Wie Sprachmodelle Sprache verstehen

Ein Blick hinter die Mathematik von ChatGPT und Co.

Martin Vogt

Große Sprachmodelle (Large Language Models, LLMs) wie ChatGPT, Gemini oder Claude haben innerhalb kurzer Zeit den Weg in unsere Lebenswirklichkeit gefunden. So gaben beispielsweise in einer Studie aus dem Jahr 2025 über 90 % der Studierenden an, KI-Tools im Studium zu nutzen (vgl. von Garrel & Mayer, 2025). Laut Welt Online (vgl. Macho, 2026) werden Sprachmodelle zudem von vielen Schülerinnen und Schülern regelmäßig bei Hausaufgaben, Texten und in der Freizeit eingesetzt. Der gleiche Artikel weist darauf hin, dass die Entwicklungen rund um das Thema Künstliche Intelligenz Jugendlichen mit Blick auf ihre berufliche Zukunft zunehmend Sorgen bereiten (vgl. Macho, 2026). Es lohnt sich also, dieses Thema aktiv im Schulunterricht aufzugreifen und zu diskutieren.

Aus mathematischer Sicht bietet es sich an, drei zentrale Ideen hinter der Sprachfähigkeit großer Sprachmodelle zu erläutern: a) die Darstellung von Wörtern als Vektoren (Word Embeddings), b) neuronale Netze zum Lernen von Mustern durch gewichtete Verbindungen und c) den Attention-Mechanismus, der relevante Wörter nach Kontext gewichtet. Alle drei Ideen lassen sich aus Vektoren, Matrizen, Matrix-Vektor-Multiplikation und Skalarprodukt aufbauen. Diese Operationen sind im Lernbereich 3 „Algebraisierung von mehrdimensionalen Verflechtungen und analytische Beschreibung des Raumes“ des Lehrplans für das berufliche Gymnasium in Rheinland-Pfalz verankert (vgl. Pädagogisches Landesinstitut Rheinland-Pfalz, 2014, S. 22 ff.).

Bezüge zur Mathematik der künstlichen Intelligenz wurden in zwei vorausgegangenen Beiträgen bereits hergestellt – für neuronale Netze und Bilder bei Bast et al. (2022), für Wörter als Vektoren und den Umgang mit Texten bei Nonnenmann et al. (2023). Dieser Beitrag greift beide Kernideen auf, bettet sie in den Kontext großer Sprachmodelle ein und widmet sich anschließend dem Herzstück moderner LLMs, dem Attention-Mechanismus (Vaswani et al., 2017).

Den Rahmen bildet ein Lernarrangement mit drei Lernsituationen im Umfeld eines Praktikums in einem Übersetzungsbüro. Die Lernenden schlüpfen in die Rolle einer Praktikantin bzw. eines Praktikanten und evaluieren einen KI-gestützten Übersetzungsassistenten. Lernsituation 1 geht der Frage nach, wie Computer mit Wörtern rechnen können. Lernsituation 2 zeigt, wie sich Wortvektoren mithilfe neuronaler Netze verändern lassen. Lernsituation 3 untersucht, wie der Kontext bei einer Übersetzung berücksichtigt werden kann. Als motivierender Stolperstein dient der mehrdeutige Begriff „Bank“, der ins Englische sowohl als „bench“ (Sitzbank) als auch als „bank“ (Finanzinstitut) übersetzt werden kann.

Mit Wörtern rechnen – Word Embedding

Lernsituation 1: *Sie absolvieren ein Praktikum in einem Übersetzungsbüro und sollen einen KI-gestützten Übersetzungsassistenten testen. Gleich zu Beginn fallen Ihnen zwei Sätze auf: „Ich sitze auf der Bank.“ und „Ich zahle bei der Bank.“. Beide Sätze enthalten das identische Wort „Bank“, müssen aber ins Englische unterschiedlich übersetzt werden. Wie kann der Computer überhaupt mit Wörtern umgehen und rechnen?*

