RUPRECHT-KARLS-UNIVERSITÄT HEIDELBERG

BACHELORARBEIT

Analysen der Relationsvorhersage im Deutschen mithilfe von Wortvektorrepräsentationen

Autor:
Dennis ULMER

Betreuende:
Dr. Viviana NASTASE
Dr. Yannick VERSLEY

Eine Arbeit zur Erlangung des Bachelorgrades

7. August 2016

„I have not failed. I've just found 10,000 ways that won't work."

THOMAS A. EDISON

RUPRECHT-KARLS-UNIVERSITÄT HEIDELBERG

Zusammenfassung

Neuphilologische Fakultät
Institut für Computerlinguistik

Bachelor of Arts

Analysen der Relationsvorhersage im Deutschen mithilfe von Wortvektorrepräsentationen

von Dennis ULMER

In sogenannten *Wissensdatenbanken* wird Wissen über die Beziehungen von Entitäten in der realen Welt beschrieben. Dieses aufbereitete Wissen ist eine nützliche Quelle, um das fehlende Weltwissen bei Computersystemen zu kompensieren. Die Erstellung solcher Datenbanken ist jedoch sehr arbeitsintensiv, weshalb Bemühungen, diese automatisch zu ergänzen, ein lohnendes Forschungsfeld darstellen.

Diese Abschlussarbeit wirft einen Blick auf verschiedene Ansätze zur Relationsvorhersage, inbesondere *TransE* von (Bordes et al., 2013), und versucht darüber hinaus, diese auf deutsche Daten anzuwenden und auf die in der maschinellen Sprachverarbeitung sehr beliebt gewordenen Wortkontextvektoren (*word embeddings*) zu übertragen. Außerdem wird ein Versuch unternommen, mithilfe einer unüberwachten Methode des maschinellen Lernens neue semantische Relationen in einem Wortvektorraum aufzufinden.

RUPRECHT-KARLS UNIVERSITY HEIDELBERG

Abstract

Faculty of Modern Languages
Department for Computational Linguistics

Bachelor of Arts

Analysis of relation prediction in German using word vector representations

by Dennis ULMER

So-called *knowledge bases* contain knowledge about relations between entities in the real world. This preprocessed knowledge is a useful source to compensate the inherent lack of universal knowledge of any computer system. However, the creation of such databases is deemed to be very labour-intensive. Thus, automatic efforts to replenish them might be worth researching.

This thesis aims to compare approaches in this area concerning various kinds of vector representations, especially *TransE* by (Bordes et al., 2013), applies them to German datasets and furthermore wants to realize similar procedures based on word embeddings, which gained a lot of popularity in recent years among the Natural Language Processing community. Also, an endeavour is made to find new types of semantic relations among word embeddings using an unsupervised machine learning procedure.

Danksagung

Heidelberg, den 06.08.16

An dieser Stelle möchte ich allen Menschen danken, die zur Fertigstellung dieser Arbeit beigetragen haben:
Zuvorderst sind hier Dr. Yannick Versley, der es mir ermöglichte, im Rahmen meiner Abschlussarbeit eine eigene Idee zu verwirklichen und mir immer mit Rat und Tat zur Seite stand, sowie Dr. Viviana Nastase, mit deren konstruktiven Feedback ich einen kühlen Kopf bewahren und diese Arbeit vollenden konnte, zu nennen.
Desweiteren gilt mein Dank meiner Familie, die mich in den letzten Jahren auf kaum aufzählbare Arten und Weisen unterstützt hat. Nur durch sie ist es mir nun möglich, diese Danksagung überhaupt schreiben zu können.
Darüber hinaus möchte ich auch meinen Freunden und Kommillitonen danken, die mir in den letzten Monaten mit Rat und Tat zur Seite standen, sei es beim Korrigieren dieser Arbeit, beim Diskutieren neuer Ideen, bei berechtigter Kritik, bei neuen Ideen und Anregungen oder einem Schulterklopfer bei einem der Tiefpunkte, die man beim Schreiben erlebt.

Ich begreife diese Arbeit nicht nur als Abschluss eines letzten Moduls meines Bachelorstudiums, sondern auch als Abschluss meines Bachelorstudiums hier in Heidelberg im Allgemeinen. Da das Studium selten nur in der Uni blieb, sondern auch von anderen Lebensbereichen in dieser Zeit kaum zu trennen war, möchte ich an dieser Stelle zuletzt meine Wertschätzung gegenüber allen Menschen ausdrücken, die diesen meinen Lebensabschnitt mit ihren Begegnungen - sei es auch noch so kurze gewesen - bereichert haben. Diejenige Person, der ich zum jetzigen Zeitpunkt entspreche, ist zu einem großen Teil auch die Summe aller Erfahrungen, die ich in den letzten Jahren hier gesammelt habe. Und dazu haben viele Menschen - vielleicht auch unwissend - ihren Teil beigetragen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

> "*Weltwissen beschreibt das einem Individuum verfügbare allgemeine Wissen, Kenntnisse und Erfahrungen über Umwelt und Gesellschaft. [...] Das Weltwissen ermöglicht es, neue Tatsachen einzuordnen und entsprechend zu handeln, auch wenn detaillierte Informationen fehlen. [...]*
>
> *Auch in der Robotik spielt Weltwissen [...] eine Rolle, da Computer [...] nicht selbst über Weltwissen verfügen.*"
>
> EINLEITUNG DES ARTIKELS ÜBER WELTWISSEN, WIKIPEDIA[1]

1.1 Knowledge Graph Completion

Computer sind dem Menschen mittlerweile beim Lösen vieler Aufgaben überlegen. Sie rechnen schneller und genauer. Sie können riesige Datenmengen in einem Bruchteil der Zeit verarbeiten, die ein Mensch dafür bräuchte. Die menschliche Überlegenheit beginnt auch in Bereichen zu bröckeln, bei denen der Einsatz von Computern lange für eine Unnmöglichkeit gehalten wurde: Menschliche Champions scheitern schon seit dem Match *Deep Blue* gegen Kasparow 1995[2] gegen Maschinen beim Schach. Zuletzt scheiterte auch der Mensch beim Spiel Go gegen ein "intelligentes" System[3]. In anderen Bereichen, wie zum Beispiel der maschinellen Übersetzung oder einer allgemeine künstlichen Intelligenz jedoch hinken die Maschinen den Prognosen hinterher. Um Probleme für künstliche Intelligenzen lösbar zu machen, wurde bisher meist versucht, die Welt um das System herum zu vereinfachen:

[1] Online unter https://de.wikipedia.org/wiki/Weltwissen (zuletzt abgerufen am 03.03.16)

[2] Online unter https://de.wikipedia.org/wiki/Deep_Blue (zuletzt abgerufen am 19.07.16)

[3] Online unter http://qz.com/639952/googles-ai-won-the-game-go-by-defying-millennia-of-basic-human-instinct/ (zuletzt abgerufen am 12.07.16)

> *"The [...] reason for the fundamental problems of AI in general is that the models do not take the real world sufficiently into account. Much work in classical AI has been devoted to abstract, virtual worlds with precisely defined states and operations, quite unlike the real world."*[4]
>
> EUROPEAN NETWORK FOR THE ADVANCEMENT OF ARTIFICIAL COGNITIVE SYSTEMS, INTERACTION AND ROBOTICS.

Für eine *allgemeinte künstliche Intelligenz*[5] reicht dies jedoch nicht aus. Entwicklungen wie die ersten Fahrten fahrerloser Autos, den Jeopardygewinner Watson[6] und ähnliche zeigen den Fortschritt in diesem Bereich auf, jedoch ist die Wissenschaft von ihrem Ziel noch weit entfernt. Ein Grund dafür ist das wie eingangs im Eröffnungszitat erwähnte Problem von Computern, dass sie im Gegensatz zum Menschen über kein Weltwissen verfügen.

Dieses bietet Informationen darüber,

- wie Objekte in der Realwelt zueinander in Beziehung stehen
- welche Attribute zu verschiedenen Entitäten gehören
- wie Ereignisse in der Welt in Verbindung zueinander stehen
- ...

Ersteres wird durch sog. *Ontologien*[7], d.h. Darstellungen von Beziehungen zwischen Elementen einer Menge, modelliert. Stellt man sich vor, zwischen allen Entitäten nun Verbindungen in Form von Ontologien zu ziehen, entsteht ein Graph, in dem die Beziehungen der Knoten untereinander durch verschiedene Arten von Kanten kodiert sind, dem *Wissensgraphen* (engl. *Knowledge Graph*).

Die Vervollständigung ebendieses ermöglicht Systemen, die langwierige menschliche Erlernung dieses Wissens zumindest annähernd zu kompensieren. In dieser Arbeit sollen deshalb verschiedene Ansätze untersucht werden, einen solchen Graphen mithilfe verschiedener kontinuierlicher Repräsentationen für Entitäten und Relation zu erweitern.

1.2 Ansatz

Innerhalb der letzten Jahre haben Neuronale Netze in der Informatik im Allgemeinen und in der Computerlinguistik im Speziellen eine Renaissance

[4] *"Problems of Traditional AI"*, online unter http://www.eucognition.org/index.php?page=2-3-problems-of-traditional-ai (zuletzt abgerufen am 12.07.16)

[5] *"Artificial general intelligence (AGI) is the intelligence of a (hypothetical) machine that could successfully perform any intellectual task that a human being can"*, siehe https://en.wikipedia.org/wiki/Artificial_general_intelligence (zuletzt abgerufen am 19.07.16)

[6] Wie Watson menschliche Teilnehmer schlägt, ist u. A. hier zu beobachten https://www.youtube.com/watch?v=YgYSv2KSyWg (zuletzt abgerufen am 19.07.16).

[7] (Giaretta und Guarino, 1995) unterscheiden sieben verschiedene Bedeutungsschattierungen des Begriffs, in dieser Arbeit wird dabei vor allem von der Lesart #3 von der Ontologie als *"a formal semantic account"* ausgegangen.

erlebt. Mit diesen konnte eine neue Art von Wortvektoren, nämlich *Wortkontextvektoren*[8] (engl. "word embeddings") erzeugt werden, die semantische Information implizit in sich kodiert. Einige Untersuchungen wie (Levy, Goldberg und Dagan, 2015) zeigen, dass dieser Ansatz älteren im Bezug auf die semantische Aussagekraft durchaus sehr ähnlich ist und nicht unbedingt zu besseren Ergebnissen führt. Der Aufruhr hat aber eine neue Welle von Forschungen im Bereich der distributionellen Semantik ausgelöst.
Ein oft zitiertes Beispiel (siehe bspw. (Levy, Goldberg und Ramat-Gan, 2014)) für die Ausdruckskraft dieser Vektoren ist das Entdecken von semantischen Relationen hinter einfachen arithmetischen Operationen:

$$\vec{v}(King) - \vec{v}(Man) + \vec{v}(Woman) \approx \vec{v}(Queen)$$

Zwar lassen sich dadurch nicht alle Arten von Relationen aus Wortvektoren extrahieren und beschränken sich die Beispiele bei dieser Herangehensweise auf 1:1-Relationen (1:N-, N:1- sowie N:N-Relationen lassen sich auf andere Art und Weise finden), jedoch lassen sie schon ein gewisses Potenzial erahnen.
Die Idee, die in dieser Arbeit angegangen werden soll, fußt auf der Hypothese, dass das Identifizieren solcher Relationen nicht möglich wäre, wenn sich die Differenzvektoren von Wortpaaren einzelner Relationen nicht ähneln würden, also z.B.

$$\vec{v}(Berlin) - \vec{v}(Germany) \approx \vec{v}(Paris) - \vec{v}(Frankreich) \approx \vec{v}(Madrid) - \vec{v}(Spain)$$

Betrachtet man die Abstandsvektoren von Wortpaaren als Punkte in einem eigenen Vektorraum, so müssten theoretisch die Punkte, die zu Ländern und deren Hauptstädten gehören, in diesem Raum nahe beieinander liegen (siehe Abbildung 1.1).

