Patrick Masny
7150991
Informatik (BA)
Patrick.Masny@hotmail.com

Bachelorarbeit

ARES: Annotation von Relationen und Eigenschaften zur Szenengenerierung

Patrick Masny

Abgabedatum: 26.08.2022

Institut für Informatik
Text Technology Lab
Prof. Dr. Alexander Mehler

Zusammenfassung

Im Fachbereich der Computerlinguistik ist die automatische Generierung von Szenen aus, in natürlicher Sprache verfassten, Text seit bereits vielen Jahrzehnten ein wichtiger Bestandteil der Forschung, welche in der „Kunst“, „Lehre“ und „Robotik“ Verwendung finden (vgl. Chang, Monroe u. a. 2015, S. 1). Mit Hilfe von neuen Technologien im Bereich der Künstlichen Intelligenzen (KI), werden neue Entwicklungen möglich, welche diese Generierungen vereinfachen, allerdings auch undurchsichtige interne vom Modell getroffene Entscheidungen fördern.
Ziel der vorgeschlagenen Lösung „ARES: Annotation von Relationen und Eigenschaften zur Szenengenerierung“ ist es, ein modulares System zu entwerfen, wobei einzelne Prozesse für den Benutzer verständlich bleiben. Außerdem sollen Möglichkeiten geboten werden, neue Entitäten und Relationen, welche über die Textanalyse bereitgestellt werden, auch in die Szenengenerierung im dreidimensionalen Raum einzupflegen, ohne dass hierfür Code zwingend notwendig wird.
Der Fokus liegt auf der syntaktisch korrekten Darstellung der Elemente im Raum. Dagegen lässt sich die semantische Korrektheit durch weitere manuelle Anpassungen, welche für spätere Generierungen gespeichert werden erhöhen. Letztlich soll die Menge der zur Darstellung benötigten Annotationen möglichst gering bleiben und neue szenenbezogene Annotationen durch die implementierten Annotationstools hinzugefügt werden.

Danksagung

Ich möchte mich an dieser Stelle bei denjenigen bedanken, die mich bei der Erstellung der Bachelorarbeit unterstützt haben.
Zuerst gebührt mein Dank Prof. Dr. Alexander Mehler, der durch seine spannenden Module mein Interesse für den NLP Bereich geweckt hat. Ebenso gilt mein Dank an Alexander Henlein und Giuseppe Abrami mit deren Hilfe ich meine Kenntnisse vielfältig erweitern konnte. Für die hilfreichen Anregungen und und Unterstützungen bei der Erstellung dieser Arbeit möchte ich mich bei allen genannten herzlich bedanken.
Außerdem bedanke ich mich bei Soufian Noor, Yannic Blecher und Dawit Terefe, die mir beim Korrekturlesen dieser Arbeit geholfen und mich darüber hinaus über das Informatik Studium hinweg stets motiviert haben. Genauso möchte ich mich bei allen Kommilitonen bedanken, die an der Umfrage zu diesem Projekt teilgenommen haben.

Inhaltsverzeichnis

1 Einleitung **9**
- 1.1 Motivation . . . 9
- 1.2 Ähnliche Arbeiten . . . 9

2 Verwendete Ressourcen **11**
- 2.1 StolperwegeVR . . . 11
- 2.2 ShapeNet . . . 11
- 2.3 Language-Driven Synthesis of 3D Scenes from Scene Databases . . . 12
- 2.4 SpaCy . . . 12

3 ARES **13**
- 3.1 Architektur . . . 13
- 3.2 Anwendungsfälle . . . 13

4 Textanalyse **14**
- 4.1 Entitäten-Erkennung . . . 14
- 4.2 Relationen-Erkennung . . . 15
- 4.3 Attribut-Erkennung . . . 16
- 4.4 Schnittstelle . . . 16

5 Modelldatenbank **18**
- 5.1 Vorhandene Annotationen . . . 18
- 5.2 Schnittstelle . . . 19

6 3D-Umgebung **20**
- 6.1 Annotation von Relationen und Eigenschaften . . . 20
  - 6.1.1 Eigenschaften definieren . . . 20
  - 6.1.2 Relationen definieren . . . 22
- 6.2 Szenengenerierung . . . 24
  - 6.2.1 Textanalyse . . . 24
  - 6.2.2 Szenengenerierung durch Drag and Drop . . . 25
  - 6.2.3 Logik: Anwendung der Eigenschaften und Relationen Annotationen 26

7 Evaluation **29**
- 7.1 Textanalyse . . . 29
  - 7.1.1 Aufbau . . . 29
  - 7.1.2 Ergebnisse . . . 29
- 7.2 Szenengenerierung . . . 30
  - 7.2.1 Aufbau . . . 30

7.2.2 Ergebnisse 30

8 Anregungen für weiterführende Arbeiten **33**
8.1 Verbesserung der automatischen Modell Wahl 33
8.2 Erweiterung der Szenengenerierung um den Ort 33
8.3 Hyperparameteranalyse für die Positionierungslogik 33
8.4 Integration neuer Textanalyse Modelle 34

9 Fazit **35**

10 Anhang **36**

Literatur **46**

Abbildungsverzeichnis

4.1 Finden von Zahlen Nennungen für eine gefundene Entität 15
4.2 Finden von Relationen wie beispielsweise around 15
4.3 Finden von Relationen wie beispielsweise next to 15
4.4 Finden von Relationen wie beispielsweise on top of 16
4.5 Finden von Relationen wie beispielsweise surounded by 16
4.6 JSON Struktur für die Rückgabe des analysierten Textes 17

5.1 JSON Struktur der Annotationsrückgabe 18

6.1 Benutzeroberfläche für die Annotation von Eigenschaften 21
6.2 Versuch zur blauen Färbung eines Bettes (Original in der Mitte), wobei das linke Bett proportional mit einer Stärke von 40% und das rechte vollständig blau gefärbt wurde. 22
6.3 Benutzeroberfläche für die Annotation von Relationen. In diesem Fall wird die Relation „on“ mit Hilfe von 2 aufeinander gestapelten Kuben dargestellt, wobei der rote Kubus für den „Head“ und der blaue für den „Tail“ der Relation steht. 23
6.4 Benutzeroberfläche für das Eingeben und Analysieren von Texten, wie auch der Ausgabe der darin enthaltenen Informationen. In diesem Fall wurde als Text „apple on table“ eingegeben, welcher bereits analysiert und als JSON zurückgegeben wurde. 24
6.5 Grundlegende Benutzeroberfläche des Scene-Builders. Im linken Bild sieht man den Menüpunkt Ressourcen, bei dem sich neue Entitäten in die Szene ziehen lassen, wohingegen im rechten die einzelnen, in der Szene vorhandenen, Entitäten und Relationen aufgelistet werden. 25
6.6 In der Mitte ist die Benutzeroberfläche zu sehen, welche für das Bearbeiten und Erstellen von Entitäten verwendet wird. 26
6.7 Mögliche syntaktisch korrekten Anordnungen der Szene „four chairs next to a table“. 27

7.1 Ergebnisse zur ARES Umfrage (vgl. Abb. 10.8). 31

10.1 Sequenzdiagramm für die Funktionallitäten der ARES-Architektur 36
10.2 Klassendiagramm samt Paketstruktur und Dateistruktur für die Textanalyse 37
10.3 Annotierte Szene mit Beschreibung der Forscher (oben) und zusätzlichen Annotationen von AMT (unten) (Chang, Monroe u. a. 2015, S. 3) 38
10.4 JSON Struktur für das Abspeichern der definierten Eigenschaften 38
10.5 JSON Struktur für das Abspeichern der definierten Relationen 39
10.6 Benutzeroberfläche für das Verwalten der Relationen 40

10.7 Beispielhafte Szenengenerierung der ARES Implementierung für die Szene „40 phones on the table“ 40
10.8 Zur Evaluierung von ARES verwendeter Umfragebogen 41

Tabellenverzeichnis

2.1 Übersicht des ShapeNet Datensatzes mit seinen Teildatensätzen ShapeNetCore und ShapeNetSem (ergänzt durch d. Verf. Chang, Funkhouser u. a. 2015, S. 7). 11

7.1 Auswertung des für ARES implementierten Textanalyse Modells. 30
7.2 Auswertung (Angabe in Sekunden) der in ARES implementierten Annotationstools (vgl. Abb. 10.8). 32

1 Einleitung

1.1 Motivation

Die meisten modernen Lösungen, welche sich mit der dreidimensionalen Szenengenerierung aus Texteingaben befassen, benötigen riesige annotierte Datenmengen, um Texte auf die vorhandenen Entitäten und Relationen zu überprüfen und Modelle zu positionieren. Ebenso sind die bei der Szenengenerierung getroffenen Entscheidungen oftmals für den Benutzer nicht mehr ersichtlich und nur unter Aufwand zu identifizieren. Daten wie beispielsweise zur Positionierung von Objekten liegen häufig in Textform vor. Dadurch muss sich der Benutzer die dazugehörige Szene zunächst abstrakt vorstellen, um die dahinterliegende Logik zu verstehen.
Die ARES-Architektur wurde zur Lösung dieser Problematiken entworfen, um dem Nutzer ein modulares Gesamtsystem zu bieten, wobei einzelne Prozesse immer noch verständlich bleiben. Bei der Nutzung des Systems erhält der Nutzer den Vorteil, einzelne Relationen wie auch Eigenschaften beliebig zu definieren und bei Bedarf nachzujustieren. Demzufolge wird der Benutzer selbst in das Schaffen der Logik integriert und benötigt aufgrund der Annotationstools (vgl. Kap. 6.1.1, 6.1.2 und 6.2.2) keinen weiteren Code. Ebenfalls kann die Logik der Positionierung durch den Debugger Modus verständlich gemacht werden, indem die Positionierung einsehbar Schritt für Schritt abgearbeitet wird.

1.2 Ähnliche Arbeiten

Die Kunst, Texteingaben zu visualisieren, wird seit Jahrzehnten erforscht, wobei einige der ältesten Arbeiten bis zu einem halben Jahrhundert zurückreichen, wie zum Beispiel die Arbeiten von Winograd 1972, Simmons 1975 oder Kahn 1979. Damals wurden die Möglichkeiten komplexere Grammatiken zu analysieren und realistische Strukturen darzustellen durch die damalige Hardware limitiert. Dennoch sind diese ersten Versuche wertvoll für die weitere Forschung gewesen und mit der Entwicklung von Computerhardware, wurde den Forschenden das Arbeiten mit besseren Werkzeugen ermöglicht.
Neue Technologien und Konzepte im Grafikbereich und im Gebiet der Künstlichen Intelligenz sorgten einerseits dafür, dass realistischere Szenen dargestellt werden können und andererseits dafür, dass komplexeren Textvisualisierungsanwendungen durch eine Verbesserung von Natural Language Processing (NLP) Analysen ermöglicht wurden. Infolgedessen wurden weitere Implementierungen für den zwei- wie auch dreidimensionalen Raum entwickelt. Für den zweidimensionalen Raum wurden vor allem Generative Adversarial Networks (GANs) (I. Goodfellow u. a. 2014) zu einem wichtigen Bestandteil, da ihre effektive Architektur für das unüberwachte Lernen, die realistische Bildgenerierung mit Hilfe eines generativen und eines diskriminativen Modells ermöglicht. Implementierungen hierzu gab es beispielsweise von H. Zhang u. a. 2017, Reed u. a. 2016 oder Z. Zhang, Xie und Yang 2018,

wobei komplexere Modelle, wie im Text2Scene Projekt (Tan, Feng und Ordonez 2018) auch im zweidimensionalen Bereich möglich sind.
Innerhalb dieser Arbeit wird an einer dreidimensionalen Lösung gearbeitet, welche hingegen durch die hinzukommende Dimension, im Vergleich zu Bildern, eine erhöhte Komplexität bereitstellt. Das Text to 3D Scene Modell von Chang, Monroe u. a. 2015 hat beispielsweise den Ansatz anhand von annotierten Szenen, die Szenengenerierung zu erlernen. Hier kommt ein Modell zum Einsatz, welches für die Klassifizierung der Entitäten und Relationen, wie auch der Erlernung der Positionierung, zuständig ist. Während ein weiteres Modell dafür zuständig ist Entitäten und Relationen aus Texten herauszulesen und anschließend regelbasiert darzustellen. Die ARES-Architektur verwendet dabei einen ähnlichen Ansatz bei dem Erlernen von Relationen, wobei Relationen anhand von abstrakten, vom Benutzer erstellten, Annotationen umgesetzt werden. Lösungen wie in dem Text2Shape Modell (Chen u. a. 2018), welche durch die Beschreibung einer Entität, ein zugehöriges Modell erstellen, wurden für die jetzige ARES Implementierung nicht mitberücksichtigt und erfolgen in der jetzigen Implementierung gegebenenfalls mit Hilfe der gefundenen Attribute (vgl. Kap. 6.1.1 und 6.2).

2 Verwendete Ressourcen

Die hier vorgestellte Implementierung bedient sich der Schnittstelle zum Laden der 3D-Modelle aus der Texttechnologylab Datenbank der StolperwegeVR Anwendung. Diese verfügt über eine Vielzahl von Modellen aus dem ShapeNetSem Datensatz. Ebenfalls wurde für die Logik der Textanalyse, die C++ Logik des „Language-Driven Synthesis of 3D Scenes from Scene Databases“ (Ma u. a. 2018) Projektes in Python übersetzt. Diese werden im Folgenden vorgestellt und näher beschrieben.

2.1 StolperwegeVR

Das StolperwegeVR Projekt (*StolperwegeVR* 2022) bietet Möglichkeiten, modellierte Gebäude auf einer Karte unter Verwendung der jeweiligen Standortdaten darzustellen und per Virtual Reality (VR) Option zu begehen. Für die ARES Implementierung wurde lediglich die Schnittstelle zu den ShapeNetSem Modellen benötigt. Diese wurde im Laufe der Implementierung ebenfalls erweitert.

2.2 ShapeNet

Bei ShapeNet (Chang, Funkhouser u. a. 2015) handelt es sich um einen Datensatz mit annotierten 3D-Modellen aus einer Vielzahl von Objektkategorien. Wie in Tabelle 2.1 zu sehen ist, gibt es vom Hauptdatensatz (ShapeNet) zwei Unterdatensätze (ShapeNetCore und ShapeNetSem), welche die aufgelisteten Vorteile mit sich bringen. Obwohl ShapeNetSem mit einer Anzahl an 270 verschiedenen Objektkategorien gegenüber der Stammbibliothek mit 3.135 eine kleinere Auswahl an darzustellenden Entitäten bietet, handelt es sich dabei um eine hochqualitative Basis für das Arbeiten mit Modellen in einer 3D-Umgebung, da die Annotationen nicht nur händisch verifiziert wurden, sondern auch realitätsbezogene Annotationen, wie das Gewicht eines Modells hinterlegt sind. Deshalb werden für die ARES Implementierung die Modelle aus dem ShapeNetSem Datensatz verwendet.

