Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterthesis

Nassim Haidar

Word Embeddings der deutschen Medizinliteratur

Fakultät Life Sciences
Studiendepartment
Medizintechnik

Faculty of Life Sciences
Department of Biomedical Engineering

Nassim Haidar

Word Embeddings der deutschen Medizinliteratur

Masterthesis eingereicht im Rahmen der Masterprüfung

im Studiengang Master of Science Biomedical Engineering
am Department Medizintechnik
der Fakultät Life Sciences
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Andreas Meisel
Zweitgutachter: Dr. Alexander Motzek

Eingereicht am: 1. Oktober 2020

Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich während der Anfertigung dieser Masterarbeit unterstützt und motiviert haben.

Zuerst gebührt mein Dank Herrn Dr. Alexander Motzek für die Vergabe dieses spannenden Themas, für das entgegengebrachte Vertrauen, für die Betreuung und für die gute Zusammenarbeit. Für die hilfreichen Anregungen und die konstruktive Kritik bei der Erstellung dieser Arbeit möchte ich mich herzlich bedanken.

Ich bedanke mich bei Herrn Prof. Dr. Andreas Meisel für die fachliche Unterstützung bei meiner Forschung.

Weiterhin möchte ich mich bei meinen Eltern bedanken, die mich während der COVID-19-Krise sowohl seelisch als auch finanziell unterstützt haben und stets ein offenes Ohr für mich hatten.

Abschließend möchte ich mich bei meiner Verlobten und zukünftigen Frau bedanken, die während dieser Zeit stets Verständnis für meine Abwesenheit hatte.

Nassim Haidar

Thema der Arbeit
Word Embeddings der deutschen Medizinliteratur

Stichworte
Word2Vec, GloVe, Fasttext, t-SNE, Word Embedding, medizinische Texte, Wort-Vektor, Worteinbettung, Ähnlichkeitstest, Analogie-Test, Textverarbeitung

Kurzzusammenfassung
In dieser Arbeit werden die drei renommierten Word Embeddings (Word2Vec, GloVe und Fastext) mit einem deutschen medizinischen Text trainiert. Dabei soll geprüft werden, inwiefern die Word Embeddings den Inhalt eines deutschen medizinischen-Textes wiedergeben können. Dies wird untersucht, indem für Word Embeddings ein Ähnlichkeitstest und Analogie-Test durchgeführt werden. Zusätzlich werden die Wort-Vektoren visualisiert, um einen tieferen Einblick in die Word Embeddings zu ermöglichen. Dadurch ist es möglich, Cluster-Bildungen von ähnlichen Worten genauestens zu beobachten.Dabei wird deutlich, dass Fasttext eine zufriedenstellende Performance mit einem deutschen medizinischen Korpus erzielen kann.

Inhaltsverzeichnis

Abkürzungsverzeichnis

CBOW = Continuous Bag-Of-Words

CNN = Convolutional Neural Network

COVID-19 = Corona Virus Disease 2019

GloVe = Global Vectores

HIV = Human Immunodeficiency Virus

LDA = Latent Dirichlet Allocation

LSA = Latent Semantic Analysis

LSI = Latent Sematic Indexing

NCE = Noise Contrastive Estimation

NK-Zellen = natürliche Killerzellen

NLP = Natural Language Processing

NLTK = Natural Language Toolkit

NS = Negative Sampling

PLSA = Probabilistic Latent Semantic Analysis

PMI = Pointwise Mutual Information

SG = Skip-Gram

SGNS = Skip Gram Negative Sampling

SRN = Simple Recurrent Networks

t-SNE = t-Distributed Stochastic Neighbor Embedding

UTF-8 = Unicode Transformation Format

1 Einleitung

1.1 Einleitung

Obwohl im Jahre 2018 weltweit 18,1 Millionen [43] Menschen neu an Krebs erkranken und regelmäßig über 3.000 wissenschaftliche Publikationen [15] veröffentlicht werden, scheint es noch kein Medikament zu geben, das infizierte Krebszellen direkt bekämpft. Jahr für Jahr werden neue Forschungsergebnisse publiziert, sowohl mit konventionellen als auch mit nicht konventionellen Methoden. Dennoch versprechen all diese Methoden und Therapien nur minimale Erfolge. Weltweit gibt es eine Vielzahl von Laboren, die unterschiedliche Ansätze verfolgen und dokumentieren. Einige versprechen aussichtsreiche Heilmöglichkeiten, andere sind wiederum weniger aussichtsreich. Gibt man das Wort *cancer* in der Sucheingabe von PubMed ein, erhält man über 4,1 Millionen Publikationen [1].

Publikationen sind essentiell für Ärzte, da sie Therapiemöglichkeiten, die sich aus der Diagnose erschließen, ableiten können. Ebenfalls können schnell Erkrankungsmuster erkannt oder Erkrankungsverläufe neuer Viren verfolgt werden. So z. B. der Verlauf der Corona-Virus-Disease-2019, auch COVID-19 genannt. Das Virus wurde in China erstmals entdeckt. In einigen Publikationen ähneln sich die Therapieverfahren für eine Erkrankung, in anderen hingegen werden unterschiedliche Verfahren genutzt. Bei einigen sind die Nebenwirkungen größer, während bei anderen kaum welche auftreten. Zudem sind für einige Publikationen Doppelblindstudien durchgeführt worden, um eine Verzerrung durch den Placebo-Effekt zu verhindern. Dagegen konnten die Studien für andere Publikationen ohne doppelte Verblindung durchgeführt werden. Studien können äußerst unterschiedlich sein, obwohl es sich um denselben Erreger handelt. Um Unterschiede zwischen oder Gemeinsamkeiten von den Studien zu entdecken, muss man all diese Publikationen durchlesen, was besonders viel Zeit in Anspruch nimmt [34]. Das gesamte Wissen der Millionen veröffentlichten Publikationen zu extrahieren, würde eventuell Jahre dauern. Aus den gesammelten Informationen können Daten

über die Entwicklung der Krebsarten der letzten Jahrzehnte bestimmt werden. Vermutlich würde es sogar gelingen, eine Therapie zu entwickeln oder ein Medikament zu entdecken, das einige Krebsarten primär bekämpft. Doch wie sollen nun all diese Informationen aus den Publikationen extrahiert werden?

In den letzten Jahren wurden vermehrt Publikationen über Word Embedding veröffentlicht, die genau diese Probleme lösen können. Durch Word Embedding werden die syntaktischen und semantischen Eigenschaften eines Wortes entnommen, die in einem Satz enthalten sind. Dadurch können verwandte Wörter dargestellt werden, die wiederum mehr Informationen über den Text geben. Ein weiterer Vorteil ist, dass Ähnlichkeiten zwischen verschiedenen wissenschaftlichen Arbeiten festgestellt werden kann. Folglich können solche Arbeiten gruppiert und gezielter für bestimmte Interessengruppen vorgeschlagen werden. Vor allem, nachdem Tomas Mikolov Word2Vec vorstellte, fing der Trend des Wort-Vektors an [23].

1.2 Forschungsstand

1.2.1 Historie

Worteinbettungen oder Word Embeddings haben ihren Ursprung in den 50er Jahren. Zellig Harris und John Firth stellten fest, dass kontextuelle Informationen eine wesentliche und tragfähige Repräsentation von linguistischen Elementen darstellen [11]. Die ersten Versuche, um Merkmale zu konstruieren, die semantische Ähnlichkeiten besitzen, wurden 10 Jahre später von Charles Osgoods unternommen [26]. Im Jahre 1990 wurden die ersten Methoden zur Nutzung automatisch generierter kontextueller Eigenschaften entwickelt. Das einflussreichste und noch aktuelle Themenmodell nennt sich Latent Semantic Analysis/Latent Semantic Indexing (LSA/LSI). Aufgrund dessen stellte Deerwester [7] fest, dass Wörter, die in Dokumenten vorkommen, eine semantische Struktur haben. Die Wörter eines Dokuments wurden in einer Matrix geordnet und durch die Singulärwertzerlegung approximiert [33].

Ende der 90er Jahre standen sich zwei Modelle gegenüber: das LSI und das Simple Recurrent Networks (SRN). SRN stammt aus dem Bereich der künstlichen Intelligenz und basiert auf neuronalen Netzen. Beide Verfahren wurden über die Jahre weiterentwickelt. So entstanden Modelle wie das probabilistische LSA (PLSA) oder auch Latent Dirichlet Allocation (LDA) [12] [13]. Hingegen wurden aus dem SRN die Modelle der Convolutional Neuronal Networks (CNN) und der Autoencoders entwickelt [33].

Der Hauptunterschied zwischen dem LSA und dem SRN liegt in der Art der Kontextinformationen, die genutzt werden. Beim LSA werden Dokumente als Kontext verwendet. Beim SRN dagegen werden Wörter als Kontext verwendet, die aus linguistischer und kognitiver Sicht natürlicher sind. Diese Verfahren erfassen unterschiedliche Arten von semantischer Ähnlichkeit. Semantische Zusammenhänge wie „Boot“ und „Wasser“ werden von dokumentenbasierten Modellen erfasst. Semantische Ähnlichkeiten wie „Boot“ und „Schiff“ wiederum von wortbasierten Modellen erfasst [33].

1.2.2 Stand der Forschung

Mikolov entwickelte die wortbasierten Modelle Word2Vec [24]. Er veröffentlichte zwei statistische Modelle: 1) das Skip-Gram (SG) und 2) das Continuous Bag-Of-Words (CBOW) [23]. Bei beiden Modellen, handelt es sich um ein neuronales Netz mit drei Schichten. Die erste Schicht ist der Input-Layer und die letzte der Output-Layer. Durch Word2vec werden Wörter trainiert, indem eine Loss-Funktion mit einem Gradienten-Abfall optimiert wird. Beim SG (Abb. 2.4) handelt es sich um ein Verfahren, bei dem anhand eines Wortes die kontextbasierten Wörter vorhergesagt werden [23].

In den darauffolgenden Jahren wurde Word2Vec verbessert, indem die mittlere Schicht aus dem neuronalen Netz durch eine hierarchische Softmax ersetzt wurde [23]. Dadurch erhielt das Modell eine Log-Linearität. Weiterhin wurde das Modell um ein Negative Sampling (NS) erweitert [23], das ebenfalls eine Verfeinerung der Noise Contrastive Estimation (NCE) darstellt. Dadurch wird die hierarchische Softmax durch das NS ersetzt, wodurch es für das Modell ermöglicht wird, die Wahrscheinlichkeit, ob es sich um ein Wortpaar handelt, zu bestimmen, anstatt wie beim SG oder CBOW über den gesamten Korpus zu bestimmen.

Vor Word2vec war es die übliche Vorgehensweise, Matrizen zu verwenden, die globale Informationen über ein Dokument enthalten, z. B. LSA. Hierbei werden Worteinbettungen durch die Zerlegung von Term-Dokumentmatrizen über die Singulärwertzerlegung erreicht. Somit nutzt LSA zur Informationsquelle eine Dokument-Wort-Matrize. Doch zeigen die dargestellten Wort-Vektoren nicht dasselbe Verhalten wie bei Word2vec, da das neue Modell eine Wort-Wort-Matrize nutzt und das gesamte Dokument vernachlässigt. Zum Beispiel können die Wort-Vektoren von Word2Vec einfache Vektor-Arithmetik ausdrücken (Abb. 2.3). Eine Verknüpfung der lokalen Kontextfenster und der globalen Informationen führt zur Publikation der Stanford Universität über Global Vectors (GloVe). Auf diese Weise wurde eine überarbeitete Word2Vec-Methode veröffentlicht. GLoVe ist eine Erweiterung des SGs [30].

Fasttext wurde von Bojanowski publiziert und besitzt einen ähnlichen Aufbau wie Word2Vec. Es wird auch als Erweiterung von Word2Vec gesehen [2]. Der Autor erkannte, dass Word2Vec eine Schwachstelle besitzt: die morphologischen Strukturen der einzelnen Wörter werden vernachlässigt. Word2Vec kann nur die Merkmale eines Wortes anhand des semantischen Kontexts zuweisen. Der Kernpunkt von Word2Vec ist

es, die externen Nachbarn des Wortes zu bestimmen. Somit lag die Beschränkung darin, nur Wort-Vektoren zu erlernen, die auch im Vokabular vorhanden waren. Sprachen wie Deutsch oder Türkisch, in denen die internen morphologischen Strukturen der Wörter eine einflussreiche Bedeutung haben, können über Word2Vec nicht erfasst werden [2]. Diese Einschränkung wird durch die Anwendung von Unterwortinformationen (engl. Subword Information) behoben, wodurch auch Wörter erkannt bzw. vorhergesagt werden können, die nicht im Vokabular des Korpus waren [14].

1.3 Ziel der Arbeit

Mit der vorliegenden Arbeit wird das Ziel verfolgt, anhand der renommierten Word Embeddings Word2Vec, GloVe und Fasttext einen ersten Eindruck über den Inhalt eines deutschen medizinischen Textes mittels Schlüsselwörter zu erhalten. Dementsprechend werden die Word Embeddings mit einem deutschen medizinischen Korpus antrainiert. Folglich wird die Evaluierung mithilfe eines Ähnlichkeitstests und eines Analogie-Tests durchgeführt. Zusätzlich sollen die Ergebnisse des Ähnlichkeitstests und die Verteilung der Wort-Vektoren aus dem Korpus mit t-Distributed Stochastic Neighbor Embedding (t-SNE) visualisiert werden. Anschließend werden zu verschiedenen Kategorien die Top 20 der ähnlichen Wörter in einem Plot dargestellt. Zum Abschluss wird ein Zeitrafer-Plot für jedes Word Embedding wiedergeben. Dieser zeigt, inwiefern die einzelnen Datenpunkte im Diagramm nach jeder *Perplexity* (Kapitel 2.1.2) zueinander finden und ein Cluster bilden.

1.4 Aufbau der Arbeit

Die vorliegende Arbeit ist in drei Kategorien untergegliedert. Im zweiten Kapitel werden die Grundlagen diskutiert. Dort werden die für diese Arbeit notwendigen Word-Embedding-Programme (Word2Vec, GloVe und Fasttext) beschrieben. Außerdem werden die Wort-Ähnlichkeit und t-SNE, das für die Evaluation relevant ist, beschrieben.

Das mittlere Kapitel behandelt die Implementierung der Word Embeddings. Hinzukommend wird die Vor- und Nachbereitung des Korpus erklärt. Es werden die verwendeten Hyperparameter für das Antrainieren des Korpus erklärt und weshalb explizit diese Hyperparameter in Betracht gezogen werden.

Im letzten Kapitel folgt die Evaluation. Dort werden die Ergebnisse und Plots aufgetragen und miteinander verglichen. Weiterhin wird der Bezug zu wissenschaftlichen Publikationen hergestellt und schließlich diskutiert, welches Word Embedding signifikant besser abgeschnitten hat.

2 Word Embeddings

In diesem Kapitel handelt es sich um die Grundlagen, die vorausgesetzt werden, um die weiteren Kapiteln zu verstehen. Darin wird die Theorie der renommierten Word Embeddings (Word2Vec, GloVe und Fasttext) erläutert. Zudem werden die Grundlagen für den Ähnlichkeitstest beschrieben. Schließlich wird die Visualisierung mit t-SNE erklärt und die darin resultierenden Komplikationen.

2.1 Grundlagen

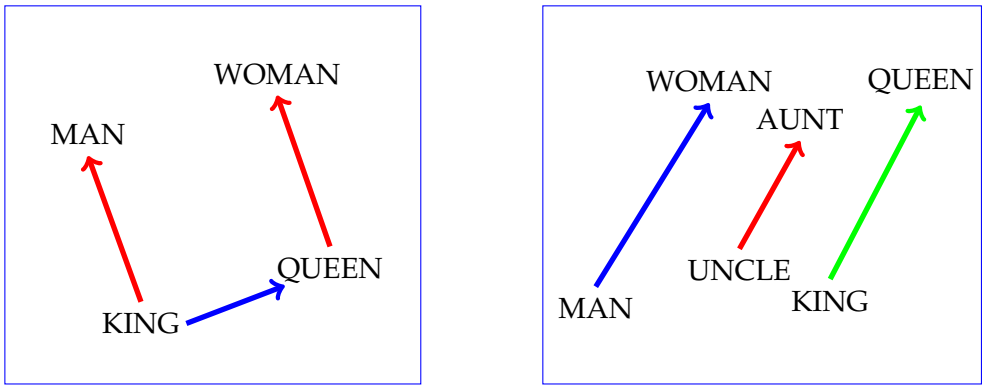
Word Embeddings werden im Natural Language Processing (NLP) eingesetzt. Sie können auf zwei unterschiedlichen Wegen erlangt werden. Mit einem vorbereiteten Datensatz, auch Korpus genannt, können die Informationen über die Anpassung der Gewichtungen (engl. weights) von neuronalen Netzwerken erlernt werden. Eine andere Möglichkeit besteht darin, vorberechnete bzw. vortrainierte Word Embeddings einzusetzen [6]. In dieser Arbeit werden die Versuche hauptsächlich mit vortrainierten Word Embeddings durchgeführt.

Hierbei geht es hauptsächlich darum, Buchstaben, die in einer bestimmten Zeichenfolge ein Wort ergeben, in eine Fließkommazahl umzuwandeln. Diesbezüglich entstehen Vektoren mit mehreren Dimensionen, wobei jede Dimension ein bestimmtes Merkmal an Informationen beinhaltet. Jeder Vektor bildet einen Datenpunkt in einem n-dimensionalen Raum. Die Vektoren spiegeln folglich eine Struktur des gesamten Korpus wider.

