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

24. Juli 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 sog. *Wissensdatenbanken* wird Wissen über die Beziehungen von Entitäten in dieser Welt beschrieben. Dieses aufbereitete Wissen ist sehr nützlich, um das fehlende Weltwissen bei Computersystemen zu kompensieren. Die Erstellung solcher Datenbanken ist jedoch sehr arbeitsintensiv, weshalb sich Bemühungen, diese automatisch zu ergänzen, ein lohnendes Forschungsfeld darstellen.
Diese Abschlussarbeit wirft einen Blick auf verschiedene Ansätze in diesem Bereich, die mit Vektorrepräsentationen arbeiten, und versucht darüber hinaus, diese auf die in der NLP-Gemeinschaft sehr beliebt gewordenen Wortkontextvektoren (*word embeddings*) zu übertragen.

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 the relations between entities in this world. This preprocessed knowledge is very useful 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 concering various kinds of vector representations and furthermore wants to realize similar procedures based on word embeddings, which gained a lot of popularity in recent years among the NLP community.

Danksagung

Work in progress. . .

Inhaltsverzeichnis

Abbildungsverzeichnis

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 vielerlei 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 auch beim Spiel Go gegen ein "intelligentes" System [3]. In anderen Bereichen 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] https://de.wikipedia.org/wiki/Weltwissen (zuletzt abgerufen am 03.03.16)

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

[3] Siehe 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 generalis 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 Jeopardy-Champion 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, welches letztere sich im Laufe ihres Lebens aneignen.

Dieses bieten Informationen darüber,

- wie Objekte in der Realwelt zueinander in Beziehung stehen
- welche Attribute von verschiedenen Entitäten besessen werden
- wie Ereignisse in der Welt in Verbindung zueinander stehen
- sich Pläne zu schmieden und Strategien zurechtzulegen
- ...

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

[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. gut 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.

1.2 Ansatz

Innerhalb der letzten Jahre haben neurale Netze in der Informatik im Allgemeinen und in der Computerlinguistik im Speziellen eine Renaissance erlebt. Mit diesen konnten eine neue Art von Wortvektoren, nämlich *Wortkontextvektoren*[8] (engl. "word embeddings") erzeugt werden, die semantische Information implizit in sich kodieren. Zwar zeigen einige Untersuchungen, dass sich 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 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 Hypthese, dass das Trainieren 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 klnnen.

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.

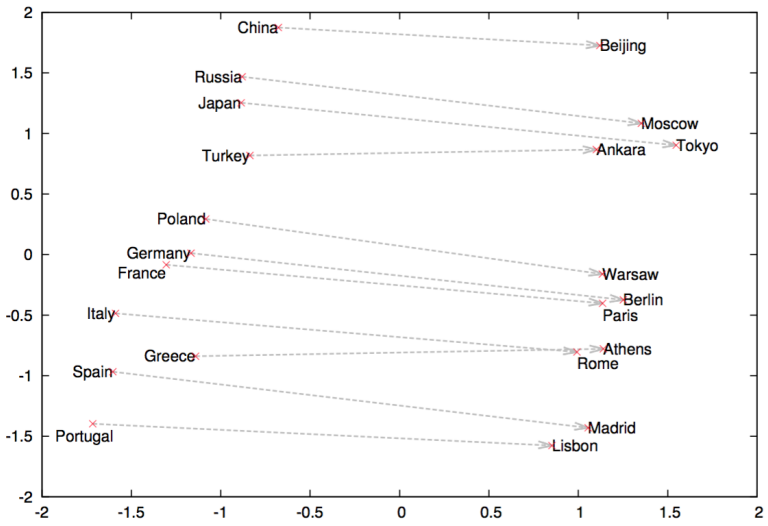


Abbildung 1.1: Hauptstadtrelationen zwischen Wortvektoren aus (Mikolov et al., 2013a). Die 1000-dimensionalen Vektoren wurden mithilfe von PCA auf zwei Dimension projiziert.

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. neurale Netze und ihre Nutzung zum Erstellen von Wortkontextvektoren.
Die in Kapitel 4 beschriebenen Vorbereitungsschritte und eine ausführliche Evaluation von Wortkontextvektoren 5 leiten auf die drei Experimente in den Kapiteln 6, 7 und 8 über:

- $\mathcal{A}$: Reproduktion des Ansatzes von (Bordes et al., 2013) mit einem kleineren Datenset
- $\mathcal{B}$: Indentifikation von Wortpaaren gleicher Relation durch Clustering
- $\mathcal{C}$: Training von Relationsrepräsentationen ähnlich (Bordes et al., 2013) mit Wortkontextvektoren

Zuletzt folgt ein Gesamtfazit der Arbeit mit Ausblick (siehe Kapitel 9) mit zwei Appendices A und B mit detaillierten Informationen über Wortkontextvektortraining und -evaluation sowie die Bibliographie.

Kapitel 2

Verwandte Arbeiten

2.1 Wortvektorrepräsentationen

Wortkontextvektoren wurden u. A. von (Bengio et al., 2006) präsentiert. Im Vergleich zu vorherigen Arbeiten mit sog. “One-Hot”-Vektoren, bei denen jede Dimension einem Wort des Vokabulars eines Korpus zugeordnet ist und nur eine davon den Wert 1 besitzt (gewöhnlich besitzen alle anderen den Wert 0), sind Informationen über ein Wort in dieser Repräsentation für den Menschen uneindeutig und kontinuerlich über den Vektor verteilt (“distributed represenation”), woher sich auch der Name des dazugehörigen Forschungsfeldes - der distributionellen Semantik - herleitet. Dies ist aber eigentlich nur 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 hypthesis*, die besagt, dass Worte, die in ähnlichen Kontexten auftauchen, dazu tendieren, eine ähnliche Bedeutung zu haben (Harris, 1954). Diese Besonderheit macht sich (Bengio et al., 2006) zunutze, da das neurale Netz, mit denen die Wortkontextvektoren erzeugt werden, sich 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 konnte, die Leistungen von Systemen selbst im Bezug auf lange bekannte und erforschte Probleme der Computerlinguistik signifikant verbessern konnte. 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 ein Fokus auf die arithmetischen Regelmäßigkeiten von Wortvektoradditionen eingegangen (siehe

Abbildung 1.1 im vorherigen Kapitel). Diese Eigenschaft soll später in Kapitel 8 erneut aufgegriffen werden.

Weitere Untersuchungen beschäftigen sich u. A. mit bilingualen Repräsentationen für maschinelle Übersetzung (Zou et al., 2013), deren Training auf der Basis von einem dependenzgeparsten Korpus (Levy und Goldberg, 2014), 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 diskutieren schon (Giaretta und Guarino, 1995) verschiedene Auslegungen des Ontologiebegriffs im Kontext ebendieser Wissensbasen. Dementsprechend wurden auch im Zuge des aufkommenden *world wide web* versucht, dieses Ressource zu nutzen, wie z.B. (Craven et al., 2000) zeigt.
Dieses Thema nach dem Vorbild von Wortkontextvektoren zu bearbeiten ist jedoch ein 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 neuralen Netzwerken trainiert. Darauf aufbauend präsentieren (Bordes et al., 2013) einen etwas simpleren Ansatz namens *TransE*, der darauf abzieht, Relationen zwischen Entitäten als eine einfach vektorarithmetische Operation ($\rightsquigarrow$ Übersetzung) zu sehen. Diese Vorgehensweise wird in Kapitel 6 etwas genauer ausgeführt und dere 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 der Unterschiedlichkeit im Kontext anderer Entitäten gerecht zu werden. Eine Übersicht über die Berwertungsfunktionen aller Verfahren findet sich in Abbildung 2.1.

MODELL	SCORING-FUNKTION
TransE	$f_r(h,t) = \parallel h + r - t \parallel_2^2$
TransH	$f_r(h,t) = \parallel h_\perp + r - t_\perp \parallel_2^2$
TransR	$f_r(h,t) = \parallel h_r + r - t_r \parallel_2^2$ $h_r = hM_r$ $t_r = tM_r$
CTransR	$f_r(h,t) = \parallel h_{r,c} + r_c - t_{r,c} \parallel_2^2 + \alpha \parallel r_c - r \parallel_2^2$ $h_{r,c} = hM_r$ $t_{r,c} = tM_r$

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

Abbildung 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 Neurale Netzwerke

Wie der Name bereits erahnen lässt, fußt die Idee von neuralen 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):

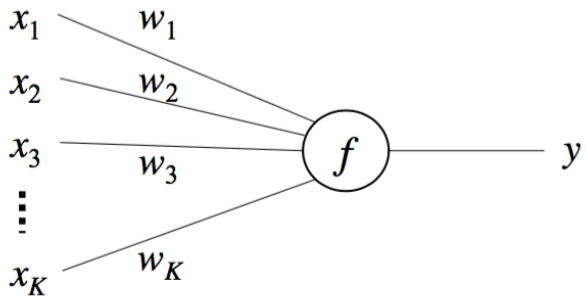


Abbildung 3.1: Mathematische Modellierung eines Neurons, auch bekannt als *Perceptron*-Algorithmus von (Rosenblatt, 1958). x bezeichnet In-, y den Output des Neurons, w bilden die Gewichte und f die Aktivierungsfunktion. Abbildung aus (Rong, 2014).

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, dem 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 neurales 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)[9].

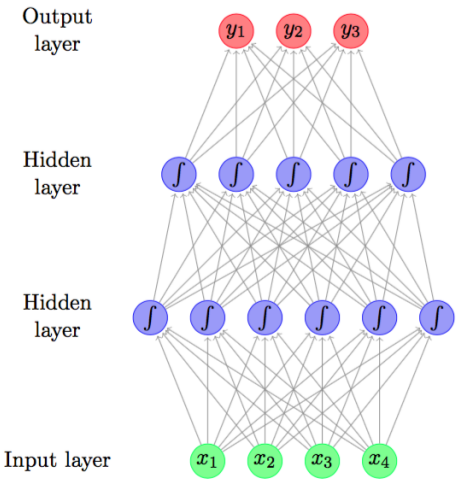


Abbildung 3.2: Darstellung eines neuralen Netzwerks mit vier Schichten, davon jeweils eine für In- und Output, sowie zwei "versteckte" Schichten (*hidden layer*). Bild aus (Goldberg, 2015).

