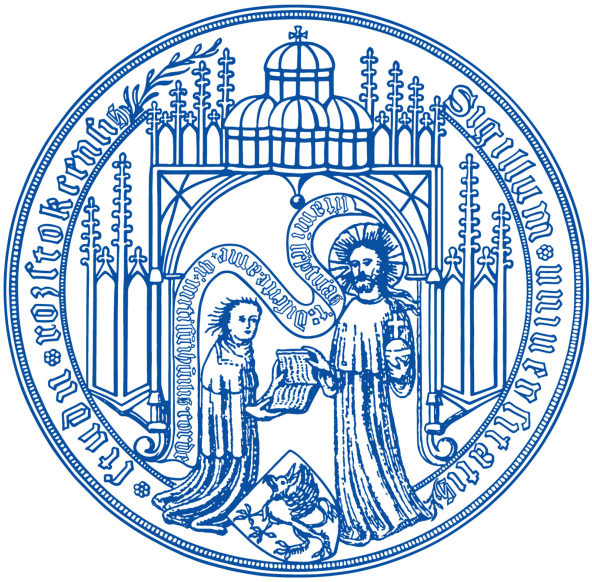
Wortvektoren

Masterarbeit

Universität Rostock
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Mathematik



vorgelegt von : Bastian Marc Laasch

Erstgutachter : Dr. Tobias Strauß
Zweitgutachter : Prof. Dr. Roger Labahn
Betreuer : Dr. Tobias Strauß

Abgabedatum : 3. September 2018
Tag der Verteidigung : 19. September 2018

Inhaltsverzeichnis

Abkürzungs- und Symbolverzeichnis

NLP	Natural Language Processing
NCE	Noise-Contrastive Estimation
NES	Negative-Sampling
DCW	Dynamic-Context-Window
AKV	Addition von Kontextvektoren

$w, \boldsymbol{w}$	Wort, Wortvektor des Wortes w
$w_1, \ldots, w_T$	Textkorpus bestehend aus T Wörtern
count(.)	Häufigkeit des Funktionsarguments im Textkorpus
$\cos_{\text{sim}}(\mathbf{x}, \mathbf{y})$	Kosinus-Ähnlichkeit der Vektoren $\mathbf{x}$ und $\mathbf{y}$
$\text{euk}_{\text{sim}}(\mathbf{x}, \mathbf{y})$	Euklid-Ähnlichkeit der Vektoren $\mathbf{x}$ und $\mathbf{y}$
$\mathcal{V}$	Vokabular auf Basis des Textkorpus
d	Dimension der Wortvektoren
s	Window-Size
$q(.)$	Noise-Verteilung
k	Anzahl der Noise-Muster bei der NCE oder beim NES
$\hat{w}$	entsprechend der Verteilung q ausgewähltes Wort
λ	Koeffizient für die Konvexkombination
$\boldsymbol{\theta}$	Menge der zu optimierenden Parameter des zugrundeliegenden Modells
$E, \mathbf{e}_w$	Einbettungsmatrix, Einbettungsvektor des Wortes w
$C, \mathbf{c}_w$	Kontextmatrix, Kontextvektor des Wortes w
(w_t, w_{t+i})	Trainingsmuster im Textkorpus mit dem Zentrumswort w_t und dem Nachbarschaftswort w_{t+i} mit $i \in \{-s, \ldots, s\}$ und $i \neq 0$; allgemeiner kann w_{t+i} auch als beliebiges Element aus $\mathcal{V}$ fungieren
$p_{\boldsymbol{\theta}}(w_{t+i} \mid w_t)$	bedingte Modellwahrscheinlichkeit für das Auftreten des Wortes w_{t+i} in der Nachbarschaft von w_t
$\hat{p}_{\boldsymbol{\theta}}(w_{t+i} \mid w_t)$	bedingte nicht-normalisierte Modellwahrscheinlichkeit für das Auftreten des Wortes w_{t+i} in der Nachbarschaft von w_t
$\tilde{p}(w_{t+i} \mid w_t)$	bedingte empirische Wahrscheinlichkeit für das Auftreten des Wortes w_{t+i} in der Nachbarschaft von w_t auf Grundlage des Textkorpus
$\mathcal{Z}_{\boldsymbol{\theta}}(w_t)$	Normalisierungsterm der Softmax-Funktion für das Zentrumswort w_t
N	Co-Occurrence-Matrix
N_{w_t}	Anzahl aller Nachbarschaftswörter des Wortes w_t im Textkorpus
$N_{w_t, w_{t+i}}$	Häufigkeit, wie oft das Wort w_{t+i} in der Nachbarschaft von w_t im Textkorpus auftritt

$N_{\max}$	Schwellwert für Worthäufigkeiten
$J_{\cdot}(\ldots), \mathcal{J}_{\cdot}(\ldots)$	Zielfunktion über ein Trainingsmuster, Zielfunktion über alle im Textkorpus vorhandenen Trainingsmuster
$H\left(p_1, p_2\right)$	Kreuzentropie der Verteilungen p_1 und p_2
$\Gamma_{\boldsymbol{\theta}}\left(w_j\right)$	Notationskonvention der Form $\Gamma_{\boldsymbol{\theta}}\left(w_j\right) \coloneqq -\mathbf{c}_{w_j}^T \mathbf{e}_{w_t}$

Kapitel 1

Einleitung

"Die Bedeutung eines Wortes ist sein Gebrauch in der Sprache."

— Ludwig Wittgenstein (1889 - 1951), *Philosophische Untersuchungen, 43*

Von Alexa bis hin zum Google Translator - Systeme, die menschliche Sprache analysieren und verstehen, sind in der heutigen Zeit in aller Munde und sollen unser alltägliches Leben vereinfachen und erleichtern. Das sogenannten *Natural Language Processing* (*NLP*) beschäftigt sich genau mit dieser Thematik und hat das Ziel, Techniken zur maschinellen Verarbeitung von Sprache unter Nutzung von *Künstlicher Intelligenz* zu entwickeln. Dieses Feld hat in den letzten Jahren eine rasante Entwicklung aufgenommen, die in Zukunft viele interessante Innovationen verspricht.

Die Problemstellung ist dabei in der Regel zweigeteilt. So muss zum einen die gesprochene Sprache oder der (hand-) geschriebene Text erkannt werden. Zum anderen soll das System anschließend die erfassten Sätze auch inhaltlich verstehen. Ein Computer, der diese beiden Aufgaben meistert, ist dann in der Lage, mit einem Menschen via Sprache zu kommunizieren.

Eine grundlegende Notwendigkeit, um dieses Ziel zu erreichen, besteht darin, dass das System ein Verständnis für die einzelnen Wörter besitzt. Dabei stellt sich dann die ganz grundsätzliche Frage: "Welche Bedeutung besitzt ein bestimmtes Wort?". Über diese Problematik lässt sich bekanntlich streiten, sodass deren Beantwortung durch ein Computersystem eine anspruchsvolle Herausforderung darstellt.

Eine Möglichkeit, um den Sinn von Wörtern aufzudecken, besteht in der Generierung von *Wortvektoren* (*Worteinbettungen*, *Word Embeddings*), die versuchen, die Bedeutung der Wörter mit einer Menge von Zahlen zu erfassen. Diese Vektoren können dann zur Lösung verschiedenster Problemstellungen aus dem Bereich des NLP beitragen (Eigennamenerkennung, Textklassifikation, Automatische Textzusammenfassung, Sentiment Analysis, Part-of-speech-Tagging, Maschinelle Übersetzung, ...), indem die Wörter in Form dieser Einbettungen den Systemen übergeben werden. Im Zuge dieser Masterarbeit wollen wir uns damit beschäftigen, auf welche Weise nützliche Wortvektoren erzeugt werden und wie wir deren Güte sinnvoll beurteilen können.

Zu diesem Zweck werden wir zunächst in Kapitel 2 einige Grundlagen erläutern, um auf das erste von zwei betrachteten Wortvektor-Modellen hinführen zu können. Dieses ist das sogenannte *Skip-Gram-Modell* (Kapitel 3), welches zu den populärsten Ansätzen zählt. Anschließend soll in Kapitel 4 das *GloVe-Modell* dargestellt werden. In Kapitel 5 wollen wir uns mit Methoden auseinandersetzen, die die Qualität der Modellausgaben einschätzen, was wir zu guter Letzt mit Experimenten und deren Auswertung in Kapitel 6 abrunden wollen. Es sei schon an dieser Stelle angemerkt, dass wir ausschließlich Vektoren für Wörter in englischer Sprache generieren werden.

Kapitel 2

Grundlagen

Das Ziel eines Wortvektor-Modells ist es, die einzelnen Wörter eines zugrundeliegenden Vokabulars, die als diskrete Objekte interpretiert werden können, als reelle Vektoren darzustellen, sodass ihre semantischen Bedeutungen mit Hilfe dieser Darstellung möglichst gut wiedergegeben werden. So sollen u.a. die Einbettungen von ähnlichen Wörtern mit identischen Bedeutungen nahe beieinander liegen, während die für entgegengesetzte Wörter weit voneinander entfernt sein sollen.

Definition 2.1 (Wortvektor)**.** *Gegeben sei eine Abbildung, welche die Wörter aus einem Vokabular $\mathcal{V}$ in den Vektorraum $\mathbb{R}^d$ projiziert. Als Wortvektor des Wortes $w \in \mathcal{V}$ wird dessen Bild $\boldsymbol{w} \in \mathbb{R}^d$ unter dieser Abbildung bezeichnet.*

Die wohl einfachste Transformation eines Vokabulars in den Vektorraum $\mathbb{R}^d$ entspricht einer *One-Hot-Kodierung* (vgl. [43]), bei der jedem einzelnen Wort aus $\mathcal{V}$ eine eigenen Dimension zugewiesen wird und ein binärer Wert anzeigt, ob das entsprechenden Wort betrachtet wird (1) oder nicht (0). In diesem Fall gilt für die Dimension unserer Wortvektoren

$$d = |\mathcal{V}| \ ,$$

was sehr unpraktikabel erscheint, wenn sich vergegenwärtigt wird, dass das Vokabular in der Regel eine Ansammlung von mehreren Tausend Wörtern ist und unsere Vektoren bei einer One-Hot-Kodierung zudem dünn-besetzt sind, da immer nur genau ein Eintrag ungleich 0 ist. Aufgrund dessen ist auch ein Vergleich von zwei Vektoren, z.B. durch Berechnung des Abstands, nicht aussagekräftig, da die Vektoren von zwei beliebigen aber verschiedenen Wörtern immer dieselbe Distanz besitzen und dies unabhängig davon, welche Bedeutungen sie tragen.

Bemerkung 2.1. *Ein gutes Wortvektor-Modell soll eine Abbildung zwischen einem Raum, in dem jedes einzelne Wort eine eigenen Dimension besitzt, in einen reellen Vektorraum mit geringerer Dimension, d.h. (deutlich) kleiner als $|\mathcal{V}|$, liefern. Weiterhin sollen die erzeugten Vektoren möglichst nicht dünn-besetzt sein.*

Wortvektoren können zum einen dazu verwendet werden, um die Ähnlichkeit zweier Wörter zu ermitteln. Dies kann durch die Nutzung der *Kosinus-Ähnlichkeit* zwischen den entsprechenden Einbettungen realisiert werden, die standardmäßig im Zusammenhang mit Wortvektoren verwendet wird. Sie berücksichtigt nicht die Längen wie beim euklidischen Abstand, sondern nur die Winkel zwischen den Vektoren und somit deren Richtungen, von denen wir hier annehmen wollen, dass diese aussagekräftiger sind.

Definition 2.2 (Kosinus-Ähnlichkeit und -Distanz)**.** *Die Kosinus-Ähnlichkeit zweier Vektoren $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d \setminus \{\boldsymbol{0}\}$ ist definiert durch*

$$\cos_{\text{sim}}(\boldsymbol{x}, \boldsymbol{y}) \coloneqq \cos\left(\sphericalangle\left(\boldsymbol{x}, \boldsymbol{y}\right)\right) = \frac{\boldsymbol{x}^T \boldsymbol{y}}{\|\boldsymbol{x}\|_2 \cdot \|\boldsymbol{y}\|_2} \quad \in [-1, 1] \ , \tag{2.1}$$

wobei ein Wert von 1 ($\hat{=}\ 0°$*, Vektoren zeigen in dieselbe Richtung*) *eine hohe und ein Wert von* -1 ($\hat{=}\ 180°$*, Vektoren zeigen in entgegengesetzte Richtung*) *eine geringe Ähnlichkeit anzeigt. Die Kosinus-Distanz ist gegeben durch*

$$\cos_{\text{dis}}(\boldsymbol{x}, \boldsymbol{y}) \coloneqq 1 - \cos_{\text{sim}}(\boldsymbol{x}, \boldsymbol{y}) \ . \tag{2.2}$$

Zum anderen können Wortvektoren, wie schon in der Einleitung erwähnt, für eine Vielzahl an Aufgabenstellungen aus dem Bereich des NLP genutzt werden, da sie Wörter nicht als `Strings` darstellen, deren Computer-interne Kodierungen in der Regel wenig brauchbare Informationen liefern, sondern als eine Menge von reellen Zahlen, die die semantischen Bedeutungen der Wörter kodieren und so in der Lage sind, Strukturen innerhalb der zugrundeliegenden Sprache zu erfassen. Natürliche Sprache kann auf diese Weise dem System verständlich gemacht werden.

2.1 Verteilungshypothese

In den vergangenen Jahren wurden verschiedene Modelle entwickelt, um Wortvektoren zu berechnen. Sie beruhen alle (mehr oder weniger) auf der *Verteilungshypothese* (vgl. [41]).

Definition 2.3 (Verteilungshypothese)**.** *Wörter, die in identischen Kontexten auftreten, neigen auch dazu, ähnliche Bedeutungen zu haben.*

Die Kernidee dieser Annahme besteht darin, dass eine Korrelation zwischen der Ähnlichkeit der Verteilungen der Wörter und der Ähnlichkeit ihrer Bedeutungen besteht. Die verschiedenen Ansätze, die diese Hypothese nutzen, können in zwei Kategorien eingeteilt werden (vgl. [4]): *Count-based-Methoden* und *Predictive-based-Methoden.*

Bei den Count-based-Methoden wird, z.B. durch Nutzung einer *Co-Occurrence-Matrix* (vgl. Definition 4.1), die Statistik ermittelt, wie oft ein Wort zusammen mit seinen entsprechenden Nachbarwörtern in einem gegebenen Dokumentenkorpus auftritt. Auf Grundlage dieser Zählungen können dann, z.B. mit Hilfe von Matrixfaktorisierungen, die gesuchten Vektoren erzeugt werden. Sie stellen die ersten Ansätze dar, um Wörter in einen Vektorraum einzubetten. Daher existieren eine Vielzahl solcher Verfahren.

Die Predictive-based-Methoden versuchen hingegen, basierend auf Wortvektoren zu lernen, ein Wort eines Textes durch Betrachtung von dessen Nachbarn bzw. dessen Kontext richtig zu prognostizieren.

Beide Herangehensweisen haben ihre Vor- und Nachteile (vgl. [36]). Die Count-based-Ansätze sind in der Lage, die statistischen Informationen, die der zugrundeliegende Text enthält, effizient zu verarbeiten. Jedoch fällt es ihnen schwerer, kompliziertere Strukturen innerhalb des Vektorraums, in dem die Wortvektoren liegen, zu erkennen (z.B. das Verhältnis von Singular und Plural oder Steigerungen von Adjektiven). Die Predictive-based-Methoden sind dagegen in der Lage, solche Beziehungen besser zu erfassen. Allerdings ist bei ihnen oftmals der Rechenaufwand höher, da sie nicht auf einer globalen Zählung über den gesamten Textkorpus, sondern auf lokalen Kontextfenstern operieren.

