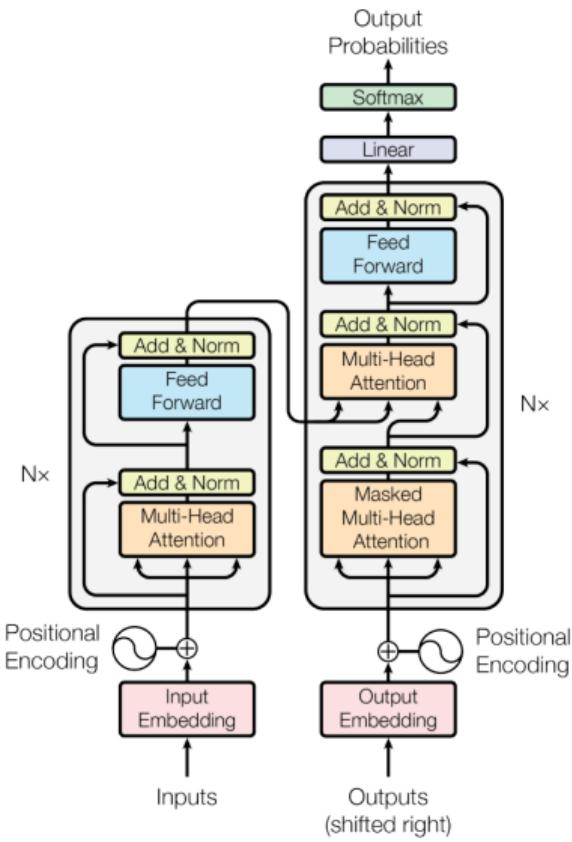
- ▶ Im Jahr 2012 begann der Hype um Deep Learning im Bereich der Bilderkennung.
- ▶ Tiefe neuronale Netzwerke **stapeln Schichten** übereinander:
Der Output von Schicht l bildet den Input für Schicht $l+1$.
- ▶ Dadurch können untere Schichten **einfache** Merkmale erkennen.
Obere Schichten setzen diese Merkmale dann zu komplexeren **zusammen**.



Transformer-Netze

Bild: [12]

- Die sogenannten **Transformer** sind der Backbone für state-of-the-art NLP.
- Transformer wurden 2017 von Vaswani et al. im Paper "**Attention is all You Need**" [12] vorgeschlagen ($>120.000\times$ zitiert!!!).
- Im Gegensatz zu früheren Modellen fokussieren Transformer stark auf Attention (siehe Titel des Papiers).
- Transformer sind **tief** (12 bis >100 "**Transformer-Blöcke**").
- Der originale Transformer ist ein sog. "Sequence-to-Sequence"-Modell für **Maschinenübersetzung**. Wir werden uns jetzt nur auf seinen **Encoder** (= linker Teil) konzentrieren. //*Input: englischer Text
Output: deutscher Text*
- Transformer werden als Sprachmodelle **vortrainiert** (engl. "pre-training") und können dann auf spezielle Tasks angepasst werden ("fine-tuning").



Attention: Idee

Beispiel: Kontextualisierung von „the language“

I used to live in France, close to the Western coast. I took no classes and found it difficult to get in touch with locals, and with only basic school knowledge from 20 years ago it took me about three years to learn the language properly.

Beobachtung

- Worte im Text folgen einer eher **losen Anordnung**, und entfernte Worte können Einfluss auf die Interpretation eines Wortes haben.

Ansatz

- Idee: Jedes Wort X ermittelt welchen anderen Worten Y im Satz es **Aufmerksamkeit** schenkt (“Attention”).
- Diese anderen Worte werden dann in das Embedding von X “hineingebacken” .