Tabelle 2.1: Übersicht des ShapeNet Datensatzes mit seinen Teildatensätzen ShapeNetCore und ShapeNetSem (ergänzt durch d. Verf. Chang, Funkhouser u. a. 2015, S. 7).

	ShapeNet	ShapeNetCore	ShapeNetSem
#Modelle	3.000.000	51.300	12.000
#Objektkategorien	3.135	55	270
manuell verifizierte Kategorien	✗	✓	✓
manuell verifizierte Ausrichtung	✗	✓	✓
Realitätsbezogene Annotationen	✗	✗	✓

2.3 Language-Driven Synthesis of 3D Scenes from Scene Databases

Die Arbeit „Language-Driven Synthesis of 3D Scenes from Scene Databases“ (Ma u. a. 2018) stellt eine Möglichkeit vor 3D-Szenen in Innenräumen zu erzeugen wie auch diese darin zu bearbeiten. Diese Arbeit dient als Vorlage für die Textverarbeitung auf der Backend Seite für ARES. Hierbei wurden die grundlegenden sprachlichen Eigenschaften, welche von den Forschenden analysiert und für die sprachliche Verarbeitung angewendet wurden, innerhalb von Python für eine vereinfachte Verwendung umgesetzt. Die implementierte Logik wird außerdem in Kap. 4 weiter erläutert und auf die für Python spezifischen Anpassungen näher eingegangen.

2.4 SpaCy

SpaCy (*SpaCy Homepage* 2022) bietet eine Vielzahl an Möglichkeiten Texte zu analysieren. Aus diesem wurden unter anderem part-of-speech tagging, dependency parsing, numerisation und lemmatization (vgl. *SpaCy Homepage* 2022) benötigt. Diese wurden als Basis verwendet, um die Texte mit Hilfe der Logik der in Kap. 2.3 vorgestellten Arbeit umzusetzen. SpaCy ersetzt dabei das in der Arbeit verwendete Stanford CoreNLP Framework (vgl. Ma u. a. 2018, S. 7).

3 ARES

ARES, kurz für Annotation von Relationen und Eigenschaften zur Szenengenerierung, ist eine Architektur, die es ermöglicht, Annotationen zu tätigen und diese direkt in die Szenengenerierung zu überführen. Im Folgenden werden die allgemeine Architektur und die Anwendungsfälle näher erläutert.

3.1 Architektur

Wie in Abb. 10.1 zu sehen ist, besteht das ARES System aus diesen drei grundlegenden Modulen: der 3D-Umgebung (s.a. Kap. 6), der Textanalyse (s.a. Kap. 4) und der Modelldatenbank (s.a. Kap. 5). Diese Aufteilung wurde vorgenommen, um die Verbesserung und Erweiterung der einzelnen Module separiert zu ermöglichen. Dadurch können unabhängig, aber auch im Gesamtsystem bessere Ergebnisse erzielt werden. Ein weiterer Vorteil ist die Übersichtlichkeit der einzelnen internen Prozesse, wodurch die Logik insgesamt nachvollziehbarer wird. Letztlich wird dadurch die Austauschbarkeit und Wiederverwendbarkeit der einzelnen Komponenten ermöglicht, wobei sich die Kapselung auch in der 3D-Umgebung wiederfindet, in welcher die einzelnen Annotationstools ebenfalls separiert voneinander funktionieren.

3.2 Anwendungsfälle

Die Implementierung von ARES kann dazu genutzt werden, um bei der Erforschung von syntaktischen Zusammenhängen innerhalb von Texten zu helfen. Hierbei kann untersucht werden, inwiefern sich bestimmte Relationen in einer dreidimensionalen Umgebung definieren lassen, ohne dass der eigentliche Sinn des Textes verloren geht.
Ebenfalls lässt sich das Tool in der Lehre verwenden. Wie einst Albert Einstein im Interview mit George Sylvester Viereck schon sagte „Fantasie ist wichtiger als Wissen, denn Wissen ist begrenzt“(*What Life Means To Einstein* 1929). Daher könnte die Anwendung auch von Kindern verwendet werden, um ihre Fantasie anzuregen und um reelle räumliche Konzepte zu erlernen, ohne das die dafür nötigen Objekte wirklich vorhanden sein müssen. Dadurch können Kinder spielerisch neue Objekte und Begriffe erlernen und sich einprägen.
Da Annotationen schwierig umzusetzen sind, können die hierin vorhandenen Tools verwendet werden, um speziell für ARES weitere Annotationen zu entwickeln. Ebenfalls können durch die Modularität, einzelne Komponenten wie zum Beispiel die Tools zur Annotation, in das eigene Projekt ohne großen Mehraufwand integriert werden.

4 Textanalyse

Die Textanalyse bildet das Fundament für die Funktionalität von ARES. Ein eingegebener Text ist ohne Weiteres für einen Computer unverständlich. Daher ist es in diesem Zusammenhang notwendig, der Maschine einen Satz verständlich zu machen, um die darin enthaltenen Informationen weiterverarbeiten zu können. Die notwendige Logik wurde im Zusammenhang mit den in Kap. 2.3 und 2.4 vorgestellten Mitteln implementiert und wird im Folgenden näher vorgestellt.

4.1 Entitäten-Erkennung

Wie in Abb. 4.6 zu sehen ist, wird für Entitäten unter anderem der Index des im Text vorkommenden Tokens, das Stammwort und die Menge der auftretenden Entität gespeichert. Zum Ermitteln der Entitäten wird der Part-Of-Speech (POS) Tagger von SpaCy (s. Kap. 2.4) verwendet. Damit wird für jedes Token die Wortart zugeordnet. Die dabei entstehenden Annotationen werden gefiltert, indem für jedes Token geprüft wird, ob das dazugehörige POS mit NN (womit die POS NN, NNP, NNPS sowie NNS zurückgegeben werden) (vgl. *SpaCy POS Tagging Glossar* 2022) beginnt, um die Nomen und damit auch die Objekte des Textes zu erhalten. Die daraus resultierende Liste wird anschließend bereinigt, indem zusammengesetzte Wörter (compounds) und bekannte Substantive wie zum Beispiel „wall“ oder „room“, welche sich für die jetzige Implementierung noch nicht eignen oder falsch erkannt werdende Entitäten wie „leave“, „right“, „side“ oder „front“, die zur Szenengenerierung nicht verwendet werden können, entfernt werden.
Sind also nun die die entsprechenden als Entitäten erkannten Tokens gefunden, so erhält man bereits die Indizes für die Rückgabe. Die gefundenen Tokens werden dann mit dem Lemmatizer von SpaCy analysiert und das entsprechende Lemma, also das Stammwort zurückgegeben. Dadurch wird vermieden, dass in der Suche nach einem passenden Modell, nicht das eigentliche Token verwendet wird und dadurch nicht alle Abwandlungen eines Wortes in die Annotation der 3D-Modelle hinzugefügt werden müssen.
Es soll auch möglich sein, die Anzahl einer Entität für die Szenengenerierung mit zu berücksichtigen. Hierzu wurden verschiedene Möglichkeiten zur Ermittlung integriert. Eine davon ist in Abb. 4.1 dargestellt, in welcher die Anzahl der Äpfel mittels der nummod-Dependency ermittelt wird. Der Numerizer kann die Textdarstellung von Zahlen in die eigentliche Zahl umwandeln, das heißt aus dem Wort „Two“ wird die Zahl „2“ umgewandelt. Im Gegensatz zu dem Text kann eine Maschine mit numerischen Werten arbeiten.

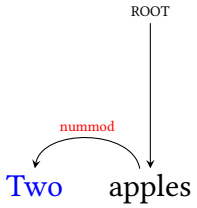

Abbildung 4.1: Finden von Zahlen Nennungen für eine gefundene Entität

Eine weitere Möglichkeit ist die Überprüfung über die Existenz eines determiners oder Attributs, konkret bedeutet das, dass überprüft wird, ob es zur Entität zugehörige Wörter gibt, die eine Anzahl beschreiben. Beispiele für diesen Fall wären das Wort „the“, welches für eine Entität steht oder das Wort „some“ bei welchem sich die Anzahl der Entitäten zufällig zwischen zwei und vier belaufen. Begründet ist diese Zufälligkeit dadurch, dass es für diesen Fall keinen einheitlichen Konsens gibt, um wie viele Objekte es sich handeln muss. All diese Funktionen tragen zu einer zuverlässigen Ermittlung der Entitäten bei.

4.2 Relationen-Erkennung

Die Relationen-Erkennung baut auf den erkannten Entitäten auf und ist stark von dem Dependency Tagger abhängig, wie es auch in Abb. 4.2 bis 4.5 zu sehen ist. Im Gegensatz zu der Darstellung in Abb. 4.1, treten komplexere Beziehungen auf. Diese wurden anhand der Logik aus der in Kap. 2.3 vorgestellten Arbeit implementiert (vgl. Ma u. a. 2018).

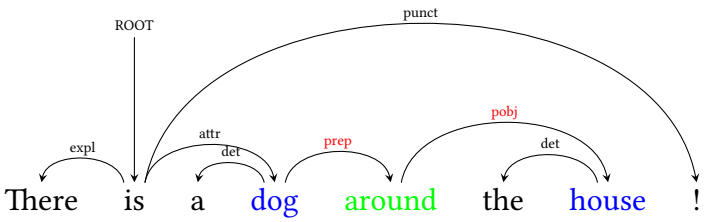

Abbildung 4.2: Finden von Relationen wie beispielsweise around

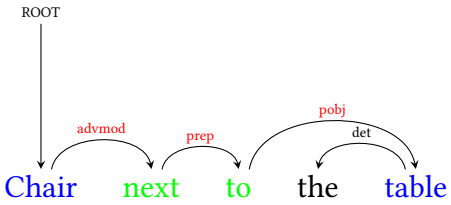

Abbildung 4.3: Finden von Relationen wie beispielsweise next to

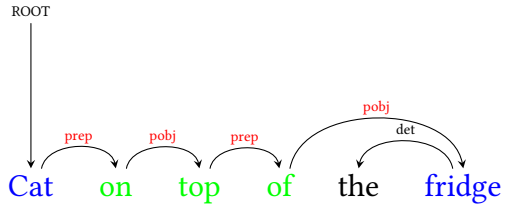

Abbildung 4.4: Finden von Relationen wie beispielsweise on top of

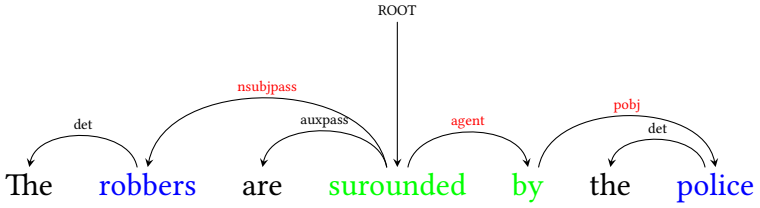

Abbildung 4.5: Finden von Relationen wie beispielsweise surounded by

4.3 Attribut-Erkennung

Die Analyse der Attribute ist im Vergleich zu den Entitäten und Relationen die Komplexeste, da sie über die meisten, in Texten auftretenden, Variationen verfügt. Die gefundenen Attribute spielen innerhalb der derzeitigen Implementierung noch eine untergeordnete Rolle, da für das zugehörige Annotationstool nicht alle Annotationsmöglichkeiten und Logiken in Verbindung mit Unity implementiert wurden. Das kann in späteren Versionen erweitert werden.

4.4 Schnittstelle

Für die Kommunikation zwischen der 3D-Umgebung und der Textanalyse wurde eine Representational State Transfer (REST) Schnittstelle mit Hilfe von Flask (*Flask Dokumentation* 2022) implementiert. Darüber lassen sich POST Anfragen tätigen. Dabei ist zu beachten, dass über den Request Body als Parameter für „inputText“ der Text übergeben werden muss, um daraufhin als Response-Message, die in dem Satz enthaltenen Informationen zu erhalten. Dieser Aufbau ist in Abb. 4.6 beschrieben.
Es ist von Vorteil den Zugriff auf das Application Programming Interface (API) überwachen zu können und für entstehende Ergebnisse eine nachträgliche Überprüfbarkeit zu erreichen, um diese für Verbesserungsanpassungen zu nutzen. Für diese Fälle wurde ein Logger implementiert, welcher jede an die Schnittstelle gesendete Transaktion und bei auftretenden Fehlern ihre Ursache speichert. Darüber hinaus wird in einem fehlerfreien Durchlauf die Antwort für den übergebenen Text in eine Logdatei, der Anwendungsinstanz, als JavaScript Object Notation (JSON) (s. Abb. 4.6) gespeichert.

```
{
    "result" :      [
                        {
                            "tokens" :      [string]
                            "entities" :    [
                                                {
                                                    "index" : int
                                                    "lemma" : string
                                                    "amount" : int
                                                }
                                            ]
                            "relations" :   [
                                                {
                                                    "head" : int
                                                    "tail" : int
                                                    "type" : string
                                                }
                                            ]
                            "attributes" :  [
                                                {
                                                    "index" : int
                                                    "type" : string
                                                }
                                            ]
                        }
                    ]
}
```

Abbildung 4.6: JSON Struktur für die Rückgabe des analysierten Textes

5 Modelldatenbank

5.1 Vorhandene Annotationen

Im Zuge der Implementierung wurden vor allem Modell Annotationen (s. Abb. 5.1) zum korrekten Ausrichten im Raum verwendet. Um physikalische Eigenschaften innerhalb von ARES darstellen zu können, wurde das Gewicht mitberücksichtigt. Das kann dazu verwendet werden, um bei Aktivierung der Schwerkraft, ein zwischen den Modellen unterschiedliches und somit auch realitätsnäheres Verhalten darzustellen.

```
{
    "success": bool
    "ShapeNetObj" : [
                        {
                            "wnsynset" : string
                            "wnlemmas" : [string]
                            "solidVolume" : string
                            "isContainer" : bool
                            "surfaceVolume" : string
                            "weight" : string
                            "tags" : [string]
                            "unit" : string
                            "supportSurfaceArea" : string
                            "staticFrictionForce" : string
                            "name" : string
                            "alignedDims" : string
                            "id" : string
                            "categories" : [string]
                            "up" : string
                            "front" : string
                            "has_parts" : bool
                        }
                    ]
}
```

Abbildung 5.1: JSON Struktur der Annotationsrückgabe

5.2 Schnittstelle

Um den benötigten Speicher für die ARES Implementierung zu reduzieren, werden nicht alle Modelle von ShapeNetSem statisch hinterlegt. Stattdessen wird auf den ShapeNetSem Datensatz mit Hilfe einer von der Texttechnologylab gehosteten Datenbank zugegriffen. Zuerst wird über die REST-API auf die Annotationen (vgl. Kap. 5.1) zugegriffen, welche die wie in der Abb. 5.1 zu sehenden Informationen als JSON Rückgabe ermöglicht.[1] Anschließend kann anhand der darin beinhalteten Informationen entschieden werden, ob für eine Entität ein geeignetes Modell vorliegt. Wird das gewollte Modell gefunden, so lässt es sich mit Hilfe der hinterlegten ID (s. Abb. 5.1) herunterladen[2], damit es in der 3D-Umgebung instanziieren werden kann. Für die in Kap. 6.2.2 vorgestellte Funktionalität bietet es sich an beim händischen annotieren der Szene, das für die Visualisierung verwendete Objekt zu sehen. Folglich werden für diesen Anwendungsfall, die Modell Thumbnails über die API[3] bezogen und dargestellt.