Dies könnte man dann insofern ausnutzen, indem man ein Clusteringverfahren anwendet, um Gruppen mit ähnlichen Differenzvektoren zu identifizieren und die Elemente jeder Gruppe danach mit der entsprechenden Relation in einen Wissensgraphen einzuordnen. Dies hätte zwei Vorteile:

1. Teile des kostenintensiven Dateneinpflegens durch menschliche Hilfe fällt weg

2. Es wird ersichtlich, welche semantischen Relationen in einem Raum von Wortkontextvektoren tatsächlich abgebildet werden können.

Die Ergebnisse, die diese Prozedur und andere Experimente zutage fördern sowie die Fallstricke, die diese mit sich bringen, werden in den nächsten Kapiteln beschrieben. Über jene wird nun ein kleiner Überblick gegeben.

[8] Diese Übersetzung wurde deshalb gewählt, da sich der Begriff "word embedding" davon ableitet, dass Worte beim Trainingsprozess gewissermaßen in ihrem umgebenden Satzkontext eingebettet sind. Darüber hinaus soll diese Art von Wortvektor bewusst gegenüber "One-Hot-Vektoren" (siehe Kapitel 2.1) und anderen Vektorrepräsentationen für Begriffe (z.B. von (Bordes et al., 2013) abgegrenzt werden.

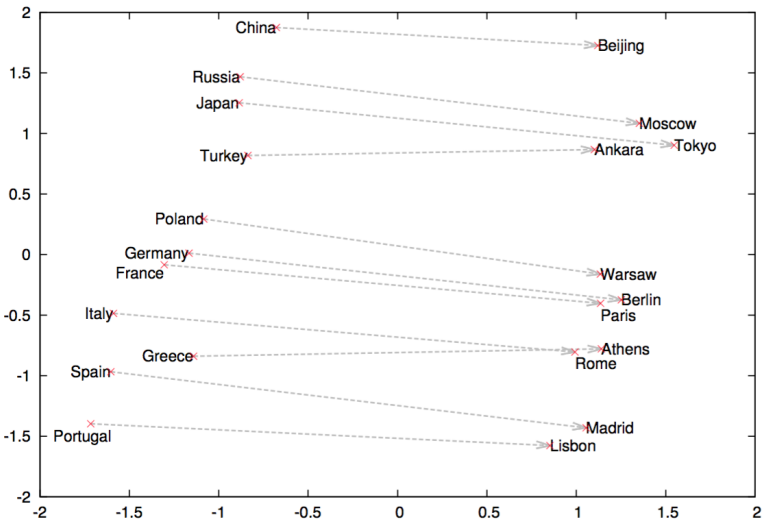

Abbildung 1.1: Hauptstadtrelationen zwischen Wortvektoren aus (Mikolov et al., 2013a). Die 1000-dimensionalen Vektoren wurden mithilfe von *Principal Component Analysis* (PCA) auf zwei Dimension projiziert.

1.3 Inhalt

In Kapitel 2 dieser Arbeit sollen verwandte Werke zu den erwähnten Themen referenziert werden. Daraufhin folgt in Kapitel 3 die Darlegung einiger für das Verständnis essenzieller Grundlagen, wie z.B. Neuronale Netze und ihre Nutzung zum Erstellen von Wortkontextvektoren.
Die in Kapitel 4 beschriebenen Vorbereitungsschritte und eine ausführliche Evaluation von Wortkontextvektoren leiten auf die drei Experimente in Kapiteln **??** über:

$\mathcal{A}$: Reproduktion des Ansatzes von (Bordes et al., 2013) mit einem kleineren Datenset

$\mathcal{B}$: Training von Relationsrepräsentationen ähnlich (Bordes et al., 2013) mit Wortkontextvektoren

$\mathcal{C}$: Identifikation von Wortpaaren gleicher Relation durch Clustering

All diese Experiment erfolgen auf deutschen Datensätzen. Zuletzt folgt ein Gesamtfazit der Arbeit mit Ausblick (siehe Kapitel 5) mit zwei Appendices A und B mit detaillierten Informationen über Wortkontextvektortraining und -evaluation sowie die Bibliographie. Der erstellte Code sowie dessen Dokumentation sind online[9] abrufbar und zur Wiederverwendung und Modifikation gekennzeichnet.

[9] Siehe https://github.com/Kaleidophon/thesis_ba (zuletzt abgerufen am 05.08.16)

Kapitel 2

Verwandte Arbeiten

2.1 Wortvektorrepräsentationen

Wortkontextvektoren wurden u. A. von (Bengio et al., 2006) präsentiert, bei welchen Informationen über ein Wort für den Menschen uneindeutig und kontinuerlich über den Vektor verteilt sind ("distributed represenation"). Daher leitet sich auch der Name des dazugehörigen Forschungsfeldes - der distributionellen Semantik - her. Im Vergleich zu vorherigen Arbeiten ist bei sog. "One-Hot"-Vektoren jede Dimension einem Wort des Vokabulars eines Korpus zugeordnet, nur jeweils eine davon hat den Wert 1 inne (gewöhnlich besitzen alle anderen den Wert 0). Die "Distributionalität" ist lediglich die Folge eines viel grundlegeren Unterschied der Methodik, der von (Baroni, Dinu und Kruszewski, 2014) weiter ausgeführt: Lange hatte man sich mit zählbasierten Methoden beschäftigt, um solche Repräsentationen zu erstellen; mit (Bengio et al., 2006) hielten vorhersagebasierte Methoden schließlich Einzug, die zuerst überlegen schienen (Baroni, Dinu und Kruszewski, 2014). Dies konnte allerdings von (Levy, Goldberg und Dagan, 2015) widerlegt werden. Der Unterschied im Vergleich zu anderen Methoden wurde dabei vor allem durch die Konfiguration von Hyperparametern und Unterschiede in den Vorbereitungen der Experimente erklärt.

Diese fußen auf der *distributional hypothesis*, die besagt, dass Worte, die in ähnlichen Kontexten auftauchen, dazu tendieren, eine ähnliche Bedeutung zu haben (Harris, 1954). Diese Besonderheit machen sich (Bengio et al., 2006) zunutze, da das Neuronale Netz, mit denen die Wortkontextvektoren erzeugt werden, beim Bewegen durch einen Trainingskorpus ein mehrwortiges Kontextfenster um ein Zielwort herum verwendet. Das System lernt, die Wortvektoren an häufig vorkommende Wortkontexte anzupassen. Selbige besitzen zudem weitere nützliche Eigenschaften, die in Kapitel 3 beschrieben werden.

Diese Forschungsarbeit löste eine Welle weiterer Forschungen in diesem Gebiet aus, da gezeigt werden konnte, dass diese Art der Wortrepräsentationen genutzt werden können, um die Leistungen von Systemen selbst im Bezug auf lange bekannte und erforschte Probleme der Computerlinguistik signifikant zu verbessern. Exemplarisch sei hier die Arbeit von (Collobert et al., 2011) genannt, die so neue Ansätze für PoS-Tagging, Named-Entity-Recognition, Chunking und Semantic Role Labeling präsentieren. (Mikolov et al., 2013b) und (Mikolov et al., 2013a) beschäftigen sich mit der Optimierung des Wortvektortrainings. Dabei wird auch auf die arithmetischen Regelmäßigkeiten von Wortvektoradditionen eingegangen (siehe Abbildung

1.1 im vorherigen Kapitel). Diese Eigenschaft soll später in Kapitel **??** erneut aufgegriffen werden.

Weitere Untersuchungen beschäftigen sich u. A. mit bilingualen Repräsentationen für maschinelle Übersetzung (Zou et al., 2013), deren Training auf der Basis eines dependenzgeparsten Korpus (Levy und Goldberg, 2014) und dem Optimieren der Parameter (Levy, Goldberg und Dagan, 2015). (Fu et al., 2014) nutzen die Differenzen von Wortkontextvektoren, um semantische Hierarchien zu erstellen.
Diese Forschungsarbeiten zeugen hierbei exemplarisch von dem riesigen Potenzial und der Vielfalt, die dieses Forschungsthema bietet.

2.2 Vektorrepräsentationen für Wissensdatenbanken

Wissendatenbanken sind aufgrund der in Kapitel 1 genannten Anwendungsmöglichkeit bereits ein lange bearbeitetes Forschungsthema. Beispielsweise diskutierten schon (Giaretta und Guarino, 1995) verschiedene Auslegungen des Ontologiebegriffs im Kontext ebendieser Wissensbasen. Dementsprechend wurde auch im Zuge des aufkommenden *world wide web* versucht, diese Ressource zu nutzen, wie z.B. (Craven et al., 2000) zeigt.
Dieses Thema nach dem Vorbild von Wortkontextvektoren zu bearbeiten ist jedoch eine eher neue Herangehensweise: Als einer der ersten Ansätze versuchen (Bordes et al., 2011), symbolische Wissensrepräsentationen in einen Vektorraum einzubetten. Diese sog. *structured embeddings* (SE) werden mithilfe von Neuronalen Netzwerken trainiert. Darauf aufbauend präsentieren (Bordes et al., 2013) einen etwas simpleren Ansatz namens *TransE*, der darauf abziehlt, Relationen zwischen Entitäten als eine einfache vektorarithmetische Operation ($\rightsquigarrow$ Übersetzung) zu sehen. Diese Vorgehensweise wird in Kapitel **??** etwas genauer ausgeführt und deren Ergebnisse für einen deutschsprachigen Datensatz repliziert.

Diese Idee wird weiter von (Wang et al., 2014) vorangetrieben: Um nicht nur 1:1- sondern auch 1:n-, n:1- und m:n-Relationen abzubilden, werden Punkte in einem Vektorraum auf eine für jede Relation separat gelernte Ebene projiziert, woher sich der Name des Verfahrens, *TransH*, herleitet.
Um die Vektorräume von Entitäten und Relationen zu trennen, stellen (Lin et al., 2015) *TransR* und *CTransR* vor. Für Ersteres wird für jede Relation eine Projektionsmatrix gelernt, die Entitäten in den jeweiligen Relationsvektorraum übersetzt. Bei Letzterem werden für jede Relation mehrere Vektoren trainiert, um den Unterschieden im Kontext anderer Entitäten gerecht zu werden. Eine Übersicht über die Verlustfunktionen aller Verfahren findet sich in Abbildung 2.1.

