




Methodenbeitrag: word2vec

Mareike Schumacher  ¹

1. Universität Regensburg

forTEXT

Thema:	word2vec	DOI:	10.48694/fortext.3815
Jahrgang:	1	Ausgabe:	10
Erscheinungsdatum:	30-10-2024	Erstveröffentlichung:	2023-04-19 auf fortext.net
Lizenz:			open  access

Allgemeiner Hinweis: Rot dargestellte **Begriffe** werden im Glossar am Ende des Beitrags erläutert. Alle externen Links sind auch am Ende des Beitrags aufgeführt.

1. Definition

word2vec ist eine computergestützte Methode (vgl. **Machine Learning**), um Ähnlichkeiten zwischen Wörtern aufgrund ihrer kontextuellen Merkmale numerisch zu erfassen (vgl. **Distant Reading**). Am häufigsten wird sie zur Analyse der semantischen Verbindungen zwischen Wörtern in einem Textkorpus (vgl. **Korpus**) eingesetzt. Dem Verfahren liegt eine Beobachtung über den Gebrauch von Wörtern in unserer Alltagssprache zugrunde: Semantisch ähnliche Wörter treten in ähnlichen Kontexten auf (die fünf ähnlichsten Wörter zum Wort „Frau“ im dProse-Korpus: „Tante“, „Witwe“, „Dame“, „Freundin“, „Jungfer“ - vgl. Gius, Guhr und Uglanova (2021) für die Beschreibung des Korpus). Das Vorkommen eines Wortes kann demnach anhand seiner Kontexte (d.h. anhand seiner unmittelbaren Nachbarschaften in einem Satz) vorhergesagt werden, und umgekehrt. In der Linguistik ist diese Eigenschaft der natürlichen Sprachen als distributionelle Hypothese (Firth 1957) bekannt.

2. Anwendungsbeispiel

Angenommen, Sie interessieren sich für fiktionale Naturräume in der deutschen Literatur der Romantik (ca. 1790er bis 1830er) und möchten diese von Darstellungen in der Epoche des Naturalismus (1880 – 1900) abgrenzen. Ihr Ziel ist es, eine Vorstellung davon zu bekommen, wie ein bestimmter Begriff in der jeweiligen literarischen Strömung konzeptualisiert wurde. Mithilfe von word2vec-Verfahren machen Sie semantische Verbindungen innerhalb des ganzen Korpus sichtbar und bekommen dadurch die Möglichkeit, beliebige Begriffe zu analysieren und miteinander zu vergleichen. Von besonderem Interesse ist dabei die Analyse der Verhältnisse zwischen den Wörtern, in welchem Kontext sie stehen und welche Bedeutungen dabei entstehen.

3. Literaturwissenschaftliche Tradition

In den nicht-digitalen, traditionellen Literaturwissenschaften gibt es keine direkten methodologischen Äquivalente zum word2vec-Verfahren. Von der zu lösenden Aufgabe ausgehend, könnte word2vec aber der Gruppe der inhaltsanalytischen Methoden zugeordnet werden.

Alle Techniken der Inhaltsanalyse haben gemeinsam, dass sie erläuternd sind und von einer Fragestellung geleitet werden:

„Wo Literaturwissenschaft einzelne Texte analysiert und erst recht, wo sie in diversen Forschungsvorhaben eine große Zahl von Texten einbezieht, untersucht sie trotz des hermeneutischen und strukturalistischen Postulats, dass sich die Bedeutung von Textteilen nur unter Berücksichtigung des ganzen Textzusammenhangs erkennen lässt, Texte explizit oder implizit immer in Abhängigkeit von vorab gestellten Fragen [...]“, so Thomas Anz (2007).

Den ganzen Textzusammenhang stets mitzudenken, ist kein einfaches Unterfangen. Das word2vec-Verfahren kann dabei unterstützen, die Beziehungen im Text und zwischen Text und Kontext mit einzubeziehen. Beim word2vec kann jedes einzelne Wort eines großen Korpus als Kontextinformation berücksichtigt werden. Das Ideal der Berücksichtigung möglichst vieler Kontextinformationen wird beim word2vec-Verfahren in einem Ausmaß umgesetzt, das von einem Menschen nicht geleistet werden kann. Gleichzeitig kann das Verfahren aber nur Kontextinformationen einbeziehen, die explizit (d.h. in Form der einzelnen Wörter) im Korpus enthalten sind. Weltwissen oder auch nur Informationen, die außerhalb des Textkorpus liegen, bleiben unberücksichtigt. Ebenso muss bedacht werden, dass alle für das Training des Verfahrens genutzten Texte in die Analyse mit einfließen. Ein Hauptunterschied zwischen den Methoden der traditionellen Literaturwissenschaften und dem word2vec-Ansatz besteht im Umgang mit den Textdaten. Bei traditionell-analogen literaturwissenschaftlichen

Ansätzen werden keine neuen Textstrukturen erschaffen, sondern Deutungsmuster bzw. Metastrukturen. Beim word2vec-Verfahren wird aus der betrachteten Textsammlung ein neues Objekt, eine neue Datenstruktur. Dabei können neben Positionen von Wörtern im Text auch weitere Informationen (vgl. **Metadaten**) wie Wortform oder Lemmata berücksichtigt werden.

Sie haben es hier dann nicht mehr mit einem einzelnen Wort, Segment oder Text zu tun, sondern mit einem geordneten Raum der semantischen Relationen. Aus diesem Grund bezeichnet Dobson (2021) Algorithmen, in denen die Idee eines Vektorraums – also in diesem Falle einer Menge von Wörtern, die in Vektoren und damit in Zahlenwerte umgerechnet wurden – implementiert sind, in einem hermeneutischen Sinne als interpretativ. Allerdings wird die hermeneutische Interpretation häufig mit Intentionalität und autororientierten Ansätzen verbunden (Anz 2007, 305ff). Bei word2vec spielt eine Autorinstanz hingegen keine Rolle, relevant sind reine Textdaten (vgl. **Reintext-Version**).

Zwar ist es theoretisch möglich, word2vec auch auf einen einzelnen Text anzuwenden oder kleine Korpora zu betrachten (Grayson u. a. 2016; Wohlgenannt, Chernyak und Ilvovsky 2016), üblicher aber ist die Verarbeitung größerer bis hin zu sehr großen Textkorpora (vgl. **Distant Reading**). Dadurch werden diachrone bzw. relationale Betrachtungsansätze besonders gut unterstützt. Ein besonders anschauliches Beispiel für diachrone Betrachtungsansätze demonstrieren Jurafsky und Martin (2021) im Kapitel 6.10 von *Speech and Language Processing*. Darin betrachten sie die Wörter „gay“, „broadcast“ und „awful“. Wie unser Anwendungsbeispiel zeigt, ist es möglich, ein bestimmtes Phänomen (wie den Naturraum) in einem Korpus (z.B. Erzähltexte der Romantik) mit der Darstellung desselben Phänomens in einem anderen Korpus (z.B. Erzähltexte des Naturalismus) zu vergleichen. Das Beispiel wurde entlehnt von Gius und Ugianova (2023).

Eine sowohl in den Sozial- als auch den Kultur- und Literaturwissenschaften häufig adaptierte Methode (vgl. **Domäneadaption**), die sehr gut in Verbindung mit word2vec eingesetzt werden kann, ist die Diskursanalyse nach Foucault. Auch hier sind Relationen und intertextuelle Verweise von besonderer Bedeutung (Köppe und Winko 2013, 350). Dass und wie word2vec innerhalb einer sozialwissenschaftlichen Diskursanalyse eingesetzt werden kann, zeigt Lindgren (2020). Ein literaturwissenschaftliches Beispiel gibt Heuser (2017a) mit seiner semantischen Analyse von Texten des 18. Jahrhunderts.

