DAV/DGVFM
Herbsttagung
2025

Corinna Walk und Yannick Richter, viadico GmbH

Defect Management Mehr Effizienz durch Data-Science-Methoden

Herbsttagung 2025, 17. und 18. November 2025

Agenda

1) Einführung Defect Management
2) Datengrundlage
3) Datenbereinigung und -aufbereitung
4) Ähnlichkeitsmessung von Texten
   - → Term Frequency – Inverse Dokument Frequency (TF-IDF)
   - → Sentence Transformer
5) Fazit

Agenda

1) **Einführung Defect Management**
2) Datengrundlage
3) Datenbereinigung und -aufbereitung
4) Ähnlichkeitsmessung von Texten
   - → Term Frequency – Inverse Dokument Frequency (TF-IDF)
   - → Sentence Transformer
5) Fazit

Defect Management

Definition

Defect Management ist der strukturierte Prozess zur Erfassung, Analyse, Priorisierung, Behebung und Nachverfolgung von Fehlern (Defects) in IT-Systemen und Softwareanwendungen.

Ziel des Defect Managements

Sicherstellung der Qualität, Stabilität und Funktionsfähigkeit von Systemen – durch frühzeitiges und effizientes Beheben von Fehlern im gesamten Entwicklungs- und Betriebsprozess

Beispiele

Jira (Atlassian), HP ALM

Problemstellung & Motivation

Ansatzpunkte für Data Science

- Keine Analyse für ähnliche oder wiederkehrende Tickets
- Mangelnde Transparenz über Test- & Fehlerstatus
- Keine zentrale Wissensnutzung früherer Lösungen

Möglichkeiten

- Tickets mit modernen Data Science Methoden analysieren
- Informationen aus den Tickets besser nutzbar machen
- Vermeidung von redundanter Arbeit

→ Unterstützung durch moderne Data Science Methoden

Use Case: Ähnlichkeitsmessung von Tickets

Ablauf

Datengrundlage

- Einlesen der Tickets aus Defect Managementsystem

Datenbereinigung

- Extrahieren relevanter Texte
- Entfernen von unnötigen Zeichen
- Normalisierung der Texte zur weiteren Verarbeitung

Umwandlung der Texte in Vektoren/Embeddings

- TF – IDF
- Sentence Transformer

Ähnliche Texte identifizieren

- Ermittlung der k ähnlichsten Tickets

Agenda

1) Einführung Defect Management
2) **Datengrundlage**
3) Datenbereinigung und -aufbereitung
4) Ähnlichkeitsmessung von Texten
   - → Term Frequency – Inverse Dokument Frequency (TF-IDF)
   - → Sentence Transformer
5) Fazit

Beispiel: Typische Versicherungstickets

Beispiel

Titel:

Maschinelle Anpassung der Hauptfälligkeit nach Änderung des Versicherungsbeginns

Beschreibung:

Bei einem Vertrag mit dem Tarif XXX wurde der Versicherungsbeginn vom 01.07.2025 auf den 01.08.2025 verschoben Die Ablaufdaten wurden nicht automatisch aktualisiert. Nach manueller Anpassung kam eine wegen abweichender Hauptfälligkeit. Dieses Attribut kann nicht händisch geändert werden.

Beispiel

Titel:

Fehler bei Entnahme

Beschreibung:

VNR XXX, Tarif XXX

Der Kunde wünscht eine Entnahme in der Höhe von 10.000 EUR. Beim Berechnen und bei der Freigabe kommt folgender Fehler:

Entnahme nicht möglich, Vorgang abgebrochen

Datengrundlage

Datenquelle

- Apache Jira Issue Tracking Dataset
  (Open dataset: https://zenodo.org/records/14253918)

→ Über 1 Mio Fehler-Tickets aus verschiedenen Apache-Projekten

Für die Analyse genutzte Felder

- *Summary*: Zusammenfassung des Ticketinhalts
- *Description*: Issue-Beschreibung in Textform

Beobachtungen

- Unterschiedliche Textqualität/-länge
- Teilweise fehlende oder sehr knappe Beschreibungen
- Viele technische Fachbegriffe, Abkürzungen

Beispiel

Summary:

```
Many read errors with parquet files
```

Description:

```
We're seeing failures reading almost any parquet
file we have with Impala 2.0. The files are
generally generated with Scalding and parquet-mr
1.6.0rc3. I've attached a small test file which
exhibits the failures (and is built under the
conditions listed above). When we run {code}
SELECT * from <table> LIMIT 100 {code} We get
errors like the following: {code} Backend
0:Metadata states that in group
hdfs://path/to/data/part-m-00000.parquet[0] there
are 10 rows, but only 0 rows were read. couldn't
deserialize thrift msg: No more data to read.
ParquetScanner: Could not deserialize page header.
{code} And no result is returned.”
```

Agenda

1) Einführung Defect Management
2) Datengrundlage
3) **Datenbereinigung und -aufbereitung**
4) Ähnlichkeitsmessung von Texten
   - → Term Frequency – Inverse Dokument Frequency (TF-IDF)
   - → Sentence Transformer
5) Fazit

Methodik der Textbereinigung

Entfernen von …

… überflüssigen Leerzeichen (Tab, Whitespace, Newline)
… HTML-Formatierungen, Hyperlinks und Dateipfaden
… Sonderzeichen
… Stoppwörtern

Textnormalisierung

- Umwandlung in Kleinbuchstaben
- Zurückführung auf den Wortstamm (Stemming, Lemmatization)
- Rechtschreibkorrektur
- Weitere spezifische Anpassungen für Ziffernfolgen
  z.B. Entfernen von Telefonnummern wie +4917483345

Hier: Verzicht auf Stemming/Lemmatization
→ Verhindert, dass spezifische Fachbegriffe auf falsche Wortstämme zurückgeführt werden

Herausforderungen bei der Textbereinigung

Häufige Fehlerquellen

- **Rechtschreibkorrektur** von Fachbegriffen oder technischen Abkürzungen
  z. B. „BearbID“ → „Barbie“
- Übermäßiges **Entfernen von Sonderzeichen und einzelnen Ziffern**
  z. B. Tarif „A4_H“ → „AH“
- Rückführung auf **falsche Wortstämme**
  z. B. „Übertr.“ → „Über“

→ Mögliche Lösung: Sammlung von **versicherungsspezifischen Abkürzungen und Fachbegriffen**, die im Zuge der Textbereinigung ignoriert werden.

Beispiel

```
Sammlung_Fachbegr = [„SchrittID“, „VOR_POL“, „NACH_POL“, „pruefgr“, … ]
```

Agenda

1) Einführung Defect Management
2) Datengrundlage
3) Datenbereinigung und -aufbereitung
4) **Ähnlichkeitsmessung von Texten**
   - → Term Frequency – Inverse Dokument Frequency (TF-IDF)
   - → Sentence Transformer
5) Fazit

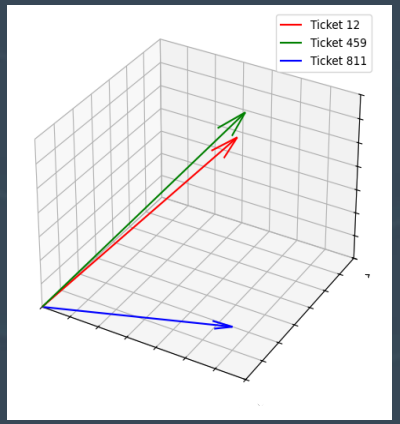
Kosinus-Ähnlichkeit

Kosinus-Ähnlichkeit (*Cosine similarity*)

- Bestimmung des **Kosinus** des zwischen beiden Vektoren **eingeschlossenen Winkels**
  → Sind zwei Vektoren **ähnlich** zueinander, so schließen diese einen **kleinen Winkel** ein.
- Maß für die Ähnlichkeit zweier Vektoren
- Output:
  - 1.0 = identisch ausgerichtet (sehr ähnlich)
  - 0.0 = orthogonal (keine Ähnlichkeit)
  - −1.0 = entgegengesetzt (in der Praxis bei Embeddings selten relevant)

$$\cos\_sim(e_1, e_2) = \frac{\langle e_1, e_2 \rangle}{\|e_1\| \cdot \|e_2\|} \in [-1,1]$$

- $\langle e_1, e_2 \rangle$ Skalarprodukt der Embeddings e_1 und e_2
- $\|e\|$ Norm des Embeddings e



Agenda

1) Einführung Defect Management
2) Datengrundlage
3) Datenbereinigung und -aufbereitung
4) **Ähnlichkeitsmessung von Texten**
   - → **Term Frequency – Inverse Dokument Frequency (TF-IDF)**
   - → Sentence Transformer
5) Fazit

Bag of Words

Bag of Words

Einfaches Modell zur Textrepräsentation, bei dem ein Dokument als **Sammlung** („Beutel“) seiner **Wörter** dargestellt wird

Beispiel

```
[…]
Data = [“Deckungskapital ist falsch berechnet, Deckungskapital prüfen.”,
        “Todesfallsumme ist falsch erfasst, Todesfallsumme prüfen.”,
        “Beitrag ist falsch berechnet, Beitrag prüfen.”]
[…]
```

	deckungskapital	todesfallsumme	beitrag	ist	falsch	berechnet	erfasst	prüfen
Text 1	2	0	0	1	1	1	0	1
Text 2	0	2	0	1	1	0	1	1
Text 3	0	0	2	1	1	1	0	1

Ähnlichkeitsmessung mit TF-IDF

Term Frequency (TF)

- Lokale Gewichtung
- Wie oft kommt das Wort in dem betrachteten Text vor ?

→ Relevanz des Wortes innerhalb eines Textes

Inverse Document Frequency (IDF)

- Globale Gewichtung
- In wie vielen verschiedenen Texten tritt das Wort auf ?

→ Relevanz des Wortes innerhalb des Textkorpus

TF-IDF-Gewichtung

- Kombination aus lokaler und globaler Gewichtung

→ Relevanz des Wortes für den betrachteten Text im gesamten Textkorpus

→ Je **häufiger** ein Wort in einem bestimmten Text vorkommt (**hohe TF**) und je **seltener** es im gesamten Korpus erscheint (**hohe IDF**), desto **relevanter** ist dieses Wort für den Inhalt des Textes.

Ähnlichkeitsmessung mit TF-IDF

Beispiel

```
[…]
Data = [“Deckungskapital ist zu hoch”,
        “Deckungskapital ist nicht zu hoch”,
        “Todesfallsumme ist zu hoch”,
        “Todesfallsumme ist zu niedrig”,
        “Beitrag ist zu niedrig”]
[…]
```

	beitrag	deckungskapital	hoch	ist	nicht	niedrig	todesfallsumme	zu
Text 1	0	0.647293	0.537311	0.382301	0	0	0	0.382301
Text 2	0	0.504883	0.419099	0.298192	0.62579	0	0	0.298192
Text 3	0	0	0.537311	0.382301	0	0	0.647293	0.382301
Text 4	0	0	0	0.359594	0	0.608845	0.608845	0.359594
Text 5	0.68924	0	0	0.328427	0	0.556075	0	0.328427

Ähnlichkeitsmessung mit TF-IDF

Beispiel

Summary: Job should be marked as Falied if it is recovered from commit.