MODELL	VERLUSTFUNKTION
TransE	$f_r(h,t) = \| h + r - t \|_2^2$
TransH	$f_r(h,t) = \| h_\perp + r - t_\perp \|_2^2$
TransR	$f_r(h,t) = \| h_r + r - t_r \|_2^2$
	$h_r = hM_r$
	$t_r = tM_r$
CTransR	$f_r(h,t) = \| h_{r,c} + r_c - t_{r,c} \|_2^2 + \alpha \| r_c - r \|_2^2$
	$h_{r,c} = hM_r$
	$t_{r,c} = tM_r$

Erklärung der Parameter

- (h, r, t): Relationstripel bestehend aus (den Vektoren) einer Kopf- (h) und Fußentität (t) sowie einer Relation r
- $f_r(\cdot, \cdot)$: Bewertungsfunktion einer Relation r im Bezug auf zwei Entitäten
- $h_\perp, t_\perp$: Projektionen zweiter Entitätsvektoren auf eine Ebene
- M: Projektionsmatrix
- $(h_{r,c}, r_c, t_{r,c})$: Relationstripel mit auf ein Subcluster einer Relation trainieren Vektorrepräsentationen
- α: Gewichtsparameter zur Einschränkung für den Abstand einer Subrelation zur Hauptrelation

Tabelle 2.1: Übersicht über verschiedene Arten der Vektorrepräsentationen für Wissensdatenbanken Für *TransE* (Bordes et al., 2013), *TransH* (Wang et al., 2014), *TransR* und *CTransR* (Lin et al., 2015) werden die verschiedenen Scoringfunktionen gegenübergestellt und vorkommende Parameter erklärt.

Kapitel 3

Grundlagen

> *"To deal with hyper-planes in a 14-dimensional space, visualize a 3-D space and say 'fourteen' to yourself very loudly. Everybody does it."*
>
> GEOFFERY HINTON

3.1 Neuronale Netzwerke

Wie der Name bereits erahnen lässt, fußt die Idee von Neuronalen Netzwerken auf einer mathematischen Modellierung des menschlichen Gehirns (siehe Abbildung 3.1 und (Rosenblatt, 1958)). Die Grundbausteine bilden dabei einzelne Neuronen, die dabei auf folgende Art und Weise modelliert werden (dieser Teil ist dabei auch als *Perceptron*-Algorithmus bekannt):

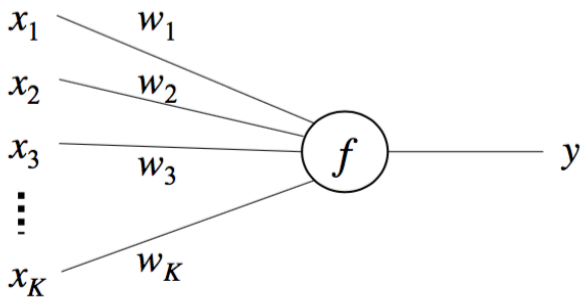

Abbildung 3.1: Mathematische Modellierung eines Neurons, auch bekannt als *Perceptron*-Algorithmus von (Rosenblatt, 1958). x bezeichnet In-, y den Output des Neurons, w bilden die Gewichte und f die Aktivierungsfunktion. Abbildung aus (Rong, 2014).

Der Analogie der menschlichen Nervenzelle folgend erhält das Neuron mehrere Inputs in Form eines Vektors x mit K Dimensionen. Der Output erhält die Bezeichnung y. Um den Output zu bestimmen enthält die Zelle eine Aktivierungsfunktion f (um zu bestimmen, wann das Neuron "feuert"):

$$y = f(u) \tag{3.1}$$

u bezeichnet dabei einen Skalar, den man durch das Summieren der mit w gewichteten Inputs erhält:

$$u = \sum_{i=0}^{K} w_i x_i = w^T x \tag{3.2}$$

Für die Aktivierungsfunktion f bieten sich mehrere Optionen. Ursprünglich wurde die *Heaviside step function* oder *Rectifier (ReLU)* (siehe bspw. (Abramowitz und Stegun, 1964)) gewählt:

$$f(u) = \begin{cases} 1 & \text{if } u > 0 \\ 0 & \text{otherwise} \end{cases} \text{ bzw.} \tag{3.3}$$

$$f(u) = max(0, u) = \begin{cases} 0 & \text{if } u < 0 \\ u & \text{otherwise} \end{cases} \tag{3.4}$$

Andere Funktionen sind z.B. *sigmoid* ($\sigma(u) \in [0, 1]$) und *tanh* ($tanh(u) \in [-1, 1]$), im Gegenteil zu den beiden zuvor sind diese durch ihre s-förmige Form kontinuierlich und dadurch ableitbar:

$$\sigma(u) = \frac{1}{1 + e^{-u}} \tag{3.5}$$

$$tanh(u) = \frac{e^{2u} - 1}{e^{2u} + 1} \tag{3.6}$$

In der jüngeren Vergangenheit fand jedoch wieder eine Rückkehr zu den alten Funktionen statt, siehe z.B. (Maas, Hannun und Ng, 2013).

Um aus einzelnen Neuronen nun ein Neuronales Netzwerk zu kreieren, werden mehrere Schichten erstellt (i.d.R. eine Eingabe- (*input layer*), eine Ausgabe- (*output layer*) und min. eine "versteckte" Schicht (*hidden layer*)). Die Schichten bestehen dann aus mehreren parallelen Zellen, wobei jede Zelle mit jeder anderen Zelle der vorhergehenden und folgenden Schicht vernetzt ist (siehe Abbildung 3.2)[10].

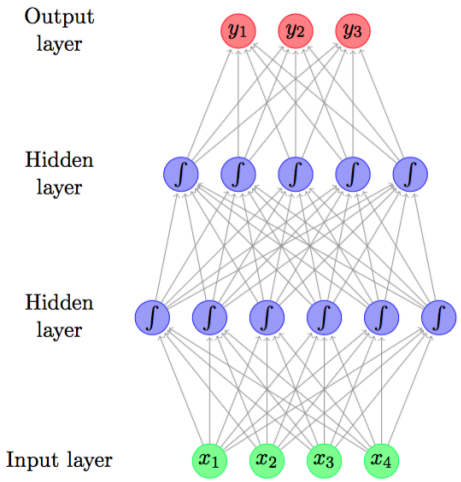

Abbildung 3.2: Darstellung eines Neuronalen Netzwerks mit vier Schichten, davon jeweils eine für In- und Output, sowie zwei "versteckte" Schichten (*hidden layer*). Bild aus (Goldberg, 2015).

Die einzelnen Gewichtsvektoren w aller Neuronen einer Schicht werden dann i.d.R. zu einer Gewichtsmatrix W zusammengefasst, wobei jede Zeile

[10] Dabei gibt es aber auch Netzwerke, bei denen absichtlich Verbindungen ignoriert werden oder Nervenzellen mit Zellen aus einer anderen Schicht als der nächstvorderen verknüpft sind.

einem Gewichtsvektor entspricht. Zum Training dieser Strukturen wird der sogenannte Backpropagation-Algorithmus verwendet, der mithilfe einer vorher definierten Verlustfunktion einen Fehler errechnet. Die Verlustfunktion zeigt dabei an, inwiefern das durch das Netzwerk erzeugte Ergebnis von dem erwünschten Ergebnis abweicht. Dieser wird rekursiv durch alle Schichten des Netzwerks zurückgegeben und passt simultan die Werte innerhalb der Gewichtsmatrix an.

3.2 Wortvektoren

Frühere Experimente mit Wortvektoren arbeiteten meist mit sog. "One-Hot"-Vektoren, bei denen jede Dimension i.d.R. einem bestimmten Wort zugeordnet wurde. Nehmen wir beispielsweise das Minikorpus "Der Hund beißt den Mann" an. Das Vokabular besteht dann aus
V = $\{beisst,\ Der,\ den,\ Hund,\ Mann\}$ (in alphabetischer Reihenfolge). Um jedes dieser Wörter als einen der oben genannten Vektoren zu repräsentieren, können wir Vektoren der Länge V benutzen[11]. Um nun zum Beispiel einen Vektor für *Hund* zu generieren, setzen wir den Wert an der Stelle des Vektors (= *Feature*) auf 1, die dem Index von Hund in V entspricht, also $\vec{v}(Hund) = (0, 0, 0, 1, 0)$.

Diese Art von Wortvektor wird gemeinhin als "sparse", also spärlich bezeichnet, da sie relativ wenig Information enthält. Wortkontextvektoren, die mithilfe von Neuronalen Netzwerken trainiert werden, beinhalten mehr (semantische) Informationen über das dazugehörige Wort, zudem lassen sich einzelne Features nicht mehr eindeutig auf bestimmte Worte zurückführen[12]. Ein populäres Werkzeug zur Erstellung der Wortkontextvektoren ist `word2vec` von (Mikolov et al., 2013b).

Das Training dieser neuen Wortkontextvektoren läuft wie folgt ab: Als Input fungieren die erwähnten "One-Hot"-Vektoren, welche genau so viele Dimensionen wie Worte im Vokabular besitzen ($\vec{v} \in \mathbb{R}^V$). Die Modelle bestehen aus drei Schichten, namentlich *Input*, *Hidden* und *Output*. Input und Output besitzen die Dimensionalität von V, Hidden die von N, was die gewünschte Anzahl der Dimensionen der Wortkontextvektoren entspricht.

Zwischen Input und Hidden liegt die Gewichtsmatrix W und zwischen Hidden und Output die Matrix W'. Um diese anzupassen, kann zwischen zwei verschiedenen Ansätzen gewählt werden: *Continuous Bag of Words* (CBOW) versucht, die Wahrscheinlichkeit eines Wortes gegeben eines Kontextes der Größe C zu maximieren[13]. Unsere Verlustfunktion, deren Wert es dabei zu minimieren gilt, besteht in der negativen logarithmischen Wahrscheinlichkeit eines Wortes w_O gegeben seines Kontextes $w_{I,1} \dots w_{I,C}$ mit der Größe C:

$$E = -log\ p(w_O|w_{I,1}, \dots, w_{I,C}) \tag{3.7}$$

[11] V steht eigentlich für ||V||, wird der Übersicht wegen aber im Folgenden stellvertrentend dafür verwendet.

[12] Die semantische Information ist *implizit* in der Vektorrepräsentation kodiert.

[13] Kontext bezieht sich in diesem Fall auf die Summe der rechts und links vom Eingabewort stehenden Wörter, siehe Abbildung 3.3.

