FH JOANNEUM
University of Applied Sciences

Mehr als nur ChatGPT: Large Language Models für KMU

Fundierte Einführung in die Welt der LLMs

Mag.ª Marie-Christine Maierbrugger MSc, DI Dr. Hannah Wimmer BSc.
Data Science and Artificial Intelligence
Institut für Wirtschaftsinformatik und Data Science, FH Joanneum
Eckertstraße 30i, 8020 Graz
{marie-christine.maierbrugger, hannah.wimmer}@fh-joanneum.at

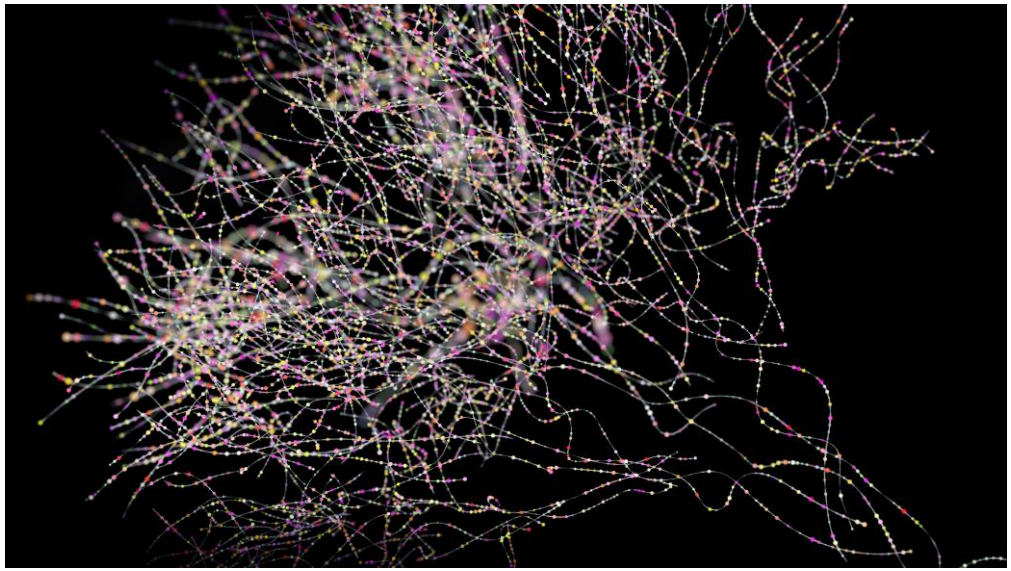
FH JOANNEUM
University of Applied Sciences

Large Language Models für KMU

Fundierte Einführung in die Welt der LLMs

Bevor wir anfangen…

- Was ist Ihr aktuelles **Verständnis** von LLMs?
- Was wären Ihrer Meinung nach **Anwendungsgebiete** von LLMs?
- Wie **funktionieren** LLMs Ihrer Meinung nach?
- Was sind **Möglichkeiten** bzw. **Grenzen** von LLMs?



FH JOANNEUM
University of Applied Sciences

Large Language Models für KMU

Fundierte Einführung in die Welt der LLMs

Kursüberblick – erster Teil (Wimmer)

- Einführung in Large Language Models (LLMs) & entsprechende ‚Lingo'
  - „Wie funktioniert das überhaupt?"
  - Tokenization & Embeddings
  - Attention & Transformers
  - Chatbot-Charakter & Prompts/Fine-Tuning
- Chancen & Risiken
  - Bias, Halluzinationen, etc.
  - Datenschutz & regulatorische Aspekte
  - Future Perspectives → God-like AI?
  - Überkonfidenz & „Sykophantische AI"

(zweiter Teil (Maierbrugger): technische Aspekte)

FH JOANNEUM
University of Applied Sciences

MEHR ALS NUR CHATGPT:
LARGE LANGUAGE MODELS FÜR KMU

1. EINFÜHRUNG IN LARGE LANGUAGE MODELS (LLMS)

„Wie funktioniert das überhaupt?“

- *Tokenization & Embeddings*
- *Attention & Transformers*
- *Chatbot-Charakter*

Large Language Models für KMU

Einführung in Large Language Models (LLMs)

LLMs sind riesige („*large*“) neuronale Netzwerke, trainiert auf Unmengen von (oft problematisch erworbenen) Daten.

Modelle wie ChatGPT:

- Generieren Text für uns / lesen sie Korrektur
- Erklären uns Konzepte (bspw. höhere Mathematik)
- Unterhalten sich mit uns („Chatbots“)
- etc.

→ **Frage:** Was ist aber eigentlich das *Ziel* von LLMs? Worauf werden diese Modelle „hintrainiert“?

Large Language Models für KMU

Einführung in Large Language Models (LLMs)

Das eigentliche Ziel von LLMs ist **die Vorhersage des wahrscheinlichsten nächsten Wortes in einem Satz.**

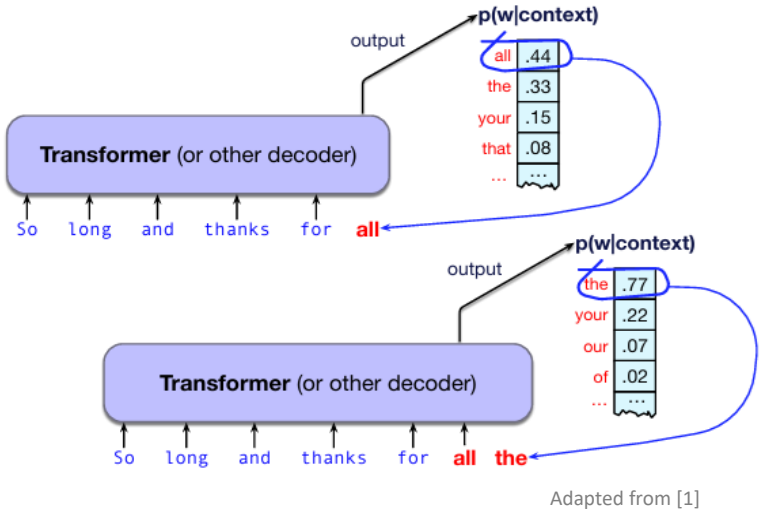
„Hallo! Freut mich Sie …“

→ „… kennenzulernen“

Das Modell:

- gibt eine **Wahrscheinlichkeitsverteilung** über alle bekannten Wörter aus
- selektiert das **wahrscheinlichste** nächste Wort
- schreibt so **iterativ** zusammenhängenden Text: Wort für Wort

→ Iterative Prädiktion „von links nach rechts“
(kausale bzw. autoregressive LMs)



Adapted from [1]

Realitätscheck

→ Das Modell erzeugt keine Antwort im „menschlichen Sinn“
wobei: menschliche Sprachverarbeitung ist ja auch noch völlig unklar…

→ Das Modell hat (noch) kein kognitives Verständnis des Inhalts
wobei: menschliches ‚Verständnis‘ ist ja auch noch völlig unklar…

[1] Chapter 7 - Speech and Language Processing. Daniel Jurafsky & James H. Martin. Copyright © 2026. All

FH JOANNEUM
University of Applied Sciences

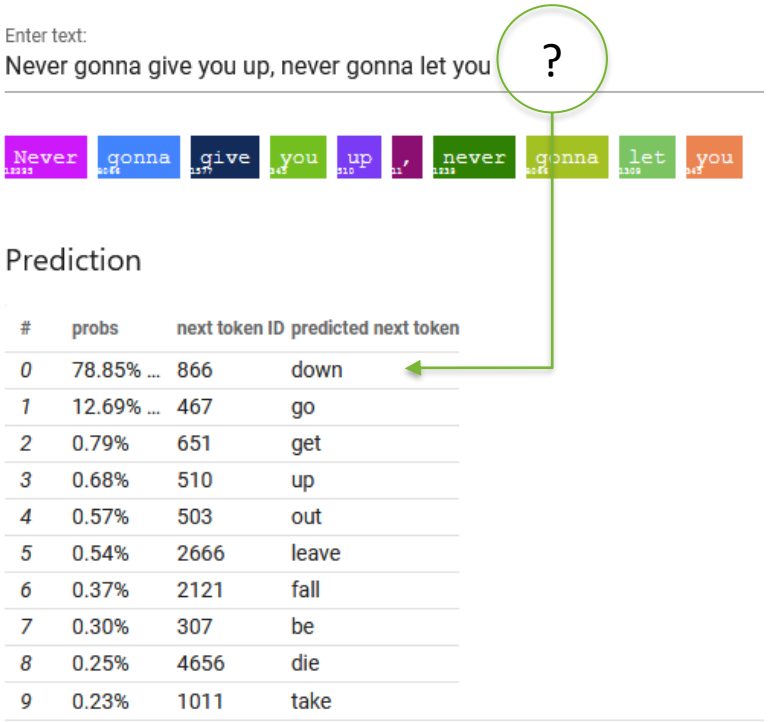
Large Language Models für KMU

Einführung in Large Language Models (LLMs)

Kurze Demo hierzu:

- https://alonsosilva-nexttokenprediction.hf.space/

- Die Demo verwendet GPT-2 (124M Parameter)
- Es werden die zehn wahrscheinlichsten nächsten Wörter ausgegeben
- Wir sehen die jeweilige Wahrscheinlichkeit (‚probs'), das korrespondierende Wort, …
- …und einiges mehr (Details folgen!)