Description: If Resource manager is restarted when a job is in commit state, The job is not able to be recovered

after RM restart and it is marked as Killed.

The job status should be Failed instead killed

Ausgabe:

Ticket 620178 | Ähnlichkeit: 0.9137

Summary: Job should be marked as Failed if it is recovered from commit.

Description: If Resource manager is restarted when a job is in commit state, The job is not able to recovered

after RM restart and it is marked as Killed.

The job status should be Failed instead killed.

Ticket 625118 | Ähnlichkeit: 0.3751

Summary: Streaming: when a job is killed, the message should say it was "killed" rather than "failed„

Description: None

Ähnlichkeitsmessung mit TF-IDF

Beispiel

Summary: Job should be marked as Falied if it is recovered from commit.

Description: If Resource manager is restarted when a job is in commit state, The job is not able to be recovered

after RM restart and it is marked as Killed.

The jo
killed

Tickets, die sich nur in einzelnen Wörtern unterscheiden (Rechtschreibfehler, Lücken, …), werden definitiv gefunden und haben eine sehr hohe Ähnlichkeit

Ausgabe:

```
Ticket 620178 | Ähnlichkeit: 0.9137

Summary: Job should be marked as Failed if it is
recovered from commit.

Description: If Resource manager is restarted when
a job is in commit state, The job is not able to
recovered

after RM restart and it is marked as Killed.

The job status should be Failed instead killed.

_____________________________________________

Ticket 625118 | Ähnlichkeit: 0.3751

Summary: Streaming: when a job is killed, the
message should say it was "killed" rather than
"failed„

Description: None
```

Ähnlichkeitsmessung mit TF-IDF

Beispiel

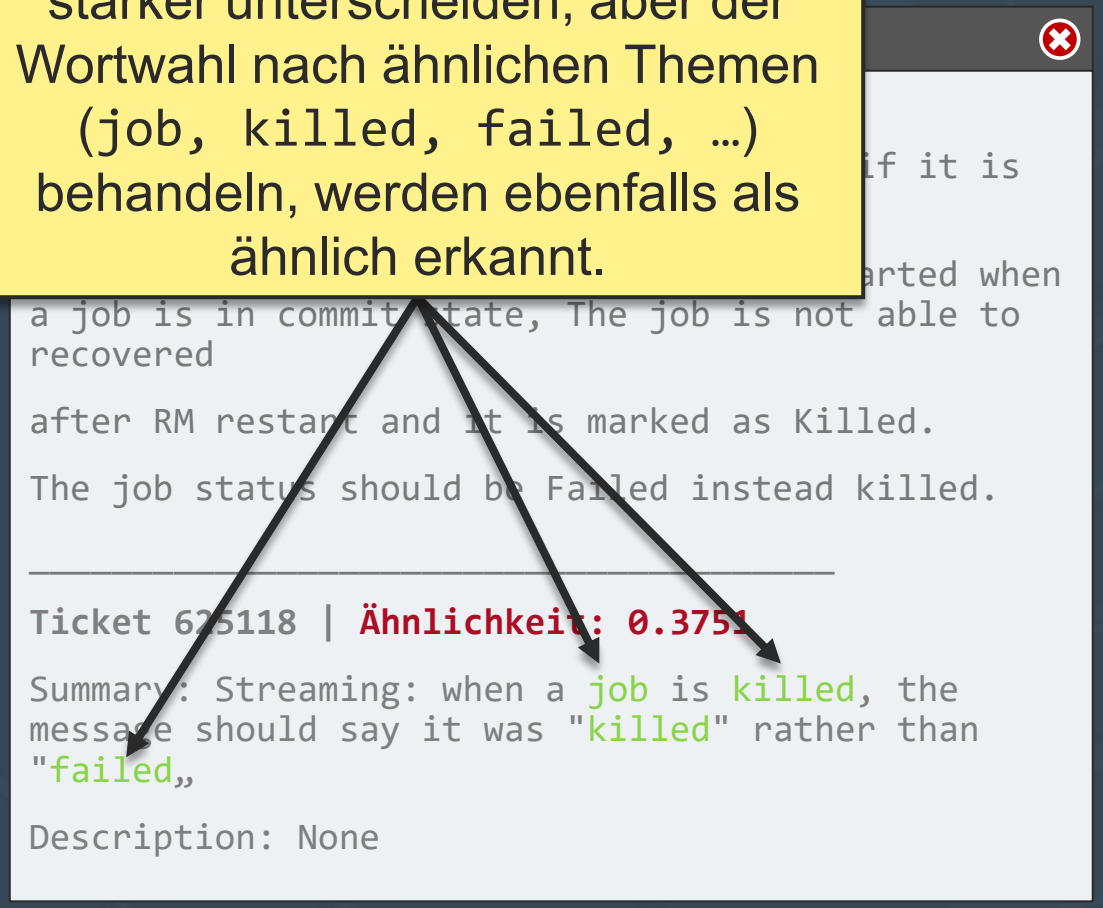
Summary: Job should be marked as Falied if it is recovered from commit.

Description: If Resource manager is restarted when a job is in commit state, The job is not able to be recovered

after RM restart and it is marked as Killed.

The job status should be Failed instead killed

Tickets, die sich vom Text her stärker unterscheiden, aber der Wortwahl nach ähnlichen Themen (`job, killed, failed, …`) behandeln, werden ebenfalls als ähnlich erkannt.



Probleme bei der Nutzung von TF-IDF

Beispiel

```
Text_1 = „Die Police deckt Beschädigungen am Haus durch Feuer ab.“
Text_2 = „Im Brandfall ersetzt die Versicherung Schäden am Eigenheim.“
```

Semantische Ähnlichkeiten

- **Unterscheiden** sich Texte in ihrer **Wortwahl**, so werden diese durch TF-IDF nicht als ähnlich erkannt.
- Texte müssen nicht zwangsläufig die **gleichen Wörter** verwenden, um ähnliche Inhalte darzustellen.

→ **Semantisch ähnliche Texte** mit unterschiedlicher Wortwahl werden durch TF-IDF **nicht erkannt.**

Agenda

1) Einführung Defect Management
2) Datengrundlage
3) Datenbereinigung und -aufbereitung
4) **Ähnlichkeitsmessung von Texten**
   - → Term Frequency – Inverse Dokument Frequency (TF-IDF)
   - → **Sentence Transformer**
5) Fazit

Mathematische Verarbeitung von Sprache

Beispiel

Wie ähnlich sind sich die Begriffe “Berufsunfähigkeitsversicherung”, “Risikolebensversicherung” und “KFZ-Versicherung”?

Word Embedding

Numerische Repräsentation eines **Wortes**, dem jeweils ein **Vektor** $v \in \mathbb{R}^n$ zugeordnet wird.

Die Wörter könnten beispielhaft anhand folgender Eigenschaften (Dimensionen) beschrieben werden:

1. **Abdeckung biometrischer Risiken**
2. **Notwendigkeit einer Gesundheitsprüfung**
3. **Leistung im Todesfall**
4. **Ist eine Versicherung**

Mathematische Verarbeitung von Sprache

Beispiel

	BU-Versicherung	Risikolebens-Versicherung	KFZ-Versicherung
Abdeckung biometrischer Risiken	1	1	0
Notwendigkeit einer Gesundheitsprüfung	1	1	0
Leistung Im Todesfall	0	1	0
Ist eine Versicherung	1	1	1

Mathematische Verarbeitung von Sprache

Kosinus-Ähnlichkeit (*Cosine similarity*)

- Bestimmung des **Kosinus** des zwischen beiden Vektoren **eingeschlossenen Winkels**
  → Sind zwei Vektoren **ähnlich** zueinander, so schließen diese einen **kleinen Winkel** ein.
- Maß für die Ähnlichkeit zweier Vektoren

Beispiel

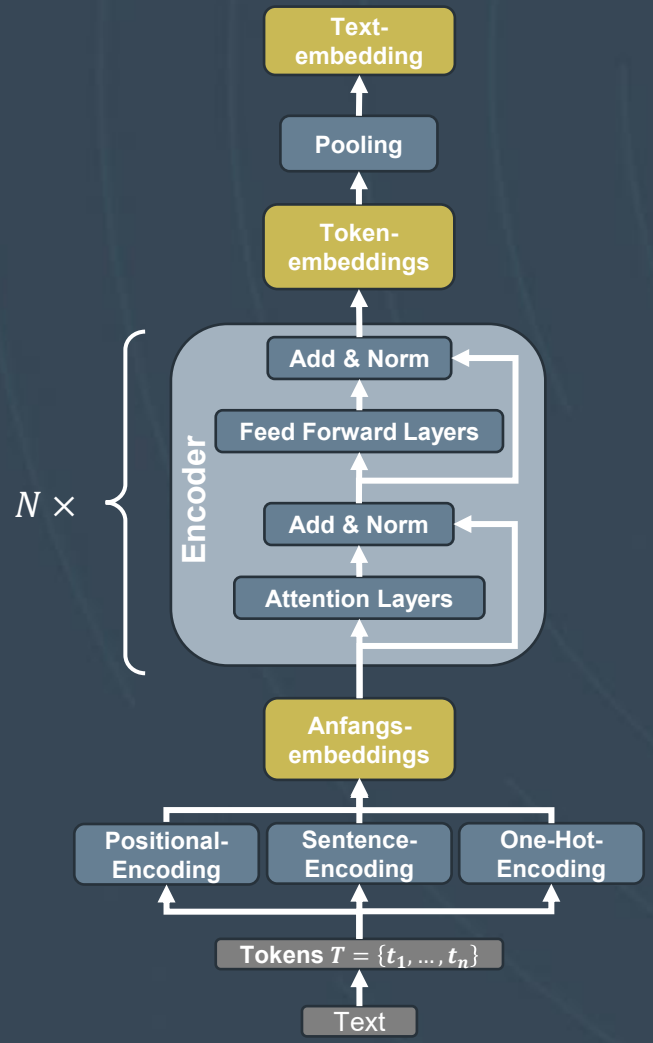
Betrachten wir die Vektoren aus dem vorherigen Beispiel:

$$\vec{v}_{BU} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}; \vec{v}_{Ris} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \text{ und } \vec{v}_{KFZ} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

Es kann auch einem **ganzen Text** eine **Repräsentation** in Form eines **Vektors** zugeordnet werden.