Vektoren, die eine hohe Dichte beinhalten, besitzen eine semantische Bindung zum Inhalt des Textes [6]. Somit ist der Computer in der Lage, linguistische Begriffe anhand ihrer Ähnlichkeit oder ihrer Beziehung zueinander zu vergleichen. In der Praxis werden Wörter bzw. Vektoren in höheren Dimensionen in einem dichtem Vektorraum eingebettet. Im Vektorraum können der Abstand und der Winkel der einzelnen Vektoren

bestimmt werden. Somit ermöglichen es Word-Embeddings-Modelle, Wörter, die in einer Beziehung zueinander stehen, zu erkennen. Solche Beziehungen können z. B. Bedeutungen, Analogien oder Gegensätze sein. Demzufolge kann das Wort *King* einen Vektor bilden, der auf *Queen* gerichtet ist.

Der Vorteil von Vektoren ist es, dass man sich die mathematischen Operationen zunutze machen kann. So können aus den Beziehungen folgende Analogien gewonnen werden: $King - Man + Woman = Queen$ (Abb. 2.1). Der Vektor, der von *King* auf *Queen* gerichtet ist, kann ebenfalls verwendet werden, um von *Man* auf *Woman* zu gelangen.



(a) Mathematische Gleichung (b) Wörter mit gleicher Semantik

Abbildung 2.1: (a) Anhand der mathematischen Rechenregel können neue Beziehungen untersucht werden. (b) Wortvektoren deuten auf ähnliche Wortvektoren hin.

2.1.1 Ähnlichkeit

Wörter, die in ähnlichen Kontexten auftauchen, sollten per se auch die gleichen Bedeutungen haben. Die Verteilungshypothese in der Linguistik besagt, dass linguistische Elemente, die im gleichen Kontext vorkommen, dazu tendieren, auch ähnliche Bedeutungen zu haben. Diese Hypothese wurde in den 50er Jahren von dem Sprachwissenschaftler Joos aufgestellt. Hierbei wurde beobachtet, dass Wörter wie *Okulist* und *Augenarzt* tendenziell in ähnlichen Kontexten vorkamen wie *Auge* und *untersucht* [11].

Die *cosinus similarity* beschreibt die Ähnlichkeit zwischen zwei Wörtern (Formel 2.1). Die variablen v und w stehen für Vektoren, derren Ähnlichkeit mittels des Skalarproduktes bestimmt wird. Zusätzlich werden die Vektoren $|v|$ und $|w|$ durch ihre Länge normiert, um einen Einheitsvektor der Länge 1 zu erhalten. Liegt der Quotient nahe 1, besteht hier eine sehr große Ähnlichkeit, während bei einem Wert von 0 die Wörter orthogonal zueinander stehen und somit v und w keine Übereinstimmung besitzen.

$$\cos(\boldsymbol{v}, \boldsymbol{w}) = \frac{\boldsymbol{v} * \boldsymbol{w}}{|\boldsymbol{v}||\boldsymbol{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}} \tag{2.1}$$

2.1.2 Visualisierung mit t-SNE

Für eine Visualisierung aller Wort-Vektoren müssen die mehrdimensionalen in zweidimensionale Wort-Vektoren umgewandelt werden. Dies wird mittels t-Distributed Stochastic Neighbor Embedding (t-SNE) umgesetzt [20].

Diese Methode benutzt lokale Beziehungen zwischen Datenpunkte im hochdimensionalen Raum, um eine niedrigere Dimension abzubilden, wodurch die Abstände zwischen den Punkten nahezu identisch bleiben. Das Besondere am t-SNE ist es, dass das Überfüllungsproblem, auch *Crowding Problem* genannt, effizient verhindert werden kann. Das Crowding-Problem basiert auf dem Fluch der Dimensionalität, auch *Curse*

Of Dimensionality genannt. Abb. 2.2 veranschaulicht dieses Phänomen. Die Oberfläche der Kugel bildet den Raum in der n-Dimensionalität. Auf der Oberfläche liegen nun die Punkte verteilt. Bei der Umwandlung in eine niedrigere Dimensionalität überlappen diese Punkte, da die Oberfläche bzw. der Raum bei gleichmäßiger Verteilung kleiner wird. [20]

Mit t-SNE ist es möglich, nichtlineare Strukturen zu erfassen. Zudem wird eine Wahrscheinlichkeitsverteilung unter Verwendung der Gaußschen Verteilung erstellt, die darauf abzielt, Beziehungen zwischen Punkten im hochdimensionalen Raum zu definieren. Im niedrigdimensionalen Raum wird die Wahrscheinlichkeitsverteilung mittels der Student t-Verteilung verwendet. [20]

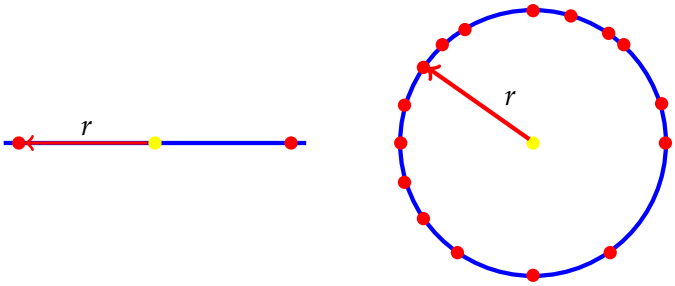


Abbildung 2.2: Die Illustration zeigt, dass in höherdimensionalen Räumen mehr Datenpunkte abgebildet werden können, als im niedrigdimensionalen Raum.

Ein zentraler Hyperparameter für t-SNE ist die *Perplexity*. Dabei geht es um die Anzahl der Nachbarn, die jeder Punkt auf dem Diagramm besitzt. Folglich gibt dieser Parameter eine Schätzung über die Anzahl der Nachbarn an, die jeder Datenpunkt besitzt. Eine geringere *Perplexity* bedeutet, dass das Diagramm sich auf die lokalen Minima bzw. auf die nächstliegenden Punkte konzentriert. Während eine hohe *Perplexity* den Einsatz eines großen gemeinsamen Bildes verfolgt [38]. In der wissenschaftlichen Arbeit von Maaten et al. 2008 heißt es weiter, dass dies eine komplexe Auswirkung auf die entstehenden Bilder hat [20].

2.2 Word2Vec

Word2Vec ist ein distributionelles Modell, das von Tomas Mikolov im Jahre 2013 vorgestellt wurde [10]. Word2Vec lernt die Bedeutung der Wörter nur durch die sequenzielle Verarbeitung eines großen Korpus. Bemerkenswert ist es, dass der Korpus aus unmarkierten Texten besteht. Somit erkennt das Modell anhand seines Korpus, dass mit FC Bayern München der Fußballverein und nicht die Stadt München gemeint ist.

Zudem erkennt Word2Vec ein Muster von Wörtern, das für Menschen auf den ersten Blick nicht ersichtlich ist. Jedes Wort kann mit einer Geografie, einer Empfindung oder einem Geschlecht assoziiert werden. Besitzt nun ein Wort im Korpus eine Eigenschaft, wie eine Ortsangabe, Menschenfreundlichkeit oder Weiblichkeit, erhalten alle Wörter einen spezifischen Wert für diese Eigenschaft in ihren Wort-Vektoren. Bei Word2Vec werden immer nur die benachbarten Wörter eines Satzes berechnet. Dabei ist es möglich, Wort-Vektoren zu addieren sowie zu subtrahieren, um neue Wort-Vektoren zu erzeugen. So können Wortanalogien entstehen, wie *Berlin* gehört zu *Deutschland* wie *Frankreich* zu *Paris* (Abb. 2.3). Der resultierende Vektor aus einer Addition und einer Subtraktion wird nie mit einem anderen Wort-Vektor identisch sein. Daher wird ein Wort-Vektor aus dem Vokabular ausgegeben, der dem resultierenden Vektor am nächsten kommt. Dieser wäre nach der Gleichung in Abb. 2.3 das Wort *Paris* [10]

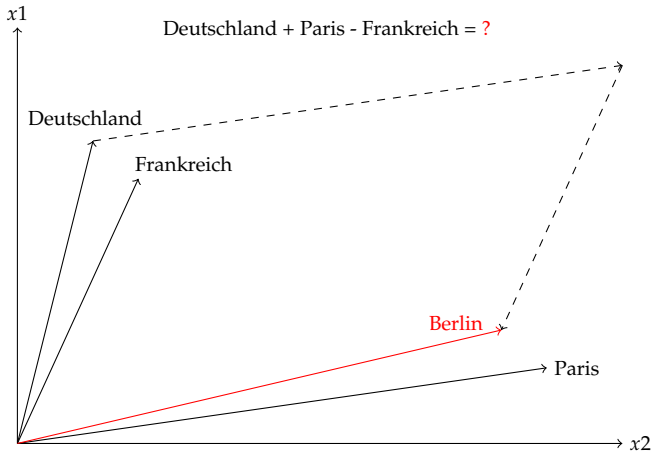
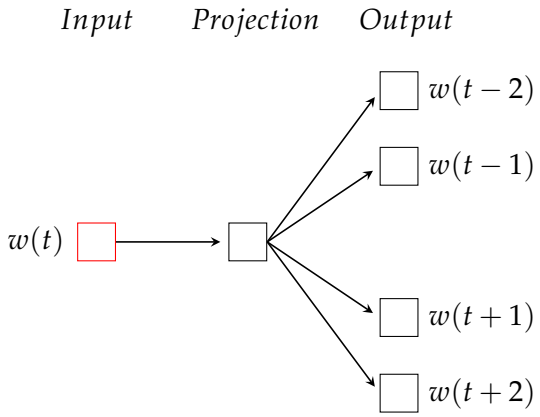


Abbildung 2.3: Geometrie von *Word2Vec*-Wort-Vektoren

Wort-Vektoren, die sich ähneln, werden mittels Skalarproduktes maximiert. Demzufolge werden Wort-Vektoren, die sich nicht ähneln, minimiert. Diese Quintessenz stammt aus der Arbeit von Mikolov et al. 2013 [24]. Es ist für die Rekonstitution der sprachlichen Kontexte von Wörtern spezialisiert. Als Input wird ein in getrennten Sätzen verarbeiteter Korpus übergeben, dessen Wörter in einem Vektorraum dargestellt werden können. Dieser besteht typischerweise aus mehreren hundert Dimensionen. Dabei wird jedes einzelne Wort im Korpus zu einem Vektor umgewandelt, der im Vektorraum repräsentativ ist. Aufgrund dessen werden Wort-Vektoren, die einen gemeinsamen Kontext haben, im Raum dicht nebeneinanderstehen.



Skip-gram

Abbildung 2.4: Beim Skip-Gram wird der Satz aus nur einem Wort vorhergesagt. Somit wird die Wahrscheinlichkeit für die restlichen Wörter des Satzes anhand des Inputs bzw. eines Wortes bestimmt.

Wort-Vektoren können mittels folgender logarithmisch-linearer Modelle ermittelt werden: Continuous Bag of Words (CBOW) und Skip-Gram (SG) . Beide Rechenmodelle haben ihre Vor- und Nachteile. Die Vorteile von SG sind es, dass es bei einem kleinen Datensatz zufriedenstellende Ergebnisse erbringt und seltene Wörter effizient erkannt werden [24]. Somit beschäftigt sich diese Arbeit mit dem SG-Modell.

Das SG-Modell benötigt als Input jeweils nur ein Wort (Abb. 2.4). Die Anzahl der erfassten Kontext-Wörter $w_{t-/+m}$ sind vom Wort w_t und dessen gewählten Radius

abhängig. Bei einem Radius von $m = 2$ (wie in Abb. 2.5) werden jeweils die zwei benachbarten Wörter auf der linken und auf der rechten Seite des Wortes ausgewertet. Dieser Vorgang verläuft über den kompletten Satz bzw. über den gesamten Korpus T. Dabei werden anhand eines Wortes die Durchschnittswahrscheinlichkeiten der benachbarten Kontext-Wörter über einen Korpus maximiert [35]. Gekoppelt mit der Maximum-Likelihood-Methode (Formel 2.2) ist die Gleichung für das SG vollständig. Somit werden Wörter, die eine Ähnlichkeit besitzen, maximiert, während Wörter, die keine Ähnlichkeit besitzen, minimiert werden [16].

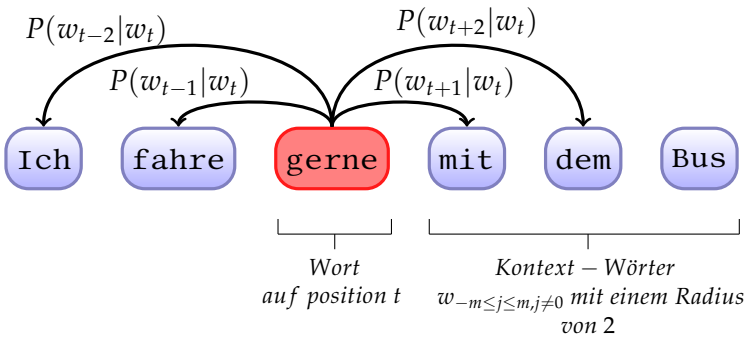


Abbildung 2.5: Von einem Wort ausgehend werden die benachbarten Wörter $-2 \leq j \leq 2$ ausgegeben.

$$L(\theta) = \frac{1}{T} \sum_{t=1}^{T} \sum_{-m \leq j \leq m, j \neq 0} \log p(w_{t+j}|w_t) \tag{2.2}$$

Außerdem besitzt das SG für die Normalisierung folgende Softmax-Funktion.

$$p(w_{t+i}|w_t) = \frac{\exp(u'^T_{w_{t+i}} v_{wt})}{\sum_{l=1}^{V} \exp(u'^T_l v_{w_t})} \tag{2.3}$$

Das stehen w_{t+1} für die Kontext-Wörter und w_t für das voraussagende Wort. Hingegen stehen v_{wt} und $u_{w_{t+1}}$ für die Vektoren des Kontextes und der voraussagenden Wörter. Die Softmax- Funktion läuft über den gesamten Korpus V [35].

2.3 GloVe

GloVe ist ein weiteres Word-Embedding-Modell, das von Pennington et.al 2014 vorgestellt wurde und bei dem ein ähnlicher Ansatz wie bei Word2Vec verfolgt wird [30]. Bollegala unterscheidet Word Embeddings in zählbasierte (*Count-Based*) oder direkte Vorhersage (*Direct Prediction*) Modelle [3]. Bei beiden Gruppen gibt es sowohl Vor- als auch Nachteile. Die Stärken der Direct-Prediction-basierten Modelle liegen bei den Analogie-Aufgaben wie z. B. $Kings - Man + Woman \approx Queen$. Zudem können sie komplexe linguistische Merkmale erfassen, die jenseits der Wortähnlichkeiten liegen, und sie erzielen bei zahlreichen Aufgaben eine ausgezeichnete Performance. Dagegen müssen Direct-Prediction-Modelle an die Größe des Korpus angepasst werden, da die Wahrscheinlichkeit für jeden einzelnen Wort-Vektor über den gesamten Korpus immer neu berechnet werden muss (Formel 2.2). Count-Based Modelle können in kürzerer Zeit trainiert werden, vorausgesetzt die gewählte Matrix ist nicht extrem groß. Weiterhin sind die angewandten statistischen Verfahren effizient, da die globale Wort-Wort-Co-Occurence (Tab. 2.1) nur einmal gebildet werden muss und dann immer verwendet werden kann. Durch die Co-Occurence wird hauptsächlich die Syntax eines Satzes übernommen, allerdings werden die semantischen Bedeutungen einzelner Wörter außer Acht gelassen. Des Weiteren erhalten häufig vorkommende Wörter bei der Bildung der Co-Occurence eine unverhältnismäßig starke Bedeutung [21] [28].

GloVe verbindet beide Modelle, das Direct-Prediction-Modell und das Count-Based-Modell [30].]. Somit entsteht eine globale Wort-Wort-Co-Occurence mit der Wahrscheinlichkeit der einzelnen Wort-Vektoren (Tab. 2.2) [28].

Tabelle 2.1: Die Wort-Wort-Co-Occurrence wurde mit den folgenden Sätzen gebildet: „Ich liebe Deutschland“, „Ich liebe Hamburg“ und „Ich studiere Medizin“.

	Ich	liebe	studiere	Hamburg	Deutschland	Medizin	.
Ich	0	2	1	0	0	0	0
liebe	2	0	0	1	1	0	0
studiere	1	0	0	0	0	1	0
Hamburg	0	1	0	0	0	0	1
Deutschland	0	1	0	0	0	0	1
Medizin	0	0	1	0	0	0	1
.	0	0	0	1	1	1	0

$$P_{ij} = \frac{X_{ij}}{X_i} \tag{2.4}$$

Tabelle 2.2: Die Wahrscheinlichkeiten, die bei unterschiedlichen Wörtern k und i ermittelt werden. Die Quotienten in der ersten Spalte ergeben sich aus den Wörtern $i = ice \mid k = solid$ und $i = steam \mid k = solid$ sowie aus dem Quotienten $\frac{P_{k=solid|i=ice}}{P_{k=solid|i=steam}}$. Wörter mit einer hohen Ähnlichkeit haben einen Quotienten über 1. Hingegen ist bei kleiner 1 keine Ähnlichkeit vorhanden [30]. (Quelle: Abbildung [30]).