(Self-)Attention: Veranschaulichung

Referenz-
Sequenz

„the jaguar ‘s spotted fur appeared in the bush“



Ziel-
Sequenz

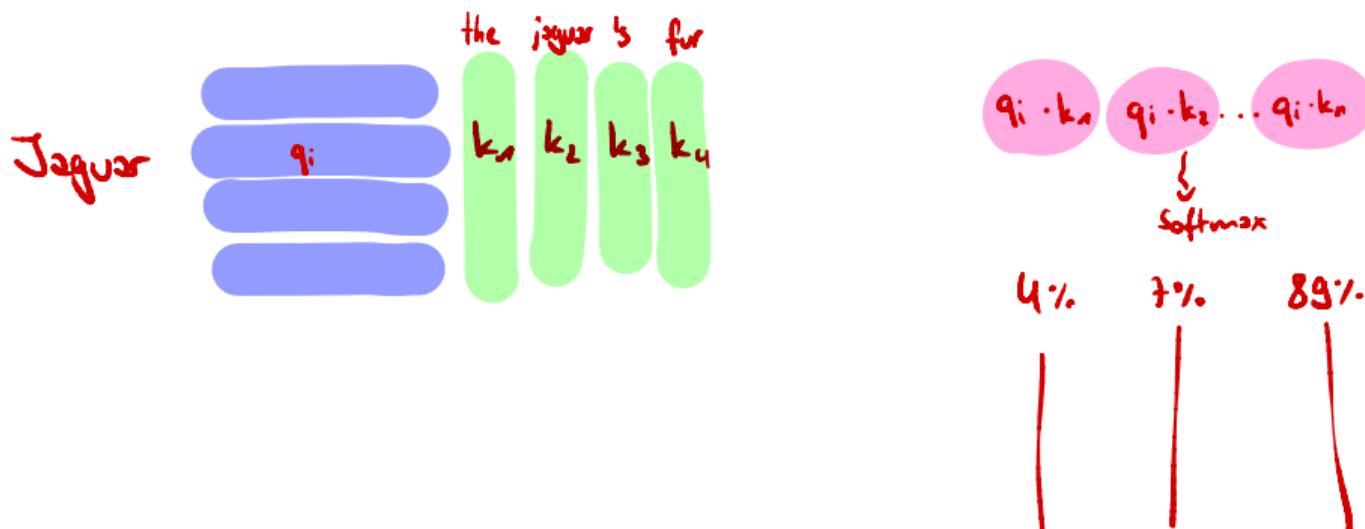
„the jaguar ‘s spotted fur appeared in the bush“

(Self-)Attention: Formalisierung

- ▶ Eingabe: Embeddings x_1, \dots, x_n *The Jaguar*
- ▶ Wir teilen diese Embeddings in drei verschiedene Embeddings auf:
 - ▶ Queries q_1, \dots, q_n
 - ▶ Keys k_1, \dots, k_n
 - ▶ Values v_1, \dots, v_n
- ▶ Die Embeddings interagieren miteinander:

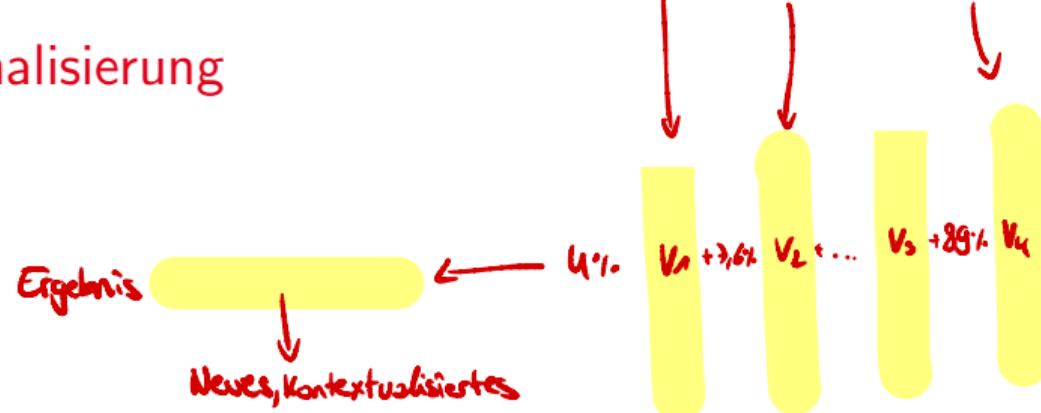
gelernt

$$\begin{aligned} q_i &= W \cdot x_i \\ k_i &= W \cdot x_i \\ v_i &= W \cdot x_i \end{aligned}$$