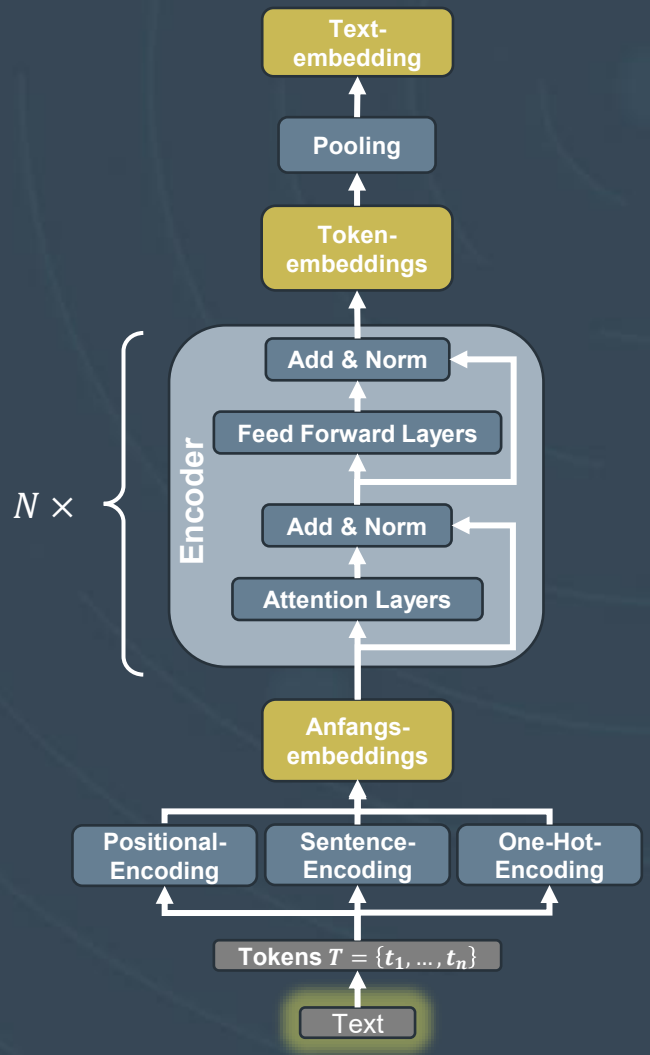
Dann ist $\cos \measuredangle(\vec{v}_{BU}, \vec{v}_{Ris}) = 0{,}866$ und $\cos \measuredangle(\vec{v}_{BU}, \vec{v}_{KFZ}) = 0{,}5$.

Ähnlichkeitsmessung mit Sentence Transformern

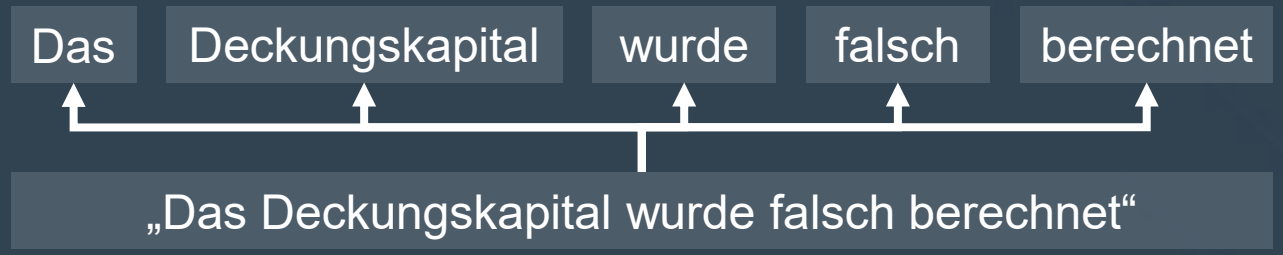


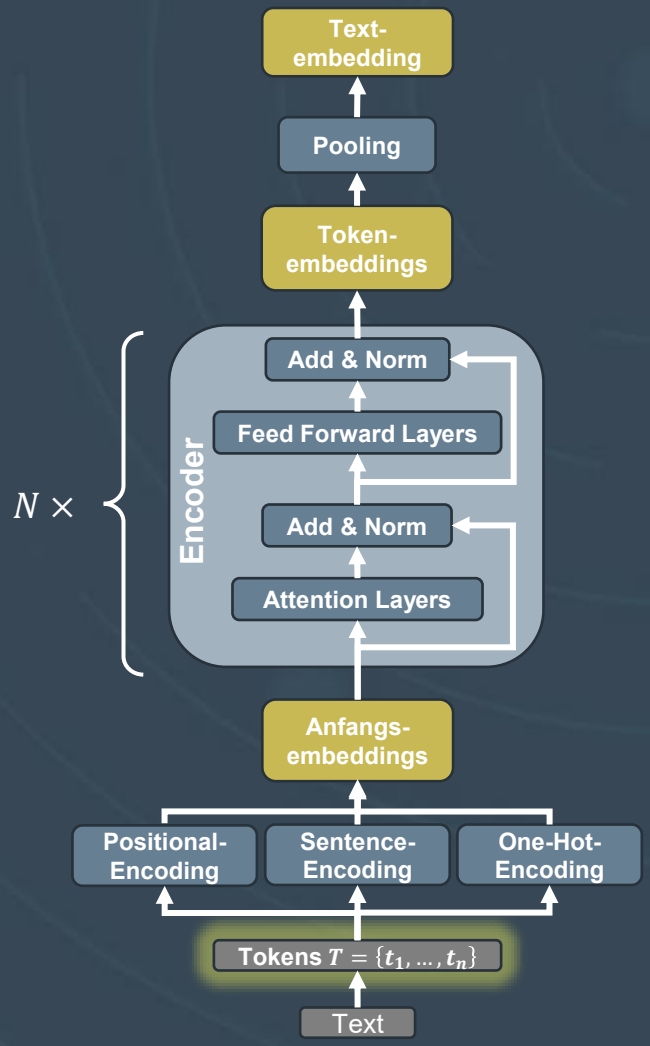
Ähnlichkeitsmessung mit Sentence Transformern

„Das Deckungskapital wurde falsch berechnet“

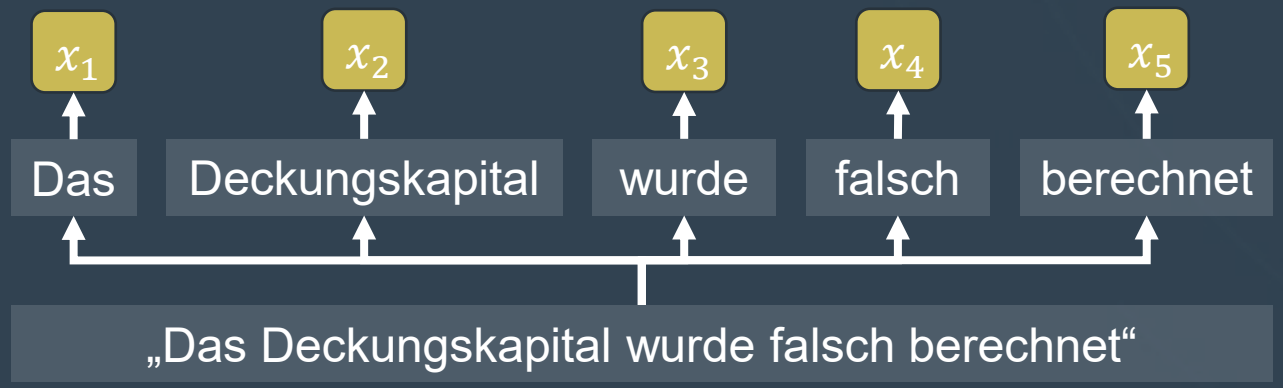


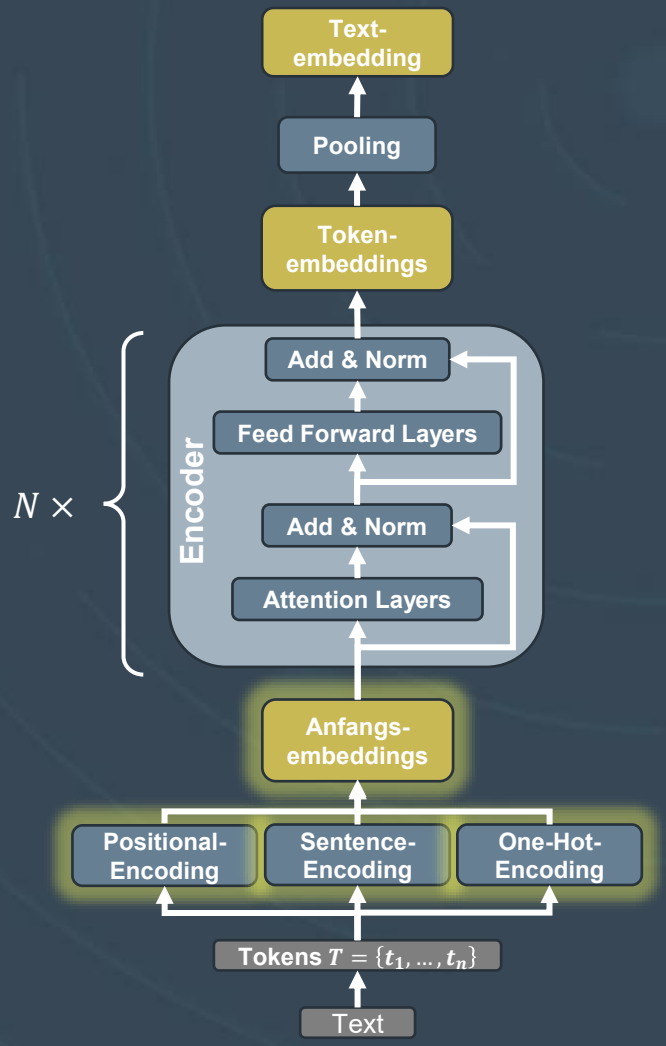
Ähnlichkeitsmessung mit Sentence Transformern