2.2 Sprachmodelle

Da insbesondere die Predictive-based-Methoden sehr eng mit sogenannten *Sprachmodellen* (*Language Models*) (vgl. [38]) verbunden sind, wollen wir an dieser Stelle einige Ausführungen zu

diesem Thema machen. Dazu betrachten wir eine Sequenz von Wörtern (z.B. ein Satz oder ein Text) $w_1, w_2, \ldots, w_M$. Sprachmodelle versuchen nun, die Wahrscheinlichkeit

$$p\left(w_1, \ldots, w_M\right) \tag{2.3}$$

ihres gemeinsamen Auftretens zu schätzen. Es ist also eine Wahrscheinlichkeitsverteilung über Wörtern zu ermitteln. Handelt es sich bei der Sequenz um eine grammatikalisch valide und sinnvolle Aussage, so soll diese mit einer hohen Wahrscheinlichkeit belegt werden. Bei Nonsens soll hingegen das Modell eine geringe Wahrscheinlichkeit angeben.

Definition 2.4 (Bedingte Wahrscheinlichkeit)**.** *Die bedingte Wahrscheinlichkeit des Ereignisses A gegeben das Ereignis B mit $p(B) > 0$ ist definiert durch*

$$p(A \mid B) \coloneqq \frac{p(A \cap B)}{p(B)} = \frac{p(A, B)}{p(B)} \ . \tag{2.4}$$

Betrachten wir allgemeiner die Sequenz / zugrundeliegende Textgesamtheit $w_1, \ldots, w_T$, deren Elemente einem Vokabular $\mathcal{V}$ angehören und auf deren Grundlage ein Sprachmodell erzeugt werden soll. Für die gemeinsame Wahrscheinlichkeit dieser Sequenz ergibt sich durch Erweiterung und unter Verwendung von (2.4)

$$\begin{aligned} p\left(w_1, \ldots, w_T\right) &= p\left(w_1\right) \cdot \frac{p\left(w_1, w_2\right)}{p\left(w_1\right)} \cdot \frac{p\left(w_1, w_2, w_3\right)}{p\left(w_1, w_2\right)} \cdot \ldots \cdot \frac{p\left(w_1, \ldots, w_T\right)}{p\left(w_1, \ldots, w_{T-1}\right)} \\ &= p\left(w_1\right) \cdot p\left(w_2 \mid w_1\right) \cdot p\left(w_3 \mid w_1, w_2\right) \cdot \ldots \cdot p\left(w_T \mid w_1, \ldots, w_{T-1}\right) \\ &= \prod_{t=1}^{T} p\left(w_t \mid w_1, \ldots, w_{t-1}\right) \ . \end{aligned} \tag{2.5}$$

Zur Vereinfachung nehmen wir an, dass unser Modell annähernd die *Markow-Eigenschaft* erfüllt. Dies bedeutet grob gesprochen, dass wir eine gewisse Gedächtnislosigkeit voraussetzen können, sodass es im Produkt von (2.5) genügt, immer nur maximal die jeweils $n-1$ vorangegangenen Wörter zu betrachten, wenn das kommende Wort vorausgesagt werden soll. Diese Annahme kann dadurch motiviert werden, weil direkte Nachbarn mehr Einfluss auf die Wahl des aktuellen Wortes haben als die weiter entfernten Textelemente. Dies führt schließlich auf

$$p\left(w_t \mid w_1, \ldots, w_{t-1}\right) \approx p\left(w_t \mid w_{t-n+1}, \ldots, w_{t-1}\right) \ , \tag{2.6}$$

was nun das Ergebnis

$$p\left(w_1, \ldots, w_T\right) \approx \prod_{t=1}^{T} p\left(w_t \mid w_{t-n+1}, \ldots, w_{t-1}\right) \tag{2.7}$$

liefert. Es ist anzumerken, dass die Wörter nicht satzübergreifend betrachtet werden, sodass z.B. bei einem Wort zu Beginn eines Satzes keine vorangegangenen Elemente beachtet werden müssen bzw. können.

Bemerkung 2.2. *Der einfachste Fall in (2.7) liegt für $n = 1$ (Unigram-Modell) vor. Es werden also keine vorangegangenen Wörter in Betracht gezogen und somit stochastische Unabhängigkeit für das Auftreten der einzelnen Elemente vorausgesetzt*

$$p\left(w_1, \ldots, w_T\right) \approx \prod_{t=1}^{T} p\left(w_t\right) \ . \tag{2.8}$$

Dies ist jedoch keine realistische Annahme, da die Verwendung von Wörtern sehr wohl vom bisherigen Kontext abhängt.

Nun stellt sich die Frage, wie die Faktoren des Produkts (2.7) berechnet werden können. In sogenannten *N-Gram-Modellen* werden N-Gramme, d.h. N aufeinanderfolgende Wörter, gezählt. Die einzelnen bedingten Wahrscheinlichkeiten berechnen sich dann basierend auf den Häufigkeiten, mit denen die entsprechenden N-Gramme im Text auftreten. Dies führt auf die Darstellung

$$p\left(w_t \mid w_{t-n+1}, \ldots, w_{t-1}\right) = \frac{\text{count}\left(w_{t-n+1}, \ldots, w_{t-1}, w_t\right)}{\text{count}\left(w_{t-n+1}, \ldots, w_{t-1}\right)} \ , \tag{2.9}$$

wobei die Funktion count (.) angibt, wie häufig ihr Funktionsargument im Text $w_1, \ldots, w_T$ vorkommt.

Aber auch mit Hilfe von *Neuronalen Netzen* können die gesuchten Faktoren ermittelt werden. In solchen Fällen ist von *Neuronalen Sprachmodellen* die Rede (vgl. [5]). Das System versucht dabei, seine Modellparameter $\boldsymbol{\theta}$ so zu aktualisieren, sodass die Wahrscheinlichkeit maximiert wird, in jedem Zeitschritt t das richtige Wort w_t des zugrundeliegenden Textes zu prognostizieren (*Maximum-Likelihood-Methode*), wobei die vorangegangenen Wörter dem Netz als Input übergeben worden sind. Einfachheitshalber werden die Wahrscheinlichkeiten logarithmiert, sodass die log-Wahrscheinlichkeit (auch *Log-Likelihood-Funktion*) des gesamten Korpus maximiert

$$\log p_{\boldsymbol{\theta}}\left(w_1, \ldots, w_T\right) \to \max \tag{2.10}$$

bzw. die negative log-Wahrscheinlichkeit minimiert werden soll. Zusammen mit (2.7) und der Einführung eines Vorfaktors folgt das Optimierungsproblem

$$-\frac{1}{T}\sum_{t=1}^{T} \log p_{\boldsymbol{\theta}}\left(w_t \mid w_{t-n+1}, \ldots, w_{t-1}\right) \to \min \ . \tag{2.11}$$

Die für uns nun wichtige Kernidee ist hierbei, dass die Wörter dem Netz in Form von Vektoren übergeben werden und dieses dann auf Basis derer die vom Sprachmodell gesuchten Wahrscheinlichkeiten ermittelt. Diese Vektoren sind ein Bestandteil der Netzparameter und werden daher während des Trainings des Modells auch aktualisiert.

Bemerkung 2.3. *Es können Wortvektor-Modelle entwickelt werden, die als "Fake-Task" ein Sprachmodell nutzen. Dies bedeutet, dass das System nicht direkt darauf ausgerichtet ist, Einbettungen von Wörtern zu erzeugen, sondern es soll eine Verteilung über einem Vokabular schätzen, wobei aber die eigentlich gesuchten Wortvektoren implizit durch bestimmte Modellparameter gegeben sind.*

Viele populäre Sprachmodelle können lediglich auf die vorangegangenen Wörter blicken, um die Prognosen zu berechnen. Wortvektor-Modelle sind hingegen nicht von dieser Restriktion betroffen.

Durch die Nutzung dieser Fake-Task sind wir dementsprechend nicht auf Datensätze angewiesen, die für jedes Wort einen Vektor als Zielwert enthalten, um das System via *Supervised Learning* zu trainieren. Stattdessen haben wir hier nun eine Anwendung des *Unsupervised Learnings*, welche als Trainingsdaten lediglich eine große Menge an Texten benötigt.

Zudem hat sich gezeigt, dass Wortvektor-Modelle entwickelt werden können, die dieses Vorgehen nutzen und auf eine komplexe *Deep-Learning-Architektur* verzichten, aber dennoch sehr gute Ergebnisse liefern. Dies liegt zum einen daran, dass aufgrund der Einfachheit die Modelle auf großen Datenmengen trainiert werden können, und zum anderen, dass sie durch die wenigen Modellparameter dazu gezwungen sind, den Großteil der Informationen mit Hilfe der Wortvektoren zu kodieren. Trotz alledem treten auch hier Probleme hinsichtlich des Rechenaufwandes auf, mit denen auch wir im Folgenden konfrontiert werden (vgl. Bemerkung 3.4).

Kapitel 3

Das Skip-Gram-Modell

Eine Predictive-based-Methode, die die beschriebene Fake-Task nutzt, ist das Word2Vec-Modell (vgl. [27], [29], [31], [38] und [43]), mit dem in der Vergangenheit sehr aussagekräftige Wortvektoren generiert werden konnten, obwohl es lediglich auf einem sehr einfachen Neuronalen Netz basiert. Es existieren zwei Varianten: das *Continuous Bag-of-Words-Modell* (*CBOW*) und das *Skip-Gram-Modell*, wobei wir im Rahmen dieser Arbeit nur das Letztere weiter ausführen wollen, da beide Ansätze dazu neigen, ähnliche Wortvektoren zu berechnen (vgl. [30])

Angenommen, wir haben ein "gleitendes" Fenster mit einer festen Größe s, welches Satz für Satz einer gegebenen Dokumentengesamtheit, für deren Wörter Einbettungen gelernt werden sollen, entlang läuft. Das Wort in der Mitte dieses Fensters bezeichnen wir als *Zentrumswort* und die s vorangegangenen und die s nachfolgenden Wörter als dessen *Nachbarschaftswörter* (oder *Kontextwörter*). Der Parameter s wird auch als *Window-Size* bezeichnet. Das Skip-Gram-Modell soll nun die Wahrscheinlichkeiten lernen, mit denen die Elemente des Vokabulars Kontextwörter eines gegebenen Zentrumswortes sind.

Ein Trainingsmuster für unser Modell ist ein Wortpaar, bestehend aus einem Zentrumswort, welches später als Input dient, und eines seiner Nachbarschaftswörter, das den Zielwert markiert. Die genaue Position des Zielwortes wird dabei nicht berücksichtigt. Es muss sich lediglich in der festgelegten Umgebung des jeweiligen Zentrumswortes befinden, wobei pro Zentrumswort maximal $2s$ Kontextwörter betrachtet werden können (vgl. Abbildung 3.1).

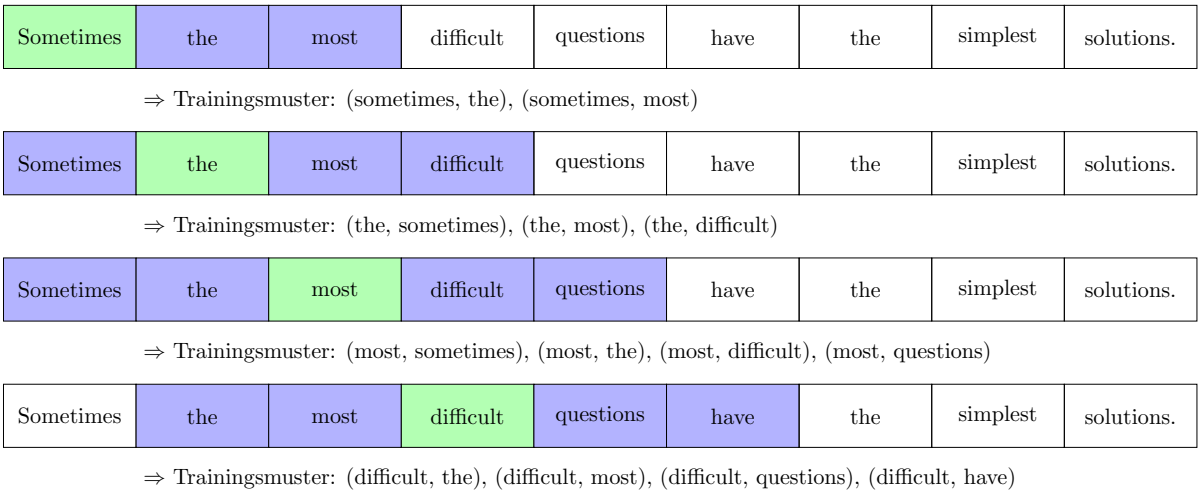


Abbildung 3.1: Trainingsmuster des Skip-Gram-Modells mit der Window-Size $s = 2$

Bemerkung 3.1. *Während beim Skip-Gram-Modell ein Zentrumswort genutzt wird, um nacheinander die einzelnen Nachbarschaftswörter zu prognostizieren, geht der CBOW-Ansatz genau den entgegengesetzten Weg. Hier werden dem Modell alle Nachbarschaftswörter auf einmal als Input präsentiert, mit deren Hilfe dann das richtige Zentrumswort zu ermitteln ist, wobei deren Reihenfolge dabei ebenfalls keine Rolle spielt.*

Nun wollen wir einige Bemerkungen zu der Struktur unseres verwendeten Neuronalen Netzes machen (vgl. Abbildung 3.2). Es handelt sich um ein *Feed-Forward-Netz*, welches eine einzige Hidden-Layer besitzt. Die Gewichtsmatrix zwischen der Input- und der Hidden-Layer wird als *Einbettungsmatrix* E bezeichnet. Die Gewichte zwischen der Hidden- und der Output-Layer entstammen der sogenannten *Kontextmatrix* C. Es gilt

$$E = \begin{bmatrix} \mathbf{e}_1^T \\ \mathbf{e}_2^T \\ \vdots \\ \mathbf{e}_{|\mathcal{V}|}^T \end{bmatrix} \in \mathbb{R}^{|\mathcal{V}| \times d} \quad \text{und} \quad C = \begin{bmatrix} \mathbf{c}_1^T \\ \mathbf{c}_2^T \\ \vdots \\ \mathbf{c}_{|\mathcal{V}|}^T \end{bmatrix} \in \mathbb{R}^{|\mathcal{V}| \times d} . \tag{3.1}$$

Jede Zeile der Matrizen E und C ist einem Element im Vokabular $\mathcal{V}$ zugeordnet, welches auf Grundlage des betrachteten Dokumentenkorpus zusammengestellt wird. Die einzelnen Zeilenvektoren der Matrix E entsprechen dabei den gesuchten Wortvektoren. Die Einbettung des Wortes $w \in \mathcal{V}$ ist also der korrespondierende d-dimensionale Vektor $\mathbf{e}_w$, wobei die Dimension d ein frei zu wählender Hyperparameter ist. Die Zeilenvektoren der Matrix C erfassen hingegen nicht die einzelnen Wörter, sondern den Kontext, in dem sie auftreten.