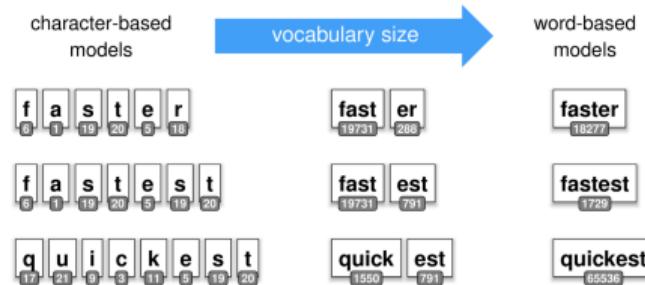


(Self-)Attention: Formalisierung

*



Eingabe von Transformer-Netzwerken: Tokenization



- ▶ Anstatt "roher" Einzelwörter geben wir **Teilwort-Token** in den Transformer ein.
- ▶ Diese werden aus Textcorpora mit sogenanntem **Bytepair Encoding** gelernt.
- ▶ Es gibt mehrere Varianten: BPE, WordPiece, Unigram, SentencePiece¹.

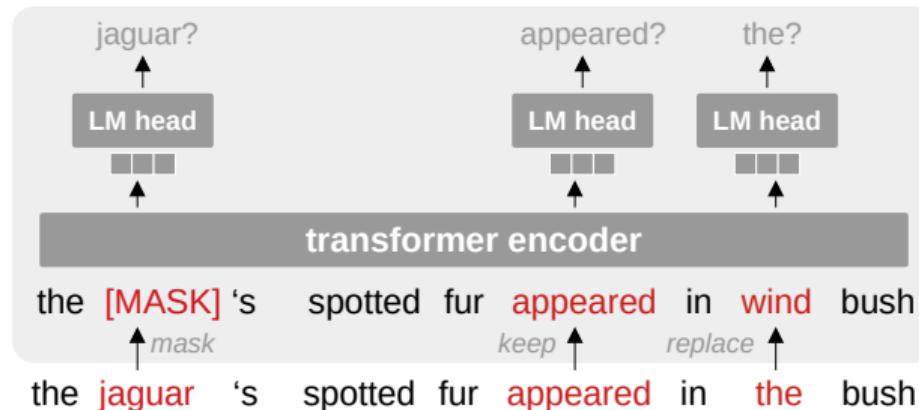
Vorteile?

- ▶ Der Vokabular – und somit die Embedding-Matrix E – ist kleiner. ☺
- ▶ Bessere Generalisierung auf Wörter außerhalb des Vokabulars
(z.B. *versicherung / s / gesellschaft*). ☺

¹https://huggingface.co/transformers/tokenizer_summary.html

Masked LM: z.B. BERT [3]

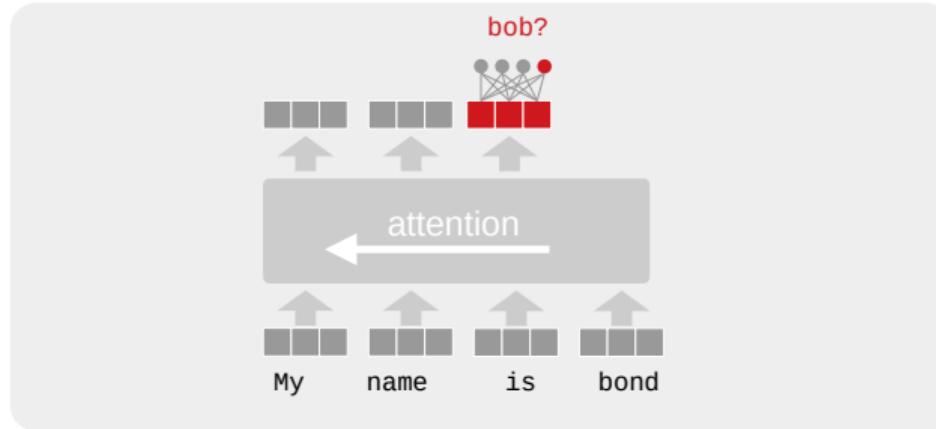
Transformer **lernen** in zwei Sprachmodell-Tasks (Masked LM vs- autoregressives LM)



Masked LM

- ▶ Zufällige **Tokens** werden maskiert/ersetzt und dann durch das Modell vorhergesagt.
- ▶ Vorteil: Das Modell kann **bidirektional** (d.h. nach links und rechts) "attend".
- ▶ Anwendungszweck: Gute Embeddings für Wörter erhalten, um sie durch weitere Modelle zu verarbeiten.

