RUPRECHT-KARLS-UNIVERSITÄT HEIDELBERG

BACHELORARBEIT

[Arbeitstitel] Relation Mining mit Wortvektoren

Author:
Dennis ULMER

Supervisor:
Dr. Yannick VERSLEY
Dr. Viviana NASTASE

Eine Arbeit zur Erlangung des Bachelorgrades

21. April 2016

Eidesstattliche Erklärung

Ich, Dennis ULMER, gebe hiermit die eidesstattliche Erklärung ab, dass ich meine Bachelorarbeit mit dem Titel „[Arbeitstitel] Relation Mining mit Wortvektoren“

- selbständig angefertigt
- keine anderen als die angegebenen Quellen und Hilfsmittel be- nutzt und alle wörtlichen oder sinngemäß übernommenen Textstellen als solche kenntlich gemacht habe
- Mir ist bekannt, dass die ungekennzeichnete Übernahme fremder Texte – auch aus dem Internet – als Täuschung gewertet wird und die entsprechende Prüfungsleistung als nicht erbracht gilt (Bachelor-Prüfungsordnung § 8, 4 und § 21, 4; Master-Prüfungsordnung § 8, 4 und § 22, 4; Magisterprüfungsordnung, Allgemeiner Teil § 22).

Unterschrift: _____

Datum: _____

„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

[Arbeitstitel] Relation Mining mit Wortvektoren

von Dennis ULMER

Deutscher Satz...

RUPRECHT-KARLS UNIVERSITY HEIDELBERG

Abstract

Faculty of Modern Languages
Department for Computational Linguistics

Bachelor of Arts

[Working Title] Relation Mining with word embeddings

by Dennis ULMER

English sentence...

Danksagung

The acknowledgments and the people to thank go here, don't forget to include your project advisor…

Inhaltsverzeichnis

Eidesstattliche Erklärung iii

Zusammenfassung vii

Abstract ix

Danksagung xi

1 Einleitung 1
- 1.1 Knowledge Graph Completion . . . 1
- 1.2 Ansatz . . . 2
- 1.3 Inhalt . . . 3

2 Verwandte Arbeiten 5
- 2.1 Verwandte Arbeiten . . . 5

3 Grundlagen 7
- 3.1 Neurale Netzwerke . . . 7
- 3.2 Wortvektoren . . . 7
- 3.3 Wortvektoren aus Dependenzen . . . 9

4 Vorbereitung 11
- 4.1 Vorbereitung . . . 11
  - 4.1.1 Extraktion von Named Entities . . . 11
  - 4.1.2 Aufbereitung des Korpus . . . 11
  - 4.1.3 Training der Wortvektoren . . . 11

5 Evaluation der Wortvektoren 15
- 5.1 Evaluation der Wortvektoren . . . 15
  - 5.1.1 Qualitative Evaluation . . . 15
  - 5.1.2 Quantitative Evaluation . . . 16
  - 5.1.3 Evaluationsdaten . . . 17
    - Wortpaarähnlichkeit . . . 17
    - Analogien . . . 18
  - 5.1.4 Evaluationsergebnisse . . . 18

6 Mapping 21
- 6.1 Mapping-Schritt . . . 21
  - 6.1.1 Features & Einschränkungen . . . 21
  - 6.1.2 Algorithmus . . . 21
  - 6.1.3 Parallelisierter Algorithmus . . . 21

7 Clustering **23**
7.1 Clustering 23
7.1.1 DBSCAN 23
7.1.2 Pre-Training 23
7.1.3 Parallelisiertes DBSCAN 23

8 Evaluation der Cluster **25**
8.1 Evaluation der Cluster 25

9 Diskussion **27**
9.1 Diskussion 27

10 Fazit **29**
10.1 Fazit 29

11 Ausblick **31**
11.1 Ausblick 31

A Übersicht über die Parameter zum Trainieren der Wortvektoren **33**

Literatur **35**

Abbildungsverzeichnis

3.1 Gegenüberstellung von Skip-Gram und CBOW 8
3.2 Erstellung von Dependenzkontexten beim Wortvektortraining 9

4.1 Gegenüberstellung von Skip-Gram und CBOW 12
4.2 Erstellung von Dependenzkontexten beim Wortvektortraining 13

5.1 Listen der *k* nächsten Nachbarn von Wörtern in verschiedenen Datensets. 15
5.2 Anzahl der Annotatoren und Agreement (als $Cohen's\ \kappa$) der WORTPAAR-Evaluationsdatensets. 18
5.3 Evaluationsergebnisse bei Wortähnlichkeit und Analogien . 19