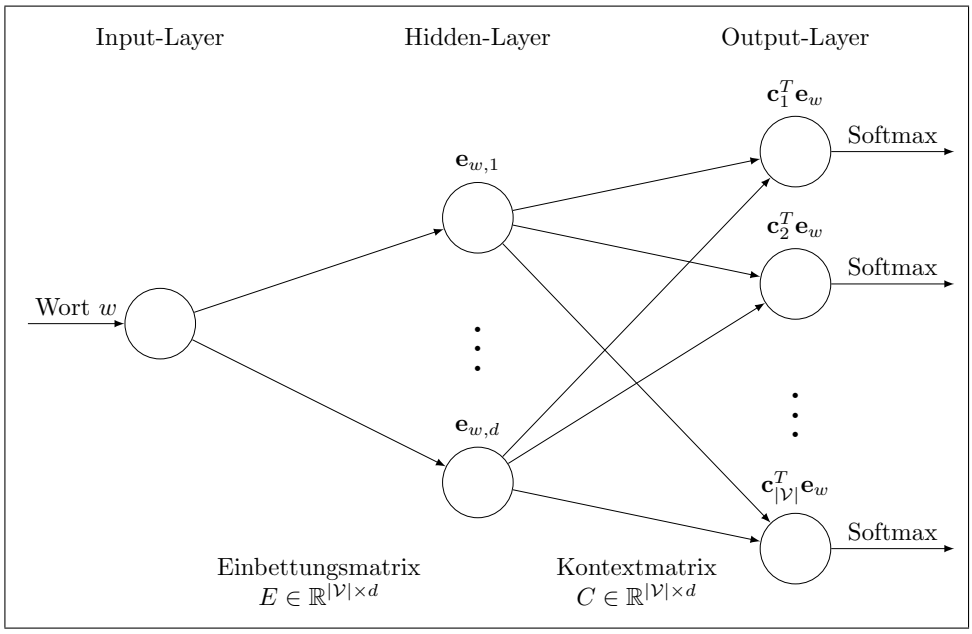


Abbildung 3.2: Aufbau des Skip-Gram-Modells

Als Trainingsgrundlage dient uns wieder eine Wortsequenz $w_1, \ldots, w_T$. Das Neuronale Netz erhält als Input das Zentrumswort w_t eines Trainingsmusters (w_t, w_{t+i}) mit $i \in \{-s, \ldots, s\}$ und $i \neq 0$, welches sich an t-ter Stelle der Sequenz befindet. Anschließend wird die zu w_t entsprechende Zeile $\mathbf{e}_{w_t}$ aus der Einbettungsmatrix, d.h. der Wortvektor des Inputs, gewählt, welche die Ausgabe der Hidden-Layer darstellt. Es wird der Einfachheit halber keine Aktivierungsfunktion in dieser Schicht verwendet. Zudem verzichten wir auch auf die Nutzung von Bias-Neuronen. Darauf wird

das Skalarprodukt zwischen $\mathbf{e}_{w_t}$ und allen Kontextvektoren berechnet. Der Output des Netzes bildet eine Verteilung über $\mathcal{V}$, die für jedes Wort die Wahrscheinlichkeiten angibt, dass gesuchte Nachbarschaftswort zu sein. Die bedingte Wahrscheinlichkeit des Zielwortes

$$p_{\boldsymbol{\theta}}\left(w_{t+i} \mid w_t\right) \rightarrow \max \tag{3.2}$$

soll dabei maximiert werden und wird mit Hilfe der Netzausgabe in Kombination mit der *Softmax-Funktion* berechnet

$$p_{\boldsymbol{\theta}}\left(w_{t+i} \mid w_t\right) = \frac{\exp\left(\mathbf{c}_{w_{t+i}}^T \mathbf{e}_{w_t}\right)}{\sum_{w_l \in \mathcal{V}} \exp\left(\mathbf{c}_{w_l}^T \mathbf{e}_{w_t}\right)} \ , \tag{3.3}$$

wobei die Menge der Modellparameter $\boldsymbol{\theta}$ in unserem Fall die Einträge der Matrizen E und C umfasst. Das Netz sorgt demnach dafür, dass die Wortdarstellung $\mathbf{e}_{w_t}$ auf das entsprechende Kontextwort in Form von $\mathbf{c}_{w_{t+i}}$ im Zähler der zu maximierenden Wahrscheinlichkeit trifft. Somit kann auch die Bezeichnung "Kontextmatrix" motiviert werden, da diese anscheinend den Kontext der einzelnen Wörter kodiert. Es ist zu beachten, dass erst die Nutzung der Softmax-Funktion dazu führt, dass der Output des Netzes als eine Wahrscheinlichkeitsverteilung interpretiert werden kann, da dann alle Ausdrücke zwischen 0 und 1 liegen und sich zu 1 aufsummieren. Es ist also nötig, die Modellausgaben explizit zu normalisieren.

Somit ergibt sich folgende *Softmax-Zielfunktion* des Skip-Gram-Modells über die Dokumentengesamtheit $w_1, \ldots, w_T$

$$\mathcal{J}_{\text{soft}}\left(w_1, \ldots, w_T, \boldsymbol{\theta}\right) \coloneqq -\frac{1}{T} \sum_{t=1}^{T} \sum_{-s \leq i \leq s,\ i \neq 0} \log p_{\boldsymbol{\theta}}\left(w_{t+i} | w_t\right) \rightarrow \min \ . \tag{3.4}$$

Im Unterschied zur Funktion (2.11) betrachten wir nun jedoch nicht mehr die Wahrscheinlichkeit für das Wort w_t gegeben die $n-1$ vorangegangenen Wörter, sondern wir ermitteln die Wahrscheinlichkeit eines Nachbarschaftswortes w_{t+i} gegeben w_t. Weiterhin summieren wir alle log-Wahrscheinlichkeiten der Elemente, die sich innerhalb der Window-Size um das Zentrumswort w_t befinden, auf. Mit Hilfe der Zielfunktion (3.4) kann nun nach einer zufälligen Initialisierung das Modell via (*stochastischem*) *Gradientenabstieg* bzw. *Backpropagation* trainiert werden (vgl. [22]).

Bemerkung 3.2. *Das Skip-Gram-Modell nutzt die vereinfachende Annahme, dass das Auftreten des Kontextwortes w_{t+i} in der Umgebung von w_t unabhängig ist von den restlichen Nachbarschaftswörtern von w_t, sodass gilt*

$$p_{\boldsymbol{\theta}}\left(w_{t-s}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+s} \mid w_t\right) = \prod_{-s \leq i \leq s,\ i \neq 0} p_{\boldsymbol{\theta}}\left(w_{t+i} | w_t\right) \ . \tag{3.5}$$

Die Fake-Task des Modells besteht darin, dass wir eigentlich Wortvorhersagen unter Verwendung bedingter Wahrscheinlichkeiten schätzen wollen, die gesuchten Wortvektoren jedoch durch die Gewichtsmatrix E des Systems gegeben sind.

Als abkürzende Schreibweise von (3.4) kann auch

$$\mathcal{J}_{\text{soft}}\left(w_1, \ldots, w_T, \boldsymbol{\theta}\right) = -\frac{1}{T} \sum_{(w_t, w_{t+i})} \log p_{\boldsymbol{\theta}}\left(w_{t+i} | w_t\right) \tag{3.6}$$

genutzt werden, wobei die Summe über alle Trainingsmuster (w_t, w_{t+i}), die die Dokumentengesamtheit unter Verwendung der Window-Size s enthält, läuft. Deren Zahl nimmt mit größerem s zu, sodass dem Netz auf der einen Seite zwar mehr Informationen in Form von mehr Wortpaaren bereitgestellt werden, was aber auf der anderen Seite zu einem erhöhten Rechenaufwand führt.

Bemerkung 3.3 (Wieso funktioniert Skip-Gram?)**.** *Wortvektor-Modelle sollen in erster Linie Wörter mit ähnlichen Bedeutungen auf ähnliche Einbettungen abbilden. Nach unserer Verteilungshypothese 2.3 haben aber ähnliche Wörter in der Regel auch einen identischen Kontext, in dem sie auftreten. Demnach muss unser System für solche Wörter auch identische Vorhersagen mittels der bedingten Wahrscheinlichkeiten treffen, was u.a. dadurch realisiert werden kann, indem ähnliche Wortvektoren gelernt werden.*

Bemerkung 3.4. *Obwohl unser Modell nur eine Hidden-Layer besitzt, kann es Probleme bzgl. des Rechenaufwandes geben. Es muss zur Bestimmung von (3.4) ständig der Normierungsausdruck im Nenner der Softmax (3.3) ermittelt werden. Da das Vokabular in der Regel sehr groß ist, kann dies sehr aufwändig sein.*

3.1 Kreuzentropie

Wir wollen in diesem Abschnitt erläutern, inwiefern der Ausdruck (3.4) mit der Minimierung der *Kreuzentropie* zwischen der Softmax-Verteilung $p_{\boldsymbol{\theta}}(w_{t+i} \mid w_t)$ und der tatsächlich vorliegenden bzw. empirischen Verteilung der Nachbarschaftswörter $\tilde{p}(w_{t+i} \mid w_t)$ zusammenhängt (vgl. [36], Abschnitt 3.1). Es sei an dieser Stelle angemerkt, dass nur endliche und diskrete Verteilungen betrachtet werden müssen, da das zugrundeliegende Vokabular stets endlich ist und dessen Elemente, d.h. die einzelnen Wörter, diskrete Objekte darstellen.

Definition 3.1 (Kreuzentropie)**.** *Es seien p_1 und p_2 zwei diskrete Wahrscheinlichkeitsverteilungen, die auf demselben Ereignisraum Ω definiert sind. Die Kreuzentropie zwischen p_1 und p_2 ist gegeben durch*

$$H(p_1, p_2) \coloneqq -\sum_{\omega \in \Omega} p_1(\omega) \cdot \log p_2(\omega) = \mathbb{E}_{\omega \sim p_1}\left[-\log p_2(\omega)\right] \ . \tag{3.7}$$

Die Kreuzentropie kann als ein Maß für die Unterschiedlichkeit zweier Verteilungen p_1 und p_2 interpretiert werden. Oft stellt p_2 eine Approximation einer unbekannten bzw. "wahren" Verteilung p_1 dar, sodass es nun das Ziel ist, $H(p_1, p_2)$ zu minimieren.

Die bereits erwähnte tatsächliche Verteilung $\tilde{p}$ der Nachbarschaftswörter resultiert aus dem zugrundeliegenden Text $w_1, \ldots, w_T$. Betrachten wir dazu das Zentrumswort w_t, welches im Dokumentenkorpus durchaus mehrere Male auftreten kann und insgesamt N_{w_t} Kontextwörter besitzt, wobei bei einer Window-Size von s die Abschätzung

$$N_{w_t} \leq 2s \cdot \operatorname{count}(w_t) \tag{3.8}$$

gilt. Weiterhin beschreibt $N_{w_t, w_{t+i}}$ die Anzahl, wie häufig das Wort $w_{t+i} \in \mathcal{V}$ in der Nachbarschaft des Zentrumswortes vorkommt. Dann ergibt sich für die bedingte empirische Wahrscheinlichkeit des Wortes w_{t+i} gegeben w_t

$$\tilde{p}(w_{t+i} \mid w_t) = \frac{N_{w_t, w_{t+i}}}{N_{w_t}} \ . \tag{3.9}$$

Bemerkung 3.5. *Diese tatsächliche Verteilung spiegelt umso besser die natürliche Verwendung von Wörtern einer Sprache wieder, je größer der Textkorpus ist, auf dem die Beobachtungen basieren, da dann möglichst viele Nutzungsszenarien der Wörter berücksichtigt werden können. Um also später gute Resultate zu erhalten, sollte eine hinreichend große Dokumentengesamtheit verarbeitet werden, da die Bedeutungen der Wörter auf deren Grundlage erfasst werden.*

Es ist zu beachten, dass hier das Kontextwort w_{t+i} jedes beliebige Element aus $\mathcal{V}$ sein kann und nicht zwangsläufig in der Umgebung der Länge s liegen muss. Tritt es nie in der Nachbarschaft von w_t auf, so beträgt die empirische Wahrscheinlichkeit (3.9) dementsprechend 0.

Lemma 3.1. *Für die Zielfunktion (3.4) erhalten wir*

$$\mathcal{J}_{\text{soft}}\left(w_1, \ldots, w_T, \boldsymbol{\theta}\right) = \frac{1}{T} \sum_{w_t \in \mathcal{V}} N_{w_t} \cdot H\left(\tilde{p}\left(. \mid w_t\right), p_{\boldsymbol{\theta}}\left(. \mid w_t\right)\right) \ . \tag{3.10}$$

Beweis.

- *mit Hilfe der abgekürzten Formulierung der Zielfunktion (3.6) ist einzusehen, dass einige Trainingsmuster in der Summe mehrfach auftreten können, sodass wir die Funktion zunächst umschreiben*

$$\begin{aligned}\mathcal{J}_{\text{soft}}\left(w_1, \ldots, w_T, \boldsymbol{\theta}\right) &= -\frac{1}{T} \sum_{(w_t, w_{t+i})} \log p_{\boldsymbol{\theta}}\left(w_{t+i} | w_t\right) \\ &= -\frac{1}{T} \sum_{w_t \in \mathcal{V}} \sum_{w_{t+i} \in \mathcal{V}} N_{w_t, w_{t+i}} \cdot \log p_{\boldsymbol{\theta}}\left(w_{t+i} | w_t\right)\end{aligned}$$

  und da nach Umstellung von (3.9) $N_{w_t, w_{t+i}} = \tilde{p}\left(w_{t+i} \mid w_t\right) \cdot N_{w_t}$ folgt, ergibt sich weiter

$$\begin{aligned}&= -\frac{1}{T} \sum_{w_t \in \mathcal{V}} N_{w_t} \sum_{w_{t+i} \in \mathcal{V}} \tilde{p}\left(w_{t+i} \mid w_t\right) \cdot \log p_{\boldsymbol{\theta}}\left(w_{t+i} | w_t\right) \\ &= \frac{1}{T} \sum_{w_t \in \mathcal{V}} N_{w_t} \cdot H\left(\tilde{p}\left(. \mid w_t\right), p_{\boldsymbol{\theta}}\left(. \mid w_t\right)\right)\end{aligned}$$

□

Bemerkung 3.6. *Die Minimierung der Zielfunktion (3.4) führt dazu, dass auch die Summe der (gewichteten) Kreuzentropien zwischen der Softmax-Verteilung, d.h. die durch das Modell modellierte bedingte Wahrscheinlichkeit, und der tatsächlich vorliegenden Verteilung für die einzelnen Zentrumswörter minimiert wird.*

3.2 Approximation der Softmax

Nun soll geklärt werden, wie das in Bemerkung 3.4 beschriebene Problem bzgl. des Rechenaufwandes gelöst werden kann. Im Laufe der letzten Jahre wurden verschiedene Methoden entwickelt, um die Softmax-Funktion zu approximieren (vgl. [39]). Diese Methoden können in zwei Gruppen eingeteilt werden: *Softmax-based-Ansätze* und *Sampling-based-Ansätze*.

Erstere halten an der Softmax-Funktion fest und versuchen, diese effizienter zu berechnen. Die Sampling-based-Ansätze konstruieren dagegen eine neue Zielfunktion, d.h. die Trainingsstrategie wird modifiziert, sodass dann z.B. die Summe zur Normierung der Wahrscheinlichkeiten im Nenner der Softmax durch einen einfacheren Ausdruck ersetzt wird. Diese Methoden sind aber in der Regel nur zur Zeit des Trainings hilfreich. Soll der Output des Netzes bei späteren Anwendungen als Wahrscheinlichkeitsverteilung interpretiert werden, ist wieder die ursprüngliche Softmax-Funktion zu berechnen. Für Wortvektor-Modelle stellt dies allerdings keine gravierende Einschränkung dar, da deren eigentliches Interesse in der Generierung von Einbettungen und nicht von Wahrscheinlichkeiten besteht.

