Einführung in die Architektur von Transformer-Architekturen von Neuronalen Netzen

Ausarbeitung Integrationsseminar

im Rahmen der Prüfung zum

Bachelor of Science (B.Sc.)

des Studienganges Wirtschaftsinformatik Software Engineering

an der Dualen Hochschule Baden-Württemberg Mannheim

von

Alexander Meinecke

Abgabedatum:	27. Januar 2025
Bearbeitungszeitraum:	Dezember 2024 - 27. Januar 2025
Matrikelnummer, Kurs:	1522347, WWI22SEB
Ausbildungsfirma:	SAP SE Dietmar-Hopp-Allee 16 69190 Walldorf, Deutschland
Gutachter:	Alexander Lütke

Eidesstattliche Erklärung

Ich versichere hiermit, dass ich meine Ausarbeitung Integrationsseminar mit dem Thema:

Einführung in die Architektur von Transformer-Architekturen von Neuronalen Netzen

gemäß § 5 der „Studien- und Prüfungsordnung DHBW Technik“ vom 29. September 2017 selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Mannheim, den 15. Dezember 2024

Meinecke, Alexander

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

BEHG	Brennstoffemissionshandelsgesetz
CBAM	Carbon Border Adjustment Mechanism (dt. Grenzausgleichsmechanismus)
DEHSt	Deutsche Emissionshandelsstelle
EPA	United States Environmental Protection Agency (dt. Umweltschutzbehörde der Vereinigten Staaten)
ETS	Emission Trading System (dt. Emissionshandelssystem)
ICAP	International Carbon Action Partnership (dt. Internationale Partnerschaft für Emissionshandel)
MSR	Market Stability Reserve (dt. Marktstabilitätsreserve)
nEHS	Nationales Emissionshandelssystem
TNAC	Total Number of Allowances in Circulation (dt. Anzahl der Zertifikate im Umlauf)

1 Einleitung

2 Grundlagen

3 Transformer-Architektur im Detail

Im ursprünglichen Werk *Attention Is All You Need* , in dem das grundlegende Transformer-Modell vorgestellt wurde, besteht ein Transformer aus zwei Hauptkomponenten: dem Encoder und dem Decoder. cite

Encoder und Decoder verarbeiten Text in Form von Tokens, die ganze Wörter oder Wortteile sein können. Jeder Token ist für den Transformer einzigartig und wird zunächst nur durch eine natürliche Zahl repräsentiert.

Der **Encoder** übersetzt die Eingabewörte in eine abstrahierte Repräsentation. Ziel ist es hierbei, die semantischen und syntaktischen Informationen des Textes zu extrahieren und zu kodieren. Die andere Komponente, der **Decoder**, nutzt diese Repräsentation, um ein passendes Output zu generieren. Dabei berechnet er in jeder Iteration den nächsten, am besten passenden Token, der zum Output hinzugefügt werden soll. Hierbei bezieht er auch den zuletzt generierten Token in die Berechnung des nächsten Tokens ein.

Grundlage für sowohl den Encoder als auch den Decoder ist das Self-Attention-Konzept, das im folgenden Abschnitt näher erläutert wird.

3.1 Wie funktioniert Attention?

Ein Attention-Zyklus kann als Übersetzungsfunktion von Eingabevektoren zu Ausgabevektoren verstanden werden.

3.1.1 Generierung von Embeddings

Grundlage jedes Attention-Zyklus sind Eingabe-Tokens. Um die mathematische Vorgehensweise besser zu veranschaulichen, wird im Folgenden der Satz „Ich sitze auf der Bank“ beispielsweise verarbeitet. Jeder dieser Worter ist ein eigener Token, der vor der Eingabe in den Attention-Zyklus von dem Transformer übersetzt wurde. [„Ich“, „sitze“, . . . , „Bank“]. Diese Tokens können für das Modell wie folgt aussehen: [„243“, „645“, . . . , „316“]. Die Schwierigkeit für den

Transformer besteht hierbei, dass nur einen Token für „Bank“ gibt und es jetzt aus dem Kontext der anderen Tokens erkennen muss, ob es sich um eine Sitzbank oder und das Finanzinstitut Bank handelt.