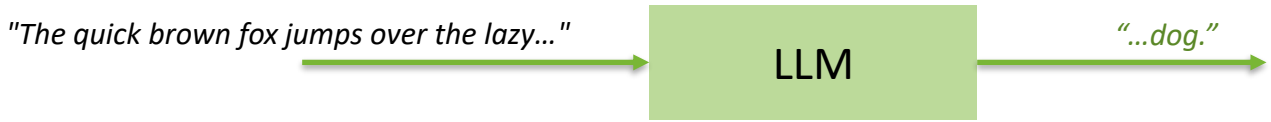


[1] Next token prediction visualization (GPT-2)

Large Language Models für KMU

Einführung in Large Language Models (LLMs)

Im Prinzip haben wir mit einem **Language Model** also folgende Situation:



Das wirft allerdings sogleich einige Fragen auf…

1. Wie bekommen wir den Input (natürliche Sprache) überhaupt ins Modell?
2. Wie kann ein solches Modell genau aussehen?
3. Wie kommen wir von der Prädiktion einzelner nächster Wörter zu Chat-GPT-Funktionalität?
4. …

→ Wir sehen uns nun der Reihe nach diese Fragen an!

→ Im Zuge dessen: relevante Konzepte, Terminologie und Hintergründe

[1] Next token prediction visualization (GPT-2)

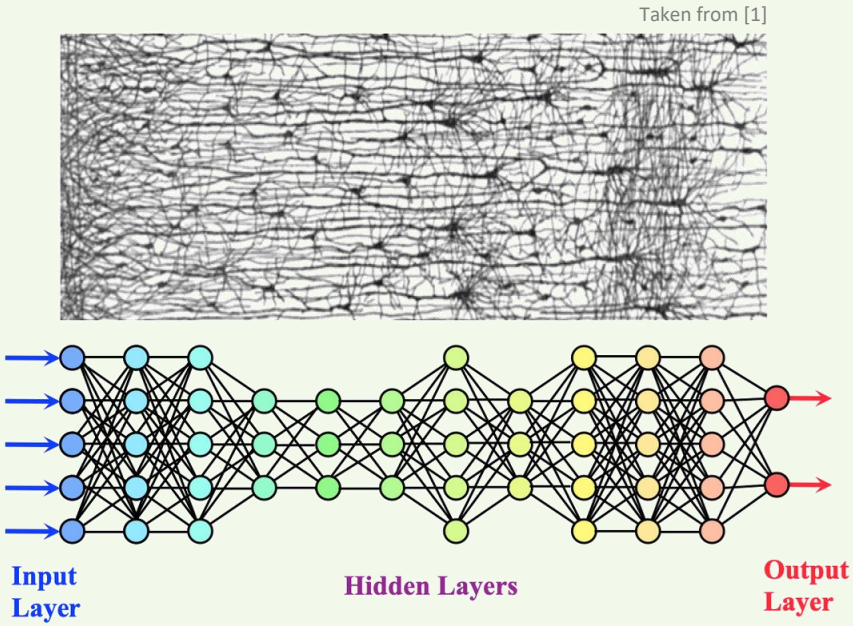
Large Language Models für KMU

1. Wie machen wir natürliche Sprache maschinenlesbar?

Kurz einen Schritt zurück:

- LMs sind **Neuronale Netze** (siehe Grafik rechts)
- Diese bestehen aus **Knoten** und **Kanten**
- Inputs werden durch **Zahlen** dargestellt und wandern durch das Netzwerk, bis schließlich ein Output erzeugt wird
- (**Im Detail**: Inputs bzw. Outputs jedes Hidden Layers werden an jeder Kante gewichtet und an jedem Knoten mit einem Offset versehen + durch eine nicht-lineare Funktion geschickt, etc.)

Taken from [1]



Frage: Wir wollen einen Satz verarbeiten.
Was ist hier unser erstes zentrale Problem?

→ Wörter kann man schwer mathematisch verarbeiten

→ Wir müssen alles **als Zahlen darstellen**, damit wir es überhaupt durch das Neuronale Netz schicken können!

[1] Deep Learning (DL) - Questions and Answers in MRI

Large Language Models für KMU

1. Wie machen wir natürliche Sprache maschinenlesbar?

"The quick brown fox jumps over the lazy dog."

Wir wissen nun, dass wir diesen Satz als Zahlen ‚codieren' müssen → **Aber wie sollen wir das nun angehen?**

Einige intuitive Möglichkeiten:

- Den gesamten Satz codieren
  "The quick brown fox jumps over the lazy dog." → 101
- Einzelne Wörter codieren
  "The" → 32, *"quick"* → 25, *"brown"* → 534, etc.
- Einzelne Zeichen
  „T" → 33, „h" → „35", etc.
- etc.

→ **Frage:** Was könnten hier jeweils dazugehörige Vor- und Nachteile sein?

[1] Stanford University - Speech and Language Processing, Chapter 2

Large Language Models für KMU

1. Wie machen wir natürliche Sprache maschinenlesbar?

- Es gibt zahllose Sätze, die man bilden könnte; wir werden nicht für jeden möglichen Satz eine Zahl zuweisen können…
- Was ist mit Leerzeichen und Satzzeichen?
- Was ist mit Groß- und Kleinschreibung (bzw. Wörtern am Satzanfang versus in der Satzmitte?)
- Was ist mit Sprachen, die gar keine klassischen Wortgrenzen kennen (Chinesisch, Japanisch)?
- Was ist mit Wörtern, die vorher nie gesehen wurden, d.h. noch keine Zuweisung einer Zahl haben?

Man sieht sehr schnell:

- Natürliche Sprache maschinenlesbar zu machen ist **nicht trivial!**
- Es gibt diverse Ansätze, die alle ihre Vor- und Nachteile haben

→ Das Problem, natürliche Sprache in einzelne (sinnvolle!) ‚Pakete' zu unterteilen, nennt man **Tokenization.**

[1] Building Blocks of LLMs: Tokenization and Embeddings

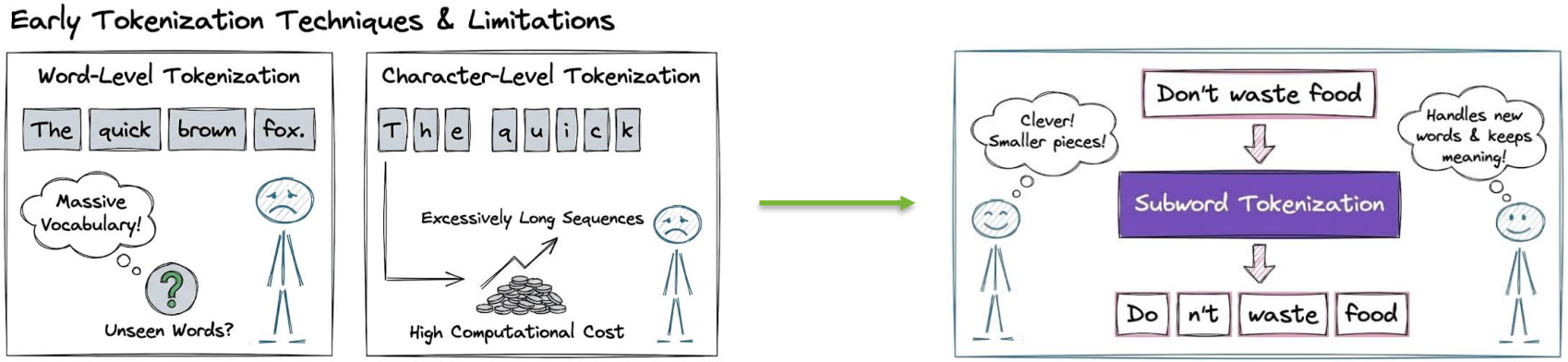
Large Language Models für KMU

1. Wie machen wir natürliche Sprache maschinenlesbar?

- Anfängliche Ansätze nutzten Wort- bzw. Buchstaben-Level Tokenization
- Große Probleme gab es hier allerdings bei unbekannten Wörtern → kein Mapping, keine Token ID!
- Ein weiteres Problem: je mehr Tokens, desto mehr Aufwand (insb. finanziell!) im LLM-Betrieb (Stichwort Token-Count)

→ Der heutige Standard liegt in **Subword Tokenization** (GPT-Versionen, Llama, …)

→ Ein Token kann hier ein **ganzes Wort** sein, oder ein **oft vorkommendes „Subword“**



Taken from [1]