3.2.1 Gradient der Softmax-Zielfunktion

Zunächst soll eine Vorstellung davon gegeben werden, wie groß der Einfluss der Softmax auf die Zielfunktion und die Trainingsphase des Systems ist (vgl. [6] und [43]). In Bezug auf das

vorangegangene Skip-Gram-Modell halten wir an den Bezeichnungen und deren Bedeutungen fest, obwohl auch eine Formulierung für einen allgemeineren Modellaufbau möglich ist.

Sei wieder (w_t, w_{t+i}) ein Trainingsmuster des Textes. Der Output unseres Systems ist dann durch (3.3) gegeben. Betrachten wir einfachheitshalber nur dieses eine Wortpaar und nicht die durchschnittliche negative log-Wahrscheinlichkeit über den gesamten Dokumentenkorpus, so vereinfacht sich die Zielfunktion (3.4) zu

$$J_{\text{soft}}\left(w_1, \ldots, w_T, \boldsymbol{\theta}\right) \coloneqq -\log p_{\boldsymbol{\theta}}\left(w_{t+i} \mid w_t\right) = -\log \frac{\exp\left(\mathbf{c}_{w_{t+i}}^T \mathbf{e}_{w_t}\right)}{\sum_{w_l \in \mathcal{V}} \exp\left(\mathbf{c}_{w_l}^T \mathbf{e}_{w_t}\right)} \ . \tag{3.11}$$

Um das Neuronale Netz mit Hilfe des Backpropagation Algorithmus trainieren zu können, benötigen wir den Gradienten der Zielfunktion bzgl. der Modellparameter $\boldsymbol{\theta}$ (= Matrizen E und C) des Neuronalen Netzes.

Lemma 3.2. *Für den Gradienten der Funktion (3.11) gilt*

$$\nabla_{\boldsymbol{\theta}} J_{\text{soft}}\left(w_1, \ldots, w_T, \boldsymbol{\theta}\right) = \nabla_{\boldsymbol{\theta}} \Gamma_{\boldsymbol{\theta}}\left(w_{t+i}\right) - \mathbb{E}_{w_l \sim p_{\boldsymbol{\theta}}}\left[\nabla_{\boldsymbol{\theta}} \Gamma_{\boldsymbol{\theta}}\left(w_l\right)\right] \ , \tag{3.12}$$

wobei die Notationskonvention

$$\Gamma_{\boldsymbol{\theta}}\left(w_j\right) \coloneqq -\boldsymbol{c}_{w_j}^T \boldsymbol{e}_{w_t} \tag{3.13}$$

getroffen wurde.

Beweis.

- *durch Anwendung der Notationskonvention erhalten wir*

$$J_{\text{soft}}\left(w_1, \ldots, w_T, \boldsymbol{\theta}\right) = -\log \frac{\exp\left(-\Gamma_{\boldsymbol{\theta}}\left(w_{t+i}\right)\right)}{\sum_{w_l \in \mathcal{V}} \exp\left(-\Gamma_{\boldsymbol{\theta}}\left(w_l\right)\right)}$$

- *zusammen mit*

$$\log(x)' = \frac{1}{x} \qquad \textit{und} \qquad \exp(x)' = \exp(x)$$

ergibt sich

$$\begin{aligned}
&\nabla_{\boldsymbol{\theta}} J_{\text{soft}}\left(w_1, \ldots, w_T, \boldsymbol{\theta}\right) \\
&= -\frac{\sum_{w_l \in \mathcal{V}} \exp\left(-\Gamma_{\boldsymbol{\theta}}\right)}{\exp\left(-\Gamma_{\boldsymbol{\theta}}\right)} \\
&\quad \cdot \frac{\exp\left(-\Gamma_{\boldsymbol{\theta}}\right) \nabla_{\boldsymbol{\theta}}\left(-\Gamma_{\boldsymbol{\theta}}\right) \cdot \sum_{w_l \in \mathcal{V}} \exp\left(-\Gamma_{\boldsymbol{\theta}}\right) - \exp\left(-\Gamma_{\boldsymbol{\theta}}\right) \cdot \sum_{w_l \in \mathcal{V}} \exp\left(-\Gamma_{\boldsymbol{\theta}}\right) \nabla_{\boldsymbol{\theta}}\left(-\Gamma_{\boldsymbol{\theta}}\right)}{\left(\sum_{w_l \in \mathcal{V}} \exp\left(-\Gamma_{\boldsymbol{\theta}}\right)\right)^2} \\
&= \nabla_{\boldsymbol{\theta}} \Gamma_{\boldsymbol{\theta}}\left(w_{t+i}\right) + \frac{1}{\sum_{w_l \in \mathcal{V}} \exp\left(-\Gamma_{\boldsymbol{\theta}}\left(w_l\right)\right)} \cdot \sum_{w_l \in \mathcal{V}} \exp\left(-\Gamma_{\boldsymbol{\theta}}\left(w_l\right)\right) \cdot \nabla_{\boldsymbol{\theta}}\left(-\Gamma_{\boldsymbol{\theta}}\left(w_l\right)\right) \\
&= \nabla_{\boldsymbol{\theta}} \Gamma_{\boldsymbol{\theta}}\left(w_{t+i}\right) + \sum_{w_l \in \mathcal{V}} \underbrace{\frac{\exp\left(-\Gamma_{\boldsymbol{\theta}}\left(w_l\right)\right)}{\sum_{w_h \in \mathcal{V}} \exp\left(-\Gamma_{\boldsymbol{\theta}}\left(w_h\right)\right)}}_{p_{\boldsymbol{\theta}}(w_l \mid w_t)} \cdot \nabla_{\boldsymbol{\theta}}\left(-\Gamma_{\boldsymbol{\theta}}\left(w_l\right)\right) \ ,
\end{aligned}$$

wobei im ersten Zwischenschritt zur kürzeren Darstellung auf die Funktionsargumente verzichtet wurde

- *der Summenausdruck über das gesamte Vokabular stellt den Erwartungswert des Gradienten von* $-\Gamma_{\boldsymbol{\theta}}(w_l)$ *dar, was schlussendlich die Behauptung liefert*

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} J_{\text{soft}}(w_1, \ldots, w_T, \boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \Gamma_{\boldsymbol{\theta}}(w_{t+i}) + \sum_{w_l \in \mathcal{V}} p_{\boldsymbol{\theta}}(w_l \mid w_t) \cdot \nabla_{\boldsymbol{\theta}}(-\Gamma_{\boldsymbol{\theta}}(w_l)) \\ &= \nabla_{\boldsymbol{\theta}} \Gamma_{\boldsymbol{\theta}}(w_{t+i}) + \mathbb{E}_{w_l \sim p_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}}(-\Gamma_{\boldsymbol{\theta}}(w_l))] \\ &= \nabla_{\boldsymbol{\theta}} \Gamma_{\boldsymbol{\theta}}(w_{t+i}) - \mathbb{E}_{w_l \sim p_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}} \Gamma_{\boldsymbol{\theta}}(w_l)]\end{aligned}$$

□

Der Gradient besteht folglich aus zwei Summanden. Der erste Summand soll die Parametermenge dahingehend verändern, dass die Modellwahrscheinlichkeit des gesuchten Zielwortes erhöht wird, während der Erwartungswert die Wahrscheinlichkeiten der übrigen Wörter im Vokabular dementsprechend verringern soll. Es gibt verschiedene Sampling-based-Ansätze, die nun versuchen, den Erwartungswertausdruck des Gradienten zu approximieren, um nicht über das gesamte Vokabular summieren zu müssen.

Eine Möglichkeit, um $\mathbb{E}_{w_l \sim p_{\boldsymbol{\theta}}}$ näherungsweise zu bestimmen, besteht in der Nutzung des Stichprobenmittels als erwartungstreuen Schätzer, d.h. wir bilden den Durchschnitt von k zufällig gewählten Beobachtungen, die der diskreten Verteilung $p_{\boldsymbol{\theta}}$ folgen

$$\mathbb{E}_{w_l \sim p_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}} \Gamma_{\boldsymbol{\theta}}(w_l)] \approx \frac{1}{k} \sum_{w_l \sim p_{\boldsymbol{\theta}}}^{k} \nabla_{\boldsymbol{\theta}} \Gamma_{\boldsymbol{\theta}}(w_l) \ , \tag{3.14}$$

wobei die Approximation umso besser wird, je größer der Stichprobenumfang k ist. Durch dieses Vorgehen wird der Gradient zwar verrauscht, da aber oftmals die Modelle unter Nutzung des stochastischen Gradientenabstiegs, der ebenfalls bloß Schätzungen des lokalen Gradienten berechnet, trainiert werden, stellt dies keine grundsätzliche Einschränkung dar.

Um allerdings Wörter entsprechend der Verteilung $p_{\boldsymbol{\theta}}$ zu generieren, muss diese bekannt sein. Dazu müsste der Nenner der Softmax-Funktion berechnet werden, was wir aber gerade umgehen möchten. Deshalb nutzen wir zur Vereinfachung eine "Noise-Verteilung" q. Das *Importance-Sampling* (vgl. [6]) ist eine Methode, die versucht, eine möglichst passende und kostengünstig zu verwendende Noise-Verteilung zu berechnen. Jedoch kann die Varianz der Approximation des Erwartungswertes sehr hoch sein, da mit zunehmender Trainingszeit die Verteilungen $p_{\boldsymbol{\theta}}$ und q immer stärker voneinander abweichen, was wiederum das Training instabil macht (vgl. [35]). Zudem muss beim Importance-Sampling die vollständige Softmax-Funktion berechnet werden, wenn später die Outputs des trainierten Modells für entsprechende Anwendungen als Wahrscheinlichkeiten interpretiert werden sollen. Aufgrund dieser Probleme betrachten wir nun ein weiteres Verfahren.

3.2.2 Noise-Contrastive Estimation

Wir wollen uns im Folgenden mit einer weiteren Sampling-based-Methode beschäftigen, die in der Vergangenheit erfolgreich zum Training von Wortvektor- und Sprachmodellen eingesetzt wurde. Die sogenannte *Noise-Contrastive Estimation* (*NCE*) (vgl. [11], [18] und [35]) nutzt nicht mehr den standardmäßigen Ansatz der Maximum-Likelihood-Methode, den wir bisher verfolgt haben und welcher die Softmax-Zielfunktion (3.4) lieferte, sondern sie stellt ein Verfahren dar, bei dem unser Skip-Gram-Modell einen probabilistischen binären Klassifikator modelliert. Dieser soll zwischen "richtig" generierten Nachbarschaftswörtern auf Basis der zugrundeliegenden Textgesamtheit und Noise-Wörtern unterscheiden. Wir werden zeigen, dass der Gradient der Zielfunktion dieses Ansatzes gegen unseren ursprünglichen Gradienten (3.12) konvergiert (vgl. Abschnitt 3.2.5).

Es sei wieder $\tilde{p}(w \mid w_t)$ die empirische Verteilung, die angibt, wie groß die Wahrscheinlichkeit für das Auftreten des Wortes w in der Nachbarschaft eines gegebenen Zentrumswortes w_t ist. Die ursprüngliche Aufgabe bestand darin, die Modellparameter $\boldsymbol{\theta}$ so zu wählen, sodass die Softmax-Verteilung $p_{\boldsymbol{\theta}}(w \mid w_t)$ die tatsächlich vorliegende Verteilung im Sinne einer minimalen Kreuzentropie gut approximiert und die Wahrscheinlichkeit des jeweiligen Zielwortes maximiert wird. Die NCE schätzt nun keine Verteilung mehr, sondern ermittelt einen Klassifikator, der zwischen Mustern unterscheidet, die durch die tatsächliche und durch eine Noise-Verteilung $q(w)$ erzeugt worden sind (vgl. Abbildung 3.3), für die z.B. die Gleichverteilung über das gesamte Vokabular gewählt werden kann, sodass es nun relativ günstig ist, Wörter entsprechen q zu generieren.

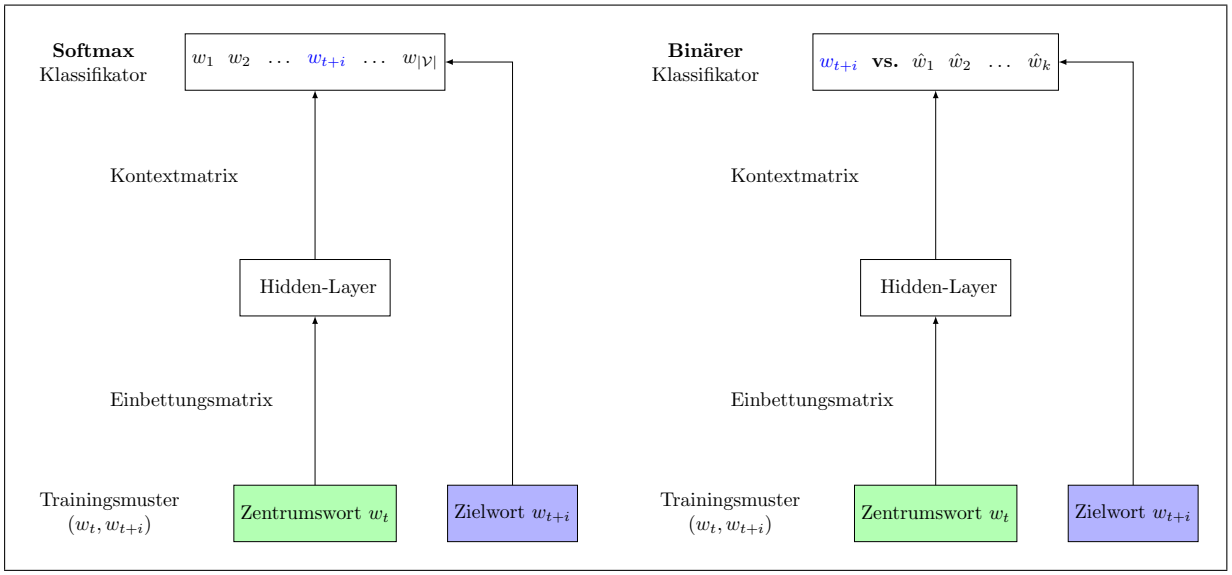


Abbildung 3.3: Vergleich des Softmax- und des binären Klassifikators

Die Trainingsdaten seien theoretisch wie folgt zusammengestellt: Zunächst wählen wir ein Zentrumswort w_t der Dokumentengesamtheit $w_1, \ldots, w_T$ entsprechend seiner relativen Häufigkeit, die wir mit

$$\tilde{p}(w_t) = \frac{\text{count}(w_t)}{T} \tag{3.15}$$

bezeichnen wollen, und darauf ein richtiges Nachbarschaftswort w_{t+i} mit Hilfe von $\tilde{p}(w \mid w_t)$ aus. Dieses Wortpaar wird mit dem Label $y = 1$ versehen, sodass erkennbar ist, dass es von der empirischen Verteilung bzw. vom betrachteten Text abstammt. Anschließend werden dazu k Noise-Wörter $\hat{w}$ entsprechend $q(w)$ generiert und mit $y = 0$ markiert. Dabei wurde die Annahme getroffen, dass das Verhältnis zwischen den positiven und den negativen Klassifikationsmustern nicht ausgeglichen ist und die Noise-Wörter mit dem Faktor k häufiger auftreten als die richtigen Elemente.