[1] Über folgenden Link verfügbar: http://shapenet.texttechnologylab.org/loadedobjects zuletzt aufgerufen am 19.08.2022

[2] Zugreifbar über (<id>:=tatsächliche id): http://shapenet.texttechnologylab.org/get?id=<id> zuletzt aufgerufen am 19.08.2022

[3] Alle Thumbnails sind über folgenden Link verfügbar: http://shapenet.texttechnologylab.org/thumbnails zuletzt aufgerufen am 19.08.2022

6 3D-Umgebung

Die 3D-Umgebung ist das Bindeglied zu den anderen Komponenten. Für die Implementierung der 3D-Umgebung wurde die Laufzeit- und Entwicklungsumgebung Unity verwendet, mit dessen Hilfe sich verschiedenste 2D wie auch 3D-Anwendungen umsetzen lassen. Dadurch wurden die verschiedenen Menüs in 2D umgesetzt, wobei die Szenengenerierung in 3D mit dreidimensionalen Objekten stattfindet. Im Folgenden werden die einzelnen Funktionen, welche der Benutzer verwenden kann näher vorgestellt.

6.1 Annotation von Relationen und Eigenschaften

Wie bereits am Titel abzulesen ist, ist die Annotation von Relationen und Eigenschaften ein wichtiger Bestandteil der ARES-Architektur. Diese sind maßgeblich für die letztendliche Szenengenerierung verantwortlich. Anders als bei manchen Lösungen, welche sich mit der Problemstellung der Szenengenerierung aus Text auseinandersetzen und riesige Datenmengen mit annotierten Szenen benötigen (vgl. Abb. 10.3), wird in diesem Fall der Versuch getätigt, die Datenmenge zu minimieren und mit möglichst wenigen Annotationen eine Szene korrekt darzustellen. Dementsprechend soll der Endanwender die Möglichkeit bekommen, die Annotationen selbst, in einer dafür vorgesehenen Benutzeroberfläche erweitern und erstellen zu können. Das trägt dazu bei, dass die benötigten Relationen und Eigenschaften schnell visualisiert werden können. Im Folgenden wird der Aufbau und die Verwendung dieser Annotationsfunktionen erläutert, die der Annotator zur Verfügung gestellt bekommt.

6.1.1 Eigenschaften definieren

Eigenschaften können verwendet werden, um Objekte in natürlicher Sprache näher zu beschreiben. Objekte können beispielsweise groß oder klein sein und zudem auch eine bestimmte Farbe tragen. Um Eigenschaften definieren zu können, wurde der ARES-Architektur eine entsprechende Funktion hinzugefügt. Dabei kann der Benutzer selbst Eigenschaften definieren für den Fall, dass die gewünschte Eigenschaft für das erforderliche Modell einer Szenengenerierung, innerhalb der dazugehörigen Annotationen nicht enthalten ist (vgl. Abb. 6.2).

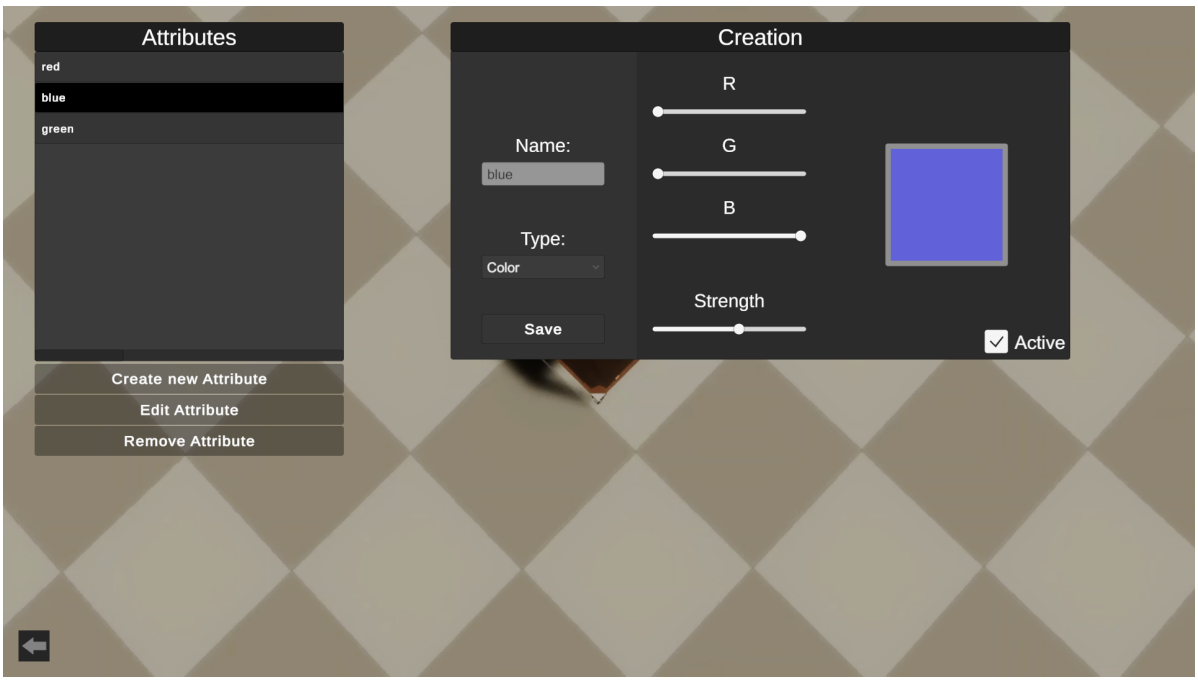

Abbildung 6.1: Benutzeroberfläche für die Annotation von Eigenschaften

Wie in Abb. 6.1 zu sehen ist, können neue Eigenschaften erstellt, alte bearbeitet oder bereits definierte wieder entfernt werden. Falls ein Attribut für ein Modell existiert, wird geprüft, ob ein Weiteres mit dem gleichen Namen in den Definitionen vorliegt, wonach alle aktivierten, definierten Typen, wie zum Beispiel „Color“ (s.a. Abb. 6.1), auf das Modell angewendet werden. Diese hinterlegten Annotationen sind intern als JSON abgespeichert und liegen in der Form von Abb. 10.4 vor.
Der Typ Color lässt sich über vier Schieberegler einstellen, wobei die daraus entstehende Farbe in Echtzeit rechts daneben visualisiert wird. Die definierte Stärke gibt an, zu wie viel Prozent die Ursprungsfarbe in die definierte Farbe übergeht. Ein Beispiel hierfür wäre, falls auf den Rot-Grün-Blau (RGB) Vektor $(1, 0, 0)$ die Annotation für die Farbe blau $(0, 0, 1)$ mit einer Stärke von 70% angewendet werden soll. Infolgedessen verschiebt sich die Farbskala vom Ursprungsvektor zum Zielvektor um 70% zum Farbwert $(0.3, 0, 0.7)$. Diese proportionale Herangehensweise ermöglicht eine natürlich wirkende Färbung der aus verschiedenen Teilen und somit auch Materialien bestehenden Modelle, da die unterschiedlichen Bestandteile, trotz Färbung, verschiedene Farben aufweisen.

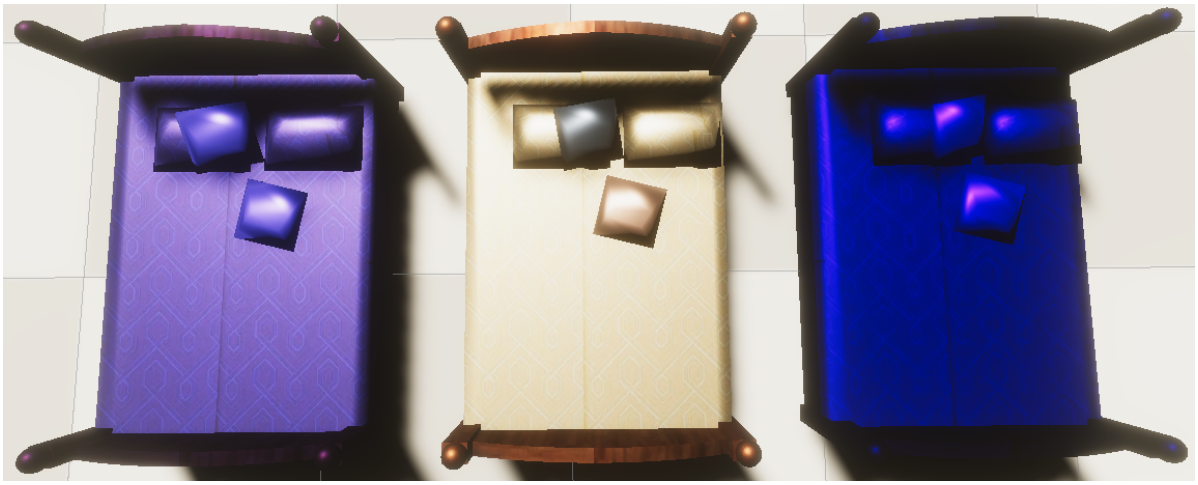

Abbildung 6.2: Versuch zur blauen Färbung eines Bettes (Original in der Mitte), wobei das linke Bett proportional mit einer Stärke von 40% und das rechte vollständig blau gefärbt wurde.

Wie aus dem Versuchsaufbau in Abb. 6.2 zu entnehmen ist, lassen sich die verschiedenen ursprünglichen Materialien für die proportionale Färbung noch erkennen, wie beispielsweise das Holz oder die unterschiedlich gefärbten Kissen. Das rechte Bett hingegen weist keine deutlich erkennbare Unterscheidung zwischen den Materialien auf und wirkt durch das hochprozentige blau unnatürlicher. Dennoch ist die linke Variante nicht makellos, da zum Beispiel das Holz nicht gefärbt werden sollte. Allerdings würde es zusätzliche Annotationen für jedes Modell erfordern, um die genauen Bestandteile zu kennen, welche für eine Färbung verwendet werden können. Andernfalls ist die beste Alternative ein Modell zu finden, das nicht nur die Merkmale eines blauen Bettes aufweist, sondern auch als ein solches blaues Bett annotiert ist. Das Problem ist, dass beide Alternativen größere Datensätze an verschiedenen Modellen wie auch Annotationen erfordern, wobei die Erstellung eines solchen Datensatzes nicht nur kosten- sondern auch zeitintensiv ist. Das hat den Grund, dass 3D-Artists benötigt werden, um die 3D-Modelle zu erstellen und professionell geschulte Annotatoren, die diese ebenfalls zuverlässig annotieren können. Das ist einer der Kernprobleme, wofür die ARES-Architektur Abhilfe schaffen soll.
Innerhalb von Unity kann auf viele weitere Eigenschaften der Modelle zugegriffen werden. Allerdings beschränkt sich die jetzige Umsetzung auf die hier vorgestellte und könnte in zukünftigen Versionen erweitert werden.

6.1.2 Relationen definieren

Der bedeutsamste Teil, um Gegenstände im Raum zu beschreiben, sind die Relationen, welche sie verbinden. Verfügt man nur über die Begriffe Apfel und Tisch, so lassen sich diese beiden auf den verschiedensten Weisen im Raum anordnen. Beispielsweise kann sich der Apfel unter dem Tisch, auf dem Tisch oder auch hinter dem Tisch befinden, wobei die Relation „hinter“ eine Frage der Perspektive darstellt. Die Perspektive des Beoachters nimmt in diesem Fall einen wesentlichen Einfluss darauf, ob die Relation als vor, hinter oder neben dem

Tisch interpretiert wird, falls die Ausrichtung des Tisches außer Acht gelassen wird. Zu diesem Zweck ist neben der Definition von Eigenschaften die Annotation von Relationen ein wesentlicher Bestandteil von ARES. Die grundlegende Verwaltung der Relationen (s. Abb. 10.7) bleibt im wesentlichen zu der von den Attributen unverändert. Allerdings ergibt sich der Unterschied, dass das Definieren der Relationen im dreidimensionalen Raum stattfindet. Das bringt mehrere Vorteile mit sich. Dem Annotator wird es ermöglicht, seine eigene definierte Relation direkt in Echtzeit zu sehen anstatt nur mit definierten Vektoren zu arbeiten. Ebenfalls erhält der Endanwender eine mögliche Betrachtung für jede einzelne definierte Relation, welches eine bessere Einsicht in die Logik der Szenengenerierung mit sich bringt, da die Szenengenerierungen nachvollziehbarer wird und adjustiert werden kann.