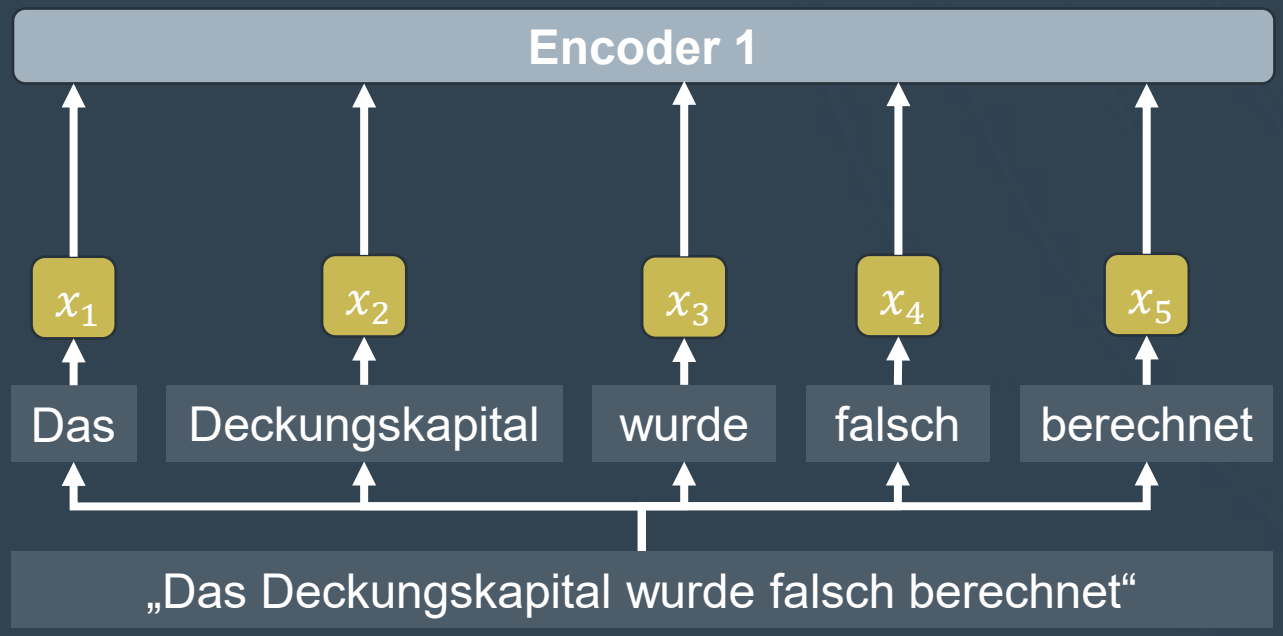


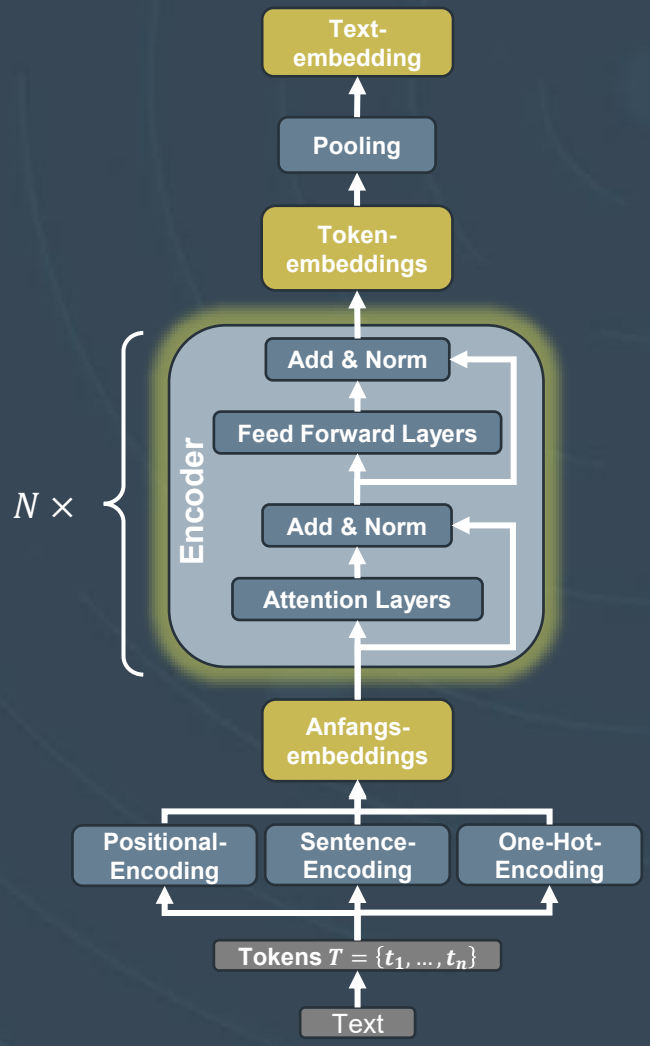
Ähnlichkeitsmessung mit Sentence Transformern



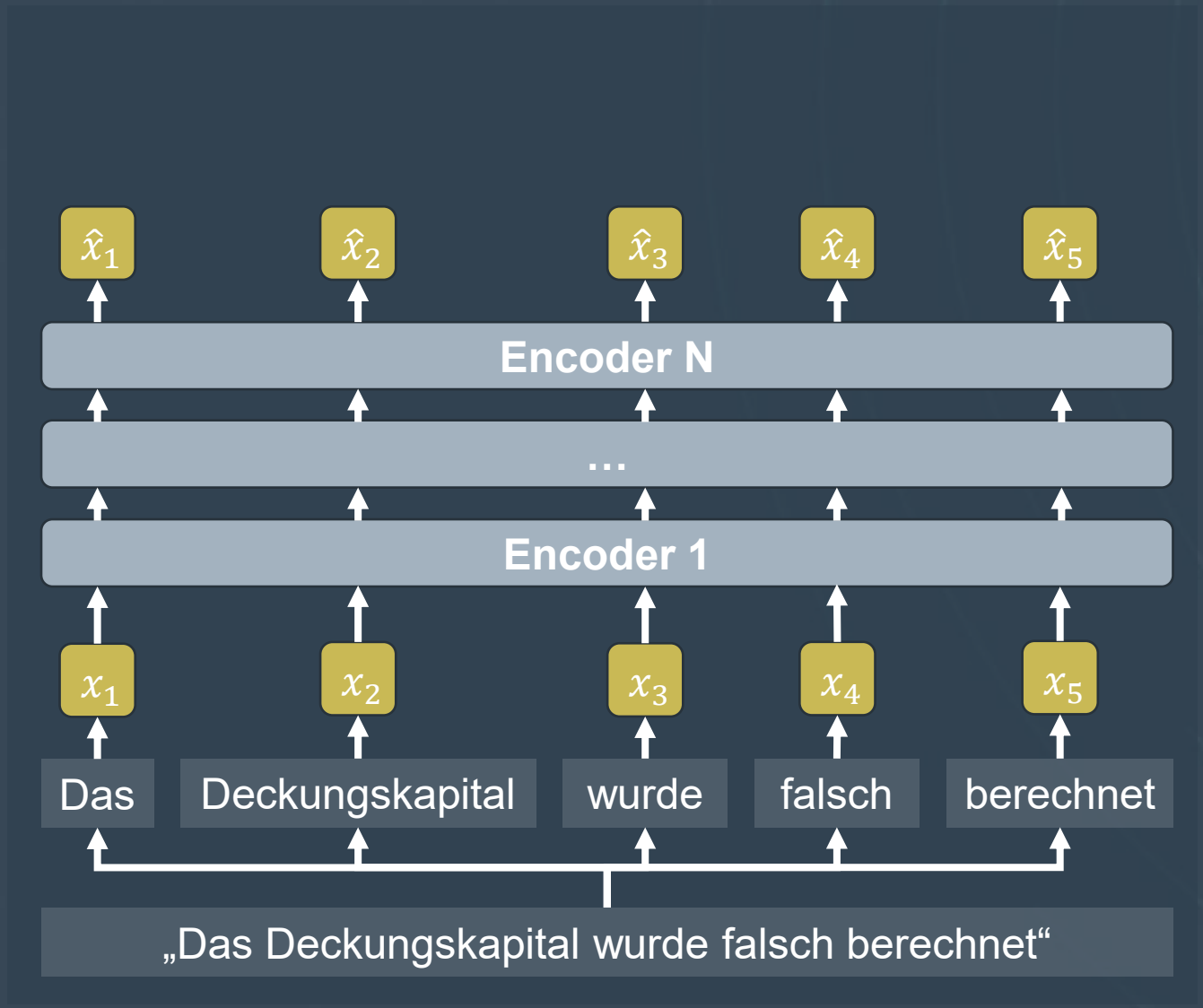


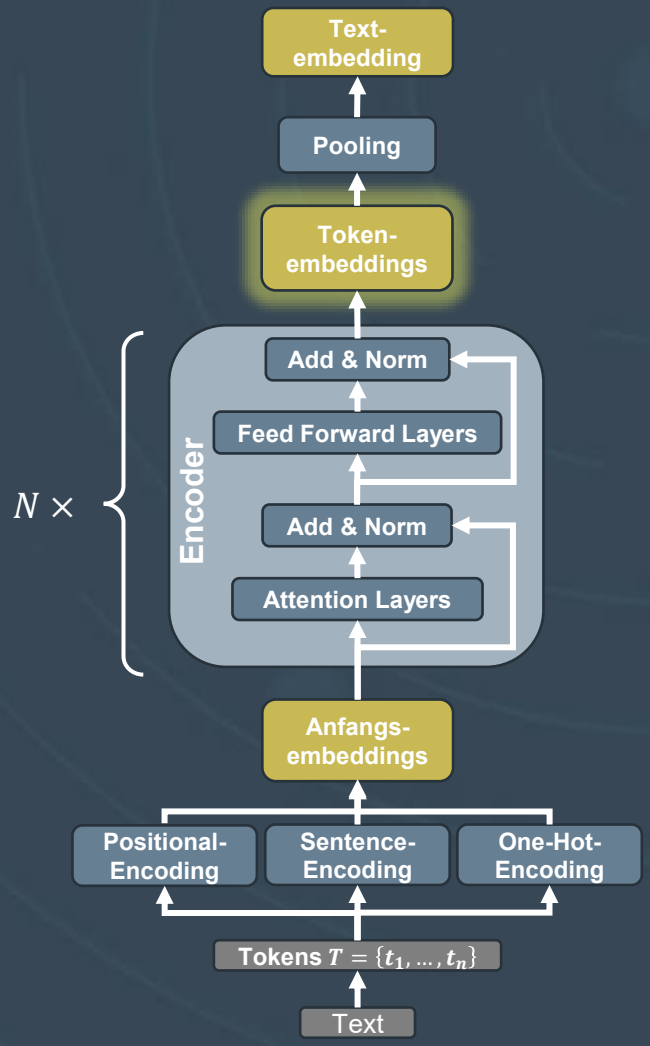
Ähnlichkeitsmessung mit Sentence Transformern