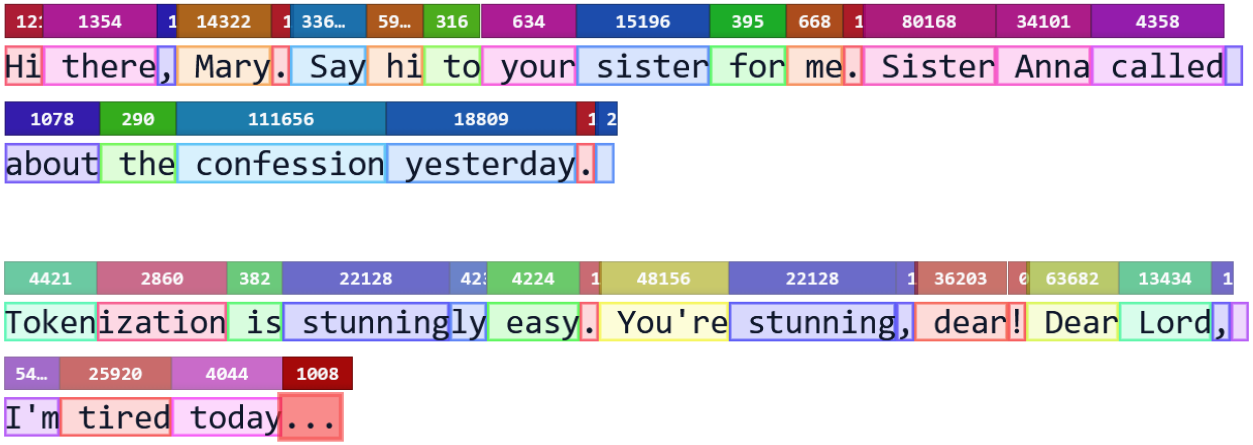
[1] Building Blocks of LLMs: Tokenization and Embeddings

Large Language Models für KMU

1. Wie machen wir natürliche Sprache maschinenlesbar?

Es gibt einige tolle Online-Demos zu Tokenization:

- Tokenizer Playground – Huggingface
- GPT Tokenizer Playground
- Tiktokenizer

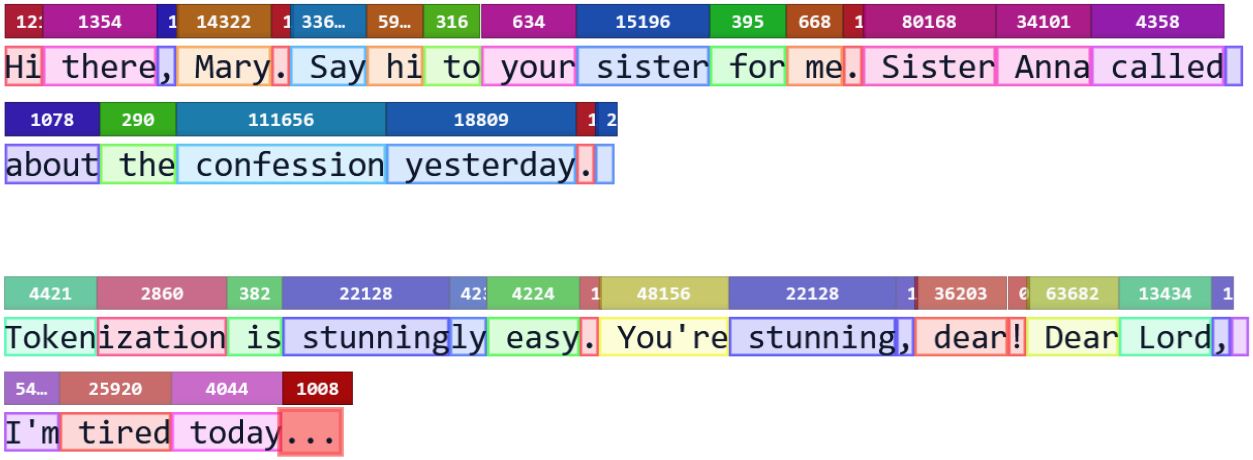


→ **Frage:** Was fällt Ihnen/euch in diesen Beispielsätzen auf (hier sind die Token IDs leider teilweise abgeschnitten…)?

[1] Stanford University - Speech and Language Processing, Chapter 2

Large Language Models für KMU

1. Wie machen wir natürliche Sprache maschinenlesbar?

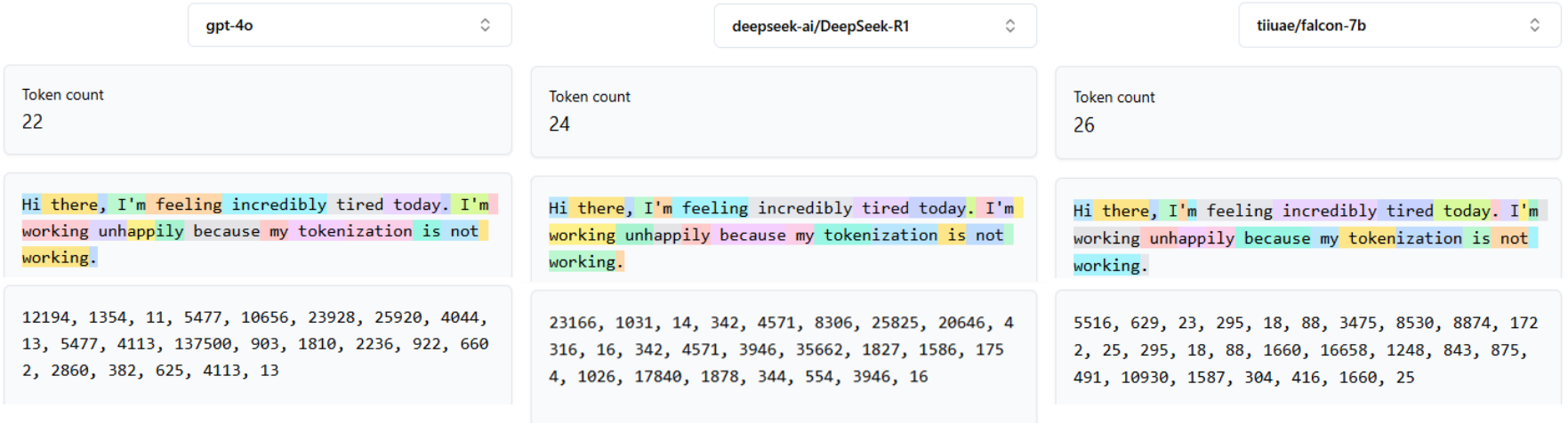


- **Tokenizer** unterteilen nicht nur Sätze in einzelne ‚Pakete' (**Tokens**) – sie weisen auch konkrete Zahlen (**Token IDs)** zu
- Gleiche Wörter bedeuten nicht gleiche Tokenization (z.B. „sister" und „Sister", oder „hi" und „Hi"). Es wird (zumeist):
  - zwischen Tokens am **Satzanfang** und **mitten im Satz** (+ vorhergehendes Leerzeichen) unterschieden
  - zwischen **groß-** und **kleingeschriebenen** Tokens unterschieden
  - auf **grammatische** Details Rücksicht genommen (z.B. „stunningly" wird nicht als Adverb repräsentiert, sondern als Adjektiv + „ly")

Large Language Models für KMU

1. Wie machen wir natürliche Sprache maschinenlesbar?

- Das **Mapping** (Token zu Token ID) wird anhand zahlloser Trainingstexte **trainiert**
  - Der Tokenizer ist ein **eigens trainiertes Modell**, das einen Input (Satz) in Tokens aufspaltet
  - Wortteile, die hier oft sequenziell auftreten, werden als ein Token ‚gelernt'
  - Erst, wenn wir den Satz als Zahlen dargestellt haben (Reihe von Token IDs), können wir ihn als **LLM-Input** übergeben
- **Wichtig**: nicht jeder Tokenizer gibt uns denselben Output für einen konkreten Inputsatz!
  - Die Art der Tokenization kann sich unterscheiden (hängt vom Training ab!)
  - Das Mapping zwischen Wort/Subword und Token ID kann (und wird!) sich unterscheiden



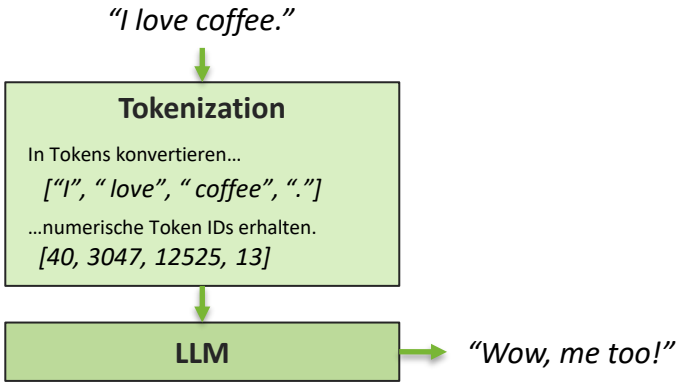
[1] How tokenizers work in AI models: A beginner-friendly guide
[2] Tiktokenizer

Large Language Models für KMU

1. Wie machen wir natürliche Sprache maschinenlesbar?

Das war nun eine Menge Information. **Wo stehen wir jetzt eigentlich?**

- Wir haben uns bisher nur den **Tokenization** Teil angesehen
- Viel Arbeit, um unseren Input überhaupt **maschinenlesbar** zu machen
- In bekannten Modellen (z.B. Chat-GPT) passiert die Tokenization intern – es handelt sich aber immer noch um **Preprocessing** und ist **nicht** Teil des LLMs!




```python
from transformers import GPT2Tokenizer

# Load pretrained GPT-2 tokenizer
tokenizer = GPT2Tokenizer.from_pretrained("gpt2")

# Step 1: Take input string
text = "Nebius is the best"

# Step 2 and 3: Encode text into token IDs
token_ids = tokenizer.encode(text)

# Optional: decode back into tokens for clarity
tokens = [tokenizer.decode([tid]) for tid in token_ids]

print("Text:", text)
print("Tokens:", tokens)
print("Token IDs:", token_ids)
```