Eine erste Antwort auf die Frage, wie ein Computer Wörter verarbeiten kann, gibt das Konzept des Word Embeddings (siehe auch Nonnenmann et al., 2023, S. 37). Die Grundidee besteht darin, jedes Wort als Punkt in einem mehrdimensionalen Vektorraum und damit als Vektor zu kodieren. In diesem Beitrag werden die Dimensionen Person/Aktion, Sitz/Ort und Finanzen gewählt. Dimensionen und Werte wurden hier didaktisch festgelegt. In echten LLMs werden sie nicht manuell bestimmt, sondern aus großen Textmengen gelernt und sind dort in der Regel nicht direkt interpretierbar (vgl. Mikolov et al., 2013). Die Grundidee, ein Wort als Punkt im Raum darzustellen, bleibt jedoch dieselbe. Für die beiden Beispielsätze ergibt sich die Kodierung in Tabelle 1, wobei die Werte wiederum manuell festgelegt wurden. Das Wort „Ich“ wird am ehesten mit einer Person in Verbindung gebracht und erhält dort eine 1, sowie in den Dimensionen Sitz/Ort

und Finanzen jeweils eine 0. Das Wort „auf“ kann auf einen Ort hinweisen, wenn auch nicht immer, und erhält dort einen Wert kleiner als 1. Hier wurde 0,4 gewählt.

Tabelle 1. Word Embedding der Beispielsätze

Wort	Person/Aktion	Sitz/Ort	Finanzen
Ich	1	0	0
sitze	0	1	0
zahle	0	0	1
auf	0	0,4	0
bei	0	0	0,4
der	0	0	0
Bank	0	1	1

Auffällig ist die Kodierung des Wortes „Bank“ als Vektor $(0 \quad 1 \quad 1)$: Es ist sowohl in der Dimension Sitz/Ort als auch in der Dimension Finanzen vertreten. Im Word Embedding allein ist daher noch nicht entschieden, welche Bedeutung gemeint ist. Eine geometrische Darstellung der Word Embeddings im dreidimensionalen Raum zeigt Abbildung 1.

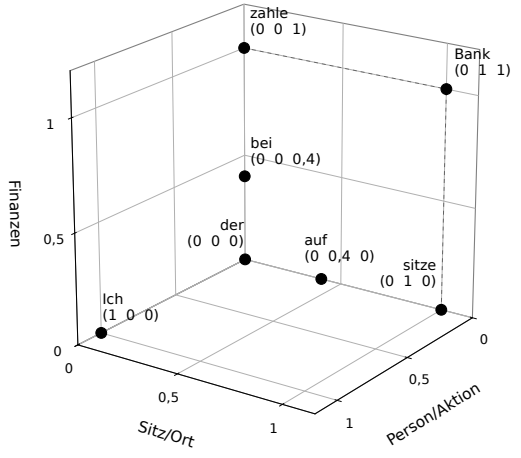

Abbildung 1. Dreidimensionale Darstellung der Word Embeddings beider Beispielsätze

Auch optisch fällt auf, dass das Wort „Bank“ zwischen den beiden Wörtern „sitze“ (Sitz/Ort) und „zahle“ (Finanzen) liegt. Die Ähnlichkeit zweier Wortvektoren lässt sich über das Skalarprodukt messen:

$$\text{sitze} \cdot \text{bank} = 0 \cdot 0 + 1 \cdot 1 + 0 \cdot 1 = 1$$
$$\text{zahle} \cdot \text{bank} = 0 \cdot 0 + 0 \cdot 1 + 1 \cdot 1 = 1$$

Beide Skalarprodukte ergeben denselben Wert. Aus Sicht der Word Embeddings ist das Wort „Bank“ also genauso nah am Wort „sitze“ wie am Wort „zahle“, während das Wort „ich“ weiter weg ist:

$$\text{ich} \cdot \text{bank} = 1 \cdot 0 + 0 \cdot 1 + 0 \cdot 1 = 0$$

Methodisch eignet sich hier das Grundprinzip des kooperativen Lernens (vgl. Brüning & Saum, 2009): In der Think-Phase erstellen die Lernenden eigene Word Embeddings für weitere Wörter (z. B. Park, Geld, Stuhl), in der Pair-Phase werden die Vektoren paarweise verglichen und die Skalarprodukte berechnet. In der Share-Phase werden die Ergebnisse im Plenum diskutiert.

Das Beispiel zeigt aber auch: Word Embeddings allein lösen die Mehrdeutigkeit des Wortes „Bank“ nicht auf. Es bedarf eines Mechanismus, der den Kontext eines Wortes berücksichtigt.