Bemerkung 3.7. *Zur Laufzeit des Modells wird der Textkorpus wieder Wort für Wort durchlaufen, wobei wir die richtigen Trainingsmuster einfach dadurch erhalten, indem die Nachbarschaftswörter w_{t+i} des aktuellen Zentrumswortes w_t ausgewählt werden.*

Wir haben nun ein binäres Klassifikationsproblem, bei dem das System zuverlässig erkennen soll, dass es sich bei (w_t, w_{t+i}) um ein Wortpaar handelt, dass auf Grundlage des beobachteten Textes generiert wurde und bei $(w_t, \hat{w})$ um ein Trainingsmuster, bei dem dies nicht so ist. Es wird dazu folgendes Produkt von bedingten Wahrscheinlichkeiten mit einem wahren und k Noise-Mustern

maximiert

$$p\left(y=1 \mid w_{t+i}, w_t\right) \cdot \prod_{\hat{w} \sim q}^{k} p\left(y=0 \mid \hat{w}, w_t\right) \rightarrow \max \,, \tag{3.16}$$

wobei $p\left(y=1 \mid w_{t+i}, w_t\right)$ die Wahrscheinlichkeit angibt, dass das gegebene Wortpaar von der beobachteten Dokumentengesamtheit abstammt und $p\left(y=0 \mid \hat{w}, w_t\right) = 1 - p\left(y=1 \mid \hat{w}, w_t\right)$ ist daher diejenige Wahrscheinlichkeit, dass das nicht der Fall ist. Durch Anwendung des negativen Logarithmus folgt die Minimierungsaufgabe

$$\begin{aligned} & -\log\left(p\left(y=1 \mid w_{t+i}, w_t\right) \cdot \prod_{\hat{w} \sim q}^{k} p\left(y=0 \mid \hat{w}, w_t\right)\right) \\ & = -\left(\log p\left(y=1 \mid w_{t+i}, w_t\right) + \sum_{\hat{w} \sim q}^{k} \log p\left(y=0 \mid \hat{w}, w_t\right)\right) \rightarrow \min . \end{aligned} \tag{3.17}$$

Die Modellparameter, durch deren Veränderung dieser Ausdruck minimiert werden soll, werden wir erst später einfügen. Es sei angemerkt, dass auch durchaus richtige Nachbarschaftswörter mit der Verteilung q erzeugt werden können, was allerdings aufgrund der Größe des Vokabulars in der Regel sehr unwahrscheinlich ist.

Betrachten wir nun wieder unsere gegebene Dokumentengesamtheit und summieren über alle vorliegenden Trainingsmuster (w_t, w_{t+i}), so liefert dies schließlich die Zielfunktion des NCE-Ansatzes

$$\begin{aligned} & \mathcal{J}_{\mathrm{NCE}}\left(w_1, \ldots, w_T, q, k\right) \\ & := -\sum_{\left(w_t, w_{t+i}\right)} \left(\log p\left(y=1 \mid w_{t+i}, w_t\right) + \sum_{\hat{w} \sim q}^{k} \log p\left(y=0 \mid \hat{w}, w_t\right)\right) \rightarrow \min \,. \end{aligned} \tag{3.18}$$

Bisher wurde die Zielfunktion dadurch abgeändert, indem Noise-Trainingsmuster hinzugefügt wurden und das System diese auch als solche identifizieren soll. Durch diese Unterscheidung ist es in der Lage, Eigenschaften und Strukturen der Trainingsdaten zu erfassen. Nun stellt sich jedoch die Frage, wie die in (3.18) enthaltenen bedingten Wahrscheinlichkeiten zu berechnen sind.

Satz 3.1. *Seien A, B und B^c drei Ereignisse, wobei B^c das Komplement von B darstellt und $p(A) > 0$, dann folgt aus der Definition (2.4) der bedingten Wahrscheinlichkeit der Satz von der totalen Wahrscheinlichkeit*

$$\begin{aligned} p(A) &= p(A, B) + p\left(A, B^c\right) \\ &= p(A \mid B) \cdot p(B) + p\left(A \mid B^c\right) \cdot p\left(B^c\right) \end{aligned} \tag{3.19}$$

und damit die Bayes-Formel

$$\begin{aligned} p(B \mid A) = \frac{p(B, A)}{p(A)} &= \frac{\frac{p(B,A)}{p(B)} \cdot p(B)}{p(A)} = \frac{p(A \mid B) \cdot p(B)}{p(A)} \\ &= \frac{p(A \mid B) \cdot p(B)}{p(A \mid B) \cdot p(B) + p\left(A \mid B^c\right) \cdot p\left(B^c\right)} \,. \end{aligned} \tag{3.20}$$

Mit Hilfe der Bayes-Formel können wir die Wahrscheinlichkeit $p\left(y=1 \mid w_{t+i}, w_t\right)$ umformulieren

$$p\left(y=1 \mid w_{t+i}, w_t\right) = \frac{p\left(w_{t+i}, w_t \mid y=1\right) \cdot p(y=1)}{p\left(w_{t+i}, w_t \mid y=1\right) \cdot p(y=1) + p\left(w_{t+i}, w_t \mid y=0\right) \cdot p(y=0)} \,. \tag{3.21}$$

Da wir insgesamt $k+1$ Muster pro gegebenen Zentrumswort w_t verarbeiten (ein richtiges Wort und k Noise-Wörter), ergibt sich

$$p(y=1) = \frac{1}{k+1} \qquad \text{und} \qquad p(y=0) = \frac{k}{k+1} \tag{3.22}$$

und aufgrund der Wahl der einzelnen Trainingsdaten folgt

$$p(w_{t+i}, w_t \mid y=1) = \tilde{p}(w_t) \cdot \tilde{p}(w_{t+i} \mid w_t) \tag{3.23}$$

und

$$p(w_{t+i}, w_t \mid y=0) \approx \tilde{p}(w_t) \cdot q(w_{t+i}) \ , \tag{3.24}$$

wobei wir die Näherung (3.24) im Folgenden als Gleichheit auffassen wollen. Es handelt sich lediglich um eine Näherung (Rauschen), da wir zum einen stochastische Unabhängigkeit zwischen der Auswahl von w_t und w_{t+i} voraussetzen und zum anderen ist die Wahl der Noise-Verteilung q völlig beliebig. Die Unabhängigkeit ist dadurch zu rechtfertigen, weil die Noise-Muster keine Wortpaare sein sollen, die in Texten tatsächlich auftreten, sodass diese unrealistische Annahme im Idealfall die Generierung solcher Paare befördert. Werden nun (3.23) und (3.24) in die Formel (3.21) eingesetzt, so erhalten wir schließlich durch Kürzen der gemeinsamen Faktoren

$$p(y=1 \mid w_{t+i}, w_t) = \frac{\tilde{p}(w_{t+i} \mid w_t)}{k \cdot q(w_{t+i}) + \tilde{p}(w_{t+i} \mid w_t)} \tag{3.25}$$

und

$$\begin{aligned} p(y=0 \mid \hat{w}, w_t) &= 1 - p(y=1 \mid \hat{w}, w_t) \\ &= \frac{k \cdot q(\hat{w})}{k \cdot q(\hat{w}) + \tilde{p}(\hat{w} \mid w_t)} \ . \end{aligned} \tag{3.26}$$

Im Zuge der NCE sollen die in den Ausdrücken (3.25) und (3.26) genutzten empirischen Verteilungen $\tilde{p}(w \mid w_t)$ durch die von einem zu trainierenden Modell berechneten Wahrscheinlichkeiten $p_{\boldsymbol{\theta}}(w \mid w_t)$ approximiert werden. Dieses ist in unserem Fall das Skip-Gram-Modell, welches von der Parametermenge $\boldsymbol{\theta}$ abhängt. Werden (3.25) und (3.26) mit dieser Substitution in (3.18) eingesetzt, so liefert dies schlussendlich

$$\begin{aligned} &\mathcal{J}_{\text{NCE}}(w_1, \ldots, w_T, \boldsymbol{\theta}, q, k) \\ &= -\sum_{(w_t, w_{t+i})} \left(\log \frac{p_{\boldsymbol{\theta}}(w_{t+i} \mid w_t)}{k \cdot q(w_{t+i}) + p_{\boldsymbol{\theta}}(w_{t+i} \mid w_t)} + \sum_{\hat{w} \sim q}^{k} \log \frac{k \cdot q(\hat{w})}{k \cdot q(\hat{w}) + p_{\boldsymbol{\theta}}(\hat{w} \mid w_t)} \right) \ . \end{aligned} \tag{3.27}$$

Bis jetzt haben wir allerdings keinerlei Probleme bzgl. des Rechenaufwandes gelöst, da auch hier die Modellwahrscheinlichkeiten (3.3) zum Einsatz kommen und deshalb Summationen über das gesamte Vokabular nötig sind. Wir werden in den drei folgenden Abschnitten jedoch mit Hilfe weiterer Überlegungen sehen, dass die NCE einen geringeren Rechenaufwand und ein wünschenswertes Verhalten im Grenzfall $k \to \infty$ besitzt.

3.2.3 Selbst-Normalisierung der NCE

Zunächst wollen wir den großen Vorteil, den uns die NCE in Bezug auf die Rechenzeit liefert, beschreiben. Sie stellt ein Verfahren dar, welches Modelle anpasst, die nicht explizit normalisiert wurden, sodass diese darauf ausgerichtet werden, im Laufe des Trainingsverfahrens *selbstnormalisierende* Modellausgaben zu generieren. Dies bedeutet wiederum, dass die Summe über

all diese Werte approximativ 1 ist. Ist diese Näherung hinreichend exakt, kann demnach auf die Berechnung eines Normalisierungsterms während des Trainings und auch darüber hinaus (z.B. bei Evaluierungen und Tests) verzichtet werden.

Fassen wir die Werte

$$\hat{p}_{\boldsymbol{\theta}}\left(w \mid w_t\right) \coloneqq \exp\left(\mathbf{c}_w^T \mathbf{e}_{w_t}\right) \tag{3.28}$$

als nicht-normalisierte Ausgaben unseres zugrundeliegenden Skip-Gram-Modells auf. Diese sind zwar immer positiv, da sie sich aber für eine beliebige Parameterkonstellation $\boldsymbol{\theta}$ in der Regel nicht zu 1 aufsummieren, müssen diese beim klassischen Maximum-Likelihood-Ansatz, der die Softmax-Zielfunktion nutzt, explizit normalisiert werden, um sie als Wahrscheinlichkeiten interpretieren zu können

$$p_{\boldsymbol{\theta}}\left(w \mid w_t\right) = \frac{\exp\left(\mathbf{c}_w^T \mathbf{e}_{w_t}\right)}{\sum_{w_l \in \mathcal{V}} \exp\left(\mathbf{c}_{w_l}^T \mathbf{e}_{w_t}\right)} = \frac{\hat{p}_{\boldsymbol{\theta}}\left(w \mid w_t\right)}{\sum_{w_l \in \mathcal{V}} \hat{p}_{\boldsymbol{\theta}}\left(w_l \mid w_t\right)} =: \frac{\hat{p}_{\boldsymbol{\theta}}\left(w \mid w_t\right)}{\mathcal{Z}_{\boldsymbol{\theta}}\left(w_t\right)} \ . \tag{3.29}$$

Die Berechnung des Nenners $\mathcal{Z}_{\boldsymbol{\theta}}\left(w_t\right)$ für jedes Zentrumswort w_t soll jedoch vermieden werden. Daher wird dieser bei der originalen NCE als neuer, separater Modellparameter $\mathcal{Z}\left(.\right)$ in Abhängigkeit des entsprechenden Zentrumswortes aufgefasst, der den Nenner im Laufe des Optimierungsprozesses lernen soll und somit für die Normalisierung sorgt (vgl. [18]). Das System ist dann durch die nicht-normalisierten "Wahrscheinlichkeiten" $\hat{p}_{\boldsymbol{\theta}}\left(. \mid .\right)$, die durch $\boldsymbol{\theta}$ bestimmt werden, und durch $\mathcal{Z}\left(.\right)$ charakterisiert.

In [35] wurde empirisch nachgewiesen, dass die Ergebnisse von Sprachmodellen nicht darunter leiden, wenn die Normalisierungsterme einfach konstant auf 1 gesetzt werden, anstatt diese vom Modell lernen zu lassen. Weitere Experimente wie in [44] haben darüber hinaus aufzeigen können, dass bei einer Fixierung von $\mathcal{Z}$ eine Selbst-Normalisierung im Laufe des Trainings stattfindet, da der tatsächliche Nenner dann im Durchschnitt einen Wert von 1 bei geringer Varianz besitzt. Wir wollen nun dieses Phänomen angelehnt an [17] analytisch nachweisen und fassen zu diesem Zweck (3.27) als Funktion in Abhängigkeit der nicht-normalisierten Wahrscheinlichkeiten $\hat{p}_{\boldsymbol{\theta}}$ auf

$$\begin{aligned} &\hat{\mathcal{J}}_{\mathrm{NCE}}\left(\hat{p}_{\boldsymbol{\theta}}\left(. \mid .\right)\right) \\ &:= -\sum_{\left(w_t, w_{t+i}\right)} \left(\log \frac{\hat{p}_{\boldsymbol{\theta}}\left(w_{t+i} \mid w_t\right)}{k \cdot q\left(w_{t+i}\right) + \hat{p}_{\boldsymbol{\theta}}\left(w_{t+i} \mid w_t\right)} + \sum_{\hat{w} \sim q}^{k} \log \frac{k \cdot q\left(\hat{w}\right)}{k \cdot q\left(\hat{w}\right) + \hat{p}_{\boldsymbol{\theta}}\left(\hat{w} \mid w_t\right)} \right) \ . \end{aligned} \tag{3.30}$$

Insbesondere fixieren wir also die Nenner zur Normalisierung auf 1.

Unser Ziel ist nach wie vor die Minimierung dieses Ausdrucks. Mit Hilfe des folgenden Satzes werden wir sehen, dass im Zuge des Optimierungsprozesses die nicht-normalisierten Modellausgaben in der Tat gegen die zu approximierende, tatsächlich vorliegende Wahrscheinlichkeit $\tilde{p}$ konvergiert. Zur Erinnerung: Es gilt

$$\tilde{p}\left(w \mid w_t\right) = \frac{N_{w_t, w}}{N_{w_t}} \ , \tag{3.31}$$

wobei N_{w_t} die Anzahl aller Nachbarschaftswörter beschreibt, die w_t in der Textgesamtheit besitzt und $N_{w_t, w}$ ist die Anzahl, wie häufig w in der Umgebung von w_t auftritt.

Satz 3.2. *Unter der Bedingung*

$$q\left(w\right) \neq 0 \qquad \forall w \in \mathcal{V} \tag{3.32}$$

besitzt die Funktion (3.30) für den Fall einer hinreichend großen Anzahl an Noise-Mustern ein eindeutig bestimmtes Minimum und sie nimmt dieses an, falls für jedes Zentrumswort w_t gilt

$$\hat{p}_{\boldsymbol{\theta}}\left(w \mid w_t\right) = \tilde{p}\left(w \mid w_t\right) \qquad \forall w \in \mathcal{V} \ . \tag{3.33}$$

Beweis.