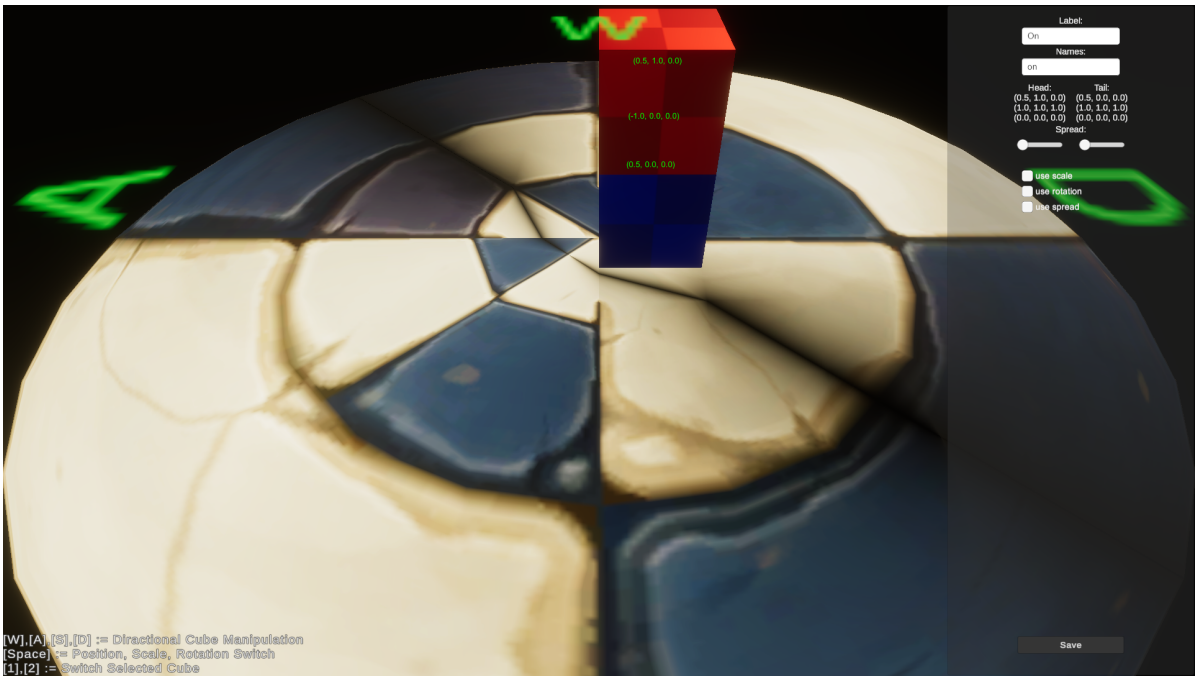

Abbildung 6.3: Benutzeroberfläche für die Annotation von Relationen. In diesem Fall wird die Relation „on“ mit Hilfe von 2 aufeinander gestapelten Kuben dargestellt, wobei der rote Kubus für den „Head“ und der blaue für den „Tail“ der Relation steht.

In Abb. 6.3 ist die dazugehörige Benutzeroberfläche zu sehen. Da eine Relation mit mehreren, in der natürlichen Sprache vorkommenden, Begriffen definiert werden kann, wie es beispielsweise für die Relation „next to“ der Fall ist, wobei ein mögliches Synonym „near“ wäre, wird es dem Annotator ermöglicht ein grundlegendes Label für die Relation zu wählen und alle möglichen Wörter aufzulisten unter welchen die Relation zu Zwecken der Szenengenerierung gefunden werden kann. Folglich gibt es mehrere Optionen mit deren Hilfe sich die Relation definieren lässt. Zumal lassen sich zwei Kuben, welche die letztendlichen Modelle der Entitäten Abstrakt zu Zwecken der Annotation darstellen, im Raum bewegen, skalieren und rotieren, wodurch auch komplexere Annotationen möglich werden. Relationen wie „around“ erfordern es nicht zwingend, dass eine Entität genau um einen festen Abstand zu

einer anderen positioniert wird. Daher ermöglicht ein Spread die Möglichkeit, dieses Objekt weiter im Raum zu streuen. Ebenfalls kann für diese Relation gesagt werden, dass diese unabhängig davon, ob sie links, rechts, vor oder hinter dem Objekt liegt, die Bedeutung beibehält, wodurch die Rotation des Objektes keinen Einfluss auf hat. Falls für eine Relation die genaue Skalierung notwendig sein sollte, da zum Beispiel für diese spezielle Relation beide Modelle gleich groß sein müssen, so lassen sich die Kuben ebenfalls skalieren und für diese Relation aktivieren.

6.2 Szenengenerierung

Die Szenengenerierung ist die Schlüsselstelle, welche alle Annotationen und Module miteinander verbindet. Mehrere Komponenten ermöglichen es, eingegebene Texte zu analysieren, nachträglich auszubessern und letztendlich in einer dreidimensionalen Umgebung zu visualisieren. Im Folgenden werden die Menübereiche der Szenengenerierung näher beleuchtet und Verwendungsmöglichkeiten aufgezeigt.

6.2.1 Textanalyse

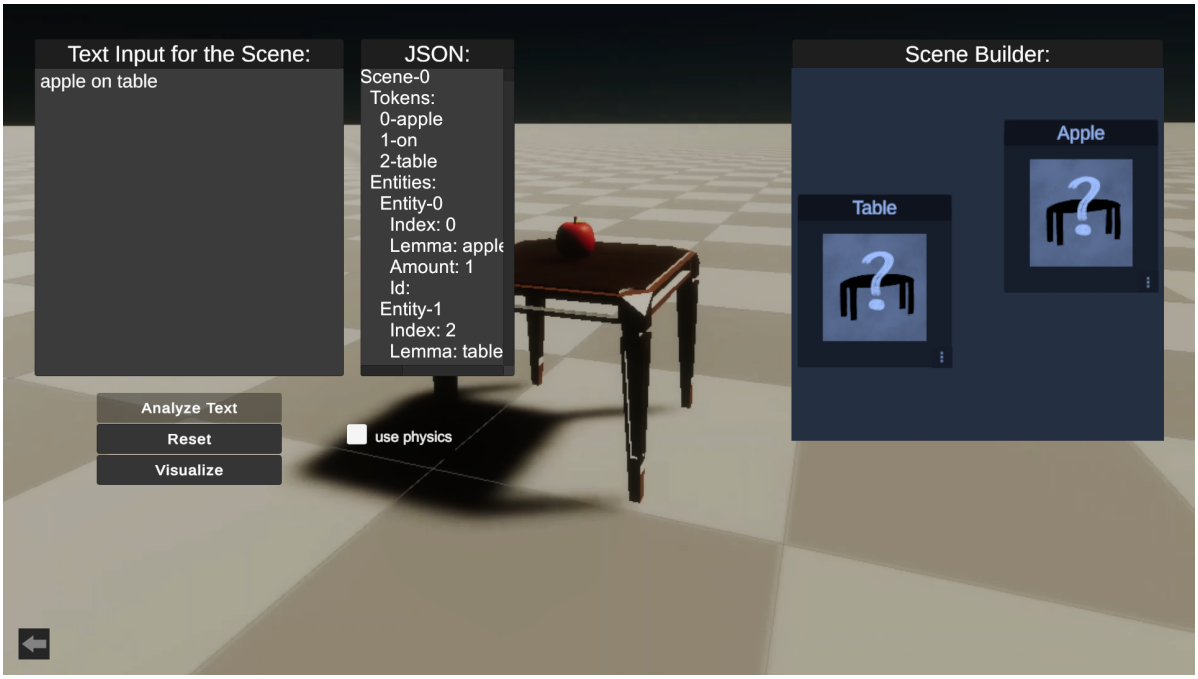

Abbildung 6.4: Benutzeroberfläche für das Eingeben und Analysieren von Texten, wie auch der Ausgabe der darin enthaltenen Informationen. In diesem Fall wurde als Text „apple on table“ eingegeben, welcher bereits analysiert und als JSON zurückgegeben wurde.

Aufgabe von Text2Scene Anwendungen ist es eine Szene direkt aus einem Text darzustellen. Das für die Textanalyse implementierte Modul, womit aus Text Informationen gewonnen

werden können, wurde bereits in Kap. 4 vorgestellt. Dementsprechend wird es innerhalb der Benutzeroberfläche ermöglicht, einen beliebigen Satz[1] einzugeben und diesen auszuwerten. Die aus der Auswertung gewonnenen Informationen werden darauffolgend in einem vorgesehenen Kasten eingeblendet, damit der Benutzer die Möglichkeit hat, den Zwischenschritt zur Szenengenerierung nachzuvollziehen und unerwartete Annotationen auch einsehen zu können.

6.2.2 Szenengenerierung durch Drag and Drop

Textanalyse-Module sind nicht Allmächtig und können ebenfalls Fehler in den automatisch generierten Annotationen aufweisen, weswegen Anpassungen möglich sein sollten. Nachdem der eingegebene Text erfolgreich analysiert wurde, erhält der Benutzer die Möglichkeit, die erhalten Annotationen nachträglich bearbeiten zu können. Inspiration für die Gestaltung der Benutzeroberfläche war der Salesforce Flow-Builder (*Salesforce Flow Builder* 2022), in dem sich deklarativ innerhalb der Salesforce Umgebung Logik umsetzen lässt, ohne dass Code notwendig ist. Dieser bietet ein Drag and Drop Prinzip, über welches sich neue Elemente hinzufügen lassen und das Ziehen von Linien, wodurch das Verbinden zweier aufeinanderfolgender Elemente ermöglicht wird. Das Verfahren des Flow-Builders wurde auf das Hinzufügen von Entitäten und Relationen übertragen und trägt den Namen Scene-Builder.

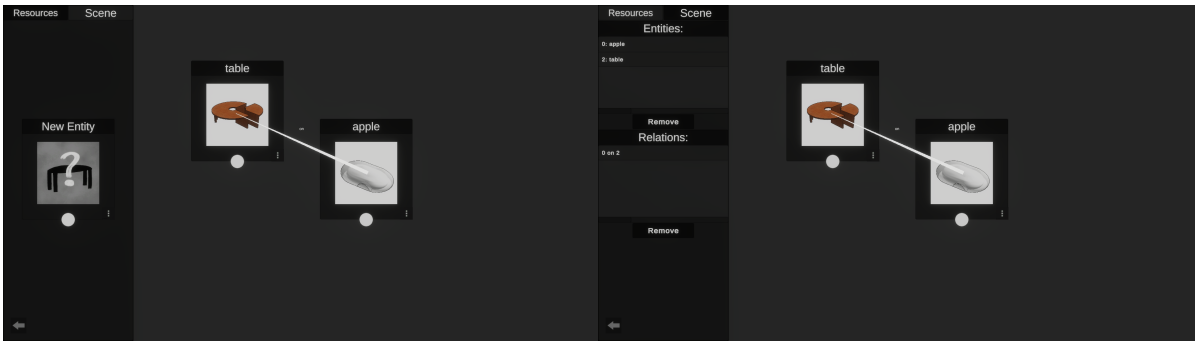

Abbildung 6.5: Grundlegende Benutzeroberfläche des Scene-Builders. Im linken Bild sieht man den Menüpunkt Ressourcen, bei dem sich neue Entitäten in die Szene ziehen lassen, wohingegen im rechten die einzelnen, in der Szene vorhandenen, Entitäten und Relationen aufgelistet werden.

In Abb. 6.5 ist die Implementierung des Scene-Builders zu sehen, wobei diese ähnlich zum Vorbild aufgebaut ist. Auf der rechten Seite ist die analysierte Szene visuell mit den Namen und Bildern der Entitäten dargestellt, wobei gerichtete und beschriftete Linien die Relationen widerspiegeln. Außerdem hat der Benutzer im linken Teilmenü einen Balken mit den bereitgestellten Werkzeugen. Einerseits gibt es im Tab „Resources“ ein Objekt für neue Entitäten, welches in die rechte Szene gezogen werden kann, um eine Entität hinzuzufügen. Andererseits lassen sich im Tab „Scene“ alle, in der Szene auftretenden, Entitäten und Re-

[1]Die implementierte Textanalyse unterstützt nur in der englischen Sprache verfasste Sätze.

lationen einsehen und wieder entfernen. Es ist anzumerken, dass falls eine Entität entfernt wird, auch alle damit verbundenen Relationen entfernt werden.

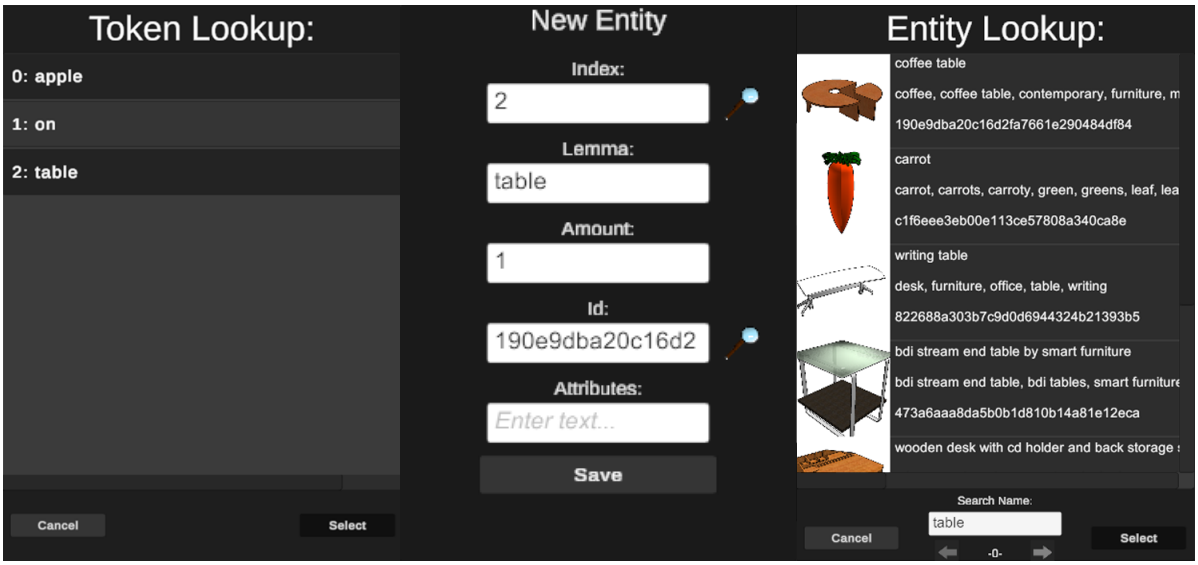

Abbildung 6.6: In der Mitte ist die Benutzeroberfläche zu sehen, welche für das Bearbeiten und Erstellen von Entitäten verwendet wird.