- Es gibt diverse ‚**gebrauchsfertige**' **Tokenizer** (z.B. in der ‚transformers' library)
- Je nach Anwendung kann sich allerdings ein **eigens trainierter Tokenizer** bewähren. Man würde hierfür:
  - einen großen Datensatz an relevantem Text sammeln
  - eine Tokenization-Strategie wählen (Wort, Subword, etc.)
  - vorhandene Tools zum Tokenizer-Training nutzen z.B. Building a tokenizer, block by block · Hugging Face
  - den fertigen Tokenizer (z.B. als json) speichern und später wieder importieren

[1] How tokenizers work in AI models: A beginner-friendly guide

Large Language Models für KMU

2. Wie kommen wir von Token IDs zu bedeutungsvollen Darstellungen?

Wir haben jetzt eine maschinenlesbare Darstellung unseres Textes erarbeitet:

“I love coffee.” → *[40, 3047, 12525, 13]*

Nun laufen wir allerdings direkt in das nächste Problem:

- Wir können die *Ähnlichkeit* von einzelnen Wörtern nicht wirklich abbilden…

Nehmen wir bspw. die Wörter: **Katze, Hund, Auto, Flugzeug, Gitarre.**

- *Katze* und *Hund* sind sich semantisch näher als bspw. *Katze* und *Auto*
- *Auto* und *Flugzeug* sind sich semantisch näher als bspw. *Auto* und *Gitarre*
- *Gitarre* ist hier eigentlich keinem anderen Wort besonders nahe…

→ Wir können diese ‚Ähnlichkeiten‘ intuitiv abschätzen (unsere Sprachverarbeitung ist so fortgeschritten)

→ Ein Tokenizer (z.B. gpt-4o) gibt uns einstweilen aber nur einzelne Werte aus:
- cat: 8837
- dog: 30146

→ Die Zahlen sind **nicht einmal annähernd** vergleichbar; wie kommen wir nun also zu **semantischem Verständnis?**

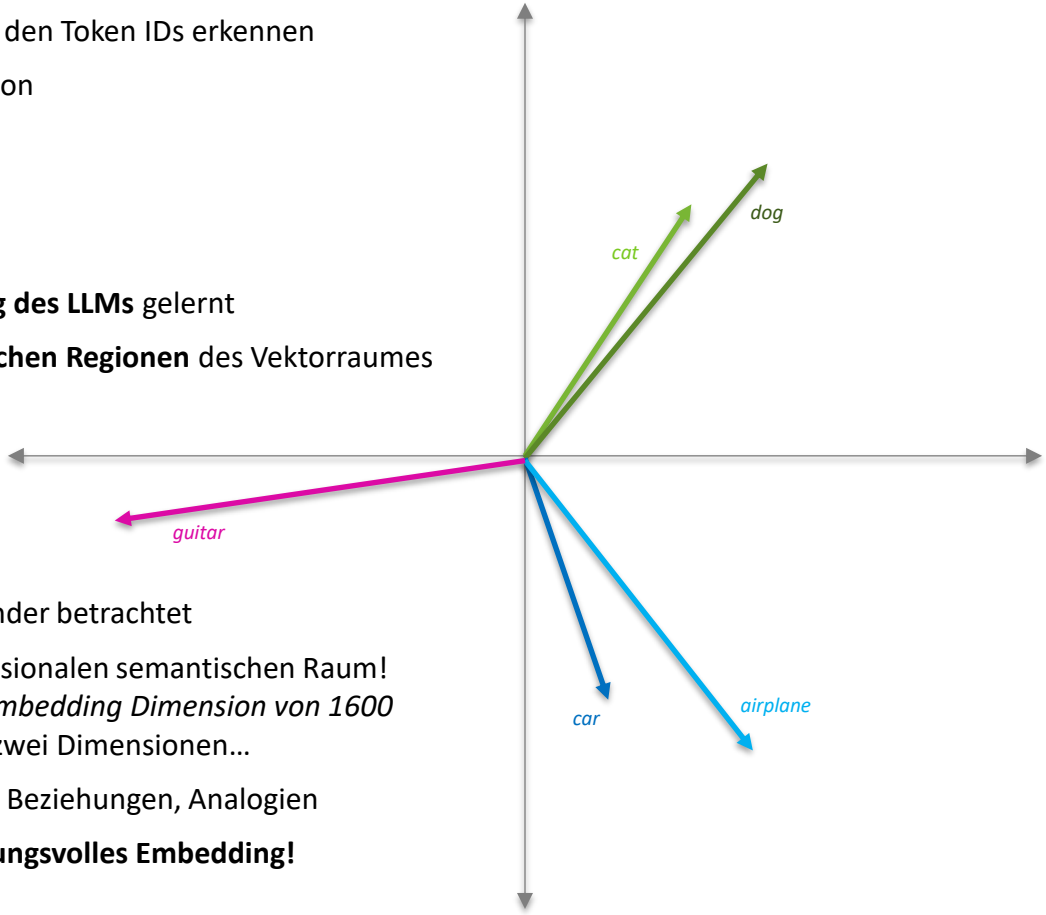
Large Language Models für KMU

2. Wie kommen wir von Token IDs zu bedeutungsvollen Darstellungen?

- Ob zwei Wörter ähnlich sind, lässt sich nicht and den Token IDs erkennen
- Wir benötigen hier eine sinnvollere Repräsentation

Die Lösung: Embeddings

- Jedes Token wird in einen **Vektor** umgewandelt
- Diese Vektordarstellungen werden beim **Training des LLMs** gelernt
- Wörter mit ähnlicher Bedeutung landen in **ähnlichen Regionen** des Vektorraumes



→ Wörter werden nicht mehr unabhängig voneinander betrachtet

→ Sie sind stattdessen Punkte in einem hochdimensionalen semantischen Raum!
GPT-2 hatte in der größten Modellversion eine Embedding Dimension von 1600
→ die Grafik hier demonstriert Embeddings for zwei Dimensionen…

→ Es entstehen plötzlich Bedeutungsähnlichkeiten, Beziehungen, Analogien

→ **Die erste ‚Stage' eines LLMs ist also ein bedeutungsvolles Embedding!**

[1] GPT-2 - Wikipedia

Large Language Models für KMU

3. Language Models – wie sehen die nun aus?

Wir haben jetzt einen maschinenlesbaren Input für unser Modell:

"I love coffee." → ***[40, 3047, 12525, 13]***

Jetzt sind wir zurück beim eigentlichen Problem:

Wie soll ein Netzwerk aussehen, das in natürlicher Sprache mit uns kommunizieren kann?

Ein paar grundlegende Beobachtungen…

- Wir wissen bereits, dass Embedding ein Thema ist – wir behalten das vorerst im Hinterkopf
- Kommunikation bedeutet im weitesten Sinne, auf Inputs (Sätze) intelligent antworten zu können
- Die Problemstellung, ein Modell zu bauen, das **ganze Sätze** als Antwort auf ganze Sätze liefert, ist allerdings **enorm**
- Daher auf die kleinstmögliche Problemstellung heruntergebrochen: das Modell soll **einzelne Wörter** prädizieren!

→ „LMs/LLMs sind Modelle, die das wahrscheinlichste nächste Wort prädizieren."

Large Language Models für KMU

3. Language Models – wie sehen die nun aus?

Ein historischer Abriss (Teil I)

- **Statistical Language Models (SLMs)**

  Erste LMs, die das wahrscheinlichste nächste Wort bspw. über Frequenzzählungen ermittelten:

  $$P(fish \mid So\ long\ and\ thanks\ for\ all\ the) = \frac{\#\ (So\ long\ and\ thanks\ for\ all\ the\ fish)}{\#\ (So\ long\ and\ thanks\ for\ all\ the)}$$

  - Wie oft wird in einem verfügbaren (riesigen!) Korpus an Text *genau dieser Satz* erwähnt („So long and thanks for all the fish“)
  - Wie oft wird im selben Korpus der *Satzanfang* erwähnt („So long and thanks for all the“)

  Wird ein Satz aus dem Korpus, der so anfängt, oft mit diesem konkreten Wort beendet (hohe Counts im Zähler), hat dieses Wort eine hohe Wahrscheinlichkeit, auf den Satzanfang zu folgen!