A.1 Quelle und Trainingsparameter für Wortvektoren 33

Abkürzungsverzeichnis

PoS	**P**wart **o**f **S**peech
SGD	**S** tochastic **G**radient **D**escent
DECOW	German **CO**rpus from the **W**eb
CBOW	**C**ontinuous-**B**ag-**O**f-**W**ords
DBSCAN	**D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise

Symbolverzeichnis

w	Ein nicht näher spezifiziertes Wort
w_1, w_2, w_3 / a, b, c	Mehrere nicht näher spezifizierte Wörter
$\vec{v}$, $\vec{y}$, $\vec{z}$	Vektoren
$\vec{v}(Hund)$ / $\vec{v}(w)$	Der zu einem Wort zugehörige Wortvektor
$\vec{v}'$ / $\vec{v}'(w)$.	Projektion eines (Wort-)Vektors
$\oplus$	Vektorkonkatenation
$\mathcal{U}, \mathcal{V}, \mathcal{W}$	Mengen
x^*, y^*, z^*	Gewünschte Ausprägung einer Variablen
$\tilde{x}$, $\tilde{y}$, $\tilde{z}$	Möglicher Bestwert einer Variablen

For/Dedicated to/To my…

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 [und KI-Forschung; Anm. des Autors] 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 unmöglichkeit: Menschliche Champions scheitern nun gegen Maschinen beim Schach. Zuletzt scheiterte auch der Mensch auch beim Spiel Go gegen ein "intelligentes" System. In anderen Bereichen jedoch hinken die Maschinen den Prognosen hinterher. Viele dieser Bereiche haben dabei eines gemeinsam: Die Anforderung an das System, nicht nur einfache Rechenoperationen auszuführen, sondern sich ein Bild von der umgebenden Welt zu machen, Schlüsse zu ziehen und neues Wissen zu Erwerben und passend in den vorhandenen Informationsbestand einzuordnen.
Entwicklungen wie die ersten Fahrten fahrerloser Autos, den Jeopardy-Champion Watson und ähnliche zeigen den Fortschritt in diesem Bereich auf, jedoch ist die Wissenschaft von einer allgemeinen Intelligenz noch weit entfernt. Ein Grund dafür ist das Problem von Computern, dass sie im Gegensatz zum Menschen über kein Weltwissen verfügen, welches letztere sich im Laufe ihres Lebens aneignen. Dabei lernen sie

- wie Objekte in der Realwelt zueinander in Beziehung stehen
- welche Attribute von verschiedenen Objekten besessen werden
- Zusammenhänge zwischen Ereignissen zu verstehen

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

- Pläne zu schmieden und sich Stragien zurechtzulegen
- ...

Ersteres wird in der Informatik durch sog. *Ontologien* (= formale Darstellung einer Beziehung zwischen Elementen einer Menge) modelliert. Zieht man zwischen den Entitäten in der Welt nun derartige Ontologien, entsteht ein Graph, in dem die Beziehungen der Knoten untereinander mithilfe verschiedener Arten von Kanten kodiert sieht, dem *Knowledge Graph*.
Die Vervollständigung ebendieses ermöglicht Systemen, die langwierige menschliche Erlernung dieses Wissens zu kompensieren. In dieser Arbeit soll deshalb ein Ansatz vorgestellt werden, der die Lösung diesen Problems einen kleinen Schritt näherbringen könnte.