Neuronale Netze und Matrix-Vektor-Multiplikation

Lernsituation 2: *Sie haben bereits gelernt, dass Wörter als Vektoren dargestellt werden können, der Kontext dabei aber noch nicht beachtet wird. Je nach Kontext muss sich eine unterschiedliche Darstellung eines Wortes ergeben. Dabei kommt Ihnen die Frage in den Sinn: Wie können Vektoren in andere Vektoren* sinnvoll *umgewandelt werden?*

Wie kann beispielsweise das Wort „Bank“ im Kontext von „sitze“ stärker als Sitz/Ort und im Kontext von „zahle“ stärker in Richtung Finanzen dargestellt werden? Hierzu muss die Frage geklärt werden, wie sich Vektoren sinnvoll in andere Vektoren umwandeln lassen. Die Antwort lautet: über neuronale Netze.

Eine Einführung in die Grundlagen neuronaler Netze findet sich bei Nielsen (2015) und Bast et al. (2022, S. 9). Bei Letztgenannten werden anhand des Beispiels eines Überholvorgangs auf der Landstraße ein einfaches neuronales Netz und die damit verbundene Matrix-Vektor-Multiplikation eingeführt.

Abbildung 2 zeigt ein neuronales Netz mit drei Eingabeneuronen, über die der Vektor x etwa des Wortes „Bank“ in das Netz eingegeben werden kann. Die Eingabe wird mit einer Matrix W gewichtet, und es ergibt sich das Ergebnis q. Jede Eingangsdimension x_i ist mit jeder Ausgangsdimension q_j über ein Gewicht w_{ij} verbunden. Mathematisch ergibt sich mittels Matrix-Vektor-Multiplikation:

$$\begin{pmatrix} q_1 & q_2 & q_3 \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \cdot \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}$$

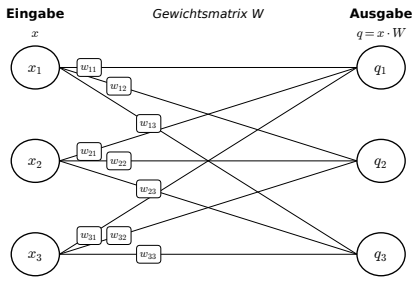

Abbildung 2. Struktur eines einfachen neuronalen Netzes

Ein solches neuronales Netz kann den Vektor des Wortes „Bank“ durch geschickte Wahl der Gewichte zum Beispiel stärker in Richtung Sitz/Ort verändern:

$$\begin{pmatrix}0 & 1,2 & 0,7\end{pmatrix} = \begin{pmatrix}0 & 1 & 1\end{pmatrix} \cdot \begin{bmatrix}0,2 & 0 & 0\\ 0 & 1 & 0,2\\ 0 & 0,2 & 0,5\end{bmatrix}$$

Methodisch eignet sich hier ein Lerntempoduett zum Erwerb der im Lernbereich 3 ausgewiesenen Kompetenzen (vgl. Pädagogisches Landesinstitut Rheinland-Pfalz, 2014, S. 22 ff.): Eine Person gibt ein Zielergebnis vor, die andere wählt die Gewichte und rechnet das Matrix-Vektor-Produkt schrittweise per Hand.

Aufmerksamkeit ist alles, was man braucht

Lernsituation 3: *Sie verstehen mittlerweile, wie Wörter als Vektoren dargestellt und durch ein neuronales Netz bzw. eine Gewichtsmatrix transformiert werden können. Doch wie kommt der Übersetzungsassistent nun zur richtigen Bedeutung des Wortes „Bank“?*

Das mathematische Herzstück moderner LLMs ist der Attention-Mechanismus, der von Vaswani et al. (2017) in dem Artikel „Attention is all you need“ (zu Deutsch: „Aufmerksamkeit ist alles, was man braucht“) beschrieben wurde. Er sorgt dafür, dass die Bedeutung eines Wortes durch seinen Kontext im Satz angepasst wird. Im Folgenden wird der Mechanismus an den beiden Beispielsätzen „Ich sitze auf der Bank.“ und „Ich zahle bei der Bank.“ Schritt für Schritt dargestellt.