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Die Wahrscheinlichkeit von Wortpaaren wird über die Formel 2.4 ermittelt. Der Zähler steht für die Anzahl der vorkommenden Wörter i im Zusammenhang mit dem Wort j. Hingegen steht der Nenner für die Anzahl der Häufigkeit, die ein beliebiges Wort im Kontext von Wort i erscheint [28].

Tabelle 2.2 zeigt anhand eines Beispiels, inwiefern Wahrscheinlichkeiten über die globale Wort-Wort-Co-Occurence errechnet werden. In der ersten Zeile werden die Wortpaare *ice* mit einem weiteren Wort k verglichen. Die höchste Wahrscheinlichkeit in der Tabelle besitzt das Wort *ice* mit dem Wort *water*. Der Quotienten $(P_{solid|ice})$ gibt an, ob Wörter sich ähneln (> 1) oder ob sie sich nicht ähneln (< 1), wie in der zweiten Spalte zu erkennen ist. Das Wort *fashion* hat kaum eine Ähnlichkeit mit dem Wort *steam*, hingegen ist die Ähnlichkeit zu *water* viel höher. Befindet sich der Quotient nahe 1, ist das Wort mit einem der beiden Wörter verwandt oder auch nicht verwandt. In dieser Tabelle 2.2 trifft das letztere auf die vierte Spalte.

$$L = \sum_{i=1}^{W} \sum_{j=1}^{W} f(X_{ij})(\vec{u}_j^T v_i - \log X_{ik})^2 \tag{2.5}$$

Die Gleichung 2.5 beschreibt die verwendete Kostfunktion. $\vec{u}_j^T$ ist der Output-Vektor, der wiederum mit dem Wort-Vektor v_i multipliziert wird. Dieser Term wird mit dem Logarithmus von X_{ik} subtrahiert [28].

2.4 Fasttext

Fasttext ist eine Erweiterung von Word2Vec, die im Jahre 2016 von Facebook vorgestellt wurde [2]. Hierbei handelt es sich um eine Open-Source-Bibliothek. Obwohl Fasttext seine Vorhersagen über SG berechnet und dem Aufbau von Word2Vec ähnelt, unterscheidet sich dieses Modell in einem Detail von GloVe und Word2Vec: Die beiden Word Embeddings vernachlässigen die Morphologie des Wortes. Diese Einschränkung nutzt Fasttext, um Vorteile bei einem großen Vokabular und seltenen Wörtern zu gewinnen. Somit soll diese Methode besonders bei größerem Korpus schnell sein. Zusätzlich ist Fasttext in der Lage, Wortdarstellungen zu ermitteln, die nicht im Korpus vorkommen.

Fasttext basiert, wie Word2Vec, auf den zwei Modellen SG und CBOW, denen zusätzlich mit Bag of Character n-grams hinzugefügt werden. Bei Bag of Character handelt es sich um ein Modell, das ein Wort in kleine Sequenzen zerlegt (Abb. 2.6). So kann bei dem Wort *apple* mit einem N-Gram=3 folgendes Ergebnis erzielt werden <ap,app,ppl,ple,le>. Jedes Wort wird durch spitze Klammern von anderen Wörtern abgegrenzt. Jede Zeichenfolge wird als Wort-Vektor deklariert. Daraufhin wird aus der Summe der Zeichenfolge ein Wort-Vektor gebildet. Dies ermöglicht die Darstellung von Wort-Vektoren, die selten sind, und auch von Wörtern, die nicht im Vokabular vorhanden sind [2]. Die daraus resultierende Summe bildet die Vektordarstellung des Wortes [14].

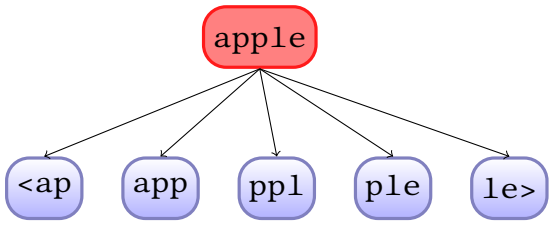


Abbildung 2.6: Der n-Gram ist frei wählbar und dient der Zerlegung des Wortes. Bei der Darstellung handelt es sich um n = 3. Die Zeichen < und > dienen als Worttrennung zu anderen Wörtern. Durch die N-Grams wird das Wort in jeweils 3 aufeinanderfolgende Zeichen getrennt. So bildet das Wort *apple* 5 verschiedene Sequenzen. Die einzelnen Sequenzen eines Wortes bilden nun als Summe die Vektordarstellung des Wortes *apple*.

$$-\frac{1}{N}\sum_{n=1}^{N} y_n \log(f(BAx_n)) \tag{2.6}$$

Die Formel 2.6 beschreibt das Fasttext-Modell. Dabei steht N für das Dokument. A und B stehen für die Gewichtsmatrizen. X_n ist der normalisierte BoW und y_n das dazugehörige Label.

2.4.1 Word Embedding als Implizierte Matrix-Faktorisierung

Das populäre Word2Vec mit SG und der Erweiterung als NS (SGNS) wurde von Mikolov als ein neuronales Netz publiziert [23]. Allerdings hat Levy im Jahre 2014 gezeigt, dass das neuronale Netz keine bedeutsame Rolle spielt, sondern es sich hierbei um eine implizierte Matrix-Faktorisierung handelt [18].

Das SGNS benötigt für die Einbettung die Wort-Matrix W und die Kontext-Matrix C. Folglich erschließen sich die Vokabulare von Wort und Kontext als V_W und V_C. Demzufolge kann SGNS als Faktorisierung einer implizierten Matrix $M = |V_W| \;\; x \;\; |V_C|$ in zwei kleinere Matrizen beschrieben werden. Ein Matrixeintrag M_{ij} entspricht dem Skalarprodukt von W_i und C_j als auch $\vec{w}_i$ und $\vec{c}_j$ [18].

Levy zeigte bei einer ausreichend großen Dimensionalität, dass jedes Skalarprodukt $\vec{w} * \vec{c}$ unabhängig von den anderen Skalarprodukten einen beliebigen Wert annehmen kann. Letzten Endes entsteht folgende Formel 2.7. Dabei kommt der Ausdruck Pointwise Mutual Information (PMI) zustande. Dieser Ausdruck stammt aus der Statistik und dient als Maß der Abhängigkeiten. Somit zeigt dieser statistische Ausdruck, inwiefern ein Wort-Kontext-Paar, das aus dem Korpus entnommen wurde, zu erwarten ist, wenn die Wörter unabhängig voneinander sind. So würden das Wort „*costa*“ und sein Kontext „*rica*“ einen hohen PMI Wert erhalten, während „*New*“ und „*rica*“ einen negativen PMI Wert erreichen würden. Der letzte Term, $\log k$, dient nur als globale Konstante [18].

Dies bedeutet nun, dass Word2Vec eine Matrix-Faktorisierung impliziert, die über ein Skalarprodukt dem Word Embedding mitteilt, wie wahrscheinlich ein Wortpaar auftaucht.

$$M_{ij}^{SGNS} = W_i * C_j = \vec{w}_i * \vec{c}_j = PMI(w_i, c_j) - \log k \tag{2.7}$$

Die Autoren Pennington und Levy weisen darauf hin, dass durch GloVe die Log-Count-Matrix faktorisiert werden soll [18] [30] [19]. Anders als bei Word2Vec wird bei GloVe die Faktorisierung von Bias-Vektoren ergänzt $\vec{b}_w$ und $\vec{b}_c$ (Formel 2.8) [19]. Aufgrund der Log-Zählmatrix wird auch ein Exponent $\gamma \in [0, 1]$ angehoben. Somit gilt die Matrix-Faktorisierung sowohl für Word2Vec als auch für GloVe [40].

$$M^{\log(\#(w,c))} \approx W * C^T + \vec{b}_w + \vec{b}_c \tag{2.8}$$

Es kann gezeigt werden, das die modernen Word Embeddings (Word2Vec, GloVe und Fasttext) durch implizierte Matrix-Faktrisierung ihre Wahrscheinlichkeiten bestimmen können. Im laufe der letzten 70 Jahre wurden Wort-Dokumenten-Matrizen stets über Matrixfaktorisierung bestimmt. Dies ändert sich nun hin zu genererischeren Wort-Wort-Matrizen.

3 Aufbau

In diesem Kapitel werden die einzelnen Schritte für die Verarbeitung des Korpus beschrieben. Weiterhin wird die Zusammenstellung des Korpus erwähnt, die Implementierung der Word Embeddings und die verwendeten Bibliotheken. Schließlich wird der Versuchsaufbau und die verwendeten Hyperparameter, um die Word Embeddings zu trainieren erläutert.

3.1 Implementierung

Die Versuche in dieser Arbeit werden hauptsächlich in Python 3.7.8 geschrieben und ausgewertet. Dies wird unter anderem in der Open-Source-Distribution ANACONDA V. 1.8.4 ausgeführt. Hierfür wird die Entwicklungsumgebung JupyterLab V. 2.1.5 verwendet. Der Programm-Code ist in Github hochgeladen und kann dort auch heruntergeladen werden: GitHub-Code

Die Textverarbeitung, die den Tokenizer und Stoppwörter beinhaltet, wird mit der Bibliothek Natural Language Toolkit V 3.5 (NLTK) realisiert. Die Lemmatisierung erfolgt mit der Germalemma, die vom Berlin Social Science Center zu Verfügung gestellt wird.

Gensim eine Open-Source-Bibliothek hat sowohl Word2Vec V 3.8.3 als auch Fasttext 3.8.3 als Bibliothek implementiert. Dies hat den Vorteil, dass man mittels minimaler Angaben bereits ein gesamtes Word Embedding trainieren kann. GloVe wurde in der Sprache C implementiert, weshalb Gensim diesen nicht anbietet. Für GloVe wird die Open-Source-Distribution von Deepset verwendet. Durch Docker wird die Implementierung von GloVe, die in einem Bash-Skript umgesetzt ist, zum Laufen gebracht. Somit werden die Input-Eingaben über ein Bash-File eingegeben, die daraufhin trainiert werden. Die Visualisierung erfolgt mit t-SNE über die Bibliothek scikit-learn V. 0.23.1. Schließlich wird für das Plotten der Diagramme standardgemäß Matplotlib V. 3.2.2 verwendet.

3.2 Korpus

Das verwendete Korpus stammt aus einer Sammlung unterschiedlicher Literaturbücher aus dem Grundstudium des Humanmedizin-Studiengangs. Tabelle 3.1 zeigt eine detaillierte Übersicht über die Titel des Buches und die Anzahl der Seiten. Die Bücher wurden mittels eines herkömmlichen Online-Converters vom Portable-Document-Format (PDF) in ein Textformat (.txt) konvertiert. Während der Konvertierung wurden jegliche Grafiken, Designs, Layouts, Tabellen und Formatierungen entfernt. Somit enthält der Korpus ausschließlich Textmaterial in Form von Buchstaben. Zusätzlich wurden manuell die Inhaltsangabe, das Literaturverzeichnis und das Vorwort entfernt. Dementsprechend besteht der Korpus nach Zusammensetzung der einzelnen Bücher aus ca. 22,6 Millionen Wörtern. Darüber hinaus befinden sich im Korpus vereinzelt auch englische Begriffe. Diese stammen aus einzelnen Quellenangaben, die nach jedem Kapitel angehängt wurden.

Tabelle 3.1: Zusammengesetzte Literatur, die in dieser Arbeit als Korpus diente.

Titel	**Seiten**
Lehrbuch Histologie: Zytologie, Histologie, mikroskopische Anatomie	630
Biologie des Menschen	910
Lehrbuch der Physiologie	930
Innere Medizin, 2. Auflage duale Reihe	1529
Duale Reihe Pädiatrie, 3. Auflage	1054
Physiologie des Menschen: mit Pathophysiologie	979
Duale Reihe Anatomie	1265

3.3 Textverarbeitung

Für die Verarbeitung des Korpus in eine einheitliche Struktur sind mehrere Schritte vonnöten. Der Korpus wird in einzelne Wort-Token gegliedert und sämtliche Wörter, die keine semantische Bedeutung besitzen, werden entfernt. Dazu zählen unter anderem Stoppwörter und Artikel. Unter Stoppwörter werden Wörter bezeichnet, die für die Informationsgewinnung eines Dokuments keine Relevanz besitzen.

Der zusammengesetzte Korpus besteht aus mehreren Büchern, die vom PDF in ein Textdokument-Format konvertiert wurden. Beim Konvertieren entstehen Formatierungen oder Sonderzeichen, die durch das Öffnen des Dokumentes nicht sichtbar sind. So beginnt der erste Satz im Dokument mit „I Allgemeine Physiologie der Zelle“. Doch beim Auslesen des Dokumentes in UTF-8 sind Sonderzeichen zu sehen, die für das NLP und für das weitere Vorgehen keine essentielle Bedeutung haben und entfernt werden müssen (Tab. 3.2). Dadurch wird unter anderem eine klare Übersicht über die einzelnen Sätze erleichtert. Hierbei kann auf die Bibliothek NLTK zurückgegriffen werden, die in der Lage ist, eine Tokenisierung nach Sätzen oder nach Worten durchzuführen. Dementsprechend wird der Korpus im ersten Schritt nach Sätzen tokenisiert (Tab. 3.3). Aus Tabelle 3.3 wird ersichtlich, dass die Trennung der Sätze nicht ausschließlich durch einen Punkt erfolgen kann, sondern auch durch ein anderes Satzzeichen wie ein Fragezeichen. Dagegen kann der Tokenizer den Übergang von einer Überschrift zu einem Satz nicht trennen, wie aus der ersten Zeile zu erkennen ist (Tab. 3.3).

Tabelle 3.2: Der Original-Text zeigt die erste Überschrift im Korpus an. Beim Auslesen des Dokumentes in UTF-8 sind zusätzlich Sonderzeichen zu lesen, die aufgrund der Formatierung entstehen.

Original-Text	**ausgelesener Text**
I Allgemeine Physiologie der Zelle	\r\n\r\n\nI Allgemeine Physiologie der Zelle\n\n

Tabelle 3.3: Korpus wird mittels NLTK-Tokenizer durch Satzzeichen in einzelne Sätze unterteilt.

Ohne NLTK-Tokenizer	Mit NLTK-Tokenizer
\n\r\n\nl Allgemeine Physiologie der Zelle \n\n> > Einleitung\n\nl\n Aus wie vielen Zellen besteht eigentlich der Mensch? 25 Billionen \n(25 x 1012) rote Blutzellen transportieren den Sauerstoff der Luft\nvon der Lunge ins Gewebe.75 Billionen weitere Zellen bauen unseren Körper auf, dies ergibt also 100 Billionen Zellen insgesamt.	\n\r\n\nl Allgemeine Physiologie der Zelle \n\n> > Einleitung\n\nl\n Aus wie vielen Zellen besteht eigentlich der Mensch?
	25 Billionen \n(25 x 1012) rote Blutzellen transportieren den Sauerstoff der Luft\nvon der Lunge ins Gewebe.
	75 Billionen weitere Zellen bauen unseren Körper auf, dies ergibt also 100 Billionen Zellen insgesamt.

Weiterhin müssen die diakritischen Zeichen, die in der deutschen Sprache vorkommen, ersetzt werden (Tab. 3.4). Zeitgleich werden sämtliche Zeichenketten oder Satzzeichen entfernt (Tab. 3.5). Zudem wird jedes Wort in einem Satz mit einem Kommazeichen getrennt. Um die Anzahl der Wörter zu verringern und bedeutungslose Wörter, die keinen Bezug zur Semantik haben, zu entfernen, müssen Stoppwörter gelöscht werden. Die Abb. 3.1 zeigt die 10 häufigsten Stoppwörter, die im Korpus vorkommen. Abbildung 3.1 wird durch die bestimmten Artikel *der* und *die* geführt. Schließlich sieht ein Satz im Korpus wie in Tabelle 3.5 dargestellt aus.

Jedoch befinden sich nach dem Entfernen der Stoppwörter weiterhin Wörter wie „Die“ oder „Dabei“. Das liegt daran, dass der Anfangsbuchstabe des Stoppwortes großgeschrieben ist. Die Stoppwörter-Liste erkennt nur sensitive Wörter. Bei einem Rechtschreibfehler oder sonstigen Abweichungen wird das Wort nicht als Stoppwort erkannt und somit nicht entfernt. Dies kann umgangen werden, indem der gesamte Korpus aus Kleinbuchstaben bzw. *Lowercase-Buchstaben* besteht. Dieser Schritt wird bewusst nach dem Entfernen der Stoppwörter durchgeführt, da der Korpus noch lem-

matisiert werden muss. Der Lemmatisierungsschritt benötigt bei einem Korpus von ca. 22,6 Millionen Wörtern mehrere Tage. Bei diesem Prozess wird jedes Wort durch sein Basismorphem ersetzt. So wird aus *läuft laufen* oder *Zellen* wird mit *Zelle* ersetzt (Tab. 3.6). Darüber hinaus werden Wörter, die im Plural stehen, durch das Singular-Wort ersetzt. Jedoch zeigt Tabelle 3.6, dass der Lemmatisierer nicht jedes Wort umwandelt. So wird das Wort „Bakteriums“ nicht durch sein Substantiv „Bakterium“ ersetzt. Der Lemmatisierer *GemmaLemma* soll eine Genauigkeit von bis zu 87 % verfügen [25].

Schlussendlich wird jedes Wort durch einen kleinen Anfangsbuchstaben ersetzt. Diese Phase muss zum Schluss erfolgen, da sonst die Lemmatisierung fehlschlägt. Abschließend können die restlichen Stoppwörter entfernt werden. Der Korpus besitzt nun nur noch 1,5 Millionen Wörter.