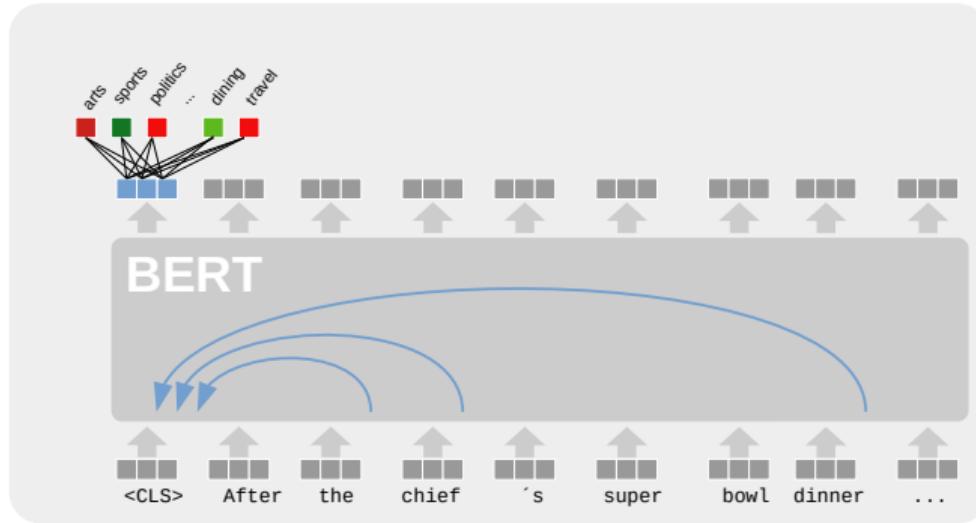
Autoregressives (Left-to-right) LM



Autoregressives LM

- ▶ Gegeben eine Sequenz t_1, \dots, t_n , sage das **nächste Token** t_{n+1} vorher.
- ▶ Solche Modelle können **Text generieren** ☺, aber keinen Kontext **rechts** des aktuellen Tokens berücksichtigen ☹.
- ▶ Prominente Beispiele: Die Generativen Pretraining Transformers (**GPTs**): GPT1, GPT2, GPT3, ChatGPT, GPT4 ...