Aus jedem Word Embedding x werden – wie in Lernsituation 2 vorbereitet – drei neue Vektoren über jeweils eigene Gewichtsmatrizen berechnet:

$$q = x \cdot W_q, \quad k = x \cdot W_k, \quad v = x \cdot W_v$$

Die drei Matrizen W_q, W_k und W_v werden in echten LLMs mit neuronalen Netzen (siehe Lernsituation 2) und großen Datenmengen trainiert (vgl. Touhid, 2024; zum Trainingsprozess mittels Backpropagation vgl. Goodfellow et al., 2016, S. 200 ff.). Aus didaktischen Gründen werden die Gewichtsmatrizen hier per Hand festgelegt. Im Folgenden werden der Attention-Mechanismus (siehe Abbildung 3) und die drei Vektoren q, k und v (in der Originalliteratur sind dies meist Matrizen, siehe Vaswani et al., 2017, S. 4; hier werden sie Wort für Wort, also als Vektoren betrachtet) näher erklärt.

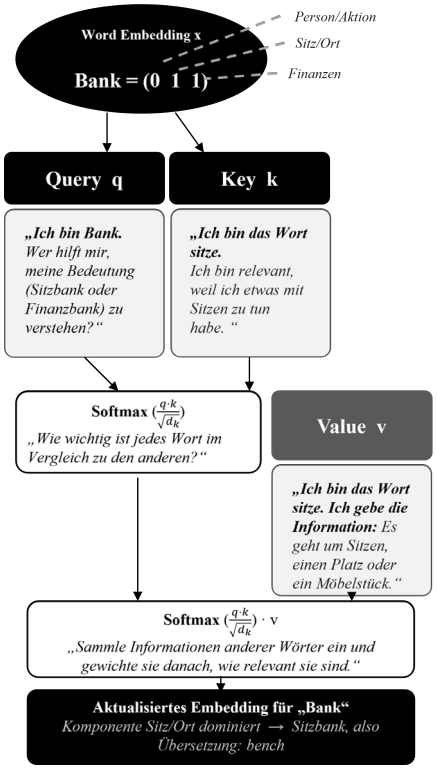

Abbildung 3. Struktur des Attention-Mechanismus (vgl. Touhid, 2024)

Der Vektor q ist die sogenannte Query (Anfrage) und entsteht durch Multiplikation des Word Embeddings x mit der Gewichtsmatrix W_q. Die Matrix W_q wurde in diesem Beispiel so gewählt, dass sie Sitz/Ort und Finanzinformationen betont ($w_{22} = w_{33} = 1$):

$$W_q = \begin{bmatrix}0,2 & 0 & 0\\ 0 & 1 & 0,2\\ 0 & 0,2 & 1\end{bmatrix}$$

Für das Wort „Bank“ ergibt sich die Query:

$$q_{\text{Bank}} = x_{\text{Bank}} \cdot W_q = \begin{pmatrix}0 & 1 & 1\end{pmatrix} \cdot \begin{bmatrix}0,2 & 0 & 0\\ 0 & 1 & 0,2\\ 0 & 0,2 & 1\end{bmatrix}$$
$$= \begin{pmatrix}0 & 1,2 & 1,2\end{pmatrix}$$

Dies bedeutet anschaulich: „Ich bin Bank. Ich suche Wörter, die mir helfen, besser zu verstehen, ob ich eine Sitz- (hohe zweite Komponente im Vektor) oder Finanzbank (hohe dritte Komponente im Vektor) bin."

Im nächsten Schritt wird die Query durch die anderen Wörter des Satzes über den Key beantwortet. Hierzu wird der Key k für alle Wörter mit der hier händisch festgelegten Matrix

$$W_k = \begin{bmatrix} 0,5 & 0 & 0 \\ 0 & 1,2 & 0,1 \\ 0 & 0,1 & 1,2 \end{bmatrix}$$

berechnet. Für das Wort „sitze" ergibt sich beispielsweise:

$$\begin{aligned} k_{\text{sitze}} &= \begin{pmatrix} 0 & 1 & 0 \end{pmatrix} \cdot \begin{bmatrix} 0,5 & 0 & 0 \\ 0 & 1,2 & 0,1 \\ 0 & 0,1 & 1,2 \end{bmatrix} \\ &= (0 \quad 1,2 \quad 0,1) \end{aligned}$$

Anschaulich besagt der Key k_{sitze} : „Ich bin das Wort „sitze". Ich bin relevant, weil ich etwas mit Sitz/Ort zu tun habe (hohe zweite Komponente)." Tabelle 2 enthält alle Keys für die Wörter der Sätze A und B.