Tabelle 3.4: Die diakritischen Zeichen der deutschen Sprache werden durch Vokale ersetzt, um Fehlerwahrscheinlichkeiten zu minimieren.

diakritische Zeichen	**ersetzte diakritische Zeichen**
Ä, ä	Ae, ae
Ü, ü	Ue, ue
Ö, ö	Oe, oe
ß	ss

Tabelle 3.5: Die rechte Spalte zeigt, wie ein Satz im Korpus untergliedert wird. Jedes Wort wird in Anführungsstriche gesetzt und semantische Wörter ohne Bedeutung werden entfernt.

ungefiltert	gefiltert
Dabei werden Pizza, Würstchen und Schokolade in Energie umgewandelt und die Abfallprodukte in die umgebende Flüssigkeit abgegeben.	'Dabei', 'Pizza', 'Wuerstchen', 'Schokolade', 'Energie', 'umgewandelt', 'Abfallprodukte', 'umgebende', 'Flüssigkeit', 'abgegeben'
Die Flüssigkeit in den Zellen unterscheidet sich drastich von \nder extrazellulären Flüssigkeit.	'Die', 'Fluessigkeit', 'Zellen', 'unterscheidet', 'drastich', 'extrazellulären', 'Fluessigkeit'
So besitzt jede Zelle eine Art Grundausstattung, die für\alle Zellen etwa gleich ist.	'So', 'besitzt', 'Zelle', 'Art', 'Grundausstattung', 'Zellen', 'gleich'

Tabelle 3.6: Nach der Lemmatisierung werden alle Wörter mit einem kleinen Anfangsbuchstaben ersetzt.

ohne Lemmatisierung	mit Lemmatisierung
'Eine', 'Muskelzelle', 'kontrahiert', 'Nervenzelle', 'informiert', 'Nierenzelle', 'transportiert'	'muskelzelle', 'kontrahieren', 'nervenzelle', 'informieren', 'nierenzelle', 'transportieren'
'Als', 'Beispiel', 'Phagozytose', 'Bakteriums', 'Autophagie', 'defekten', 'Mitochondrions', 'Endozytose', 'Makromolekuelen', 'extrazellulaeren', 'Umgebung', 'gezeigt'	'beispiel', 'phagozytose', 'bakteriums', 'autophagie', 'defekt', 'mitochondrions', 'endozytose', 'makromolekuelen', 'extrazellulaere', 'umgebung', 'zeigen'
'finden', 'globulaere', 'Proteine', 'Synthese', 'jeweiligen', 'Bestimmungsorte'	'finden', 'globulaer', 'proteine', 'synthese', 'jeweilig', 'bestimmungsort'

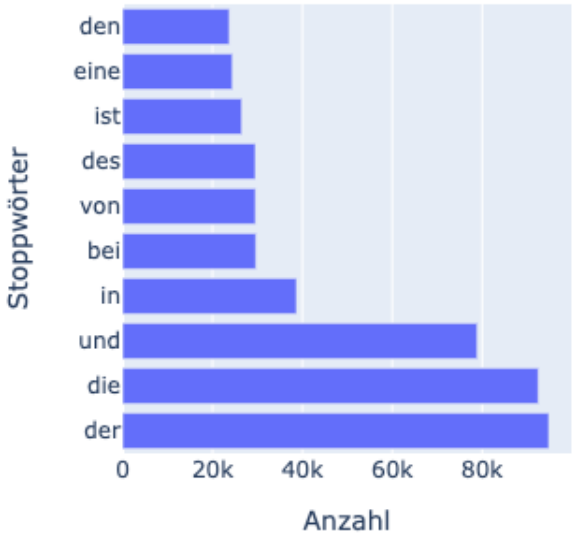


Abbildung 3.1: Zeigt die Häufigkeit der Stoppwörter im verwendeten Korpus.

3.4 Hyperparameter

In diesem Abschnitt handelt es sich um die Erklärung der einzelnen Hyperparemeter, die für die Verwendung der Word Embedding notwendig waren.

size Die Dimensionalität (*size*) repräsentiert die Gesamtzahl der Merkmale, die in einem Wort-Vektor kodiert sind [16]. Sie hat einen großen Einfluss auf die Performance von Word Embeddings [42].
Eine hohe Dimension führt dazu, dass fast alle Merkmale aus den Wörtern gelernt werden. Dies verursacht wiederum eine schnellere Überanpassung bzw. Overfitting. Außerdem besteht die Gefahr, dass die Visualisierung mit t-SNE zum erwähnten Fluch-der-Dimensionalität-Problem führt. Dieses Phänomen findet bei Wort-Vektoren mit hohen Dimensionen statt, die aufgrund der Visualisierung von mehrdimensionalen zu zweidimensionalen Diagrammen reduziert werden. Hingegen kann bei einer geringen Dimension der Fall auftreten, dass die semantischen Merkmale und Beziehungen der Wort-Vektoren nicht erlernt werden können [42] [41].

Zudem ist die Größe der Dimensionalität eine lineare oder quadratische Funktion, die die Trainingszeit und Rechenzeit erheblich verlängert. Hauptsächlich führen hohe Dimensionalitäten zu einer Modell-Komplexität und zur Verlangsamung der Trainingsgeschwindigkeit. Dies führt ausnahmslos dazu, dass die Anwendbarkeit und der Einsatz eines Word Embeddings eingeschränkt werden [39].

Die Auswahl der Dimensionalität in der NLP ist ein weltweit vorkommendes Problem. Dabei wird die Größe der Dimensionalität nach Gefühl oder nach empirischen Versuchen gewählt, die wiederum nur eine sub-optimale Performance des Word Embeddings hervorrufen [41]. Oren Melamud et.al 2016 hat bereits in seinen Forschungen bewiesen, dass bei intrinsic Tasks der Peak bei einer Dimension um 300 liegt [22]. Zudem wird dieser Wert häufig in wissenschaftlichen Arbeiten verwendet [4] [24] [30]. Die Anzahl der Dimensionen ist auch von Korpus zu Korpus abhängig [27].

Für diese Arbeit wurde eine Dimension von 300 gewählt. Diese Zahl kam dadurch zustande, dass die Word Embeddings mit unterschiedlichen Dimensionen trainiert wurden. Anschließend wurden sie mittels Analogie-Test oder Wortähnlichkeiten evaluiert. Somit wurde die Dimension gewählt, die die passendsten Ergebnisse darstellte. Diese lag bei 300.

window Window gibt die maximale Fenstergröße an. Das Fenster wandert über jeden Satz und berechnet den maximalen Abstand zwischen den aktuellen Wörtern und dem prognostizierten Wort. Dieser Wert soll möglichst groß sein, damit möglichst viele semantische Merkmale übernommen werden können [31]. Diese führen gleichzeitig zu einer gesteigerten Genauigkeit des Word Embeddings. In einer weiteren Arbeit wurde für Word2Vec herausgefunden, dass bei einem Window von über 10 keine signifikanten Veränderungen stattfinden [27]. Dieser Wert ist ebenfalls vom Aufbau und der Verarbeitung des Korpus abhängig. Er wurde auch empirisch ermittelt und liegt bei 10.

min_ count Ein Word Embedding bildet ein Wörterbuch aus allen Wörtern, die im Korpus vorhanden sind. Dies hat den Vorteil, dass man Wörter, die nur eine geringe Häufigkeit haben, herausfiltern kann. Seltene Wörter wie z. B. Augenweide kommen statistisch gesehen nicht häufig vor und beeinflussen kaum die semantischen Merkmale einzelner Wörter. In der NLP können Wörter, die nur eine geringe Präsenz aufweisen, entfernt werden [27]. Selbst dieser Wert variiert vom Korpus. Üblicherweise werden

alle Wörter, die weniger als fünfmal auftauchen, nicht einbezogen. Da dieser Wert erheblich die Dauer des Trainingsverlaufs beeinflusst, wird er bei diesem Versuch auf 10 gesetzt.

SG Bei diesem Hyperparameter ist nur zwischen CBOW und SG zu entscheiden. Beim CBOW wird das ausgegebene Wort anhand der benachbarten Wörter statistisch erfasst. Beim SG hhingegen werden die benachbarten Wörter anhand eines Wortes erfasst. CBOW überzeugt bei großen Trainingsdaten und bei einer höheren Präsenz an Wörtern. SG dagegen zeigt bei kleineren Trainingsdaten und bei seltenen Worten eine überzeugendere Leistung [32]. Die verwendeten Trainingsdaten stammen ausschließlich aus der medizinischen Literatur. Dadurch kommen seltene Wörter häufiger vor als in einem üblichen Text. Somit lässt sich diese Arbeit mit dem SG-Modell realisieren.

sample Die Samplerate steht für den Threshold. Sie gilt für Wörter, die eine erhöhte Präsenz aufweisen. Wird der festgelegte Threshold eines Wortes überschritten, wird er „gedownsampled“. Als Standartwert gilt e^{-3}. Darüber hinaus hat dieser Standartwert die überzeugendste Performance bei der Ähnlichkeitsbewertung gezeigt [4]. Dieser Wert wurde auch in dieser Arbeit verwendet.

Epoche Jede Trainingsiteration über den gesamten Textkorpus wird als Epoche bezeichnet. Somit bedeutet eine n Epoche, dass der Algorithmus n-mal über den ausgewählten Textkorpus iteriert. Die Epochenanzahl stellt einen wesentlichen Indikator für das Overfitting dar [17]. Aus wissenschaftlichen Arbeiten war zu entnehmen, dass Epochen über 15 bei Word2Vec nicht signifikant sind [27]. So wurde die Epoche für alle drei Word Embeddings empirisch ermittelt. Da das Overfitting bei allen unterschiedlich stark geprägt war, musste die Epoche unterschiedlich gewählt werden. So ergab sich bei Word2Vec eine Epoche von 16, bei Fasttext 11 und schließlich bei GloVe eine Epoche von 50.

3.5 Versuchsablauf

Für einen transparenten Vergleich wird nur ein Korpus für alle Word Embeddings benutzt. So wird beim Ähnlichkeitstest in vier Kategorien aus der Medizin ausgewertet. Es werden die Kategorien Erkrankung, Symptome, Medikamente und Drogen beurteilt. Jede Kategorie besteht aus drei weiteren Wörtern. Es werden lediglich die Top-3-Antworten der Word Embeddings angezeigt (Tab. 4.3). Die Wörter, die fett markiert sind, liegen dem gesuchten Wort in der entsprechenden Kategorie am nächsten. Diese wurden nicht rechnerisch, sondern intuitiv ermittelt. So wurde geprüft, inwiefern die drei ausgegebenen Wörter eines Word Embeddings dem gesuchten Wort ähneln. Hierbei wurde sowohl auf die Semantik als auch auf die Syntax geachtet. So sind z. B. im ersten Fall in Tabelle 4.3 für das Wort **diabetes** die Wörter von Word2Vec als am nächsten gewählt worden. Dies liegt daran, dass **diabetes** in der Literatur nicht alleine vorkommt, sondern aus zwei Wörtern besteht. So lag Word2vec sowohl mit **diabetes-mellitus** als auch mit **diabetes-insipidus** nahe am gewünschten Ergebnis, denn beide Erkrankungsformen kommen vor. Bei GloVe und Fasttext dagegen wurde nur ein präziser Treffer ausgegeben.

Für einen tieferen Eindruck in den Ähnlichkeitstest wird für jede Kategorie ein Diagramm mit den umliegenden Wort-Vektoren dargestellt. Dieses soll verdeutlichen, welche Wörter noch in der Umgebung des gesuchten Wortes zu finden sind.

Ein weiterer Testvorgang ist der Analogie-Test. Folglich werden sechs Analogie-Fragen gestellt, die ich selbst formuliert habe. Dabei werden ein Wortpaar und ein weiteres Wort vorgegeben. Die Word Embeddings sollen nun den passenden Partner für das einzelne Wort finden. Im Gegensatz zum Ähnlichkeitstest wird nur das erste Wort ausgewertet. Auch hier gilt das fett markierte Wort als das dem gesuchten Wort am nächsten.

Schließlich werden die Word Embeddings mit t-SNE visualisiert. Somit können die Plots als Ganzes betrachtet und subjektiv begutachtet werden. Dabei wird überprüft, inwiefern und wie viele Cluster gebildet wurden. So kann anhand der Cluster hergeleitet werden, inwiefern sich Wort-Vektoren aufgrund der Semantik und aufgrund der Syntax im Raum verteilt ähneln. Weiterhin wird ein Plot dargestellt, in dem die Bildung von Clustern nur mit spezifisch ausgewählten Worten untersucht wird. Dieser Plot

visualisiert 15 ausgesuchte Wörter. Zu jedem Wort werden die Top 20 der ähnlichen Wörter dargestellt. Die 15 ausgesuchten Wörter sind intuitiv gewählt worden. Diese Vorgehensweise gibt einen tieferen Einblick, wie sich Cluster des jeweiligen Word Embeddings bilden und wie sie zueinander stehen.

Abschließend werden mehrere Plots folgen, die die Bildung und Entwicklung von Clustern nach Erhöhung der *Perplexity* im Sinne eines Zeitraffers darstellen.

4 Evaluation

Im letzten Kaptiel werden die Ergebnisse aus den Versuche dargestellt. Darüber hinaus wird im Fazit geprüft, inwiefern die Forschungsfrage beantwortet wird. Im Abschnitt Diskussion werden die Ergebnisse reflektiert und kritisch untersucht. Im Ausblick hingegen werden weitere Ansätze, die aufgrund des Zeitmangels und der begrenzten Zeit, die für diese Arbeit fesgelegt ist, beschrieben.

4.1 Ergebnisse

In diesen Unterkaptiel werden die Ergebnisse des Ähnlichkeitstest, des Analogie Tests und die zusätzliche Visualisierung der einzelnen Word Embeddings durch t-SNE. Dabei werden hauptsächlich nur die Ergebnisse vermittelt. Zu den Ähnlichkeitstest werden für einen besseren Rundblick t-SNE Diagramme verwendet, die die Verteilung der Wort-Vektoren darstellen. Bei der Visualisierung werden Zeitraffer Diagramme vorgestellt, die den Verlauf der Cluster Bildung anhand unterschiedlicher Perplexity darstellt.

4.1.1 Ähnlichkeitstest

Alle drei Word Embeddings wurden mit einem zusammengesetzten Textkorpus trainiert und evaluiert. Der Textkorpus ist eine Ansammlung von unterschiedlichen medizinischen Büchern. Die Hyperparameter wurden für alle Word Embeddings nahezu identisch gewählt, insofern gibt es nur minimale Unterschiede. Zusammengefasst finden sich die Hyperparameter für das Antrainieren des jeweiligen Word Embeddings in Tabelle 4.1. Für die Visualisierung und die Reduzierung der Dimensionalität zu einem zweidimensionalen Raum wurden die Hypermeter von t-SNE in der folgenden Tabelle 4.2 aufgelistet.

Dass nicht alle Word Embeddings mit den gleichen Hyperparametern trainiert wurden, liegt sowohl an den Empfehlungen für Hyperparameter aus der wissenschaftlichen Literatur als auch an den Schlussfolgerungen der empirischen Versuche am Korpus. So

wurde die Empfehlung für Word2Vec aus [29] [27] [5] und [8] entnommen, für GloVe aus [19].

Tabelle 4.1: Hyperparameter, die für das Antrainieren des Word Embeddings genutzt wurden.

Hyperparameter	**Word2Vec**	**GloVe**	**Fasttext**
Size	300	300	300
Min_count	7	7	7
Window	10	4	10
Epoche	16	50	11

Tabelle 4.2: Hyperparameter für die Visualisierung der Wort-Vektoren in t-SNE.

Hyperparameter	**Word2Vec**	**GloVe**	**Fasttext**
Perplexity	30	30	50
N_iter	3000	3000	3000
Init	pca	pca	pca
Learning_rate	200	500	1000

Die Ergebnisse des Ähnlichkeitstests sind in Tabelle 4.3 zusammengefasst. Fasttext lag bei 7 von 12 gesuchten Wörtern am nächsten zum gewünschten Ergebnis. Nur knapp dahinter lag Word2Vec mit 5 gesuchten Wörtern. Bei der Kategorie **Medikamente** hat Word2Vec ausnahmslos Wörter ausgegeben, die dem gesuchten Wort von der Semantik am nächsten sind. Auch in den Kategorien **Erkrankung** und **Symptome** konnte Word2Vec passende Wörter ausgeben, die sich ähneln.

Tabelle 4.3: Die Ergebnisse des Ähnlichkeitstests zusammengefasst.