Die einzelnen Gewichtsvektoren w aller Neuronen einer Schicht werden dann i.d.R. zu einer Gewichtsmatrix W zusammengefasst, wobei jede Zeile einem Gewichtsvektor entspricht. Zum Training dieser Strukturen wird

[9] 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.

der sog. *Backpropagation*-Algorithmus verwendet, der mithilfe einer vorher definierten Verlustfunktion (die anzeigt, inwiefern das durch das Netzwerk erzeugt Ergebnis von dem erwünschtne Ergebnis abweicht) einen Fehler errechnet. Dieser wird rekursiv durch alle Schichten des Netzwerks zurückgegeben und simultan die Werte innerhalb der Gewichtsmatrix anpasst.

3.2 Wortvektoren

Frühere Experimente mit Wortvektoren arbeiteten meist mit sog. "One-Hot"-Vektoren, bei denen jede Dimension i.d.R. einem betimmten Wort zugeordnet wurde. Nehmen wir beispielsweise das Minikorpus "Der Hund beißt den Mann" an. Das Vokabular besteht dann aus V $= \{beit,\ Der,\ den,\ Hund,\ Mann\}$ (in alphabetischer Reihenfolge). Um jedes dieser Wörter als einen der oben genannten Vektoren zu repräsentieren, können wir Vektoren der Länge V[10] benutzen. Um nun zum Beispiel einen Vektor für *Hund* zu generieren, setzen wir den Wert der Stelle des Vektors (= *Feature*) auf 1, der 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*[11], die mithilfe von Neuralen Netzwerken trainiert werden, beinhalten mehr (semantische) Informationen über das dazugehörige Wort, zudem lassen sich einzelne Features nicht mehr auf eindeutig auf bestimmte Worte zurückführen[12]. Ein populäres Werkzeug zu diesem Zweck ist `word2vec` von (Mikolov et al., 2013b).

Das Training dieser neuen Wortkontextvektoren läuft folgendermaßen 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'. CBOW versucht, die Wahrscheinlichkeit eines Wortes gegeben eines Kontextes der Größe C[13] zu maximieren. Unsere Verlustfunktion, deren Wert es dabei zu minimieren gilt, besteht darin in der negativen logarithmischen Wahrscheinlichkeit eines Wortes gegeben seines Kontextes:

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

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

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

[10] V steht eigentlich für ‖V‖, wird der Übersicht halber aber im Folgenden stellvertrentend dafür verwendet.
[11] Im Englischen zur Abgrenzung *word embeddings* genannt.
[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.

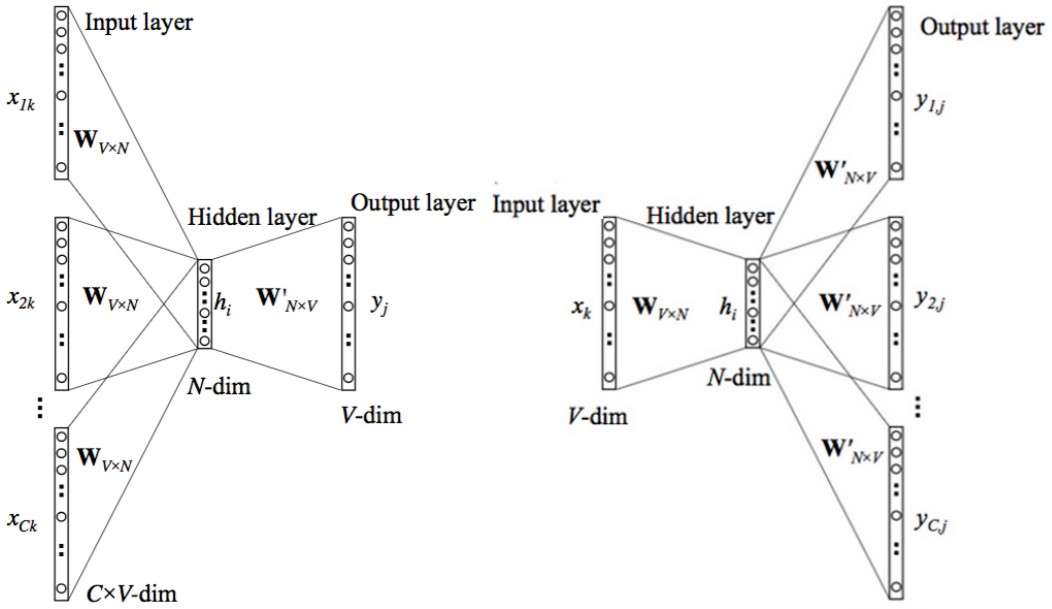


Abbildung 3.3: Gegenüberstellung der beiden Trainingsmethoden CBOW (links) und Skip-Gram (rechts). CBOW versucht die Wahrscheinlichkeit eines Wortes gegeben seines Kontexts zu trainieren, Skip-Gram die Wahrscheinlichkeit eines Kontextes gegeben eines Wortes. Modifizierte Abbildung nach (Rong, 2014).

In beiden Fällen wird daraufhin überprüft, ob die Vorhersage mit den tatsächlichen Daten übereinstimmt und die Abweichung errechnet, mit der dann die Parameter der Gewichtsmatrizen W und W' rekursiv angepasst werden, um zukünftige Prognosen zu verbessen, wofür der *Backpropagation*-Algorithmus verwendet wird. Die Wortkontextvektoren, die dann nach 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 computationelle 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 "schlechte" (also Negativ-)Beispiele zum Training hinzugezogen, woher sich auch der Name des Verfahrens ableitet.

3.3 Clustering

Clustering bezeichnet eine Art von unüberwachten 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

Clustering 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 direkt erreichbar sind, sind Ausreißer (*Outlier*).
- Zwei Punkte p und q sind *direkt verbunden*, sofern es einen Punkt o gibt, von dem aus beide Punkte direkt erreichbar sind.
- Cluster bestehen aus Punkten, die alle miteinander direkt verbunden sind.

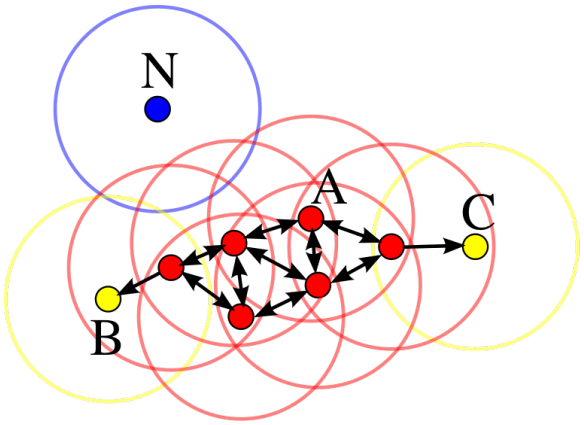


Abbildung 3.4: Darstellung der Funktionsweise von DBSCAN. Punkt A ist ein Kernpunkt, die Punkte B und C sind von A aus erreichbar. Punkt N ist nicht erreichbar und damit ein Ausreißer.[15]

[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).

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

Versuche, den Algorithmus zu parallelisieren und somit eine bessere Skalierbarkeit zu ermöglichen wurden beispielsweise von (Arlia und Coppola, 2001) unternommen. Solche Vorgehen sind wichtig, um die Skalierbarkeit des Algorithmus auf große Datenmengen wie beispielsweise in Kapitel 7 zu sichern.

[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)

Kapitel 4

Vorbereitung

4.1 Vorbereitung

Für die nachfolgenden Experimente in den Kapiteln 5, 6, 7 und 8 mussten einige Daten zuerst beschafft und unter Umständen aufbereitet werden. Die ergriffenen Maßnahmen werden deshalb in diesem Kapitel beschrieben.

4.1.1 Verwendete Daten

Die anfangs verwendeten Daten lassen sich grob in vier Kategorien einteilen: Den Korpus, das Relationsdatenset, Wortähnlichkeitslisten sowie Analogiedatensets (siehe Abbildung 4.1).
Als Korpus wurde der 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	Deutschsprachiger Webkorpus	(Schäfer und Bildhauer, 2012)
Relationsdaten	FB14K	592k Relationstripel aus Freebase	(Bordes et al., 2013)
Wortähnlichkeiten	SCHM280	280	(Köper, Scheible und Walde, 2015)
	WORTPAARE65	65	(Rubenstein und Goodenough, 1965)
	WORTPAARE222	222	
	WORTPAARE350	350	
		nach Ähnlichkeit bewertete Wortpaare	
Analogien	GOOGLE	Datenset mit 18.552 Analogien	(Köper, Scheible und Walde, 2015)
	SEMREL	Datenset mit 2.462 Analogien	

Abbildung 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. Diese werden vor allem in den Versuchen von Kapiteln 6 und 8 relevant.
Alle anderen Datensets sind zur Evaluation der Wortvektoren in Kapitel 5 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.
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

[17] Siehe https://plus.google.com/109936836907132434202/posts/bu3z2wVqcQc (zuletzt abgerufen am 21.07.16).
[18] Eigentlich: GOOGLE SEMANTIC/SYNTACTIC ANALOGY DATASETS

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 7 alle Eigennamen sowie die Satz-IDs für alle Eigennamen gesammelt (siehe die Kookkurrenzeinschränkung 7.2).

4.1.3 Training der Wortvektoren

Wortvektoren werden 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 benötigt es eine Textressource, die einen Satz pro Zeile enthält, Tokens durch Leerzeichen getrennt und gibt die Wortvektoren entweder einem einfach Text- oder Binärformat aus.
Das Tool lässt zudem dem Nutzer offen, einige Parameter zu verändern. Jene, die in dieser Arbeit berücksichtigt wurden, sollen dabei näher erläutert werden:

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

[19] Siehe http://ilk.uvt.nl/conll/ (zuletzt abgerufen am 11.04.16)

- `-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.