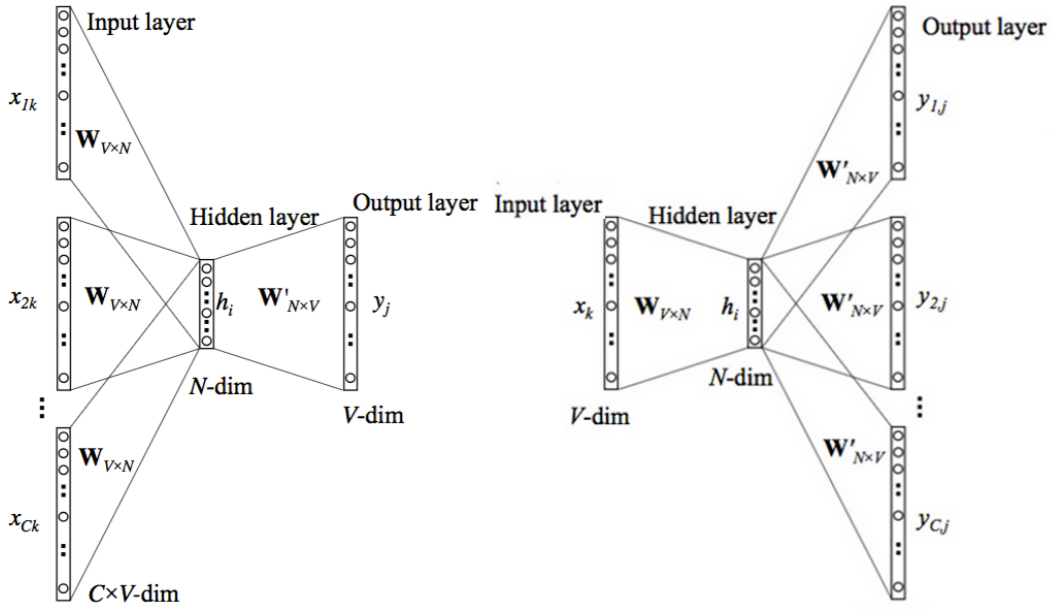

Abbildung 3.3: Gegenüberstellung der beiden Trainingsmethoden CBOW (links) und Skip-Gram (rechts). CBOW versucht die Wahrscheinlichkeit eines Wortes gegeben seines Kontexts zu trainieren, Skip-Gram die Wahrscheinlichkeit eines Kontextes gegeben eines Wortes. Modifizierte Abbildung nach (Rong, 2014).

Beim Skip-gram-Modell hingegen verhält sich das Ganze genau umgekehrt, es wird versucht, den Kontext gegeben eines Eingabewortes vorherzusagen:

$$E = -log\ p(w_{I,1}, \ldots, w_O) \tag{3.8}$$

In beiden Fällen wird daraufhin überprüft, ob die Vorhersage mit den tatsächlichen Daten übereinstimmt und die Abweichung errechnet, mit der die Parameter der Gewichtsmatrizen W und W' rekursiv angepasst werden, um zukünftige Prognosen zu verbessen, wofür der *Backpropagation*-Algorithmus verwendet wird. Die Wortkontextvektoren, die aus dem Durcharbeiten aller Trainingsdaten resultieren, sind dann die Zeilen von W', wobei die *i*-te Zeile der Matrix dem Wortvektor des Wortes mit dem Index i im Vokabular entspricht.

Eigentlich erfordert das Training eine aufwendige Berechnung über alle Wörter des Vokabulars, was der Skalierbarkeit dieses Verfahrens entgegensteht. Der rechnerische Aufwand kann allerdings durch Techniken wie *Hierarchisches Softmax*, bei dem der Aufwand durch einen binären Baum von $O(V)$ auf $O(log\ V)$ verkleinert wird sowie *Negativem Sampling* reduziert werden (Rong, 2014; Goldberg und Levy, 2014). Bei letzterem werden Negativbeispiele zum Training hinzugezogen, woher sich auch der Name des Verfahrens ableitet.

3.3 Clustering

Clustering bezeichnet eine Art von unüberwachtem maschinellen Lernen, die versucht, Elemente, die nach vorher definierten Maßgaben als ähnlich erachtet werden sollen, zu gruppieren.
Die Literatur zu diesem Thema bietet dabei eine Fülle von Algorithmen mit verschiedenen Grundannahmen, aus denen es zu wählen gilt. Für die Aufgaben in dieser Arbeit wurde der DBSCAN-Algorithmus (**D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise) von (Ester et al., 1996) gewählt, da er folgende Kriterien erfüllt:

- DBSCAN erlaubt es, dass nicht alle Datenpunkte nach der Terminierung des Algorithmus einem Cluster zugeordnet sein müssen (*Outlier detection*)
- Die Anzahl der Cluster muss bei Beginn des Algorithmus nicht festgelegt werden.
- Cluster werden nicht nach der räumlichen Distanz zwischen den Punkten gebildet, sondern nach deren Dichte im Raum. Dies lässt auch nicht-sphärische Clusterformen zu[14].

Zur Erklärung von DBSCAN müssen zuerst einige Definitionen erstellt werden (siehe Abbildung 3.4 zur visuellen Darstellung):

- Ein Punkt p in den Daten wird dann als Kernpunkt (*core point*) bezeichnet, sofern mindesten $minPts$ Punkte innerhalb einem Radius ϵ vorhanden sind.
  Diese Punkte sind von p *direkt erreichbar*.
- Ein Punkt q ist von p erreichbar, sofern es einen Pfad $p_1, \ldots, p_n$ mit $p_1 = p$ und $p_n = q$ gibt, wobei jeder Punkt p_{i+1} von einem Punkt p_i direkt erreichbar ist.
- Alle Punkte, die nicht von einem anderen Punkt aus erreichbar sind, sind Ausreißer (*Outlier*).
- Zwei Punkte p und q sind *direkt verbunden*, sofern es einen Punkt o gibt, von dem aus beide Punkte erreichbar sind.
- Cluster bestehen aus Punkten, die alle miteinander direkt verbunden und vom Kernpunkt aus erreichbar sind.

Der Algorithmus ist von linearer Komplexität, sofern ihm eine geeignete Indexierung der Daten zugrunde gelegt wird. Ansonsten verhält sich die Komplexität quadratisch zur Anzahl der Datenpunkte[16].

Versuche, den Algorithmus zu parallelisieren und somit eine bessere Skalierbarkeit zu ermöglichen wurden beispielsweise von (Arlia und Coppola,

[14] Vgl. dazu z.B. http://scikit-learn.org/stable/modules/clustering.html.

[15] Abbildung von Chire - Eigenes Werk, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=17045963 (zuletzt abgerufen am 25.04.16).

[16] Bei der `scikit-learn`-Implementation wird beispielsweise ein Nearest-Neighbour-Graph verwendet: http://scikit-learn.org/stable/modules/generated/sklearn.cluster.dbscan.html (zuletzt abgerufen am 25.04.16)

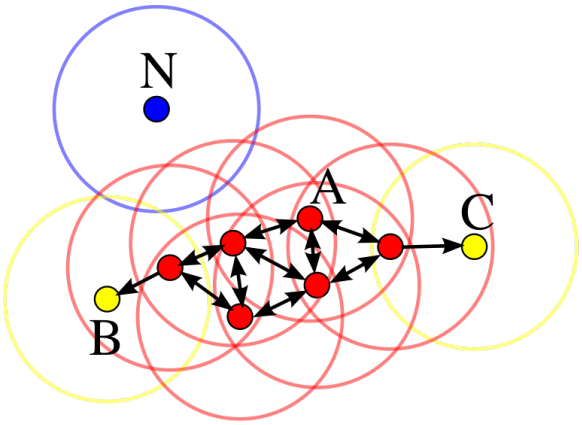

Abbildung 3.4: Darstellung der Funktionsweise von DBSCAN. Punkt A ist ein Kernpunkt, die Punkte B und C sind von A aus erreichbar. Punkt N ist nicht erreichbar und damit ein Ausreißer.[15]

2001) unternommen. Solche Vorgehen sind wichtig, um die Skalierbarkeit des Algorithmus auf große Datenmengen wie beispielsweise in Kapitel **??** zu sichern.

Kapitel 4

Vorbereitung

4.1 Vorbereitung

Für die nachfolgenden Experimente in Kapitel **??** mussten einige Daten zuerst beschafft und unter Umständen aufbereitet werden. Die dazu ergriffenen Maßnahmen werden deshalb in diesem Kapitel beschrieben.

4.1.1 Verwendete Daten

Die anfangs verwendeten Daten lassen sich grob in vier Kategorien einteilen: Das Korpus, das Relationsdatenset, Wortähnlichkeitslisten sowie Analogiedatensets (siehe Abbildung 4.1).
Als Korpus wurde das DECOW14X-Korpus verwendet. Genaueres über dessen Beschaffenheit und Vorverarbeitung für das Training von Wortvektoren werden in 4.1.2 und 4.1.3 beschrieben.

Art	Name	Beschreibung	Quelle
Korpora	DECOW14X	Deutschsprachiges Webkorpus	(Schäfer und Bildhauer, 2012)
Relationsdaten	FB14k	592k Relationstripel aus Freebase	(Bordes et al., 2013)
Wortähnlichkeiten	Schm280	280 nach Ähnlichkeit bewertete Wortpaare	(Köper, Scheible und Walde, 2015)
	Wortpaare65	65	(Rubenstein und Goodenough, 1965)
	Wortpaare222	222	
	Wortpaare350	350	
Analogien	Google	Datenset mit 18.552 Analogien	(Köper, Scheible und Walde, 2015)
	SemRel	Datenset mit 2.462 Analogien	

Tabelle 4.1: Übersicht über in dieser Arbeit verwendete, extern generierte Datensets sowie deren Quelle.

Relationsdaten aus der mittlerweile in Wikidata integrierten Wissensdatenbank[17] Freebase wurden im Datenset FB14k von (Bordes et al., 2013) gesammelt.

Alle anderen Datensets sind zur Evaluation der Wortvektoren in Kapitel 4.2 angedacht: Die Sets Schm280, Wortpaare65, Wortpaare222 und Wortpaare350 bestehen aus einer Reihe von Wortpaaren, die von menschlichen Annotatoren auf verschiedenen Skalen nach Ähnlichkeit bewertet wurden.

[17] Siehe https://plus.google.com/109936836907132434202/posts/bu3z2wVqcQc (zuletzt abgerufen am 21.07.16).

GOOGLE[18] und SEMREL sind in verschiedene Kategorien eingeteilte Analogiepaare der Form "W verhält sich zu X wie Y zu Z". Bei der Evaluation wird dann die Fähigkeit eines Systems mit Wortvektoren getestet, diese korrekt zu vorvervollständigen, wenn Z nicht gegeben ist.

4.1.2 Aufbereitung des Korpus

Als Textressource wurde das DECOW14X-Korpus (DE = Deutsch, COW = "**CO**rpus from the **W**eb") verwendet. Dieses Korpus von (Schäfer und Bildhauer, 2012) besteht aus 21 Teilkorpora, die in den Jahren 2011 und 2014 von deutschsprachigen Internetseiten extrahiert und aufbereitet wurden. Diese beinhalten Informationen im Bezug auf PoS-Tagging, Chunking, Lemmatisierung, Eigennamen (*named entities*) und einige Metadaten. Die Sätze liegen im CONLL-Format[19] vor, wobei jedem Wort und dessen Annotationen eine ganze Zeile gewidmet ist. Stazgrenzen werden durch XML-Tags getrennt. Summa summarum enthält das Korpus 624.767.747 Sätze mit 11.660.894.000 Tokens.