Von einem weiten Intertextualitätsbegriff ausgehend kann word2vec als Methode zur Intertextualitätsanalyse eingesetzt werden. Zwar kann auch hier keine Begrifflichkeit und / oder Methodik unangepasst übertragen werden, da word2vec-Verfahren zwar ähnliche Wortgebräuche in Texten aufspüren, nicht aber zwischen einem intertextuellen Verweis und Textelementen ohne Verweischarakter unterscheiden können. Das von Burghardt und Liebl und Burghardt (2020) entwickelte Tool *The Vectorian* zeigt aber, dass es u.a. auf Basis von einer Implementierung der word2vec-Weiterentwicklung *fastText* (Bojanowski u. a. 2017) möglich ist, für bestimmte Formulierungen eine Ähnlichkeit mit Textstellen aus Shakespeare-Texten zu errechnen und dann basierend auf einem Schwellenwert für die Ähnlichkeit in einem Korpus nach möglichen Intertextualitäten zu suchen (Liebl und Burghardt 2020). Genau genommen, kann in dem Tool *The Vectorian* eine Suche entweder auf Basis von *fastText* oder einer anderen Form von Word Embeddings (wnet2vec) oder aber mit einer Kombination der beiden vorgenommen werden (Burghardt und Liebl 2020). Je ähnlicher eine Formulierung denen von Shakespeare ist, desto wahrscheinlicher ist es, dass es sich um einen intertextuellen Verweis handelt (Burghardt und Liebl 2020; Liebl und Burghardt 2020).

Weitere literaturwissenschaftlich relevante Traditionslinien, in die word2vec-Analysen integriert werden können, sind die ebenfalls häufig in Form von Text-Kontext-Studien umgesetzten Ansätze der Kulturwissenschaften und die eng verwandten Gender Studies. Es ist z.B. möglich, mit word2vec die Darstellung bestimmter kultureller Phänomene in einem Korpus im Vergleich mit einem anderen herauszuarbeiten, wie in unserem Anwendungsbeispiel verdeutlicht. Auch Fragen nach der spezifischen Darstellung bestimmter Genderkategorien können mit word2vec betrachtet werden (Schmidt 2015a).

Zusammenfassend lässt sich festhalten, dass das word2vec-Verfahren in den literaturwissenschaftlichen Workflow unterschiedlichster Traditionslinien integriert werden kann. Außerdem wird das Verfahren häufig als eine Art Hilfstechnologie zur Verbesserung anderer digitaler Methoden eingesetzt (Schöch 2023, 12). Eine word2vec-Komponente wurde z.B. schon sinnvoll angewendet in der Sentimentanalyse (Kocher und Savoy 2018; Al-Saqqa und Awajan 2019; Jacobs 2019; Schöch 2023, 13), der Autorschaftsattributions (Kocher und Savoy 2018), der Named Entity Recognition (Kocher und Savoy 2018), bei der Extraktion sozialer Netzwerke (Wohlgenannt, Chernyak und Ilvovsky 2016) oder beim Topic Modeling (Schöch 2023, 12–13).

4. Diskussion

Da mithilfe von word2vec Texte in ihrer Linearität aufgebrochen und in eine grundlegend andere Form gebracht werden, benötigt die Interpretation von word2vec-Modellen auch eine andere Form der Lesekompetenz als die in den traditionellen Literaturwissenschaften vorherrschende Textinterpretationsmethodik. Einige Schwierigkeiten bei der Betrachtung von word2vec-Modellen wie z.B. die Auswahl der betrachteten Dimensionen und die damit verbundene Komplexitätsreduktion beschreibt z.B. Schöch (2023, 7). Substanziell ist dabei die Repräsentation der Ursprungstexte. Es handelt sich darum um ein Verfahren, das relativ weit von nicht-digitalen Textzugängen abweicht. Hinzu kommt, dass es sich bei Word Embeddings – der Einbettung von Wörtern in

einen Vektorraum, also der Zuordnung einzelner Wörter zu Vektoren –, zu denen auch word2vec gehört, um eine aktuell hochdynamisch weiterentwickelte Forschungsmethode handelt. Derzeit gibt es neben der oben bereits erwähnten word2vec-Weiterentwicklung *fastText*, auch Word-Embeddings, wie z.B. BERT (Bidirectional Encoder Representations from Transformers), die stärker kontextbasiert ausgerichtet sind und einem Wort abhängig von dessen Kontext unterschiedliche Vektoren zuweisen können (Ehrmanntraut u. a. 2021, 16). Eine Einarbeitung in die Grundlagen vektorbasierter Einbettungsverfahren kann dennoch lohnend sein, da es sich um eine sehr mächtige und – wie oben dargelegt – dehnbare Methode handelt, mit der an viele Traditionslinien der Literaturwissenschaften angeknüpft werden kann. Außerdem ist es möglich, bei vergleichenden Studien auf vortrainierte Vektormodelle zurückzugreifen und diese z.B. mit eigenen word2vec-Modellen zu vergleichen (Ehrmanntraut u. a. 2021). Vektormodelle können als Datenpublikation veröffentlicht werden, um dann von anderen Forschenden nachgenutzt zu werden. Es handelt sich dabei um abgeleitete Textformate (Schöch u. a. 2020). Darüber hinaus können Sie von fertig trainierten Sprachmodellen auch profitieren, wenn diese in anderen Tools eingesetzt werden. Ein grundlegendes Verständnis von word2vec und anderen Word-Embedding-Verfahren kann dazu beitragen, die Funktionsweisen dieser Tools besser zu verstehen und einem sogenannten *Black Boxing* entgegenwirken (Röhle 2012) zu unterschiedlichen in den digitalen Geisteswissenschaften relevanten Formen des *Black Boxings*.

Einige fundamentale Vorteile der Methode gehen auf die besondere Repräsentationsform des Korpus zurück. In diachronen Untersuchungen können z.B. semantische Felder auf konzeptuelle Verwendungen einzelner oder zusammenhängender Begriffe untersucht werden. Statt einzelne Wörter zu betrachten, können ganze Netzwerke von häufig gemeinsam verwendeten Begriffen in die Betrachtungen einbezogen werden (Wevers und Koolen 2020, 239). Steht das Wort „Natur“ in einem Korpus romantischer Texte z.B. häufig in Zusammenhang mit Wörtern wie „Kunst“, „Schönheit“, „Sinnlichkeit“ und „Phantasie“ (s.u.), so kann das auf ein Naturkonzept hindeuten, das eng mit dem der Ästhetik verknüpft ist und sich grundsätzlich von dem des Naturalismus unterscheidet. Darüber hinaus können mithilfe von word2vec semantische Cluster aufgedeckt und deren Relationen zueinander betrachtet werden (Bubenhof 2020, 586). Bei einer solchen Betrachtung kann offenbar werden, dass das semantische Feld „Natur“ in Texten der Romantik z.B. näher bei dem der „Kunst“ steht als bei dem des „Menschen“ (s.u.). Auch wenn in einer solchen Analyse nicht immer überraschende Funde gemacht werden können (Bubenhof 2020, 585), so kann sie dennoch als empirisch-quantitative Basis einer Verifizierung von Thesen gelten, die in Einzelfallstudien erarbeitet wurden.

Ein weiterer wichtiger Vorteil dieser Methode ergibt sich aus der Transformation des ganzen Vokabulars des Korpus in einen semantischen Vektorraum. Ein Vektor als mathematische Darstellung eines Wortes ermöglicht es, mit den Wörtern des Korpus genauso wie mit Zahlen zu hantieren. Wortvektoren können addiert oder subtrahiert werden, wie ein klassisches word2vec-Anwendungsbeispiel zeigt:

$v(\text{„König“}) - v(\text{„Mann“}) + v(\text{„Frau“}) = \text{„Königin“}$.

```
1 model.most_similar(positive=['frau', 'könig'], negative=['mann'])
[('königin', 0.4313160181045532),
 ('kaiser', 0.41552186012268066),
 ('giazinta', 0.3882979452610016),
 ('kaiserin', 0.37558481097221375),
 ('erwählt', 0.3678188920021057),
 ('kastilien', 0.3602077066898346),
 ('kurfürst', 0.355290997028351),
 ('albrechten', 0.35490235686302185),
 ('toskana', 0.35442689061164856),
 ('theresia', 0.351487398147583)]
```

Abb. X: Vektorarithmetik am Beispiel eines Modells zu 115 Romanen der Romantik (Schumacher, Uglanova und Gius 2022)

Diese einzigartige Funktion lässt es zu, ein neues Niveau des konzeptuellen Verständnisses bei der Analyse eines Korpus zu erreichen. Zunächst kann durch diese Vektorarithmetik anhand einfacher Beispiele, wie dem oben genannten, kontrolliert werden, ob das Modell auf Basis ausreichender Textinformationen gelernt hat, einfache semantische Konzepte abzubilden. Ist dies der Fall, so kann man versuchen, mit in weniger offensichtlichen Beziehungen zueinander stehenden Ausdrücken zu rechnen, um mehr über Konzeptualisierungen einzelner Begriffe in einem Korpus zu erfahren.