Kapitel 5

Evaluation der Wortkontextvektoren

> *"[Y]ou'll see that there are a lot of pairings where words with similar meanings are nearby. [...] On the other hand, there's a lot of junk. [...] You'd want the closest word to 'grandma to be 'grandpa', not 'gym.'"*
>
> BEN SCHMIDT ÜBER DIE EVALUATION VON WORTKONTEXTVEKTOREN[20]

5.1 Evaluationsarten für Wortkontextvektoren

An dieser Stelle sollen die verschiedenen Ansätze zum Evaluieren von Wortkontextvektoren, die im vorherigen Kapitel vorgestellt werden, 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.

5.1.1 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

[20] Blogeintrag "Word Embeddings for the digital humanities" von Ben Schmidt (2015), online unter http://bookworm.benschmidt.org/posts/2015-10-25-Word-Embeddings.html (zuletzt abgerufen am 21.04.15)

Nachbarn[21] 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 5.1).

Nr.	Frankreich	Bank	Computerlinguistik	gekratzt
I	Belgien	Anlagebank	Linguistik	gekrazt
	Italien	(unknown)_Bank	Informationswissenschaft	weggekrazt
	Niederland	BAnk	Texttechnologie	geschubbert
	Niederlande	Bankstatus	Sprachwissenschaft	gekratz
	Spanien	Geldinstitut	Softwaretechnik	abgeleckt
VIII	Italien	Hausbank	Bioinformatik	gekrazt
	Ungarn	Geschäftsbank	Texttechnologie	weggekrazt
	Polen	Grossbank	Informationswissenschaft	geschubbert
	Spanien	Kreditbank	Softwaretechnik	angeknabbert
	Belgien	Mutterbank	Linguistik	rausgerissen
XXII	Italien	Kreditinstitut	Bioinformatik	geritzt
	Ungarn	Geldinstitut	Wirtschaftsgeographie	angesengt
	Polen	Sparkasse	Grundschulpädagogik	zusammengenäht
	Spanien	Landesbank	Computerwissenschaft	aufgesprungen
	Belgien	Finanzinstitut	Humangeographie	tupfen

Abbildung 5.1: 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 Fall eher für den ersten Eindruck eignen, weiterhin aber Prozeduren mit quantifizierbaren Ergebnissen verwendet werden sollten, wie z.B. nächsten Abschnitt 5.1.2 beschrieben werden.

5.1.2 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

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

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):

$$cos(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|\|\vec{b}\|} \tag{5.1}$$

   Danach kann mit *Spearman's* ρ [22]. 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 wird, welches besonders ähnlich zu b und a' aber unähnlich zu a ist:

$$b' = \underset{\tilde{b} \in \mathcal{V}}{argmax}\ cos(\tilde{b}, b - a + a') \tag{5.2}$$

   Sind alle Vektoren der Länge eins, so kann diese Gleichung aufgrund der obigen Definition der Kosinusähnlichkeit (5.1) umformuliert werden:

$$b' = \underset{\tilde{b} \in \mathcal{V}}{argmax}\ cos(\tilde{b}, b) - cos(\tilde{b}, a) + cos(\tilde{b}, a') \tag{5.3}$$

   Diese Methode wird gemeinhin als 3COSADD bezeichnet. (**levy2014linguistic**) etablierten dazu jedoch eine Alternative namens 3COSMUL[23], die bei

[22] 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.

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

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

Tests bessere Ergebnisse produziert:

$$b' = \underset{\tilde{b} \in \mathcal{V}}{argmax} \frac{cos'(\tilde{b}, b)\ cos'(\tilde{b}, a')}{cos'(\tilde{b}, a) + \epsilon} \tag{5.4}$$

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 5.1.3 näher beleuchtet werden.

5.1.3 Evaluationsdaten

Wortpaarähnlichkeit

Im Englischen wird für die Wortähnlichkeitsevaluation häufig das WORDSIM353-Datenset von (**finkelstein2001placing**) 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.[24]

WORTPAARE65, WORTPAARE222 und WORTPAARE350 entstammen der Arbeit von (Rubenstein und Goodenough, 1965). Dabei werden Wortpaaren Werte von 0 ($\hat{=}$ vollkommen unzusammenhängend) bis 4 ($\hat{=}$ stark zusammenhängend) zugeschrieben. Die Anzahl der menschlichen Annotatoren sowie deren Übereinstimmung in Form von $Cohen's\ \kappa$[25] sind in Abbildung 5.2 festgehalten.

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

[24] 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.

[25] 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.

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

Abbildung 5.2: Anzahl der Annotatoren und Agreement (als $Cohen's\ \kappa$) der Wortähnlichkeitsdatensets.

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 Abschnitt5.1.4).

5.1.4 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.

NR.	WORTPAARE65	WORTPAARE222	WORTPAARE350	SCHM280	GOOGLE	SEMREL
I	**-0,8096**	**-0,4640**$_{(13)}$	**-0,7302**$_{(13)}$	**-0,7094**$_{(2)}$	**44,56**	1,71
XIII	**-0,8247**$_{(1)}$	**-0,5066**$_{(102)}$	**-0,7494**$_{(132)}$	-0,7097$_{(48)}$	15,51	**3,01**
XIV	-0,8106$_{(1)}$	-0,4953$_{(102)}$	-0,7362$_{(132)}$	**-0,7205**$_{(48)}$	14,51	2,56

Abbildung 5.3: 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.

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 (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 [26] 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 [27]. Deshalb stellen sich die Ergebnisse bei diesem Vergleichsset ohne klare Tendenz in den Resultaten und generell sehr schlecht dar.

Für die nachfolgenden Schritte wurde die bestabschneidenen Wortkontextvektorsets bei GOOGLE und den anderen Wortpaarsets, namentlich Mark I, XIII und XIV ausgewählt.

[26] 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.

[27] *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*

Kapitel 6

Experiment $\mathcal{A}$: TransE für deutsche Wissensdaten

In diesem Kapitel soll der Ansatz von (Bordes et al., 2013) für deutsche Daten nachvollzogen werden. Dabei wird erst erklärt, wie die Daten erstellt wurden. Danach wird die Idee zum Training der Daten weiter ausgeführt und auf die neuen Datensätze angewendet, bevor schlußendlich eine Evaluation und eine Gegenüberstellung zu den Originalergebnissen erfolgt.

6.1 Datenerzeugung

In (Bordes et al., 2013) werden mehrere Relationsdatensets erstellt, darunter eines namens FB15K. Dieses besteht aus Relationstripeln der Form (h, l, t)[28], wobei h und t Entitäten und l die verbindende Relation bezeichnen. Diese stammen aus *Freebase*, einer von einer Onlinegemeinschaft gepflegten Datenbank, in der mehr als 23 Millionen Entitäten durch Relationen miteinander verknüpft sind. Mittlerweile ist die Seite offline; das Projekt wurde sukzessive in *Wikidata*[29] integriert. Auch die *Freebase API*, die als Programmierschnittstelle zum Abfragen von Informationen dient, wird langsam abgeschaltet [30].

Die FB15K-Daten enthalten 592.213 Tripeln mit 14.951 einzigartigen Entitäten und 1.345 einzigartigen Relationen. In *Freebase* sind Entitäten generell sprachlich unabhängig gehalten. So wird die Entität mit dem Kürzel `/m/02vk52` im Englischen mit dem Begriff *World Bank* und im Deutschen mit *Weltbank* bezeichnet. Somit ist auch FB15K zumindest theoretisch vielsprachig, da es nur die FREEBASE-Kürzel enthält. Jedoch sind die Entitäten darin oft hauptsächlich im englischen bzw. amerikanischen Sprachraum relevante Entitäten, die im deutschen Raum teils nicht sehr bekannt sind. Diese 1:1 ins Deutsche zu übernehmen gestaltet sich schwierig, da nicht jedes Kürzel in *Freebase* mit einer deutschen Bezeichnung aufgeführt ist. Das könnte jeweils drei Gründe haben:

1. Es gibt keine deutsche Übersetzung
2. Die Entität ist für den deutschen Sprachraum nicht relevant genug

[28] $= (head,\ link,\ tail)$

[29] Siehe https://www.wikidata.org/wiki/Wikidata:Main_Page(zuletztabgerufenam20.05.16)

[30] Siehe https://en.wikipedia.org/wiki/Freebase(zuletztabgerufenam(20.05.16))

3. Bisher hat einfach noch kein Nutzer einen deutschen Begriff hinzugefügt

Die Plausibilität dieser Gründe ist diskutabel: Nicht alle Begriffe brauchen eine Übersetzung, so sind beispielsweise Personennamen i.d.R. durch alle Sprachen hinweg gleich. Schwieriger wird es bei Ortsnamen oder Namen von Organisationen: Intuitiv drängt sich der Anschein auf, dass populäre Bezeichnungen eher übersetzt werden als unpopulärere (*United States of America* $\rightarrow$ *Vereinigte Staaten von Amerika* **aber** *University of Denver* $\rightarrow$ *University of Denver*).
Im Falle der Relevanz ist davon auszugehen, dass diese mit der Anzahl durch Nutzer einher geht: Bei einer großen Nutzerbasis ist davon auszugehen, dass diese primär Einträge von Entitäten bearbeiten, die im Kontext der Geschichte, des Tagesgeschehens o. Ä. relevant sind. Gegeben genug Zeit und aktive Nutzer ist also anzunehmen, dass eine immer größer werdende Prozentzahl von relevanten Entitäten an das Deutsche angepasst wird. Bedenkt man die Laufzeit von Freebase seit 2007 (also zum Zeitpunkt des Schreibens dieser Arbeit rund 9 Jahre), so nehmen wir an, dass dieses Bedenken zwar nicht ganz aus der Welt zu räumen, aber zumindest zu vernachlässigen ist.