Für diese Arbeit wurden auf Basis der Ressource zwei Versionen für das Training der Wortvektoren erstellt:

- Eine Datei mit den originalen Tokens durch Leerzeichen getrennt, je ein Satz pro Zeile.
- Eine Datei mit den lemmatisierten Tokens durch Leerzeichen getrennt, je ein Satz pro Zeile.

Um die beiden Varianten zu erstellen, wurden die jeweils relevanten Informationen aus den dazugehörigen Spalten des CONLL-Formats verwendet.
Darüber hinaus wurden für spätere Anwendungen in Kapitel **??** alle Eigennamen sowie die Satz-IDs für alle Eigennamen gesammelt (siehe die Kookkurrenzeinschränkung **??**).

4.1.3 Training der Wortvektoren

Wortvektoren wurden mithilfe des Tools `word2vec` und zwei verschiedenen Modellen trainiert: *Continuous Bag of Words* (CBOW) und Skip-Gram. Das CBOW-Model wurde zuerst von (Mikolov et al., 2013b) vorgestellt. Die Erklärung der Funktionsweise wird im nachfolgenden Teil recht klein gehalten, für eine ausführlichere und verständliche Ausführung wird beispielsweise die Arbeit von (Rong, 2014) oder die Ausführung in Kapitel 3.2 empfohlen.

Als Eingabe wird eine Textressource benötigt, die einen Satz pro Zeile enthält, wobei Tokens durch Leerzeichen getrennt sind; die Wortvektoren können entweder in einem einfachen Text- oder Binärformat gespeichert werden.
Das Tool lässt zudem dem Nutzer offen, einige Parameter zu verändern. Jene, die in dieser Arbeit berücksichtigt wurden, sollen dabei näher erläutert werden:

[18] Eigentlich: GOOGLE SEMANTIC/SYNTACTIC ANALOGY DATASETS
[19] Siehe http://ilk.uvt.nl/conll/ (zuletzt abgerufen am 11.04.16)

- `-sample`
  Die Wahrscheinlichkeit, mit der der Einfluss hochfrequenter Worte im Training geregelt wird

- `-cbow`
  Bestimmt, welche Trainingsmethode verwendet wird ($0 \mathrel{\hat{=}}$ Skip-gram, $1 \mathrel{\hat{=}}$ Continuous-Bag-of-Words)

- `-negative`
  Anzahl von negativen Beispielen beim Training.

Zwar bietet das Tool auch noch andere Parameter, jedoch soll aufgrund der Empfehlungen in (Levy, Goldberg und Dagan), in der eine große Anzahl von Konfigurationen ausprobiert wurde, im Rahmen dieser Arbeit nur mit den oben genannten Werten experimentiert werden. Weiterführende Optimierungen wurden zudem unterlassen, um den Rahmen dieser Arbeit nicht zu sprengen.

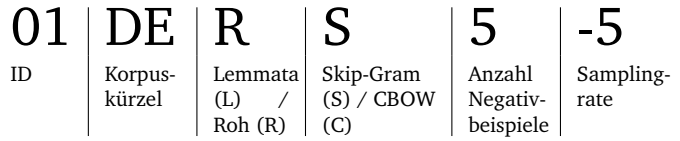

Abbildung 4.1: Erklärung der Nomenklatur für Datensets aus Wortkontextvektoren, hier für 01DERS5-5.

Im Folgenden wird für ein spezielle Nomenklatur für jedes Vektordatenset eingeführt (siehe Abbildung 4.1): So bezeichnen die ersten beiden Ziffern die Nummer des jeweiligen Datensets, die nächsten beiden Buchstaben das Korpus, das für das Training verwendet wurde, das nächste Zeichen, ob das Korpus lemmatisiert (L) oder roh vorlag (R). Danach folgt ein Hinweis auf das im Training verwendete Modell (also Skip-Gram (S) oder CBOW (C)) und schlussendlich die Anzahl der negativen Trainingsbeispielen sowie die Samplingrate.

4.2 Evaluation der Wortkontextvektoren

An dieser Stelle sollen die verschiedenen Ansätze zum Evaluieren von Wortkontextvektoren miteinander verglichen werden. Zu diesem Zweck sollte zuerst eine Frage gestellt werden: Was macht eine Menge von Wortkontextvektoren “besser” bzw. “schlechter” als andere?
Da der Vorteil von Wortkontextvektoren darin besteht, semantische Information zu beinhalten, wird diese Frage meist dahingehend beantwortet, dass Vektoren dann als überlegen anzusehen sind, wenn sie eine höhere semantische “Ausdruckskraft” besitzen. Um diese festzustellen, haben sich in Veröffentlichungen zu diesem Thema bestimmte Vorgehensweisen durchgesetzt, die in den folgenden Abschnitten, vorgestellt, erläutert, angewendet und kritisch reflektiert werden sollen. Gegen Ende des Kapitel werden darüber hinaus die in Kapitel 3 erzeugten Wortkontextvektoren mit gesagten Methoden evaluiert.

4.2.1 Qualitative Evaluation

Qualitative Verfahren zur Evaluation sind meist recht simple Ansätze, die für das menschliche Auge leicht zu interpetierende Ergebnisse liefern. Deshalb sind sie für einen ersten Eindruck auch durchaus geeignet, sollten wenn möglich aber nicht als alleinige Kriterium für eine Bewertung verwendet werden, da sie meistens nur einen kleinen Ausschnitt aller in den Ergebnissen enthaltenen Informationen darstellen können.
Beim Beispiel der Wortkontextvektoren werden beispielsweise einige Wörter des Vokabulars stellvertretend ausgewählt und zu diesen die k nächsten Nachbarn[20] im Vektorraum gesucht. Unter der Annahme, dass in Vektorräumen von Wortkontextvektoren ähnliche Wörter nahe beieinanderliegen, sollte diese Liste im Idealfall eng verwandte Begriffe zutage fördern (siehe Abbildung 4.2).

NAME	FRANKREICH	BANK	COMPUTERLINGUISTIK	GEKRATZT
01DERS5-5	BELGIEN	ANLAGEBANK	LINGUISTIK	GEKRAZT
	ITALIEN	(UNKNOWN)_BANK	INFORMATIONSWISSENSCHAFT	WEGGEKRAZT
	NIEDERLAND	BANK	TEXTTECHNOLOGIE	GESCHUBBERT
	NIEDERLANDE	BANKSTATUS	SPRACHWISSENSCHAFT	GEKRATZ
	SPANIEN	GELDINSTITUT	SOFTWARETECHNIK	ABGELECKT
08DERC5-4	ITALIEN	HAUSBANK	BIOINFORMATIK	GEKRAZT
	UNGARN	GESCHÄFTSBANK	TEXTTECHNOLOGIE	WEGGEKRAZT
	POLEN	GROSSBANK	INFORMATIONSWISSENSCHAFT	GESCHUBBERT
	SPANIEN	KREDITBANK	SOFTWARETECHNIK	ANGEKNABBERT
	BELGIEN	MUTTERBANK	LINGUISTIK	RAUSGERISSEN
22DELC5-2	ITALIEN	KREDITINSTITUT	BIOINFORMATIK	GERITZT
	UNGARN	GELDINSTITUT	WIRTSCHAFTSGEOGRAPHIE	ANGESENGT
	POLEN	SPARKASSE	GRUNDSCHULPÄDAGOGIK	ZUSAMMENGENÄHT
	SPANIEN	LANDESBANK	COMPUTERWISSENSCHAFT	AUFGESPRUNGEN
	BELGIEN	FINANZINSTITUT	HUMANGEOGRAPHIE	TUPFEN

Tabelle 4.2: Listen der k nächsten Nachbarn von Wörtern in verschiedenen Datensets. Inspiriert von (Collobert et al., 2011).

Das Problem bei dieser Methode liegt in der menschlichen Subjektivität: Die präsentierte Auswahl der Begriffe muss nicht zwangsläufig repräsentativ für die restlichen Daten sein und könnte theoretisch aus den wenigen, gut funktionierenden Beispielen bestehen. Darüber hinaus bleibt es in einigen Fällen schwierig, die Ergebisse verschiedener Datensets zu vergleichen, da sich die Qualität der k Nachbarn nicht quantifizieren lässt: Es lässt sich vielleicht erkennen, das diese in einem Fall wenig Sinn machen und im anderen Fall die Erwartungen erfüllen; an anderer Stelle scheinen die Resultate für den Betrachter jedoch nicht unbedingt schlechter, sondern einfach nur anders.
Darum ist wiederum festzuhalten, dass sich qualitative Methoden in diesem

[20] I.d.R. basierend auf der euklidischen Distanz.

Fall eher für den ersten Eindruck eignen, weiterhin aber Prozeduren mit quantifizierbaren Ergebnissen verwendet werden sollten, wie z.B. nächsten Abschnitt 4.2.2 beschrieben werden.

4.2.2 Quantitative Evaluation

Bei der quantitativen Evaluation von Wortkontextvektoren werden die folgenden Ideen aufgegriffen:

1. **Benchmark-Tests**
   Bei dieser pragmatischen Art der Bewertung werden wird das Datenset als Grundlage für einfach Aufgaben wie Sentiment-Klassifikation oder Part-of-Speech-Tagging verwendet. Die Qualität der Daten wird dann anhand der Ergebnisse des Systems gemessen.
   Diese extrinische Evaluationsmethode macht ergo nur dann Sinn, wenn man mehr als ein Datenset miteinander vergleicht. Dabei muss sichergestellt werden, dass die Tests immer unter den selben Bedingungen ablaufen, damit eine Vergleichbarkeit gewährleistet bleibt.

2. **Wortähnlichkeit**
   Hierbei werden Wortpaaren Ähnlichkeitswerte von menschlichen Annotatoren zugeordnet. Anschließend werden mit den zu Verfügung stehenden Vektoren Ähnlichkeitswerte für die gleichen Paare berechnet, in der Regel mithilfe der Kosinus-Ähnlichkeit. Diese beschreibt die Ähnlichkeit zweier Vektoren als den Winkel zwischen ihnen, mit einem Wert von -1 ($\hat{=}180°$ bzw. komplett unterschiedlich) über 0 ($\hat{=}90°$) und $+1$ ($\hat{=}0°$ bzw. Äquivalenz):

   $$cos(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|\|\vec{b}\|} \tag{4.1}$$

   Danach kann mit *Spearman's* ρ [21]. anschließend festgestellt werden, ob die beiden Werte für die Wortpaare korrelieren, sprich ob das System Paaren, denen von Menschen ein hoher Ähnlichkeitswert zugewiesen wurde, auch eine hohe Ähnlichkeit zuschreibt. Dabei zeigt $\rho \in [-1, 1]$ den Grad der Korrelation, wobei -1 einer starken negativen, $+1$ einer starken positiven Korrelation entspricht.