Bei der Anwendung der Methode muss vor allem der Umfang des Textkorpus berücksichtigt werden. Je mehr Daten zur Erstellung eines word2vecs-Modells genutzt werden, desto zuverlässiger sind diese. Kleine Datensätze, die typisch für geisteswissenschaftliche Forschungsansätze sind, reichen meist nicht aus (Wevers und Koolen 2020). Um ein gut interpretierbares Ergebnis zu erhalten, benötigt man ein relativ großes Korpus. Zur Orientierung: Wenn Sie mit Langtexten wie z.B. Romanen arbeiten, so scheint eine Startgröße von ca. 90–100 Texten (~ 10 Millionen Tokens (vgl. **Type/Token**)) angemessen zu sein, um interpretierbare Wortvektoren zu bekommen. Zum Vergleich: Heuser (2017b), empfiehlt ca. 30 Millionen Tokens, Siobhan Grayson mit Kolleg*innen haben word2vec-Modelle für 12 Prosatexte erstellt, was dem Umfang 1,5 Millionen Wörter entspricht (Grayson u. a. 2016). In jedem Fall sollte ein Modell zunächst getestet und das Korpus ggf. erweitert werden, bevor eine abschließende, tragfähige Interpretation entwickelt wird. Die Qualität eines word2vec-Modells hängt jedoch

nicht nur von der Größe des Datensatzes ab, sondern auch von dessen Qualität (OCR-Fehler können sich z.B. negativ auswirken) und von der Zusammensetzung des Korpus (vor allem, ob es versteckte Vorannahmen, den sogenannten Korpus-Bias gibt) (Wevers und Koolen 2020).

Ein Nachteil des Einsatzes von word2vec in literaturwissenschaftlichen Studien liegt darin, dass individuelle Konnotationen einzelner Texte verloren gehen. Stattdessen wird das zu untersuchende Phänomen auf der Makroebene abgebildet. Die Perspektive auf den Forschungsgegenstand ist also eine grundlegend andere als bei Einzeltextanalysen. Es handelt sich bei word2vec um ein **Distant Reading**-Verfahren.

5. Technische Grundlagen

Der Name der Methode – word2vec (in etwa: „Wort zu Vektor“) – deutet auf die Art und Weise hin, wie das Textkorpus transformiert wird: Die Wörter werden in Vektoren umgewandelt. Ein Vektor ist hier eine mathematische Repräsentationsform der Kontextpositionen eines Wortes. Vektoren können als Pfeile dargestellt werden, von denen zwei einen zweidimensionalen Raum ergeben, drei einen dreidimensionalen und so weiter. In einem zweidimensionalen Vektorraum bilden Zahlenpaare den Vektor, in einem dreidimensionalen Triplets und so fort. Auf Textdaten angewendet, können Vektoren z.B. die Positionen von Wörtern im Satz abbilden. Jede mögliche Position bildet dabei eine Dimension. Bei einem Satz wie „Der König starb“ gibt es demnach drei Dimensionen und jedes Wort kann wie in Abb.1 (in der Informatik als *one hot encoding* bezeichnet) dargestellt innerhalb des Koordinatensystems verortet werden.

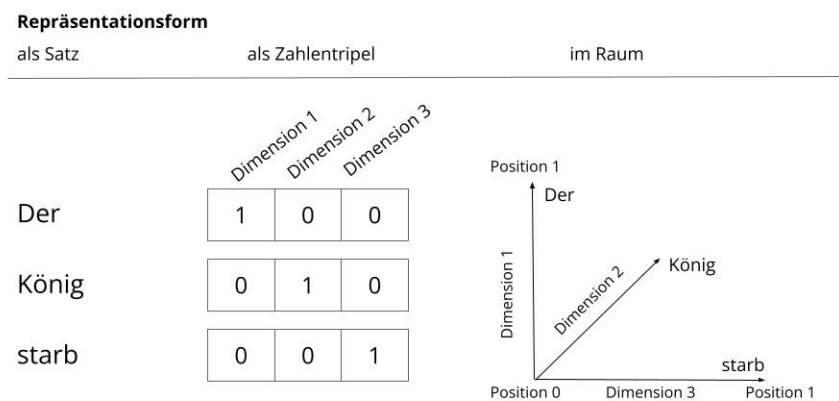


Abb. 1: Umrechnung von Wörtern in Vektoren

Ein (Wort-)Vektor ist also eine Reihe von Zahlen, in der jede Zahl für die Position des Wortes auf einer Dimensionsachse des Vektorraums steht. Beinhaltet das auf Positionen beruhende word2vec Modell nicht, wie in Abb. 1, nur einen Satz, sondern zwei Sätze, so kommen weitere Informationen hinzu und es zeigt sich, an welchen Positionen unterschiedliche Wörter stehen (vgl. Abb. 2).

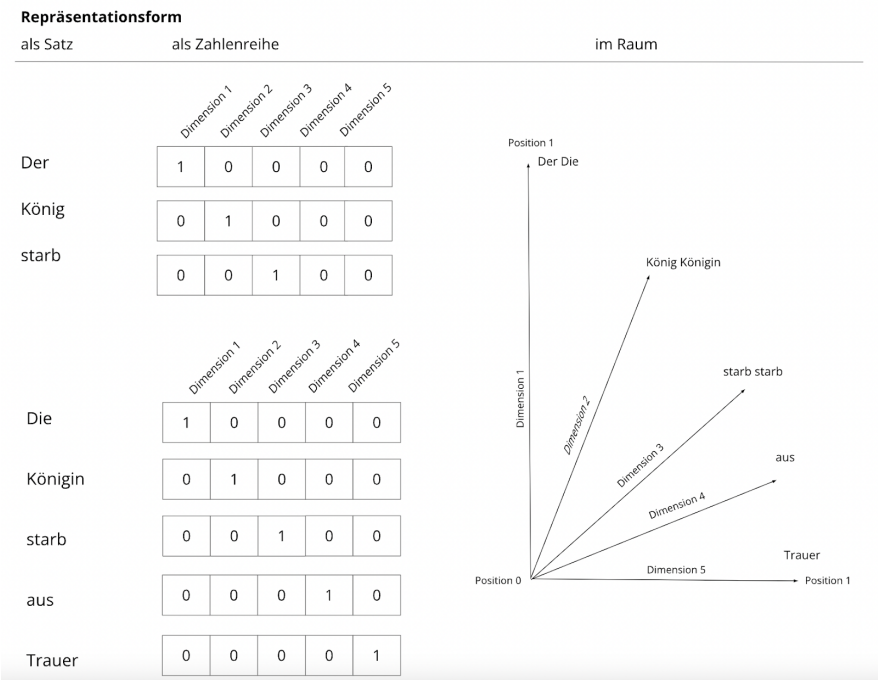


Abb. 2: Von zwei Sätzen zum Vektorraum mit sieben Dimensionen

Ein solcher auf Positionen von Wörtern im Satz beruhender Vektorraum kann sehr schnell sehr viele Dimensionen erreichen. Bei der Darstellung von word2vec-Modellen wird darum häufig eine mathematische Dimensionsreduktion angewendet, bei der die x Dimensionen des Modells auf zwei heruntergerechnet werden.