Das Entitäten Menü (s. Abb. 6.6) erscheint sobald eine neue Entität erstellt oder eine alte bearbeitet wird. Falls auf die Lupe neben dem Index Feld geklickt wird, erscheint das Token-Lookup-Menü (linkes Bild), in welchem die Tokens des eingegebenen Textes angezeigt und ausgewählt werden können. Wenn auf die Lupe neben dem Id Feld geklickt wird, findet man das Entity-Lookup-Menü (rechtes Bild), in welchem Modelle nach Tags gefiltert werden können. Die gefundenen Modelle sind über mehrere Seiten verteilt, zwischen denen gewechselt werden kann (die Anzahl der pro Seite auftretenden Modelle, lässt sich in den Einstellungen ändern). In diesem Fall stellt jedes verfügbare Modell in der Liste ein Thumbnail, den Namen, die annotierten Tags wie auch die Id des Objektes für den Benutzer zur Verfügung.

6.2.3 Logik: Anwendung der Eigenschaften und Relationen Annotationen

Die Logik der Szenengenerierung verwendet die verfügbaren Annotationen, welche in Kap. 6.1 näher erläutert wurden, wie auch die im Scene-Builder (s.a. Kap. 6.2.2) vorgenommenen Anpassungen. Der Algorithmus beginnt, indem für jede Entität das dazugehörige Modell instanziiert wird, wobei für das nicht vorhanden sein der Id, zunächst nach einem zum Lemma passenden Modell gesucht wird. Sobald alle Entitäten instanziiert wurden, werden die Eigenschaften der Entitäten auf die Modelle angewendet und falls einige Entität-Relation-Kombinationen bereits bekannt sind, so werden diese direkt angewendet. Sonst wird wie im Algorithmus 1, die Szenengenerierung ausgeführt und die Szene für den eingegebenen Text generiert. Der im Algorithmus berechnete errorValue wird durch die Verschiebung der Modelle beeinflusst, wobei sich die Gewichtung jeder Operation in den Einstellungen ändern

lässt.

Input :

- E, die im Text vorhandenen Entitäten;
- R, die im Text vorhandenen Relationen.

Output : S, Szene mit allen implementierten Entitäten und Relationen.

$S =$ Erstelle für jede Entität E_i eine Szene S_i mit dem instanziierten Modell;
while $|S| > 1$ **do**
    Finde $S_i, S_j \in S : S_i \neq S_j \wedge max(|\{S_i, S_j\} \in R|)$;
    $R_{ij} =$ finde alle $r \in R$, welche zwischen S_i und S_j verlaufen;
    Verbinde beide Szenen mit Hilfe einer Relation aus R_{ij};
    do
        Platziere S_i und S_j, ausgehend von den definierten Ursprungspunkten der Relation, an der nächstmöglichen Stelle, sodass der errorValue minimal bleibt;
    while S_i *und* S_j *haben überschneidende Modelle*;
end
return S;

Algorithmus 1 : Algorithmus zum Positionieren der Objekte

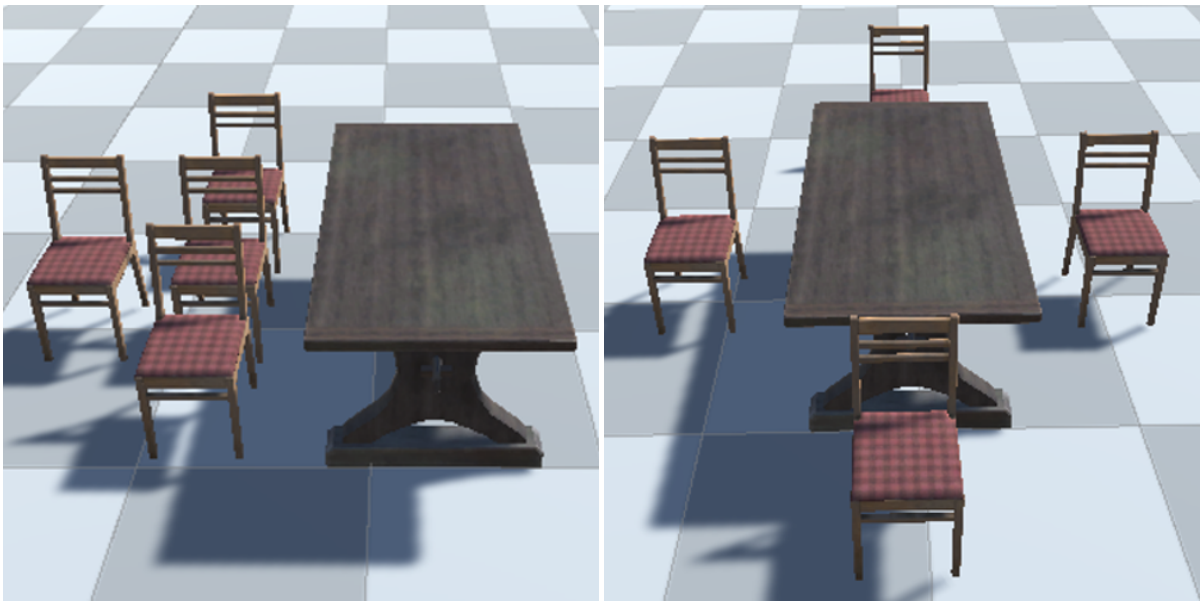

Abbildung 6.7: Mögliche syntaktisch korrekten Anordnungen der Szene „four chairs next to a table“.

In der erzeugten Umgebung sollen Korrekturen stattfinden können, wie das Auswechseln von Modellen, falls das Modell, welches für den Entitätsbegriff angewendet wurde, unangemessen ist oder falls kein Modell für die Entität gefunden wurde. Nicht gefundene Objekte

werden als Kubus dargestellt. Zudem benötigen Relationen wie „pillow on top of bed“ gesonderte Aufmerksamkeit, da das platzieren des Kissens in der Mitte des Bettes zwar syntaktisch korrekt wäre, allerdings die semantische Korrektheit erhöht werden könnte, durch das Positionieren am Bettende. Ein anderes Beispiel ist die Szene „four chairs next to a table“, bei welcher die Stühle zwar neben dem Tisch stehen, allerdings nicht zum Tisch hingerichtet sind (vgl. a. Abb. 6.7). Daher können Relationen und Entitäten zu einer Menge zusammengelegt und die Objekte im Raum angepasst werden, um diese als bekannte Entität-Relation-Kombinationen abzuspeichern und für weitere Szenengenerierungen nutzen zu können.
Hierfür wird eine Menge $S = \{G_1, ..., G_n\}$ definiert, wobei $G_i = (V, E, P)$ mit $i \in \{1, ..., n\}$ den i-ten Graphen beziehungsweise die i-te Teilszene darstellen soll, welche abgespeichert und wiederverwendbar gemacht werden sollen. V stellt die Knoten bzw. die Entitäten und E die Kanten bzw. die Relationen dar. Außerdem müssen die weiteren Eigenschaften wie die zusätzliche Skalierung oder Positionierung der Entitäten im Raum abgespeichert werden, welche hier als P dargestellt werden. Wenn nun die Szene $G' = (V, E)$ dargestellt werden soll, wird zunächst geprüft, ob ein Teilgraph G'_T existiert, für welchen gilt $(G'_T \subseteq G') \wedge (G'_T \in \{(V, E) : (V, E, P) \in S\})$. Falls G'_T existiert, sollen die Konfigurationen aus dem korrespondierenden Graphen in S zur Positionierung der Entitäten in der Szene von G' verwendet werden. Anzumerken ist, dass die Entitätsbezeichnung bei der Graphenfindung mitberücksichtigt werden muss, da nicht alle Knoten gleich sein müssen, welches eine zusätzliche Erhöhung der Zeitkomplexität mit sich führt. Ebenfalls kann es dazu kommen, dass mehrere G'_T existieren. Um dieses Problem zu vereinfachen, wird für den Fall das zwei solcher Graphen eine Schnittmenge untereinander haben, der Graph mit den meisten Entitäten gewählt, da sich beide P in dem Fall widersprechen können. Falls beide allerdings disjunkt sind, können beide P ohne gegenseitige Widersprüche realisiert werden.
Mit Hilfe dieser Erweiterung der grundlegenden Szenengenerierung wurde es ermöglicht die semantische Bedeutung hervorzuheben, wodurch komplexere Relationen wie „in“ oder „under“ besser dargestellt werden können. Da es bei diesen Einschränkungen auf Grund der Logik zur Platzierung von Modellen gibt, welche zu großen Teilen auf der Vereinfachung der Modellränderfindung und der daraus resultierende Interpretation als Boxen entstehen.

7 Evaluation

7.1 Textanalyse

7.1.1 Aufbau

Um die Zuverlässigkeit der Textanalyse überprüfen zu können, wurden einige von Menschen erstellte Annotationen verwendet. Die bereits annotierten Texte werden nochmals durch die implementierte Textanalyse verarbeitet und mit den Annotationen verglichen. Hierfür wurden vom Texttechnologylab bereitgestellte Annotationen verwendet, welche nach *Language resource management — Semantic annotation framework — Part 7: Spatial information* 2020 und Ide und Pustejovsky 2017 angelegt wurden. Allerdings werden für die Auswertung, nur Annotationen der Entitäten und Relationen verwendet, welche für Szenengenerierungen relevant sind. Für Entitäten wurden die annotierten „SPATIAL_ENTITY“ überprüft, welche räumlich relevant sind und nur als solche annotiert werden, falls sie sich entweder bewegen oder die Möglichkeit haben in Bewegung gebracht zu werden (welches Orte aufschließt) (vgl. Ide und Pustejovsky 2017, S. 997). Hingegen werden für die Relationen, „OLINK“, welche gerichtete oder auch relative Bezüge zwischen Entitäten angeben (vgl. Ide und Pustejovsky 2017, S. 22) bzw. auch „QSLINK“ Annotationen, welche die topologische Anordnung der Objekte im Raum beschreiben (vgl. Ide und Pustejovsky 2017, S. 21), verwendet. Mit Hilfe dieser Annotationen kann überprüft werden, ob die Indizes für die Entitäten als auch die Relationen übereinstimmen und somit richtig generiert werden.

7.1.2 Ergebnisse

Durch die Auswahl und Filterung des Datensatzes, wie in Kap. 7.1.1 beschrieben, entstand ein Datensatz mit 511 Sätzen, 13605 Tokens, 2389 Entitäten und 457 Relationen. Um die Ergebnisse besser auswerten zu können, wurden die gängigen Evaluationsmetriken fürs Maschinelle Lernen (Recall, Precision und F1) (vgl. *Precision and recall* 2022) mit den unten genannten Formeln 7.1-7.3 berechnet.

$$Recall = \frac{TP}{TP + FN} \tag{7.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{7.2}$$

$$F1 = \frac{2TP}{2TP + FP + FN} \tag{7.3}$$

Im Zuge dieser Evaluierung ergaben sich die in Tabelle 7.1 befindlichen Ergebnisse. Hierbei fällt auf, dass zwar eine hohe Genauigkeit bei der Entitäten-Bestimmung erzielt wird, allerdings die Relationen-Erkennung schlechter ausfällt. Dies hat vermutlich den Grund, dass die Relationen Bestimmung deutlich komplexer ist. Ein weiterer Grund könnte sein, dass das

Dependency Mapping von SpaCy (Kap. 2.4) und der ursprünglichen Textanalyse (Kap. 2.3) anders verläuft, wodurch dies angepasst werden müsste. Zudem ist von den Metriken vor allem der Recall interessant, da False Positive (FP) Ergebnisse von ARES rausgefiltert werden, wenn diese innerhalb der Annotationen nicht gefunden werden. Wohingegen False Negative (FN) Auswertungen entscheidender sind, da hierdurch Informationen in Form von fehlenden Entitäten bzw. Relationen verloren gehen. Daraus folgt, dass nur 18,81% aller Relationen in den analysierten Texten gefunden werden, welches verbesserungswürdig ist. Da der verwendete Datensatz jedoch nur über 457 Relationen verfügt, ist dies nur eine grobe Einschätzung und kann bei Erweiterung des Datensatzes mit einer erhöhten Genauigkeit bestimmt werden.

Tabelle 7.1: Auswertung des für ARES implementierten Textanalyse Modells.

	Recall	Precision	F1
Entitäten	74,88%	62,24%	67,98%
Relationen	18,81%	28,85%	22,78%

Die Relationen wurden ebenfalls innerhalb der Annotationen kategorisiert, wobei eine Gesamtmenge an 34 verschiedenen Relationen entstanden ist. Hingegen werden die Relationen für die verwendete Implementierung grob eingeteilt, wodurch einzelne Begriffe, wie bei Relationen die das Wort „left“ beinhalten, zu „leftside“ zusammengefasst werden. Mit einer daraus resultierenden Anzahl an 83 verschiedenen Relationen, lassen sie sich mit einem geringen Mehraufwand innerhalb von ARES annotieren, indem für alle Relationentypen ebenfalls Alternativnamen angegeben werden. Dadurch wäre es möglich mit bereits 34 verschiedenen definierten Relationen die Meisten abzudecken ohne eine größere Datenmenge generieren zu müssen.

7.2 Szenengenerierung

7.2.1 Aufbau

Im Zuge der Evaluierung der Szenengenerierung, wurden 10 Personen mit Informatik Hintergrund befragt, um zu ermitteln, ob die ARES-Architektur den erwarteten Anforderungen entspricht. Hierzu wurde den Personen ein Umfragebogen (s. Abb. 10.8) zum Ausfüllen gegeben, wobei sie Zeit bekamen neben ARES noch das „Text to 3D Scene“ (vgl. Kap.1.2) wie auch das „Language-Driven Synthesis of 3D Scenes from Scene Databases“ (vgl. Kap.2.3) Modell näher zu testen.

7.2.2 Ergebnisse

Anhand der Umfrage (s. Abb. 7.1) lassen sich erste Erkenntisse erzielen. Die Annotationstools wurden von den Benutzern überwiegend akzeptiert, wobei die Annotation der Relationen beliebter ist als die der Eigenschaften. Dies könnte die Folge davon sein, dass Relationen komplexer zu annotieren sind und ein solches Annotationstool die Erstellung deutlich vereinfachen kann. Im Gegensatz dazu ist die Varianz bei den Eigenschaften größer, welches

allerdings auch eine Folge davon sein könnte, dass für die jetzige Version nicht genug Menüpunkte für die Integration neuer Eigenschaften, implementiert wurden. Benutzer die eher die Architektur bewertet haben, könnten höhere Punkte gegeben haben im Gegensatz zu denjenigen, welche einen Fokus auf die implementierten Möglichkeiten legten.
Ebenfalls lässt sich ablesen, dass der Scene-Builder bevorzugt wird, welches allerdings auch an der schlechter ausfallenden Textanalyse (vgl. Tabelle 7.1) liegen könnte.
Nichtsdestotrotz fällt die Szenengenerierung der anderen Implementierung besser aus, welches sich in den Wertungen niederschlägt, was zu erwarten war, da sie auch auf einer höheren Datenlast aufbaut. Jedoch wird durch die Integration des Benutzers in die internen Prozesse der Szenengenerierung, diese auch besser für ihn verständlich und führt somit zu einem Wissenszuwachs.