Basierend auf dieser Argumentation werden korrespondierende "deutsche" Daten folgendermaßen erzeugt: Es wird bei allen Relationstripeln eine Prüfung durchgeführt, ob beide Entitäten h und t eine deutsche Bezeichnung besitzen. Falls nicht, wird dieses Tripel ausgelassen. Danach wird ggf. noch die inverse Relation ergänzt (diese wird später beim Training benötigt), z.B. bei */location/location/contains* und */location/location/containedby*.

DATENSET	#TRIPEL	#ENTITÄTSTYPEN	#RELATIONSTYPEN
FB15k	592.213	14.951	1.345
GER14k	459.724	14.334	1.236

Abbildung 6.1: Daten über die Relationsdatensets FB15k und GER19k. Aufgelistet ist die Anzahl der Tripel (Datensätze), Entitäts- und Relationstypen.

Somit gilt für die Menge englischer Relationstripel S_{EN} und die Menge deutscher Tripel S_{DE}, dass letztere eine echte Teilmenge ist: $S_{EN} \supsetneq S_{DE}$ [31]. Verschiedene Informationen über die beiden Datensets sind in Abbildung 6.1 aufgelistet.

6.2 Training

Gegeben ist eine Menge von Relationstripeln S der Form $(h, l, t) \in S$. Zusätzliche gibt es noch eine Entitätsmenge E sodass $h, t \in E$ und eine Menge von Relationen L mit $l \in L$. Den Entitäten und Relationen werden zuästzlich noch Vektoren zugewiesen, sodass $h, l, t \in \mathbb{R}^k$ gilt. Idealerweise gelten nach dem Training für ein gültige Tripel (h, l, t) dann $h + l \approx t$. Hinzu wird noch eine Menge aus korrumpierten bzw. "falschen" Tripeln S' erstellt, indem für jedes Tripel in S entweder h oder t durch eine andere, zufällig gewählte Entität ersetzt wird:

$$S' = \{(h', l, t) | h' \in E\} \cup \{(h, l, t') | t' \in E\} \tag{6.1}$$

[31] $\forall e \in S_{DE} : e \in S_{EN} \wedge S_{DE} \neq S_{EN}$

Diese Methode wird von (Bordes et al., 2013) rauschkonstrastierendes Lernen (*Noise-constrastive learning*) genannt.

Um die für das Training benötigte Verlustfunktion zu definieren, wird das Unähnlichkeitsmaß d_p für ein Tripel bestimmt, welcher entweder aus der L_1- oder der L_2-Norm abgeleitet wird, also:

$$d_1(h+l,t) = \sum_{i=1}^{k} \|h_i + l_i - t_i\| \tag{6.2}$$

$$d_2(h+l,t) = \sqrt{\sum_{i=1}^{k} \|h_i + l_i - t_i\|^2} \tag{6.3}$$

Zuletzt werden noch Tripel s aus S und zufällige Tripel $s' = (h', l, t')$ aus S' in Pärchen in einem neuen Datenset S^* gruppiert:

$$S^* = \{(s, sample(S'))|s \in S\} \tag{6.4}$$

Nun kann die Verlustfunktion $\mathcal{L}$ erstellt werden:

$$\mathcal{L} = \sum_{((h,l,t),(h',l,t'))\in S^*} max(0, \gamma + d_p(h+l,t) - d_p(h'+l,t')) \tag{6.5}$$

Der Parameter γ bezeichnet hier einen Hyperparameter, der dem System einen gewissen Spielraum lässt. Indem versucht wird, den durch diese Funktion errechneten Verlust zu minimieren, wird beim Training sichergestellt, dass die Vektorrepräsentationen von korrekten Tripeln die Gleichung $h + l \approx t$ bestmöglichst erfüllen und für korrumpierte Tripel möglichst weit verfehlen.
Vor dem Training werden die Repräsentationen der Entitäten und Relationen mit einer uniformen Verteilung im Intervall $[-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}}]$ initialisiert und im Falle von $l \in L$ zusätzlich normiert. Während jedes Trainingsschritt werden für die aktuellen Repräsentation der Verlust berechnet und deren Werte mithilfe des *minibatch stochastic gradient descent*-Algorithmus angepasst. Die Minibatch enthält dabei ebenso viele gültige wie korrumpierte Tripel. Das Training wird solange ausgeführt, bis die Fehlerrate auf dem Validationsset konvergiert[32].

Für das Reproduzieren der Ergebnisse wurde den Empfehlungen von (Bordes et al., 2013) gefolgt: So wird das Training nach maximal 1000 Epochen beendet, außerdem gilt $k = 50$, $\gamma = 1$, $\lambda = 0,01$ und d_1 als Unähnlichkeitsmaß.

6.3 Evaluation

Um die erstellten Daten zu evaluieren, werden bei den Relationen im Testset jeweils die Kopf- und Fußentitäten entfernt. Danach werden alle bekannten Entitäten aus E eingesetzt und mithilfe des Unähnlichkeitsmaßes ansteigend sortiert. Der Rang der korrekten (ursprünglich entfernten) Entität wird dann

[32] Konvergenz auf dem Trainingsset könnte ein Zeichen für Overfitting sein.

gespeichert. Dieses Verfahren wurde zuerst von (Bordes et al., 2011) vorgeschlagen.
Daraus lassen sich zwei Evaluationsmaße ableiten: Den gemittelten Rang der korrekten Entität sowie den Anteil der Evaluationsläufe, bei denen sich die richtige Entität innerhalb der Top 10 befand (Hits@10). Zusätzlich wurde noch zwischen roh und gefiltert unterschieden: Bei letzterem werden mögliche Tripel ignoriert, die in dieser Form schon in den Trainingsdaten vorkamen und so auf "unfaire" Art und Weise vor allen anderen Tripeln bevorzugt werden. Die Ergebnisse für die auf FB15k und GER14k trainierten Daten sind in Abbildung 6.2 festgehalten.

Datenset	Gemittelter Rang		Hits@10	
	Roh	*Gefiltert*	*Roh*	*Gefiltert*
FB15k	575,54	**197,02**	**34,8**	**48,79**
GER14k	**234,73**	222,05	**34,07**	41,83

Abbildung 6.2: Resultate für TransE, trainiert auf dem FB15k- und dem GER14k-Datenset.

An den Ergebnissen ist zuerst zu erkennen, dass sie von den Ergebnissen in (Bordes et al., 2013) für FB15k etwas abweichen. Dies könnte daran liegen, das bei der Beschreibung des Trainingsverfahren angegeben wurde, dass das Prozedere nicht unbedingt 1000 Epochen lang ausgeführt wurde. In dem Code, der von den Autoren online zur Verfügung gestellt und zum Training verwendet wurde [33], wurde dazu leider keine entsprechende Option gefunden, weshalb vermutlich davon auszugehen ist, dass dies manuell ausgeführt oder nicht im Code festgehalten wurde. Daraus ist zu schließen, dass die hier aufgeführten Ergebnisse schlechter sind als die, die im Paper angegeben wurde, weil die Vektorrepräsentationen zu sehr auf die Trainingsdaten angepasst wurden und dann schlecht auf den Testdaten generalisieren. Da die Daten aus GER14k eine Untermenge von FB15k sind, sind deren Ergebnisse i.d.R. gleichwertig oder schlechter. Die Frage, warum der rohe gemittelte Rang bei GER14k signifikant besser ist als bei FB15k, könnte dahingehend beantwortet werden, dass auch hier eine übermäßige Anpassung auf die Trainingsdaten stattfand und zuviele Trainingstripel vor der eigentlich richtigen Lösung rangieren. Deshalb ist der Wert für die gefilterten Tripel umso besser.

6.4 Zwischenfazit

In diesem Kapitel wurde ein Verfahren vorgestellt, dass es erlaubt, Vektorrepräsentationen zu trainieren, die Entitäten und Relationen aus Wissendatenbanken modellieren und zur Relationsvorhersage zu nutzen. Dabei bestehen die Daten aus Relationstripeln, also 3-Tupeln mit einer Kopf- und einer Fußentität und einer dazugehörigen semantischen Relation. Es werden außerdem noch Negativbeispiele erzeugt, indem einer der beiden Entitäten durch eine andere in den Trainingsdaten vorkommende Entität zufällig ausgetauscht wird. Der Trainingsvorgang wird hierbei mithilfe eines auf Vektorarithmetik basierenden Unähnlichkeitsmaß betrieben, mit der die "Stimmigkeit" korrekter Tripel versucht wird zu maximieren. Das Gegenteil

[33] https://github.com/glorotxa/SME

gilt für korrumpierte Tripel.
Es wurde gezeigt, dass sich die (Bordes et al., 2011) präsentierten Ergebnisse im Rahmen der gegebenen Möglichkeiten annähernd replizieren lassen. Darüber hinaus wurde die Methodik auch auf eine Menge deutscher Daten angewendet, die eine echte Teilmenge der englischen Daten darstellt. Für diese konnten ähnlich gute Ergebnisse erziehlt werden, wobei die qua definition kleinere Menge an Übungsbeispielen dem Unterschied in Performanz Rechnung trägt.

Bei dieser Vorgehensweise wurden die Vektorrepräsentationen für Entitäten und Relationen gemeinsam trainiert. In den folgenden Kapiteln soll ein Versuch unternommen werden, mithilfe von distributioneller Semantik erstellte Wortvektorrepräsentationen für die Entitäten zu nutzen und darauf aufbauend für ähnliche Zwecke wie in diesem Kapitel zu nutzen.