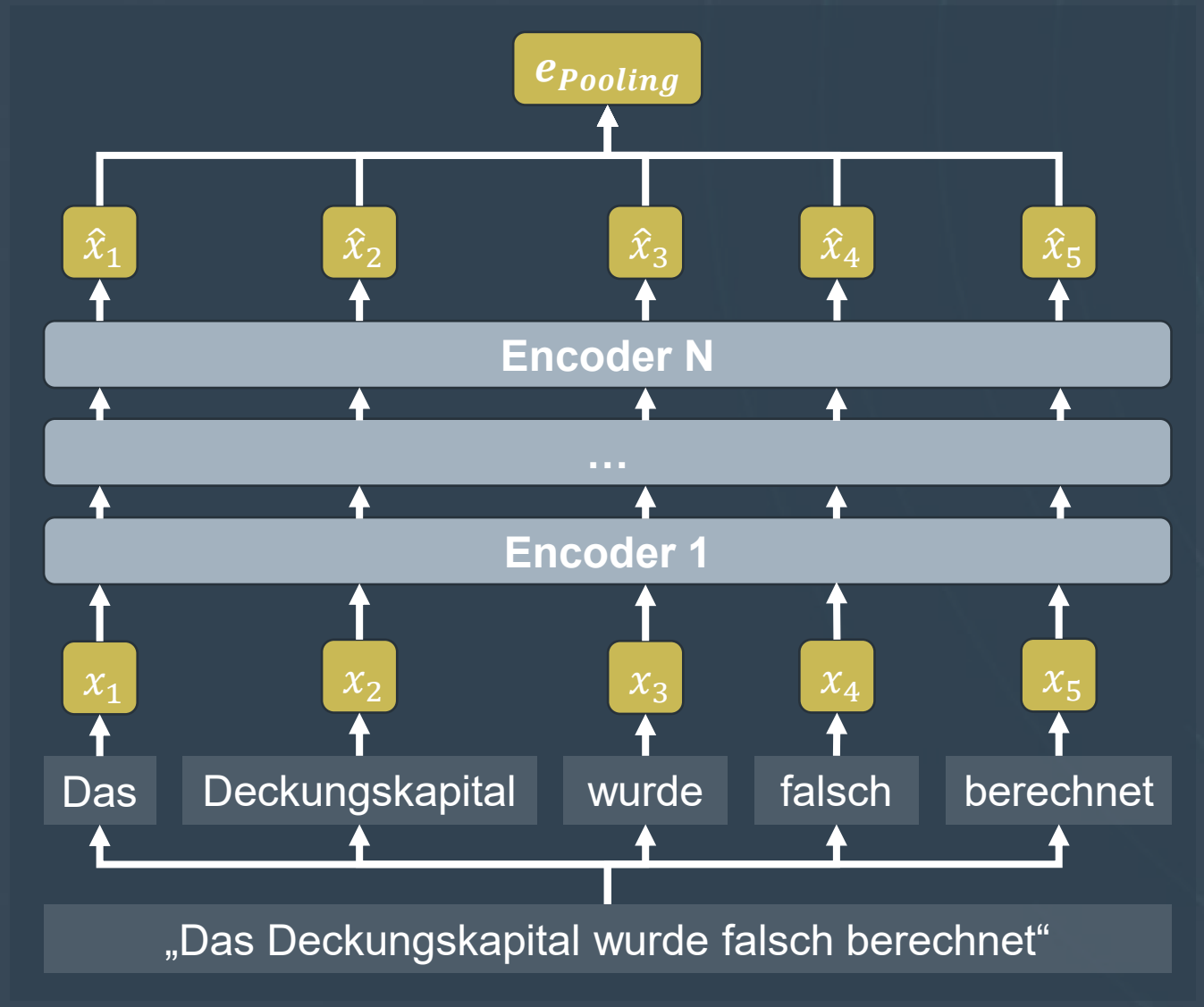


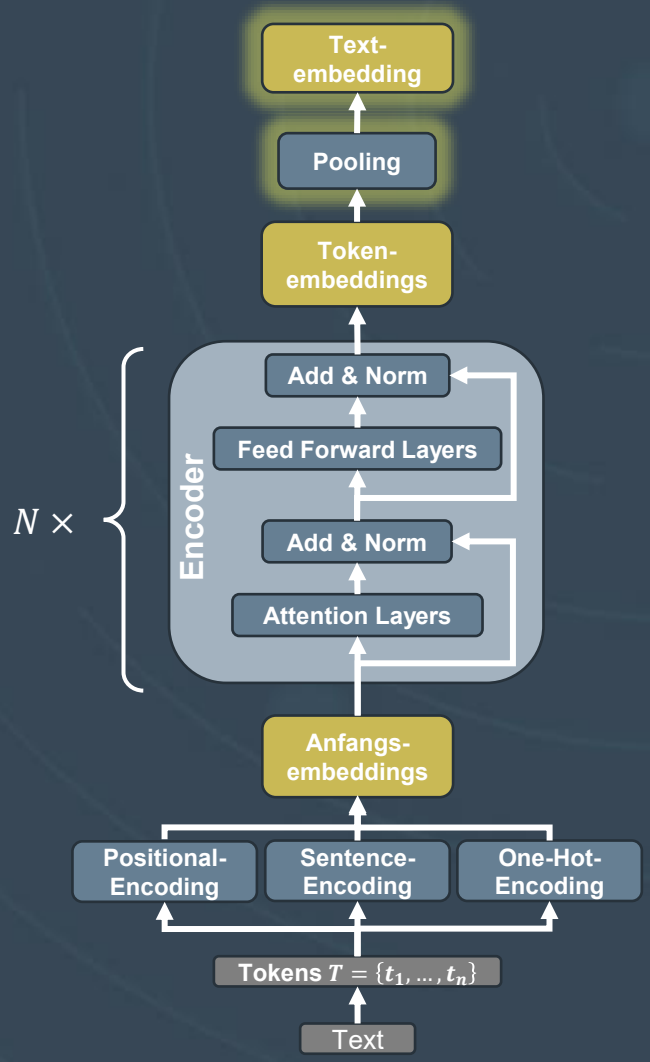
Ähnlichkeitsmessung mit Sentence Transformern





Ähnlichkeitsmessung mit Sentence Transformern





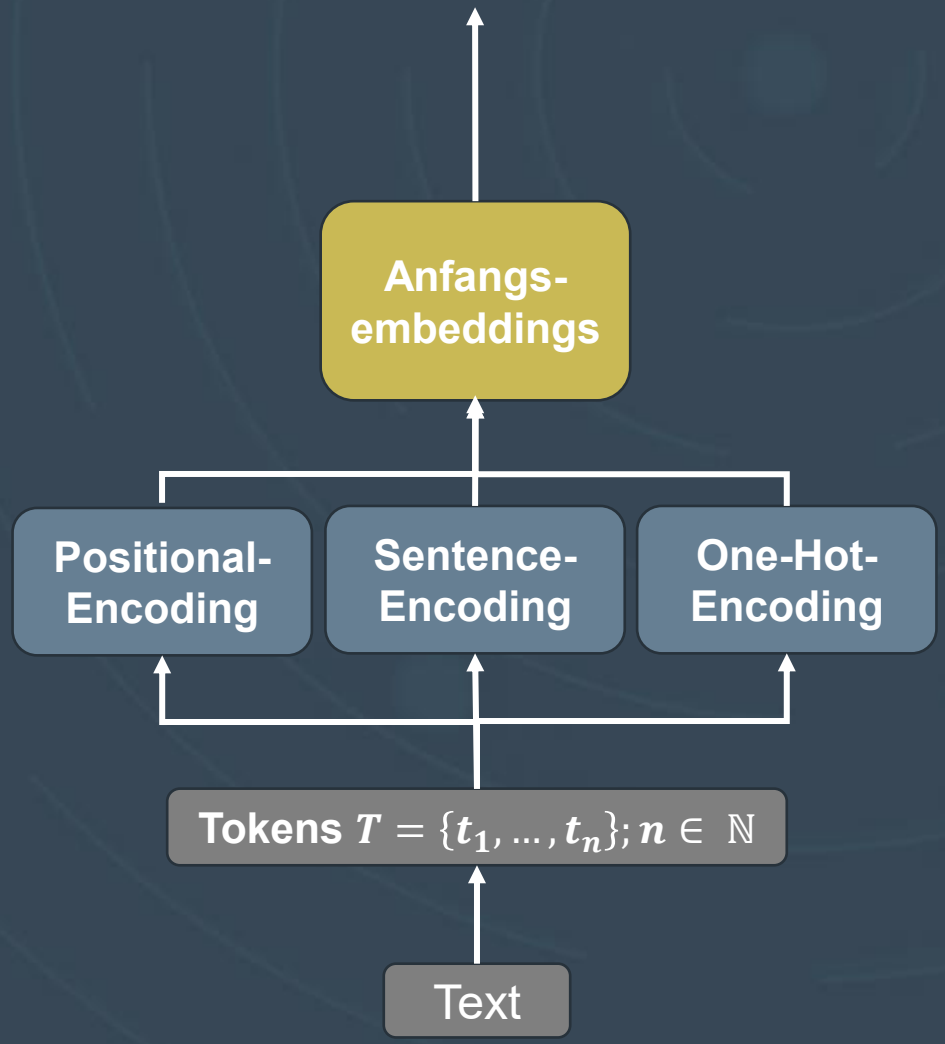
Funktionsweise des Modells

- Aufteilung des Texts in ***Tokens***
- Mapping der *Tokens* auf zugehörige **Anfangsembeddings**

$$x_i \in \mathbb{R}^d,\ d \in \mathbb{N}$$

→ Bildet **Bedeutung** des Worts mathematisch ab

- Zusätzlich fließt die **Position** des Worts innerhalb des Texts mit in die Embeddingvorschrift mit ein