Tabelle 2. Berechnete Keys aller Wörter in Satz A und Satz B

Satz A	Key	Satz B	Key
Ich	(0,5 0 0)	*Ich*	(0,5 0 0)
sitze	(0 1,2 0,1)	*zahle*	(0 0,1 1,2)
auf	(0 0,48 0,04)	*bei*	(0 0,04 0,48)
der	(0 0 0)	*der*	(0 0 0)
Bank	(0 1,3 1,3)	*Bank*	(0 1,3 1,3)

Nun werden die Query des Wortes „Bank" und der Key aller Wörter über das Skalarprodukt zusammengeführt, um die Relevanz des jeweiligen Wortes (z. B. „sitze") für die Query zu bestimmen. Beispielhaft ergeben sich in Satz A für die Wörter „sitze" und „auf":

$$\begin{aligned} q_{\text{Bank}} \cdot\ k_{\text{sitze}} &= \begin{pmatrix} 0 & 1,2 & 1,2 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1,2 \\ 0,1 \end{pmatrix} \\ &= 1,44 + 0,12 = 1,56 \end{aligned}$$

$$\begin{aligned} q_{\text{Bank}} \cdot\ k_{\text{auf}} &= \begin{pmatrix} 0 & 1,2 & 1,2 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0,48 \\ 0,04 \end{pmatrix} \\ &= 0,576 + 0,048 = 0,624 \end{aligned}$$

Beide Wörter antworten also auf die Query des Wortes „Bank". Da das Skalarprodukt für „sitze" (1,56) größer ist als das für „auf" (0,624), besitzt Ersteres eine höhere Relevanz für die Query des Wortes „Bank".

Im nächsten Schritt soll die Frage beantwortet werden, wie wichtig jedes Wort *im Vergleich* zu den anderen Wörtern ist. Hierzu werden die Ergebnisse des Skalarproduktes noch skaliert und in Wahrscheinlichkeiten umgerechnet. Zunächst wird das Ergebnis durch $\sqrt{d_k}$ geteilt, wobei d_k die Dimension des Key-Vektors angibt (hier $d_k = 3$). Diese Skalierung verhindert in echten Modellen mit großen Vektordimensionen, dass die nachfolgende Softmax-Funktion zu extreme Werte erzeugt (vgl. Vaswani et al., 2017, S. 4). Für den Unterrichtseinsatz spielt sie nur eine Nebenrolle und kann aus didaktischen Gründen auch weggelassen werden. Anschließend wird die Softmax-Funktion angewendet, die aus den Zahlen (z. B. 1,56) Wahrscheinlichkeiten macht, also Zahlen zwischen 0 und 1, deren Summe 1 ergibt. Der softmax-Wert eines Eingabewertes s_i ist definiert als:

$$\text{softmax}(s_i) = \frac{e^{s_i}}{\sum_j e^{s_j}},$$

wobei der Index j die Werte für alle Wörter i abläuft. Es wird somit der folgende Wert berechnet:

$$\text{softmax}\left(\frac{qk^T}{\sqrt{d_k}}\right).$$

Tabelle 3 gibt die berechneten softmax-Werte für das Wort „Bank" in den Sätzen A und B an. Interessant ist hier eine Beobachtung, die zunächst kontraintuitiv wirkt: Da die Wörter „sitze" und „zahle" in unserem Beispiel strukturell symmetrisch (nur mit vertauschten Rollen bezüglich Sitz/Ort und Finanzen) gewählt wurden, ergeben sich für beide Sätze identische softmax-Werte. In beiden Sätzen sind die Wörter „Ich" und „der" weniger wichtig, während „sitze" bzw. „zahle" stärker beachtet werden.