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, auf Englisch "word embeddings" trainiert werden, die semantische Information in sich kodierten. Zwar zeigen einige Untersuchung, dass sich dieser Ansatz älteren 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. Eine Reihe von Papern nutzt nun Methoden des maschinellen Lernens, um mithilfe von Trainingsdaten die Differenzvektoren für bestimmte, im Voraus ausgewählte Relationen zu trainieren und die Ergebnisse in einem Testset anzuwenden.
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. Dies wiederum könnte man dann insofern ausnutzen, indem man ein Clusteringverfahren anwendet, um diese Gruppen zu identifizieren und die Elemente jeder Gruppe danach mit der entsprechenden Relation in einen Knowledge Graph einzuordnen. Dies hätte zwei Vorteile:

1. Teile des kostenintensiven Dateneinpflegens durch menschliche Hilfe fällt weg
2. Es wird möglich abzuschätzen, welche Relationen in einem Raum von Wortvektoren tatsächlich festgehalten werden (darunter vielleicht auch einige, die man bei vorherigen Experimenten nicht berücksichtigt hatte)

Die Ergebnisse, die diese Prozedur zutage fördert sowie die Fallstricke, die sie mit sich bringt, werden in den nächsten Kapiteln beschrieben. Über jene wird nun ein kleiner Überblick gegeben.

1.3 Inhalt

In Kapitel 2 dieser Arbeit sollen verwandte Arbeiten zu diesem Thema vorgestellt werden. Darauf folgt eine Beschreibung der ein Vorbereitungsschritte, die erforderlich waren (siehe Kapitel 3) sowie eine Charakterisierung der verwendeten Ressourcen. Dabei wird auch auf das Training verschiedener Wortvektoren eingegangen, die in Kapitel 4 auf qualitative und quantitative Art und Weise auf ihre semantische Aussagekraft evaluiert werden.
Das Kapitel Nummer 5 handelt von Mapping-Vorgang, bei dem aus den Wortvektoren in einem neuen Vektorraum Punkte entstehen, die jeweils einem Wortpaar entsprechen. Dabei wird auf das theoretische Verfahren genauso eingegangen wie auch auf den benötigen Algorithmus und notwendige Einschränkungen.
In Kapitel 6 wird der Clustering-Schritt im Bezug auf die Wahl des Algorithmus, seiner Parameter und seiner Skalierbarkeit beschrieben. Daran schließt in Kapitel 7 die Evaluation der entstehenden Cluster an, bevor in Kapitel 8 die Ergebnisse sowie die Möglichkeiten und Grenzen des Verfahrens ausdrücklich diskutiert werden.
Die Arbeit schließt mit einem Fazit (Kapitel 9) und einem Ausblick für auf diese Arbeit aufbauende Untersuchungen an (Kapitel 10).

Kapitel 2

Verwandte Arbeiten

2.1 Verwandte Arbeiten

Bla

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.

UNKNOWN

3.1 Neurale Netzwerke

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 (V steht eigentlich für $\|\text{V}\|$, wird der Übersicht halber aber im Folgenden stellvertrentend dafür verwendet) 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. Wortvektoren, die mithilfe von Neuralen Netzwerken trainiert werden (im Englischen zur Abgrenzung *word embeddings* genannt), beinhalten mehr (semantische) Informationen über das dazugehörige Wort, zudem lassen sich einzelne Features nicht mehr auf eindeutig auf bestimmte Worte zurückführen.

Das Training dieser neuen Wortvektoren 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.

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 zu maximieren (Kontext bezieht sich in diesem Fall auf die Summe der rechts und links vom Eingabewort stehenden Wörter). Unsere Verlustfunktion, deren Wert es dabei zu minimieren gilt, besteht darin in der negativen logarithmischen Wahrscheinlichkeit eines Wortes gegeben seines Kontextes:

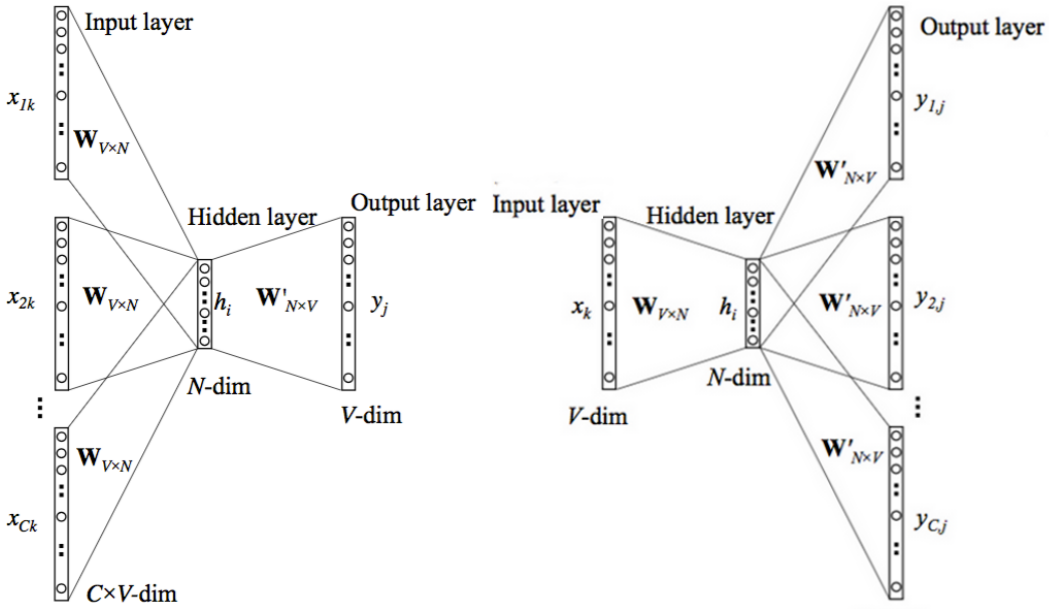


ABBILDUNG 3.1: Gegenüberstellung der beiden Trainingsmethoden CBOW (links) und Skip-Gram (rechts). CBOW versucht die Wahrscheinlichkeit eines Wortes gegeben seines Kontexts zu trainieren, Skip-Gram die Wahrscheinlichkeit eines Kontextes gegeben eines Wortes.

$$E = -log\ p(w_O|w_{I,1}, \ldots, w_{I,C}) \tag{3.1}$$

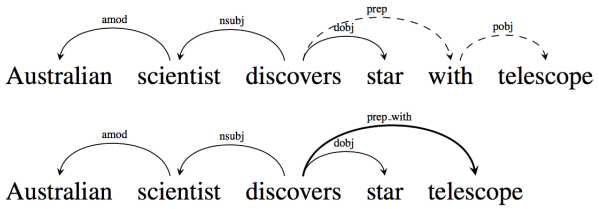
Beim Skip-gram-Modell verhält sich das Ganze genau umgekehrt, es wird versucht, den Kontext gegeben eines Eingabewortes vorherzusagen:

$$E = -log\ p(w_{I,1}, \ldots, w_O) \tag{3.2}$$

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 Wortvektoren, die dann nach dem Abarbeiten 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 Vokabels, was der Skalierbarkeit dieses Verfahrens entgegensteht. Der Aufwand kann allerdings durch Techniken wie *Hierarchisches Softmax*, bei dem der Aufwand durch einen binären Baum von $O(V)$ auf $O(log\ V)$ reduziert wird sowie *Negativem Sampling*. Bei letzterem werden "schlechte" (also Negativ-)Beispiele zum Training hinzugezogen, woher sich auch der Name des Verfahrens ableitet.