Funktionsweise des Modells: Attention-Layer

- **Aktualisierung** der **Anfangs-embeddings**

$$x_i \in \mathbb{R}^d, d \in \mathbb{N}$$

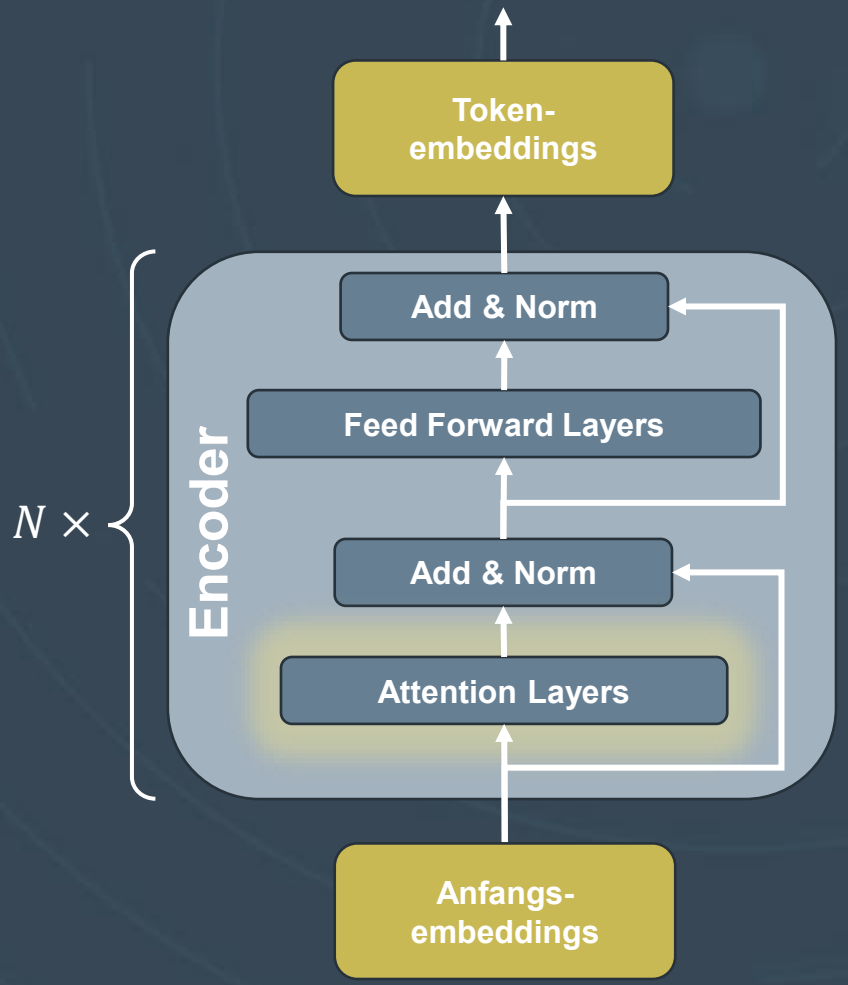
durch Attention Layer

→Erfassen der **Bedeutung** eines Wortes **im Kontext** des Texts.

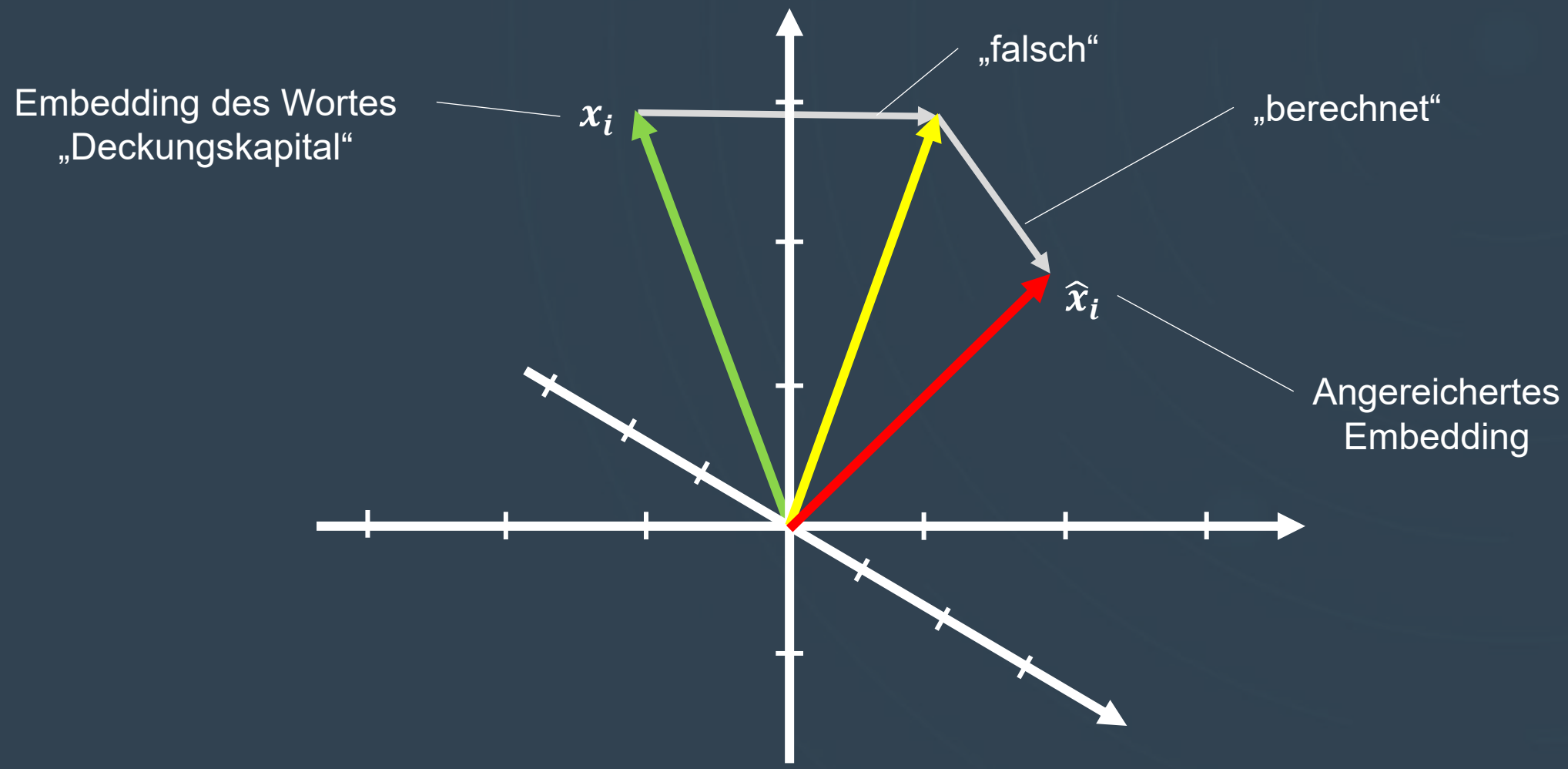
- Ziel: **Anreichern** der Anfangs-embeddings mit Informationen der anderen Wörter

→ Resultat: **angepasste Vektorrepräsentationen**

$$\hat{x}_i \in \mathbb{R}^d$$



Funktionsweise des Modells: Attention-Layer



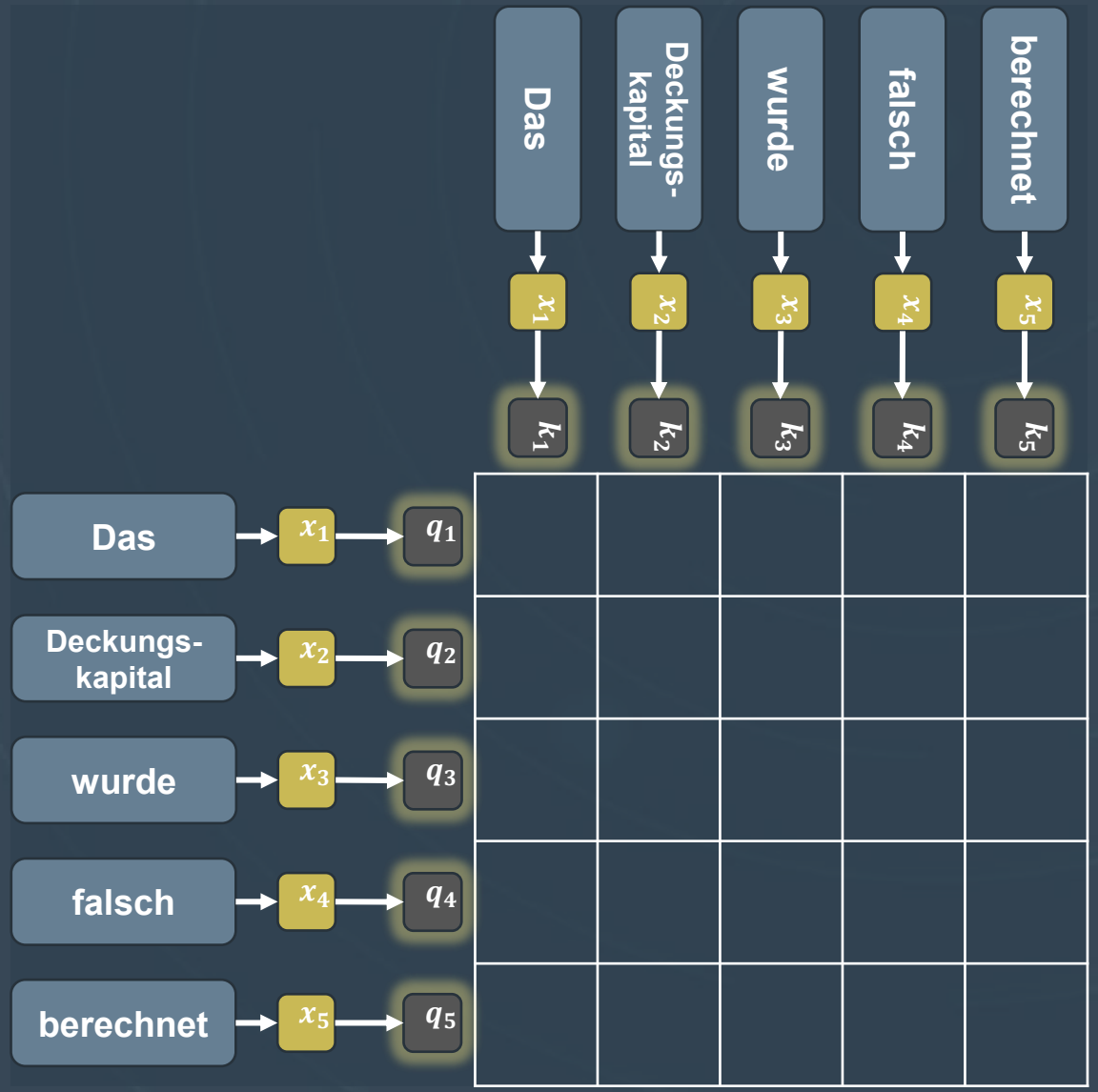
Funktionsweise des Modells: Attention-Layer

- **Kodierung des *i*-ten Wortes** in eine ***Query*** $q_i \in \mathbb{R}^{d_k}$, einen ***Key*** $\mathrm{k}_i \in \mathbb{R}^{d_k}$ und einen ***Value*** $v_i \in \mathbb{R}^{d_v}$ durch **gelernte Gewichtsmatrizen** $W_Q, W_k \in \mathbb{R}^{d \times \mathrm{d}_k}, W_v \in \mathbb{R}^{d \times \mathrm{d}_v}$

- Für die Matrix $\mathrm{X} = \begin{bmatrix} x_1 \\ \dots \\ x_n \end{bmatrix} \in \mathbb{R}^{n \times d}$ der **Anfangsembeddings** erhält man z.B. die ***Query*-Matrix** durch

$$\mathrm{Q} = \mathrm{X} \cdot W_Q = \begin{bmatrix} q_1 \\ \dots \\ q_n \end{bmatrix} \in \mathbb{R}^{n \times d_k}$$

(Analog: ***Key*-Matrix** K und **Value-Matrix** V)



Funktionsweise des Modells: Attention-Layer

- Aus den Matrizen Q und K (*Queries* und *Keys* aller Token) erhält man die **Ähnlichkeitsmatrix**

$$S = \frac{QK^T}{\sqrt{d_k}}$$

→Ein **Eintrag** s_{ij} von S gibt dabei an, wie **relevant** das durch k_j kodierte j-te Wort für den **Kontext des i-ten Wortes** ist

- Anschaulich: q_i ist die **kodierte Anfrage des i-ten Wortes** an die anderen Wörter des Textes und k_j die Antwort darauf.