Tabelle 3. Berechnete softmax-Werte für das Wort „Bank" in den Sätzen A und B

Wort	$qk := q_{\text{Bank}} \cdot k_{\text{Wort}}$	$\frac{qk}{\sqrt{d_k}}$	$\text{softmax}\left(\frac{qk}{\sqrt{d_k}}\right)$
Ich	0,000	0,000	0,084
sitze zahle	1,560	0,901	0,206
auf bei	0,624	0,360	0,120
der	0,000	0,000	0,084
Bank	3,120	1,801	0,507

Wie also kann die KI dennoch unterscheiden, ob „Bank" eine Sitzbank oder ein Finanzinstitut bezeichnet? Die Antwort liegt in den Value-Vektoren v. Die softmax-Werte geben nur an, wie wichtig jedes Wort im Vergleich zu den anderen ist. Die Information, dass

es zum Beispiel um einen Sitz/Ort geht, wird erst durch den Value v eingebracht. Analog zu W_q und W_k wird hierzu die Gewichtsmatrix W_v über ein neuronales Netz trainiert (siehe Touhid, 2024). In diesem Artikel werden die Werte wiederum händisch nach didaktischen Kriterien festgelegt:

$$W_v = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

Für die Wörter „sitze“ $\begin{pmatrix} 0 & 1 & 0 \end{pmatrix}$ und „zahle“ $\begin{pmatrix} 0 & 0 & 1 \end{pmatrix}$ ergibt sich nach Multiplikation:

$$v_{\text{sitze}} = x_{\text{sitze}} \cdot W_v = (0 \quad 1 \quad 0) \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} = (0 \quad 2 \quad 0)$$

und analog

$$v_{\text{zahle}} = (0 \quad 0 \quad 2)$$

Das Wort „sitze“ trägt also die Information bei, dass es um einen Sitz/Ort geht, und das Wort „zahle“ die Information, dass es um Finanzen geht. Diese Werte werden im nächsten Schritt verwendet, um die sogenannte Attention zu berechnen (vgl. Vaswani et al., 2017):

$$\text{Attention}(q,\ k, v) = \text{softmax}\left(\frac{qk^T}{\sqrt{d_k}}\right) \cdot v$$

Hierbei werden die Informationen v aller Wörter gesammelt und entsprechend ihrer Relevanz (softmax) gewichtet. Für Satz *A* ergeben sich für das Wort „Bank“ die in der dritten Spalte von Tabelle 4 dargestellten Werte. Das Wort „Ich“ ist weniger relevant und liefert nur einen Beitrag zu Person/Aktion. Das Wort „Bank“ liefert hohe Beiträge zu Sitz/Ort und Finanzen. Interessant ist, dass es im Kontext zwei weitere Wörter gibt, die einen Beitrag zu Sitz/Ort leisten: „sitze“ und „auf“. Durch komponentenweise Summierung pro Dimension (z. B. für die zweite Dimension: $0+0,412+0,096+1,014=1,522$) ergibt sich der neue Vektor für „Bank“ in Satz *A*: $\begin{pmatrix} 0,084 & 1,522 & 1,014 \end{pmatrix}$. Für Satz *B* ergibt sich analog der Vektor $(0,084 \quad 1,014 \quad 1,522)$.

Obwohl das ursprüngliche Embedding des Wortes „Bank“ und sogar die softmax-Werte in beiden Sätzen identisch waren, unterscheidet sich die kontextabhängige Repräsentation deutlich.

Tabelle 4. Aktualisierung des Vektors für das Wort „Bank“ in Satz *A*

Wort	Value v	softmax$(\frac{q_{\text{Bank}} \cdot k_{\text{Wort}}}{\sqrt{d_k}}) \cdot v$
Ich	(1 0 0)	(0,084 0 0)
sitze	(0 2 0)	(0 0,412 0)
auf	(0 0,8 0)	(0 0,096 0)
der	(0 0 0)	(0 0 0)
Bank	(0 2 2)	(0 1,014 1,014)
Summe		**(0,084 1,522 1,014)**

In Satz *A* überwiegt die Komponente Sitz/Ort (zweite Komponente 1,522), die durch die Wörter „sitze“ und „auf“ erhöht wurde. In Satz *B* überwiegt die Komponente Finanzen (dritte Komponente 1,522), die durch „zahle“ und „bei“ verstärkt wird. Entsprechend wird in Satz *A* die englische Übersetzung „bench“ gewählt, in Satz *B* „bank“. Damit ist die Mehrdeutigkeit des Wortes „Bank“ mathematisch aufgelöst.

Methodisch eignet sich auch hier eine Erarbeitung nach Think-Pair-Share (vgl. Brüning & Saum, 2009): In der Think-Phase berechnen die Lernenden für einen weiteren Beispielsatz (z. B. „Ich gehe zur Bank.“) Query, Keys und Attention-Gewichte. In der Pair-Phase werden die Ergebnisse abgeglichen und in der Share-Phase im Plenum diskutiert.