Kapitel 7

Experiment $\mathcal{B}$: Relationsvorhersage mit Wortvektorrepräsentationen

Q: Why did the multithreaded chicken cross the road?
A: to To other side . get the

JASON WHITTINGTON

7.1 Idee

Gegeben ist ein Vektorraum V mit Wortkontextvektoren $\vec{u}, \vec{v} \in V = R^d$. Gesucht ist eine Funktion ϕ, die ein Vektorenpaar $(\vec{u}, \vec{v})$ in einen Relationsraum abbildet: $\phi : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^e$, wobei nicht zwangsläufig $d = e$.
In ihrer einfachsten Form bildet sie einfach die Differenz $\vec{d}$ der beiden Vektoren:

$$\phi(\vec{u}, \vec{v}) = \vec{v} - \vec{u} = \vec{d} \tag{7.1}$$

7.2 Algorithmus

Der Grundalgorithmus (siehe Abbildung 7.1) versucht nun, alle Kombinationen von Differenzen der Wortkontextvektoren zu bilden. Alle Kombinationen würde bei n Vektoren in $0(n) = n * n = n^2$ resultieren. Zwar ist $\phi(\vec{u}, \vec{v}) \neq \phi(\vec{v}, \vec{u})$, jedoch wäre die Berechnung beider Differenzvektoren redundant, da sie lediglich Spiegelungen voneinander im Raum sind und so diesselbe Information enthalten. Die Berechnung der Differenz in nur eine Richtung reduziert die Anzahl der Vektoren dadurch zu $O(n) = \frac{n*(n-1)}{2}$ [34].

Data : Menge von Vektorpaaren $\mathcal{C}$
for $(\vec{u}, \vec{v}) \in \mathcal{C}$ **do**
  $\vec{d} = \vec{v} - \vec{u}$;
end

Abbildung 7.1: Einfacher Projektionsalgorithmus.

[34] Gegeben einer Menge relevanter Kombinationen $\mathcal{C}$ mit Vektorpaaren gilt also:

$$\forall(\vec{u}, \vec{v}) \in \mathcal{C} : (\vec{v}, \vec{u}) \notin \mathcal{C}$$

Eine Modifikation des Algorithmus besteht daran, das Berechnen von $\vec{d}$ an eine Bedingung zu knüpfen, da selbst mit der obigen Einschränkung die Ausgabe des Algorithmus gegeben einer Datenmenge immer noch sehr groß ist.
Gegeben sei eine Menge von Sätzen (ein Korpus) $\mathcal{K}$ mit n Sätzen $s_0, \ldots, s_n$ sodass $\mathcal{K} = \{s_i\}_{i=1}^{n}$ mit m Wörtern w_{ij} pro Satz $s_i = \{w_{ij}\}_{j=1}^{m}$. Eine Kookkurrenz von zwei Begriffen (*types*) t_1 und t_2 besteht demnach, falls im Korpus mindestens ein Satz existiert, in dem beide gleichermaßen vorkommen. Dazu können wir eine Funktion $\Lambda(\cdot)$ definieren, die die Anzahl der Kookkurrenzen bestimmt:

$$\Lambda(t_1, t_2) = |\{s_i | \exists w_{ij} = t_1 \wedge \exists w_{ij} = t_2 \wedge t_1 \neq t_2\}_{i=1}^{n}| \tag{7.2}$$

Eine mögliche Einschränkung besteht darin, in einem geänderten Algorithmus (siehe Abbildung 7.2) das Berechnen von $\vec{d}$ nur dann zu erlauben, wenn die Anzahl der Kookkurrenzen der zu den Vektoren $\vec{v}(t_1), \vec{v}(t_2)$ gehörenden Begriffe t_1, t_2 einen bestimmten Schwellenwert γ überschreitet, also $\Lambda(t_1, t_2) \geq \gamma$.

Data : Menge von Vektorpaaren $\mathcal{C}$
for $(\vec{v}(t_1), \vec{v}(t_2)) \in \mathcal{C}$ **do**
  if $\Lambda(t_1, t_2) > \gamma$ **then**
    $\vec{d} = \vec{v}(t_2) - \vec{v}(t_1)$;
  end
end

Abbildung 7.2: Modifizierter Projektionsalgorithmus, bei dem die zu den Vektoren gehörigen Begriffe über γ Mal im Korpus im gleichen Satz aufgetreten sein müssen, damit $\vec{d}$ errechnet wird.

7.3 Parallelisierter Algorithmus

Doch selbst mit den erwähnten Einschränkungen skaliert dieser Algorithmus nur bedingt. Um dem entgegenzuwirken, soll nun versucht werden, diesen mithilfe von *Multithreading* zu parallelisieren. Bei dieser Technik beschäftigt ein Rechenprozess mehrere Prozessstränge (*Threads*), die miteinander kommunizieren und bei Bedarf synchronisiert werden können, während sie Teile eines Problems gleichzeitig lösen.
Ein beliebtes Muster, das Verhältis zwischen mehreren Threads zu definieren, besteht im *Master-Slave*-Muster. Dabei fungiert ein Subprozess als Aufseher, der seine Arbeiterprozesse überwacht, sie mit Daten versorgt und in manchen Fällen untereinander koordiniert, während die anderen Prozesse Teilprobleme lösen und ggf. Rückmeldung an den Aufseherprozess geben.

In diesem konkreten Fall ist der Master-Thread dafür verantwortlich, alle benötigten Daten in entsprechende Datenstrukturen zu laden, sie den Worker-Threads bereitzustellen und letztere gemeinsam zu starten und zu beenden, sobald ein bestimmtes Abschlusskriterium der Aufgabe erfüllt ist.

Data : Menge von Vektorpaaren $\mathcal{C}$
Data : Menge von erledigten Vektorpaaren $\mathcal{C}'$ (Anfangs $\mathcal{C}' = \varnothing$)
Data : Menge von Threads $\mathcal{T} = \{th\}_i^n$

Klasse MASTERTHREAD
data = ladeDaten();
for $th \in \mathcal{T}$ **do**
  starteThread(th, data);
end
for $th \in \mathcal{T}$ **do**
  beendeThread(th);
end

Klasse WORKERTHREAD
for $(\vec{v}(t_1), \vec{v}(t_2)) \in \mathcal{C}$ **do**
  if $(\vec{v}(t_1), \vec{v}(t_2)) \notin \mathcal{C}'$ **then**
    if $\Lambda(t_1, t_2) > \gamma$ **then**
      $\vec{d} = \vec{v}(t_2) - \vec{v}(t_1)$;
    end
    $\mathcal{C}' = \mathcal{C}' \cup (\vec{v}(t_1), \vec{v}(t_2))$;
  end
end

Abbildung 7.3: Parallelisierter Projektionsalgorithmus, bei dem die zu den Vektoren gehörigen Begriffe über γ Mal im Korpus im gleichen Satz aufgetreten sein müssen, damit $\vec{d}$ errechnet wird. Ein Master-Thread verteilt zudem die Aufgaben an Worker-Threads, die diese Berechnungen übernehmen und erledigt Vektorpaare in einer Menge ablegen.

Zusätzlich wird eine Menge eingeführt, in die Paare von Vektoren hinzugefügt werden, sofern für sie von einem Thread ein Differenzvektor ausgerechnet wurde (siehe Abbildung 7.3). Damit nun keiner der Threads redundante Berechnungen durchführt, prüft er, ob sein aktuelles Vektorpaar sich in dieser Menge befindet und schon abgehakt wurde[35].

Als Grundlage der Daten wurde das Datenset Mark I gewählt, da es in den meisten der Evaluationsaufgaben als bestes Abschnitt. Mit $\gamma = 100$ resultierte das Procedere in einem neuen Menge an Daten mit insgesamt X Differenzvektoren [GENAUE ZAHL EINFÜGEN] mit einer Größe von X,X GB [GENAUE GRÖSSE EINFÜGEN].

7.4 Evaluation

Die Evaluation der im vorherigen Schritt gewonnen Daten findet folgendermaßen statt: Für jedes gewonnene Cluster c wird für jedes Paar von Vektoren die Zughehörigkeit der entsprechenden Wörter zu einer FREEBASE-Relation

[35] Damit dies möglichst schnell funktioniert, wird das Vektorpaar durch eine Hashfunktion h in einen Wert umgewandelt und in einem *Hashset* gespeichert.. Diese wird so gewählt, sodass $h(\vec{v}(t_1), \vec{v}(t_2)) = h(\vec{v}(t_2), \vec{v}(t_1))$.

$r \in R$ geprüft. Dem gesamten Cluster kann dann die Relation r_c zugeordnet werden, die die meisten Treffer zu verzeichnet hat[36]:

$$r_c = \underset{r \in R}{argmax} \sum_{(\vec{v}(w_1),\vec{v}(w_2)) \in c} \mathbb{1}_r((w_1, w_2)) \tag{7.3}$$

Die "Reinheit" P eines Clusters kann dann errechnet werden, indem der Anteil der zur Relation des gesamten Clusters zugehörigen Wortpaaren bestimmt wird:

$$P(c) = \sum_{(\vec{v}(w_1),\vec{v}(w_2)) \in c} \mathbb{1}_{r_c}((w_1, w_2)) \ / \ |c| \tag{7.4}$$

Diese Metrik bestraft allerdings, wenn in einem Cluster neue Paare einer Relation vorkommen, die noch nicht in FREEBASE vorhanden sind. Da dies allerdings auch ein Teilziel dieses Ansatzes ist, ebensolche Paare neu aufzufinden, wird noch eine modifizierte Version dieser Metrik behandelt: Es gilt dabei vor allem, unbekannte richtige Wortpaare von falschen zu unterscheiden. Zu diesem Zweck wird die Annahme getroffen, dass alle Vektoren $\vec{v}(h)$ und $\vec{v}(t)$, die zu den Kopfentitäten H_c und den Fußentitäten T_c[37] gehören, recht ähnlich sind[38]. So wird für jedes Cluster zunächst für ebendiese Entitäten ein Durchschnitt errechnet, deren Zugehörigkeit zu r_c festgestellt wurde:

$$\overline{h} = \sum_{\vec{v}(w_1) \in H_c} \vec{v}(w_1) \ / \ |H_c| \tag{7.5}$$

$$\overline{t} = \sum_{\vec{v}(w_2) \in H_c} \vec{v}(w_2) \ / \ |T_c| \tag{7.6}$$

Demnach erweitern wir r_c um die Wortpaare, deren Vektoren den durchschnittlichen Kopf- und Fußvektoren ähnlich genug sind. Zu diesem Zweck wird ein Schwellenwert τ definiert, den ein potenziell zu r_c zugehöriges Wortpaar im Bezug auf Cosinusänlichkeit (siehe 5.1) nicht unterschreiten darf:

$$r_c^+ = r_c \cup \{(w_1, w_2) | (\vec{v}(w_1), \vec{v}(w_2)) \in c \wedge cos(\vec{v}(w_1), \overline{h}) \geq \tau \wedge cos(\vec{v}(w_2), \overline{t}) \geq \tau\} \tag{7.7}$$

Auf Basis der Gleichung 7.4 folgt für die modifizierte Reinheit eines Wortpaarclusters P^+:

$$P^+(c) = \sum_{(\vec{v}(w_1),\vec{v}(w_2)) \in c} \mathbb{1}_{r_c^+}((w_1, w_2)) \ / \ |c| \tag{7.8}$$

[36] Die Indikatorfuntion mit $\mathbb{1}_{r_c}((w_1, w_2))$ liefert 1, wenn $(w_1, w_2) \in r$ gilt, ansonsten 0.