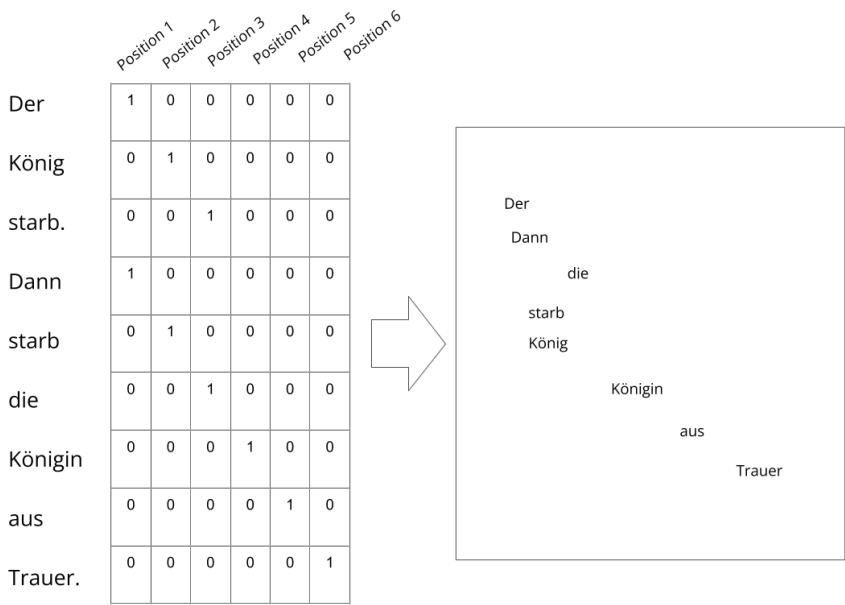


Abb. 3: Von zwei Sätzen zum zweidimensionalen Vektorraum

Word2Vec setzt die Grundannahme einer sinnhaften Nachbarschaft von Wörtern um. Das heißt, dass ähnliche Repräsentationsformen, in diesem Falle ähnliche Vektoren, für ähnliche Distributionen im Text stehen. Die distributionelle Hypothese, die die Basis des Verfahrens bildet, besagt, dass Wörter mit ähnlichen Positionen im Kontext auch ähnliche Bedeutungen haben. Das Ergebnis der Transformation von Wörtern zu Vektoren, also zu Zahlenreihen, ist ein (semantischer) Raum, in dem ähnliche Wörter räumlich nebeneinander platziert sind. Ein weiteres Konzept, das in Verbindung mit word2vec oft zusammen vorkommt, ist das des *Word Embeddings* (Worteinbettung). Word Embedding ist eine allgemeine Bezeichnung für eine Gruppe von Methoden, die auf der Transformation der (Text-)Daten in einen Vektorraum beruhen. Eine ausführliche Erklärung von Word-Embedding-Modellen findet sich in Schöch (2023).

Die Idee, dass die Bedeutung eines Wortes aus seinem Kontext abgeleitet werden kann, war seit den 30er Jahren des letzten Jahrhunderts im sprachtheoretischen Diskurs präsent (Bloomfield 2001; Firth 1957; Harris 1951). Technisch wurde sie von Mikolov u. a. (2013) realisiert. Das Prozedere läuft wie folgt ab: Man bestimmt den Umfang des Kontextes (eine festgelegte Tokenanzahl rund um das zu bestimmende Wort, die auch als „Fenster“ bezeichnet wird) und lässt den Computer auf solche Weise die Texte „lesen“. Ziel ist es, die Wahrscheinlichkeit des Vorkommens eines Wortes in einem bestimmten Kontext vorherzusagen. Algorithmisch ist das auf zweierlei Weise umsetzbar. Man kann ein Wort anhand seiner Kontexte vorhersagen, was dem CBOW-Modell (engl. „continuous bag-of-words“) entspricht. Alternativ können Kontexte anhand eines Wortes vorhergesagt werden, was im skip-gram-Modell implementiert ist. Wenn Sie ein relativ kleines Korpus haben, dann wird empfohlen, das skip-gram-Modell anzuwenden, weil dieser Algorithmus besser mit niedrigfrequenten Wörtern arbeitet (s. die Empfehlungen von Google).

Bisher gibt es keine word2vec-Tools, die über eine graphische Benutzeroberfläche (vgl. GUI) verfügen. Es ist darum nicht möglich, die Methode anzuwenden ohne ein Minimum an Code in einem Kommandozeilen-Tool (vgl. Commandline) einzugeben. Sie können aber mithilfe der Lerneinheit word2vec (Schumacher 2024) die Methode Schritt für Schritt erlernen und anhand eines Beispiels erproben. Die Einstellung der technischen Parameter, die variabel angepasst werden können, verlangt allerdings Grundkenntnisse dessen, was sie für die Modellierung bewirken. Es handelt sich bei word2vec darum insgesamt nicht um eine niederschwellige und einsteigsfreundliche Methode. Sind aber Grundlagen der Nutzung von Code in einer Kommandozeile vorhanden, so ist die Anwendung relativ einfach und schnell zu erlernen. Es gibt spezielle Toolpackages, wie z.B. eine Python-basierte Implementierung in gensim [Schumacher und Akazawa (2024); s. models.word2vec], die Ihnen ermöglichen, Ihr eigenes Modell mit wenigen Code-Zeilen zu trainieren (vgl. Machine Learning) und zu evaluieren. Dafür sind keine tiefgreifenden Programmierkenntnisse erforderlich. Sie müssen lediglich in der Lage sein, nötige Pakete zu installieren, das gewünschte Textkorpus zu laden und einzulesen. Dieses Verfahren erfordert kein besonderes Preprocessing (mehr darüber im Methodenbeitrag zur Korpusbildung (Bläß 2024)). Die einzige Voraussetzung ist, dass das Korpus in Sätze aufgeteilt werden muss, was Sie zum Beispiel mithilfe unserer Lerneinheit „Preprocessing mit NLTK“ (Schumacher und Vauth 2024) machen können. Bei NLTK (Natural Language ToolKit) handelt es sich um eine Sammlung von Python-Bibliotheken und Programmen zur Verarbeitung natürlicher Sprachen (vgl. NLP), die separat installiert werden muss. Das word2vec-Modell (Řehůřek 2022) kann mit unterschiedlichen Parametern trainiert werden (vgl. die Liste der Parameter auf der Webseite des Entwicklers). Sie können jedoch Ihr Modell erst einmal mit den Voreinstellungen (vgl. Default) erstellen. Sie übergeben dann dem word2vec-Modul nur das in Sätze segmentierte Korpus. Zu den wichtigsten Voreinstellungen gehören:

- **vector_size** = 100: Die Anzahl der Dimensionen (= Anzahl der Wortkontexte);
- **window** = 5: Die Größe des Fensters (in Token), mit dem die Texte gelesen werden; zur Information: bei einem kleinen Fenster (=2) werden die strukturellen (grammatischen) Regularitäten des Vokabulars abgebildet, bei der Vergrößerung des Fensters werden mehr semantische Regularitäten erfasst (Grayson u. a. 2016).
- **epochs** = 1: Anzahl der Durchgänge (Iterationen), in denen das Modell trainiert wird bzw. der Algorithmus die häufigen Kontexte der Wörter erlernt. Die Grundeinstellung in gensim geht von fünf Iterationen aus. Mehr Iterationen können die Qualität des Modells deutlich erhöhen, brauchen aber auch mehr Rechenkapazität.
- **min_count** = 5: Wörter, die weniger als 5 Mal im Korpus vorkommen, werden ignoriert;
- **sg** = 0: Das Modell wird mit einem CBOW-Algorithmus trainiert (s.o.);
- **hs** = 0: Dieser Parameter muss berücksichtigt werden, wenn Sie den rechnerischen Aufwand bei der Berechnung der Wahrscheinlichkeiten für das Korpusvokabular verringern möchten. Die Abkürzung „hs“ steht für die hierarchische Softmax-Funktion (s. das Tutorial zum Verfahren von McCormick, https://www.youtube.com/watch?v=pyzIWCelt_E29). Diese Funktion ist bei den Voreinstellungen auf 0 gesetzt und wird dadurch deaktiviert.

