

NAO Spracherkennung Studienprojekt

|  |  |
| --- | --- |
| **Name, Vorname:** | Anna Stabel, |

Caroline Sarah Schäfer, Sofie Wagner,   
**Semester:**  WiSe 2024/25   
**Fachbereich:**  Duales Studium (FB 2) **Studiengang:**  Duales Studium Informatik **Modul:**  Studienprojekt II   
**Betreuer Hochschule:**  Dagmar Monet Diaz   
**Anzahl der Wörter:**  0

NAO Studienprojekt

**Unterschriften**

|  |  |
| --- | --- |
| Ort, Datum | Anna Stabel |

|  |  |
| --- | --- |
| Ort, Datum | Caroline Sarah Schäfer |
| Ort, Datum | Sofie Wagner |
| Ort, Datum | Ausbilder\*in SAP |
| Ort, Datum | Ausbilder\*in HZB |

ii

NAO Studienprojekt

**Abstract**

Hier kommt das Abstract hin

|  |  |
| --- | --- |
| Abstract | iii |

NAO Studienprojekt

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Inhaltsverzeichnis**  **Abstract**  **Akronyme** | | | | | | | | **iii v**  **1**  **1**  **1**  1  1  2  2  2  **6**  6  6  6  7  7  7  7  7  **10** 10 10 **11 11** |
| **1** | **Einleitung** | | | | | | |
| **2** | **Problemstellung** | | | | | | |
| **3** | **IST Zustand des Programms** | | | | | | |
| 3.1 | | Programmbeschreibung | | | | . . . . . . . . . . . . . . . . . . . . . . . . . . . | |
| 3.1.1 | | | Datenbank und Schlüsselwörter | | | | . . . . . . . . . . . . . . . . . . |
| 3.1.2 | | | Satzanalyse | | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | |
| 3.1.3 | | | Antwortermittlung . . . . . . . . . . . . . . . . . . . . . . . . . . | | | | |
| 3.2 | | Leistungsfähigkeit . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | | | | |
| **4** | **NLP Suchalgorithmen** | | | | | | |
| 4.1 | | Knowledge Graphs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | | | | |
| 4.1.1 | | | Beschreibung | | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | |
| 4.1.2 | | | Implementierung . . . . . . . . . . . . . . . . . . . . . . . . . . . | | | | |
| 4.2 | | Ontologie . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | | | | |
| 4.2.1 | | | Beschreibung | | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | |
| 4.2.2 | | | Implementierung . . . . . . . . . . . . . . . . . . . . . . . . . . . | | | | |
| 4.3 | | Vektor Suche . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | | | | |
| 4.4 | | Latent semantic analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . | | | | | |
| **5** | **Fazit** | | | | | | |
| 5.1 | | Ergebnis | | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | | |
| 5.2 | | Ausblick | | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | | |
| **Bibliothek**  **Abbildungsverzeichnis** | | | | | | | |
| Inhaltsverzeichnis | | | | | | | | iv |

NAO Studienprojekt

**Akronyme**   
**KG** Knowlegde Graphs   
**HWR** Hochschule für Wirtschaft und Recht **OpenIE** Open Information Extraction   
**ML** Maschinelles Lernen   
**TSP** Traveling Salesman Probelem   
**LSA** Latent semantic analysis   
**SVD** Singular Value Decomposition   
**DB** Datenbank   
**POS-Tags** Part-of-Speech-Tags