Jeder Token ist ein Schlüssel für ein Schlüssel-Werte-Paar. Der korrespondierende Wert hinter einem Token ist ein Vektor, der die Bedeutung eines Tokens hinsichtlich mehrer Dimensionen erklärt. Diese Vektoren sind aus Trainingsdaten für das Transformermodel entstanden.

Im ersten Schritt im Attention-Zyklus wird jeder Token in den dazugehörigen Vektor übersetzt. Diese Vektoren werden in der Matrix $\mathbf{X}$ gespeichert, wobei jede Zeile ein Vektor ist und einen Token repräsentiert. Dieser Prozess wird **Embedding** genannt. Jeder dieser Vektoren hat laut Literatur mindestens 512 Dimensionen. Es wird von einem $d_{\text{model}} = 512$ gesprochen. Es gilt je größer das d_{model}, desto präziser kann ein Transformer die Zusammenhänge zwischen Tokens erkennen.

Wenn sich Token im Vektorraum nahe liegen, dann haben sie eher Gemeinsamkeiten als Token, die weit auseinander liegen. Angenommen es gäbe nur ein $d_{\text{model}} = 2$ für jeden Token, könnten diese zwei Dimensionen als Koordinaten genutzt werden, um Zusammenhänge visuell in einem Koordinatensystem als Cluster sichtbar zu machen. Hier wären z.B. die Token für „Hund“ und „Katze“ nah beieinander.

Für das oben genannte Bespiel „Ich sitze auf der Bank“, werden für die Übersichtlichkeit nun ein $d_{\text{model}} = 4$. So liegt letztendlich eine Embeddingmatrix $\mathbf{X}$ von 5 Zeilen für 5 Token und 4 Spalten für jeweils 4 Dimensionen vor.

$$X = \begin{bmatrix} 0.4 & 0.8 & 1.5 & 1.6 \\ 3.2 & 0.4 & 0.7 & 0.2 \\ 0.6 & 0.9 & 1.2 & 0.5 \\ 2.1 & 0.5 & 2.0 & 0.2 \\ 0.7 & 2.4 & 0.1 & 0.9 \end{bmatrix}$$

3.1.2 Lineare Transformation in Query-, Key- und Value-Matrizen

Die Embeddingmatrix $\mathbf{X}$ wird mithilfe dreier Gewichtungsmatrizen $\mathbf{W_Q}$, $\mathbf{W_K}$ und $\mathbf{W_V}$, die aus dem Training des Transformer-Modells stammen, in drei neue Matrizen mit Matrixmultiplikation transformiert:

$$\mathbf{Q} = \mathbf{X} \cdot \mathbf{W_Q}$$

$$\mathbf{K} = \mathbf{X} \cdot \mathbf{W_K}$$

$$\mathbf{V} = \mathbf{X} \cdot \mathbf{W_V}$$

Die drei Matrizen haben dabei für den Attention-Zyklus umgangssprachlich formuliert folgende Fragen zu beantworten:

- **Query-Matrix (Q)**: Was fragt ein Token?
- **Key-Matrix (K)**: Welche Token im Kontext antworten am besten auf die Frage?
- **Value-Matrix (V)**: Welche Informationen fließen letztendlich in den nächsten Schritt ein.

In dem Beispiel könnten die jeweiligen Zeilen von Q, K, V für das Token „Bank“ folgend aussehen:

$$Q_{\text{Bank}} = [1.0, 0.7, 0.9, 1.1] \quad Q_{\text{Bank}} = [0.8, 0.6, 1.0, 0.9] \quad Q_{\text{Bank}} = [0.9, 0.5, 0.7, 1.0]$$

3.1.3 Berechnung der Attention-Scores