Nachdem Sie Ihr Modell trainiert und auf dem Computer gespeichert haben, können Sie mit der Analyse anfangen. Sie laden das lokal gespeicherte Modell, das einem Vokabular entspricht, in dem jedem Wort der entsprechende Wortvektor zugeteilt ist. Jetzt können Sie die semantischen Ähnlichkeiten zwischen den Wörtern im Korpus untersuchen. Sie können Wörter miteinander addieren oder bestimmte Wörter aus den Kontexten, die Sie interessieren, herausrechnen. Auf diese Weise können Sie sich den Konzepten, die hinter der Wortverwendung in dem betrachteten Korpus stehen, nähern. In Abb. 4 sehen Sie einige Beispiele dafür, bei denen auch die cosine similarity, ein Wert für die Stärke der Ähnlichkeit, mit angegeben wird. Die cosine similarity liegt immer zwischen 0 und 1, wobei 0 für keine Ähnlichkeit steht und 1 für Bedeutungsähnlichkeit.


```

In : my_model.wv.most_similar(positive=['Natur'], topn=5)
Out : [('Kunst', 0.784726083278656),
      ('Schönheit', 0.7686553001403809),
      ('Sinnlichkeit', 0.7597049474716187),
      ('Phantasie', 0.757921040058136),
      ('Selbstsucht', 0.7509210109710693)]

In : my_model.wv.most_similar(positive=['Natur', 'Mensch'], topn=5)
Out : [('Welt', 0.705716073513031),
      ('Kunst', 0.7008111476898193),
      ('Rasse', 0.6808986663818359),
      ('Geist', 0.6710090637207031),
      ('Schönheit', 0.6558336019515991)]

In : my_model.wv.most_similar(positive=['Natur'], negative = ['Mensch'], topn=5)
Out : [('Liebesglut', 0.5549246072769165),
      ('Vielheit', 0.5322432518005371),
      ('Sprödigkeit', 0.5317336320877075),
      ('Unsichtbarkeit', 0.5148810744285583),
      ('Skepsis', 0.5103720426559448)]

```

Abb. 4: Vektorenarithmetik am Beispiel des Korpus dProse

Zum besseren Verständnis der Zusammenhänge zwischen den Konzepten innerhalb des Korpus können Sie Ihre Daten visualisieren (s. Abb. 5). Mit der plot-Funktion aus der Python-Bibliothek matplotlib, einer weiteren Python-Bibliothek, die zusätzlich installiert werden muss, gewinnen Sie einen visuellen Einblick in die Daten. Im letzten Teil des [Tutorials](#) von Paul Vierthaler finden Sie ein Code-Beispiel, wie Sie Ihre Daten zur Visualisierung vorbereiten und mit dieser Funktion Platten können. Bei dieser Art der Visualisierung handelt es sich um eine **PCA**, eine Principle Component Analysis (oder Hauptkomponentenanalyse). Die Eigenschaften einzelner Wörter werden dabei auf zwei Hauptkomponenten heruntergerechnet, sodass sich – ähnlich wie in Abschnitt 5 beschrieben – ein zweidimensionaler Raum ergibt.

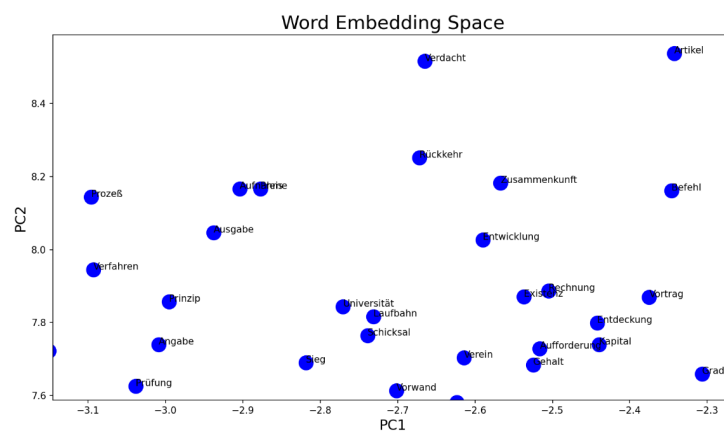


Abb. 5: Ein Fragment des semantischen Raums für das dProse-Korpus (~ 79,262,043 Tokens)

Mit einer anderen Art der Dimensionsreduktion auf einen zwei (bis drei-) dimensionalen Raum namens tSNE (engl. *t-distributed stochastic neighbor embedding*), erreichen Sie eine stärker in Cluster gegliederte Art der Darstellung der Daten (vgl. Abb. 6). Bei diesem Verfahren werden für Objekte in einem hochdimensionalen Raum paarweise Abstände errechnet, die bei der anschließenden Dimensionsreduktion so gut wie möglich erhalten bleiben. Im Gegensatz zum PCA-Verfahren werden also lokale Positionen (nämlich die Relationen zwischen jeweils zwei Objekten) statt globale Positionen (jeweils die Position eines Objektes im gesamten Raum) berücksichtigt. Es ergeben sich Gruppierungen, die bereits die Anmutung einer sinnhaften Gliederung haben (dennoch aber ebenso interpretationsbedürftig sind wie die Ergebnisse einer PCA-Visualisierung). Die Bedeutungen jedes Wortes im Korpus sind bei word2vec in einem N-dimensionalen Vektor gespeichert (der in gensim vorgegebene Wert beträgt 100 Dimensionen). Zur besseren Übersichtlichkeit, werden die Daten in ihrer Dimensionalität reduziert. Der Preis einer solchen Umwandlung ist ein gewisser Informationsverlust. Dem t-SNE-Algorithmus gelingt es allerdings die Daten so in eine zweidimensionale Repräsentation zu transformieren (s. Abb. 5), dass die ursprüngliche Struktur so gut wie möglich erhalten bleibt (Müller und Guido 2017, 154; Maaten und Hinton 2008). Eine Illustration zur Anwendung dieses Algorithmus mit literarischen Daten sowie ihre technische Implementierung in Python finden Sie bei Smetanin (2018).

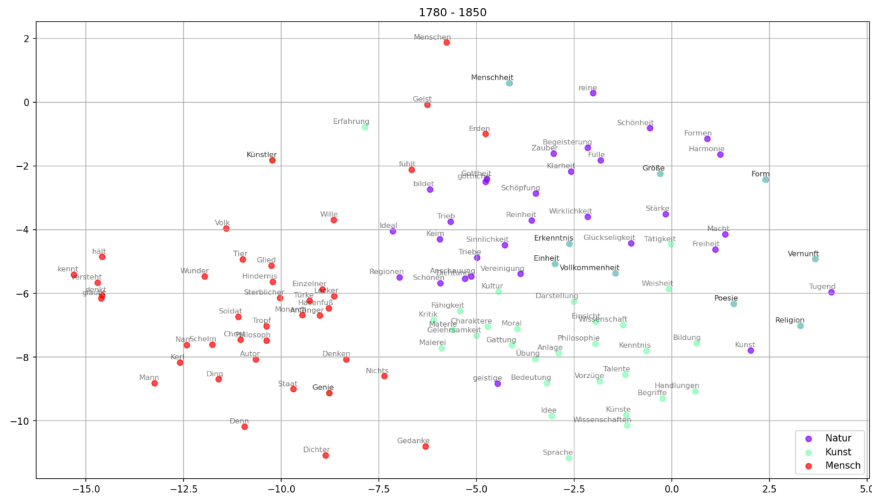


Abb. 6: t-SNE-Repräsentation für die Wortvektoren „Natur“, „Kunst“, „Mensch“ (ein kleines Romankorpus 1780 – 1850 mit einem Umfang von ~ 9,593,602 Tokens)