|  |  |
| --- | --- |
| Inhaltsverzeichnis | v |

NAO Studienprojekt

|  |  |
| --- | --- |
| **1** | **Einleitung** |

Einleitung

**2 Problemstellung**

TODO

**3 IST Zustand des Programms**

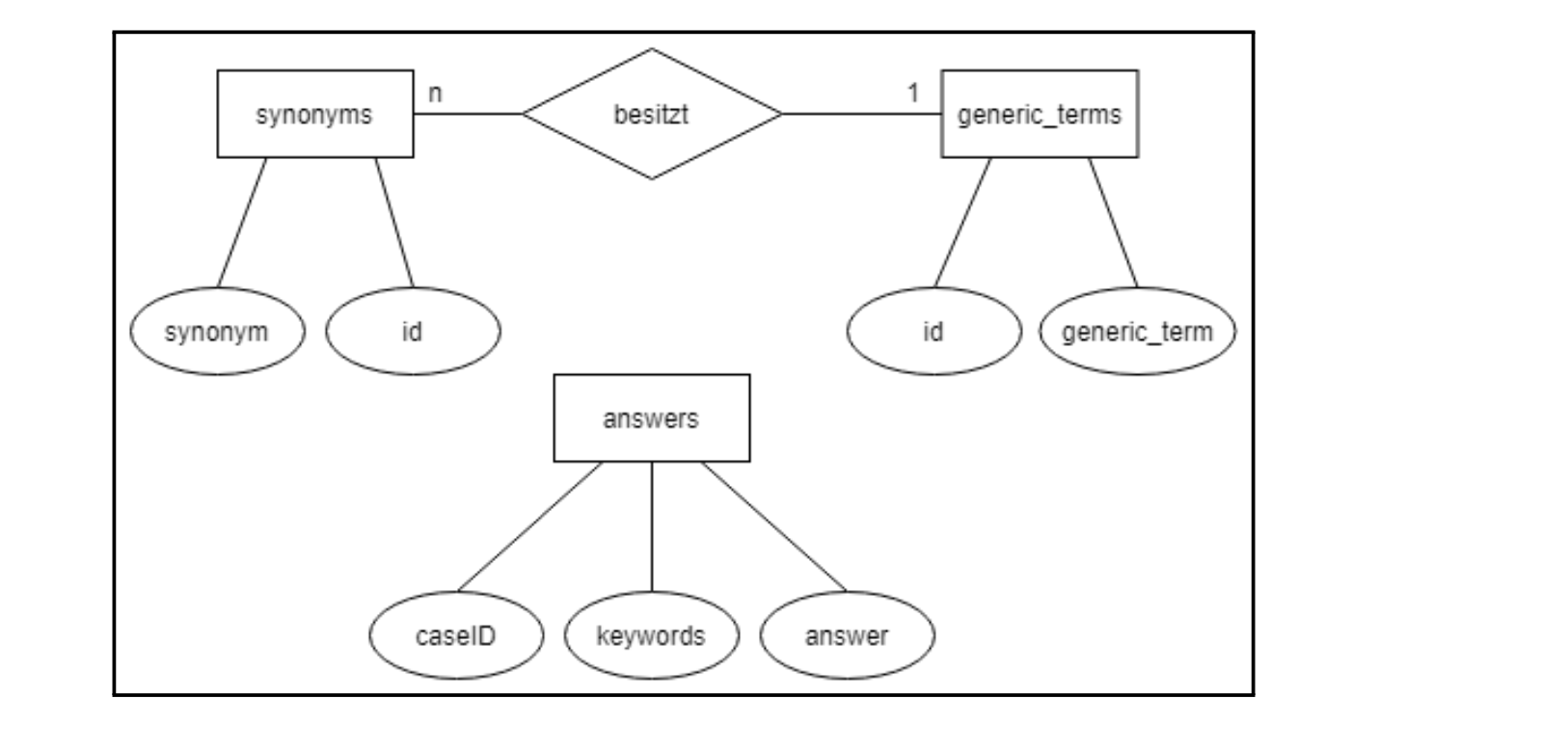
Das vorliegende Programm hat das Ziel, Antworten zu spezifischen Fragen aus Au-dioaufnahmen zu extrahieren. Im Folgenden wird der aktuelle Stand der Implemen-tierung beschrieben. Die wichtigsten schritte erfolgen innerhalb der get\_answer Methode.

**3.1**  **Programmbeschreibung**

**3.1.1 Datenbank und Schlüsselwörter**

Das System basiert auf drei Datenbank (DB): der Schlüsselwörter-Datenbank, der Synonyme-Datenbank und der Antworten-Datenbank. Die Schlüsselwörter (generic\_term) sind Begriffe, die mit hoher Wahrscheinlichkeit darauf hinweisen, dass eine bestimm-te Frage gestellt wurde. Ein Schlüsselwort kann mehrere Synonyme besitzen. Synony-me werden mithilfe von Oberbegriffen gruppiert, die durch SQL-Abfragen ermittelt werden.

Um eine effektive Zuordnung zu gewährleisten, sind Schlüsselwörter zusammen mit den zugehörigen Antworten in einer Tabelle organisiert. Jede Antwort wird durch eine eindeutige case\_id identifiziert, was eine schnelle und effiziente Suche ermög-licht.



