Die Python-Bibliotheken NLTK und SpaCy stellen entsprechende wortbasierte Tokenizer für eine möglichst schnelle Implementierung bereit. Beide sind überdies in einer für die deutsche Sprache ausgereiften Version verfügbar. Oftmals werden hierbei weitere Funktionalitäten mitgeliefert, darunter meist das Entfernen sprachbezogener Stoppwörter, die Lemmatisierung oder auch das Part-of-Speech-Tagging [Bird et al., 2009, S. 111].

3.2 Word Embeddings

Algorithmen können Texte nicht in ihrer Rohform verarbeiten. Texte bedürfen einer geeigneten mathematischen Form. Word Embeddings überführen Texte oder ganze Korpora hierfür in einen Vektorraum (engl. Vector Space), um Wörter syntaktisch, semantisch und insbesondere untereinander in Kontext zu bringen. Dabei wird ein Wortschatz benutzt, welcher die entsprechenden Vektoren, bestehend aus eindeutig kodierbaren ganzzahligen Werten, aufbaut [Karani, 2018]. Die Ableitung der Vektoren aus den Textdaten wird auch als Feature Extraction oder Feature Encoding bezeichnet. Insgesamt befindet man sich hier im Bereich des Language Modeling [Brownlee, 2019].

Word Embeddings werden üblicherweise noch vor der Entwicklung der ursprünglich anvisierten NLP-Aufgabe trainiert, weshalb ihnen ein unmittelbarer qualitativer Einfluss zugesprochen wird. Die entstehenden Modelle sind in der Folge schnell implementierbar. Weiterhin sind sie skalierbar, da sie als Grundlage verschiedenster nachgelagerter NLP-Aufgaben eingesetzt werden können [Nitsche, 2019]. Word Embeddings können durch verschiedene mehr oder minder komplexe Ansätze realisiert werden. Diese werden nachfolgend vorgestellt, wobei stets verdeutlicht wird, wie der entsprechende Vektorraum aufgebaut und in Kontext gebracht wird. Dabei wird außerdem deutlich, dass sich die verschiedenen Ansätze nicht für jede NLP-Aufgabe eignen, sondern sie vielmehr einschränken. Obgleich nicht alle Ansätze für die ATS relevant sein werden, sind sie als Grundlage zu verstehen.

3.2.1 One-Hot-Encoding

One-Hot-Encoding (OHE) ist einer der einfachsten Ansätze, um Texte in einen Vektorraum einzubetten. Dabei werden allgemein kategorische Variablen in ein numerisches und somit mathematisch verarbeitbares Format gebracht [Karani, 2018]. Seien nun zwei gleichbedeutende Sätze gegeben. Eben jene Sätze sowie der entsprechende Wortschatz und die binär kodierten Vektoren sind in Abbildung 3.2 ersichtlich. Die Vektoren sind hierbei als Matrix zusammengefasst, wobei die Zeilen und Spalten anhand der Anfangsbuchstaben der Wörter kenntlich gemacht sind.

Versucht man nun, diese Vektoren in einem Vektorraum zu visualisieren, dann entspricht jeder Vektor einer eigenen Dimension. Dabei wird allerdings klar, dass keine dimensionsübergreifenden Projektionen existieren [Karani, 2018]. Dies bedeutet, dass die Wörter *gutes* und *schönes* genauso verschieden sind, wie die Wörter *heute* und *ist*. Dies ist offensichtlich falsch. OHE ist dennoch als Grundlage zu verstehen, wobei etwaige Probleme innerhalb der nachfolgenden Ansätze weiterbehandelt werden.