[37] $H_c = \{\vec{v}(w_1) | (w_1, \cdot) \in r_c\}$ und $T_c = \{\vec{v}(w_2) | (\cdot, w_2) \in r_c\}$

[38] In einem Cluster der Relation *Hauptstadt von* würden sich vermutlich die Vektoren aller Hauptstädte wie *Paris*, *Berlin*,... ebenso ähneln wie die der dazugehörigen Länder *Frankreich*, *Deutschland*,...

7.5 Ergebnisse

Vorgreifend soll angemerkt werden, dass es nicht möglich war, endgültige Ergebnisse mithilfe der vorher beschriebenen Evaluationsmetrik $P^{+}(\cdot)$ zu erzeugen. Die genauen Gründe hierfür werden im nächsten Abschnitt 7.6 genauer beleuchtet. Dementsprechend sollen an dieser Stelle lediglich Zwischenergebnisse vorgestellt werden.

Cluster A	**Cluster B**
OLDENBURG - GÖRLITZ	PFEIFFER - EU-LÄNDER
TSCHECHIEN - KARIN	FRANZ - EU-LÄNDER
TSCHECHIEN - UWE	UWE - EU-LÄNDER
HOCKENHEIM - OFFENBURG	KARIN - EU-LÄNDER
OLDENBURG - OFFENBURG	HENNING - EU-LÄNDER
⋮	⋮
Cluster C	**Cluster D**
NORDHAUSEN - OBERLIGA	DIETER - ITALIENISCHEN
BOCHUM - REGIONALLIGA	PIRNA - ITALIENISCHEN
HAMELN - REGIONALLIGA	DAMON - ITALIENISCHEN
CHEMNITZER - REGIONALLIGA	SHANGHAI - ITALIENISCHEN
CHEMNITZER - BUNDESLIGA	PANKOW - ITALIENISCHEN
⋮	⋮

Abbildung 7.4: Auszüge aus einigen Wortpaarclustern, die mit in dem Abschnitt 7.3 vorgestellten Ansatz erzeugt wurden.

Wie Abbildung 7.4 zu entnehmen ist, in der einige Auszüge aus Clustern beispielhaft dargestellt sind, fällt es schwer, die dort aufgeführten Wortpaare insgesamt einer Relation zuzuordnen. Im Gegenteil gelingt dies selbst für einzelne Paare nicht, siehe beispielweise “Uwe” und “EU-Länder”.
Es zeigt sich, dass auch das Kookkurrenz-Feature zu keiner Besserung der Ergebnisse geführt hat. Aufgrund der scheinbaren Zufälligkeit der Ergebnisse erschien es auch nicht mehr sinnvoll, weitere Schritte auf Basis dieser Daten (vor allem Evaluation) durchzuführen.

7.6 Zwischenfazit

Wie der letzte Abschnitt der gezeigt hat, entsprechen die Ergebnisse nicht den vorher gefassten Hoffnungen. Im Folgenden soll ein Versuch unternommen zu erklären, welche Probleme zu diesen Resultaten geführt haben und welche Implikationen diese für generelle Bestrebungen auf diesem Gebiet besitzen.

7.6.1 Daten

Will man das Scheitern unmittelbar auf einen Faktor zurückführen, so liegen die zugrunde liegenden Daten am nächsten. Es lassen sich im Bezug auf jene folgende Punkte feststellen:

- **Menge**
  Selbst mit der Reduzierung der resultierenden Daten von n^2 auf $\frac{n*(n-1)}{2}$ und weiter mit $\Lambda(\cdot)$ ist die Datenmenge noch sehr groß, gerade wenn das anfänglich Wortkontextvektorset auf einem großen Vokabular wie dem des DECOW-Korpus aufbaut. Dies stellt hohe Anforderung an die Skalierbarkeit aller involvierter Algorithmen und fordert Einschränkungen auf verschiedenen Ebene, wodurch mögliche spätere Entdeckung eventuell vorenthalten oder verzerrt werden könnten.

- **Rauschen**
  In den Enddaten ist der überwiegende Teil der Datenpunkte keiner sinnvollen Relation zuzuordnen. Dadurch verrauschen mögliche sinnvolle Relationscluster, die darin untergehen (vgl. Abbildung 7.5). Die resultierenden Cluster sind sehr schwammig, da sie sich schlecht vom Hintergrundrauschen abgrenzen. Dies ist beispielsweise durch den Wert des Silhouettenkoeffizienten[39] erkennbar, der bei den Experimenten immer kurz unter Nulls lag[40]. Das in diesem Rauschen valide Ergebnisse vorliegen müssten, lassen das Clustering vorgefilterter Wortpaare von (Fu et al., 2014) und Abbildung 1.1 zwar erahnen, sie sind jedoch nicht auszumachen.

- **Ähnliche Distanzvektoren**
  Der beschriebene Ansatz wurde unter der Annahme verfolgt, dass für Wortpaare einer Relation $r = \{(h_j, t_j)\}_{j=0}^{m}$ folgendes gilt:

$$\vec{v}(t_0) - \vec{v}(h_0) \approx \vec{v}(t_1) - \vec{v}(h_1) \approx \ldots \approx \vec{v}(t_m) - \vec{v}(h_m) \tag{7.9}$$

  Anders ausgedrückt wurde aufgrund vorheriger Arbeiten die Hypothese aufgestellt, dass sich die Distanzvektoren von Wortpaaren derselben Relation ähneln. Durch die Ergebnisse wurde diese Hypothese nicht widerlegt (im Gegenteil scheinen andere Arbeiten wie (Bordes et al., 2013) oder (Lin et al., 2015) diese Annahmen zu bestätigen), jedoch gilt diese Eigenschaft nicht **exklusiv** für diese Untermenge an Vektorenpaaren.
  Nehmen wir beispielsweise im ursprünglichen Vektorraum mit Wortvektoren das folgende Szenario an: Wir finden dort zwei Cluster c_1 und c_2 vor. Die zu den Vektoren zugehörigen Cluster weisen eine semantische Ähnlichkeit auf und liegen deshalb nahe beieinander, z.B. Vornamen wie *Julia* und *Hans* in c_1 und Städte wie *Barcelona* und *Paris* in c_2. Daraus ergibt sich Folgendes:

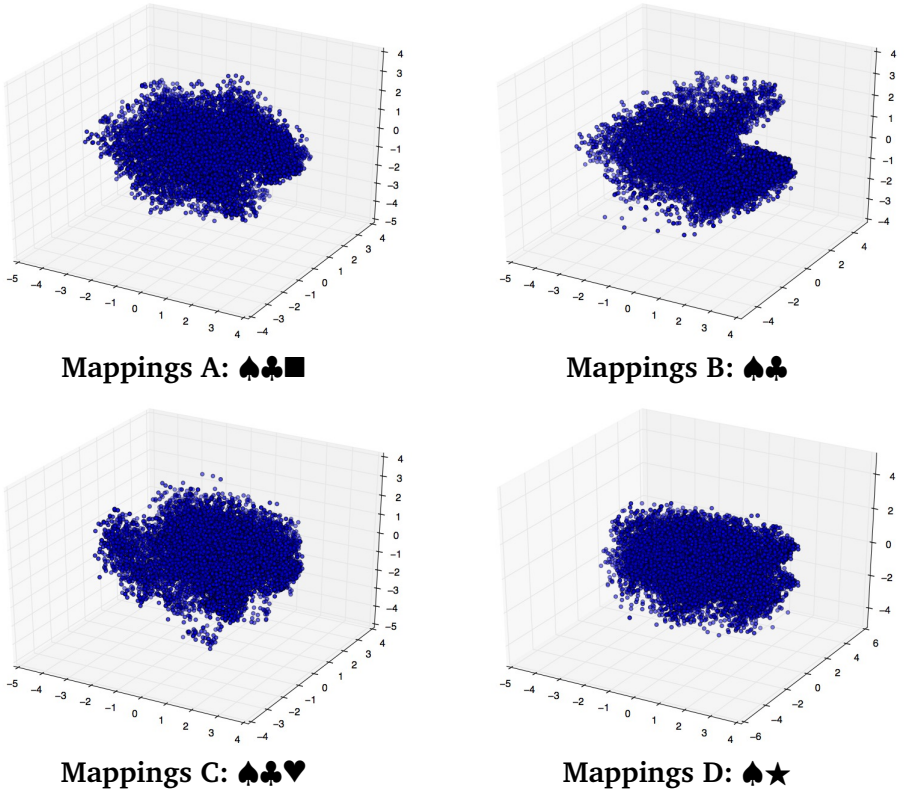
$$\begin{aligned} \vec{v}(Julia) \approx \vec{v}(Hans) \wedge \vec{v}(Barcelona) \approx \vec{v}(Paris) \implies \\ \vec{v}(Julia) - \vec{v}(Barcelona) \approx \vec{v}(Hans) - \vec{v}(Paris) \end{aligned} \tag{7.10}$$

  Anders ausgedrückt: Auch wenn sich semantische Information in den Wortvektoren dadurch manifestiert, dass ähnliche Ausdrücke in ihren

[39] Der Silhoettekoeffizient s_C beschreibt das durchschnittliche Verhältnis vom Abstand eines Punktes zu seinem Clusterzentrum gegnüber des nächsten Clusterzentrums. $-1 \leq s_C \leq 1$.