Externe und weiterführende Links

- Skip-gram-Modell (Google): <https://web.archive.org/save/https://code.google.com/archive/p/word2vec/> (Letzter Zugriff: 06.10.2024)
- Word2vec-Modell Parameter: <https://web.archive.org/save/https://radimrehurek.com/gensim/models/word2vec.html> (Letzter Zugriff: 06.10.2024)
- Hierarchische Softmax-Funktion in word2vec (YouTube-Tutorial): https://web.archive.org/save/https://www.youtube.com/watch?v=pzyIWCelt_E (Letzter Zugriff: 06.10.2024)
- Codebeispiel zur Visualisierung von Word Embeddings mit Python (matplotlib; YouTube-Tutorial): <https://web.archive.org/save/https://www.youtube.com/watch?v=6PYeLFh-N1E> (Letzter Zugriff: 06.10.2024)

Bibliographie

- Al-Saqqa, Samar und Arafat A. Awajan. 2019. The use of word2vec model in sentiment analysis: A survey. *Proceedings of the 2019 International Conference on Artificial Intelligence, Robotics and Control*: 39–43. doi: 10.1145/3388218.3388229, <https://api.semanticscholar.org/CorpusID:219156043>.
- Anz, Thomas, Hrsg. 2007. Inhaltsanalyse. In: *Handbuch Literaturwissenschaft*, 2: Methoden und Theorien:55–69. Stuttgart, Weimar: Metzler.
- Bläß, Sandra. 2024. Methodenbeitrag: Korpusbildung. Hg. von Evelyn Gius. *forTEXT* 1, Nr. 2. Korpusbildung (12. Juni). doi: 10.48694/fortext.3708, <https://fortext.net/routinen/methoden/korpusbildung>.
- Bloomfield, Leonard. 2001. *Die Sprache*. Edition Praesens. <https://library.oapen.org/handle/20.500.12657/33414> (zugegriffen: 12. Oktober 2021).
- Bojanowski, Piotr, Edouard Grave, Armand Joulin und Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5: 135–146. doi: 10.1162/tacl_a_00051,.
- Bourdieu, Pierre. 1999. *Die Regeln der Kunst: Genese und Struktur des literarischen Feldes*. Frankfurt am Main: Suhrkamp.
- Bubenhofer, Noah. 2020. Semantische Äquivalenz in Geburtserzählungen: Anwendung von Word Embeddings. *Zeitschrift für germanistische Linguistik* 48, Nr. 3 (25. November): 562–589. doi: 10.1515/zgl-2020-2014, <https://www.degruyter.com/document/doi/10.1515/zgl-2020-2014/html> (zugegriffen: 5. Februar 2022).
- Burghardt, Manuel und Bernhard Liebl. 2020. „The Vectorian“ – eine parametrisierbare Suchmaschine für intertextuelle Referenzen. In: Paderborn: Zenodo. doi: 10.5281/zenodo.4621836, <https://doi.org/10.5281/zenodo.4621836>.
- Cherny. 2014. Visualizing Word Embeddings in Pride and Prejudice. <http://blogger.ghostweather.com/2014/11/visualizing-word-embeddings-in-pride.html> (zugegriffen: 24. Januar 2022).
- Dobson, James E. 2021. Vector hermeneutics: On the interpretation of vector space models of text. *Digital Scholarship in the Humanities* 37, Nr. 1 (September): 81–93. doi: 10.1093/llc/fqab079, <https://doi.org/10.1093/llc/fqab079>.
- Ehrmanntraut, Anton, Thora Hagen, Leonard Konle und Fotis Jannidis. 2021. Type- and Token-based Word Embeddings in the Digital Humanities: 23.

- Firth, J. R. 1957. A synopsis of linguistic theory 1930-55. In: *Studies in Linguistic Analysis*, 1–32. Oxford: The Philological Society.
- Gius, Evelyn, Svenja Guhr und Inna Uglanova. 2021. „d-Prose 1870–1920“ a Collection of German Prose Texts from 1870 to 1920. *Journal of Open Humanities Data* 7 (8. Juli): 11. doi: 10.5334/johd.30, <http://openhumanitiesdata.metajnl.com/articles/10.5334/johd.30/> (zugegriffen: 2. August 2021).
- Gius, Evelyn und Inna Uglanova. 2023. Natur als Agens. Versuch einer computationellen Annäherung. In: *Romantische Ökologien: Vielfältige Naturen um 1800*, hg. von Roland Borgards, Frederike Middelhoff, und Barbara Thums, 4:265–289. Neue Romantikforschung. Berlin, Heidelberg: Springer. doi: 10.1007/978-3-662-67186-3_13, https://doi.org/10.1007/978-3-662-67186-3_13 (zugegriffen: 3. Oktober 2023).
- Grayson, Siobhán, Maria Mulvany, Karen Wade, Gerardine Meaney und Derek Greene. 2016. Novel2Vec: Characterising 19th Century Fiction via Word Embeddings. In: <https://researchrepository.ucd.ie/handle/10197/8360> (zugegriffen: 22. April 2021).
- Harris, Zellig S. 1951. *Methods in structural linguistics*. Methods in structural linguistics. Chicago, IL, US: University of Chicago Press.
- Heuser, Ryan James. 2017b. Word Vectors in the Eighteenth Century. In: 5. <https://dh2017.adho.org/abstracts/582/582.pdf> (zugegriffen: 24. Januar 2022).
- . 2017a. Word Vectors in the Eighteenth Century, Episode 2: Methods. <https://ryanheuser.org/word-vectors-2/> (zugegriffen: 24. Januar 2022).
- Jacobs, Arthur M. 2019. Sentiment Analysis for Words and Fiction Characters From the Perspective of Computational (Neuro-)Poetics. *Frontiers in Robotics and AI* 6 (17. Juli): 53. doi: 10.3389/frobt.2019.00053, <https://www.frontiersin.org/article/10.3389/frobt.2019.00053/full> (zugegriffen: 19. Oktober 2022).
- Jeßing, Benedikt. 2013. Kritische Theorie. In: *Metzler-Lexikon Literatur- und Kulturtheorie: Ansätze, Personen, Grundbegriffe*, hg. von Ansgar Nünning, 409–411. 5. Aufl. Stuttgart, Weimar.
- Jurafsky, Daniel und James H. Martin. 2021. Vector Semantics and Embeddings. In: *Speech and Language Processing*. <https://web.stanford.edu/~jurafsky/slp3/6.pdf> (zugegriffen: 14. Januar 2022).
- Kim, Eric. 2019. Optimize Computational Efficiency of Skip-Gram with Negative Sampling. *Python Excursions: Stories About Python and Data Science*. https://aegis4048.github.io/optimize_computational_efficiency_of_skip-gram_with_negative_sampling (zugegriffen: 24. Januar 2022).
- Kocher, Mirco und Jacques Savoy. 2018. Distributed language representation for authorship attribution. *Digital Scholarship in the Humanities* 33, Nr. 2: 425–441. doi: 10.1093/lc/fqx046, <https://doi.org/10.1093/lc/fqx046>.
- Köppe, Tilmann und Simone Winko. 2013. *Neuere Literaturtheorien. Eine Einführung*. 2. Aufl. Stuttgart (u.a.): Metzler.
- Liebl, Bernhard und Manuel Burghardt. 2020. „Shakespeare in the Vectorian Age“ – An evaluation of different word embeddings and NLP parameters for the detection of Shakespeare quotes. In: *Proceedings of the 4th joint SIGHUM workshop on computational linguistics for cultural heritage, social sciences, humanities and literature*, 58–68. Online: International Committee on Computational Linguistics, Dezember. <https://aclanthology.org/2020.latechclfl-1.7>.
- Lieske, Stephan. 2013. Strukturalismus, amerikanischer, französischer, genetischer. In: *Metzler-Lexikon Literatur- und Kulturtheorie: Ansätze, Personen, Grundbegriffe*, hg. von Ansgar Nünning, 721–724. 5. Aufl. Stuttgart Weimar: Metzler Lexikon.
- Lindgren, Simon. 2020. *Data Theory: Interpretive Sociology and Computational Methods*. Cambridge, Medford, MA: Polity.
- Maaten, Laurens van der und Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9, Nr. 86: 2579–2605. <http://jmlr.org/papers/v9/vandermaaten08a.html>.
- Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado und Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In: *Advances in Neural Information Processing Systems*, hg. von C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, und K.Q. Weinberger, 26:3111–3119. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.
- Müller, Andreas und Sarah Guido. 2017. *Einführung in Machine Learning mit Python*. Heidelberg: dpunkt.verlag GmbH.
- O A. 2019. *Hierarchical Softmax in word2vec*. https://www.youtube.com/watch?v=pzyIWCelt_E%3E (zugegriffen: 24. Januar 2022).
- Pedregosa, Fabian, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, u. a. 2011. Scikit-learn. Machine Learning in Python. *Journal of Machine Learning Research*: 2825–2830. doi: 10.5555/1953048.2078195.
- Řehůřek, Radim. 2014. Word2vec Tutorial. <https://rare-technologies.com/word2vec-tutorial/> (zugegriffen: 24. Januar 2022).
- Řehůřek, Radim. 2022. Gensim: topic modelling for humans. 6. Mai. https://radimrehurek.com/gensim/auto_examples/tutorials/run_word2vec.html (zugegriffen: 27. Oktober 2022).
- Röhle, Bernhard Rieder Theo. 2012. Digital Methods: Five Challenges. In: *Understanding Digital Humanities*, hg. von David M. Berry, 67–84. London: Palgrave Macmillan UK. http://link.springer.com/10.1057/9780230371934_4 (zugegriffen: 7. November 2016).