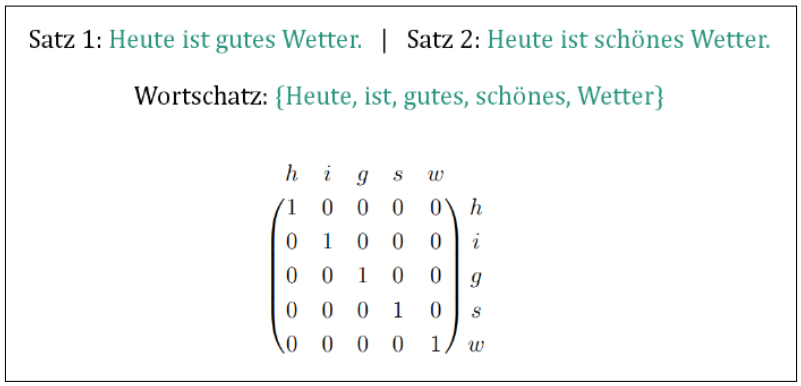

Abbildung 3.2: One-Hot-Encoding mit zwei beispielhaften Sätzen.

3.2.2 Bag-of-Words

Bag-of-Words (BOW) realisieren die Feature Extraction, indem entsprechende Modelle das Aufkommen von in einem Wortschatz definierter Wörter über eine Vielzahl von Texten zählen. Dabei ist allerdings ein gewisser Informationsverlust zu erwarten, da beispielsweise die Reihenfolge der Wörter nicht berücksichtigt wird [Raschka et al., 2019, S. 262]. Tabelle 3.1 zeigt beispielhaft eine Matrix eines solchen BOW-Modells.

In den Zeilen sind die betrachteten Texte indiziert, in den Spalten hingegen die Wörter des frei erfundenen Wortschatzes. Die Matrix hat demnach eine daran orientierte fixe Größe. In den Zellen ergeben sich somit die Häufigkeiten der Wörter, bezogen auf ihr Aufkommen in den einzelnen Texten [Brownlee, 2019].

ID	**Heute**	**ist**	**schönes**	**Wetter**
Text 1	2	8	1	3
Text 2	1	3	0	2
Text 3	0	4	1	4

Tabelle 3.1: Bag-of-Words mit einem beispielhaften Wortschatz [Huilgol, 2020].

Nachteilig ist dieser Ansatz hinsichtlich der ATS insbesondere dadurch, dass die Reihenfolge der Wörter nicht berücksichtigt wird. Wie in Tabelle 3.1 ersichtlich wurde, lassen sich aus der Matrix keine Informationen über die Semantik oder Grammatik rekonstruieren. Aus technischer Sicht würde die Matrix bei steigender Wortschatzgröße nicht nur mitwachsen, sondern zudem viele Null-Einträge enthalten [Huilgol, 2020]. Vorteilig ist dieser Ansatz hingegen für eher schlichtere NLP-Aufgaben. Hier sei die Klassifikation von Dokumenten als mögliches Einsatzgebiet genannt.

3.2.3 Skip-Gram-Model

Ein weiterer frequenzbasierter Ansatz besteht in sogenannten Skip-Gram-Modellen. Diese unterliegen der Annahme, dass sich der Kontext eines gegebenen Wortes in Form von einer Textsequenz generieren lässt. Sei hierfür nun beispielhaft und gleichermaßen abstrakt die Sequenz $\{a,\ b,\ c,\ d,\ e\}$ gegeben. Zudem sei c das Zielwort und die lokale Fenstergröße zwei. Ein Skip-Gram-Modell modelliert die bedingten Wahrscheinlichkeiten für die vor- und nachstehenden Kontextwörter [Zhang et al., 2020, S. 640]. Hierfür gilt die folgende Formel:

$$P(a, b, d, e \mid c)$$

Gemäß der weitergehenden Annahme, die Kontextwörter ließen sich auf Grundlage eines gegebenen Zielwortes unabhängig voneinander generieren, kann die Formel wie folgt umgeschrieben werden:

$$P(a \mid c) \cdot P(b \mid c) \cdot P(d \mid c) \cdot P(e \mid c)$$

Sei darüber hinaus abstrakter Wortschatz gegeben. Dabei erfordert jedes darin enthaltene Wort zwei mehrdimensionale Vektoren. Einen, um das Wort als Zielwort zu evaluieren, und einen, um das Wort in den unterschiedlichen Kontexten einzuordnen. Mithilfe dieser Vektoren können die bedingten Wahrscheinlichkeiten des entsprechenden Modells trainiert werden. Autark ist jedoch auch dieser Ansatz ungeeignet für die ATS [Zhang et al., 2020, S. 641].