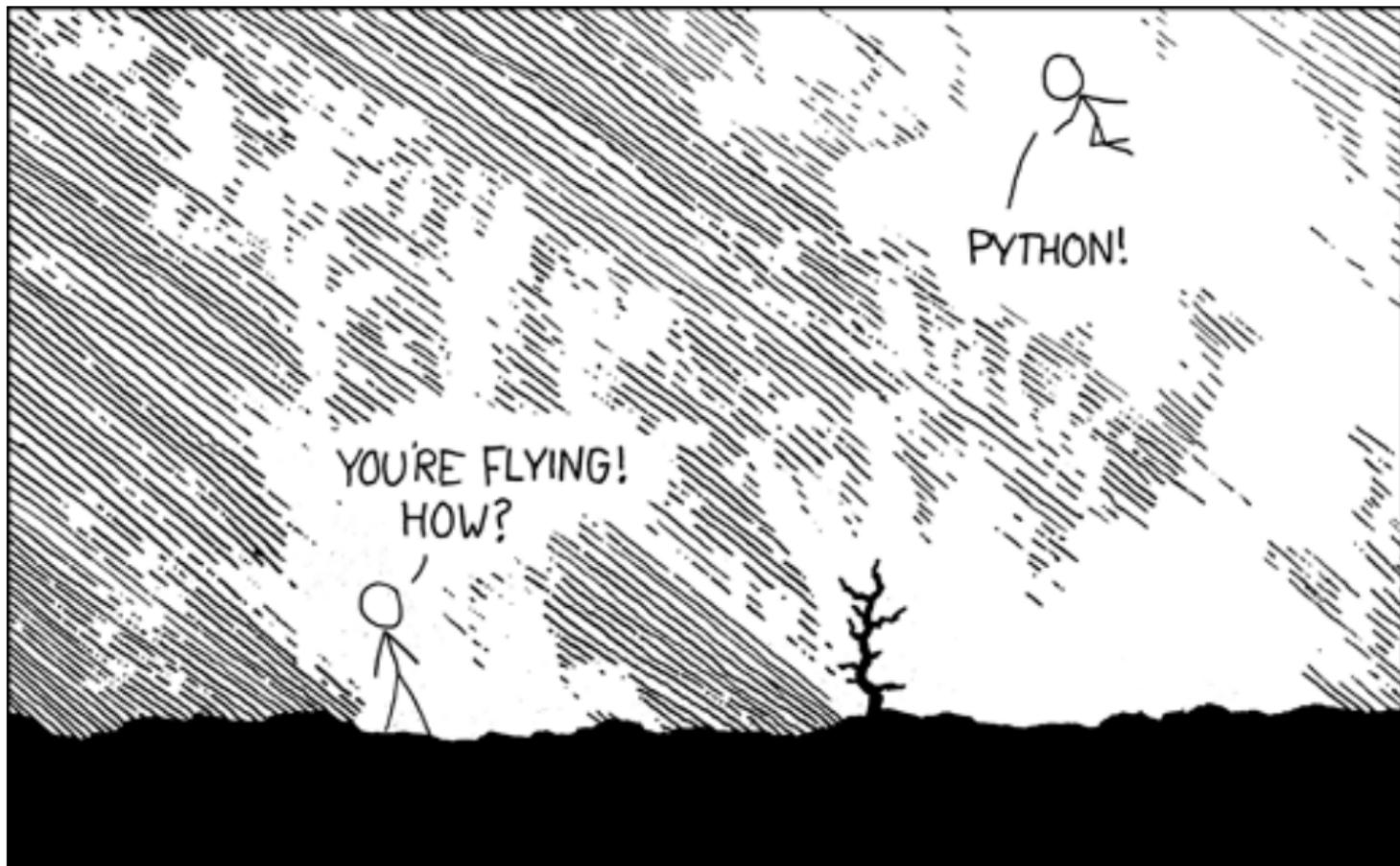
Transformer auf spezielle Tasks anpassen



Ein Transformer-basierter News Classifier

- ▶ Nehme das Embeddings des Start- (oder sog. “CLS”-)Tokens.
- ▶ Füttere dieses Embedding in ein fully connected layer.
- ▶ Wir fine-tunen dieses Layer und den kompletten Encoder.