- **N-gram Models:**

  Weiterentwicklung, die nur n-1 Wörter ‚in die Vergangenheit‘ blickt:

  $$P(fish \mid thanks\ for\ all\ the) = \frac{\#\ (thanks\ for\ all\ the\ fish)}{\#\ (thanks\ for\ all\ the)}$$

  - Etwas robuster, da hier die Wahrscheinlichkeit höher ist, dass die Phrase im Korpus bereits einmal vorkam

→ Soweit alles **ohne Neuronale Netze** (~1950-1990)**!**

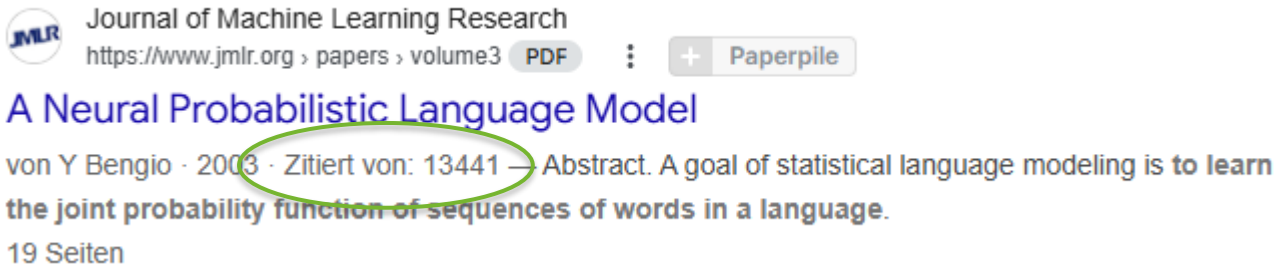
Large Language Models für KMU

3. Language Models – wie sehen die nun aus?

Ein historischer Abriss (Teil I)

- **Neural Probabilistic Language Models**

  Erstes auf neuronalen Netzen basierendes Language Model:



  - Diese Publikation wird vielseits als „Urvater moderner LLMs“ aufgefasst (wohl zurecht – Zitierungen!!!)
  - Verwendete erstmals ein neuronales Netz um das wahrscheinlichste nächste Wort zu prädizieren
  - **Wichtig:** noch kein RNN, CNN, LSTM, Transformer, …
  - **Das Problem:** verwendete ebenfalls ein fixes Fenster an vorhergehenden Wörtern zur Prädiktion…

[1] Bengio et al. (2003) - A Neural Probabilistic Language Model

Large Language Models für KMU

3. Language Models – wie sehen die nun aus?

- **Frage:** Warum haben diese Ansätze nicht ausreichend gut performt? Wo gibt es Probleme?

„An adorable little boy was spreading smiles.“

- Unter Umständen gilt: $P(smiles \mid \textit{boy was spreading}) < P(insults \mid \textit{boy was spreading})$
- Der anfängliche Teil „An adorable little“ wird hier ignoriert!

„The professor who supervised the student
who won the award after several years of difficult research in multiple countries
finally said that he ...”

- Auf wen bezieht sich „he“?
- Das Modell müsste sehr weit zurückblicken können, um den Zusammenhang mit „The professor“ zu sehen
- Ob der semantische Zusammenhang überhaupt wahrgenommen werden kann, hängt von der Fensterlänge (wie viele vorhergehende Wörter werden betrachtet?) ab

→ **Man war gezwungen, sich andere Lösungen einfallen zu lassen, um den semantischen Zusammenhang miteinzubeziehen!**

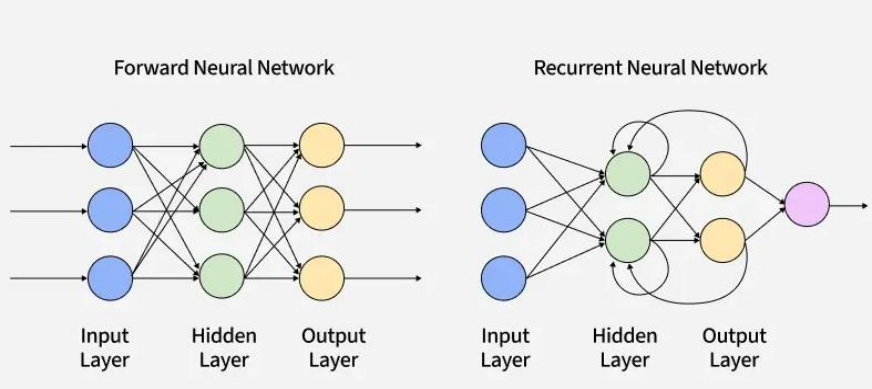
Large Language Models für KMU

3. Language Models – wie sehen die nun aus?

Ein historischer Abriss (Teil II)

- **Recurrent Neural Networks (RNNs)**

  Man wollte bessere Modelle, die einen beliebig langen Kontext (Fenster) haben könnten
  - Usprüngliche Idee: M. Jordan (1980er) und J. Elman (1990)
  - Man erlaubt Feedback-Zyklen in der Architektur, die eine „Erinnerung“ an vorherige Outputs ermöglichen
  - Vergangene Information wird so **implizit** „weitergetragen“



Taken from [1]

→ **Erwartung**: nun kann Wort 1 immer noch Wort 500 beeinflussen – NLP ist gelöst
→ **Realität**: Vanishing Gradient Problem (Backpropagation through time or BPTT) –
je weiter (temporal) die Information entfernt ist, desto eher wird sie wieder ‚vergessen‘

[1] Difference Between Feed-Forward Neural Networks and Recurrent Neural Networks - GeeksforGeeks

Large Language Models für KMU

3. Language Models – wie sehen die nun aus?

Ein historischer Abriss (Teil II)

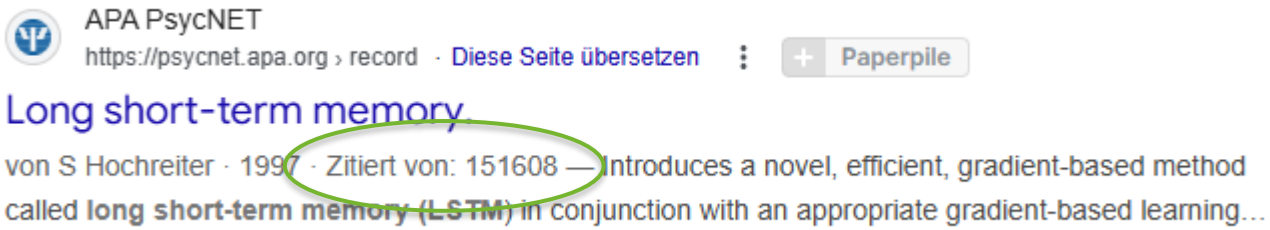
- **Long Short-Term Memory (dominant von Ende 1990er bis 2017!)**

  Wurden entwickelt, um das „Vergessen“ von klassischen RNNs zu lösen
  (federführend von Sepp Hochreiter – Österreich kann Deep Learning 😉)

  - Während RNNs sich in einem kontinuierlich ändernden Memory State an **alles** erinnern (allerdings schlecht)…
  - … schlagen LSTMs vor, das Memory aktiv zu ‚managen‘!

  Die Idee:

  → Das Netzwerk soll selbst entscheiden, was „merkwürdig“ ist und was nicht!
  → Wie Menschen soll auch das Netzwerk eine Art **selektive Wahrnehmung** anwenden können
  → Bspw. hier in diesem Workshop:
  - → „Soll ich mir das aufschreiben?“ → Input Gate
  - → „Soll ich das wieder durchstreichen?“ → Forget Gate
  - → „Soll ich diese aufgeschriebene Information nutzen, um jetzt etwas zum Thema zu sagen?“ → Output Gate
  - → Der Notizblock → Cell State



[1] Long Short-Term Memory | MIT Press Journals & Magazine | IEEE Xplore

Large Language Models für KMU

3. Language Models – wie sehen die nun aus?

Es gibt allerdings immer noch Probleme…

Unser Beispiel:

*„The professor who supervised the student
who won the award after several years of difficult research in multiple countries
finally said that he ...”*

- Wenn unser LSTM gut funktioniert, wird es sich bei ‘he’ and ‘The professor’ erinnern
- **Das Problem allerdings**: die Information muss **sequenziell** durch die dazwischenliegenden Wörter passieren („Scenic Route“…)

  professor *-> who -> supervised -> … -> said -> that ->* ***he***

→ Die Sequenz könnte potenziell hunderte Schritte umfassen – **aber ist das wirklich nötig?**

→ Menschen würden hier nicht Wort für Wort vorgehen, um den Zusammenhang zu sehen – es muss also auch **anders gehen**

Large Language Models für KMU

3. Language Models – wie sehen die nun aus?

Ein historischer Abriss (Teil III)

- **Attention (2014, 2017)**

  Die zentrale Frage: muss Information wirklich durch jedes dazwischenliegende Wort transportiert werden?
  - Wir wollen eigentlich nur den **direkten Zusammenhang** zwischen einzelnen Wörtern erkennen
  - Das Modell soll von selbst auf für ein konkretes Wort relevante andere Wörter blicken
  - Es soll sich nicht alles (auch Unnötiges!) merken, sondern gezielt auf Relevantes achten!

→ Die Idee an sich überzeugt schnell, aber wie wird das nun umgesetzt?