- Schmidt, Ben. 2015a. Rejecting the gender binary: a vector-space operation. *Ben's Bookworm Blog*. <http://bookworm.benschmidt.org/posts/2015-10-30-rejecting-the-gender-binary.html> (zugegriffen: 24. Januar 2022).
- . 2015b. Vector Space Models for the Digital Humanities. *Ben's Bookworm Blog*. <http://bookworm.benschmidt.org/posts/2015-10-25-Word-Embeddings.html> (zugegriffen: 24. Januar 2022).
- Schöch, Christof. 2022. @Burghardt @M_K_Schumacher Danke! Es fehlt mMn auch im Addendum der Hinweis, dass man jetzt ja auch von der Performance (fertiger, trainierter) Sprachmodelle auch mit minimalstem Aufwand, ganz ohne Keras / Pytorch etc. profitiert, einfach weil bspw. #spaCy und #Stanza sie für Annotationstasks einsetzen. Tweet. @christof77. 28. Januar. <https://twitter.com/christof77/status/1487163684749291528> (zugegriffen: 3. Februar 2022).
- Schöch, Christof. 2023. Quantitative Semantik. Word Embedding Models für literaturwissenschaftliche Fragestellungen. In: *Digitale Literaturwissenschaft: DFG-Symposion 2017*, hg. von Fotis Jannidis, 535–562. Stuttgart: J.B. Metzler. https://doi.org/10.1007/978-3-476-05886-7_22.
- Schöch, Christof, Frédéric Döhl, Achim Rettinger, Evelyn Gius, Peer Trilcke, Peter Leinen, Fotis Jannidis, Maria Hinzmann und Jörg Röpke. 2020. Abgeleitete Textformate: Text und Data Mining mit urheberrechtlich geschützten Textbeständen. *Zeitschrift für digitale Geisteswissenschaften*, Nr. 5. doi: 10.17175/2020_006, http://zfdg.de/2020_006 (zugegriffen: 9. November 2020).
- Schumacher, Mareike. 2024. Lerneinheit: word2vec mit Gensim. Hg. von Evelyn Gius. *forTEXT* 1, Nr. 10. word2vec (30. Oktober). doi: 10.48694/fortext.3816, <https://fortext.net/routinen/lerneinheiten/word2vec-mit-gensim>.
- Schumacher, Mareike und Mari Akazawa. 2024. Toolbeitrag: Gensim. Hg. von Evelyn Gius. *forTEXT* 1, Nr. 10. word2vec (30. Oktober). doi: 10.48694/fortext.3817, <https://fortext.net/tools/tools/gensim>.
- Schumacher, Mareike, Inna Uglanova und Evelyn Gius. 2022. d-Romane-Romantik (d-RoRo). Zenodo, 17. Oktober. doi: 10.5281/ZENODO.7215170, <https://zenodo.org/record/7215170> (zugegriffen: 18. Oktober 2022).
- Schumacher, Mareike und Michael Vauth. 2024. Lerneinheit: Preprocessing mit NLTK. Hg. von Evelyn Gius. *forTEXT* 1, Nr. 2. Korpusbildung (12. Juni). doi: 10.48694/fortext.3809, <https://fortext.net/routinen/lerneinheiten/preprocessing-mit-nltk>.
- Smetanin, Sergey. 2018. Google News and Leo Tolstoy: Visualizing Word2Vec Word Embeddings using t-SNE. <https://towardsdatascience.com/google-news-and-leo-tolstoy-visualizing-word2vec-word-embeddings-with-t-sne-11558d8bd4d> (zugegriffen: 24. Januar 2022).
- Vierthaler, Paul. 2020. Word Embedding Models in Python. <https://www.youtube.com/watch?v=6PYeLFh-N1E> (zugegriffen: 24. Januar 2022).
- Wevers, Melvin und Marijn Koolen. 2020. Digital begriffsgeschichte: Tracing semantic change using word embeddings. *Historical Methods: A Journal of Quantitative and Interdisciplinary History* 53, Nr. 4 (1. Oktober): 226–243. doi: 10.1080/01615440.2020.1760157, <https://www.tandfonline.com/doi/full/10.1080/01615440.2020.1760157> (zugegriffen: 4. Februar 2022).
- Wohlgenannt, Gerhard, Ekaterina Chernyak und Dmitry Ilvovsky. 2016. Extracting social networks from literary text with word embedding tools. In: *Proceedings of the workshop on language technology resources and tools for digital humanities (LT4DH)*, 18–25. Osaka, Japan: The COLING 2016 Organizing Committee, Dezember. <https://aclanthology.org/W16-4004>.

Glossar

- Annotation** Annotation beschreibt die manuelle oder automatische Hinzufügung von Zusatzinformationen zu einem Text. Die manuelle Annotation wird händisch durchgeführt, während die (teil-)automatisierte Annotation durch **Machine-Learning-Verfahren** durchgeführt wird. Ein klassisches Beispiel ist das automatisierte **PoS-Tagging** (Part-of-Speech-Tagging), welches oftmals als Grundlage (**Preprocessing**) für weitere Analysen wie Named Entity Recognition (NER) nötig ist. Annotationen können zudem deskriptiv oder analytisch sein.
- Browser** Mit Browser ist in der Regel ein Webbrowser gemeint, also ein Computerprogramm, mit dem das Anschauen, Navigieren auf, und Interagieren mit Webseiten möglich wird. Am häufigsten genutzt werden dafür Chrome, Firefox, Safari oder der Internet Explorer.
- Close Reading** Close Reading bezeichnet die sorgfältige Lektüre und Interpretation eines einzelnen oder weniger Texte. Close Reading ist in der digitalen Literaturwissenschaft außerdem mit der manuellen **Annotation** textueller Phänomene verbunden (vgl. auch **Distant Reading** als Gegenbegriff).
- Code** Der Code, oder auch Programmcode/ Maschinencode, bezieht sich auf eine Sammlung von Anweisungen, die durch verschiedene Programmiersprachen wie Java, Python oder C realisiert werden können. Für die Ausführung der Anweisungen wird der Code durch einen Compiler oder einen Interpreter in die Maschinensprache, einen Binärcode, des Computers übersetzt.
- Commandline** Die Commandline (engl. *command line interface* (CLI)), auch Kommandozeile, Konsole, Terminal oder Eingabeaufforderung genannt, ist die direkteste Methode zur Interaktion eines Menschen mit einem Computer. Programme ohne eine grafische Benutzeroberfläche (**GUI**) werden i. d. R. durch Texteingabe in die Commandline gesteuert. Um die Commandline zu öffnen, klicken Sie auf Ihrem Mac „cmd“ + „space“.

geben „Terminal“ ein und doppelklicken auf das Suchergebnis. Bei Windows klicken Sie die Windowstaste + „R“, geben „cmd.exe“ ein und klicken Enter.