3.3 Wortvektoren aus Dependenzen

Dependenzgrammatiken untersuchen die Abhängigkeiten zwischen Wörtern eines Satzes und fügen diese in eine Dependenzstruktur ein. Anders als in der Phrasenstrukturgrammatik entsteht dabei kein Syntaxbaum mit Knoten. Worte stehen in Abhängigkeitsverhältnissen, wobei das das die Dependenz verursachende Wort alt *Regens*, das davon abhängige als *Dependenz* bezeichnet wird.
(Levy und Goldberg) machen sich dies zunutze, um den Kontext beim Training von Wortvektoren neu zu definieren: Er besteht nun nicht mehr als den umgebenden Wörtern im Satz, sondern aus den Depenzen: Für ein Word w mit den Modifizierern $m_1, \dots m_k$ und den Kopf h besteht der Kontext nun aus $(m_1, lbl_1), \dots (m_k, lbl_k), (h, lbl_h^{-1})$, wobei lbl stellvertretend für eine Dependenzrelation steht, ein -1 im Exponenten zeigt das Inverse einer solchen Relation an. Ein Beispiel dafür ist in Abb. X zu sehen.



WORD	CONTEXTS
australian	scientist/amod^{-1}
scientist	australian/amod, discovers/nsubj^{-1}
discovers	scientist/nsubj, star/dobj, telescope/prep_with
star	discovers/dobj^{-1}
telescope	discovers/prep_with^{-1}

ABBILDUNG 3.2: Beispiel der Erstellung von Wortkontexten aus Dependenzen. Dependenzen mit Präposition werden zu einer Abhängigkeit zusammengefasst. **Oben**: Dependenzstruktur. **Unten**: Extrahierte Kontexte.

Kapitel 4

Vorbereitung

4.1 Vorbereitung

Bla

4.1.1 Extraktion von Named Entities

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 Texten, die in den Jahren 2011 und 2014 von deutschsprachigen Internetseiten gecrawled und aufbereitet wurden. Dies beinhaltet PoS-Tagging, Chunking, Lemmatisierung, das Markieren von Eigennamen (Named Entities) und dem Hinzufügen von Metadaten. Die Sätze liegen darin im CoNLL-Format[1] 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 drei Version 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.
- Eine Datei mit dem lemmatisierten Tokens, sortiert nach den für jeden Satz geparsten Dependenzen, ein Satz pro Zeile.

Die Dependenzen wurden dabei mit dem Tool X erzeugt. [Blabla erläutern wenn Punkt erledigt.]

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 u. a., 2013) vorgestellt. Die Erklärung der Funktionsweise wird im nachfolgenden Teil recht klein gehalten, für eine ausführlichere und verständliche Ausführung wird beispielsweise

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

die Arbeit von (Rong, 2014) empfohlen.

Als Input fungieren "One-Hot"-Vektoren, welche genau so viele Dimensionen wie Worte im Vokabular besitzen. Der Wert der Dimensionen entspricht 1, wenn die Dimension der Stelle des aktuellen Wortes im Vokabular entspricht und ansonsten 0. Die Modelle bestehen aus drei Schichten, namentlich *Input*, *Hidden* und *Output*. Input und Output besitzen die Dimensionalität von V (V steht eigentlich für $\|\mathrm{V}\|$, wird der Übersicht halber aber im Folgenden stellvertrentend dafür verwendet), Hidden die von N.