[40] Für ein gutes Clustering wird ein Wert von $s_C \geq 0,75$ erwartet.

Clustern nahe beinander liegen, können Distanzvektoren aus Wortenpaaren der gleichen Cluster ähnlich sein, ohne semantisch in irgendeinem Zusammenhang zu stehen. Dies führt schlussendlich dazu, dass die wenigen Cluster, die im Relationsraum gefunden werden, zwar aus Wortpaaren mit ähnlichem Differenzvektor stehen, jedoch keine "sinnvolle" Relation bilden, was sicht mit dem in 6.1 eingeführtern Parameter γ zwar verringern, allerdings nicht gänzlich verhindern lässt. Das Ziel, neue und legitime Relationen bzw. Wortpaare zu finden, wird durch diesen Trugschluss ad absurdum geführt.



Mappings A: ♠♣■ **Mappings B: ♠♣**

Mappings C: ♠♣♥ **Mappings D: ♠★**

Feature-Legende

- ♠ $\vec{v}(t_2) - \vec{v}(t_1)$
- ★ $\Lambda(t_1, t_2) \geq 1$
- ♣ $\Lambda(t_1, t_2) \geq 100$
- ■ $cos(\vec{v}(t_2), \vec{v}(t_1))$
- ♥ $\parallel \vec{v}(t_2) - \vec{v}(t_1) \parallel$

Abbildung 7.5: Dreidimensionale Projektionen einiger durch das Mappingverfahren resultierender Vektorräume. Jeweils dargestellt: 10000 Vektoren. Symbole zeigen die jeweils genutzten Features an.

7.6.2 Ansatz

Das Fehlschlagen des Vorgehens kann auch auf einer theoretischen Ebene festgestellt werden: Das Ziel bestand darin, Wissen über Relationen zwischen Wortpaaren zu extrahieren. Um dieses Wissen gewissermaßen "freilegen", muss es aber auf eine Art und Weise innerhalb der Daten "kodiert"

sein, wenn auch implizit (so wie die semantische Ähnlichkeit zwischen Wörtern durch die räumliche Nähe ihrer Vektoren und deren Dimension kodiert ist).
Wie beim Aufzeigen des Trugschlusses am Ende von 7.6.1 gezeigt wurde, sind semantische Relationen nicht eindeutig innerhalb der Daten aufzuzeigen bzw. nur dann, wenn bereits vorher bruchstückhaftes Wissen darüber vorliegt. Ohne dieses Vorwissen sind richtige von nur scheinbaren Relationen nicht zu trennen. Um den Ansatz zukünftig zu einen erfolgreichen Ende zu führen, müsste also überlegt werden, ihn von einem unüberwachten in einen mindestens semi-überwachten Ansatz zu transformieren und weiteres Wissen aus einer anderen Domäne hinzuzufügen.

Kapitel 8

Experiment $\mathcal{C}$: Relationsvorhersage mit Wortkontextvektoren

In diesem Kapitel soll versucht werden, die Methodik des rauschkonstrastierenden Lernen aus Kapitel 6 auf die Wortkontextvektoren (vgl. Kapitel 5) anzuwenden und nur die Relationsvektoren zu lernen. Zu diesem Zweck wird eine Untermenge von FB15K, die Tripel enthält, deren Kopf- und Fußentität jeweils als Wortvektorrepräsentation vorliegen. Schlußendlich werden die Ergebnisse evaluiert und diskutiert.

8.1 Datenerzeugung

Zu Trainingszwecken wird eine Untermenge aus FB15K bzw. GER14K gebildet. Da die Vektorrepräsentationen für die Entitäten nicht von Grund auf erstellt sorden mit `word2vec` trainierte Vektoren verwendert werden, können nur solche Worte in dem neuen Trainingsset vorhanden sein, zu denen ebensolche Wortvektoren auch vorliegen. Dies ist leider nur für einen substantiell geringeren Teil der Fall, wie in Abbildung 8.1 zu erkennen ist.

DATENSET	#TRIPEL		#ENTITÄTSTYPEN		#RELATIONSTYPEN	
FB15k	592.213		14.951		1.345	
GER14k	459.724	(↓-22,37 %)	14.334	(↓-4,12 %)	1.236	(↓-8,1 %)
WE4k	50.518	(↓-91,47 %)	3.623	(↓-74,72 %)	749	(↓-39,40 %)

Abbildung 8.1: Daten des neuen Sets WE4K im Vergleich mit FB15K und GER14K. Aufgelistet ist die Anzahl der Tripel (Datensätze), Entitäts- und Relationstypen und die Veränderung der Anzahlen in Prozent im Vergleich zu FB15K in Klammern.

Dies ist dadurch zu erklären, dass in der Quelle der Originaldaten, nämlich *Freebase*, sehr viele sehr seltene Entitäten vertreten sind. Die vielen Freiwilligen, die über die Jahre dazu beigetragen haben, die Wissensdatenbank weiter zu vervollständigen, haben dabei auch viele unbekanntere Filme, Personen etc. hinzugefügt, die in einem Korpus eher selten zu finden sind, sei er auch so groß wie DECOW.

Da die resultierende Menge an Tripeln gerundet ungefähr viertausend einzigartig Entitäten enthält, die auch als Wortkontextvektoren vorliegen, wird dieses Datenset im Folgenden WE4K genannt.

8.2 Training

Das Training der Relationsvektoren findet in einer sehr zu der in Kapitel 6 beschrieben Methode sehr ähnlichen Art und Weise wie in Kapitel 6 statt: Es wird zuerst eine Menge korrumpierter Tripel erstellt, bei denen entweder die Kopf- oder Fußentität ersetzt wurde. Mithilfe der ursprünglichen Tripel wird eine Menge aus Tripelpaaren erstellt, von denen der korrumpierte Teil jeweils zufällig aus S' ausgewählt wird. Auch die Verlustfunktion gestaltet sich prinzipiell gleich (siehe Formel 6.5).

Die wenigen Unterschiede bestehen darin, dass diesmal nur die Repräsentationen der Relationen trainiert werden, da die der Enitäten und schon vorliegen sowie im Trainingsschritt selbst: So wird wird wegen der geringeren Menge an Daten darauf verzichtet, für jede Trainingsanzahl eine feste Anzahl von Tripelpaaren zufällig auszuwählen. Stattdessen richtet sich die Größe von S_{batch} nach folgender Formel[41]:

$$S_{batch} \leftarrow sample(S^*, \left\lfloor \frac{|S^*|}{10} \right\rfloor + 1) \tag{8.1}$$

8.3 Evaluation

Die Evaluation fand auf diesselbe Art und Weise wie im Abschnitt 6.3 statt. So wurde wieder untersucht, inwiefern die Relationsvektoren und die Vektorrepräsentationen der Entitäten dazu genutzt werden können, ein Relationstripel durch "Raten" zu vervollständigen und gemessen, welcher Rang der richtigen Antwort zugewiesen wurde und in wieivel Prozent diese unter den Top Zehn anzutreffen war.

DATENSET	GEMITTELTER RANG	HITS@10
WE4K	1122,03	5,67
FB15K	**197,02**	**48,79**
GER14K	222,05	41,83

Abbildung 8.2

Im Blick auf Abbildung 8.2 zeigt ein ziemlich ernüchterndes Fazit auf. So schnitt die Variante mit Wortvektoren aus `word2vec` und seperat trainierten Relationsrepräsentationen ($\rightarrow$ WE4K) im Bezug auf den gemittelten Rang mehr als fünf Mal so schlecht und im Bezug auf die HITS@10 mehr als acht Mal so schlecht ab (sic!).
Die möglichen Gründe für das weite Verfehlen guter Ergebnisse werden nun in Kapitel 8.4 diskutiert.

[41] Die Addtion von 1 dient dazu, bei weniger von 10 Tripeln für eine Relation wenigstens mit einem Tripel zu trainieren. Der Rest entspricht einer Heuristik. Es wurden verschiedene Größen für den Minibatch ausprobiert, die jedoch keinen signifikaten Einfluss auf die Ergebnisse hatten.

8.4 Zwischenfazit

Um die Diskrepanz zwischen den Ergebnissen mit gemeinsam trainierten Repräsentationen für Entitäten und Relation im Vergleich zu denen mit Wortkontext- als Entitätsvektoren zu erklären, sollte die Unterschiedlichkeit der Ansätze hervorgehoben werden.
Im ersten Fall werden wie erwähnt alle eine Relation betreffende Repräsentationsvektoren gemeinsam trainiert; es ist also davon auszugehen, dass diese bestmöglichst angepasst sind, um $\parallel h + r - t \parallel$ für alle h, t zu minimieren.
Dies kann im zweiten Fall nicht unbedingt gewährleistet werden: Da h und t im Vornherein schon feststehen und nicht an r angepasst werden (sondern genau umkehrt r an h und t angepasst wird), kann der minimale Wert der Verlustfunktion unter Umständen nur schwer erlangt werden.
Als eine weitere Begründung ist anzuführen, dass a) WE4K auf **deutlich** kleineren Datenmengen arbeitet (weshalb maschinelle Lernverfahren unter Umständen gar keine Lösung finden können, die gut auf ungesehene Beispiele generalisiert) und b) Wortkontextvektoren von sehr speziellen Entitäten enthält, die in *Freebase* als Knoten existieren, u. U. aber sehr selten im Korpus auftauchen. Da die Beschaffenheit der korpusbasierten Repräsentationen vor allem von Kookkurrenzen mit anderen Begriffen abhängt, bedeutet eine niedrige Termfrequenz vor allem weniger aussagekräftige Wortkontextvektoren, worunter die Ergebnisse in der Konsequenz zu leiden haben.
Zuletzt sollte mit in Betracht gezogen werden, dass die trainierten Relationen sehr feingliedrig sind. Anderen Forschungsarbeiten, die mit grobaufgelösten Beziehungen wie “Cause-Effect” oder “Entitiy-Origin” (Hendrickx et al., 2009) arbeiten, können dadurch auf größere Datenmengen zugreifen und bessere Ergebnisse auf ungesehenen Beispielen erzielen.