Notebook: Transformer 101



Outline

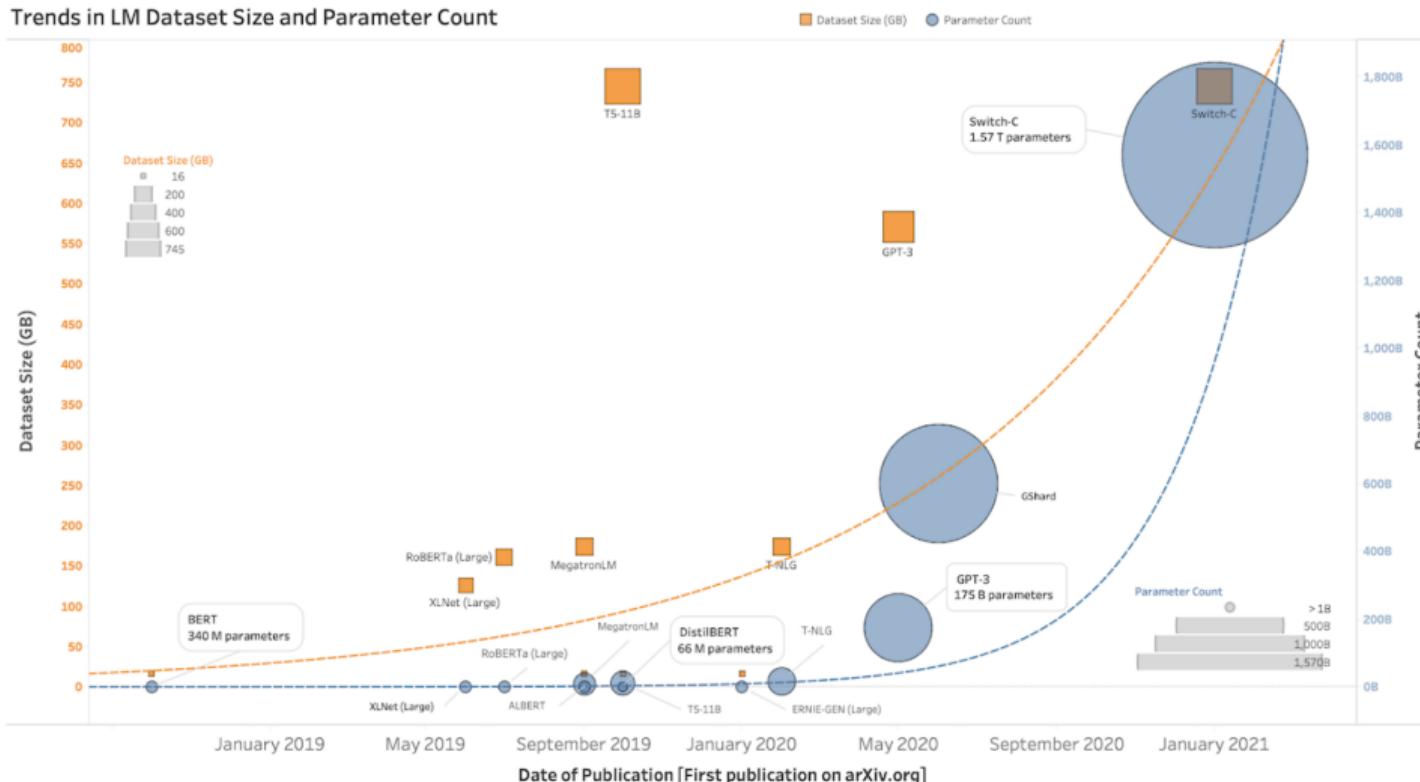


1. Phase 1: Embedding-based Language Models (2013-17)
2. Phase 2: Contextualized Language Models (2018-21)
3. Phase 3: Large Language Models (seit 2022)

Seit 2018-22: Transformer wurden immer größer (und besser...)

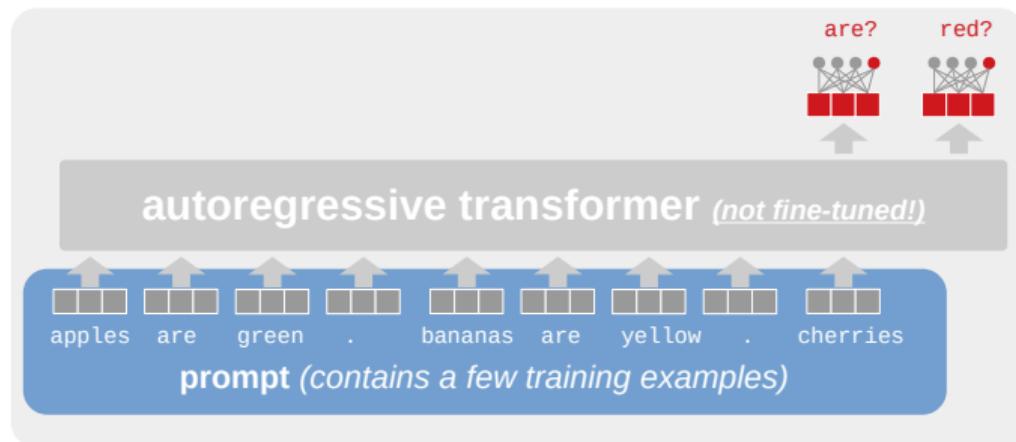


Bild: [1]



Few-shot und Zero-shot Learning

- ▶ LM ist ein äußerst leistungsstarker Pre-training Task!
- ▶ Große Transformer können sogar ohne Fine-tuning generelle Aufgaben lösen.
- ▶ Dies geht per **Few-shot Prompting**: Wir geben dem Modell einfach ein paar Trainingsbeispiele als Teil der Eingabe.



Wenn die Aufforderung gar keine Beispiele enthält, ist das Modell ein sog. **Zero-shot-Lerner** (vgl. ChatGPT).