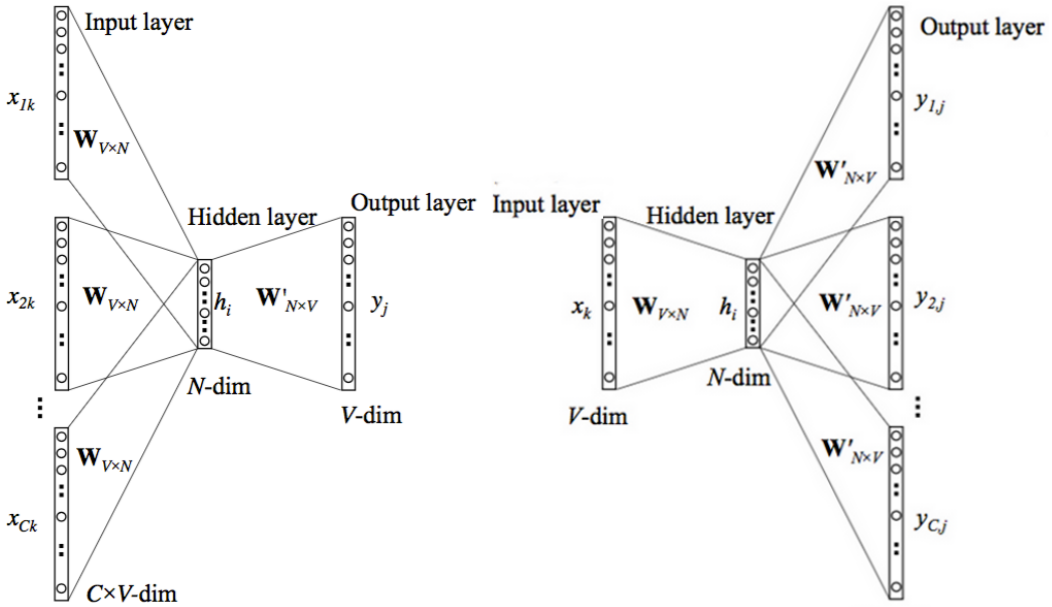


ABBILDUNG 4.1: Gegenüberstellung der beiden Trainingsmethoden CBOW (links) und Skip-Gram (rechts). CBOW versucht die Wahrscheinlichkeit eines Wortes gegeben seines Kontexts zu trainieren, Skip-Gram die Wahrscheinlichkeit eines Kontextes gegeben eines Wortes.

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 zu maximieren (Kontext bezieht sich in diesem Fall auf die Summe der rechts und links vom Eingabewort stehenden Wörter). Unsere Verlustfunktion, deren Wert es dabei zu minimieren gilt, besteht darin in der negativen logarithmischen Wahrscheinlichkeit eines Wortes gegeben seines Kontextes:

$$E = -log\ p(w_O|w_{I,1}, \ldots, w_{I,C}) \tag{4.1}$$

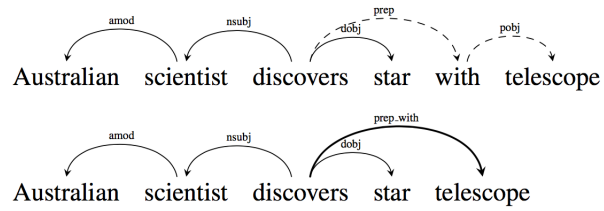
Beim Skip-gram-Modell verhält sich das Ganze genau umgekehrt, es wird versucht, den Kontext gegeben eines Eingabewortes vorherzusagen:

$$E = -log\ p(w_{I,1}, \ldots, w_O) \tag{4.2}$$

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 Wortvektoren, die dann nach dem Abarbeiten 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 Vokabels, was der Skalierbarkeit dieses Verfahrens entgegensteht. Der Aufwand kann allerdings durch Techniken wie *Hierarchisches Softmax* oder, wie im Falle von `word2vec`, *Negativem Sampling* von $O(V)$ auf $O(log\ V)$ reduziert werden. Bei letzterem werden "schlechte" Beispiele zum Training hinzugezogen, woher sich auch der Name des Verfahrens ableitet.

Dependenzgrammatiken untersuchen die Abhängigkeiten zwischen Wörtern eines Satzes und fügen diese in eine Dependenzstruktur ein. Anders als in der Phrasenstrukturgrammatik entsteht dabei kein Syntaxbaum mit Knoten. Worte stehen in Abhängigkeitsverhältnissen, wobei das das die Dependenz verursachende Wort alt *Regens*, das davon abhängige als *Dependenz* bezeichnet wird.
(Levy und Goldberg) machen sich dies zunutze, um den Kontext beim Training von Wortvektoren neu zu definieren: Er besteht nun nicht mehr als den umgebenden Wörtern im Satz, sondern aus den Depenzen: Für ein Word w mit den Modifizierern $m_1, \dots m_k$ und den Kopf h besteht der Kontext nun aus $(m_1, lbl_1), \dots (m_k, lbl_k), (h, lbl_h^{-1})$, wobei lbl stellvertretend für eine Dependenzrelation steht, ein -1 im Exponenten zeigt das Inverse einer solchen Relation an. Ein Beispiel dafür ist in Abb. X zu sehen.