→ Je **höher der Wert** von s_{ij}, desto **relevanter** ist das Wort j für den **Kontext** des Wortes i.

Funktionsweise des Modells: Attention-Layer

- Anwendung von $softmax$ auf die Ähnlichkeitsmatrix, damit die Relevanz-Scores in $[0,1]$ liegen und sich zeilenweise zu 1 summieren.

$$A = \text{softmax}(S) \in \mathbb{R}^{n \times n}$$

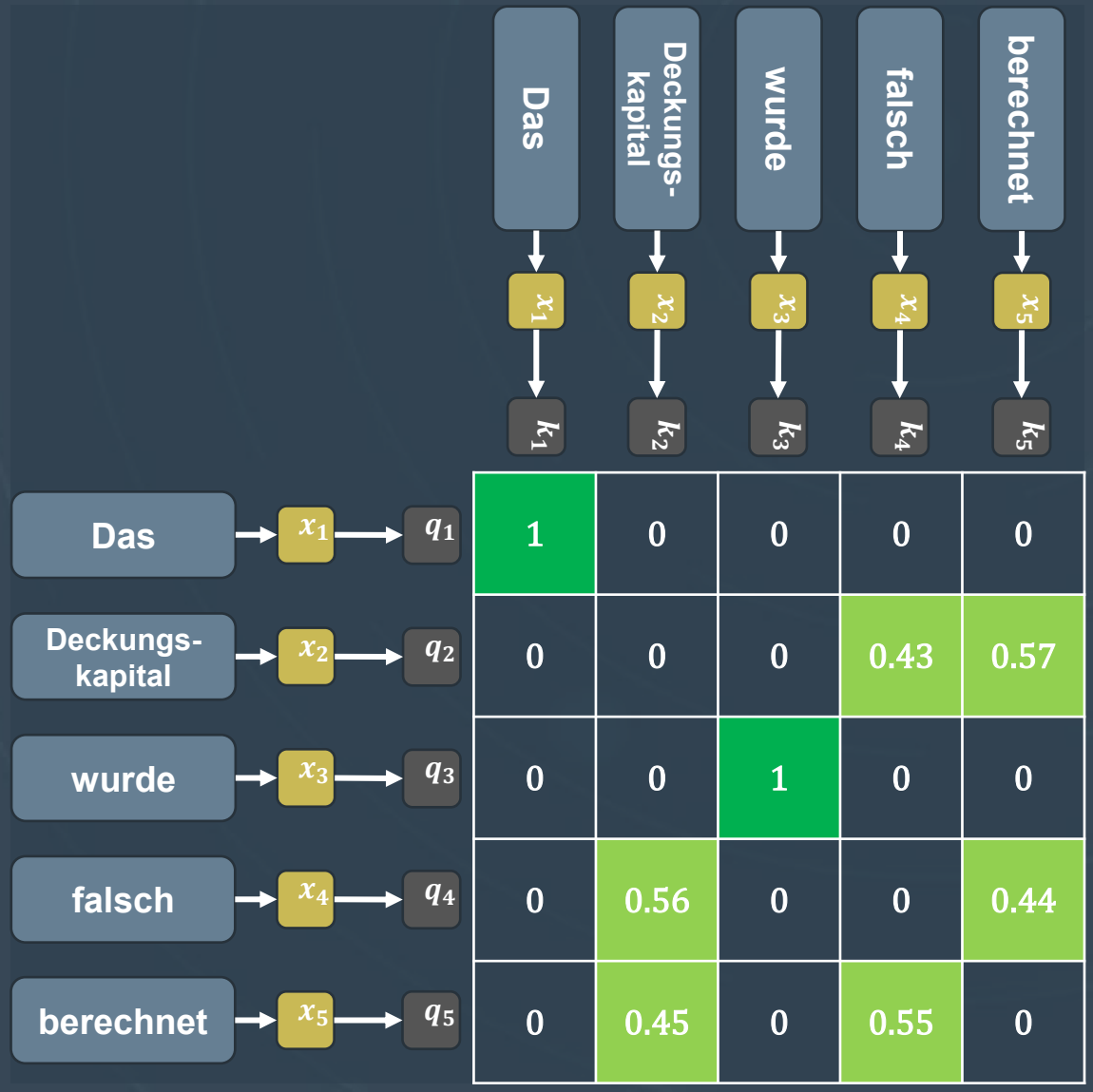
Funktionsweise des Modells: Attention-Layer

- Anwendung von $softmax$ auf die Ähnlichkeitsmatrix, damit die Relevanz-Scores in $[0,1]$ liegen und sich zeilenweise zu 1 summieren.

$$A = \text{softmax}(S) \in \mathbb{R}^{n \times n}$$

Softmax:

$$\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$$



Funktionsweise des Modells: Attention-Layer

- Anwendung von $softmax$ auf die Ähnlichkeitsmatrix, damit die Relevanz-Scores in $[0,1]$ liegen und sich zeilenweise zu 1 summieren.

$$A = \text{softmax}(S) \in \mathbb{R}^{n \times n}$$

- **Multiplikation** der **normierten Ähnlichkeitsmatrix** A mit der ***Value*-Matrix** V ergibt die **Änderungsmatrix**

$$\Delta X = A \cdot V \in \mathbb{R}^{n \times d_v}$$

→ Der Eintrag v_j gibt an, wie die **Bedeutung des i-ten Wortes** durch das **j-te Wort verändert** werden soll, **gewichtet** durch den **Relevanz-Score** a_{ij}.

Funktionsweise des Modells: Attention

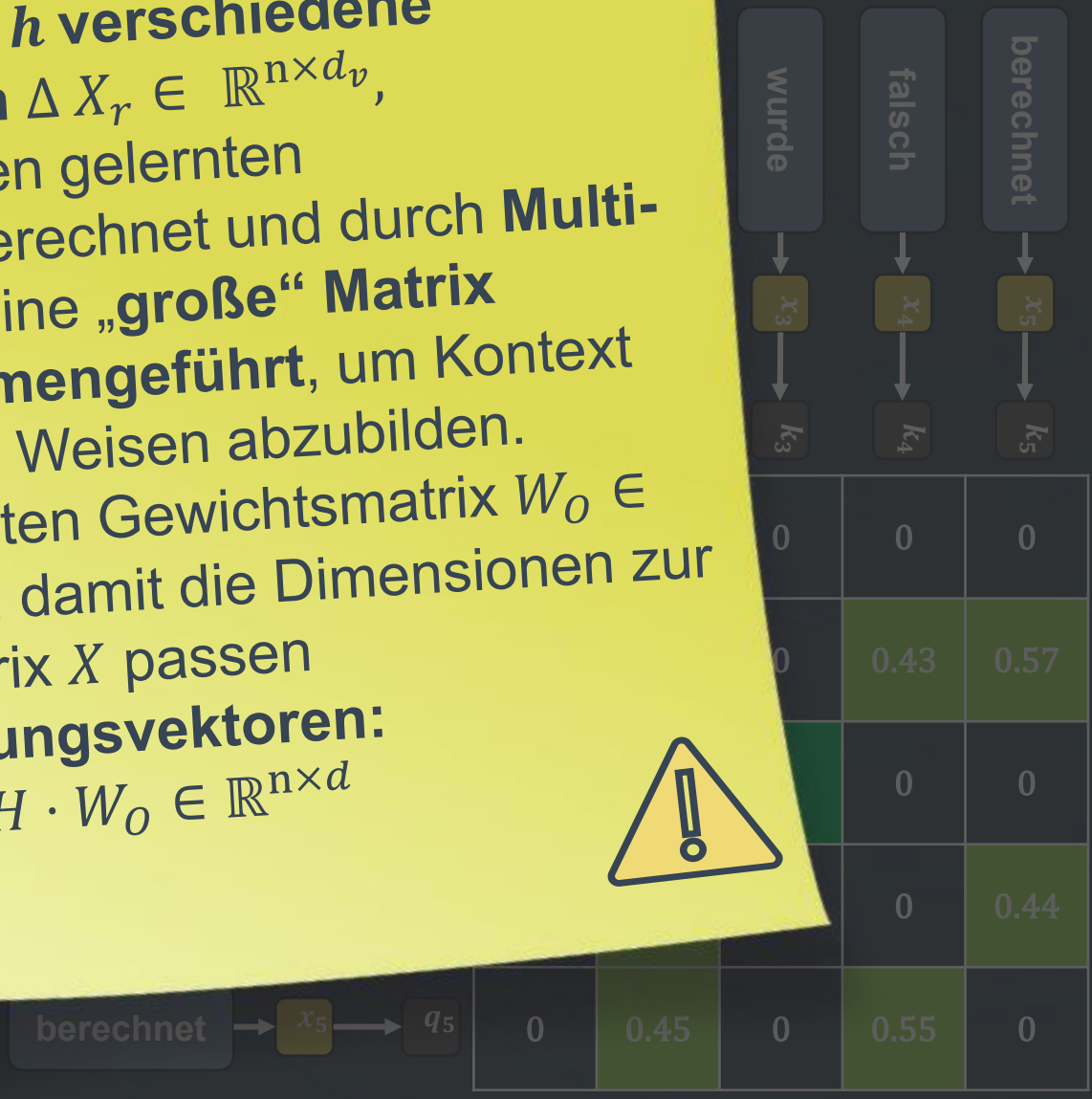
- Anwendung
  Ähnlichkeitsm
  Scores in [0,
  zeilenweise z
  $A = s$
- Multiplikation
  Ähnlichkeitsm
  Matrix V ergib
  $\Delta X =$

→ Der Eintrag v_j
Bedeutung des i-te
Wort verändert werd
durch den Relevanz-

- In der Praxis werden h **verschiedene Änderungsmatrizen** $\Delta X_r \in \mathbb{R}^{n \times d_v}$, $r = 1, \ldots, h$ mit eigenen gelernten Gewichtsmatrizen berechnet und durch **Multi-Head-Attention** in eine **„große" Matrix** $\boldsymbol{H} \in \mathbb{R}^{n \times h \cdot d_v}$ **zusammengeführt**, um Kontext auf unterschiedliche Weisen abzubilden.
- H wird mit einer letzten Gewichtsmatrix $W_O \in \mathbb{R}^{h \cdot d_v \times d}$ multipliziert, damit die Dimensionen zur ursprünglichen Matrix X passen