Kategorie	Wort	Word2Vec	GloVe	Fasttext
Erkrankung	diabetes	**mellitus** **insipidus** **mellitusen**	mellitus dependent mellitusen	diabetesmellitus mellitus diabetestyp
	kardiakarzinom	zenkerdivertikel analfissur hodentorsion	durchwandern oesophagus korpus	**karzinom** **nnrkarzinom** **mammakarzinom**
	thorax	atemapparat gleichgewichtslage ppl	dehnungskurve gleichgewichtslage lunge	**fassthorax** **thoraxraum** **thoraxwand**
Symptome	atemnot	**atemabhaengig** **husten** **stoehnen**	husten throraxschmerz anfallsartig	atem atemzuege atemtiefe
	fieber	schuettelfrost krankheitsgefuehl halsschmerz	schuettelfrost bsymptomatik gelbfieber	**qfieber** **fieber** **maltafieber**
	muedigkeit	abgeschlagenheit leistunsschwaeche ohrensausen	abgeschlagenheit schwaeche gewichtsverlust	**tagesmuedigkeit** **appetitlosigkeit** **nackensteifigkeit**
Medikamente	ibuprofen	**indometacin** **metamizol** **diclofenac**	glukosetoleranztest antirheumatika paracetamol	diclofenac ciprofloxacin carbimazol
	metformin	**oad** **basalinsulin** **sulfonylharnstoff**	mgden alkylanzien herpesvirus	methanol metoprolol methotrexat
	diclofenac	**metamizol** **indometacin** **nitrofurantoin**	meropenem acylaminopencillin succedaneum	ibuprofen clopidogrel ethambutol
Drogen	opiate	amphetamin kokain cannabinoide	sinusknotenzelle positivinotrop erwuenscht	**opioide** **amphetamin** **barbiturate**
	nikotin	genussmittel alkohol kaffee	alkohol koffein nikotinkonsum	**nikotinabusus** **nikotinisch** **nikotinsaeure**
	alkohol	nikotin genussmittel kaffee	nikotin droge koffein	**alkoholgenuss** **alkoholiker** **alkoholkonsum**

4.1.1.1 Erkrankung

Beim Wort **diabetes** liegt Word2Vec am nächsten zum gewünschten Ergebnis. Für einen tieferen Einblick dient die Abb. 4.1. Dort wird ersichtlich, dass durch Word2Vec neben dem Wort **diabetes** weitere Begriffe gefunden wurden, z. B. **glukosetoleranz** oder **glukosebelastung**.

Bemerkenswert ist, das alle drei Word Embeddings die Beziehung zwischen **diabetes** und **mellitus** erkannt haben. So besitzen in Abb. 4.1 beide Datenpunkte nahezu die identischen Kooridinaten.

Bei den beiden letzten Wörtern in der Kategorie **Erkrankung** lag Fasttext am nächsten zum gewünschten Ergebnis.

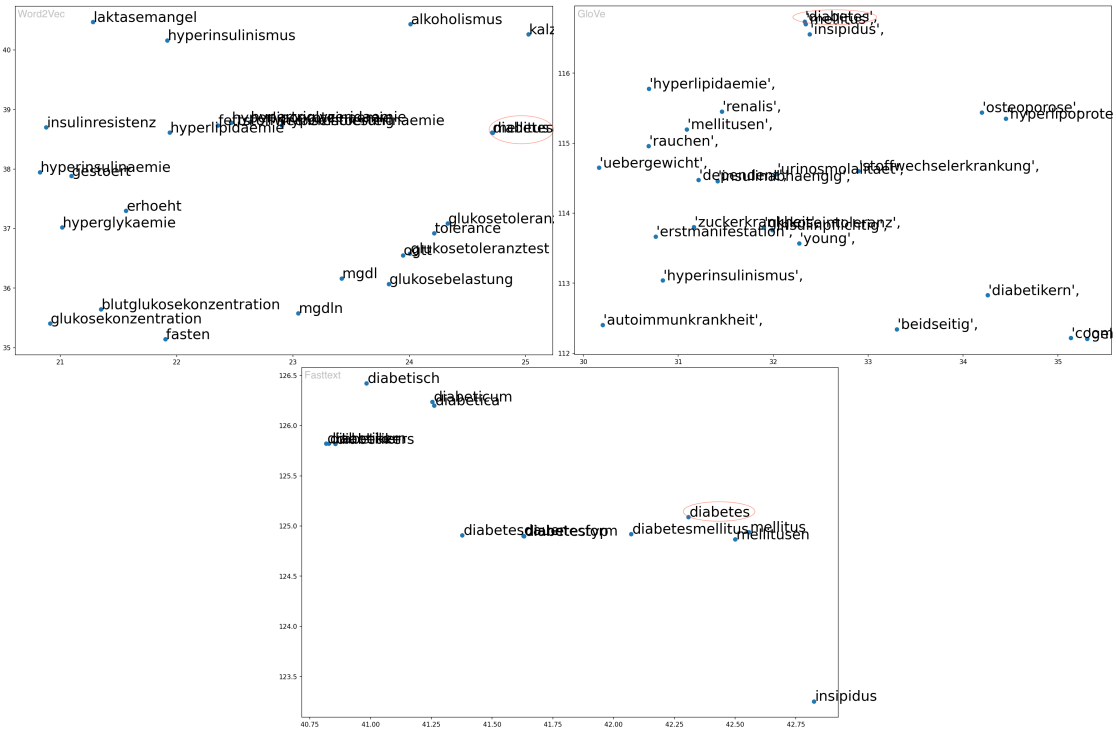


Abbildung 4.1: Visualisierung der Word Embeddings für das Wort **diabetes**.

4.1.1.2 Symptome

In der Kategorie **Symptome**, konnte Fasttext erneut aufgrund der Unterwort- Informationen passendere Ergebnisse zeigen, als die anderen beiden Modelle. Dass beim Wort **fieber** Fasttext selbst **fieber** ausgibt, ist besonders bemerkenswert. Blickt man auf die Abb. 4.2 wird klar, das Fasttext noch weitere Wörter mit dem Präfix **fieber** gebildet hat. So finden sich dort unter anderem **fieberhaft**, **fiebersenkung** und **fieberverlauf**.

Nur bei **atemnot** konnte Word2Vec passendere Wörter vorschlagen als Fasttext. Obwohl Fasttext erneut Wörter gebildet hat, die das Präfix **atem** enthielten, geht dieser Punkt an Word2Vec. Die Semantik spielt doch eine ausschlaggebende Rolle.

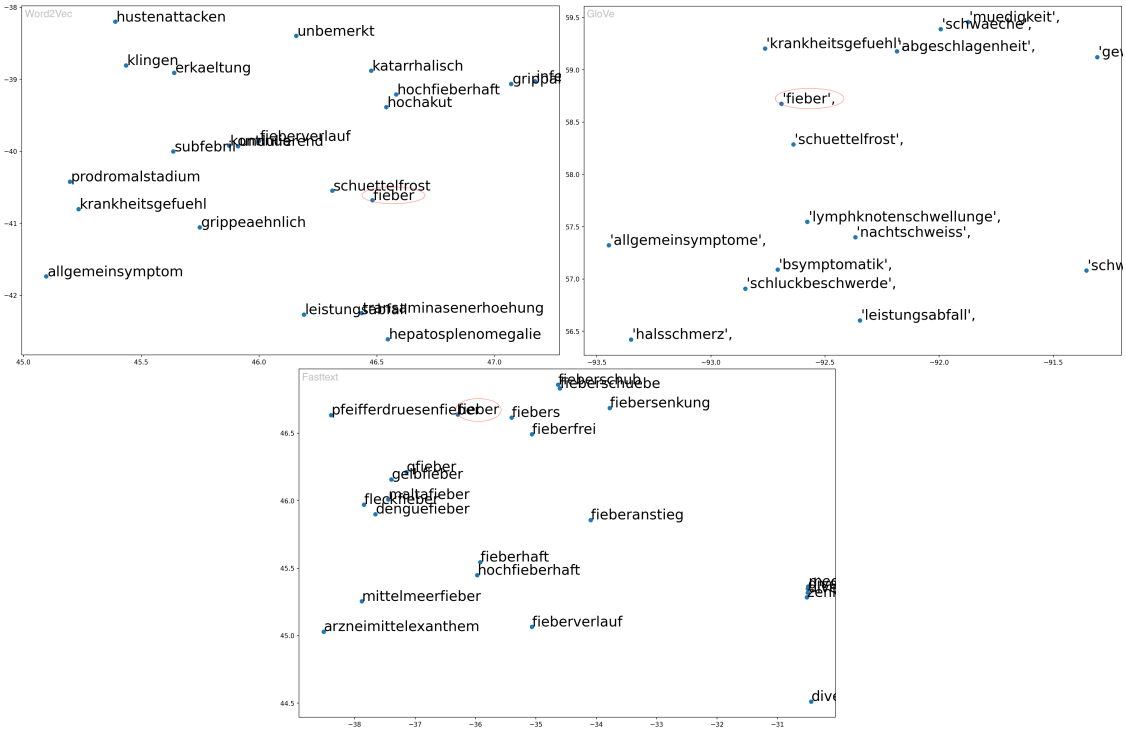


Abbildung 4.2: Visualisierung der Word Embeddings für das Wort **fieber**.

4.1.1.3 Medikamente

In der Kategorie **Medikamente** hat Word2Vec passendere Wörter ausgegeben. So hat Word2Vec beim Wort **ibuprofen** nur Medikamente vorgeschlagen, die ebenfalls zur Gruppe der Analgetika gehören. Fasttext lag ebenfalls nahe am gewünschten Ergebnis, doch **ciprofloxacin** ist ein Antibiotikum. Ein Blick auf die Abb. 4.3 zeigt, das zusätzlich Wörter wie **schmerzlinderung, schmerzmittel** und die schmerzstillende und fiebersenkende **acetylsalicylsäure**, die unter anderem in Aspirin zu finden ist, vorkommen. Bei den Wörtern **metformin** und **diclofenac** verhält es sich nicht anders. Word2Vec hat in der Kategorie **Medikamente** die passendsten Ergebnisse vorgelegt.

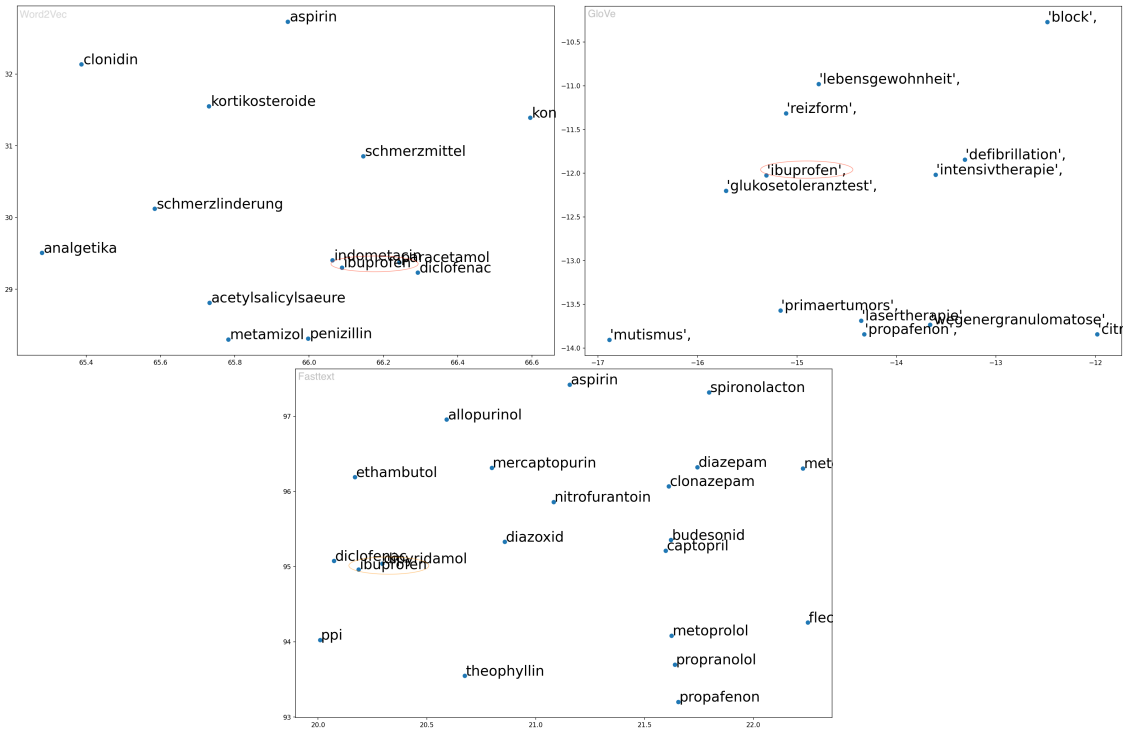


Abbildung 4.3: Visualisierung der Word Embeddings für das Wort **ibuprofen**.

4.1.1.4 Drogen

In der letzten Kategorie **Drogen** hat sich von den drei Word Embeddings Fasttext durchgesetzt. Hierbei fällt auf, dass Fasttext nicht nur anhand des Prä- und des Suffixes seine Wörter bildet, sondern auch anhand der Semantik. Dies wird beim Suchwort **opiate** deutlich. Auch in Abb. 4.4 fällt die Gewichtung auf Fasttext. Dort finden sich neben anderen Drogen auch **lsd**, **cocain** und **ecstasy**.

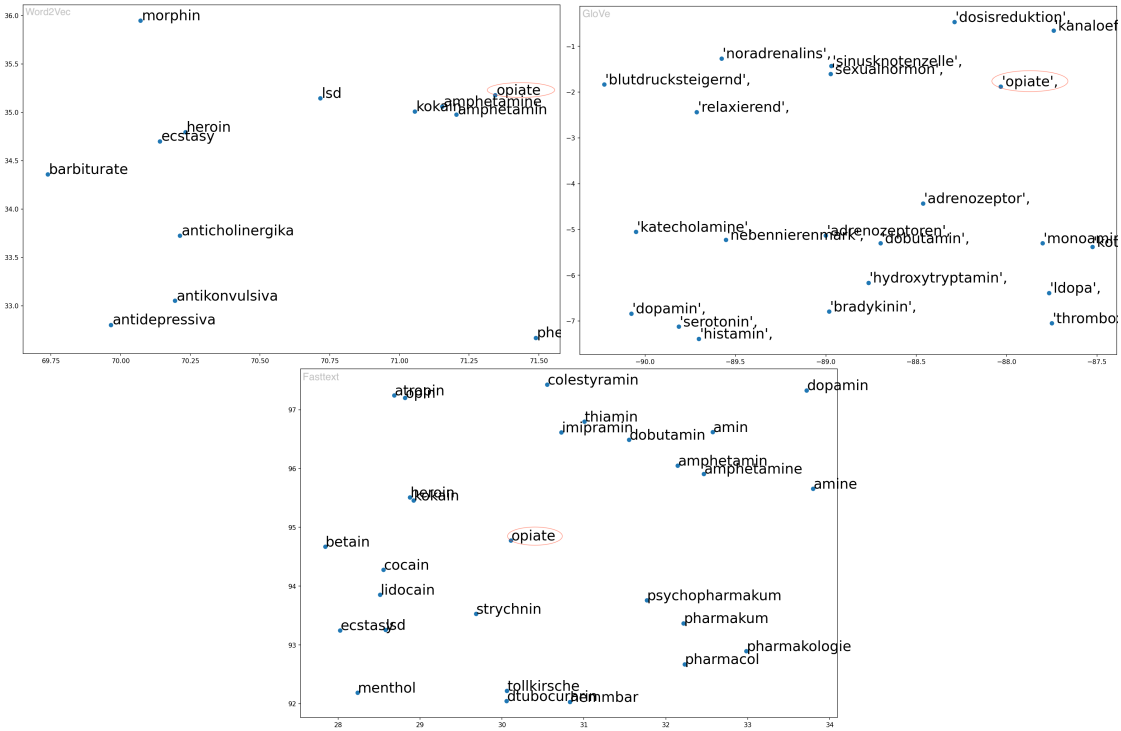


Abbildung 4.4: Visualisierung der Word Embeddings für das Wort **opiate**.

4.1.2 Analogie

Im Analogie-Test liegt Word2Vec eindeutig vorne und erzielt die passendsten Analogie-Wortpaare (Tab. 4.4). Im ersten Analogie-Test handelt es sich um Bakterien und ihre Lipidmembran, die in gramnegativ und grampositiv unterschieden werden kann. Hier sollte ein gramnegatives Bakterium ausgegeben werden. GloVe hat mit **meningokokken** als einziges Modell ein Bakterium ausgegeben, das gramnegativ ist. Das Ergebnis von Word2Vec hingegen war mit **influenza** nicht ausreichend überzeugend, obwohl ein gramnegatives Bakterium mit dem Namen Haemophilus influenzae existiert.

Obwohl Word2Vec in den restlichen Analogien Fragen passendere Wortpaare ausgegeben hat, muss jedes einzeln begutachtet werden. Bei der zweiten Frage wird nach Zellpaaren gesucht. In diesem Fall gehört **megakaryozyt** zu der Zelle **thrombozyt**. Zu diesem Analogie-Paar soll nun eine Zelle vom Typ eines **lymphozyt** genannt werden. Mit **nkzellen**, die zu den natürlichen Killerzellen und zu den lymphatischen Zellen gehören, hat Word2Vec einen passenden Treffer gelandet.

Beim dritten Analogie-Test wird nach dem Organ **leber** und der dazugehörigen Erkrankung **leberzirrhose** gefragt. GloVe gibt **tachypnnoe** an, was eine erhöhte Atemfrequenz bedeutet. Lungenödem ist eine Bezeichnung für eine Flüssigkeitsansammlung in der Lunge. Auch wenn Word2Vec mit **lungenkrankheit** erneut am nächsten liegt, wäre die Antwort **pneumonie** doch eher wünschenswert gewesen.

Mit der nächsten Frage wird nach einem Sinnesorgan und der darauffolgenden Funktion gesucht. So wird der **fuss** benötigt, um zu **gehen**. Dementsprechend benötigt man das Auge, um zu **sehen**. Bei dieser Kategorie liegt Word2Vec mit **augenhaelfte** nur ausreichend neben dem gesuchten Wort.