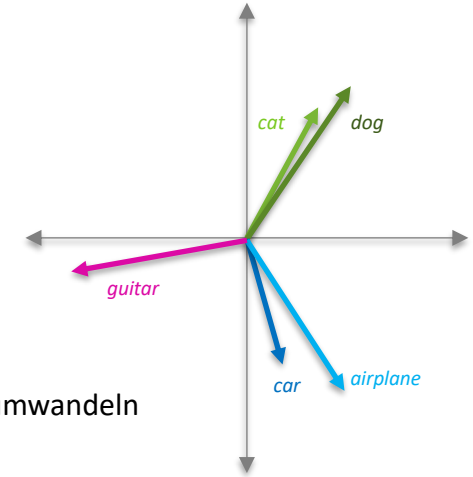
Large Language Models für KMU

3. Language Models – wie sehen die nun aus?

Attention Mechanismen

„The woman did not go to work because she was sick.“

- Das Modell sollte erkennen, dass „she“ sich auf „The woman“ bezieht
- Wie kann das nun funktionieren?



Ein kurzer Rückblick zu Embeddings:

- Wir haben gesagt: einem ersten Schritt muss ein LLM Token IDs in hochdimensionale Vektoren umwandeln
- Wenn das gut funktioniert, gibt es Metriken, die uns sagen, wie ähnlich sich zwei Vektoren sind:
  z.B. Cosine Similarity: +1 für exakte übereinstimmung, 0 für fehlenden Zusammenhang, -1 für völlig konträre Konzepte

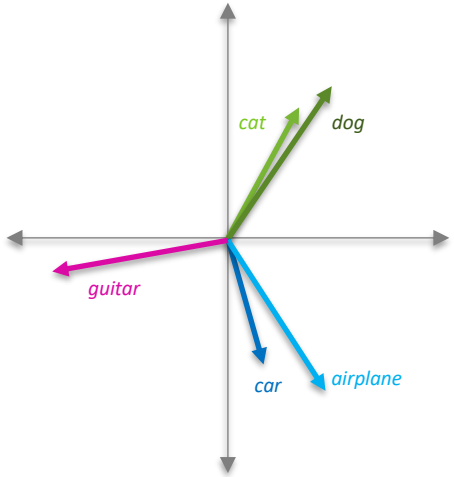
Das Problem: solche Metriken auf den reinen Embeddings bringen wenig

- Embeddings repräsentieren generelle semantische Bedeutung
- Um Zusammenhänge wie ‚she‘ und ‚The woman‘ zu sehen, braucht es aber Fokus auf Kontext-relevante Information
- *Ein Beispiel: „The professor supervised the student because she was very talented.“*
  → Welches vorhergehende Token soll nun ‚am ähnlichsten‘ zu ‚she‘ sein? ‚professor‘? ‚student‘?
  → Wir brauchen hier Verständnis vom Kontext, nicht nur von der Ähnlichkeit einzelner Wörter

[1] Self-Attention Concept: From Cross-Attention to Contextual Representations - Interactive | Michael Brenndoerfer | Michael Brenndoerfer
[2] Query, Key, Value: The Foundation of Transformer Attention - Interactive | Michael Brenndoerfer | Michael Brenndoerfer

Large Language Models für KMU

3. Language Models – wie sehen die nun aus?



a) Self-Attention

Bei Self-Attention betrachtet jedes Token **alle anderen Tokens** im aktuellen Kontext

- Für n Tokens im aktuellen Kontext wären das n^2 Paare, die wir betrachten
- Warum ist das dennoch effizienter als die RNN-/LSTM-Ansätze?

→ Weil Self-Attention in LLMs über Matrizen gelöst wird:

$$\vec{q} = \vec{x}Q, \qquad \vec{k} = \vec{x}K, \qquad \vec{v} = \vec{x}V,$$

Für jedes Embedding eines einzelnen Tokens (hier x) erzeugen wir über Projektionen:

1. einen Query-Vektor: „Nach welcher Information sucht Token x?“
2. einen Key-Vektor: „Welche Information *enthält* Token x?“
3. einen Value-Vektor: „Welche Information kann Token x zum aktuellen Kontext beitragen?“

→ **Wir fragen dann paarweise**: Passt die Information in Token x_i zur gesuchten Information von Token x_j? ($score = QK^T$)

→ Über diesen Matrix-Ansatz sind wir um längen schneller, als RNNs und LSTMs zuvor 💪 💪 💪

[1] Self-Attention Concept: From Cross-Attention to Contextual Representations - Interactive | Michael Brenndoerfer | Michael Brenndoerfer
[2] Query, Key, Value: The Foundation of Transformer Attention - Interactive | Michael Brenndoerfer | Michael Brenndoerfer

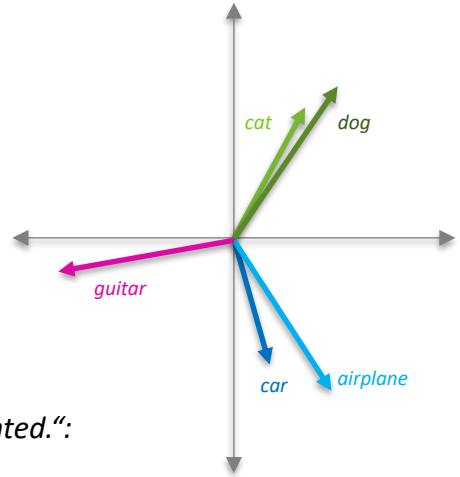
Large Language Models für KMU

3. Language Models – wie sehen die nun aus?

a) Self-Attention

Vereinfacht gesagt:

- Das Modell lernt, wie einzelne Tokens dargestellt werden müssen, um ihre **Frage** (Query), ihren **Inhalt** (Key), und ihren tatsächlichen **Beitrag** (Value) zu isolieren
- Wir erhalten am Ende dieser wunderschönen Mathematik Gewichte, die uns sagen, wie wichtig der Beitrag anderer Tokens für das aktuelle Token ist:



Zum Beispiel für das Wort „she“ in *„The professor supervised the student because she was very talented.“:*

Wort	Attention
professor	0.72
student	0.18
supervised	0.08
because	0.02

→ Das Modell teilt uns selbstständig mit, dass ‚professor‘ in diesem Kontext die größte Aufmerksamkeit erhalten sollte!

[1] Self-Attention Concept: From Cross-Attention to Contextual Representations - Interactive | Michael Brenndoerfer | Michael Brenndoerfer
[2] Query, Key, Value: The Foundation of Transformer Attention - Interactive | Michael Brenndoerfer | Michael Brenndoerfer

Large Language Models für KMU

3. Language Models – wie sehen die nun aus?

a) Self-Attention

Wie gesehen funktioniert das recht gut, **aber**:

Sprache enthält **diverse Arten** von Beziehungen zwischen Wortpaaren.

Wort	Attention
professor	0.72
student	0.18
supervised	0.08
because	0.02

Es gibt bspw. Zusammenhänge aufgrund von:

- Grammatik
- Referenzen
- Faktenwissen
- Zeitlichen Beziehungen
- etc.

→ In der Self-Attention wird jeweils **genau eine** Matrix für Queries (Q), Keys (K) und Values (V) gelernt

→ Dieses einzige Attention-Modul müsste alle die Zusammenhänge von oben unter einen Hut bekommen → unrealistisch!

Eine bessere Idee: wir lassen das Modell **viele Self-Attention** Module parallel lernen!

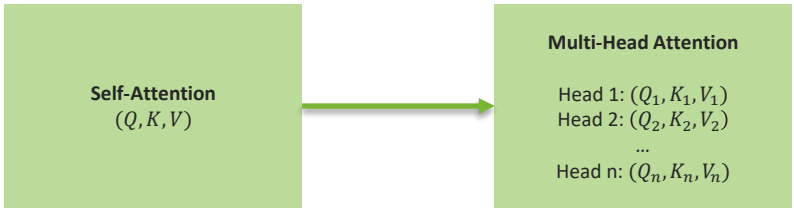
[1] Self-Attention Concept: From Cross-Attention to Contextual Representations - Interactive | Michael Brenndoerfer | Michael Brenndoerfer
[2] Query, Key, Value: The Foundation of Transformer Attention - Interactive | Michael Brenndoerfer | Michael Brenndoerfer

Large Language Models für KMU

3. Language Models – wie sehen die nun aus?

b) Multi-Head Attention

Anstatt nur jeweils eine Matrix Q, K und V zu lernen, lassen wir das Modell gleich **mehrere Self-Attention Module** erlernen!



- Jeder *Head* erlernt seine eigenen Projektionsmatrizen und damit seine eigene ‚Sicht' auf den Satz
- Was genau diese Projektionen ‚im realen Leben' bedeuten ist nicht immer klar
- Man kann es sich aber in etwa so vorstellen:
  - *ein Head fokussiert sich auf grammatikalische Zusammenhänge* (z.B. Subjekt – Verb)
  - *ein anderer auf Referenzen* (z.B. ‚she' – ‚the professor')
  - *etc.*

Die Vorteile:

- man kann mehrere Zusammenhänge gleichzeitig berücksichtigen
- die Umsetzung ist extrem schnell, da wir einfach Matrixmultiplikationen ausführen

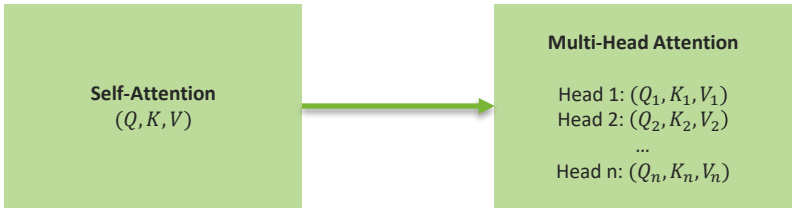
[1] Self-Attention Concept: From Cross-Attention to Contextual Representations - Interactive | Michael Brenndoerfer | Michael Brenndoerfer
[2] Query, Key, Value: The Foundation of Transformer Attention - Interactive | Michael Brenndoerfer | Michael Brenndoerfer

Large Language Models für KMU

3. Language Models – wie sehen die nun aus?

b) Multi-Head Attention

Anstatt nur jeweils eine Matrix Q, K und V zu lernen, lassen wir das Modell gleich **mehrere Self-Attention Module** erlernen!