**Abbildung 1: Datenbank-Diagramm**

1

NAO Studienprojekt

|  |  |
| --- | --- |
| **3.1.2** | **Satzanalyse** |

Die Verarbeitung der Audiodaten erfolgt durch eine Tokenisierung, bei der nicht signifikante Wörter (z. B. Artikel) aus dem Satz entfernt werden. Die verbleiben-den Wörter werden mit dem NLP-Paket spaCy analysiert, das Part-of-Speech-Tags (POS-Tags) verwendet, um die jeweilige Wortart zu identifizieren. Die Gewichtung der Schlüsselwörter basiert auf der Häufigkeit ihres Auftretens. Dies wird in der Me-thode distinct\_list umgesetzt:

Gewichtungswert = 1 *−*

Häufigkeit des Wortes   
Gesamtanzahl aller Schlüsselwörter

Dieses Verfahren ermöglicht eine priorisierte Analyse seltener, aber wahrscheinlich bedeutungsvoller Wörter. Seltener auftretende Wörter haben in der Regel eine höhere Bedeutung für die Zuordnung.

**3.1.3 Antwortermittlung**

Zur Bestimmung der passenden Antwort werden die analysierten Wörter mit der Da-tenbank abgeglichen. Die case\_id mit den meisten Treffern wird durch die Metho-de count\_ids ermittelt. Die Methode akzeptiert eine Liste von Wörtern als Eingabe und iteriert über diese. Mithilfe von db\_connector.get\_caseIDs\_by\_keywords(word) werden die zugehörigen case\_ids für jedes Schlüsselwort aus der Datenbank abge-rufen. Für jede gefundene case\_id wird die Methode check\_list aufgerufen, um die Gewichtung zu aktualisieren. Dabei wird das aktuelle Gewicht für jede case\_id gespeichert. Schließlich wird die case\_id mit dem höchsten Gewicht zurückgege-ben. Falls mehrere case\_ids mit identischer Gewichtung existieren, erfolgt eine zusätzliche Überprüfung anhand der primären Schlüsselwörter. Die finale Antwort wird mit der Methode get\_answer\_from\_db abgerufen, welche die Antwort für die ermittelte case\_id aus der Datenbank extrahiert und ausgibt.

**3.2**  **Leistungsfähigkeit**

Die momentane Performance des Algorithmus wird mit Hilfe der Python-Bibliothek *time* gemessen. Diese Bibliothek bietet Funktionen, die es Entwicklern ermöglichen, mit Zeiten zu arbeiten und verschiedene Zeitoperationen durchzuführen. Um die Zeit zu messen, die seit den Funktionsaufrufen vergangen ist, wird die Differenz zwi-schen der Endzeit und der Startzeit berechnet. Dazu wird der Timer mit der Funktion time\_nc() initialisiert. Diese Funktion gibt die Anzahl der Nanosekunden seit der In-itialisierung des Timers als Integer zurück [**pythonTimer**]. time\_nc() wurde gewählt,

2

NAO Studienprojekt

um potentiellen Präszisionsfehlern, die aufgrund einer Floating Nummer passieren

können, zu vermeiden. Die folgende Tabelle zeigt die Messergebnisse:

|  |  |  |  |
| --- | --- | --- | --- |
| **Frage** | **Antwort-** **Algorithmuszeit (ns)** | **Transkriptionszeit (ns)** | **Gesamtzeit (ns)** |
| Wie oft muss man einen PTB schreiben? | 33659000 | 2432785000 | 2474059000 |
| Welche Fachbereiche gibt es in der HWR? | 18816000 | 388357000 | 411529000 |
| Wie kann ich mich für ein duales Studium bewer-ben? | 14467000 | 360204000 | 379107000 |
| |  |  |  |  | | --- | --- | --- | --- | | Erzähl | mir | über | den |   Informatik-Studiengang. | 13036000 | 308227000 | 325151000 |
| |  |  |  |  | | --- | --- | --- | --- | | Wann | wurde | die | HWR |   Berlin gegründet? | 22216000 | 379835000 | 405591000 |
| Erzähl mir was über die HWR. | 17349000 | 316136000 | 337483000 |
| Welche Voraussetzungen gibt es für ein Informatik-Studium? | 12492000 | 405805000 | 422220000 |
| Was ist eine Studienarbeit? | 13245000 | 291517000 | 308290000 |
| Was ist ein Studiengang? | 12957000 | 305227000 | 320896000 |
| Was bedeutet PTB? | 14044000 | 307653000 | 323718000 |