HIV als humanes Immundefizienz-Virus ist der Erreger, der unter Viren klassifiziert wird. Somit wird ein Erreger gesucht, der unter Bakterien klassifiziert wird. GloVe gibt mit **ceftriaxon** ein Antibiotikum aus. Fasttext gibt aufgrund seiner Teilwort-Information nur das Wort **bakteriums** aus. Word2Vec liegt mit **treponema** nahe am gewünschten Ergebnis.

Die letzte Analogie-Frage sucht einen gut- und einen bösartigen Tumor. Während **adenokarzinom** zu den gutartigen Tumoren (**benigne**) gehört, wird nun ein bösartiger

Tumor (**maligne**) gesucht. **Barrettoesophaggus** ist eine Vorstufe zum Tumor. Somit kann dies als bösartiger Tumor ausgewertet werden.

Abschließend kann gesagt werden, das Word2Vec in mehr als vier Kategorien sehr gute Analogie-Paare ausgegeben hat.

Tabelle 4.4: Die Ergebnisse des Analogie-Tests zusammengefasst.

Analogien	**Word2Vec**	**GloVe**	**Fasttext**
pneumokokken=grampositiv, **?** = gramnegativ	influenza	**meningo-kokken**	pneumokokken-pneumonie
megakaryozyten=thrombozyt, **?** = lymphozyt	**nkzellen**	deletionen	lymphozyten
leberzirrhose = leber, **?** = lunge	**lungen-krankheit**	tachypnoe	lungenoedems
fuss = gehen, **?** = sehen	**augenhaelfte**	sonne	fussnote
hiv = virus, **?** = bakterium	**treponema**	ceftriaxon	bakteriums
adenokarzinom = benigne, **?** = maligne	**barrett-oesophagus**	frueh-geburt	malignom

4.1.3 Visualisierung

In diesen Unterabschnitt folgt nun die Visualisierung von Word2Vec, GloVe und Fasttext. Es folgen Plots über die trainierten Word-Vektoren der Word Embeddings. Gefolgt von Plots, die die Cluster-Bildung grafisch dokumentieren.

4.1.3.1 Word2Vec

In Abb. 4.6 sind alle Wort-Vektoren, die mit Word2Vec trainiert wurden, in einem Diagramm abgebildet. Auffällig sind die großen Cluster in der Mitte. Dies führt zu der Annahme, dass ein geringer Grad an Overfitting stattgefunden hat. Ansonsten fallen einige kleinere Cluster-Wolken auf, in denen sich meist Wörter befinden, die die gleiche bzw. eine ähnliche Bedeutung besitzen. Obwohl es sich um deutsche medizinische Texte handelt, so befindet sich darunter ein Cluster, das nur aus englischen Worten besteht. Diese befinden sich in der Literatur in der Regel am Ende eines Kapitels, häufig als Verweis auf eine Quelle (Abb.4.5).

Einen genaueren Eindruck über Cluster-Bildung gibt Abb. 4.6. Dort ist die Cluster-Bildung von einigen ausgesuchten Wörtern abgebildet. Word2Vec ist durch bestimmte Begriffe in der Lage, geeignete Cluster zu bilden. Wörter wie **nikotin** und **diclofenac**, die sich auch in der Bedeutung ähneln, finden sich beide in Clustern nahe beieinander. Ein aussagekräftiges Cluster hat sich um das Wort **small** gebildet. Dieses besteht nur aus englischen Begriffen. Nur die Cluster, die zur **haut** gehören, haben sich über **karzinom** und **epiglottes** verteilt. Das mag daran liegen, dass die Haut das größte Organ des menschlichen Körpers ist und somit überall Berührungspunkte besitzt.

In Abb. 4.7 ist die Entstehung der Cluster-Wolken nach unterschiedlichen *Perplexity* Schritten abgebildet. Die zu den Clustern gehörenden Farben sind konvergent zu den Farben in Abb. 4.6.

Bei erhöhter *Perplexity* neigen sich die Cluster-Wolken zum Zentrum hin. Somit besteht auch hier die Gefahr, auf Overfitting zu stoßen. Ebenso ist klar zu erkennen, dass die Cluster-Wolke **nikotin** und **diclofenac** nebeneinander liegen und bei erhöhter *Perplexity* miteinander fusionieren.

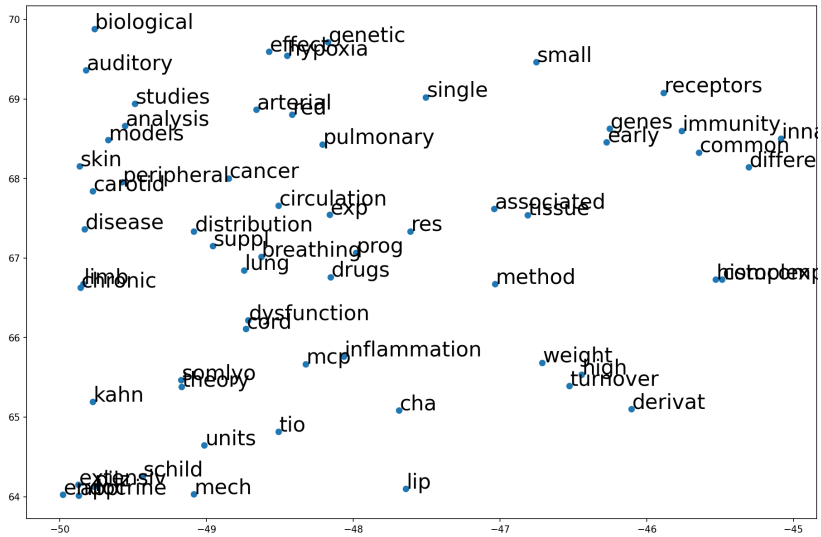


Abbildung 4.5: Word2Vec: Cluster-Wolke die nur aus englischen Begriffen gebildet wird.

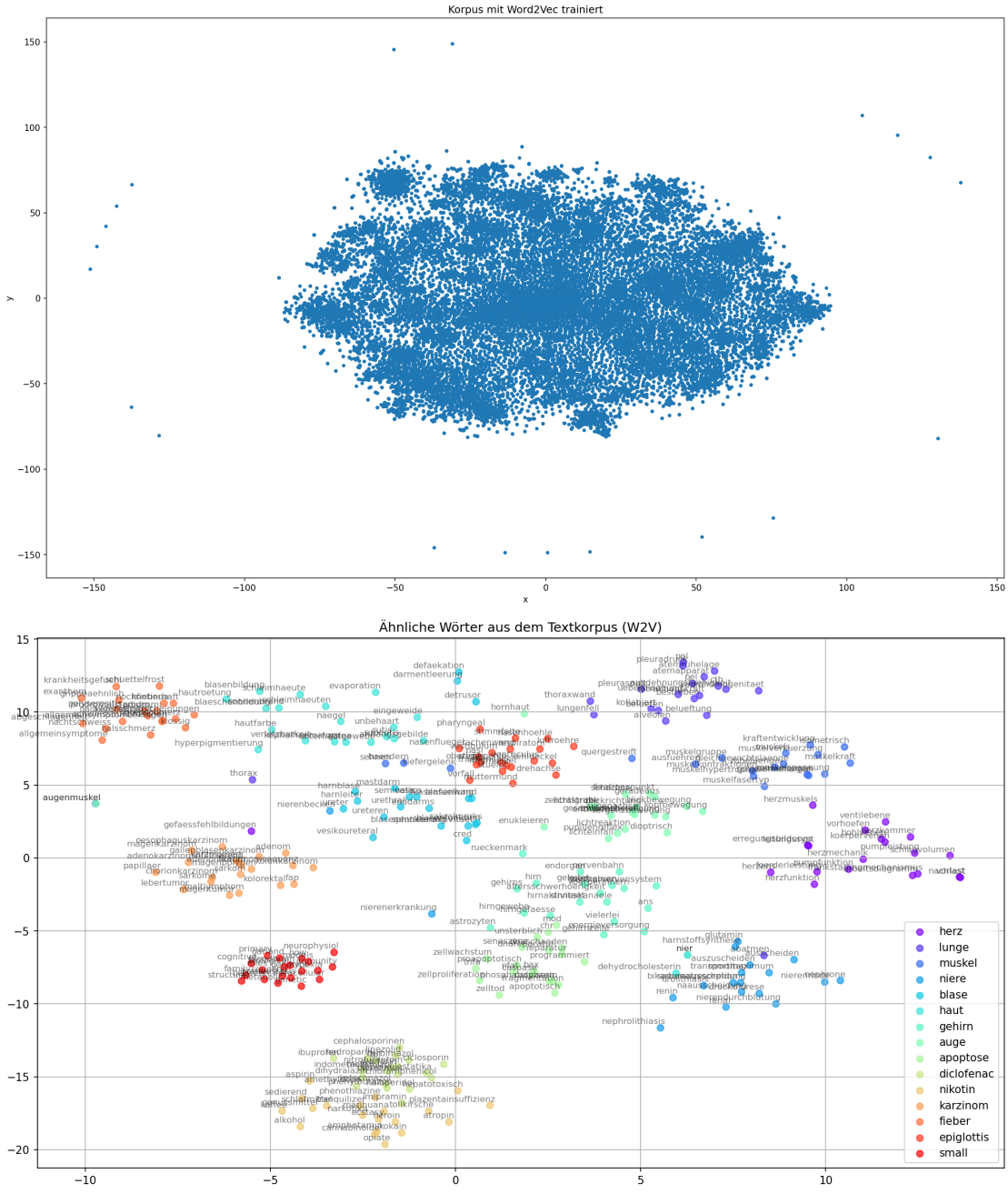


Abbildung 4.6: Oben: Gesamte Übersicht aller Wort-Vektoren in einem Diagramm, die mit Word2Vec trainiert wurden.
Unten: Wort-Cluster-Bildung durch 14 Wörter mit Word2Vec und t-SNE.

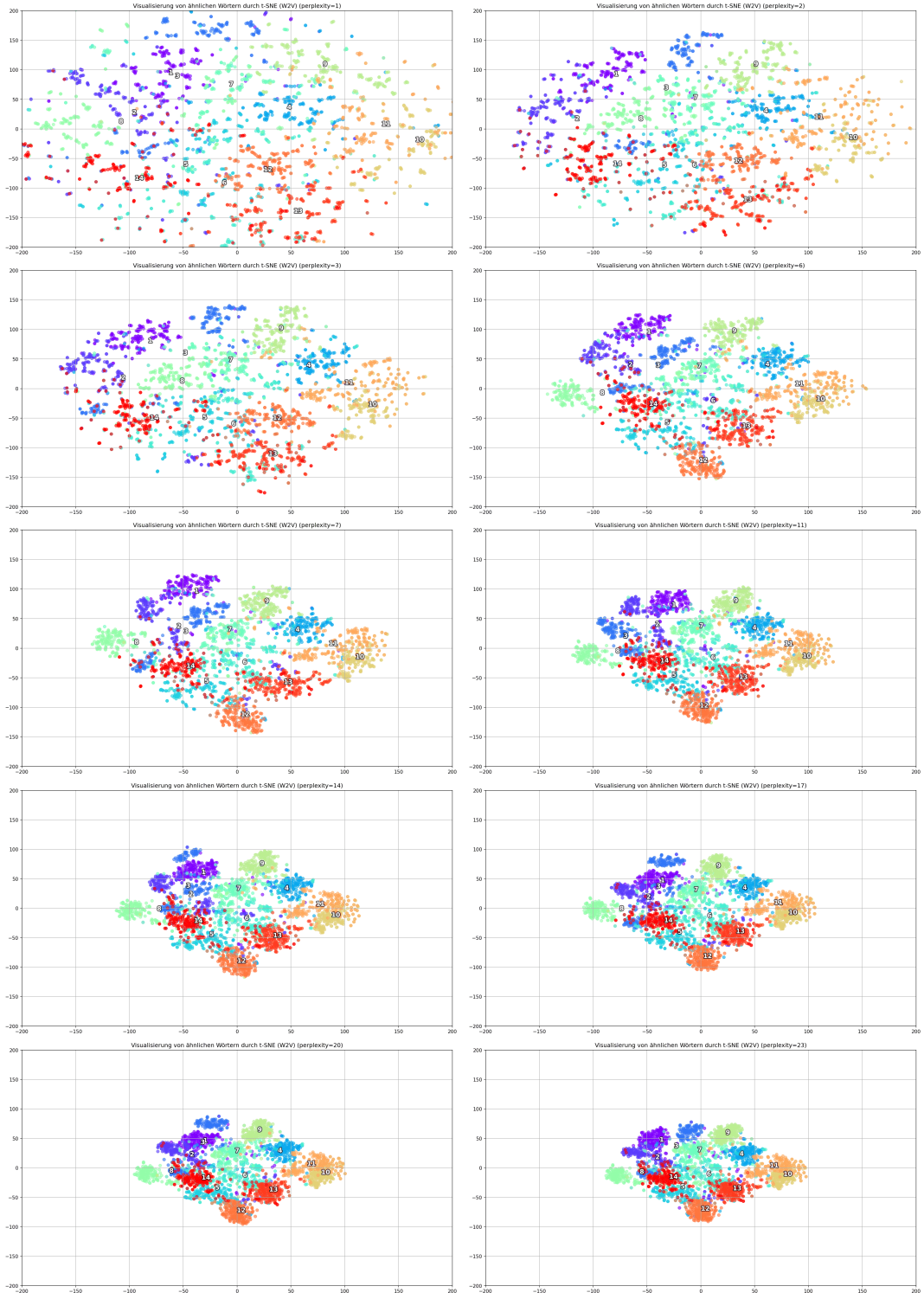


Abbildung 4.7: Zeitraffer von Word2Vec, der zur Bildung von Wort-Clustern führt.

4.1.3.2 GloVe

Die Verteilung der Wort-Vektoren von GloVe ist in Abb. 4.8 dargestellt. Nur im äußeren Rand der Verteilung haben sich einige Cluster gebildet. Vom Zentrum bis zum Rand scheinen die Wort-Vektoren vereinzelt gestreut. GloVe konnte keinen geeigneten Bezug zur Semantik der Wörter herstellen. Dies wurde bereits im Ähnlichkeitstest (Tab 4.3) und im Analogie Test (Tab. 4.4) zum Ausdruck gebracht. Hier wird erneut klar, weshalb es mit GloVe nicht gelingt, im Ähnlichkeits- und Analogie-Test passende Wörter vorgeschlagen zu bekommen. Die Beziehungen zu den einzelnen Wörtern wurden nicht aufgegriffen.

Einen weiteren Beweis liefert Abb. 4.8. Das GloVe-Modell ist nicht im Stande, anhand vordefinierter Wörter passende Wörter zu finden, die sich ähneln. Es haben sich nur vereinzelt, um die Worte **lunge** und **apoptse**, Cluster gebildet. Der Großteil der Wörter bildet keine Cluster.

Wie in Abb. 4.9 dargestellt, bilden sich bei höherer *Perplexity* gewisse Cluster. Diese sind aber nicht klar von anderen Clustern abgegrenzt, sondern vielmehr fusioniert. Es wirkt, als bilden sie gemeinsam ein einheitliches Cluster. Während im Word2Vec-Modell eine klare Differenzierung der Cluster stattfindet, sind hier alle miteinander verschmolzen.

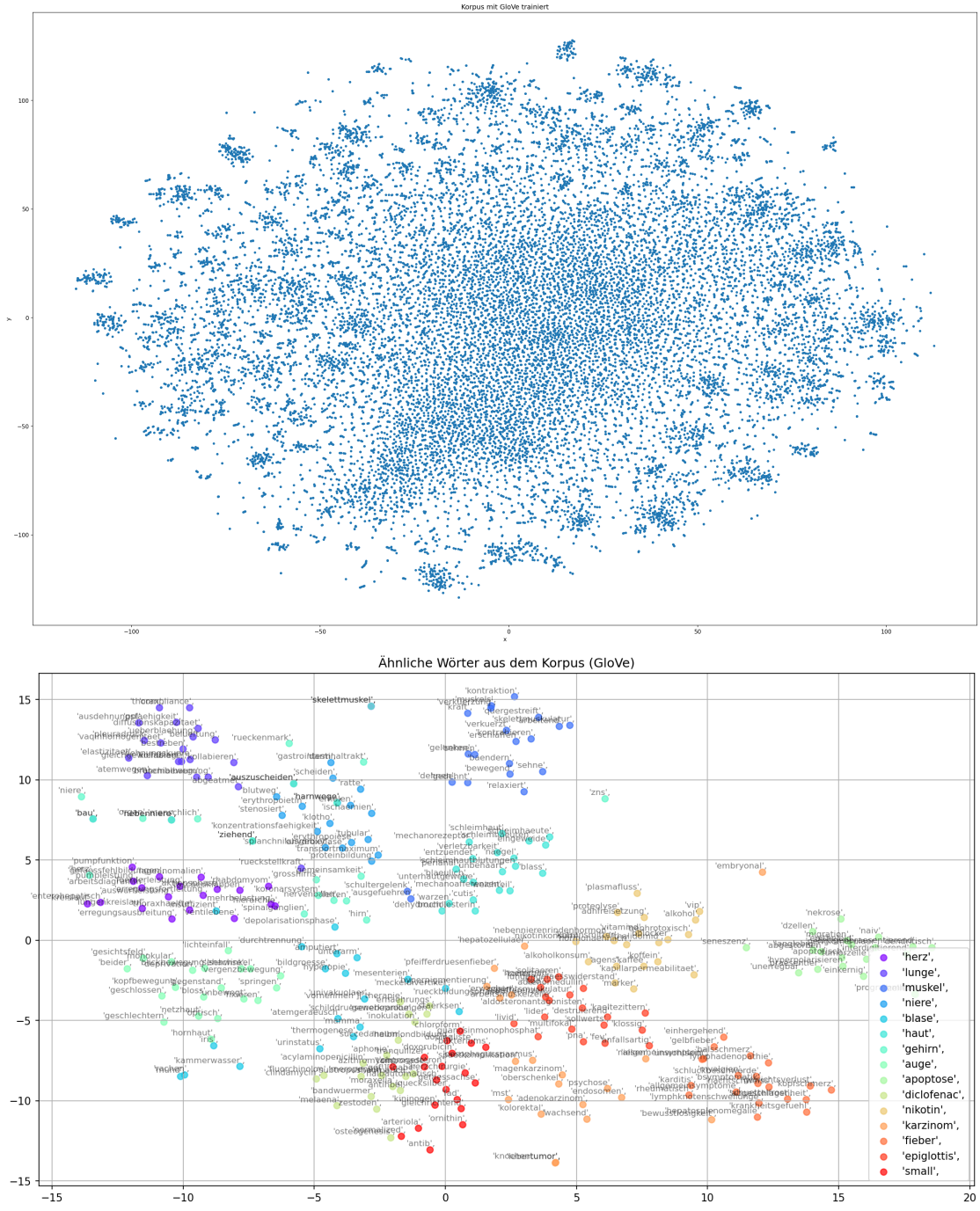


Abbildung 4.8: Oben: Gesamte Übersicht aller Wort-Vektoren in einem Diagramm, die mit GloVe trainiert wurden.
Unten: Wort-Cluster-Bildung durch 14 Wörter mit GloVe und t-SNE.