Zusammenfassung und Ausblick

Stolz präsentieren Sie die Funktionsweise des KI-Übersetzungsassistenten in der wöchentlichen Praktikumsrunde. Aus den drei Lernsituationen ist klar geworden: Hinter KI-Übersetzungen stecken Vektoren, Matrix-Vektor-Multiplikationen und Skalarprodukte – also genau die Mathematik, die im Lehrplan verankert ist.

Die rasante Entwicklung von LLMs wirft Fragen auf, mit denen sich die Lernenden auch außerhalb des Mathematikunterrichts auseinandersetzen. In diesem Beitrag wurden drei zentrale mathematische Ideen moderner LLMs – Word Embeddings, neuronale Netze und der Attention-Mechanismus – in drei Lernsituationen aufbereitet, die sich alle dem Lernbereich 3 des Lehrplans zuordnen lassen (vgl. Pädagogisches Landesinstitut Rheinland-Pfalz, 2014, S. 22 ff.).

Anknüpfungspunkte für eine Vertiefung gibt es zahlreiche: Das Umfeld des Übersetzungsbüros lädt zu einer Diskussion über die Zukunft der Arbeit ein; fächerübergreifend bietet sich eine Verzahnung mit Deutsch (Mehrdeutigkeit natürlicher Sprache), Englisch (Vergleich maschineller und eigener Übersetzung), Ethik (Verantwortung und Bias) sowie Informatik (Implementierung in R oder Python) an.

Insgesamt zeigt sich: Die Mathematik hinter ChatGPT und vergleichbaren Modellen ist auf einem im Lehrplan verankerten Niveau darstellbar. So lässt sich Lernenden ein mathematisch fundierter Einblick in eine Technologie eröffnen, die zunehmend ihre Lebens- und Berufswirklichkeit prägt – und damit zugleich eine kritische Reflexion ermöglichen.

Literaturverzeichnis

Bast, S., Vogt, M., & Wallerath, R. (2022). Autonomes Fahren – Ein Blick hinter die Kulissen der Mathematik der künstlichen Intelligenz. *GDM Mitteilungen, 112*, 7–10.

Brüning, L., & Saum, T. (2009). *Erfolgreich unterrichten durch Kooperatives Lernen. Strategien zur Schüleraktivierung. Band 1* (5. Auflage). Neue Deutsche Schule Verlagsgesellschaft.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. www.deeplearningbook.org

Macho, A. (2026). Die neue Job-Angst der Jugend. In: *DIE WELT*. www.welt.de/wirtschaft/article6a046b01214d7d4d4aac5a4d/kuenstliche-intelligenz-die-neue-job-angst-der-jugend.html, zuletzt abgerufen am 15. 5. 2026

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems* (Vol. 26, S. 3111–3119).

Nielsen, M. A. (2015). Neural networks and deep learning. Determination Press. neuralnetworksanddeeplearning.com/

Nonnenmann, M., Vogt, M., Bast, S., & Lübke, K. (2023). Mathematik und Sprache – Textanalyse im Mathematikunterricht. *GDM-Mitteilungen, 115*, 33–38.

Pädagogisches Landesinstitut Rheinland-Pfalz (2014). Lehrplan für das berufliche Gymnasium. Unterrichtsfach: Mathematik, Grund- und Leistungsfach. bildung.rlp.de/fileadmin/user_upload/bbs/Informationen_und_Materialien/Lehrplaene/Berufliches_Gymnasium/2015-01-08_LP_BG_Mathe.pdf

Touhid (2024). The (surprisingly simple!) math behind the transformer attention mechanism. *Medium*. medium.com/@touhid3.1416/the-surprisingly-simple-math-behind-transformer-attention-mechanism-d354fbb4fef6 (Zugriff am 15. 5. 2026).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems* (Vol. 30, S. 5998–6008).

von Garrel, J., & Mayer, J. (2025). Künstliche Intelligenz im Studium – Eine quantitative Längsschnittstudie zur Nutzung KI-basierter Tools durch Studierende (2023 & 2025). Hochschule Darmstadt. opus4.kobv.de/opus4-h-da/533

Martin Vogt, Hochschule Trier
m.vogt@wir.hochschule-trier.de