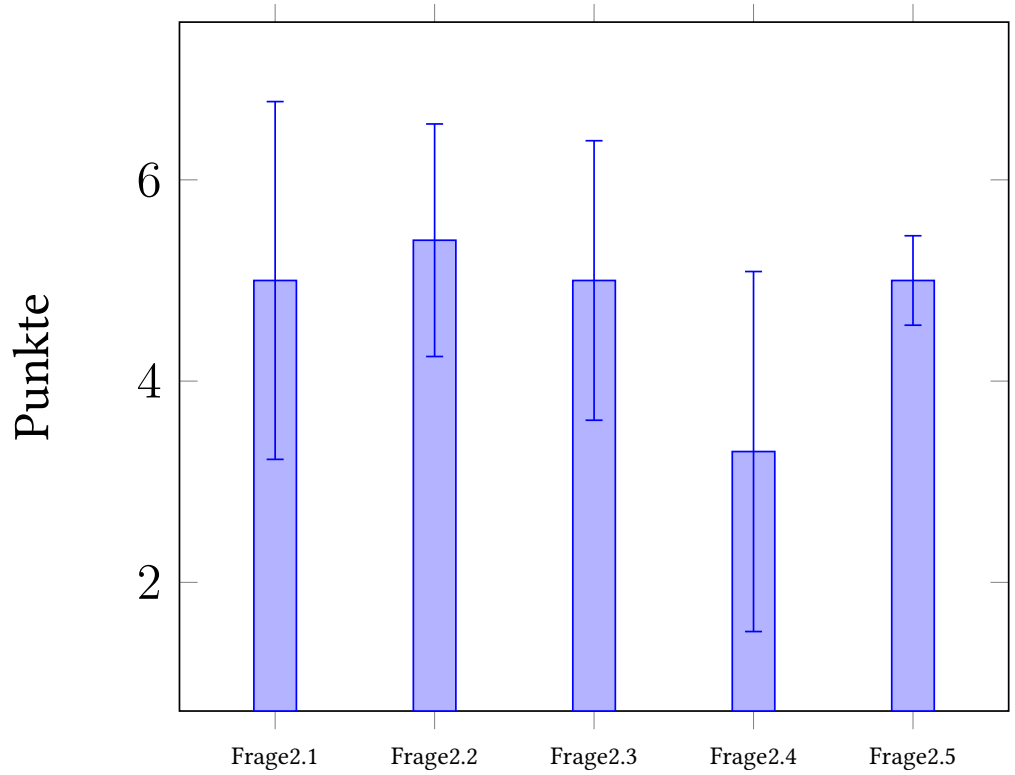

Abbildung 7.1: Ergebnisse zur ARES Umfrage (vgl. Abb. 10.8).

Es wurde ebenfalls geprüft, wie lange ein Annotator pro Eigenschaft, Relation oder aber auch für eine Szene im Scene-Builder braucht (vgl. Tabelle 7.2). Für eine Eigenschaft wurden rund 15,4 Sekunden benötigt. Dies ist zwar nicht viel, allerdings hat die Farbe gelb wahrscheinlich mehr Zeit in Anspruch genommen, da diese eine Kombination aus mehreren Farben benötigte. Daher wäre es zumindest für Eigenschaften wie Farben sinnvoll, Bibliotheken für die Farbnamen mit den dazugehörigen Hexadezimalcodes zu verwenden, um diese schneller integrieren zu können. Hingegen wurde eine Relation innerhalb von 24,25 Sekunden erstellt, welches für ein solch komplexes Konstrukt beachtlich ist. Für die Relation „next-to“ weicht die Zeit vermutlich deswegen ab, weil die Annotatoren keine Restriktionen für das

Anwenden von zusätzlichen Annotationen wie Spreads (s.a. Abb. 10.5) hatten, wodurch manche diese ebenfalls angaben.
Da die Durchschnittszeit für drei Entitäten und 2 Relationen beim Scene-Builder bei ca. 50 Sekunden liegt und die Umsetzung positiv aufgenommen wurde (vgl. 7.1), ist davon auszugehen, dass es sich bei ca. 10 Sekunden pro zu annotierenden Textbestandteil, um eine ausreichend kurze Zeit handelt, da es sich im Regelfall nur um Korrekturen und nicht komplette Szenen handelt.

Tabelle 7.2: Auswertung (Angabe in Sekunden) der in ARES implementierten Annotationstools (vgl. Abb. 10.8).

PersonNr	Frage1.1	Frage1.2	Frage1.3	Frage1.4	Frage1.5
1	12	20	20	25	57
2	13	16	18	21	48
3	13	17	19	26	46
4	13	16	23	26	53
5	13	20	19	39	49
6	12	19	21	28	59
7	11	13	16	24	40
8	14	18	20	37	45
9	15	19	19	34	57
10	14	20	18	32	52
Durchschnitt	13,00	17,80	19,30	29,20	50,60
Max	15,00	20,00	23,00	39,00	59,00
Min	11,00	13,00	16,00	21,00	40,00
Varianz	1,33	5,29	3,57	35,73	37,16

8 Anregungen für weiterführende Arbeiten

Damit möglichst viele von den vorgestellten Möglichkeiten profitieren können, wird die Implementierung ebenfalls veröffentlicht.[1] Aufgrund der Modularität, lässt sich diese Anwendung leicht weiterentwickeln und kann im Zuge der Veröffentlichung, als Open Source Projekt weitergeführt werden. Ebenfalls wäre die künftige Integration einer Datenbank möglich, um das Kollaborieren der Annotatoren und das Erweitern der Annotationen zu vereinfachen. Im Folgenden werden einige Erweiterungen vorgestellt.

8.1 Verbesserung der automatischen Modell Wahl

Zur Zeit wird einer Entität, ein Modell, nur mit Hilfe des Namens wie auch der Attribute und der anschließenden Suche innerhalb der Modell-Tags, zugewiesen. Dieses Vorgehen führt selbstverständlich dazu, dass der Kontext außer Acht gelassen wird, wobei ein Apfel beispielweise mit einem Gerät der Marke „Apple“ verwechselt werden kann. Folglich wäre die Integration einer Kontextbezogenen-Suche, eine mögliche Erweiterung für zukünftige Versionen.

8.2 Erweiterung der Szenengenerierung um den Ort

Bisher werden die Modelle auf einem leeren Terrain platziert, wodurch für den Benutzer Emersion verloren geht. Daher wäre ein weiterer Vorschlag, die Implementierung um weitere Terrains zu erweitern. Beispielsweise kommen hierfür die Kategorien „Stadt“ und „Natur“ in Frage. Ebenfalls ist die Implementierung einer Ort-Suche innerhalb des Textes wünschenswert, um die Objekte in der gewünschten Umgebung Platzieren zu können.

8.3 Hyperparameteranalyse für die Positionierungslogik

Die Szenengenerierung baut auf der Ermittlung eines Fehlerwertes auf, welcher den Unterschied zwischen der definierten Relation und der letztendlichen Platzierung aufzeigt. Die für das Verschieben im dreidimensionalen Raum benutzten Fehler-Konstanten, lassen sich in den Einstellungen anpassen. Folglich kann eine Hyperparamteranalyse erfolgen, um die bestmöglichen Parameter für eine möglichst syntaktisch und semantisch korrekte Szene zu ermitteln.

[1] https://github.com/texttechnologylab/ARES

8.4 Integration neuer Textanalyse Modelle

Abseits der hier implementierten Textanalyse, lassen sich ebenfalls geeignetere oder für die eigenen Bedürfnisse nötigen Modelle implementieren. Im Gegensatz zum verwendeten SpaCy Modell, könnte ein Modell implementiert werden, welches die Relationen direkt kategorisiert (unter einem Sammelbegriff für z.B. „around“, „near“ oder „nextto“ zusammengefasst) und an die 3D-Umgebung weitergeleitet, um Alternativnamen für die einzelnen Relationen nicht selbst aufzählen zu müssen.

9 Fazit

Aufgabe dieser Arbeit war es, eine Architektur zu entwickeln, welche die benötigte Datenmenge reduziert und die internen Prozesse übersichtlich veranschaulicht. Ebenfalls sollten die im Rahmen dieses Projektes implementierten Tools dabei helfen, Annotationen vereinfacht in einer benutzerfreundlichen Umgebung zu erstellen.
Wie in Kap. 7.2.2 zu sehen ist, sind die Szenengenerierungen mit bereits wenigen Annotationen möglich und die Entscheidungslogik bleibt verständlicher als bei vergleichbaren Text2Scene Anwendungen. Jedoch kann ARES mit der semantisch korrekten Darstellung der anderen Modelle nicht mithalten, da hierfür mehr Annotationen nötig sind. Allerdings war dies war auch nicht das Ziel dieser Arbeit.
Die implementierten Annotationstools wurden ebenfalls überwiegend von den Annotatoren angenommen (vgl. Abb. 7.1) und konnten nicht nur bei dem Verständnis der Szenengenerierung, sondern auch bei dem Erweitern der Annotationen helfen, um inhaltlich bessere Szenen zu erhalten.
Zusammenfassend lässt sich sagen, dass die für das Projekt gesetzten Ziele überwiegend erfüllt wurden, wobei die Architektur von der Anwendung, Annotationen und Erweiterungen lebt. Diese sind wichtig, um komplexere Szenen, semantisch korrekter darstellen zu können, wobei die vorhandene Logik in diesem Sinne zukünftig verbessert werden kann.