- *da das Strichprobenmittel ein erwartungstreuer Schätzer für den Erwartungswert ist, können wir die Funktion (3.30) für hinreichend große k approximieren durch*

$$
\begin{aligned}
&- \sum_{(w_t, w_{t+i})} \left(\log \frac{\hat{p}_{\boldsymbol{\theta}}\left(w_{t+i} \mid w_t\right)}{k \cdot q\left(w_{t+i}\right) + \hat{p}_{\boldsymbol{\theta}}\left(w_{t+i} \mid w_t\right)} + k \cdot \mathbb{E}_{\hat{w} \sim q} \left[\log \frac{k \cdot q\left(\hat{w}\right)}{k \cdot q\left(\hat{w}\right) + \hat{p}_{\boldsymbol{\theta}}\left(\hat{w} \mid w_t\right)} \right] \right) \\
&= - \sum_{(w_t, w_{t+i})} \left(\log \frac{\hat{p}_{\boldsymbol{\theta}}\left(w_{t+i} \mid w_t\right)}{k \cdot q\left(w_{t+i}\right) + \hat{p}_{\boldsymbol{\theta}}\left(w_{t+i} \mid w_t\right)} + k \cdot \sum_{\hat{w} \in \mathcal{V}} q\left(\hat{w}\right) \cdot \log \frac{k \cdot q\left(\hat{w}\right)}{k \cdot q\left(\hat{w}\right) + \hat{p}_{\boldsymbol{\theta}}\left(\hat{w} \mid w_t\right)} \right) \\
&=: \mathcal{J}\left(\hat{p}_{\boldsymbol{\theta}}\right)
\end{aligned}
$$

- *zum Zwecke der Optimierung betrachten wir ein beliebiges aber festes Wortpaar (w_Z, w_N), bestehend aus dem Nachbarschaftswort w_N und dem Zentrumswort w_Z*

- *leiten wir den Ausdruck $\mathcal{J}\left(\hat{p}_{\boldsymbol{\theta}}\right)$ nun nach $\hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)$ ab, so können aufgrund der beiden auftretenden Summenausdrücke, wobei die äußere Summe über alle vorhandenen Trainingspaare (w_t, w_{t+i}) läuft, folgende drei Fälle unterschieden werden*

- ***1. Fall:** $w_t = w_Z$ und $w_{t+i} = w_N$: dieser Fall tritt insgesamt N_{w_Z, w_N}-mal auf*

$$
\begin{aligned}
&\frac{\partial}{\partial \hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)} \\
&= -N_{w_Z, w_N} \cdot \left(\frac{k \cdot q\left(w_N\right) + \hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)}{\hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)} \cdot \frac{k \cdot q\left(w_N\right) + \hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right) - \hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)}{\left(k \cdot q\left(w_N\right) + \hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)\right)^2} \right. \\
&\quad \left. + k \cdot q\left(w_N\right) \cdot \frac{k \cdot q\left(w_N\right) + \hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)}{k \cdot q\left(w_N\right)} \cdot \frac{-k \cdot q\left(w_N\right)}{\left(k \cdot q\left(w_N\right) + \hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)\right)^2} \right) \\
&= -N_{w_Z, w_N} \cdot \frac{k \cdot q\left(w_N\right)}{k \cdot q\left(w_N\right) + \hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)} \cdot \left(\frac{1}{\hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)} - 1 \right)
\end{aligned}
$$

- ***2. Fall:** $w_t = w_Z$ und $w_{t+i} \neq w_N$: dieser Fall tritt insgesamt $(N_{w_Z} - N_{w_Z, w_N})$-mal auf*

$$
\begin{aligned}
&\frac{\partial}{\partial \hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)} \\
&= -\left(N_{w_Z} - N_{w_Z, w_N}\right) \cdot \left(k \cdot q\left(w_N\right) \cdot \frac{k \cdot q\left(w_N\right) + \hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)}{k \cdot q\left(w_N\right)} \cdot \frac{-k \cdot q\left(w_N\right)}{\left(k \cdot q\left(w_N\right) + \hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)\right)^2} \right) \\
&= -\left(N_{w_Z} - N_{w_Z, w_N}\right) \cdot \frac{k \cdot q\left(w_N\right)}{k \cdot q\left(w_N\right) + \hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)} \cdot (-1)
\end{aligned}
$$

- ***3. Fall:** $w_t \neq w_Z$ und $w_{t+i} = w_N$ oder $w_{t+i} \neq w_N$:*

$$
\frac{\partial}{\partial \hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)} = 0
$$

- *führen wir diese drei Fälle zusammen, so ergibt sich für die Ableitung von* $\mathcal{J}\left(\hat{p}_{\boldsymbol{\theta}}\right)$ *schließlich*

$$\begin{aligned}
&\frac{\partial \mathcal{J}\left(\hat{p}_{\boldsymbol{\theta}}\right)}{\partial \hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)} \\
&= -\frac{k \cdot q\left(w_N\right)}{k \cdot q\left(w_N\right) + \hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)} \cdot \left(N_{w_Z, w_N} \cdot \left(\frac{1}{\hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)} - 1 \right) + \left(N_{w_Z} - N_{w_Z, w_N}\right) \cdot (-1) \right) \\
&= -\frac{k \cdot q\left(w_N\right)}{k \cdot q\left(w_N\right) + \hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)} \cdot \left(\frac{N_{w_Z, w_N}}{\hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)} - N_{w_Z} \right) \qquad (3.34) \\
&\stackrel{!}{=} 0
\end{aligned}$$

- *das Produkt (3.34) ist* 0*, falls einer der beiden Faktoren* 0 *ist; unter der Bedingung (3.32) ist der erste Faktor immer ungleich* 0 *und daher analysieren wir den zweiten Faktor*

$$\frac{N_{w_Z, w_N}}{\hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)} - N_{w_Z} = 0 \qquad \Leftrightarrow \qquad \hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right) = \frac{N_{w_Z, w_N}}{N_{w_Z}} = \tilde{p}\left(w_N \mid w_Z\right) \ ,$$

  sodass in diesem Fall sogar ein eindeutig bestimmtes Minimum vorliegt

- *um die Behauptung final zu beweisen, muss noch die zweite Ableitung berechnet werden*

$$\begin{aligned}
\frac{\partial^2 \mathcal{J}\left(\hat{p}_{\boldsymbol{\theta}}\right)}{\partial \hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)^2} &= \frac{\partial}{\partial \hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)} \left(-\frac{k \cdot q\left(w_N\right)}{k \cdot q\left(w_N\right) + \hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)} \cdot \left(\frac{N_{w_Z, w_N}}{\hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)} - N_{w_Z} \right) \right) \\
&= \frac{k \cdot q\left(w_N\right)}{\left(k \cdot q\left(w_N\right) + \hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)\right)^2} \cdot \left(\frac{N_{w_Z, w_N}}{\hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)} - N_{w_Z} \right) \\
&\quad - \frac{k \cdot q\left(w_N\right)}{k \cdot q\left(w_N\right) + \hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)} \cdot \left(-\frac{N_{w_Z, w_N}}{\hat{p}_{\boldsymbol{\theta}}\left(w_N \mid w_Z\right)^2} \right)
\end{aligned}$$

- *setzen wir nun die empirische Verteilung* $\tilde{p}$ *in diesen Ausdruck ein, so verschwindet der erste Summand und zusammen mit (3.32) ist die hinreichende Bedingung an ein Minimum erfüllt*

$$\underbrace{\frac{k \cdot q\left(w_N\right)}{k \cdot q\left(w_N\right) + \tilde{p}\left(w_N \mid w_Z\right)}}_{>0} \cdot \underbrace{\frac{N_{w_Z, w_N}}{\tilde{p}\left(w_N \mid w_Z\right)^2}}_{>0} > 0 \qquad (3.35)$$

- *eigentlich müsste bei der zweiten Ableitung eine Hesse-Matrix ausgewertet werden, da es viele unterschiedliche Wortpaare* (w_Z, w_N) *gibt, die in* $\hat{p}_{\boldsymbol{\theta}}\left(. \mid .\right)$ *eingesetzt werden können; diese ist aber hier eine Diagonalmatrix, sodass wir uns auf einzelne eindimensionale Probleme zurückziehen können*

- *das letzte noch zu lösenden Problem entsteht in (3.35) für ein Muster* $(\hat{w}_Z, \hat{w}_N)$*, welches durch die Noise-Verteilung erzeugt wurde und somit in der Zielfunktion auftritt, aber nicht im Text vorkommt, d.h.* $\tilde{p}\left(\hat{w}_Z \mid \hat{w}_N\right) = 0$ *(Division durch* 0 *nicht möglich)*

- *ein solches Paar tritt nur in der inneren Summe von (3.30) auf, deren Summanden den Minimalwert* 0 *besitzen, welcher gerade für*

$$\hat{p}_{\boldsymbol{\theta}}\left(\hat{w}_Z \mid \hat{w}_N\right) = \tilde{p}\left(\hat{w}_Z \mid \hat{w}_N\right) = 0$$

  angenommen wird, sodass es sich auch hier bei der empirischen Verteilung in der Tat um die gesuchte Minimalstelle handelt □

Bemerkung 3.8. *Die Bedingung (3.32) an die Noise-Verteilung ist nicht sehr einschränkend. Wird für q z.B. die Gleichverteilung über $\mathcal{V}$ oder die Unigram-Verteilung der einzelnen Wörter gewählt, so ist sie offenbar erfüllt.*

Die bestimmte Minimalstelle ist in unserem Modell ein Infimum, gegen das die Parameter während des Optimierungsprozesses streben, da

$$\hat{p}_{\boldsymbol{\theta}}(w \mid w_t) = \exp\left(\boldsymbol{c}_w^T \boldsymbol{e}_{w_t}\right) > 0 \tag{3.36}$$

und somit $\hat{p}_{\boldsymbol{\theta}}$ bei den Noise-Mustern den optimalen Wert 0 lediglich näherungsweise annehmen kann. Die Tatsache, dass die Noise-Muster bzw. die nicht im Text enthaltenen Muster mit einer Wahrscheinlichkeit nahe 0 belegt werden sollen, erscheint zudem auch sehr plausibel.

Der Satz hat uns das überraschende Resultat geliefert, dass die nicht-normalisierten Modellausgaben während des Trainings gegen die empirischen Wahrscheinlichkeiten, die sich zu 1 aufsummieren, konvergieren, falls eine hinreichend große Anzahl an Noise-Trainingsmustern betrachtet wird. Mit dem nächsten und nun sehr naheliegenden Lemma wollen wir unter eben dieser Voraussetzung ($\hat{p}_{\boldsymbol{\theta}}(w \mid w_t) \approx \tilde{p}(w \mid w_t)$) zeigen, dass der Normalisierungsterm $\mathcal{Z}_{\boldsymbol{\theta}}(w_t)$ tatsächlich den Wert 1 approximiert und somit dann die Ausdrücke $\hat{p}_{\boldsymbol{\theta}}$ näherungsweise als Wahrscheinlichkeiten aufgefasst werden können.

Lemma 3.3. *Gilt für ein gegebenes Zentrumswort w_t und ein beliebiges $\epsilon > 0$*

$$\left|\log \hat{p}_{\boldsymbol{\theta}}(w \mid w_t) - \log \tilde{p}(w \mid w_t)\right| \leq \epsilon \qquad \forall w \in \mathcal{V}\ , \tag{3.37}$$

so folgt

$$\left|\log \mathcal{Z}_{\boldsymbol{\theta}}(w_t)\right| \leq \epsilon\ . \tag{3.38}$$

Beweis.

- *zum einen gilt*

$$\begin{aligned}
\log \mathcal{Z}_{\boldsymbol{\theta}}(w_t) &= \log \sum_{w_l \in \mathcal{V}} \hat{p}_{\boldsymbol{\theta}}(w_l \mid w_t) \\
&= \log \sum_{w_l \in \mathcal{V}} \exp\left(\underbrace{\log \hat{p}_{\boldsymbol{\theta}}(w_l \mid w_t) - \log \tilde{p}(w_l \mid w_t)}_{\leq \epsilon} + \log \tilde{p}(w_l \mid w_t)\right) \\
&\leq \log \sum_{w_l \in \mathcal{V}} \exp(\epsilon) \cdot \exp \log \tilde{p}(w_l \mid w_t) = \log\left(\exp(\epsilon) \underbrace{\sum_{w_l \in \mathcal{V}} \tilde{p}(w_l \mid w_t)}_{=1}\right) = \epsilon
\end{aligned}$$

- *auf der anderen Seite erhalten wir*

$$\begin{aligned}
\log \mathcal{Z}_{\boldsymbol{\theta}}(w_t) &= \log \sum_{w_l \in \mathcal{V}} \exp\left(\log \hat{p}_{\boldsymbol{\theta}}(w_l \mid w_t) - \log \tilde{p}(w_l \mid w_t) + \log \tilde{p}(w_l \mid w_t)\right) \\
&= \log \sum_{w_l \in \mathcal{V}} \tilde{p}(w_l \mid w_t) \cdot \exp\left(\log \hat{p}_{\boldsymbol{\theta}}(w_l \mid w_t) - \log \tilde{p}(w_l \mid w_t)\right) \\
&\geq \sum_{w_l \in \mathcal{V}} \tilde{p}(w_l \mid w_t) \cdot \log \exp\left(\underbrace{\log \hat{p}_{\boldsymbol{\theta}}(w_l \mid w_t) - \log \tilde{p}(w_l \mid w_t)}_{\geq -\epsilon}\right) \\
&\geq \sum_{w_l \in \mathcal{V}} \tilde{p}(w_l \mid w_t) \cdot (-\epsilon) = -\epsilon
\end{aligned}$$

- *die erste Abschätzung nutzt die Konkavität des Logarithmus und die Tatsache, dass die Wahrscheinlichkeiten $\tilde{p}(w_l \mid w_t)$ nicht-negative Gewichtsfaktoren darstellen, die sich zu 1 aufsummieren (Jensensche Ungleichung)* □

Zusammenfassend können wir feststellen, dass das Ziel der NCE bzw. der Minimierung von (3.30) darin besteht, die beste nicht-normalisierte Approximation der empirischen Verteilung zu finden. Bei einer hinreichend guten Näherung wird eine Normalisierung implizit vorgenommen, sodass wir

$$\mathcal{Z}_{\boldsymbol{\theta}}(w_t) = \sum_{w_l \in \mathcal{V}} \hat{p}_{\boldsymbol{\theta}}(w_l \mid w_t) = 1 \tag{3.39}$$

annehmen können. Kehren wir nun wieder zum Skip-Gram-Modell zurück, so reicht es aus, von vornherein nur

$$p_{\boldsymbol{\theta}}(w \mid w_t) = \hat{p}_{\boldsymbol{\theta}}(w \mid w_t) = \exp\left(\mathbf{c}_w^T \mathbf{e}_{w_t}\right) \tag{3.40}$$

für die NCE-Zielfunktion (3.27) zu nutzen, sodass wir diese nun vollständig auswerten können mit

$$\begin{aligned} &\mathcal{J}_{\text{NCE}}(w_1, \ldots, w_T, \boldsymbol{\theta}, q, k) \\ &= -\sum_{(w_t, w_{t+i})} \left(\log \frac{\exp\left(\mathbf{c}_{w_{t+i}}^T \mathbf{e}_{w_t}\right)}{k \cdot q(w_{t+i}) + \exp\left(\mathbf{c}_{w_{t+i}}^T \mathbf{e}_{w_t}\right)} + \sum_{\hat{w} \sim q}^{k} \log \frac{k \cdot q(\hat{w})}{k \cdot q(\hat{w}) + \exp\left(\mathbf{c}_{\hat{w}}^T \mathbf{e}_{w_t}\right)} \right) . \end{aligned} \tag{3.41}$$

Folglich wird es durch die Verwendung dieser Funktion vermieden, bei jedem Trainingsmuster über das gesamte Vokabular der Mächtigkeit $|\mathcal{V}|$ zu summieren. Stattdessen reicht es aus, immer nur je k ($\ll |\mathcal{V}|$) weitere, entsprechend einer bestimmten Verteilung gewählte Noise-Wörter zu betrachten, was den Rechenaufwand deutlich verringert, da jetzt die innere Summe der Zielfunktion unabhängig von $|\mathcal{V}|$ ist.