Auch hier gibt es einige großartige Online Demos:

- https://www.nn-visual.com/attention?utm_source=chatgpt.com
- https://simulations4all.com/simulations/transformer-attention-visualizer?utm_source=chatgpt.com

[1] Self-Attention Concept: From Cross-Attention to Contextual Representations - Interactive | Michael Brenndoerfer | Michael Brenndoerfer
[2] Query, Key, Value: The Foundation of Transformer Attention - Interactive | Michael Brenndoerfer | Michael Brenndoerfer

Large Language Models für KMU

3. Language Models – wie sehen die nun aus?

Ein historischer Abriss (Teil IV)

- **Transformer-Architekturen (nach 2017, state-of-the-art)**

  Transformer haben das Multi-Head Attention Konzept erfolgreich in die Architektur integriert:
  - **Embeddings** erzeugen zunächst bedeutungsvolle Repräsentationen der Token IDs
  - **Multi-Head Attention** erlaubt dem Modell, kontextuelle Zusammenhänge zwischen Tokens zu erkennen
  - Zusätzliche **Feed-Forward Netzwerke** verarbeiten diese Information weiter
  - Diese Architektur wird **wiederholt hintereinander gestapelt** („Transformer Layers“)

Für jedes Token wiederholt das Modell immer wieder:

➢ „Welche anderen Tokens sind relevant?“

➢ „Welche Informationen sollte ich übernehmen?“

➢ „Wie verändert sich dadurch meine aktuelle Repräsentation?“

→ Jedes Token erhält eine kontextabhängige Bedeutung

→ Das Wort „bank“ kann dann plötzlich ein Geldinstitut oder ein Flussufer meinen

→ Die Bedeutung entsteht erst durch den Kontext

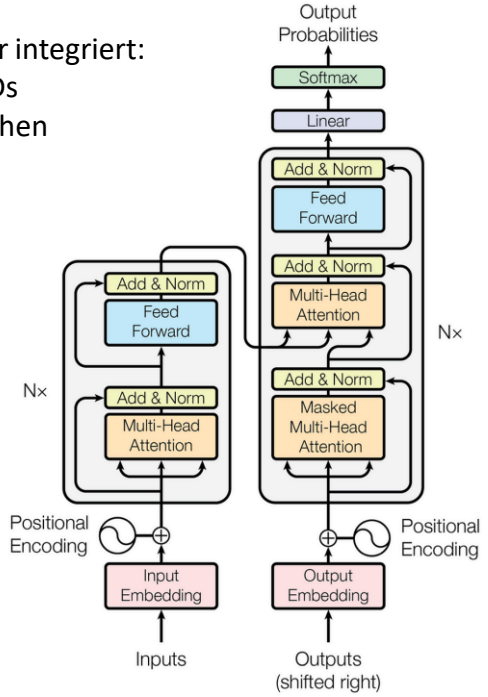


Figure 1: The Transformer - model architecture.

Taken from [1]

[1] Transformer Architecture Part -1. In recent years, transformers have… | by Sachin Soni | Towards AI

FH JOANNEUM
University of Applied Sciences

Large Language Models für KMU

3. Language Models – wie sehen die nun aus?

Transformer-Architekturen wurden historisch in unterschiedlichen Varianten entwickelt.
Alle diese Varianten verwenden Bausteine, die wir nun (teilweise) diskutiert haben:

- Tokenization
- Embeddings
- Self-Attention / Multi-Head Attention
- Feed Forward Layers
- etc.

→ Transformer stapeln zahllose Layers aneinander

→ Dadurch entsteht die „Power“: jedes Layer erkennt konkrete Muster

→ GPT-4, Claude oder Gemini verwenden dutzende bis hunderte solcher Layer

Generell unterscheidet man:

- Encoder-Decoder
- Encoder
- Decoder

[1] Transformer Architecture Part -1. In recent years, transformers have… | by Sachin Soni | Towards AI

Large Language Models für KMU

3. Language Models – wie sehen die nun aus?

Encoder-Decoder Architekturen

Die im ursprünglichen „Attention Is All You Need“-Paper vorgeschlagene Transformer-Architektur.

Der **Encoder:**

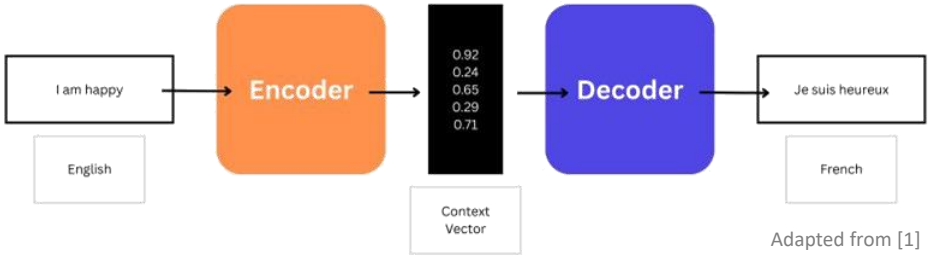
- verarbeitet den gesamten Input auf einmal und
- erstellt eine kontextuelle Repräsentation (Stichwort: Tokenization, Embedding, Attention!)

→ Unser Input wird auf eine bedeutungsvolle Weise ‚verschlüsselt‘ (=*encoded*): „**Hidden representation“**

Der **Decoder:**

- übernimmt diese Repräsentation und
- generiert anschließend einen Output (Prädiktion des wahrscheinlichsten nächsten Wortes!)

→ Das Encoding wird wieder ‚entschlüsselt‘ (=*decoded*) und die wahrscheinlichste Antwort generiert



Adapted from [1]

[1] Encoder-Decoder Architecture Explained: | AMIT SHEKHAR

Large Language Models für KMU

3. Language Models – wie sehen die nun aus?

Wichtig: je nach Anwendungsfall greift man auf **reine Encoder** oder **Decoder** zurück
Nicht immer ist die volle Encoder-Decoder Architektur notwendig oder zielführend!

Encoder-Only: Ziel ist es, den Input zu **verstehen**

- Wir betrachten / analysieren den gesamten Input (der **gesamte Text** ist auf einmal fertig verfügbar!)
- Wir wollen aber keinen darauf basierenden Text generieren
- **Beispiele:** BERT (Bidirectional Encoder Representations from Transformers), RoBERTa (Robustly Optimized BERT Pretraining Approach), etc.
- **Anwendungsfälle:** Textklassifizierung, Stimmungsanalyse (Sentiment Analysis), Named Entity Recognition (NER), etc.

Decoder-Only: Ziel ist es, **Text zu generieren**

- Wir sind nur daran interessiert, auf Basis des Inputs Outputs zu erstellen
- D.h.: ein Token wird erzeugt, wird neuer Teil des Kontexts, das nächste Token wird erzeugt, etc.
- Das Modell schreibt sich also **selbst fort** (Attention Layers sehen immer nur den aktuellen Kontext – was noch nicht generiert wurde, wird nicht betrachtet – Unterschied zu Decoder Layers!)
- **Beispiele:** Chat-GPT, Llama, Claude, Mistral, DeepSeek, …
- **Anwendungsfälle:** Chatbots/Assistenten, Zusammenfassungen, Code-Generierung, etc.

Large Language Models für KMU

3. Language Models – wie sehen die nun aus?

Noch eine interessante Online-Demo zum Abschluss: Nano-gpt: Architektur-Walk-Through

- Kaum jemand versteht, wie komplex die Architektur von LLMs tatsächlich sein kann
- Bereits nano-gpt mit nur ~86k Parametern ist kaum mehr zu durchblicken
- Als User denkt man sich oft „Wieso geht das nicht besser??“ → vielleicht hilft die Demo, zwischendurch geduldiger zu sein 😄

Large Language Models für KMU

4. Vom Next-Token-Predictor zum Chatbot

Was wir bisher gesehen haben:

- LLMs sind dafür trainiert, das wahrscheinlichste nächste Token vorherzusagen
- Decoder-Only Modelle erzeugen Text Token für Token
- Es gibt (noch) kein Verständnis im menschlichen Sinn

Dennoch können moderne LLMs Fragen beantworten, Texte schreiben, Übersetzen, Zusammenfassen, etc.

→ **Die zentrale Frage**: Wie ist das überhaupt möglich? Wie sind wir jetzt vom Next-Token-Predictor zu Chat-GPT gelangt?