3. **Analogien**
   Die dritte Methode basiert auf Analogien der Form *a verhält sich zu a' wie b zu b'*. Die Daten werden nun dahingehend getestet, indem unter Gebrauch der Cosinus-Ähnlichkeit das $\tilde{b}$ aus dem Vokabular $\mathcal{V}$ gesucht

[21] Auch *Spearman's rank correlation coefficient*. Berechnet wird dieser für zwei Mengen von Datenpunkten $X = \{x_i\}_{i=1}^n$ und $Y = \{y_i\}_{i=1}^n$ durch

$$r_s = \frac{\sum_i (rg(x_i) - \overline{rg}_x)(rg(y_i) - \overline{rg}_y)}{\sqrt{\sum_i (rg(x_i) - \overline{rg}_x)^2} \sqrt{\sum_i (rg(y_i) - \overline{rg}_y)^2}}$$

wobei jedem Wert x_i und x_j ein Rang $rg(\cdot)$ zugewiesen wird. $\overline{rg}_{(\cdot)}$ entspricht dem durchschnittlichen Rang.

wird, welches besonders ähnlich zu b und a' aber unähnlich zu a ist:

$$b' = \underset{\tilde{b} \in \mathcal{V}}{argmax}\ cos(\tilde{b}, b - a + a') \tag{4.2}$$

Sind alle Vektoren der Länge eins, so kann diese Gleichung aufgrund der obigen Definition der Kosinusähnlichkeit (4.1) umformuliert werden:

$$b' = \underset{\tilde{b} \in \mathcal{V}}{argmax}\ cos(\tilde{b}, b) - cos(\tilde{b}, a) + cos(\tilde{b}, a') \tag{4.3}$$

Diese Methode wird gemeinhin als 3COSADD bezeichnet. (Levy, Goldberg und Ramat-Gan, 2014) etablierten dazu jedoch eine Alternative namens 3COSMUL[22], die bei Tests bessere Ergebnisse produziert:

$$b' = \underset{\tilde{b} \in \mathcal{V}}{argmax} \frac{cos'(\tilde{b}, b)\ cos'(\tilde{b}, a')}{cos'(\tilde{b}, a) + \epsilon} \tag{4.4}$$

Dabei ist $\epsilon = 0,001$, um die Divison durch Null zu verhindern.
Der Erfolg der Evaluation kann dann als Anteil der richtig vervollständigten Analogien (bei denen $\tilde{b}^* = b^*$) gemessen werden.

In dieser Arbeit sollen die Datensets durch die zweit- und drittgenannte Methode evaluiert werden. Ein weiterer Fallstrick liegt allerdings in der Zusammenstellung der Datensets: So liefern die genannten Evaluationsdatensets nur dann aussagekräftige Ergebnisse, wenn bei der Wortähnlichkeit die menschlichenen Annotatoren zuverlässig und sinnvoll die Paare bewertet haben (zu messen z.B. mit $Cohen's\ \kappa$) und bei den Analogien aus der Zusammenstellung ebendieser (soll heißen: Welche Entitäten sind enthalten? Wie oft kommen diese im Datenset vor? Welche semantische Relationen wurden ausgewählt? Wurden diese maschinell oder per Hand erzeugt?).
Aus diesem Grund sollen die benutzten Evaluationssets im hierauf folgenden Abschnitt 4.2.3 näher beleuchtet werden.

4.2.3 Evaluationsdaten

Wortpaarähnlichkeit

Im Englischen wird für die Wortähnlichkeitsevaluation häufig das WORDSIM353-Datenset von (Finkelstein et al., 2001) verwendet. Dieses wurde unter dem Namen SCHM280 in deutsche portiert, wobei die Paare nicht nur einfach übersetzt, sondern die Ähnlichkeit auch noch von deutschen Muttersprachlern neu bewertet wurde. Es enthält insgesamt 280 Wortpaare.[23]

WORTPAARE65, WORTPAARE222 und WORTPAARE350 entstammen der Arbeit von (Rubenstein und Goodenough, 1965). Dabei werden Wortpaaren

[22] Um nur Werte im Bereich $[0, 1]$ zu erhalten, wird die Formel für die Kosinusähnlichkeit folgendermaßen angepasst:

$$cos'(\vec{a}, \vec{b}) = \frac{cos(\vec{a}, \vec{b}) + 1}{2}$$

[23] Es konnte jedoch nicht festgestellt werden, wieviele Menschen die Wortpaare neu bewertet haben. Während des Übersetzungsschritts wurde eine Zahl von 12 Teilnehmern angegeben, ob jedoch genauso viele im Bewertungsschritt partizipiert haben und wie groß ihre Übereinstimmung war, ist nicht festzustellen.

Werte von 0 ($\hat{=}$ vollkommen unzusammenhängend) bis 4 ($\hat{=}$ stark zusammenhängend) zugeschrieben. Die Anzahl der menschlichen Annotatoren sowie deren Übereinstimmung in Form von $Cohen's\ \kappa$[24] sind in Abbildung 4.3 festgehalten.

DATENSET	#ANNOTATOREN	κ
SCHM280	12 (?)	???
WORTPAARE65	24	0,81
WORTPAARE222	21	0,49
WORTPAARE350	8	0,69

Tabelle 4.3: Anzahl der Annotatoren und Agreement (als $Cohen's\ \kappa$) der Wortähnlichkeitsdatensets.

Analogien

Die GOOGLE SEMANTIC/SYNTACTIC ANALOGY DATASETS wurden von (Mikolov et al., 2013b) eingefügt und bestehen aus Analogien der Form *a verhält sich zu a' wie b zu b'*. (Köper, Scheible und Walde, 2015) haben diese manuell übersetzt und durch drei menschliche Prüfer validieren lassen. Dabei wurde die Kategorie "adjektiv - adverb" ausgelassen, da sie im Deutschen nicht (im gleichen Ausmaß wie im Englischen) existiert, wodurch 18.552 Analogien übrigbleiben. Diese werden im Folgenden einfach als GOOGLE bezeichnet.

SEMREL wurde aus Synonomie-, Antonomie- und Hypernomie-Beziehungen von (Köper, Scheible und Walde, 2015) für das Deutsche und Englische konstruiert. Dabei werden Substantive, Verben und Adjektive berücksichtigt. In der deutschen Variante sind 2.462 (recht schwierige zu vervollständigende) Analogien enthalten, die aus teilweise sehr seltenen Wörter kreiert wurden (siehe eine Kritik dazu im nächsten Abschnitt 4.2.4).

4.2.4 Evaluationsergebnisse

Im Folgenden sollen die Ergebnisse der Wortkontextvektor-Evaluation diskutiert werden. Am Ende dieses Abschnitts werden dabei nur die Datensets mit den besten Ergebnissen und deren Konfiguration von Parametern vorgestellt, eine ausführliche Übersicht findet sich aber im Appendix B.

Nach der Evaluation war zuallererst festzustellen, dass sowohl bei Datensets, die auf dem normalen oder lemmatisierten Korpus trainiert wurden, Ergebnisse mit steigender Downsamplingrate schlechter wurden. Dies war sowohl bei Training mit dem CBOW- als auch mit dem Skip-Gram-Modell feststellbar. Darüber hinaus lieferte Letzteres im Schnitt bessere Resultate

[24] Dies ist definiert als

$$\kappa = \frac{A_o - A_e}{1 - A_e}$$

wobei A_o die beobachtete Übereinstimmung der Annotatoren und A_e die statistisch zu erwartende Übereinstimmung der Annotatoren bezeichnet.

NAME	WORTPAARE65	WORTPAARE222	WORTPAARE350	SCHM280	GOOGLE	SEMREL
01DERS5-5	**-0,8096**	**-0,4640**$_{(13)}$	**-0,7302**$_{(13)}$	**-0,7094**$_{(2)}$	**44,56**	1,71
13DELS5-5	**-0,8247**$_{(1)}$	**-0,5066**$_{(102)}$	**-0,7494**$_{(132)}$	-0,7097$_{(48)}$	15,51	**3,01**
22DELC5-2	-0,8106$_{(1)}$	-0,4953$_{(102)}$	-0,7362$_{(132)}$	**-0,7205**$_{(48)}$	14,51	2,56

Tabelle 4.4: Evaluationsergebnisse der drei besten Vektordatensets. Links: Bewertung durch Spearman's ρ. Rechts: Treffer beim Vervollständigen von Analogien in Prozent. Nicht gefundene Wortkontextvektoren klein in Klammern hinter dem Wert. Für eine vollständige Liste der Trainingsparameter jedes Datensets siehe A. Für die gesamte Liste aller Ergebnisse siehe B.

(und benötigte während des Trainings auch weniger Zeit). Das Lemmatisieren führte dazu, das während der Bewertung der Sets oft einige Schritte nicht durchgeführt werden könnten, was anhand der in Subskript in Klammern stehenden Zahlen abzulesen ist. Darunter leidet leider auch wenig die Vergleichbarkeit; es erscheint aber zumindest logisch, dass bei gleichem Evaluationswert einem Datenset mit weniger nicht gefundenen Wortkontextvektoren eine höhere Qualität beizumessen ist. Je nach späterem Anwendungsgebiet kann diese Unterscheidung aber nicht relevant sein.
Bei den Analogie-Datensets sind ähnliche Tendenzen sichtbar: Beim GOOGLE-Datenset sind die Diskrepanzen jedoch deutlich stärker und Vektoren des unlemmatisierten Korpus schneiden deutlich besser ab, was vor allem dadurch zu erklären ist, dass in diesem Set auch Flektion geprüft wird (bspw. *schreiben* verhält sich zu *schreibt* wie *sagen* zu *sagt*) und diese Formen durch die Lemmatisierung velorengehen. Beim SEMREL-Datenset verhält sich das ganze allerdings genau umgekehrt, wenn auch die Unterschiede wesentlich feiner sind. Da hier für alle möglichen Formen eines Wortes nur eine Vektorrepräsentation trainiert wird, ist zu schließen, dass diese dann auch mehr Information in sich kodiert.

Es ist ferner zu erkennen, dass einige der extern bereitgestellten Evaluationsdaten nicht sehr gut zusammengestellt wurden. Bei WORTPAARE222 prägen sich die Korrelationswerte nicht unter $-0,54$ aus, was dafür spricht, das die Ähnlichkeit der Wortpaare für sowohl für das System als möglicherweise auch für die menschlichen Annotatoren schwer einzuschätzen war [25] Beim SEMREL-Analogienset sticht dieser Fakt noch viel drastischer heraus: Im besten Fall wurden 3% (sic!) der Analogien richtig vom System vervollständigt. Es sei angemerkt, dass es auch dem Autor und anderen gefragten Personen schwerfiel, stichprobenartig ausgewählte Fragen richtig zu beantworten [26]. Deshalb stellen sich die Ergebnisse bei diesem Vergleichsset ohne klare Tendenz in den Resultaten und generell sehr schlecht dar.

[25] Beispielhaft seien hier einige Wortpaare aus WORTPAARE222 genannt, die nach Ähnlichkeit bewertet werden mussten, um das Problem zu illustrieren:
wahrnehmen - Grundsatzfrage / groß - Arbeitszeitregelung / Hubschraubertyp - einschließlich / Büroequipment - Institut.