Bemerkung 3.9. *In unserer ursprünglichen Softmax-Zielfunktion (3.4) hätte nicht ohne weiteres auf den Normierungsausdruck verzichtet werden können, da sich dann das Optimierungsproblem*

$$\mathcal{J}_{\text{soft}}(w_1, \ldots, w_T, \boldsymbol{\theta}) \coloneqq -\frac{1}{T} \sum_{t=1}^{T} \sum_{-s \leq i \leq s,\ i \neq 0} \boldsymbol{c}_{w_{t+i}}^T \boldsymbol{e}_{w_t} \to \min \tag{3.42}$$

ergeben hätte. Dies kann das Neuronale Netz dadurch lösen, indem es einfach nur lernt, die Einträge der Gewichtsmatrizen so zu modifizieren, sodass lediglich die Skalarprodukte maximiert werden.

Wird (3.41) nur für ein einzelnes Trainingsmuster (w_t, w_{t+i}) ausgewertet, so bezeichnen wir dies wieder mit

$$\begin{aligned} &J_{\text{NCE}}(w_1, \ldots, w_T, \boldsymbol{\theta}, q, k) \\ &\coloneqq -\left(\log \frac{\exp\left(\mathbf{c}_{w_{t+i}}^T \mathbf{e}_{w_t}\right)}{k \cdot q(w_{t+i}) + \exp\left(\mathbf{c}_{w_{t+i}}^T \mathbf{e}_{w_t}\right)} + \sum_{\hat{w} \sim q}^{k} \log \frac{k \cdot q(\hat{w})}{k \cdot q(\hat{w}) + \exp\left(\mathbf{c}_{\hat{w}}^T \mathbf{e}_{w_t}\right)} \right) . \end{aligned} \tag{3.43}$$

3.2.4 Gradient der NCE-Zielfunktion zur Laufzeit

Es soll nun darauf eingegangen werden, welche Gestalt der Gradient der vereinfachten NCE-Zielfunktion (3.43) zur Laufzeit besitzt, d.h. es werden keine Grenzwertbetrachtungen vorgenommen und exakt k Noise-Muster verarbeitet (vgl. [35]). Um diesen zu bestimmen, nutzen wir wieder die Notationskonvention

$$\Gamma_{\boldsymbol{\theta}}(w_j) \coloneqq -\mathbf{c}_{w_j}^T \mathbf{e}_{w_t} . \tag{3.44}$$

Zum Zweck einer verkürzten Darstellung verzichten wir erneut im Zwischenschritt der folgenden Rechnung auf die Nennung der Funktionsargumente und es folgt

$$\begin{aligned}
&\nabla_{\boldsymbol{\theta}} J_{\mathrm{NCE}}\left(w_1, \ldots, w_T, \boldsymbol{\theta}, q, k\right) \\
&= -\Bigg(\frac{kq + \exp\left(-\Gamma_{\boldsymbol{\theta}}\right)}{\exp\left(-\Gamma_{\boldsymbol{\theta}}\right)} \cdot \frac{\exp\left(-\Gamma_{\boldsymbol{\theta}}\right) \nabla_{\boldsymbol{\theta}}\left(-\Gamma_{\boldsymbol{\theta}}\right) \cdot \left(kq + \exp\left(-\Gamma_{\boldsymbol{\theta}}\right)\right) - \exp^2\left(-\Gamma_{\boldsymbol{\theta}}\right) \nabla_{\boldsymbol{\theta}}\left(-\Gamma_{\boldsymbol{\theta}}\right)}{\left(kq + \exp\left(-\Gamma_{\boldsymbol{\theta}}\right)\right)^2} \\
&\quad + \sum_{\hat{w} \sim q}^{k} \frac{kq + \exp\left(-\Gamma_{\boldsymbol{\theta}}\right)}{kq} \cdot \frac{-kq \cdot \exp\left(-\Gamma_{\boldsymbol{\theta}}\right) \nabla\left(-\Gamma_{\boldsymbol{\theta}}\right)}{\left(kq + \exp\left(-\Gamma_{\boldsymbol{\theta}}\right)\right)^2} \Bigg) \\
&= \underbrace{\frac{k \cdot q\left(w_{t+i}\right)}{k \cdot q\left(w_{t+i}\right) + \exp\left(-\Gamma_{\boldsymbol{\theta}}\left(w_{t+i}\right)\right)}}_{p(y=0 \mid w_{t+i}, w_t)} \cdot \nabla_{\boldsymbol{\theta}} \Gamma_{\boldsymbol{\theta}}\left(w_{t+i}\right) - \sum_{\hat{w} \sim q}^{k} \underbrace{\frac{\exp\left(-\Gamma_{\boldsymbol{\theta}}\left(\hat{w}\right)\right)}{k \cdot q\left(\hat{w}\right) + \exp\left(-\Gamma_{\boldsymbol{\theta}}\left(\hat{w}\right)\right)}}_{p(y=1 \mid \hat{w}, w_t)} \cdot \nabla_{\boldsymbol{\theta}} \Gamma_{\boldsymbol{\theta}}\left(\hat{w}\right).
\end{aligned} \tag{3.45}$$

Dieser Gradient kann ähnlich interpretiert werden wie der Gradient der vereinfachten Softmax-Zielfunktion (3.12). Der erste Summand von (3.45) soll dafür sorgen, dass durch die Parameteraktualisierung das Modell das richtige Wortpaar besser erkennt, während der Summenausdruck das gleiche Ziel in Bezug auf die auftretenden Noise-Muster verfolgt. Ist sich das System sehr sicher und es kann das richtige und die Noise-Muster eindeutig voneinander unterscheiden, d.h. $p\left(y = 0 \mid w_{t+i}, w_t\right) = 0$ und $p\left(y = 1 \mid \hat{w}, w_t\right) = 0$, so werden dementsprechend keine Updates mehr an den Parametern vorgenommen.

3.2.5 Gradient der NCE-Zielfunktion im Grenzfall

Mit dem folgenden Lemma wollen wir zeigen, dass für eine wachsende Anzahl an Noise-Mustern, d.h. $k \to \infty$, der Gradient der vereinfachten NCE-Zielfunktion (3.43) gegen den Gradienten der vereinfachten Softmax-Zielfunktion (3.11) konvergiert.

Lemma 3.4. *Für den Grenzwert des Gradienten der Funktion (3.43) gilt*

$$\lim_{k \to \infty} \nabla_{\boldsymbol{\theta}} J_{\mathrm{NCE}}\left(w_1, \ldots, w_T, \boldsymbol{\theta}, q, k\right) = \nabla_{\boldsymbol{\theta}} \Gamma_{\boldsymbol{\theta}}\left(w_{t+i}\right) - \mathbb{E}_{w_l \sim p_{\boldsymbol{\theta}}}\left[\nabla_{\boldsymbol{\theta}} \Gamma_{\boldsymbol{\theta}}\left(w_l\right)\right] \ , \tag{3.46}$$

wobei wieder die Notationskonvention

$$\Gamma_{\boldsymbol{\theta}}\left(w_j\right) \coloneqq -\boldsymbol{c}_{w_j}^T \boldsymbol{e}_{w_t} \tag{3.47}$$

getroffen wurde.

Beweis.

- *nach (3.45) wissen wir bereits, dass*

$$\begin{aligned}
&\nabla_{\boldsymbol{\theta}} J_{\mathrm{NCE}}\left(w_1, \ldots, w_T, \boldsymbol{\theta}, q, k\right) \\
&= \frac{k \cdot q\left(w_{t+i}\right)}{k \cdot q\left(w_{t+i}\right) + \exp\left(-\Gamma_{\boldsymbol{\theta}}\left(w_{t+i}\right)\right)} \cdot \nabla_{\boldsymbol{\theta}} \Gamma_{\boldsymbol{\theta}}\left(w_{t+i}\right) - \sum_{\hat{w} \sim q}^{k} \frac{\exp\left(-\Gamma_{\boldsymbol{\theta}}\left(\hat{w}\right)\right)}{k \cdot q\left(\hat{w}\right) + \exp\left(-\Gamma_{\boldsymbol{\theta}}\left(\hat{w}\right)\right)} \cdot \nabla_{\boldsymbol{\theta}} \Gamma_{\boldsymbol{\theta}}\left(\hat{w}\right)
\end{aligned}$$

- *da das Strichprobenmittel ein erwartungstreuer Schätzer für den Erwartungswert ist, können wir diesen Ausdruck für hinreichend große k approximieren durch*

$$\frac{k \cdot q\left(w_{t+i}\right)}{k \cdot q\left(w_{t+i}\right) + \exp\left(-\Gamma_{\boldsymbol{\theta}}\left(w_{t+i}\right)\right)} \cdot \nabla_{\boldsymbol{\theta}} \Gamma_{\boldsymbol{\theta}}\left(w_{t+i}\right) - k \cdot \mathbb{E}_{\hat{w} \sim q}\left[\frac{\exp\left(-\Gamma_{\boldsymbol{\theta}}\left(\hat{w}\right)\right)}{k \cdot q\left(\hat{w}\right) + \exp\left(-\Gamma_{\boldsymbol{\theta}}\left(\hat{w}\right)\right)} \cdot \nabla_{\boldsymbol{\theta}} \Gamma_{\boldsymbol{\theta}}\left(\hat{w}\right)\right]$$

$$= \frac{k \cdot q\left(w_{t+i}\right)}{k \cdot q\left(w_{t+i}\right) + \exp\left(-\Gamma_{\boldsymbol{\theta}}\left(w_{t+i}\right)\right)} \cdot \nabla_{\boldsymbol{\theta}} \Gamma_{\boldsymbol{\theta}}\left(w_{t+i}\right) - k \cdot \sum_{\hat{w} \in \mathcal{V}} q\left(\hat{w}\right) \cdot \frac{\exp\left(-\Gamma_{\boldsymbol{\theta}}\left(\hat{w}\right)\right) \cdot \nabla_{\boldsymbol{\theta}} \Gamma_{\boldsymbol{\theta}}\left(\hat{w}\right)}{k \cdot q\left(\hat{w}\right) + \exp\left(-\Gamma_{\boldsymbol{\theta}}\left(\hat{w}\right)\right)}$$

- *im Grenzfall $k \to \infty$ strebt der Vorfaktor des ersten Summanden gegen 1, während für den Summenausdruck gilt*

$$\lim_{k \to \infty} \sum_{\hat{w} \in \mathcal{V}} \frac{k \cdot q\left(\hat{w}\right)}{k \cdot q\left(\hat{w}\right) + \exp\left(-\Gamma_{\boldsymbol{\theta}}\left(\hat{w}\right)\right)} \cdot \exp\left(-\Gamma_{\boldsymbol{\theta}}\left(\hat{w}\right)\right) \cdot \nabla_{\boldsymbol{\theta}} \Gamma_{\boldsymbol{\theta}}\left(\hat{w}\right) = \sum_{\hat{w} \in \mathcal{V}} \exp\left(-\Gamma_{\boldsymbol{\theta}}\left(\hat{w}\right)\right) \cdot \nabla_{\boldsymbol{\theta}} \Gamma_{\boldsymbol{\theta}}\left(\hat{w}\right)$$

$$= \sum_{\hat{w} \in \mathcal{V}} p_{\boldsymbol{\theta}}\left(\hat{w} \mid w_t\right) \cdot \nabla_{\boldsymbol{\theta}} \Gamma_{\boldsymbol{\theta}}\left(\hat{w}\right) \ ,$$

  wobei wir bei der zweiten Gleichheit (3.40) verwendet haben

- *somit folgt nun die Behauptung*

$$\lim_{k \to \infty} \nabla_{\boldsymbol{\theta}} J_{\mathrm{NCE}}\left(w_1, \ldots, w_T, \boldsymbol{\theta}, q, k\right) = \nabla_{\boldsymbol{\theta}} \Gamma_{\boldsymbol{\theta}}\left(w_{t+i}\right) - \sum_{\hat{w} \in \mathcal{V}} p_{\boldsymbol{\theta}}\left(\hat{w} \mid w_t\right) \cdot \nabla_{\boldsymbol{\theta}} \Gamma_{\boldsymbol{\theta}}\left(\hat{w}\right)$$

$$= \nabla_{\boldsymbol{\theta}} \Gamma_{\boldsymbol{\theta}}\left(w_{t+i}\right) - \mathbb{E}_{\hat{w} \sim p_{\boldsymbol{\theta}}}\left[\nabla_{\boldsymbol{\theta}} \Gamma_{\boldsymbol{\theta}}\left(\hat{w}\right)\right]$$

□

Dieses Lemma liefert uns eine Erklärung, wieso die NCE für eine hinreichend große Anzahl an gewählten Noise-Wörtern funktioniert, da im Endeffekt deren Gradient gegen den Gradienten der zu approximierenden Softmax-Zielfunktion konvergiert. Je größer die Anzahl der Noise-Muster ist, desto länger dauern jedoch auch die Berechnungen. Bei der Festsetzung des Hyperparameters k ist somit stets zwischen Rechenzeit und Genauigkeit abzuwägen.

3.2.6 Negative-Sampling

Nachdem die NCE in ihren Einzelheiten dargestellt wurde, soll abschließend auf die Trainingsmethode aus dem originalen Word2Vec-Paper [31] eingegangen werden. Diese wird als *Negative-Sampling* (*NES*) bezeichnet und kann als grobe Approximation der NCE aufgefasst werden (vgl. [11] und [39]). Der Ausgangspunkt der Ausführung bildet dabei die NCE-Zielfunktion (3.41)

$$\mathcal{J}_{\mathrm{NCE}}\left(w_1, \ldots, w_T, \boldsymbol{\theta}, q, k\right)$$

$$= -\sum_{\left(w_t, w_{t+i}\right)} \left(\log \frac{\exp\left(\mathbf{c}_{w_{t+i}}^T \mathbf{e}_{w_t}\right)}{k \cdot q\left(w_{t+i}\right) + \exp\left(\mathbf{c}_{w_{t+i}}^T \mathbf{e}_{w_t}\right)} + \sum_{\hat{w} \sim q}^{k} \log \frac{k \cdot q\left(\hat{w}\right)}{k \cdot q\left(\hat{w}\right) + \exp\left(\mathbf{c}_{\hat{w}}^T \mathbf{e}_{w_t}\right)}\right) \ . \tag{3.48}$$

Die Neuerung beim NES besteht nun darin, dass wir den noch relativ aufwändig zu berechnenden Term $k \cdot q\left(.\right)$ durch den Wert 1 ersetzen wollen. Dies ist dann der Fall, falls

$$k = |\mathcal{V}| \tag{3.49}$$

und q die Gleichverteilung über das Vokabular darstellt. Damit vereinfacht sich (3.48) zu

$$\begin{aligned} &- \sum_{(w_t, w_{t+i})} \left(\log \frac{\exp\left(\mathbf{c}_{w_{t+i}}^T \mathbf{e}_{w_t}\right)}{1 + \exp\left(\mathbf{c}_{w_{t+i}}^T \mathbf{e}_{w_t}\right)} + \sum_{\hat{w} \sim q}^{|\mathcal{V}|} \log \frac{1}{1 + \exp\left(\mathbf{c}_{\hat{w}}^T \mathbf{e}_{w_t}\right)} \right) \\ &= - \sum_{(w_t, w_{t+i})} \left(\log \frac{1}{1 + \exp\left(-\mathbf{c}_{w_{t+i}}^T \mathbf{e}_{w_t}\right)} + \sum_{\hat{w} \sim q}^{|\mathcal{V}|} \log \frac{1}{1 + \exp\left(\mathbf{c}_{\hat{w}}^T \mathbf{e}_{w_t}\right)} \right) . \end{aligned} \tag{3.50}$$

Allerdings besteht nun das Problem, dass die innere Summe über ganz $\mathcal{V}$ läuft, sodass wir im Vergleich zur Ursprungsaufgabe, bei der die Summe im Nenner der Softmax zu umgehen war, keinerlei Verbesserung erreicht haben.