Large Language Models für KMU

4. Vom Next-Token-Predictor zum Chatbot

Ein Beispiel. Unser Input:

„How should I bake my cake?“

Unser Modell weißt aber nicht:

- dass es ein Assistent sein soll
- dass es Fragen beantworten soll
- dass es höflich und hilfsbereit sein soll
- dass es Anweisungen zu befolgen hat
- …

Ein möglicher Output zu diesem Input: eine natürliche Fortsetzung unserer Frage!

„How should I bake my cake? Should I use flour? Should I use marmelade?“

→ Es wird einfach stur, iterativ das wahrscheinlichste nächste Wort prädiziert

→ Das Modell denkt es muss unsere Frage fortsetzen, obwohl wir gerne *eine Antwort* hätten!

Large Language Models für KMU

4. Vom Next-Token-Predictor zum Chatbot

Wie lernt ein Modell nun, dass es ein Chatbot sein soll?

→ Nach dem ursprünglichen Training erfolgt häufig ein **Instruction Tuning**

Instruction Tuning:

- Das Modell erhält Beispiele von Input und dazugehörigem Output

  „Generate a summary of …“ → „Here is die summary: …“

  „Translate this sentence to german: Hello, my name is …“ → „Here is a translation: Hallo, mein Name ist …“
- Über **Millionen** solcher Beispiele lernt das Modell: Anweisungen erkennen, Aufgaben verstehen, passende Antwortformate, etc.

Woher die Daten dafür stammen …

- Time Artikel über Arbeiter aus Kenia, die ermöglichen, Chat-GPT zu 'erziehen'
- Guardian Artikel zum gleichen Thema
- Arbeiter aus Indien, die für LLM-Training annotieren

→ Ganz zu schweigen von Diskussionen um geistiges Eigentum bei den Trainingsdaten der ursprünglichen LLMs…

Large Language Models für KMU

4. Vom Next-Token-Predictor zum Chatbot

Etwas zum Nachdenken…

Welche Hauptstadt liegt näher an Wien: Ulaanbaatar (Mongolei) oder Astana (Kasachstan)?

- Wahrscheinlich wird niemand die Antwort wissen
- Ihr könnt nur mutmaßen, basierend auf euren geographischen Vorkenntnissen

Wenn ich nun unbedingt eine Antwort will („Ich weiß nicht“ gilt nicht!):

- Ihr würdet eine Antwort geben (generieren),
- obwohl euch die eigentliche Information dazu fehlt (fehlende Trainingsdaten),
- und dabei möglicherweise sehr überzeugend klingen (Overconfidence)

→ Ihr habt im wesentlichen **halluziniert**

Aktueller State-of-the-art, um das zu reduzieren:

- Tool-Using: Modelle greifen auf das Internet zurück (Websuche), rufen relevante Informationen ab, und formulieren eine Antwort
- Antworten zu Wetter, Geografie, etc. werden möglich

MEHR ALS NUR CHATGPT: LARGE LANGUAGE MODELS FÜR KMU

2. CHANCEN UND RISIKEN

- *Bias, Halluzinationen, etc.*
- *Datenschutz & regulatorische Aspekte*
- *Future Perspectives / God-like AI?*
- *Überkonfidenz & „Sykophantische AI“*

FH JOANNEUM
University of Applied Sciences

Large Language Models für KMU

5. Chancen und Risiken: Biases, Datenschutz, etc.

Mittlerweile sind LLMs enorm leistungsfähig (Stichwort OpenClaw [1]…).

Mögliche Einsatzgebiete, speziell für KMU:

- Kundenkommunikation (Chatbots, automatisierte E-Mailbeantwortung + Terminvereinbarung, etc.)
- Wissensmanagement und RAG (Retrieval-Augmented Generation)
- Recherche und Dokumentation
- Datenanalyse und Marketing
- Softwareentwicklung und Prozessautomatisierung
- ….

Was allerdings zu bedenken ist: Je leistungsfähiger das Modell, desto wichtiger seine korrekte Funktionsweise!

→ Compliance, Datenschutz, …

[1] OpenAI schnappt sich österreichischen OpenClaw-Erfinder Peter Steinberger - KI - derStandard.at › Web

Large Language Models für KMU

5. Chancen und Risiken: Biases, Datenschutz, etc.

Bias – voreingenommene LLMs

- LLMs werden auf riesigen Datenmengen trainiert
- Diese Daten werden (oftmals unter sehr fragwürdigen Bedingungen) aus dem Internet gescraped
- Es ist nicht möglich, die Korrektheit (sowohl fachlich als ‚politisch') solcher Datenmengen manuell sicherzustellen
  Niemand schaut da persönlich durch, ob alles passt!

Das Problem nun: Die Trainingsdaten aus dem Internet enthalten leider Gottes

- kulturelle Stereotype (wie bspw. eine Person aus einem konkreten Land auszusehen hat)
- Politische Perspektiven (bspw. pro oder contra Trump)
- Historische Ungleichheiten (Gender Bias)
- Gesellschaftliche Vorurteile (Gender Bias!)
- Sprachliche Verzerrungen (Gender Bias!!)

→ „Garbage in, garbage out":
Wird das Modell auf problematischen Daten trainiert, verhält sich das Modell auch problematisch!

Large Language Models für KMU

5. Chancen und Risiken: Biases, Datenschutz, etc.

Bekanntere Arten von Bias:

- Unterschiedliche Zuschreibungen von Berufen zu Geschlechtern („Ballett dancer“ -> Frau, „Pilot“ -> Mann)
- Unterschiedliche Häufigkeit in der Repräsentation (Bildgeneration von jungen, schönen, schlanken, perfekten Personen)
- Politische Tendenzen (Ausländerfeindlichkeit, Rassismus, …)
- Geopolitische Perspektive (meister Content aus Amerika, meister Content auf Englisch, …)

Aber auch weniger bekannte Biases:

- Provider Bias (wenige Anbieter werden bspw. für Softwarelösungen empfohlen – Azure, AWS, …)
- Authority Bias (Bevorzugung von vertrauenswürdig scheinenden Institutionen/Quellen – Harvard, Yale, …)
- Corporate Bias (vielbeachtete Firmen/Platformen werden bevorzugt)
- Documentation Bias (gut dokumentierte Tools werden bevorzugt – sind aber nicht immer die besten…)

→ Es gibt hier haufenweise Probleme, an die man denken sollte!

Large Language Models für KMU

5. Chancen und Risiken: Biases, Datenschutz, etc.

Datenschutzthemen

Sprache enthält leider oft (besonders im Internet bzw. in Firmenarchiven…):

- Personenbezogene und nicht ausreichend pseudo-/anonymisierte Daten
- Kundendaten und Geschäftsgeheimnisse
- Verträge und Interna
- Finanzielle Informationen
- etc.

Die Fragen nun:

- Was passiert mit diesen Daten?
- Werden bspw. bei Chat-GPT Eingaben gespeichert?
- Werden diese für weiteres Training verwendet und stehen evtl. inhaltlich nächsten Usern zur Verfügung?
- Wo und wie werden die Daten verarbeitet?
- Wie lange und wo/wie werden sie gespeichert?

Large Language Models für KMU

5. Chancen und Risiken: Biases, Datenschutz, etc.

Datenschutzthemen

Beispiel: ein Mitarbeiter / eine Mitarbeiterin kopiert einen Kundenvertrag in Chat-GPT

Mögliche Probleme:

- Verstöße gegen interne Richtlinien/NDAs
- Datenschutzverletzungen
- Offenlegungen vertraulicher Informationen
- …

In diesem Kontext zu bedenken:

- Cloud oder on-prem Ausführung
- Datenspeicherung innerhalb/außerhalb der EU
- Nutzung für Modelltraining oder eben nicht
- AI-Act (Schleichwerbung: KI-Verordnung (AI-Act) und NIS-2 / NISG 2026. Sind Sie gut vorbereitet? - DIH Süd)

Noch ein Beispiel: Antragschreiben mit Chat-GPT – europäischer Erfindergeist liegt nun auf amerikanischen Serverfarmen…

Large Language Models für KMU

5. Chancen und Risiken: Biases, Datenschutz, etc.

Noch ein paar letzte Anregungen…

- Durch ‚friktionslose' Interaktion mit LLMs verlernen wir scheinbar, wie wichtig Anstrengungen für den Lernprozess sind:
  Against frictionless AI
- Passive, unkritische Nutzung von AI ‚erodiert' unsere Kognition
  The brain side of human-AI interactions in the long-term
- Ein großer Teil verfügbaren Wissens (sogar in der Wissenschaft) ist mittlerweile AI-generiert…
  AI-generated text surges in research papers
- Sykophantische AI bestärkt uns in fragwürdigen moralischen und interpersonellen Ansichten
  In defense of social friction | Science
  (wahrscheinlich der interessanteste Artikel, aber leider nicht open-source verfügbar –
  bitte bei Interesse per E-mail bei mir melden, dann sende ich ihn weiter!)
- Unser Gehirn ist ebenfalls nur ein Neuronales Netz –
  wenn wir ein Bewusstsein entwickeln können, spricht eigentlich nichts gegen ein maschinelles Bewusstsein

Zusammenfassend…:

→ LLMs machen vieles besser

→ Sie machen aber auch einiges schlechter

→ Sorgfältiger Umgang ist hier wirklich dringend notwendig!