**Durchschnittszeiten:**

• Antwort-Algorithmuszeit: 17228100 ns

• Transkriptionszeit: 549574600 ns

• Gesamtzeit: 570804400 ns

3

NAO Studienprojekt

Im folgenden wird die Antwort-Algorithmuszeit analysiert, um Verbesserungspoten-tial im Suchalgorithmus festzustellen. Die Funktionsabfolge wurde in **??** erläutert.

**Funktion db\_connector.get\_generic\_term:**

• Beschreibung: Für jedes relevante Wort wird eine Datenbankabfrage durchge-führt, um dessen generische Form zu finden. Dies ist der teuerste Schritt, da jede Abfrage Zeit beansprucht und die Abfragen in einer Schleife ausgeführt werden.

• Zeitaufwand: Dies hängt von der Anzahl der Wörter und der Geschwindigkeit der Datenbank ab, typischerweise im Bereich von Millisekunden.

• Laufzeitkomplexität: *O*(*n·m*), wobei *n* die Anzahl der Wörter und *m* die durch-schnittliche Zeit für eine einzelne Datenbankabfrage ist. Falls für jedes Wort eine Abfrage ausgeführt wird, summieren sich die Datenbankoperationen linear zur Anzahl der Wörter.

**Funktion caseID = counter.count\_ids:**

• Beschreibung: Dieser Schritt durchsucht die IDs basierend auf Gewichtungen der Wörter und sucht nach dem relevantesten caseID. Die Laufzeit hängt von der Implementierung von count\_ids und der Anzahl der Wörter ab.

• Zeitaufwand: Variiert von Millisekunden bis Sekunden, abhängig von der Da- tenbank und der Anzahl der IDs in der Datenbank.

• Laufzeitkomplexität: *O*(*n · k*), wobei *n* die Anzahl der relevanten Wörter und *k*  die Anzahl der verfügbaren caseIDs in der Datenbank ist.

**Funktion db\_connector.get\_answer\_from\_db:**

• Beschreibung: Die Funktion ruft die Antwort basierend auf einem einzelnen caseID ab. Da nur eine Datenbankabfrage erforderlich ist, ist die Laufzeit unab-hängig von der Eingabelänge.

• Zeitaufwand: Typischerweise Millisekunden, aufgrund nur einer Abfrage.

• Laufzeitkomplexität: *O*(1), da nur eine einzige Datenbankoperation ausgeführt wird, um die Antwort abzurufen.

4

NAO Studienprojekt

Die zeitintensivsten Vorgänge sind demnach die Abfrage der generischen Form des Wortes und die Gewichtung der relevantesten caseIDs. Das Ziel dieser Studienarbeit besteht folglich darin, die Laufzeit der zuvor beschriebenen Funktionen zu mindern bzw. den Suchalgorithmus so zu modifizieren, dass diese obsolet werden.

5

NAO Studienprojekt

|  |  |
| --- | --- |
| **4** | **NLP Suchalgorithmen** |

TODO: Intro für NLP Suchalgorithmen

**4.1**  **Knowledge Graphs**

**4.1.1 Beschreibung**

Knowlegde Graphs (KG) stellen eine strukturierte Darstellungsform von Informa-tionen dar, welche aus unstrukturierten Texten gewonnen werden. Sie setzen sich aus Informationsentätiten, welche Knoten genannt werden, und Beziehungen zwi-schen den Informationsentätiten, welche Kanten genannt werden, zusammen. Die-se werden aus Textdaten abgeleitet. Dadurch wird die Integration, der Abruf und die Analyse von Informationen erleichtert [**Hojas-Mazo2018A**]. Um einen KG aus einem Text zu konstruieren, werden verschiedene Methoden. Beispiele dafür sind Techniken wie Open Information Extraction (OpenIE), Maschinelles Lernen (ML) und semantische Analyse zum Einsatz [**OpenIEbased**]. Die strukturierte und semanti-sche Darstellungsform von Informationen, wie sie in Knowledge Graphen erfolgt, ermöglicht eine präzisere und effizientere Beschaffung von textbasierten Informatio-nen [**Dietz2017Utilizing**]. Dies wird durch folgende Faktoren begünstigt:

• Die verbesserte Textdarstellung ermöglicht die Rückgabe reichhaltiger seman- tischer Strukturen.