10 Anhang

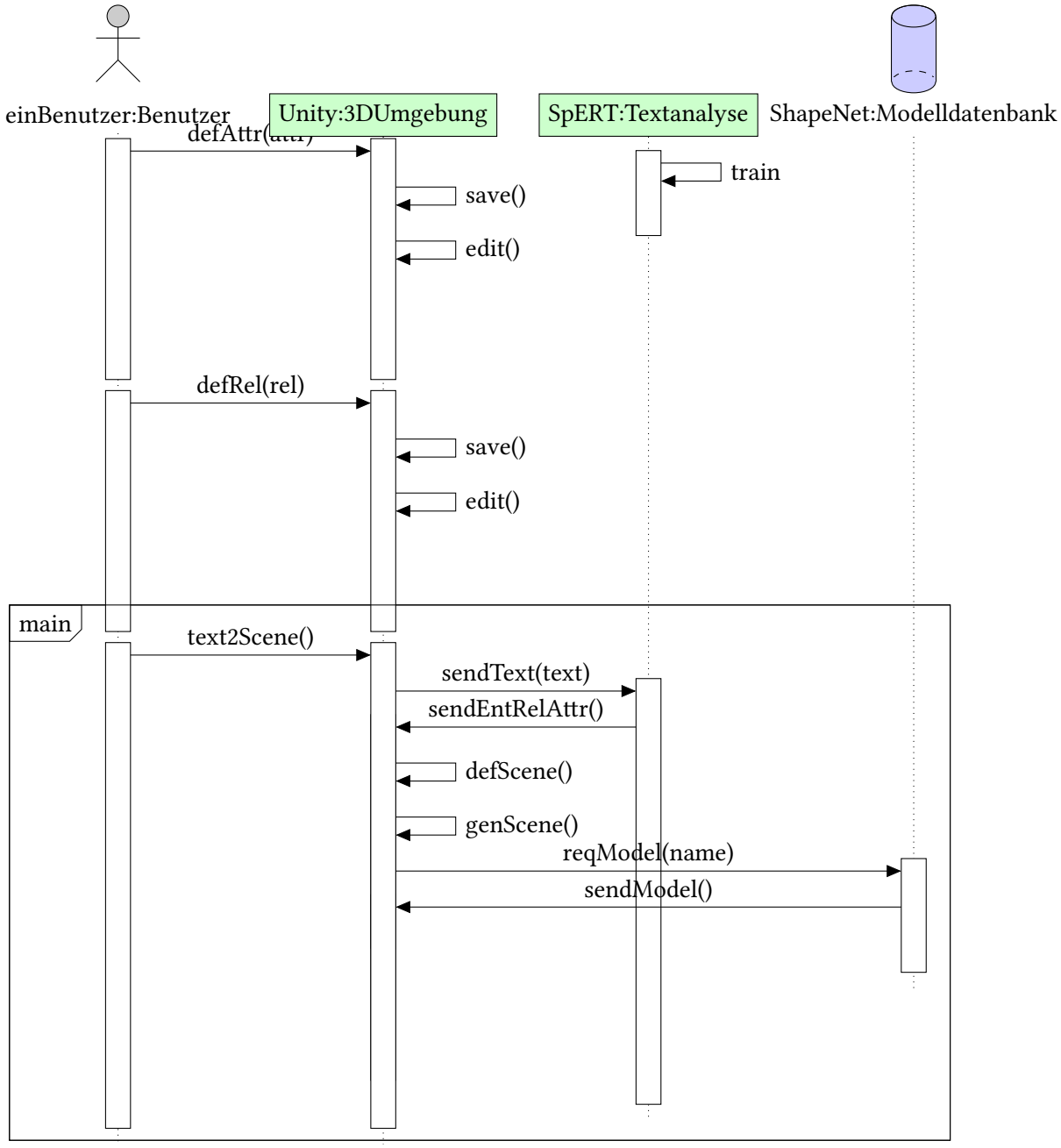

Abbildung 10.1: Sequenzdiagramm für die Funktionallitäten der ARES-Architektur

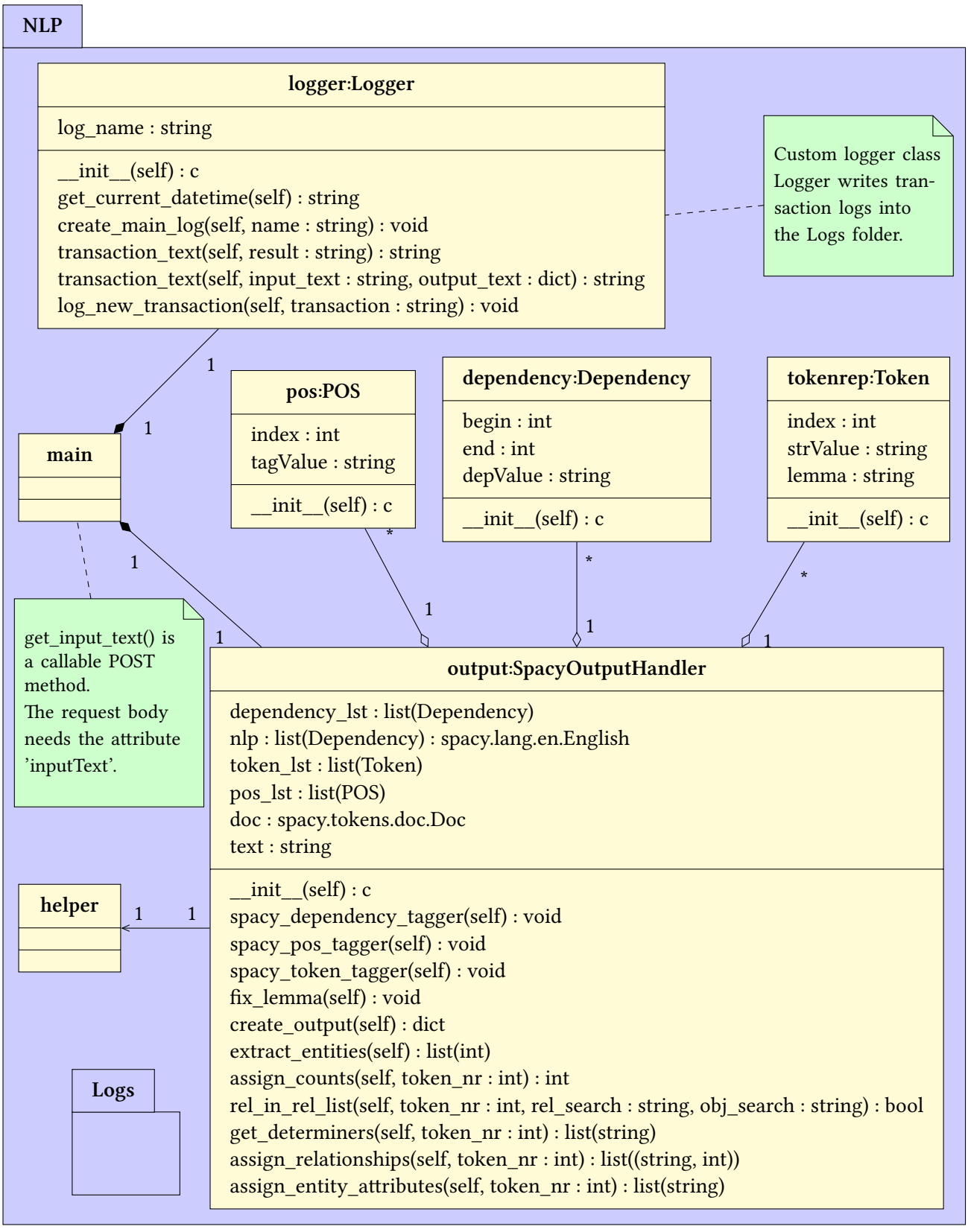

Abbildung 10.2: Klassendiagramm samt Paketstruktur und Dateistruktur für die Textanalyse

There is a bed and there is a nightstand next to the bed.

There is a bed with three pillows and a bedside table next to it.

…

Abbildung 10.3: Annotierte Szene mit Beschreibung der Forscher (oben) und zusätzlichen Annotationen von AMT (unten) (Chang, Monroe u. a. 2015, S. 3)

```
{
    "result" :      [
                        {
                            "name" : string
                            "attributeColor" :  {
                                                    "r" : float
                                                    "g" : float
                                                    "b" : float
                                                    "a" : float
                                                    "isActive" : bool
                                                }
                        }
                    ]
}
```

Abbildung 10.4: JSON Struktur für das Abspeichern der definierten Eigenschaften

```
{
    "result" :  [
                    {
                        "label" : string
                        "names" : [string]
                        "head" :    {
                                    "position" :    {   "x" : float
                                                        "y" : float
                                                        "z" : float
                                                    }
                                    "scale" :       {   "x" : float
                                                        "y" : float
                                                        "z" : float
                                                    }
                                    "rotation" :    {   "x" : float
                                                        "y" : float
                                                        "z" : float
                                                    }
                                    "spread" : bool
                                    }
                        "tail" :    {
                                    "position" :    {   "x" : float
                                                        "y" : float
                                                        "z" : float
                                                    }
                                    "scale" :       {   "x" : float
                                                        "y" : float
                                                        "z" : float
                                                    }
                                    "rotation" :    {   "x" : float
                                                        "y" : float
                                                        "z" : float
                                                    }
                                    "spread" : bool
                                    }
                        "useScale" : bool
                        "useRotation" : bool
                        "useSpread" : bool
                    }
                ]
}
```

Abbildung 10.5: JSON Struktur für das Abspeichern der definierten Relationen

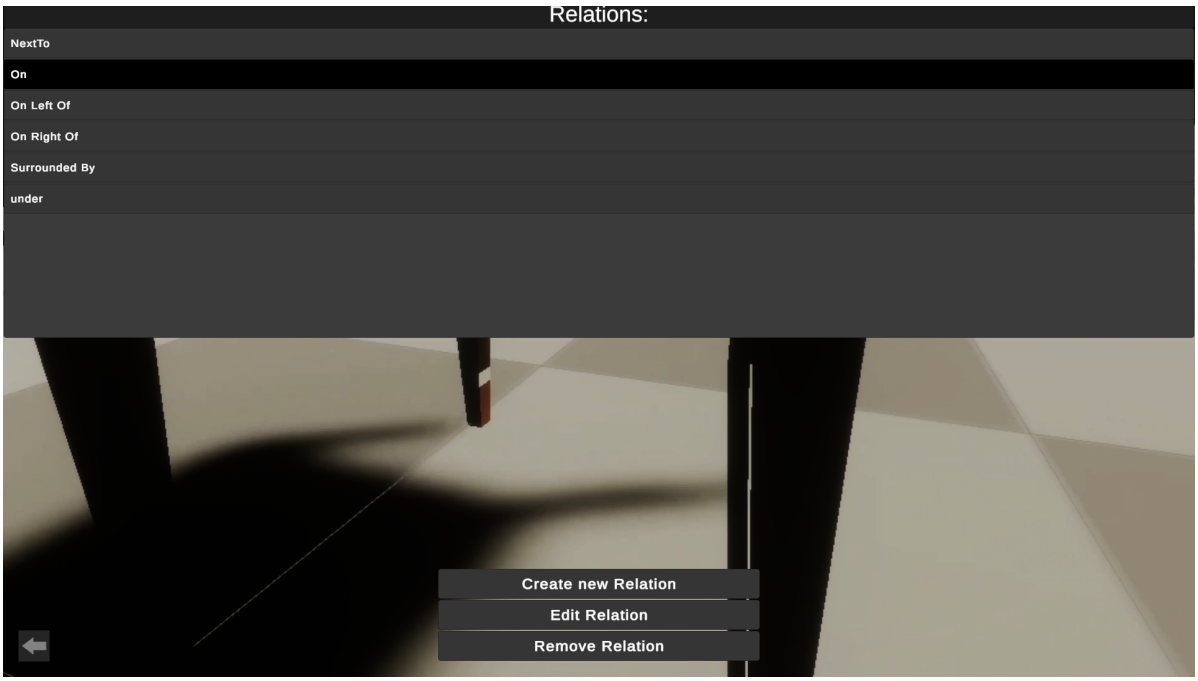

Abbildung 10.6: Benutzeroberfläche für das Verwalten der Relationen

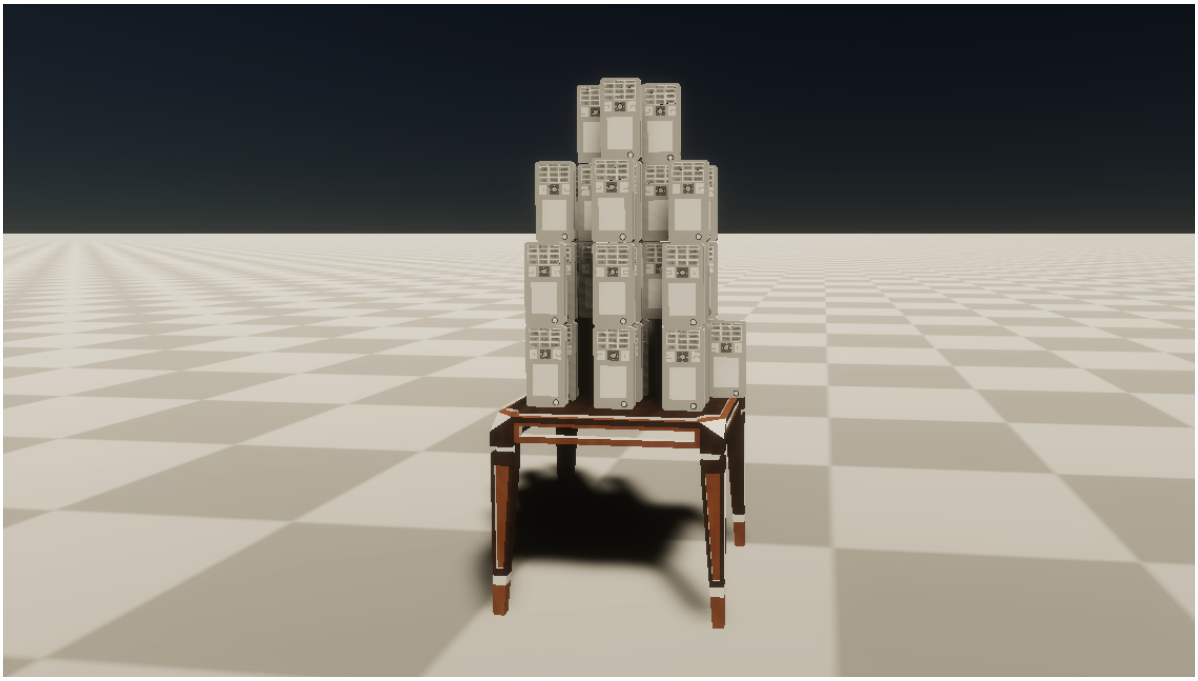

Abbildung 10.7: Beispielhafte Szenengenerierung der ARES Implementierung für die Szene „40 phones on the table“

ARES Umfragebogen

Bitte füllen Sie den Umfragebogen wahrheitsgemäß aus, nach dem Testen der darin genannten Implementierungen (ARES, Language-Driven Synthesis of 3D Scenes from Scene Databases[1])...

Nachdem Sie sich innerhalb von ARES mit den Funktionen zur Eigenschaften und Relationen Annotation, wie auch dem verwenden des Scene Builders, auseinandergesetzt haben, notieren Sie bitte ihre benötigte Zeit für folgende Aufgaben...

1.1. Eigenschaften: blau definieren =

1.2. Eigenschaften: gelb definieren =

1.3. Relationen: on definieren =

1.4. Eigenschaften: nextto definieren =

1.5. Scene Builder[2]: chair $\underset{\text{nextto}}{\longrightarrow}$ table $\underset{\text{on}}{\longleftarrow}$ apple =

	Strongly disagree						**Strongly agree**
2.1. Ich finde, dass man mit ARES effektiv Eigenschaften definieren kann	1	2	3	4	5	6	7
2.2. Ich finde, dass man mit ARES effektiv Relationen definieren kann	1	2	3	4	5	6	7
2.3. Ich finde den Scene Builder besser als die Textanalyse	1	2	3	4	5	6	7
2.4. Ich finde die Szenengenerierung von ARES besser als die vom Language-Driven Synthesis of 3D Scenes from Scene Databases Modell	1	2	3	4	5	6	7
2.5. Ich kann die Entstehung der Szene innerhalb von ARES besser nachvollziehen als beim Language-Driven Synthesis of 3D Scenes from Scene Databases Modell	1	2	3	4	5	6	7

[1] https://manyili12345.github.io/Publication/2018/T2S/index.html

[2] Schreiben Sie hierfür einen geeigneten Satz, löschen Sie alle Entitäten und Relationen im Scene Builder und starten Sie die Zeit, wenn sie im leeren Scene Builder beginnen

Abbildung 10.8: Zur Evaluierung von ARES verwendeter Umfragebogen

Glossar

Annotation Anmerkung zur Erweiterung der verfügbaren Informationen. 1, 2, 4, 6, 9–11, 13, 14, 16, 18–26, 29, 30, 38

Annotator Ein Mensch, welcher für das Erstellen von Annotationen qualifiziert ist. 20, 22, 23

API Eine API bzw. eine „Programmierschnittstelle […] ist ein Programmteil, der von einem Softwaresystem anderen Programmen zur Anbindung an das System zur Verfügung gestellt wird“ (*Programmierschnittstelle* 2022). 16, 19, 44

Architektur Architektur im Sinne der Softwarearchitektur, also dem Aufbau und Zusammenspiel der zu Grunde liegenden Komponenten (vgl. *Softwarearchitektur* 2022). 4, 6, 9, 10, 13, 20, 22, 30, 31, 36

attr Mit der „attr“ Dependency, kurz für „attribute“(Choi und Palmer 2012, S. 27), werden Token annotiert, welche „Substantivgruppen, die kein VP Prädikat sind und in der Regel nach einem Korpulaverb folgen“(Choi und Palmer 2012, S. 27), darstellen. 15

compound Mit der „compound“ Dependency werden Token annotiert, welche „entweder ein Substantiv, das den Kopf einer Substantivgruppe modifiziert, eine Zahl, die den Kopf einer Mengenangabe modifiziert, ein Wort mit Bindestrich oder eine Präposition, die den Kopf einer Präpositionalphrase modifiziert“ (*ClearNLP Dependency Labels* 2022), ist. 14

deklarativ Deklarativ bezeichnet im Kontext der deklarativen Programmierung „ein Programmierparadigma, bei dem die Beschreibung des Problems im Vordergrund steht. Der Lösungsweg wird dann automatisch ermittelt“ (*Deklarative Programmierung* 2022). 25

Dependency Satzbeziehung innerhalb einer binären Relation (vgl. Marneffe und Manning 2016, S. 1–2). 14, 15

det Mit der „det“ Dependency, kurz für „determiner“(Choi und Palmer 2012, S. 35), werden Token annotiert, welche „einer der POS Tags DT, WDT oder WP tragen und den Kopf einer Substantivgruppe modifizieren“(Choi und Palmer 2012, S. 35). 15, 16

diskriminatives Modell Das diskriminative Modell hat innerhalb eines GANs die Aufgabe, die Wahrscheinlichkeit zu maximieren, zwischen einem von dem generativen Modell erzeugten Datensatz und dem Trainingsdatensatz korrekt zu unterscheiden (vgl. I. J. Goodfellow u. a. 2014, S. 1). 9