3.2.4 Word2Vec

Der Ansatz des Word2Vec (W2V) kombiniert BOW-Modelle mit Skip-Gram-Modellen, um die Nachteile des OHE weitergehend aufzuarbeiten. Dabei werden BOW-Modelle jedoch in kontinuierlicher Form genutzt. Diese funktionieren in umgekehrter Weise zu den Skip-Gram-Modellen. Damit ist folglich eine beidseitige Herangehensweise möglich. Der Kontext kann also aus einem gegebenen Zielwort und das Zielwort wiederum aus einem gegebenen Kontext ermittelt werden [Zhang et al., 2020, S. 644].

Die entsprechenden bedingten Wahrscheinlichkeiten können gemäß der nachstehenden Formeln modelliert werden. Dabei werden Skip-Gram-Modelle (erstgenannt) den BOW-Modellen (zweitgenannt) anhand des oben genannten Beispiels gegenübergestellt.

$$P(a, b, d, e \mid c) \qquad P(c \mid a, b, d, e)$$

W2V-Modelle können mithilfe neuronaler Netze trainiert werden. Dies befähigt sie, Zusammenhänge zwischen Wörtern zu erlernen. Hierbei werden Distanzen minimiert, die aus den zuvor beschriebenen Vektoren hervorgehen. Dieser Ansatz eignet sich in der Folge etwa dazu, Synonyme für gegebene Wörter zu bestimmen. ATS-Modelle, welche den W2V-Ansatz verfolgen, konnten sich in der Vergangenheit bereits in akzeptablem Maße beweisen [Karani, 2018].

Abbildung 3.3 zeigt die beispielhaften Projektionen von etwa 10.000 Wörtern, welche mit dem Embedding Projektor von TensorFlow und einer entsprechenden Hauptkomponentenanalyse generiert wurden. Dabei ist das Ergebnis der Suche nach dem Wort *play* zu sehen. Es ist erkennbar, dass bedeutungsähnliche Wörter kürzere Distanzen aufweisen.

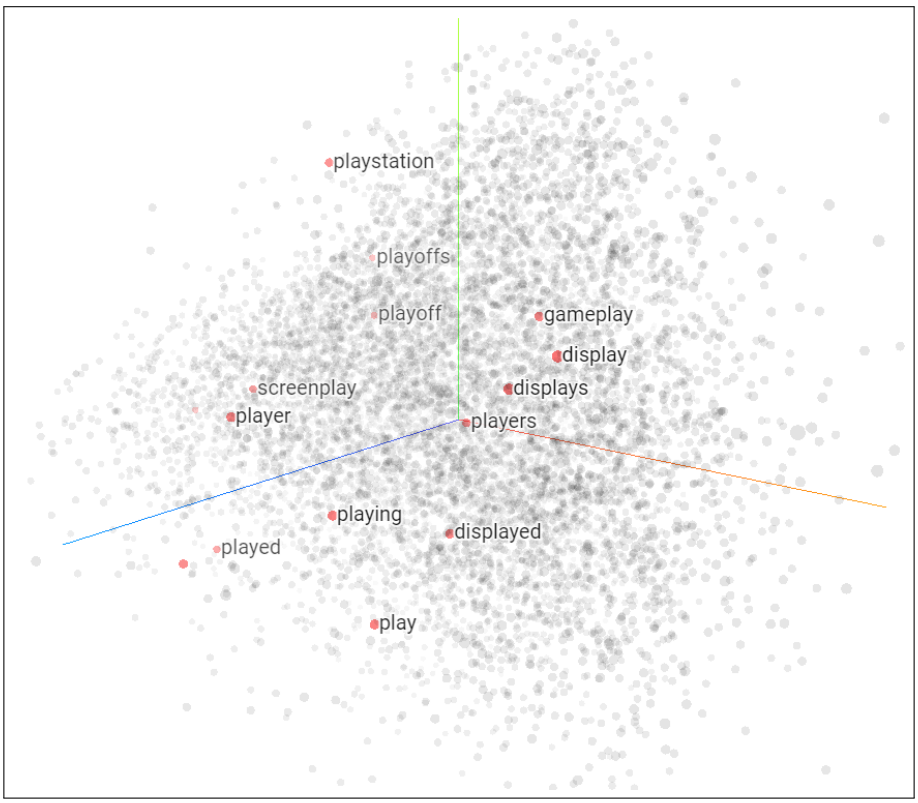

Abbildung 3.3: Word2Vec mit dem Embedding Projector von TensorFlow.