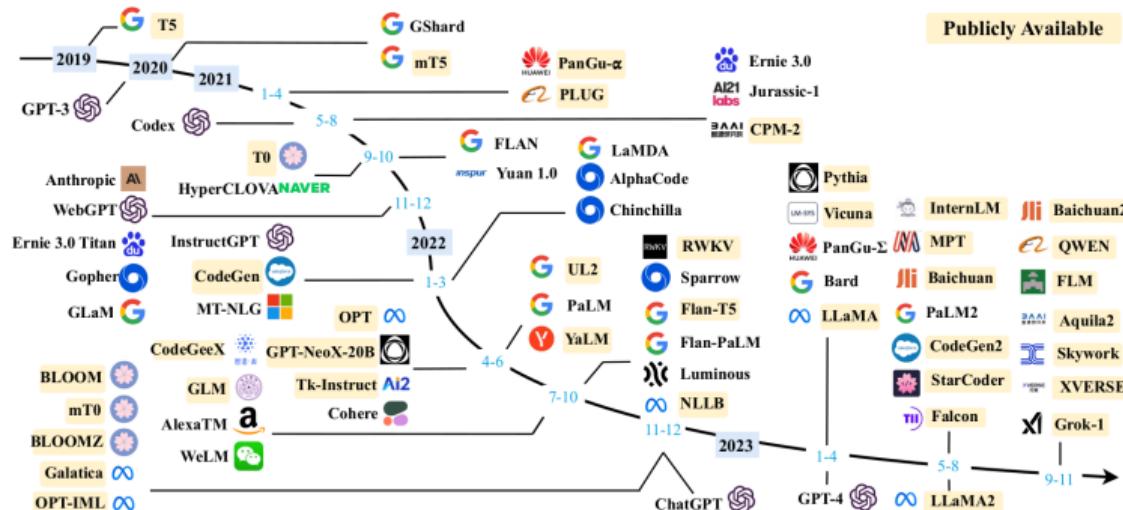
Language Models are Few-Shot Learners

Tom B. Ervin^{*} Rajanikant Manohar^{*} Nick Ryder^{*} Melanie Seltman^{*}
Jesse Kaplan^{*} Pratiksha Bhambhani^{*} Arvind Neelakantan^{*} Prafull Shyam^{*} Girish Savani
Anmol Adai^{*} Sandhini Agarwal^{*} Arpit Bhagavatula^{*} Gresham Krueger^{*} Tom Houghton
Brown Child^{*} Aditya Ramesh^{*} Daniel M. Ziegler^{*} Jeffrey Wu^{*} Clemens Winter^{*}
Christopher Home^{*} Mark Chen^{*} Eric Sigler^{*} Matheus Litoia^{*} Scott Gray^{*}
Benjamin Chase^{*} Jack Clark^{*} Christopher Berndt^{*}
Sam McCandlish^{*} Alex Radford^{*} Ilya Sutskever^{*} David Amodei^{*}
OpenAI

Abstract
Recent work has demonstrated substantial gains on many NLP tasks and benchmarks by pre-training on a large corpus of text followed by fine-tuning on a specific task. While typically task-specific, it is often the case that such tasks will require thousands of examples, while humans can learn from only a few. In this paper, we show that scaling up language models greatly improves task-agnostic, few-shot performance, sometimes even matching competitors with prior state-of-the-art fine-tuned models. We do this by training a new sparse language model, and test its performance on the zero-shot setting. Our new model, GPT-3, is trained on a large dataset of prompts along with full and few-shot demonstrations sped fast per-prompt via text interaction with the model. GPT-3 achieves state-of-the-art performance on a wide range of benchmarks, including text generation and cloze tests, as well as several tasks that require on-the-fly reasoning or domain adaptation, such as in-domain few-shot learning and cross-domain few-shot learning. In addition to these quantitative results, we also identify some datasets where GPT-3's few-shot learning skill struggles, as well as some datasets where GPT-3 fails methodologically related to training on large web corpora. Finally, we report on a few-shot study comparing GPT-3 to humans, and find that GPT-3 is able to quickly distinguish less articles written by humans. We discuss broader societal impacts of this finding and of GPT-3's general.