[26] *Lieblichkeit* verhält sich zu *Anmut* wie *Mittelklasse* zu...? *Mittelschicht*
Zivilgesellschaft verhält sich zu *Bürgergesellschaft* wie *Gegenargument* zu...? *Widerspruch*
Trio verhält sich zu *Solo* wie *Arzt* zu...? *Patient*

Für die nachfolgenden Schritte wurde die bestabschneidenen Wortkontextvektorsets bei GOOGLE und den anderen Wortpaarsets, namentlich 01DERS5-5, 13DELS5-5 und 14DELS5-4 ausgewählt.

Kapitel 5

Fazit

“Mit dem Wissen wächst der Zweifel.”

JOHANN WOLFGANG VON GOETHE

5.1 Zusammenfassung

In dieser Arbeit wurden verschiedene Möglichkeiten präsentiert, Wissen aus Wissensdatenbanken durch kontinuierliche Vektorrepräsentationen darzustellen und zu erweitern. In Kapitel **??** wurden unterschiedlich komplexe Herangehensweisen vorgestellt, Repräsentationen für Entitäten und Relationen einer solchen Datenbank gleichzeitig mithilfe von Methoden des Maschinellen Lernens zu trainieren. Für das TransE genannte Verfahren wurden zusätzlich Ergebnisse mit einer deutschsprachigen Untermenge der Daten (GER14K) durchgeführt, die zwar bei der Evaluation etwas schlechter ausfielen, aber dennoch vergleichbar waren.

In Kapitel **??** wurde versucht, Wortkontextvektoren in das Verfahren von Kapitel **??** einzubinden und nur noch Vektorrepräsentationen für Relationen zu trainieren. Hierfür wurde ebenfalls ein Datenset namens WE4K erzeugt. Wegen der Datenknappheit und anderen in **??** diskutierten Gründen konnten hierbei keine vergleichbaren Ergebnisse erlangt werden. Dennoch konnten einige Erkenntnisse über Wortkontextvektoren im Speziellen und Vektorrepräsentationen im Allgemeinen daraus gezogen werden, die im nächsten Abschnitt reflektiert werden.

In Kapitel **??** wurde ein eigener Versuch unternommen, ähnliche Ergebnisse mithilfe von Wortkontextvektoren zu erreichen, zu deren Erstellung das Tool `word2vec` verwendet wurde, welches auf großen Korpora arbeitet. Dazu wurde der deutschsprachige Internetkorpus DECOW14X aufbereitet. Im Folgenden wurde angestrebt, Wortvektorpaare durch ihre Differenzvektoren verschiedenen Relationen zuzuteilen. Dies schlug jedoch fehl, da der Ansatz auf einem fehlerhaften Annahme fußte. Deshalb konnten keine evaluierbaren Ergebnisse erzeugt werden. Eine ausführliche Diskussion darüber findet sich in **??**.

5.2 Diskussion

Die Implikationen der in dieser Arbeit gewonnenen Erkenntnisse sollten mit Bedacht diskutiert werden. Das Fehlschlagen eines Ansatzes auf der Basis von Wortkontextvektoren heißt im Umkehrschluss nicht, dass alle Vorstöße

in dieser Richtung zum Scheitern verurteilt sind (im Gegenteil soll darauf in 5.3 eingegangen werden).
Es scheint jedoch unbestreitbar leichter, Repräsentationen gleich im Kontext der Struktur einer Wissensdatenbank als im Kontext eines großen Korpus zu trainieren, da diese so von Anfang an auf für Wissensbasen wichtige Eigenschaften getrimmt werden (nämlich die Beziehung zu anderen Relevanten Entitäten). Jedoch erfordern solche Repräsentationen bereits viele vorstrukturierte Datensätze, die von Menschen erstellt werden müssen.

Das Fehlschlagen des vorgestellten Ansatzes in Kapitel **??** scheint vor allem auf einen Fehler in der Grundannahme und die schlechte Skalierbarkeit zurückzuführen. Um Letzteres zu verhindern, könnten in zukünftigen Ansätzen weitere Informationsquellen hinzugezogen werden, um "sinnvolle" von "sinnlosen" Wortpaaren zu trennen. Da davon auszugehen ist, dass in der Menge aller Möglichen Wortpaare nur ein sehr kleiner Prozentsatz tatsächlich Sinn ergeben dürfte, sollte dieses Vorgehen die totale Rechenzeit signifikaten verringern.
Der der Hypothese zugrunde liegende logische Fehler scheint im Nachhinein offensichtlich. Dieser war aber schwer vorherzusehen, nachdem in vielen anderen Arbeiten die semantischen Eigenschaften von arithmetischen Rechnungen mit Wortvektoren so stark hervorgehoben wurden. Zudem wird dieser Aspekt immer nur anhand von in einer offensichtlichen Beziehung zueinander stehenden Wortpaaren illustriert. Dadurch fiel dieses Versäumnis erst auf, als es aus zeitlichen Gründen nicht mehr möglich war, im Rahmen dieser Abschlussarbeit andere Verfahren zu testen und zudem Versuche, auf Basis der vorliegenden Daten diesen Fehler zu korrigieren fehlgeschlagen waren.

5.3 Ausblick

Der vorliegende Ansatz bietet vielerlei Möglichkeiten zur Verbesserung. Im Folgenden sollen einige davon skizziert werden:
Zuallererst erscheint es sinnvoll, das Konzept von einem unüberwachten zu einem überwachten Ansatz zu ändern, also eine maschinellen Lernmethode mit Daten zu versorgen, denen bereits eine Klasse zugewiesen wurde. Beispielsweise könnten eine sog. *Support Vector Machine* (SVM) oder ein anderer Lernalgorithmus ein zu einer bestimmten Relation gehöriges Wortpaar als Eingabe nehmen, um anhand dessen zu lernen zu unterscheiden, welche Differenzvektoren für eine spezifische Relationsart charakteristisch sind. Sollte es bei einem unüberwachten Ansatz bleiben, sollten zusätzliche Informationen einer anderen Domäne den Daten angefügt werden.
Innerhalb des gesamten Forschungsbereichs des maschinellen Lernens ergeben sich so verschiedenste Möglichkeiten, sei es mit verschiedenen Features oder Algorithmen, z.B. SVMs mit Kerneln oder Neuronalen Netzwerken. Es könnte für jede Relation eine Klasse (plus ggf. eine Klasse für Wortpaare ohne Relation) definiert werden, denen später unklassifizierte Wortpaare zugeordnet werden.

Die auf dem Korpus erstellten Wortvektoren sind ein sehr aktuelles Forschungsthema. Verschiedene Ansätze wurden präsentiert, diese weiter zu

verbessern, beispielsweise auf Depenzgrammatik basierende Repräsentationen von (Levy und Goldberg, 2014), oder die als *GloVe* bekanntgewordenen globalen Wortvektoren von (Pennington, Socher und Manning, 2014). Es wäre interessant herauszufinden, inwiefern sich diese auf die Performanz eines Systems zur Relationsvorhersage auswirken.

Zudem sollte sich auf eher grober aufgelöste Relationen wie in (Hendrickx et al., 2009) fokussiert werden, da dies im Bezug auf zweierlei Probleme Abhilfe verschaffen dürfte: Weil diese sich auf weniger spezielle Entitäten beziehen, ist gewährleistet, dass die dazugehörigen Wortkontextvektoren qualitativ besser sind. Außerdem ist es in diesem Fall wahrscheinlicher, dass sich ebenjene Relationen als Differenzvektor identifizieren lassen als wesentlich seltenere und spezielle Relationen wie in FB14K.
Das daraus gewonnene Ergebnisse selbst in dieser Auflösungsebene hilfreich sind, wird in der Einleitung von (Hendrickx et al., 2009) dargestellt:

> *"The automatic recognition of semantic relations has many applications, such as information extraction, document summarization, machine translation, or construction of thesauri and semantic networks[,] [...] word sense disambiguation, language modeling, para-phrasing, and recognizing textual entailment."*

Abschließend bleibt festzustellen, dass Wortkontextvektoren nur eine von vielen verschiedenen Arten von kontinuierlichen Wortvektorrepräsentationen sind, die in ihren Eigenschaft zwar beim Lösen vieler Probleme der maschinellen Sprachverarbeiten neue Ansätze geliefert haben, aber auch nicht für jede Problemstellung als Werkzeug nützlich erscheinen.
Stattdessen sollten im Vornherein immer genau die Anforderungen und die Vor- und Nachteile verschiedenster Repräsentationen gegeneinander abgewogen werden.

Der Wert dieser Arbeit besteht unter Anderem demnach darin, Grenzen der Wortkontextvektoren aufgezeigt zu haben, die sich abseits des berühmten Beispiels

$$\vec{v}(King) - \vec{v}(Man) + \vec{v}(Woman) \approx \vec{v}(Queen)$$

in der Begeisterung über ebendiese Eigenschaft nur schwer erkennen lassen.

Literatur

Abramowitz, Milton und Irene A Stegun (1964). *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*. Bd. 55. Courier Corporation.

Arlia, Domenica und Massimo Coppola (2001). „Experiments in parallel clustering with DBSCAN“. In: *European Conference on Parallel Processing*. Springer, S. 326–331.

Baroni, Marco, Georgiana Dinu und Germán Kruszewski (2014). „Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors.“ In: *ACL (1)*, S. 238–247.

Bengio, Yoshua et al. (2006). „Neural probabilistic language models“. In: *Innovations in Machine Learning*. Springer, S. 137–186.

Bordes, Antoine et al. (2011). „Learning structured embeddings of knowledge bases“. In: *Conference on Artificial Intelligence*. EPFL-CONF-192344.

Bordes, Antoine et al. (2013). „Translating embeddings for modeling multi-relational data“. In: *Advances in Neural Information Processing Systems*, S. 2787–2795.

Collobert, Ronan et al. (2011). „Natural language processing (almost) from scratch“. In: *The Journal of Machine Learning Research* 12, S. 2493–2537.

Craven, Mark et al. (2000). „Learning to construct knowledge bases from the World Wide Web“. In: *Artificial intelligence* 118.1, S. 69–113.

Ester, Martin et al. (1996). „A density-based algorithm for discovering clusters in large spatial databases with noise.“ In: *Kdd*. Bd. 96. 34, S. 226–231.

Finkelstein, Lev et al. (2001). „Placing search in context: The concept revisited“. In: *Proceedings of the 10th international conference on World Wide Web*. ACM, S. 406–414.

Fu, Ruiji et al. (2014). „Learning Semantic Hierarchies via Word Embeddings.“ In: *ACL (1)*, S. 1199–1209.

Giaretta, Pierdaniele und N Guarino (1995). „Ontologies and knowledge bases towards a terminological clarification“. In: *Towards very large knowledge bases: knowledge building & knowledge sharing* 25, S. 32.

Goldberg, Yoav (2015). „A Primer on Neural Network Models for Natural Language Processing“. In: *CoRR* abs/1510.00726. URL: http://arxiv.org/abs/1510.00726.