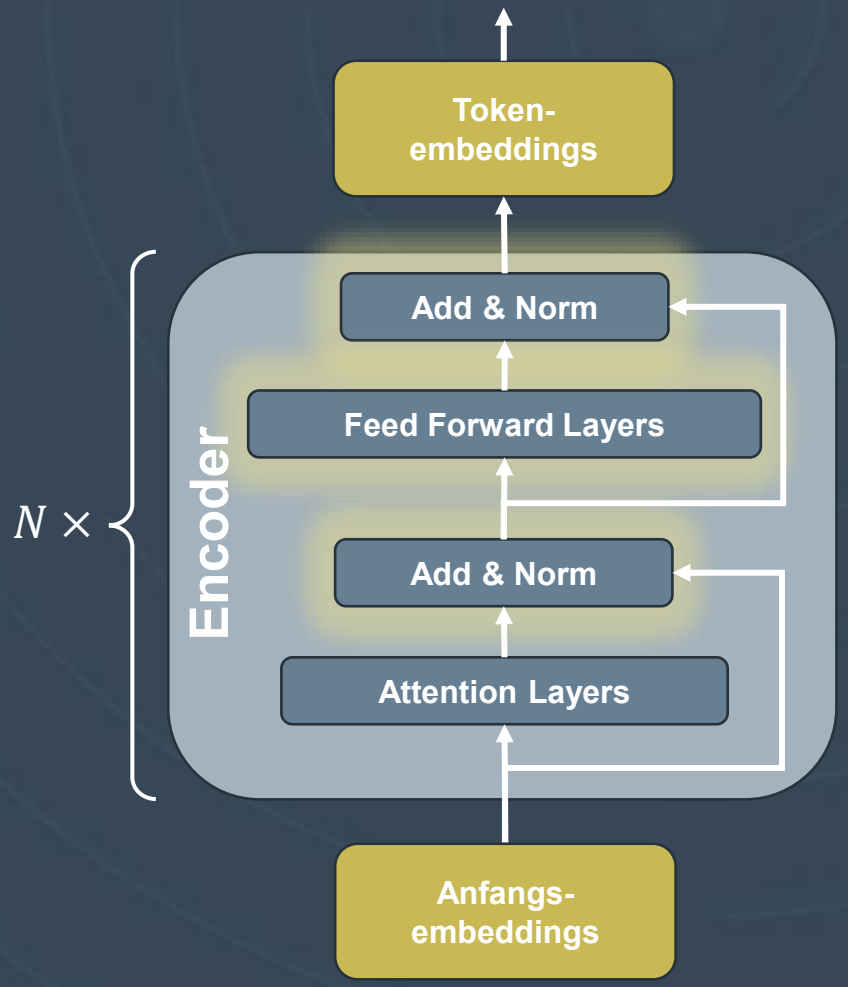
→**Endgültige Änderungsvektoren:**

$$\Delta\tilde{X} = H \cdot W_O \in \mathbb{R}^{n \times d}$$



Funktionsweise des Modells: Normalisierung und FFN

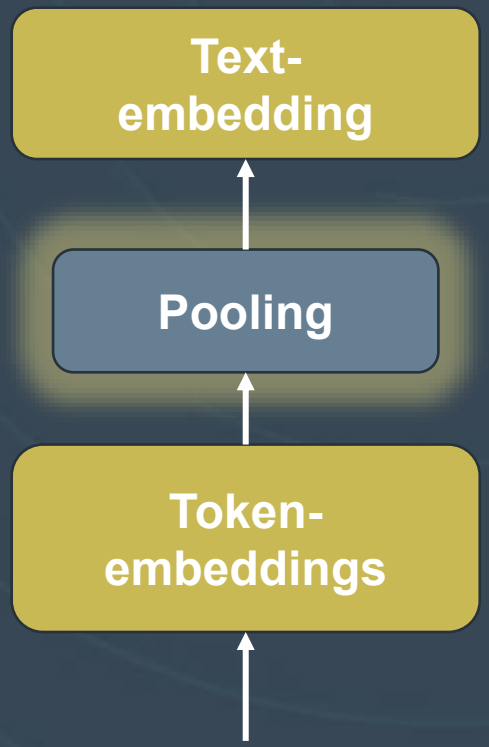
- Die **mit Kontext angereicherten** Embeddings durchlaufen anschließend eine ***Layer-Normalization***, in welcher die Embeddings normalisiert und skaliert werden.
- Die normalisierten Embeddings werden anschließend positionsweise in einem **Feed-Forward Network** (*FFN*) **transformiert**.
  → **komplexere nicht-lineare Merkmalsdarstellungen** der einzelnen Wörter können abgebildet werden



Funktionsweise des Modells: Normalisierung und FFN

- Im ***Pooling***-Layer werden die **einzelnen Token-Embeddings** zu einem **gesamten Textembedding** zusammengeführt.
- z.B. *Mean-Pooling*: Hierbei wird der Mittelwert aller Embeddings $e_1, \ldots, e_n$ für $n \in \mathbb{N}$ gebildet

$$e_{mean} = \frac{1}{n} \cdot \sum_{i=1}^{n} e_i$$



Verwendetes Modell

Sentence - BERT

→ **Sentence B**idirectional **E**ncoder **R**epresentations from **T**ransformers

- Spezialisierung der Transformer-Architektur, die **nur den Encoder** nutzt.
- Sprachmodell für die Erstellung **komplexer Word-Embeddings**, die Semantik und Syntax besser abbilden können.
- Ermöglicht das Erfassen **bidirektionalen Kontexts**.
- Trainiert mit **Masked Language Modelling** (MLM)

Verwendete Parameter:

- **Embedding-Dimension** $d = 384$
- **Anzahl Attention-Heads** $h = 6$
- **Transformer-Encoder-Stacks** $N = 6$

Ähnlichkeitsmessung mit Sentence Transformern

Beispiel

Summary:
TestMoveApplication.testMoveRejectedByScheduler randomly fails

Description:
TestMoveApplication#testMoveRejectedByScheduler fails because of NullPointerException. It looks caused by unhandled exception handling at server-side.

Ausgabe:

```
Ticket 476344

Summary: TestTCTLSeparatedProtocol is broken

Description: Some of the tests in TestTCTLSeparatedProtocol throws
a NullPointedException that is catched, logged and not rethrown.
This means the tests don't fail even though an unexpected exception
is thrown.


_____________________________

Ticket 389599

Summary: testHighMemoryJobWithInvalidRequirements in
TestCapacityScheduler fails with NullPointerException

Description: testHighMemoryJobWithInvalidRequirements fails
consistently with NullPointerException when TestCapacityScheduler
is run.


_____________________________________

Ticket 833848

Summary: AutoCommitTest randomly failed with NullPointerException

Description: Saw a failure of AutoCommitTest with a NullPointer
exception. Couldn't replicate easily. Noticed thatAutoCommit prints
that same NullPointer exception many times even when it doesnt
fail.
```

Ähnlichkeitsmessung mit Sentence Transformern

Beispiel

Summary:
TestMoveApplication.testMoveRejectedByScheduler randomly fails

Description:
TestMoveApplication#testMoveRejectedByScheduler fails because of NullPointerException. It looks caused by unhandled exception handling at server-side.

Sentence Transformer „verstehen“, dass es sich in den Tickets um „Test“-Funktionen handelt, bei welchen Probleme aufgrund einer NullPointerException auftreten.

Ausgabe:

```
Ticket 476344

Summary: TestTCTLSeparatedProtocol is broken

Description: Some of the tests in TestTCTLSeparatedProtocol throws
a NullPointedException that is catched, logged and not rethrown.
This means the tests don't fail even though an unexpected exception
is thrown.

_____________________________

Ticket 389599

Summary: testHighMemoryJobWithInvalidRequirements in
TestCapacityScheduler fails with NullPointerException

Description: testHighMemoryJobWithInvalidRequirements fails
consistently with NullPointerException when TestCapacityScheduler
is run.

_____________________________________

Ticket 833848

Summary: AutoCommitTest randomly failed with NullPointerException

Description: Saw a failure of AutoCommitTest with a NullPointer
exception. Couldn't replicate easily. Noticed thatAutoCommit prints
that same NullPointer exception many times even when it doesnt
fail.
```

Interessante Funde bei der Ähnlichkeitsanalyse

Beispiel

Summary:

Buy Oxycontin No Prescription Needed | Buy Oxycontin Overnight

Description:

What is Oxycontin?

[Oxycontin | LINK] is a medicine that helps to relieve severe pain.

Oxycontin is a brand name for the drug oxycodone. Oxycodone is associated with a class of drugs known as opioid analgesics. These drugs act on the brain to change how your body feels and responds to pain.

[*GET 10% OFF USE CODE “SALE10”*| LINK]

[…]

Ausgabe:

Summary: Buy Tramadol 100mg Online Without Prescription | Mexican Pharmacy Store

Description:
What is the use of Tramadol pills?

In the United States, there are a lot of pain medications that a doctor can prescribe, and Tramadol is the most popular one. It is available in the form of oral tablets that comes in generic (like Tramadol) and brand (like Ultram) drugs as well.

The work of Tramadol, as a medication, is to provide relief from the ongoing pain in a human body. If the pain is severe or mild, you will be given the dosage as per the situation keeping in mind the various other factors.

[>>Buy Tramadol 100mg Online Without Prescription<<Click Here<<|LINK]*[…]

Agenda

1) Einführung Defect Management
2) Datengrundlage
3) Datenbereinigung und -aufbereitung
4) Ähnlichkeitsmessung von Texten
   - → Term Frequency – Inverse Dokument Frequency (TF-IDF)
   - → Sentence Transformer
5) **Fazit**

Fazit

Ausblick & Weiterentwicklung

- **Ticketwissen** systematisch nutzbar machen – aus Historie lernen statt Einzelfall lösen
- **Transparenz** über potentiell fehleranfällige Bereiche erhöhen
- **Unterstützung** bei der Erfassung neuer Tickets und beschleunigte Analyse bestehender Tickets
- **Integration in Ticket-Systeme** (z. B. Jira) mit automatischer Ähnlichkeitserkennung– gibt es bereits ähnliche Tickets?
- **Chat-Bot** auf der Ticket-Datenbank für schnellen Self-Service
- **Automatisierte Erstbewertung** neuer Tickets (z.B. Fehlerschwere, Priorität, Routing)
- Übertragbar auf weitere **Textquellen**

DAV/DGVFM
Herbsttagung
2025

Vielen Dank für Ihre Aufmerksamkeit!

Corinna Walk
corinna.walk@viadico.com

Yannick Richter
yannick.richter@viadico.com

viadico GmbH
Nürtinger Straße 11
70794 Filderstadt