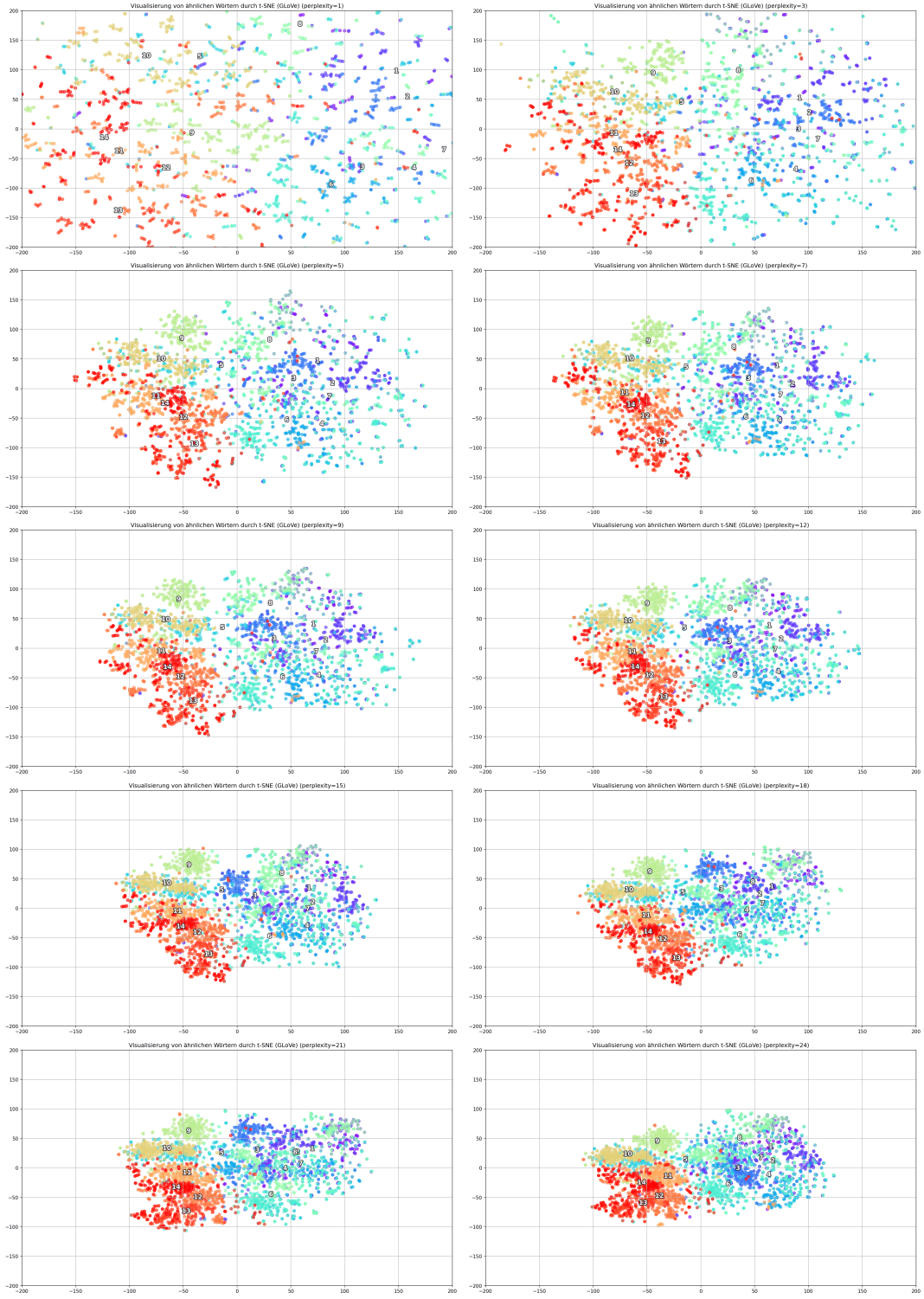


Abbildung 4.9: Zeitraffer von GloVe, der zur Bildung von Wort-Clustern führt.

4.1.3.3 Fasttext

Fasttext hat im Ähnlichkeitstest die passendsten Wörter vorgeschlagen. In Abb. 4.10 lässt sich diese Aussage bestätigen. Es sind zahlreiche kleine Cluster-Wolken zu sehen. Nur vereinzelt haben sich große Cluster gebildet. Im Zentrum befindet sich, wie bei Word2Vec, eine Ansammlung von mehreren Worten, die erneut ein Indiz für ein Overfitting ist. Obwohl das gesamte Bild mehr von kleinen Clustern geprägt ist als bei den anderen Word-Embedding-Modellen, finden sich weiterhin mehrere Wort-Vektoren, die im Raum verstreut sind.

In Abb. 4.10 befinden sich klare und voneinander abgegrenzte Cluster, insbesondere die Cluster-Bildungen mit den Worten **karzinom** und **herz**. Wie beim Ergebnis von Word2Vec befinden sich die Wörter, die mit **niktoin** und **diclofenac** assoziiert werden, nahe beieinander. Allerdings finden sich auch Wörter wie **apoptose**, deren Distanz zu den ähnlichen Wörtern noch zu hoch ist. Es wird somit kein Cluster gebildet.

Ein Blick auf die Entwicklung der Cluster in Abb. 4.11 zeigt, dass sich die Wort-Cluster recht früh bilden. Einige Cluster haben eine klare Abtrennung, wie z. B. die um die Wörter **herz** und **muskel**. Auch in diesem Bild liegen die Cluster von **niktoin** und **diclofenac** nahe zusammen und bilden ein gemeinsames Cluster.

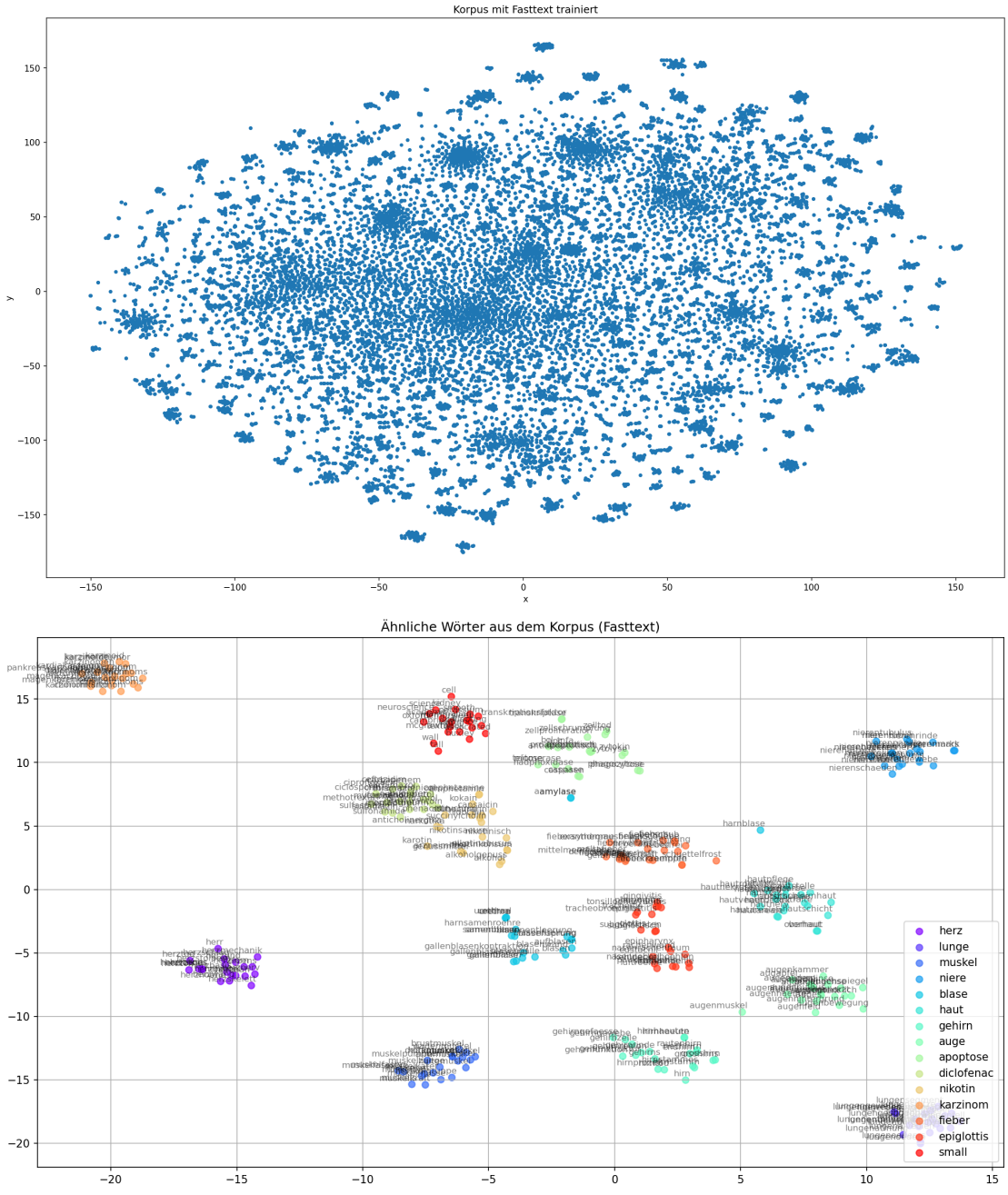


Abbildung 4.10: Oben: Gesamte Übersicht aller Wort-Vektoren in einem Diagramm, die mit Fasttext trainiert wurden.
Unten: Wort-Cluster-Bildung durch 14 Wörter mit Fasttext und t-SNE.

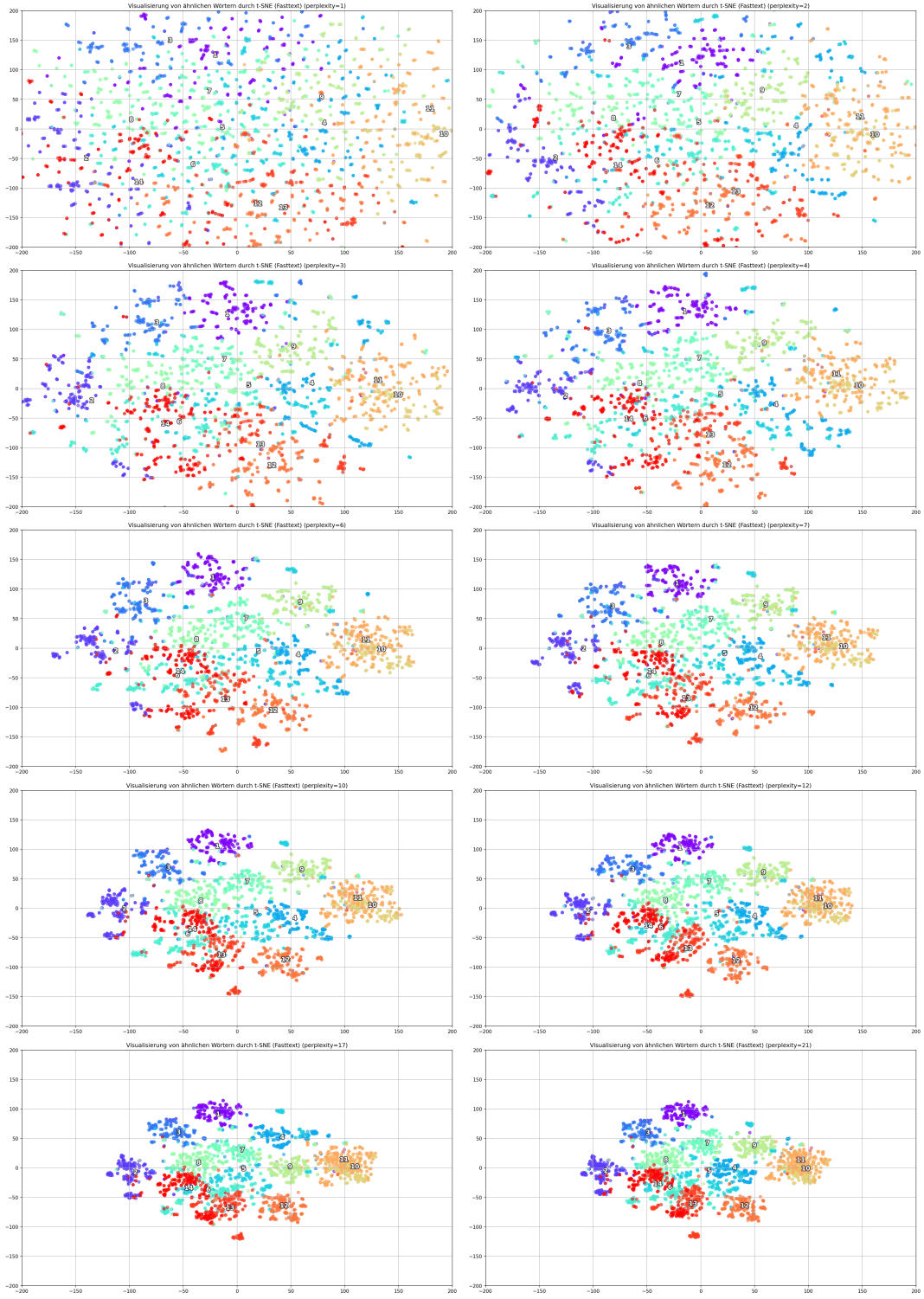


Abbildung 4.11: Zeitraffer von Fasttext, der zur Bildung von Wort-Clustern führt.

4.2 Diskussion

Um den Einsatz von englischen Word Embeddings auf deutsche medizinische Literatur zu veranschaulichen, wurden die drei renommiertesten Word Embeddings explizit mit einem deutschen medizinischen Korpus antrainiert. Der Korpus bestand aus einer Ansammlung von unterschiedlichen medizinischen Büchern. Für die Veranschaulichung wurden intrinsische Evaluierungsmodelle wie Ähnlichkeitstest und Analogie-Test angewendet. Um eine qualitative Bewertung der Ergebnisse zu garantieren, wurden zusätzlich Diagramme von der Verteilung der Wort-Vektoren der einzelnen Word Embeddings untersucht. Auf dieser Basis lässt sich ein gesamter Eindruck der Word Embeddigs über ihre Effizienz mit deutschen Texten wiedergeben.

Der Ähnlichkeitstest zeigt, dass durch Fasttext in den meisten Kategorien die geeignetsten Wörter vorgeschlagen wurden (Abb. 4.3). Dabei sind folgende Muster zu erkennen: Fasttext trennt die gesuchten Wörter aufgrund seiner Teilwort-Informationen in einzelne Morpheme. Diese Morpheme wiederum bilden als Prä- oder Suffix mit einem anderen Morphem ein neues Wort. So hat Fasttext das Wort **muedigkeit** in einzelne Morpheme zerlegt. Aus dem Morphem **keit** wurden folgende neue Wörter gebildet: **tagesmuedigkeit**, **appetitlosigkeit** und **nackensteifigkeit**.

Weiterhin fällt auf, dass Word2Vec und GloVe viel mehr Wörter vorschlagen, die sich in der Semantik ähneln. Während sich Fasttext doch eher auf die Syntaktik konzentriert. So wird bei **fieber** von Word2Vec und GloVe **schuettelfrost** ausgegeben, der ein Symptom von fiebrigen Infektionen ist. Dagegen lässt Fasttext **qfieber** ausgeben, das bei Rindern und Schafen vorkommt.

In den Abbildungen 4.1, 4.2, 4.3 und 4.4 wird ein Teil der Ergebnisse aus Tabelle 4.3 grafisch dargestellt. Die Tabelle zeigt aufgrund des Platzmangels nur die drei passendsten Antworten. Somit sind in dem Diagramm zusätzlich die umliegenden Wort-Vektoren dargestellt. So lässt sich leichter nachvollziehen, welches Word-Embedding-Modell die geeigneteren Worte assoziiert. So lassen sich in Abb. 4.4 für das Suchwort **opiate** Wörter wie **cocain**, **lsd**, **ecstasy** oder auch **ampetamine** finden, die die anderen Word Embeddings nicht assoziieren.