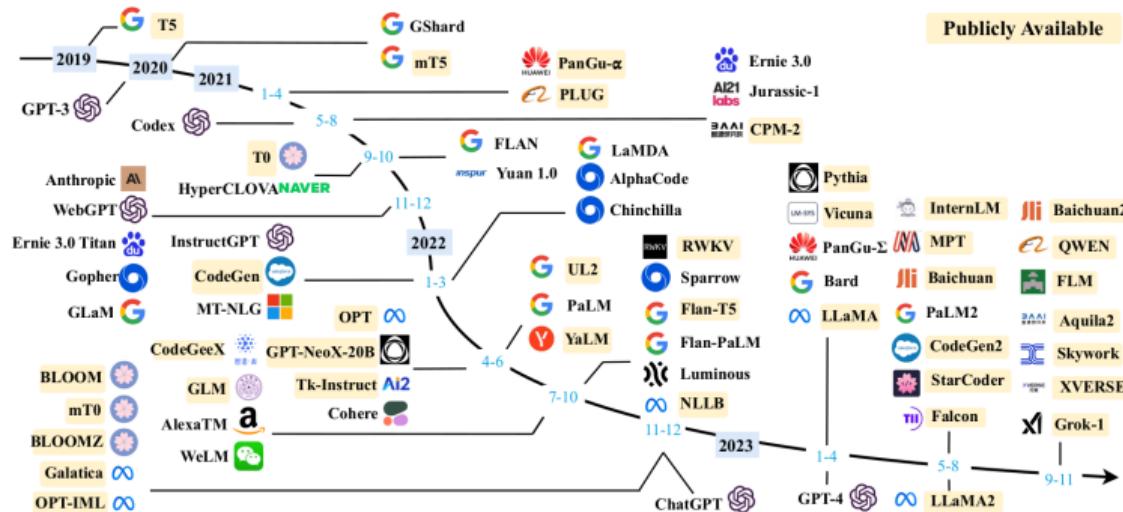
*Equal contribution
John Hopkins University, OpenAI
Author contributions: Israel et al. out of 20 pages

Überblick der aktuellen Entwicklung [14]



- Derzeit: mehrere neue “Large” (d.h. > 10 Mrd. Parameter) LMs **jede Woche**.
- Diese sind entweder (a) “nur” per **LM Pre-Training** trainiert, oder (b) außerdem per speziellem **“Instruction Tuning”** darauf gefinetunet, Nutzerfragen zu beantworten.

Überblick der aktuellen Entwicklung [14]



- Open-source holt schnell auf! (Bsp. Llama-3) ☺
- Empfehlenswertes Paper: Zhao et al., “A Survey of Large Language Models” [14], auch auf [Github](#).
- Empfehlenswerter Link: [Huggingface Leaderboard](#).



References I

- [1] Bender, Emily.
On the dangers of stochastic parrots: Can language models be too big?, keynote at allen institute.
<https://www.youtube.com/watch?v=N5c2X8vhfBE>, retrieved: Oct2021.).
- [2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov.
Enriching word vectors with subword information.
TACL, 5:135–146, 2017.
- [3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova.
BERT: pre-training of deep bidirectional transformers for language understanding.
CoRR, abs/1810.04805, 2018.
- [4] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer.
Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, 2019.
- [5] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov.
Roberta: A robustly optimized BERT pretraining approach.
CoRR, abs/1907.11692, 2019.
- [6] McCormick, Chris.
A Primer on Neural Network Models for Natural Language Processing.
<http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/>, retrieved: Apr 2017).
- [7] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean.
Distributed Representations of Words and Phrases and their Compositionality.
In Advances in Neural Information Processing Systems 26, pages 3111–3119. Curran Associates, Inc., 2013.
- [8] Jeffrey Pennington, Richard Socher, and Christopher D. Manning.
Glove: Global vectors for word representation.
In EMNLP, pages 1532–1543. ACL, 2014.

References II



- [9] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer.
Deep contextualized word representations.
In Proc. of NAACL, 2018.
- [10] Alec Radford.
Improving language understanding by generative pre-training.
2018.
- [11] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu.
Exploring the limits of transfer learning with a unified text-to-text transformer, 2019.
- [12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin.
Attention is all you need.
In Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.
- [13] Matthew D. Zeiler and Rob Fergus.
Visualizing and Understanding Convolutional Networks.
CoRR, abs/1311.2901, 2013.
- [14] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen.
A survey of large language models, 2023.