• Die automatische Strukturierung von Textinhalten wird durch KGs signifikant vereinfacht, da eine Kategorisierung von Textinformationen in kürzerer Zeit er-folgt [**Hojas-Mazo2018A**].

• Die Berechnung der semantischen Ähnlichkeit, welche eine Steigerung der Ef-fizienz und Genauigkeit von Suchergebnissen zum Ziel hat, kann mittels KGs ohne großen Aufwand durchgeführt werden [**Wang2018Information**].

• Die Integration von KGs in multimediale Modelle, wie beispielsweise Richpe-  
dia, ermöglicht die Nutzung zusätzlicher Ressourcen, beispielsweise visueller   
Art, für die semantische Suche sowie die Beantwortung von Fragen [**Wang2020Richpedia:**].

**4.1.2 Implementierung**

Die Implementierung von KGs erfolgte unter Zuhilfenahme der Python-Bibliothek NetworkX in Kombination mit spaCy, einer bereits zuvor im Code verwendeten Python-

6

NAO Studienprojekt

Bibliothek. Auf spaCy wurde bereits zuvor eingegangen. NetworkX ist ein Python-Paket, welches die Erstellung, Bearbeitung und Untersuchung der Struktur, Dynamik und Funktionen komplexer Netzwerke ermöglicht. Die Python-Bibliothek wurde als Werkzeug zur Umsetzung der vorliegenden Anforderungen gewählt. Das Paket er-möglicht es Entwicklern verschiedene Arten von Graphen (bspw. Diagraphen und Multigraphen) aus diversen Datenstrukturen wie Text oder XML zu erstellen oder zu generieren. Zudem können Operationen wie das Löschen von Knoten an Gra-phen durchgeführt, Graphen analysiert (beispielsweise die Anzahl der Knoten ge-zählt) oder Algorithmen wie z.B. zum Lösen des Traveling Salesman Probelem (TSP) implementiert werden [**networkX:Docs**].

**4.2**  **Ontologie**

**4.2.1 Beschreibung**

Ontologie im NLP beinhaltet die Verwendung eines strukturierten Rahmens zur Re-präsentation von Wissen innerhalb einer bestimmten Domäne und erleichtert Auf-gaben. Die Ontologie zeigt Eigenschaften und Beziehungen zwischen einer Reihe von Konzepten und Kategorien innerhalb der Domäne auf. Ein Beispiel für die An-wendung von Ontologie wäre, dass eine Maschine die Bedeutung des Wortes „Dia-mond“ in Bezug auf einen Baseballspieler, einen Juwelier oder eine Kartenfarbe ge-nau interpretieren kann. In NLP wird Ontologie zum Beispiel zur Wiederauffindung von Informationen, dem Beantworten von Fragen und dem Annotieren von Entitä-ten eingesetzt, da sie semantisch angereicherte Antworten in ihren Domänen liefern [**Adelkhah2019The**] [**Naderian2018Ontology**].

**4.2.2 Implementierung**

**4.3**  **Vektor Suche**

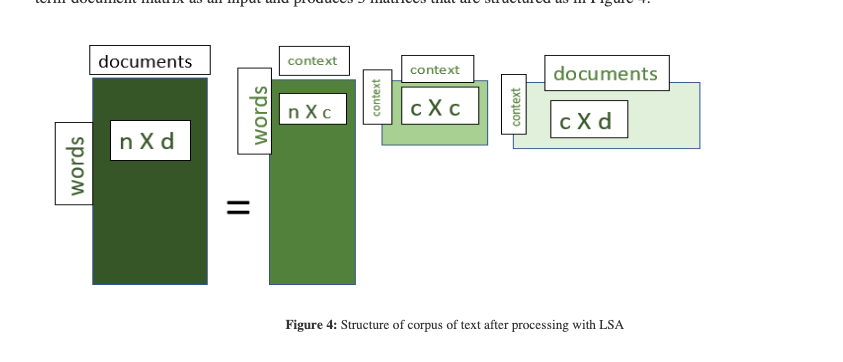
**4.4**  **Latent semantic analysis**

Latent semantic analysis (LSA) ist eine statistische Methode zur Schätzung der Wort-bedeutungen. Diese Bedeutung basiert auf zugrunde liegenden Konzepten. Diese Konzepte werden durch Matrixoperationen extrahiert, die auf beobachteten Mustern der Wortverwendung basieren. Die grundlegende Idee hinter LSA ist, dass die Be-deutung jedes Textabschnitts als Summe der Bedeutungen der darin enthaltenen Ein-zelwörter ist, während eine Sammlung von Dokumenten (ein „Korpus“) als ein Ge-