WORD	CONTEXTS
australian	scientist/amod^{-1}
scientist	australian/amod, discovers/nsubj^{-1}
discovers	scientist/nsubj, star/dobj, telescope/prep_with
star	discovers/dobj^{-1}
telescope	discovers/prep_with^{-1}

ABBILDUNG 4.2: Beispiel der Erstellung von Wortkontexten aus Dependenzen. Dependenzen mit Präposition werden zu einer Abhängigkeit zusammengefasst. **Oben**: Dependenzstruktur. **Unten**: Extrahierte Kontexte.

Zum Training der Vektoren wurde das C-Tool *Word2Vec* von (Mikolov u. a.) verwendet. 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 hochfrequente Worte

- `-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 mit 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.

Kapitel 5

Evaluation der Wortvektoren

5.1 Evaluation der Wortvektoren

An dieser Stelle sollen die verschiedenen Ansätze zum Trainieren von Wortvektoren, die im vorherigen Kapitel vorgestellt werden, miteinander verglichen werden. Zu diesem Zweck sollte zuerst eine Frage gestellt werden: Was macht eine Menge von Wortvektoren "besser" bzw. "schlechter" als andere?
Da der Vorteil von Wortvektoren darin besteht, semantische Informationen zu beinhalten, wird diese Frage meist dahingehend beantwortet, dass Vektoren dann als überlegen an zu sehen sind, wenn sie eine höhere semantische Ausdruckskraft besitzen. Um dies festzustellen, haben sich in Veröffentlichungen zu diesem Thema bestimmte Vorgehensweisen durchgesetzt, die in dne 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 interpetierbare Ergebnisse liefern. Deshalb sind sie für einen ersten Ausdruck auch durchaus geeignet, sollten wenn möglich aber nicht als alleinige Kriterium für eine Bewertung hinzugezogen werden, da sie meisten nie die Gesamtheit aller in den Ergebnissen enthaltenen Informationen darstellen können.
Im Beispiel der Wortvektoren werden beispielsweise einige Wörter des Vokabulars stellvertretend ausgewählt und zu diesen die k nächsten Nachbarn im Vektorraum gesucht. Unter der Annahme, dass in Vektorräumen von Wortvektoren ähnliche Wörter nahe zusammenliegen, sollte diese Liste nah verwandte Begriffe zutage fördern (siehe Abb. X).

	WORT 1	WORD 2	WORT 3	WORT 4	WORT 5
Datenset 1					
Datenset 2					
Datenset 3					

ABBILDUNG 5.1: Listen der k nächsten Nachbarn von Wörtern in verschiedenen Datensets.

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 beschrieben werden.

5.1.2 Quantitative Evaluation

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

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

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

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

   Danach kann mit *Spearman's* ρ bzw. *Spearman's rank correlation coefficient* 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 ist $\rho \in [-1, 1]$ den Grad der Korrelation anzeigt, 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 $b*$ aus dem Vokabular $\mathcal{V}$ gesucht wird, welches besonders ähnlich zu b und $a*$ aber unähnlich zu a ist:

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

   Sind alle Vektoren der Länge eins, so kann diese Gleichung umformuliert werden:

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

   Diese Methode wird gemeinhin als 3COSADD bezeichnet. (Autor) etablierten dazu jedoch eine Alternative namens 3COSMUL, die bei Tests bessere Ergebnisse produziert:

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

   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 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 (welche Entitäten sind enthalten, wie oft kommen diese vor, welche semantische Relationen wurden ausgewählt, wurden diese maschinell oder per Hand erzeugt).
Aus diesem Grund sollen die benutzten Evaluationssets im hierauf folgenden Abschnitt näher beleuchtet werden.

5.1.3 Evaluationsdaten

Wortpaarähnlichkeit

Im Englischen wird für die Wortähnlichkeitsevaluation häufig das WORDSIM353-Datenset 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.