Zudem fällt auf, dass bei der Kategorie **Medikamente** Word2Vec zufriedenstellendere Vorschläge hatte. Dies kann daran liegen, dass die zerlegbaren Morpheme der Wörter **ibuprofen**, **metformin** und **diclofenac** schwierig mit deutschen Wörtern verknüpft werden können. Das fällt bei dem Wort **metformin** auf: Fasttext hat mit dem Morphem **met** die Wörter **methanol**, **metoprolol** und **methotrexat** gebildet. Obwohl das Suchwort ein Medikament für die Behandlung von Diabetes mellitus Typ 2 ist, bietet Fasttext als Alternative einen Alkohol, einen Blutdrucksenker und schließlich ein Medikament für eine Krebserkrankung an.

Das Gesamtbild des Ähnlichkeitstests entspricht den Erwartungen, dass bei Morphologie reichen Sprachen wie Deutsch, Fasttext aufgrund seiner Teilwort-Informationen passendere Ergebnisse erzielt. Diese Aussage stützt sich auf das Entwicklunsgteam von Fasttext [2]. Eine weitere Studie, die mit einer Morphologie reichen Sprache (Kroatisch) gearbeitet hat, kam auf das gleiche Ergebnis [36].

Beim Analogie-Test zeigt sich ein anderes Bild. Hier konnte Fasttext in keiner Analogie überzeugen. Statt das passende Wortpaar auszugeben, hat Fasttext erneut nur das zerlegte Morphem erweitert. Als Beispiel dienen hier folgende Analogie-Paare: **lymphozyten - lyhmpozyt** oder **bakteriums - bakterium**.

Word2Vec hat beim Analogie-Test überzeugende Ergebnisse gezeigt. Bei einigen Analogie-Fragen hat Word2Vec das passende Wortpaar gefunden, wie bei **treponema - bakterium** oder **nkzellen - bakterium**. Dieses Ergebnis war zu erwarten. Word2Vec und Fasttext sind sich ähnlich. Der Unterschied besteht lediglich darin, dass bei Fasttext die Zerlegung in N-Gram durchgeführt wird. Dadurch schafft Fasttext überzeugendere Leistungen bei syntaktischen Analogien. Dies ist wiederrum ein Hindernis, wenn es die Aufgabe ist, sich auf eine semantische Analogie zu konzentrieren. Mit anderen Worten würden Word2Vec und Fasttext ähnliche Ergebnisse beim Analogie-Test erzielen, wenn bei Fasttext die Zerlegung der N-Grams ausgeschaltet werden würde. Dies wurde in einem *Blog* nachsimuliert und bestätigte die These [9].

Werden die gesamten Diagramme der Word Embeddings in Abb. 4.6, Abb. 4.8 und Abb. 4.10 betrachtet, ist das Bild von Fasttext am überzeugendsten, da hier einigermaßen differenzierte Cluster gebildet wurden. Dennoch ist das Ergebnis nicht zufriedenstellend. Dies liegt daran, dass t-SNE nicht als Vorverarbeitungs- oder Analysewerkzeug

konzipiert ist, sondern nur als Visualisierungswerkzeug. Somit werden die Abstände zwischen den Datenpunkten verzerrt, um ansprechende Visualisierungsdiagramme herzustellen. Weiterhin bildet t-SNE nie das gleiche Diagramm. Jede Visualisierung, auch mit den gleichen Hyperparametern, erzeugt ein anderes Diagramm. Dies erschwert die Fehlersuche. Zudem bildet sich bei höheren Iterationen oder Epochen ein überdimensional großes Cluster im Zentrum. Dies könnte ein Indiz für ein Overfitting sein.

Werden die Daten reduziert, wie in den unteren Diagrammen in Abb. 4.6, Abb. 4.8 und Abb. 4.10 dargestellt, entstehen klar differenzierte Cluster. Besonders in Abb. 4.10 entspricht das Diagramm den Erwartungen von Cluster-Bildungen. Wörter, die sich nicht ähneln, trennen sich klar von anderen Worten ab. Des Weiteren fällt auf, dass die Cluster **nikotin** und **diclofenac** nahe beieinander liegen. Es macht eher den Eindruck, als würden sie ein gemeinsames Cluster bilden. Dies macht unter der Annahme, dass es sich bei beiden um Medikamente handelt, auch Sinn. In Abb. 4.6 von Word2Vec haben sich einige Wortgruppen zu Clustern gebildet. Nur bei GloVe (Abb. 4.8) ist die Bildung von Clustern vollkommen gescheitert. Dies kann darauf zurückgeführt werden, dass das Antrainieren mit GloVe nicht funktioniert hat, da GloVe in fast allen Aufgaben gescheitert ist.

Abb. 4.7, Abb. 4.9 und Abb. 4.11 zeigen die Visualisierung nach unterschiedlicher *Perplexity*. Dabei wird die Relevanz des Hyperparameters deutlich, da bei erhöhter *Perplexity* der Abstand zu den Nachbarn geringer wird und somit die Bildung von Clustern möglich ist. In allen drei Abbildung ist ein Perplexity-Wert von über 20 ausreichend für eine zufriedenstellende Cluster-Visualisierung. Lediglich in Abb. 4.8 ergibt sich trotz hoher Perplexity keine klare Trennung unter den Daten. Auch hier steht die Vermutung, dass das Training mit GloVe nicht ausreichend war, um ähnliche Wörter exakt zu klassifizieren.

Es muss jedoch berücksichtigt werden, dass sich diese Forschung nur mit deutschen medizinischen Texten befasst. Die erwähnten Beispiele beziehen sich auch nur auf den medizinischen Kontext. Bei einem anderen Korpus, wie dem von Wikipedia, würden ganz andere Ergebnisse erzielt werden. Somit muss berücksichtigt werden, dass der Fall in dieser Arbeit äußerst spezifisch ist. Aus diesem Grund kann keine generelle Aussage über die Word Embeddings getroffen werden.

4.3 Ausblick

Die vorliegende Arbeit wurde innerhalb der festgeregelten Zeit von sechs Monaten erstellt. Somit ergeben sich weitere Forschungsfelder, die aufgrund der Zeit nicht behandelt werden konnten und in Kürze erwähnt werden sollen. Word2Vec, GloVe und Fasttext sind Word Embeddings, die bereits mit der englischen Literatur vortrainiert wurden. Dennoch ist es möglich, diese vortrainierten Word Embeding zu nutzen, um damit deutsche Wörter zu klassifizieren. Ein Forschungsgebiet wäre es, Word Embeddings zu untersuchen, die mit deutschen Texten vortrainiert werden.

Durch die Verarbeitung des Korpus, indem Stoppwörter entfernt werden und die Großschreibung der Nomen verändert wird, ergibt sich ein einheitlicherer Text. Dennoch gehen dadurch Informationen verloren. Insbesondere in der deutschen Sprache macht es aufgrund der Semantik einen großen Unterschied, ob z. B. das Verb suchen oder das entsprechende Substantiv im Text vorkommt. Dieses würde dann großgeschrieben werden und bekäme eine ganz andere Bedeutung. Ein weiteres Forschungsgebiet wäre es daher zu überprüfen, inwiefern sich die Leistung von Word Embeddings verbessert oder ggf. verschlechtert, wenn einzelne Schritte, die für die Verarbeitung des Korpus angewandt wurden, entfallen.

In dieser Arbeit wurde ein medizinischer Korpus verwendet. Dies liegt nahe, da auch ein medizinischer Analogien- und ein Ähnlichkeitstest geprüft wurden. Dennoch bleibt die Frage offen, wie sich Word Embeddings verhalten, wenn ein Wikipedia-Korpus verwendet wird. Können dadurch die Leistungen der Word Embeddings verbessert werden? Eine ähnliche wissenschaftliche Arbeit existiert bereits, allerdings wurde dies anhand von englischen Texten untersucht [37]. Es wurde festgestellt, dass der medizinische Korpus überzeugendere Ergebnisse erzielt, als der Google-News-Korpus. Allerdings variiert die englische Sprache von der deutschen Sprache und es sollte daher erneut untersucht werden, zu welcher Leistungsdifferenz ein unterschiedlicher Korpus führen kann.

4.4 Fazit

Mit dieser Studie wurde versucht, die Frage zu beantworten, ob Word Embeddings in der Lage sind, inhaltliche Aussagen über deutsche medizinische Texte wiederzugeben. Zu diesem Zweck wurde eine quantitative Studie zu einem Ähnlichkeits- und Analogie-Test durchgeführt. Darüber hinaus wurde die Umgebung der Wort-Vektoren in einem Diagramm visualisiert, um einen tieferen Einblick in die Word Embeddings zu bekommen.

Die Ergebnisse zeigen, dass Word2Vec im Analogie-Test und Fasttext im Ähnlichkeitstest die überzeugendsten Ergebnisse erzielt haben, während die Ergebnisse mit GloVe in allen Kategorien nicht zufriedenstellend waren. Aus dem Gesamtdiagramm der Wort-Vektoren lässt sich hingegen entnehmen, dass es nur mit Fasttext möglich war, ähnliche Wörter differenzierter klassifizieren zu können. Auch in dem differenzierten Bild mit nur 14 Schlüsselwörtern ist zu erkennen, dass mit Fasttext klare und eindeutige Cluster gebildet werden konnten.

Das Ergebnis zeigt, dass besonders durch Fasttext deutsche medizinische Texte klassifiziert werden können. Durch die Teilwort-Information ist es mit Fasttext möglich, ähnliche Wörter vorgeschlagen zu bekommen und medizinische Texte anhand von mehreren Wort-Clustern zu klassifizieren. Auch wenn sich mit Fasttext Probleme beim Analogie-Test ergaben, so werden einzelne Fachbegriffe zufriedenstellend erkannt.

Cluster haben eine vielfältige Anwendung und können dazu genutzt werden, einen ersten Eindruck über den Text zu erhalten. So können anhand der Cluster erste Einblicke des Inhalts wiedergegeben werden. Die Informationen durch die Cluster bilden ein Grundgerüst für weitere Forschungsthemen. Dennoch schließt dies nicht aus, dass Word Embeddings in naher Zukunft noch effizienter den Inhalt eines Textes wiedergeben können.

Diese qualitative Forschung hat gezeigt, dass man durch Word Embeddings einen zufriedenstellenden Einblick in einen deutschen medizinischen Korpus erlangen kann. Die evaluierten Word Embeddings haben gezeigt, das sie in Lage sind deutsche Texte zu klassifizieren, mit Ausnahme von GloVe. Dies wurde übermittelt, indem Word Embeddings Wörter sowohl im Sinne der Ähnlichkeit als auch der Analogie wiedergeben

sollten. Mit Fasttext konnte ein zufriedenstellendes Ergebnis erzielt werden. Word2Vec lieferte nur beim Analogie-Test überzeugendere Ergebnisse.

Literatur

[1] https://pubmed.ncbi.nlm.nih.gov/. Zugriff: 14.09.2020.

[2] Piotr Bojanowski u. a. "Enriching word vectors with subword information". In: *Transactions of the Association for Computational Linguistics* 5 (2017), S. 135–146.

[3] Danushka Bollegala, Kohei Hayashi und Ken-ichi Kawarabayashi. "Learning linear transformations between counting-based and prediction-based word embeddings". In: *PloS one* 12.9 (2017), e0184544.

[4] Eduardo Brito u. a. "Towards German word embeddings: A use case with predictive sentiment analysis". In: *Data Science–Analytics and Applications*. Springer, 2017, S. 59–62.

[5] Hugo Caselles-Dupré, Florian Lesaint und Jimena Royo-Letelier. "Word2vec applied to recommendation: Hyperparameters matter". In: *Proceedings of the 12th ACM Conference on Recommender Systems*. 2018, S. 352–356.

[6] Francois Chollet. *Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek*. MITP-Verlags GmbH & Co. KG, 2018.

[7] Scott Deerwester u. a. "Indexing by latent semantic analysis". In: *Journal of the American society for information science* 41.6 (1990), S. 391–407.

[8] Jan Dillenberger. "Evaluation of Model and Hyperparameter Choices in word2vec". In: (2019).

[9] *FastText and Word2Vec*. https://jayantj.github.io/posts/fasttext-gensim-word-embeddings. Zugriff: 09.09.2020. 2016.

[10] Hannes Max Hapke, Hobson Lane und Cole Howard. *Natural language processing in action*. 2019.

[11] Zellig S Harris. "Distributional structure". In: *Word* 10.2-3 (1954), S. 146–162.

[12] Matthew Hoffman, Francis R Bach und David M Blei. "Online learning for latent dirichlet allocation". In: *advances in neural information processing systems*. 2010, S. 856–864.

[13] Thomas Hofmann. "Probabilistic latent semantic analysis". In: *arXiv preprint arXiv:1301.6705* (2013).

[14] Armand Joulin u. a. "Bag of tricks for efficient text classification". In: *arXiv preprint arXiv:1607.01759* (2016).

[15] *Journal Overview*. https://onlinelibrary.wiley.com/journal/10970215. Zugriff: 23.08.2020.

[16] Dan Jurafsky und James H. Martin. "Chapter 6 : Vector Semantics and Embeddings". In: (2019), S. 2–26.

[17] Maximilian Lam. "Word2bits-quantized word vectors". In: *arXiv preprint arXiv:1803.05651* (2018).

[18] Omer Levy und Yoav Goldberg. "Neural word embedding as implicit matrix factorization". In: 2014.

[19] Omer Levy, Yoav Goldberg und Ido Dagan. "Improving distributional similarity with lessons learned from word embeddings". In: *Transactions of the Association for Computational Linguistics* 3 (2015), S. 211–225.

[20] Laurens van der Maaten und Geoffrey Hinton. "Visualizing data using t-SNE". In: *Journal of machine learning research* 9 (2008), S. 2579–2605.

[21] Christopher Manning und Richard Socher. "Natural Language Processing with Deep Learning CS224N/Ling284". In: Lecture 3 (2019).

[22] Oren Melamud u. a. "The role of context types and dimensionality in learning word embeddings". In: *arXiv preprint arXiv:1601.00893* (2016).

[23] Tomas Mikolov u. a. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*. 2013, S. 3111–3119.

[24] Tomas Mikolov u. a. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[25] Katrin Ortmann, Adam Roussel und Stefanie Dipper. "Evaluating Off-the-Shelf NLP Tools for German." In: *KONVENS*. 2019.

[26] Charles E Osgood. "Semantic differential technique in the comparative study of cultures". In: *American Anthropologist* 66.3 (1964), S. 171–200.

[27] Makbule Gulcin Ozsoy. "From word embeddings to item recommendation". In: *arXiv preprint arXiv:1601.01356* (2016).

[28] II Part. "CS224n: Natural Language Processing with Deep Learning1". In: (2017).

[29] Kevin Patel und Pushpak Bhattacharyya. "Towards lower bounds on number of dimensions for word embeddings". In: 2017, S. 31–36.

[30] Jeffrey Pennington, Richard Socher und Christopher D Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, S. 1532–1543.

[31] Radim Řehůřek. "Word2vec Tutorial". In: *https://rare-technologies.com/word2vec-tutorial/* (2014).

[32] Matthias Richter. *Comparing Word Embeddings*. https://towardsdatascience.com/comparing-word-embeddings-c2efd2455fe3. Zugriff: 23.08.2020. 2019.

[33] Magnus Sahlgren. *A brief history of word embeddings (and some clarifications)*. https://www.gavagai.io/text-analytics/a-brief-history-of-word-embeddings/. Zugriff: 23.08.2020. 2015.

[34] Richard Smith. "The trouble with medical journals". In: *Journal of the Royal Society of Medicine* (2006).

[35] Yan Song u. a. "Directional skip-gram: Explicitly distinguishing left and right context for word embeddings". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. 2018, S. 175–180.

[36] Daniel Vasić und Emil Brajković. "Development and Evaluation of Word Embeddings for Morphologically Rich Languages". In: *2018 26th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. IEEE. 2018, S. 1–5.

[37] Yanshan Wang u. a. "A comparison of word embeddings for the biomedical natural language processing". In: *Journal of biomedical informatics* 87 (2018), S. 12–20.

[38] Martin Wattenberg, Fernanda Viégas und Ian Johnson. "How to use t-SNE effectively". In: *Distill* 1.10 (2016), e2.

[39] Yonghui Wu u. a. "Google's neural machine translation system: Bridging the gap between human and machine translation". In: *arXiv preprint arXiv:1609.08144* (2016).

[40] Zi Yin. "Understand functionality and dimensionality of vector embeddings: the distributional hypothesis, the pairwise inner product loss and its bias-variance trade-off". In: (2018).

[41] Zi Yin und Yuanyuan Shen. "On the dimensionality of word embedding". In: *Advances in Neural Information Processing Systems*. 2018, S. 887–898.

[42] Tony Yiu. *The Curse of Dimensionality.* https://towardsdatascience.com/the-curse-of-dimensionality-50dc6e49aa1e. Zugriff: 23.08.2020. 2019.

[43] *Zahl der Krebsneuerkrankungen weltweit nach Krebsart im Jahr 2018*. https://de.statista.com/statistik/daten/studie/286545/umfrage/zahl-der-krebsneuerkrankungen-weltweit/. Zugriff: 23.08.2020.

Tabellenverzeichnis

Abbildungsverzeichnis

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 1. Oktober 2020 Nassim Haidar