7

NAO Studienprojekt

leichhungssystem dargestellt wird, welches die Ähnlichkeit von Wörtern zu anderen Wörtern und Dokumenten zu anderen Dokumenten zueinander bestimmen kann. LSA stellt die Beziehung zwischen Dokumenten und Begriffen durch eine Term-Dokument-Matrix dar, die durch Singular Value Decomposition (SVD) weiter in das Produkt von drei Matrizen zerlegt wird. SVD ist das mathematische Werkzeug hin-ter LSA [**TheUseofLatentSemanticAnalysis**]. Es ist eine grundlegende Matrixfak-torisierungstechnik in der linearen Algebra mit vielseitigen Anwendungsbereichen [**Paige1981Towards**]. Es zerlegt eine Matrix A in drei Matrizen (siehe Bild). Für eine



gegebene Abfrage transformiert LSA diese in einen Pseudo-Dokumentenvektor und   
berechnet die Ähnlichkeiten mithilfe des SVD-Ergebnis aus der Term-Dokument-  
Matrix zwischen der Abfrage und dem durchsuchten Dokumenten [**SystematicReviewofSemanticA** . Im Gegensatz zu präzisen Abgleichmethoden wird die Matrix durch SVD zerlegt,   
was sie in einen neuen Raum mit niedriger Dimension komprimiert.SVD kann nicht   
nur die Datenmenge reduzieren, sondern auch die zugrunde liegenden Beziehungen   
zwischen Begriffen erkennen. Aus den oben genannten Gründen, gilt LSA als eine   
sehr flexible Technik ist und wird oft in der Sprachsuche benutzt wird [**TextMiningUsingLatentSem** .

1. **Fett 1** Nummerierte liste 1

2. *Kursiv 1* Nummerierte liste 1

3. Nummerierte liste 1

Eine Matheformel (Satz des Pythagoras):

*a*2+ *b*2= *c*2

wobei *a* und *b* die Längen der Katheten eines rechtwinkligen Dreiecks sind, und *c* die Länge der Hypotenuse.

|  |  |
| --- | --- |
| 1 | Python Code Section 1 |

**Listing 1: Pip Update**

8

NAO Studienprojekt

”’Deutsche Anführungszeichen”’ **??** referenz Zu Label1 „Anführungszeichen unten,

Anführungszeichen oben“

**??** Bild Referenz

|  |  |
| --- | --- |
| **Column1** | **Column2** |
| row1 | row2 |

**Tabelle 2: Your table caption here**

9

NAO Studienprojekt

|  |  |
| --- | --- |
| **5** | **Fazit** |

Fazit   
**5.1**  **Ergebnis**   
Ergebnis   
**5.2**  **Ausblick**   
Ausblick

10

NAO Studienprojekt

|  |  |
| --- | --- |
| **Abbildungsverzeichnis**  Abbildung 1: Datenbank-Diagramm . . . . . . . . . . . . . . . . . . . . . . . . . | 1 |
| Abbildungsverzeichnis | 11 |

**AI-Verzeichnis**

• ChatGPT 4o und o1-preview  
• Consensus AI  
• Perplexity  
• DeepL

**Ehrenwörtliche Erklärung**

Wir erklären hiermit ehrenwörtlich:

1. dass wir unsere Studienarbeit selbstständig verfasst habe,

2. dass wir die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwen-dung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe,

3. dass wir unsere Studienarbeit bei keiner anderen Prüfung vorgelegt habe.

Wir sind uns bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

|  |  |
| --- | --- |
| Ort, Datum | Anna Stabel |

|  |  |
| --- | --- |
| Ort, Datum | Caroline Sarah Schäfer |
| Ort, Datum | Sofie Wagner |