Kapitel 9

Fazit

“Mit dem Wissen wächst der Zweifel.”

JOHANN WOLFGANG VON GOETHE

9.1 Zusammenfassung

In dieser Arbeit wurden verschiedene Möglichkeiten präsentiert, Wissen aus Wissensdatenbanken durch kontinuierliche Vektorrepräsentationen darzustellen und zu erweitern. In Kapitel 6 wurden unterschiedlich komplexe Herangehesweisen vorgestellt, Repräsentationen für Entitäten und Relationen einer solchen Datenbank gleichzeitig zu 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 7 wurde ein eigener Versuch unternommen, ähnliche Ergebnisse mithilfe von Wortkontextvektoren zu erreichen, zur 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 Trugschluss fußte. Deshalb konnten keine evaluierbaren Ergebnisse erzeugt werden. Eine ausführliche Diskussion darüber findet sich in 7.6.

In Kapitel 8 wurde versucht, Wortkontextvektoren in das Verfahren von 6 einzubinden und nur noch Vektorrepräsentationen für Relationen zu trainieren. Hierfür wurde ebenfalls ein Datenset namens WE4K erzeugt. Wegen der Datenknappheit und anderen in 8.4 diskutierten Gründen konnten hierbei keine vergleichbaren Ergebnisse erlangt werden. Dennoch konnten einige Erkenntnisse über Wortkontextvektoren im Speziellen und Vektorrepräsentationen daraus gezogen werden, die im nächsten Abschnitt reflektiert werden.

9.2 Diskussion

Die Implikationen der in dieser Arbeit gewonnenen Erkenntnisse sollte 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 9.3 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 dafür wichtige Eigenschaften getrimmt werden (nämliche 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 7 scheint vor allem auf einen Fehler in der Grundannahme und die schlechte Skalierbarkeit zurückzuführen sein. 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 arithmentischen Rechnungen mit Wortvektoren so stark hervorgehoben werden. 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 zeitliche 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.

9.3 Ausblick

Der vorliegende Ansatz bietet vielerlei Möglichkeiten zur Verbesserung. Im Folgenden sollen einige davon skizziert werden:
Zuallerst erscheint es sinnvoll, das Konzept von einem unüberwachten zu einem überwachten Ansatz zu ändern, sprich einer maschinellen Lernmethode mit Daten zu versorgen, denen bereits eine Klasse zugewiesen wurde. Beispielweise könnten eine sog. *Support Vector Machine* (SVM) oder ein anderer Lernalgorithmus ein zu einer bestimmten Relation gehöriges Wortpaar als Eingabe nehmen, um anhand dessen 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 dutzende Möglichkeiten, sei es mit verschiedenen Features oder Algorithmen, z.B. SVMs mit Kerneln oder Neuralen 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 durch auf dem Korpus erstellten Wortvektoren sind ein sehr aktuelles Forschungsthema. Verschiedene Ansätze wurden präsentiert, diese weiter zu verbessern, beispielweise auf Depenzgrammatik basierede 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), 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 Auflösungebene 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 kontinuerlichen 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 diese Eigenschaft nur schwer erkennen lassen.

Anhang A

Übersicht über Trainingsparameter der Wortkontextvektoren

NR.	KORPUS	PREP	TRAINING	NEG	SAMPLING
I	Decow	-	Skip-gram	5	$1^{-}5$
II	Decow	-	Skip-gram	5	$1^{-}4$
III	Decow	-	Skip-gram	5	$1^{-}3$
IV	Decow	-	Skip-gram	5	$0,01$
V	Decow	-	Skip-gram	5	$0,1$
VI	Decow	-	Skip-gram	5	1
VII	Decow	-	CBOW	5	$1^{-}5$
VIII	Decow	-	CBOW	5	$1^{-}4$
IX	Decow	-	CBOW	5	$1^{-}3$
X	Decow	-	CBOW	5	$0,01$
XI	Decow	-	CBOW	5	$0,1$
XII	Decow	-	CBOW	5	1
XIII	Decow	Lemmatisiert	Skip-gram	5	$1^{-}5$
XIV	Decow	Lemmatisiert	Skip-gram	5	$1^{-}4$
XV	Decow	Lemmatisiert	Skip-gram	5	$1^{-}3$
XVI	Decow	Lemmatisiert	Skip-gram	5	$0,01$
XVII	Decow	Lemmatisiert	Skip-gram	5	$0,1$
XVIII	Decow	Lemmatisiert	Skip-gram	5	1
XIX	Decow	Lemmatisiert	CBOW	5	$1^{-}5$
XX	Decow	Lemmatisiert	CBOW	5	$1^{-}4$
XXI	Decow	Lemmatisiert	CBOW	5	$1^{-}3$
XXII	Decow	Lemmatisiert	CBOW	5	$0,01$
XXIII	Decow	Lemmatisiert	CBOW	5	$0,1$
XXIV	Decow	Lemmatisiert	CBOW	5	1

Abbildung 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 (5.1) 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 5).
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[42].

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

Nr.	**Wortähnlichkeit ($\rho \in [-1, 1]$)**				**Analogien (in %)**	
	WORTPAARE65	WORTPAARE222	WORTPAARE350	SCHM280	GOOGLE	SEMREL
I	**-0,8096**	**-0,4640**$_{(13)}$	**-0,7302**$_{(13)}$	**-0,7094**$_{(2)}$	**44,56**	1,71
II	-0,7856	-0.4409$_{(13)}$	-0,7156$_{(13)}$	-0,6928$_{(2)}$	40,37	1,83
III	-0,7718	-0,4242$_{(13)}$	-0,6886$_{(13)}$	-0,6778$_{(2)}$	27,42	1.87
IV	-0,7681	-0,3902$_{(13)}$	-0,6834$_{(13)}$	-0,6545$_{(2)}$	25,48	1,83
V	-0,7725	-0,3889$_{(13)}$	-0,6738$_{(13)}$	-0,6529$_{(2)}$	25,54	1,67
VI	-0,7697	-0,3907$_{(13)}$	-0,6784$_{(13)}$	-0,6541$_{(2)}$	25,52	1,46
VII	-0,7255	-0,4198$_{(13)}$	-0,6924$_{(13)}$	-0,6361$_{(2)}$	30,60	1,50
VIII	-0,6149	-0,4321$_{(13)}$	-0,6530$_{(13)}$	-0,5983$_{(2)}$	26,98	1,83
IX	-0,6784	-0,4288$_{(13)}$	-0,6090$_{(13)}$	-0,5553$_{(2)}$	22,26	1,62
X	-0,5949	-0,4185$_{(13)}$	-0,5791$_{(13)}$	-0,5003$_{(2)}$	19,22	1,22
XI	-0,5890	-0.4245$_{(13)}$	-0,5700$_{(13)}$	-0,4993$_{(2)}$	18,60	1,38
XII	-0,5874	-0.4251$_{(13)}$	-0,5706$_{(13)}$	-0,5021$_{(2)}$	8,62	1,50
XIII	**-0,8247**$_{(1)}$	**-0,5066**$_{(102)}$	**-0,7494**$_{(132)}$	-0,7097$_{(48)}$	15,51	**3,01**
XIV	-0,8106$_{(1)}$	-0,4953$_{(102)}$	-0,7362$_{(132)}$	**-0,7205**$_{(48)}$	14,51	2,56
XV	-0,7786$_{(1)}$	-0,4991$_{(102)}$	-0,7136$_{(132)}$	-0,6956$_{(48)}$	13,15	2,44
XVI	-0,7576$_{(1)}$	-0,4991$_{(102)}$	-0,6990$_{(132)}$	-0,6702$_{(48)}$	11,76	2,56
XVII	-0,7578$_{(1)}$	-0,5074$_{(102)}$	-0,6986$_{(132)}$	-0,6638$_{(48)}$	11,41	2,80
XVIII	-0,7551$_{(1)}$	-0,5102$_{(102)}$	-0,6981$_{(132)}$	-0,6715$_{(48)}$	11,38	**3,01**
XIX	-0,7589$_{(1)}$	-0,4899$_{(102)}$	-0,7476$_{(132)}$	-0,6849$_{(48)}$	14,41	2,23
XX	-0,7519$_{(1)}$	-0,5144$_{(102)}$	-0,7239$_{(132)}$	-0,6731$_{(48)}$	12,82	2,15
XXI	-0,7188$_{(1)}$	-0,5371$_{(102)}$	-0,6875$_{(132)}$	-0,6520$_{(48)}$	9,68	1,75
XXII	-0,6926$_{(1)}$	**-0,5377**$_{(102)}$	-0,6429$_{(132)}$	-0,6138$_{(48)}$	6,85	1,71
XXIII	-0,6851$_{(1)}$	-0,5373$_{(102)}$	-0,6359$_{(132)}$	-0,5882$_{(48)}$	6,11	1,95
XXIV	-0,6913$_{(1)}$	-0,5349$_{(102)}$	-0,6369$_{(132)}$	-0,5930$_{(48)}$	6,15	2,07

Abbildung 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.

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.

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.

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.