WORTPAARE65, WORTPAARE222 und WORTPAARE350 entstammen der Arbeit von [REFERENZ]. Dabei werden Wortpaaren Werte von 0 ($\hat{=}$ vollkommen unzusammenhängend) bis 4 ($\hat{=}$ stark zusammenhängend) bewertet. Die Anzahl der menschlichen Annotatoren sowie deren Übereinstimmung sind in Fig. X festgehalten.

Datenset	#Annotatoren	κ
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 WORTPAAR-Evaluationsdatensets.

Analogien

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

SEMREL wurde aus Synonomie-, Antonomie- und Hypernomie-Beziehungen von [REFERENZ] für das Deutsche und Englische konstruiert. Dabei werden Substantive, Verben und Adjektive berücksichtigt. In der deutschen Variante sind 2.462 (recht schwierige) Analogien enthalten, die aus teilweise sehr seltenen Wörter kreiert wurden.

5.1.4 Evaluationsergebnisse

Datenset	Wortähnlichkeit ($\rho \in [-1, 1]$)				Analogien (in %)	
	WORTPAARE65	WORTPAARE222	WORTPAARE350	SCHM280	GOOGLE	SEMREL
Mark I	-0.8068	$-0.2455_{(13)}$			44,56	
Mark II	-0.8012	$-0.2576_{(13)}$			40,37	
Mark III	-0.7694	$-0.2417_{(13)}$			27,42	35,11
Mark IV	-0.7706	$-0.2383_{(13)}$			25,48	32,03
Mark V	-0.7679	$-0.2299_{(13)}$			25,54	
Mark VI	-0.7648	$-0.2337_{(13)}$				
Mark VII	-0.7246	$-0.2180_{(13)}$				1,50
Mark VIII	-0.6775	$-0.2377_{(13)}$				
Mark IX	-0.6161	$-0.2481_{(13)}$				1,62
Mark X	-0.5807	$-0.2753_{(13)}$				1,22
Mark XI	-0.5775	$-0.2838_{(13)}$				1,38
Mark XII	-0.5743	$-0.2770_{(13)}$				
Mark XIII	$\mathbf{-0.8218}_{(2)}$	-0.3145				3,01
Mark XIV	$-0.8097_{(2)}$					2,56
Mark XV	$-0.7759_{(2)}$					2,44
Mark XVI	$-0.7596_{(2)}$					2,56
Mark XVII	$-0.7533_{(2)}$					2,80
Mark XVIII	$-0.7498_{(2)}$					3,01
Mark XIX	$-0.7543_{(2)}$					2,23
Mark XX	$-0.7502_{(2)}$					2,15
Mark XXI	$-0.7182_{(2)}$					1,75
Mark XXII	$-0.6949_{(2)}$					1,71
Mark XXIII	$-0.6927_{(2)}$					1,95
Mark XXIV	$-0.6865_{(2)}$					2,07

ABBILDUNG 5.3: Evaluationsergebnisse bei Wortähnlichkeit und Analogien für die verschiedenen Datensets. Für weitere Informationen über die Grundlage der Vektoren siehe Appendix A. Wörter außerhalb des Vokabulars wurden entweder als Fehler gerechnet, oder werden, falls anders nicht möglich, als Zahl im Index in runden Klammern angegeben.

Kapitel 6

Mapping

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

JASON WHITTINGTON

6.1 Mapping-Schritt

6.1.1 Features & Einschränkungen

6.1.2 Algorithmus

6.1.3 Parallelisierter Algorithmus

Kapitel 7

Clustering

7.1 Clustering

Bla

7.1.1 DBSCAN

7.1.2 Pre-Training

7.1.3 Parallelisiertes DBSCAN

Kapitel 8

Evaluation der Cluster

8.1 Evaluation der Cluster

Bla

Kapitel 9

Diskussion

9.1 Diskussion

Bla

Kapitel 10

Fazit

10.1 Fazit

Bla

Kapitel 11

Ausblick

11.1 Ausblick

Bla

Anhang A

Übersicht über die Parameter zum Trainieren der Wortvektoren

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

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

Literatur

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.

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

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

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