3.2.5 Byte-Pair-Encoding

Byte-Pair-Encoding (BPE) ist ein Algorithmus zur Datenkompression. Hierfür werden zusammenhängende Bytes mit einem Byte ersetzt, welches nicht in den sonstigen Daten auftritt [Nitsche, 2019, S. 24]. Sei folgendes Beispiel gegeben:

$$aaabcaacab = \boldsymbol{ZY}c\boldsymbol{Z}c\boldsymbol{Y}$$

$$\text{mit } \boldsymbol{Z} = aa \text{ und } \boldsymbol{Y} = ab$$

BPE eignet sich insbesondere dafür, seltene Wörter zu berücksichtigten. In diesen verbirgt sich mitunter eine nicht zu vernachlässigende Bedeutung. Aufgrund der kodierenden Funktion wird BPE auch als Subword Embedding bezeichnet, während die komprimierende Funktion kleinere Modellgrößen verursacht [Nitsche, 2019, S. 24]. ATS-Modelle, welche den W2V-Ansatz verfolgen, konnten sich in der Vergangenheit ebenfalls in akzeptablem Maße beweisen.

3.2.6 GloVe

Global Vectors for Word Representation (GloVe) ist ein weiterer Algorithmus zur Einbettung von Texten in einen Vektorraum, welcher syntaktische und semantische Bedeutungen repräsentiert [Penninglon et al., 2014, S. 1].

Hierfür wird sich einer Matrix bedient, welche das gemeinsame Aufkommen der im Korpus enthaltenen Wörter paarweise gegenüberstellt. Sie wird daher auch als Word-Word-Co-Occurrence-Matrix bezeichnet und besitzt eine maximale Komplexität von $O(|V|^2)$, wenn V die Wortschatzgröße ist. GloVe sieht dabei vor, die Matrix unter Nutzung lokaler Kontextfenster global zu faktorisieren [Penninglon et al., 2014, S. 2].

Hierbei werden sehr große Matrizen durch mehrere niederrangige Matrizen approximiert. Dies ermöglicht es, die verborgenen statistischen Informationen des zugrundeliegenden Korpus anhand linearer Substrukturen zu explorieren und letztlich zu extrahieren. Dabei werden lokale Kontextfenster verwendet, wie sie bei verschiedenen oben genannten Ansätzen bereits beschrieben wurden, beispielsweise bei W2V [Nitsche, 2019, S. 24].

GloVe ist praktisch als vortrainiertes Modell verfügbar. Dies basiert auf vier verschiedenen Korpora mit insgesamt etwa 55 Milliarden Token und einer durchschnittlichen Wortschatzgröße von etwa 1,5 Millionen Token. Im Training werden hierbei in der Matrix nur von Null verschiedene Werte berücksichtigt. Die mittlere quadratische Abweichung wird zudem als Fehlerfunktion ausgewählt. Das initiale Training ist zwar sehr rechenintensiv, dafür allerdings im Sinne des TL a priori nur einmalig auszuführen. NLP-Aufgaben, darunter auch die ATS, profitierten stark vom Erfolg des GloVe-Ansatzes. Trotzdem ist der Einsatz stets sorgfältig zu evaluieren, da innerhalb der modellinternen Vorverarbeitung die zugrundeliegenden Korpora beispielsweise auf Kleinschreibung forciert wurden. Dies müsste bei nachstehenden NLP-Aufgaben ebenfalls berücksichtigt werden [Penninglon et al., 2014, S. 6-9].