Drag and Drop Menüfunktion einer graphischen Benutzeroberfläche für das Bewegen von Komponenten mit Hilfe des Mauszeigers, wobei zum Beginn eine Maustaste gedrückt werden muss, um einen Vorgang zu starten, welcher durch das Loslassen der Taste beendet wird. 4, 25

DT Mit dem „DT“ POS, werden Token annotiert, welche die Zugehörigkeit des Nomen spezifizieren (vgl. *Universal Dependencies: POS - DET* 2022). 42

Entität Nach dem Duden ist eine Entität in der Informationstechnik eine „eindeutig identifizierbare Größe (z. B. Person oder Objekt), über die Informationen gespeichert werden“(*Duden online: Entität* 2022). 2, 4, 6, 9, 10, 14–16, 19, 23, 25–28

expl Mit der „expl“ Dependency, kurz für „explative“(Choi und Palmer 2012, S. 27), werden Token annotiert, welche „ein existenzielles ‚there‘ in der Subjektposition stehen haben“(Choi und Palmer 2012, S. 27). 15

FN FN, kurz für „True Negative“, beschreibt einen vom Modell falsch negativ ausgewerteten Ausgang. 29

FP FP, kurz für „False Positive“, beschreibt einen vom Modell falsch positiv ausgewerteten Ausgang. 29

GAN Ein Generative Adversial Network(GAN) ist ein Machine Learning Modell für das unüberwachte Lernen, welches ein generatives wie auch diskriminatives Modell verwendet (vgl. I. J. Goodfellow u. a. 2014). 9, 42, 43

generatives Modell Das generative Modell hat innerhalb eines GANs die Aufgabe, die Wahrscheinlichkeit zu maximieren, dass das diskriminative Modell Fehler macht, indem es seine Ausgabe verbessert (vgl. I. J. Goodfellow u. a. 2014, S. 1). 9

Head Anfangspunkt einer Relation. 6, 23, 44

JSON JavaScript Object Notation ist ein verbreitetes Dateiformat, welches über die letzten Jahre zum wichtigsten Datenaustauschformat wurde (vgl. Lv, Yan und He 2018, S. 1). 6, 16–19, 21, 24, 38, 39

Lemma Grundform eines Wortes. 14, 26

Logdatei „Eine Logdatei […] enthält das automatisch geführte Protokoll aller oder bestimmter Aktionen von Prozessen auf einem Computersystem“ (*Logdatei* 2022). 16

Modell Ein Machine Learning-Modell wird auf Grundlage eines Algorithmus und bereitgestellten Daten trainiert, um darauf folgend Daten analysieren und darin Muster erkennen zu können (vgl. *Was ist ein Machine Learning-Modell?* 2022). Hingegen handelt es sich bei 3D Modellen, um Objekte, welche im dreidimensionalen Raum visualisiert werden können.. 2, 4, 5, 8–11, 13, 14, 18, 19, 21–24, 26–28, 30, 33–35

modul Abgeschlossenes Programm, welches über eine definierte Schnittstelle verfügbar ist. 2, 9, 13, 24, 25

NLP NLP, kurz für „Natural Language Processing“, ist ein Forschungsteilbereich der Künstlichen Intelligenzen, welche sich damit beschäftigt, Computern das gleiche Textverständins eines Menschen beizubringen (vgl. *Natural Language Processing (NLP)* 2022). 9

NN Mit dem „NN“ POS, werden Token annotiert, welche als Nomen im Singular oder als Menge angegeben sind (vgl. *SpaCy POS Tagging Glossar* 2022). 14

NNP Mit dem „NNP“ POS, werden Token annotiert, welche Nomen im Singular sind und keinen Artikel annehmen außer „the“ (vgl. *SpaCy POS Tagging Glossar* 2022) . 14

NNPS Mit dem „NNPS“ POS, werden Token annotiert, welche Nomen im Plural sind und keinen Artikel annehmen außer „the“ (vgl. *SpaCy POS Tagging Glossar* 2022). 14

NNS Mit dem „NNS“ POS, werden Token annotiert, welche als Nomen im Plural angegeben sind (vgl. *SpaCy POS Tagging Glossar* 2022). 14

nummod Mit der „nummod“ Dependency, kurz für „numeric modifier“(Choi und Palmer 2012, S. 36), werden Token annotiert, welche „eine Nummer oder ein Gewichtungsausruck sind und den Head der Substantivgruppe modifizieren“(Choi und Palmer 2012, S. 36). 14, 15

POS Wortart eines Tokens. 14

POST Eine HTTP Methode, welche für jeden Aufruf mit der selben URI ein neues Objekt, statt dem selben zurück gibt (vgl. *Representational State Transfer* 2022). 16

prep Mit der „prep“ Dependency, kurz für „prepositional modifier“(Choi und Palmer 2012, S. 38), werden Token annotiert, welche „eine Präpositionsphrase sind, die die Bedeutung des Heads modifizieren“(Choi und Palmer 2012, S. 38). 15, 16

punct Mit der „punct“ Dependency, kurz für „punctuation“(Choi und Palmer 2012, S. 40), werden „jegliche [Tokens der] Zeichensetzung“(Choi und Palmer 2012, S. 40) annotiert. 15

Relation Nach dem Duden ist eine Relation eine „Beziehung, in der sich [zwei] Dinge, Gegebenheiten, Begriffe vergleichen lassen oder [wechselseitig] bedingen“(*Duden online: Relation* 2022). Im Kontext dieser Arbeit handelt es sich um Relationen, welche die räumlichen Gegebenheiten zwischen zwei Entitäten beschreiben.. 1, 2, 4, 6, 9, 10, 13, 15, 16, 20, 22–28, 30, 39, 40

Request Body Beinhaltet die für das Ansprechen der API Schnittstelle benötigten Informationen. 16

REST REST, kurz für „Representational State Transfer“, ist ein „Paradigma für die Softwarearchitektur von verteilten Systemen, insbesondere für Webservices“ (*Representational State Transfer* 2022). 16, 19

ROOT Mit der „ROOT“ Dependency werden Token annotiert, die „Wurzeln eines Baumes, welche nicht von anderen Knoten [bzw. Token] abhängig sind“ (Choi und Palmer 2012, S. 40) darstellen. 15, 16

Salesforce Ein Customer Relationship Management System. 25

Tagger Anwendung zu automatischen Annotation von Texten. 14, 15

Tail Endpunkt einer Relation. 6, 23

Text2Scene Text2Scene im Sinne einer Anwendung für die Erzeugung einer Szene aus Text. 24

Token Wort beziehungsweise Satzzeichen. 14, 26

TP TP, kurz für „True Positive“, beschreibt einen vom Modell richtig positiv ausgewerteten Ausgang. 29

unüberwachte Lernen Ein Algorithmus des Maschinellen Lernens, welches Muster in nicht annotierten Daten erkennt (vgl. Khanum u. a. 2015). 9

VP Mit dem „VP“ POS, werden Token annotiert, welche Verbalphrasen sind (vgl. *SpaCy POS Tagging Glossar* 2022). 42

WDT Mit dem „WDT“ POS, werden Token annotiert, welche Determiner sind und mit „wh“ beginnen (vgl. *SpaCy POS Tagging Glossar* 2022). 42

WP Mit dem „WP“ POS, werden Token annotiert, welche Personalpronomen sind und mit „wh“ beginnen (vgl. *SpaCy POS Tagging Glossar* 2022). 42

Literatur

Chang, Angel X., Thomas A. Funkhouser u. a. (2015). „ShapeNet: An Information-Rich 3D Model Repository“. In: *CoRR* abs/1512.03012. arXiv: 1512.03012. URL: http://arxiv.org/abs/1512.03012.

Chang, Angel X., Will Monroe, Manolis Savva, Christopher Potts und Christopher D. Manning (2015). „Text to 3D Scene Generation with Rich Lexical Grounding“. In: *CoRR* abs/1505.06289. arXiv: 1505.06289. URL: http://arxiv.org/abs/1505.06289.

Chen, Kevin u. a. (2018). „Text2Shape: Generating Shapes from Natural Language by Learning Joint Embeddings“. In: *CoRR* abs/1803.08495. arXiv: 1803.08495. URL: http://arxiv.org/abs/1803.08495.

Choi, Jinho D. und Martha Palmer (2012). „Guidelines for the CLEAR Style Constituent to Dependency Conversion“. URL: https://www.researchgate.net/publication/324940566_Guidelines_for_the_CLEAR_Style_Constituent_to_Dependency_Conversion.

ClearNLP Dependency Labels (2022). URL: https://github.com/clir/clearnlp-guidelines/blob/master/md/specifications/dependency_labels.md (besucht am 19. 08. 2022).

Deklarative Programmierung (2022). URL: https://de.wikipedia.org/wiki/Deklarative_Programmierung (besucht am 19. 08. 2022).

Duden online: Entität (2022). URL: https://de.wikipedia.org/wiki/Softwarearchitektur (besucht am 19. 08. 2022).

Duden online: Relation (2022). URL: https://www.duden.de/rechtschreibung/Relation (besucht am 19. 08. 2022).

Flask Dokumentation (2022). URL: https://flask.palletsprojects.com/en/2.2.x/ (besucht am 19. 08. 2022).

Goodfellow, Ian u. a. (2014). „Generative Adversarial Nets“. In: *Advances in Neural Information Processing Systems*. Hrsg. von Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence und K.Q. Weinberger. Bd. 27. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

Goodfellow, Ian J. u. a. (2014). *Generative Adversarial Networks*. DOI: 10.48550/ARXIV.1406.2661. URL: https://arxiv.org/abs/1406.2661.

Ide, Nancy und James Pustejovsky, Hrsg. (2017). *Handbook of Linguistic Annotation*. Dordrecht: Springer. ISBN: 978-94-024-0879-9. DOI: 10.1007/978-94-024-0881-2.

Kahn, Kenneth Michael (1979). „Creation of computer animation from story descriptions“. URL: https://dspace.mit.edu/handle/1721.1/16012.

Khanum, Memoona, Tahira Mahboob, Warda Imtiaz, Humaraia Abdul Ghafoor und Rabeea Sehar (Juni 2015). „Article: A Survey on Unsupervised Machine Learning Algorithms for Automation, Classification and Maintenance“. In: *International Journal of Computer Applications* 119.13. Full text available, S. 34–39.

Language resource management — Semantic annotation framework — Part 7: Spatial information (2020).

Logdatei (2022). URL: https://de.wikipedia.org/wiki/Logdatei (besucht am 19. 08. 2022).
Lv, Teng, Ping Yan und Weimin He (Aug. 2018). „Survey on JSON Data Modelling“. In: *Journal of Physics: Conference Series* 1069, S. 012101. DOI: 10.1088/1742-6596/1069/1/012101.
Ma, Rui u. a. (2018). „Language-driven synthesis of 3D scenes from scene databases“. In: *SIGGRAPH Asia 2018 Technical Papers*. ACM, S. 212.
Marneffe, Marie-Catherine de und Christopher D. Manning (2016). *Stanford typed dependencies manual*. URL: http://nlp.stanford.edu/software/dependencies_manual.pdf (besucht am 19. 08. 2022).
Natural Language Processing (NLP) (2022). URL: https://www.ibm.com/cloud/learn/natural-language-processing (besucht am 19. 08. 2022).
Precision and recall (2022). URL: https://en.wikipedia.org/wiki/Precision_and_recall (besucht am 19. 08. 2022).
Programmierschnittstelle (2022). URL: https://de.wikipedia.org/wiki/Programmierschnittstelle (besucht am 19. 08. 2022).
Reed, Scott u. a. (20–22 Jun 2016). „Generative Adversarial Text to Image Synthesis“. In: *Proceedings of The 33rd International Conference on Machine Learning*. Hrsg. von Maria Florina Balcan und Kilian Q. Weinberger. Bd. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, S. 1060–1069. URL: https://proceedings.mlr.press/v48/reed16.html.
Representational State Transfer (2022). URL: https://de.wikipedia.org/wiki/Representational_State_Transfer (besucht am 19. 08. 2022).
Salesforce Flow Builder (2022). URL: https://help.salesforce.com/s/articleView?id=sf.flow_builder.htm&type=5 (besucht am 19. 08. 2022).
Simmons, Robert F. (1975). *The clowns microworld*. Association for Computational Linguistics. URL: https://aclanthology.org/T75-2004.pdf.
Softwarearchitektur (2022). URL: https://de.wikipedia.org/wiki/Softwarearchitektur (besucht am 19. 08. 2022).
SpaCy Homepage (2022). URL: https://spacy.io/ (besucht am 19. 08. 2022).
SpaCy POS Tagging Glossar (2022). URL: https://github.com/explosion/spaCy/blob/master/spacy/glossary.py (besucht am 19. 08. 2022).
StolperwegeVR (2022). URL: https://github.com/texttechnologylab/StolperwegeVR.git/ (besucht am 05. 07. 2022).
Tan, Fuwen, Song Feng und Vicente Ordonez (2018). „Text2Scene: Generating Abstract Scenes from Textual Descriptions“. In: *CoRR* abs/1809.01110. arXiv: 1809.01110. URL: http://arxiv.org/abs/1809.01110.
Universal Dependencies: POS - DET (2022). URL: http://universaldependencies.org/docs/u/pos/DET.html (besucht am 19. 08. 2022).
Was ist ein Machine Learning-Modell? (2022). URL: https://docs.microsoft.com/de-de/windows/ai/windows-ml/what-is-a-machine-learning-model (besucht am 19. 08. 2022).
What Life Means To Einstein (1929). URL: http://www.saturdayeveningpost.com/wp-content/uploads/satevepost/what_life_means_to_einstein.pdf (besucht am 19. 08. 2022).
Winograd, Terry (1972). *Understanding natural language*. Academic Press.

Zhang, Han u. a. (2017). „StackGAN: Text to Photo-Realistic Image Synthesis with Stacked Generative Adversarial Networks“. In: *2017 IEEE International Conference on Computer Vision (ICCV)*, S. 5908–5916. DOI: 10.1109/ICCV.2017.629.

Zhang, Zizhao, Yuanpu Xie und Lin Yang (2018). *Photographic Text-to-Image Synthesis with a Hierarchically-nested Adversarial Network*. DOI: 10.48550/ARXIV.1802.09178. URL: https://arxiv.org/abs/1802.09178.