Goldberg, Yoav und Omer Levy (2014). „word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method“. In: *arXiv preprint arXiv:1402.3722*.

Harris, Zellig S (1954). „Distributional structure“. In: *Word* 10.2-3, S. 146–162.

Hendrickx, Iris et al. (2009). „Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals“. In: *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*. Association for Computational Linguistics, S. 94–99.

Köper, Maximilian, Christian Scheible und Sabine Schulte im Walde (2015). „Multilingual reliability and “semantic” structure of continuous word spaces“. In: *IWCS 2015*, S. 40.

Levy, Omer und Yoav Goldberg (2014). „Dependency-Based Word Embeddings.“ In: *ACL (2)*, S. 302–308.

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

Levy, Omer, Yoav Goldberg und Israel Ramat-Gan (2014). „Linguistic Regularities in Sparse and Explicit Word Representations.“ In: *CoNLL*, S. 171–180.

Lin, Yankai et al. (2015). „Learning Entity and Relation Embeddings for Knowledge Graph Completion.“ In: *AAAI*, S. 2181–2187.

Maas, Andrew L, Awni Y Hannun und Andrew Y Ng (2013). „Rectifier nonlinearities improve neural network acoustic models“. In: *Proc. ICML*. Bd. 30. 1.

Mikolov, Tomas et al. (2013a). „Distributed representations of words and phrases and their compositionality“. In: *Advances in neural information processing systems*, S. 3111–3119.

Mikolov, Tomas et al. (2013b). „Efficient estimation of word representations in vector space“. In: *arXiv preprint arXiv:1301.3781*.

Pennington, Jeffrey, Richard Socher und Christopher D Manning (2014). „Glove: Global Vectors for Word Representation.“ In: *EMNLP*. Bd. 14, S. 1532–43.

Rong, Xin (2014). „word2vec parameter learning explained“. In: *arXiv preprint arXiv:1411.2738*.

Rosenblatt, Frank (1958). „The perceptron: a probabilistic model for information storage and organization in the brain.“ In: *Psychological review* 65.6, S. 386.

Rubenstein, Herbert und John B Goodenough (1965). „Contextual correlates of synonymy“. In: *Communications of the ACM* 8.10, S. 627–633.

Schäfer, Roland und Felix Bildhauer (2012). „Building Large Corpora from the Web Using a New Efficient Tool Chain.“ In: *LREC*, S. 486–493.

Wang, Zhen et al. (2014). „Knowledge Graph Embedding by Translating on Hyperplanes.“ In: *AAAI*. Citeseer, S. 1112–1119.

Zou, Will Y et al. (2013). „Bilingual Word Embeddings for Phrase-Based Machine Translation.“ In: *EMNLP*, S. 1393–1398.

Anhang A

Übersicht über Trainingsparameter der Wortkontextvektoren

NAME	KORPUS	PREP	TRAINING	NEG	SAMPLING
01DERS5-5	Decow	-	Skip-gram	5	1^{-5}
02DERS5-4	Decow	-	Skip-gram	5	1^{-4}
03DERS5-3	Decow	-	Skip-gram	5	1^{-3}
04DERS5-2	Decow	-	Skip-gram	5	$0,01$
05DERS5-1	Decow	-	Skip-gram	5	$0,1$
06DERS5-0	Decow	-	Skip-gram	5	1
07DERC5-5	Decow	-	CBOW	5	1^{-5}
08DERC5-4	Decow	-	CBOW	5	1^{-4}
09DERC5-3	Decow	-	CBOW	5	1^{-3}
10DERC5-2	Decow	-	CBOW	5	$0,01$
11DERC5-1	Decow	-	CBOW	5	$0,1$
12DERC5-0	Decow	-	CBOW	5	1
13DELS5-5	Decow	Lemmatisiert	Skip-gram	5	1^{-5}
14DELS5-4	Decow	Lemmatisiert	Skip-gram	5	1^{-4}
15DELS5-3	Decow	Lemmatisiert	Skip-gram	5	1^{-3}
16DELS5-2	Decow	Lemmatisiert	Skip-gram	5	$0,01$
17DELS5-1	Decow	Lemmatisiert	Skip-gram	5	$0,1$
18DELS5-0	Decow	Lemmatisiert	Skip-gram	5	1
19DELC5-5	Decow	Lemmatisiert	CBOW	5	1^{-5}
20DELC5-4	Decow	Lemmatisiert	CBOW	5	1^{-4}
21DELC5-3	Decow	Lemmatisiert	CBOW	5	1^{-3}
22DELC5-2	Decow	Lemmatisiert	CBOW	5	$0,01$
23DELC5-1	Decow	Lemmatisiert	CBOW	5	$0,1$
24DELC5-0	Decow	Lemmatisiert	CBOW	5	1

Tabelle A.1: Quelle und Trainingsparameter für verschiedenen Sets von Wortkontextvektoren. PREP = Aufbereitung des Korpus vor dem Training; TRAINING = Verwendete Trainingsmethode; NEG = Anzahl der Negativbeispiele beim Training; SAMPLING = Ausmaß des Downsamplings häufiger Wörter.

Anhang B

Evaluation der Wortkontextvektoren

Evaluationsergebnisse bei Wortähnlichkeit und Analogien für die verschiedenen Datensets. Wert bei Wortähnlichkeit entspricht der Korrelation zwischen der Kosinusähnlichkeit (4.1) zweier Wörter und einem von Menschen vergebenen Ähnlichkeitsweit mit Spearman's ρ.
Wert bei Analogien entspricht der Anzahl der richtig vervollständigten Wortreihen (für genauere Informationen siehe Kapitel 4.2).
Für weitere Informationen über die Grundlage der Vektoren siehe Appendix A. Wörter außerhalb des Vokabulars (*OOVs*) wurden entweder als Fehler gerechnet, oder werden, falls anders nicht möglich, als Zahl im Index in runden Klammern angegeben[27].

[27] Bei zwei gleichbewerteten Datensets ist deshalb auch das mit weniger OOVs als besser anzusehen, da es über ein größeres Vokabular verfügt.

	Wortähnlichkeit ($\rho \in [-\mathbf{1}, \mathbf{1}]$)				Analogien (in %)	
NAME	WORTPAARE65	WORTPAARE222	WORTPAARE350	SCHM280	GOOGLE	SEMREL
01DERS5-5	**-0,8096**	**-0,4640**$_{(13)}$	**-0,7302**$_{(13)}$	**-0,7094**$_{(2)}$	**44,56**	1,71
02DERS5-4	-0,7856	-0.4409$_{(13)}$	-0,7156$_{(13)}$	-0,6928$_{(2)}$	40,37	1,83
03DERS5-3	-0,7718	-0,4242$_{(13)}$	-0,6886$_{(13)}$	-0,6778$_{(2)}$	27,42	1.87
04DERS5-2	-0,7681	-0,3902$_{(13)}$	-0,6834$_{(13)}$	-0,6545$_{(2)}$	25,48	1,83
05DERS5-1	-0,7725	-0,3889$_{(13)}$	-0,6738$_{(13)}$	-0,6529$_{(2)}$	25,54	1,67
06DERS5-0	-0,7697	-0,3907$_{(13)}$	-0,6784$_{(13)}$	-0,6541$_{(2)}$	25,52	1,46
07DERC5-5	-0,7255	-0,4198$_{(13)}$	-0,6924$_{(13)}$	-0,6361$_{(2)}$	30,60	1,50
08DERC5-4	-0,6149	-0,4321$_{(13)}$	-0,6530$_{(13)}$	-0,5983$_{(2)}$	26,98	1,83
09DERC5-3	-0,6784	-0,4288$_{(13)}$	-0,6090$_{(13)}$	-0,5553$_{(2)}$	22,26	1,62
10DERC5-2	-0,5949	-0,4185$_{(13)}$	-0,5791$_{(13)}$	-0,5003$_{(2)}$	19,22	1,22
11DERC5-1	-0,5890	-0.4245$_{(13)}$	-0,5700$_{(13)}$	-0,4993$_{(2)}$	18,60	1,38
12DERC5-0	-0,5874	-0.4251$_{(13)}$	-0,5706$_{(13)}$	-0,5021$_{(2)}$	8,62	1,50
13DELS5-5	**-0,8247**$_{(1)}$	**-0,5066**$_{(102)}$	**-0,7494**$_{(132)}$	-0,7097$_{(48)}$	15,51	**3,01**
14DELS5-4	-0,8106$_{(1)}$	-0,4953$_{(102)}$	-0,7362$_{(132)}$	**-0,7205**$_{(48)}$	14,51	2,56
15DELS5-3	-0,7786$_{(1)}$	-0,4991$_{(102)}$	-0,7136$_{(132)}$	-0,6956$_{(48)}$	13,15	2,44
16DELS5-2	-0,7576$_{(1)}$	-0,4991$_{(102)}$	-0,6990$_{(132)}$	-0,6702$_{(48)}$	11,76	2,56
17DELS5-1	-0,7578$_{(1)}$	-0,5074$_{(102)}$	-0,6986$_{(132)}$	-0,6638$_{(48)}$	11,41	2,80
18DELS5-0	-0,7551$_{(1)}$	-0,5102$_{(102)}$	-0,6981$_{(132)}$	-0,6715$_{(48)}$	11,38	**3,01**
19DELC5-5	-0,7589$_{(1)}$	-0,4899$_{(102)}$	-0,7476$_{(132)}$	-0,6849$_{(48)}$	14,41	2,23
20DELC5-4	-0,7519$_{(1)}$	-0,5144$_{(102)}$	-0,7239$_{(132)}$	-0,6731$_{(48)}$	12,82	2,15
21DELC5-3	-0,7188$_{(1)}$	-0,5371$_{(102)}$	-0,6875$_{(132)}$	-0,6520$_{(48)}$	9,68	1,75
22DELC5-2	-0,6926$_{(1)}$	**-0,5377**$_{(102)}$	-0,6429$_{(132)}$	-0,6138$_{(48)}$	6,85	1,71
23DELC5-1	-0,6851$_{(1)}$	-0,5373$_{(102)}$	-0,6359$_{(132)}$	-0,5882$_{(48)}$	6,11	1,95
24DELC5-0	-0,6913$_{(1)}$	-0,5349$_{(102)}$	-0,6369$_{(132)}$	-0,5930$_{(48)}$	6,15	2,07

Tabelle B.1: Evaluationsergebnisse der Wortkontextvektoren. Vollzogen mithilfe von Wortähnlichkeits- und Analogiedatensets. Werte beschreiben entweder die Größe des Korrelationswertes $\rho \in [-1, 1]$ oder einen Prozentwert. Fettgedruckte Zahlen bezeichnen Bestwerte.

„I think of "real" failure as the point at which you know [that] what you're working on is the wrong thing to be working on or you are working on it in the wrong way. You can't call the work up to the moment where you've figured out that you are doing the wrong thing failing, that's called learning."

ASTRO TELLER, SUPERVISOR OF GOOGLE'S X DEPARTMENT