CSV CSV ist die englische Abkürzung für *Comma Separated Values*. Es handelt sich um ein Dateiformat zur einheitlichen Darstellung und Speicherung von einfach strukturierten Daten mit dem Kürzel `.csv`, sodass diese problemlos zwischen IT-Systemen ausgetauscht werden können. Dabei sind alle Daten zeilenweise angeordnet. Alle Zeilen wiederum sind in einzelne Datenfelder aufgeteilt, welche durch Trennzeichen wie Semikola oder Kommata getrennt werden können. In Programmen wie Excel können solche Textdateien als Tabelle angezeigt werden.

Default Das/der Default (engl. für Voreinstellung oder Standardwert) bezeichnet den Wert einer Softwareeinstellung oder einer Eingabevariable, der verwendet wird, falls Nutzer*innen selbst keinen Wert oder keine Einstellungen vornehmen. Es handelt sich also um die standardmäßig festgelegten Einstellungen eines Tools oder Programms. Per Default festgelegte Parameter lassen sich i. d. R. manuell umstellen.

Distant Reading Distant Reading ist ein Ansatz aus den digitalen Literaturwissenschaften, bei dem computationale Verfahren auf häufig große Mengen an Textdaten angewandt werden, ohne dass die Texte selber gelesen werden. Meist stehen hier quantitative Analysen im Vordergrund, es lassen sich jedoch auch qualitative **Metadaten** quantitativ vergleichen. Als Gegenbegriff zu *Close Reading* wurde der Begriff insbesondere von Franco Moretti (2000) geprägt.

Domäneadaption Domäneadaption beschreibt die Anpassung einer in einem Fachgebiet entwickelten digitalen Methode an ein anderes Fachgebiet.

GUI GUI steht für *Graphical User Interface* und bezeichnet eine grafische Benutzeroberfläche. Ein GUI ermöglicht es, Tools mithilfe von grafischen Schaltflächen zu bedienen, um somit beispielsweise den Umgang mit der **Commandline** zu umgehen.

HTML HTML steht für *Hypertext Markup Language* und ist eine textbasierte Auszeichnungssprache zur Strukturierung elektronischer Dokumente. HTML-Dokumente werden von **Webbrowsern** dargestellt und geben die Struktur und Online-Darstellung eines Textes vor. HTML-Dateien können außerdem zusätzliche **Metainformationen** enthalten, die auf einer Webseite selbst nicht ersichtlich sind.

Korpus Ein Textkorpus ist eine Sammlung von Texten. Korpora (Plural für „das Korpus“) sind typischerweise nach Textsorte, Epoche, Sprache oder Autor*in zusammengestellt.

Lemmatisieren Die Lemmatisierung von Textdaten gehört zu den wichtigen **Preprocessing**-Schritten in der Textverarbeitung. Dabei werden alle Wörter (**Token**) eines Textes auf ihre Grundform zurückgeführt. So werden beispielsweise Flexionsformen wie „schneller“ und „schnelle“ dem Lemma „schnell“ zugeordnet.

Machine Learning Machine Learning, bzw. maschinelles Lernen im Deutschen, ist ein Teilbereich der künstlichen Intelligenz. Auf Grundlage möglichst vieler (Text-)Daten erkennt und erlernt ein Computer die häufig sehr komplexen Muster und Gesetzmäßigkeiten bestimmter Phänomene. Daraufhin können die aus den Daten gewonnen Erkenntnisse verallgemeinert werden und für neue Problemlösungen oder für die Analyse von bisher unbekannten Daten verwendet werden.

Markup Language Markup Language bezeichnet eine maschinenlesbare Auszeichnungssprache, wie z. B. **HTML**, zur Formatierung und Gliederung von Texten und anderen Daten. So werden beispielsweise auch **Annotationen** durch ihre Digitalisierung oder ihre digitale Erstellung zu Markup, indem sie den Inhalt eines Dokumentes strukturieren.

Metadaten Metadaten oder Metainformationen sind strukturierte Daten, die andere Daten beschreiben. Dabei kann zwischen administrativen (z. B. Zugriffsrechte, Lizenzierung), deskriptiven (z. B. Textsorte), strukturellen (z. B. Absätze oder Kapitel eines Textes) und technischen (z. B. digitale Auflösung, Material) Metadaten unterschieden werden. Auch **Annotationen** bzw. **Markup** sind Metadaten, da sie Daten/Informationen sind, die den eigentlichen Textdaten hinzugefügt werden und Informationen über die Merkmale der beschriebenen Daten liefern.

Named Entities Eine Named Entity (NE) ist eine Entität, oft ein Eigenname, die meist in Form einer Nominalphrase zu identifizieren ist. Named Entities können beispielsweise Personen wie „Nils Holgerson“, Organisationen wie „WHO“ oder Orte wie „New York“ sein. Named Entities können durch das Verfahren der Named Entity Recognition (NER) automatisiert ermittelt werden.

NLP *Natural Language Processing* (NLP), maschinelle Sprachverarbeitung zu Deutsch, ist ein Teilgebiet der Linguistik, der Informatik und der künstlichen Intelligenz, welches sich damit beschäftigt, wie Computer so programmiert werden, dass sie große Mengen an natürlichsprachlichen Daten verarbeiten und analysieren können.

OCR OCR steht für *Optical Character Recognition* und bezeichnet die automatische Texterkennung von gedruckten Texten, d. h. ein Computer „liest“ ein gescanntes Dokument, erkennt und erfasst den Text darin und generiert daraufhin eine elektronische Version.

- PCA** PCA steht für *Principal Component Analysis*. Die Hauptkomponentenanalyse ist ein komplexes, statistisches Verfahren zur Reduktion und Veranschaulichung umfangreicher Datensätze.
- POS** PoS steht für *Part of Speech*, oder „Wortart“ auf Deutsch. Das PoS- **Tagging** beschreibt die (automatische) Erfassung und Kennzeichnung von Wortarten in einem Text und ist ein wichtiger **Preprocessing**-Schritt, beispielsweise für die Analyse von **Named Entities**.
- Preprocessing** Für viele digitale Methoden müssen die zu analysierenden Texte vorab „bereinigt“ oder „vorbereitet“ werden. Für statistische Zwecke werden Texte bspw. häufig in gleich große Segmente unterteilt (*chunking*), Großbuchstaben werden in Kleinbuchstaben verwandelt oder Wörter werden **lemmatisiert**.
- Reintext-Version** Die Reintext-Version ist die Version eines digitalen Textes oder einer Tabelle, in der keinerlei Formatierungen (Kursivierung, Metadatenauszeichnung etc.) enthalten sind. Reintext-Formate sind beispielsweise TXT, RTF und **CSV**.
- Type/Token** Das Begriffspaar „Type/Token“ wird grundsätzlich zur Unterscheidung von einzelnen Vorkommnissen (Token) und Typen (Types) von Wörtern oder Äußerungen in Texten genutzt. Ein Token ist also ein konkretes Exemplar eines bestimmten Typs, während ein Typ eine im Prinzip unbegrenzte Menge von Exemplaren (Token) umfasst.
- Es gibt allerdings etwas divergierende Definitionen zur Type-Token-Unterscheidung. Eine präzise Definition ist daher immer erstrebenswert. Der Satz „Ein Bär ist ein Bär.“ beinhaltet beispielsweise fünf Worttoken („Ein“, „Bär“, „ist“, „ein“, „Bär“) und drei Types, nämlich: „ein“, „Bär“, „ist“. Allerdings könnten auch vier Types, „Ein“, „ein“, „Bär“ und „ist“, als solche identifiziert werden, wenn Großbuchstaben beachtet werden.