Die zweite Vereinfachung besteht daher darin, dass trotz (3.49) die Summe lediglich aus k ($\ll |\mathcal{V}|$) Elementen gebildet wird, wobei diese wieder entsprechend einer beliebigen Noise-Verteilung q ausgewählt werden. Dies liefert die NES-Zielfunktion

$$\begin{aligned} &\mathcal{J}_{\text{NES}}\left(w_1, \ldots, w_T, \boldsymbol{\theta}, q, k\right) \\ &:= - \sum_{(w_t, w_{t+i})} \left(\log \frac{1}{1 + \exp\left(-\mathbf{c}_{w_{t+i}}^T \mathbf{e}_{w_t}\right)} + \sum_{\hat{w} \sim q}^{k} \log \frac{1}{1 + \exp\left(\mathbf{c}_{\hat{w}}^T \mathbf{e}_{w_t}\right)} \right) \to \min , \end{aligned} \tag{3.51}$$

deren Minimierung dazu führt, dass das Skalarprodukt $\mathbf{c}_{w_{t+i}}^T \mathbf{e}_{w_t}$ für richtige Trainingsmuster groß wird und demgegenüber $\mathbf{c}_{\hat{w}}^T \mathbf{e}_{w_t}$ für Noise-Muster klein wird.

Bemerkung 3.10. *Aufgrund der Vereinfachungen stellt das NES eine Näherung der NCE dar. Bei dieser konnten wir im vorangegangen Abschnitt zeigen, dass deren Gradient den der ursprünglichen Softmax in der Tat approximiert. Diese Garantie haben wir beim NES nun nicht mehr, weshalb diese Methode ungeeignet ist für das Training von Sprachmodellen. Es hat sich jedoch gezeigt, dass es für die Generierung von Wortvektoren erfolgreich eingesetzte werden kann, da in diesem Fall das Sprachmodell nur im Rahmen der Fake-Task Verwendung findet und wir eigentlich nicht an einer Verteilung über Wörtern interessiert sind. Dies rechtfertigt zugleich die verwendeten Vereinfachungen.*

3.3 Verbesserungsmöglichkeiten des Modells

Es existieren verschiedene Möglichkeiten für die Vor- und Nachverarbeitung, um die Performance unseres Skip-Gram-Modells zu verbessern (vgl. [26], [31] und [40]), von denen wir einige kurz vorstellen möchten.

Dynamic-Context-Window (DCW)

Zur Generierung der Trainingsmuster haben wir uns immer die jeweils s vorangegangenen und s nachfolgenden Wörter eines Zentrumswortes angesehen, ohne deren genaue Reihenfolge zu berücksichtigen. Intuitiv erscheint es nun sinnvoll, Nachbarschaftswörter, die weiter vom Zentrumswort entfernt sind, geringer zu gewichten als direkt benachbarte Elemente, da sie wohl weniger mit dem Zentrumswort verbunden sein werden.

Eine Möglichkeit, um dies zu realisieren, besteht darin, dass mit s die maximale Window-Size gekennzeichnet ist. Bei der Erstellung der Trainingsmuster wird nun für jedes Zentrumswort individuell eine zufällig gewählte Window-Size zwischen 1 und s gewählt (z.B. entsprechend einer

Gleichverteilung). Somit werden Wörter mit einer geringeren Distanz häufiger vom Modell in Betracht gezogen, d.h. höher gewichtet.

Subsampling

Sogenannte *Stopwords* ("and", "is", "for", ...) sind sehr häufig verwendete Wörter, die keine nennenswerten Informationen mitteilen. Um zu verhindern, dass das Modell diese Textelemente überbewertet, können diese *vor* Erstellung der Trainingsdaten dezimiert werden. Ein auftretendes Wort w, dessen Häufigkeit $\text{count}(w)$ im zugrundeliegenden Dokumentenkorpus $w_1, \ldots, w_T$ einen vorgegebenen Schwellwert $N_{\max}$ überschreitet, wird mit der Wahrscheinlichkeit

$$1 - \sqrt{\frac{N_{\max}}{\text{count}(w)}} \qquad \text{mit} \qquad \text{count}(w) \geq N_{\max} \ , \tag{3.52}$$

die sich proportional zur Häufigkeit von w verhält, entfernt (angelehnt an [31], vgl. Abbildung 3.4). Dieses Vorgehen erweitert die effektive Window-Size, da nun durch die vorangegangene De-

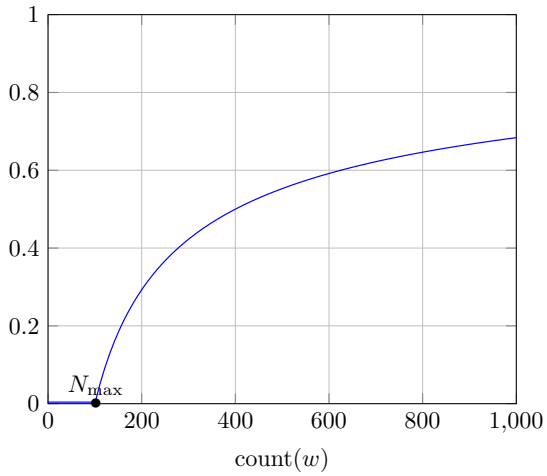


Abbildung 3.4: Verlauf der Dezimierungswahrscheinlichkeiten mit $N_{\max} = 100$

zimierung einzelner Elemente Wörter mit einer größeren Distanz zu den Zentrumswörtern erreicht werden können. In unseren Experimenten werden wir $N_{\max}$ immer auf 100 setzen.

Sowohl die Nutzung des DCW als auch die Reduzierung häufig auftretender Wörter haben zudem den angenehmen Nebeneffekt, dass sich die Anzahl der Trainingsmuster verringert und so der Rechenaufwand abnimmt.

Bemerkung 3.11. *Auf der anderen Seite können natürlich auch sehr selten auftretende Wörter entfernt werden, was wir im Rahmen unserer Untersuchungen allerdings nicht durchführen möchten, da sich gezeigt hat, dass die Auswirkungen dessen nicht gravierend sind (vgl. [26]).*

Noise-Verteilung

Bei der Wahl der Noise-Muster bei der NCE und beim NES liegt es nahe, diese nicht nur entsprechend einer Verteilung auszuwählen, mit der die einzelnen Wörter möglichst kostengünstig generiert werden können, sondern sie sollte auch die Struktur des Textes berücksichtigen. Demnach sollten häufiger auftretende Elemente auch mit einer höheren Wahrscheinlichkeit ausgewählt werden, was mit der Unigram-Verteilung der Wörter sichergestellt werden kann. Auf diese Weise kann das System die Beschaffenheit des Dokumentenkorpus und somit die der Sprache besser erfassen.

Wir folgen nun der Empfehlung der Autoren des Skip-Gram-Modells, die mit Hilfe von Experimenten beobachtet haben, dass die Unigram-Verteilung zur Potenz $\frac{3}{4}$ bessere Ergebnisse liefert als die Gleich- und die gewöhnliche Unigram-Verteilung (vgl. [31]). Die Wahrscheinlichkeit, dass das Wort $w \in \mathcal{V}$ durch die Noise-Verteilung q ausgewählt wird, beträgt danach

$$q(w) = \frac{\text{count}(w)^{\frac{3}{4}}}{\sum_{w_l \in \mathcal{V}} \text{count}(w_l)^{\frac{3}{4}}} \,. \tag{3.53}$$

Die Potenz führt dazu, dass seltenere Wörter nun etwas öfter durch q in Betracht gezogen werden, als ihre relativen Häufigkeiten eigentlich zulassen würden.

Addition von Kontextvektoren (AKV)

Bisher haben wir die Zeilen $\mathbf{e}_w^T$ unserer Einbettungsmatrix E als gesuchte Vektoren der einzelnen Wörter w aufgefasst. Eine neue Repräsentation kann durch eine Konvexkombination mit den Zeilen der Kontextmatrix C erreicht werden, sodass sich der Wortvektor von w durch

$$\boldsymbol{w} = \lambda \cdot \boldsymbol{e}_w + (1-\lambda) \cdot \boldsymbol{c}_w \qquad \text{mit} \qquad \lambda \in [0,1] \tag{3.54}$$

ergibt. In der Literatur werden die Einbettungs- und die Kontextvektoren immer gleichermaßen gewichtet, d.h. $\lambda = 0.5$. Wir wollen aber den allgemeineren Ansatz (3.54) wählen, wobei mit wachsendem λ der Einfluss der Kontextvektoren abnimmt und der der Einbettungsvektoren zunimmt. Zum Abschluss werden die resultierenden Vektoren $\boldsymbol{w}$ normiert

$$\boldsymbol{w} = \frac{\lambda \cdot \boldsymbol{e}_w + (1-\lambda) \cdot \boldsymbol{c}_w}{\|\lambda \cdot \boldsymbol{e}_w + (1-\lambda) \cdot \boldsymbol{c}_w\|_2} \,. \tag{3.55}$$

Dies hat zur Folge, dass sich die Kosinus-Ähnlichkeit (2.1) zwischen zwei Wortvektoren zum gewöhnlichen Skalarprodukt vereinfacht.

Durch dieses Vorgehen wird zudem die Kosinus-Ähnlichkeit erweitert, die wir nun in Terme erster und zweiter Ordnung aufteilen wollen, welche verschiedenen Einfluss auf die Ähnlichkeit zweier Wörter w_1 und w_2 besitzen. Zusammen mit (3.55) erhalten wir

$$\begin{aligned}
\cos_{\text{sim}}(\boldsymbol{w}_1, \boldsymbol{w}_2) &= \boldsymbol{w}_1^T \boldsymbol{w}_2 \\
&= \frac{(\lambda \cdot \boldsymbol{e}_{w_1} + (1-\lambda) \cdot \boldsymbol{c}_{w_1})^T (\lambda \cdot \boldsymbol{e}_{w_2} + (1-\lambda) \cdot \boldsymbol{c}_{w_2})}{\|\lambda \cdot \boldsymbol{e}_{w_1} + (1-\lambda) \cdot \boldsymbol{c}_{w_1}\|_2 \cdot \|\lambda \cdot \boldsymbol{e}_{w_2} + (1-\lambda) \cdot \boldsymbol{c}_{w_2}\|_2} \\
&= \frac{\lambda^2 \cdot \boldsymbol{e}_{w_1}^T \boldsymbol{e}_{w_2} + (1-\lambda)^2 \cdot \boldsymbol{c}_{w_1}^T \boldsymbol{c}_{w_2} + \lambda(1-\lambda) \cdot \boldsymbol{e}_{w_1}^T \boldsymbol{c}_{w_2} + \lambda(1-\lambda) \cdot \boldsymbol{c}_{w_1}^T \boldsymbol{e}_{w_2}}{\|\lambda \cdot \boldsymbol{e}_{w_1} + (1-\lambda) \cdot \boldsymbol{c}_{w_1}\|_2 \cdot \|\lambda \cdot \boldsymbol{e}_{w_2} + (1-\lambda) \cdot \boldsymbol{c}_{w_2}\|_2} \,.
\end{aligned} \tag{3.56}$$

Die *Terme erster Ordnung* sind hierbei die gemischten Ausdrücke im Zähler, d.h. die beiden Skalarprodukte mit je einem Einbettungsvektor $\boldsymbol{e}$ und einem Kontextvektor $\boldsymbol{c}$. Sie geben die Tendenz an, dass ein Wort im Kontext des jeweils anderen auftritt. Die *Terme zweiter Ordnung* sind die nicht-gemischten Skalarprodukte, die die Tendenz angeben, dass beide Wörter im selben Zusammenhang auftreten und daher identische Bedeutungen besitzen oder gar Synonyme darstellen. Die Kosinus-Ähnlichkeit (3.56) besagt also, dass zwei Wörter ähnlich sind, falls sie häufig in einem identischen Kontext auftreten (Verteilungshypothese) oder falls sie häufig im Kontext des jeweils anderen Wortes auftreten (oder eine Kombination aus beidem).

Bemerkung 3.12. *Mit diesen Überlegungen können wir direkt einsehen, dass die NES-Zielfunktion (3.51) versucht, die Terme erster Art für tatsächlich auftretende Trainingsmuster zu maximieren und für die Noise-Muster zu minimieren, was auch intuitiv sinnvoll erscheint.*

Kapitel 4

Das GloVe-Modell

Beim vorangegangenen Skip-Gram-Modell bestand die Grundidee darin, dass es Wort für Wort den Text entlangläuft und versucht, die entsprechenden Nachbarn richtig zu prognostizieren. Daher zählt es zu den Predictive-based-Methoden, wobei bei diesem Vorgehen offenbar auch implizit die Statistik erfasst wird, wie häufig Wörter gemeinsam auftreten. Daher stellt sich nun die Frage, ob diese Statistik auch direkt für ein Wortvektor-Modell betrachtet werden kann, indem die Co-Occurrence-Matrix der Wörter verwendet wird.

Definition 4.1 (Co-Occurrence-Matrix)**.** *Die Co-Occurrence-Matrix N einer gegebenen Textgesamtheit $w_1, \dots, w_T$ basierend auf dem Vokabular $\mathcal{V}$ ist eine $|\mathcal{V}| \times |\mathcal{V}|$ Matrix, bei der jede Zeile und jede Spalte einem Vokabulareintrag zugeordnet ist. Das Matrixelement $N_{w_t,w_{t+i}}$ gibt an, wie häufig das Wort $w_{t+i} \in \mathcal{V}$ in der Umgebung des Zentrumswortes w_t auftritt, welche wieder über eine Window-Size s definiert ist.*

Bereits in Abschnitt 3.1 haben wir indirekt von der Co-Occurrence-Matrix Gebrauch gemacht, da auch dort die Werte $N_{w_t,w_{t+i}}$ verwendet wurden. Für die Zeilensummen von N gilt offenbar

$$\sum_{w_{t+i} \in \mathcal{V}} N_{w_t,w_{t+i}} = N_{w_t} \qquad \text{mit} \qquad w_t \in \mathcal{V} \ , \tag{4.1}$$

was wieder die Gesamtanzahl aller Nachbarn des Zentrumswortes w_t darstellt. Für das Beispiel

"Sometimes the most difficult questions have the simplest solutions." ,

welches nur aus einem einzigen Satz besteht, ergibt sich bei einer konstanten Window-Size von 3 die symmetrische 8×8 Co-Occurrence-Matrix

	sometimes	the	most	difficult	questions	have	simplest	solutions
sometimes	0	1	1	1	0	0	0	0
the	1	0	1	2	2	1	1	1
most	1	1	0	1	1	1	0	0
difficult	1	2	1	0	1	1	0	0
questions	0	2	1	1	0	1	1	0
have	0	1	1	1	1	0	1	1
simplest	0	1	0	0	1	1	0	1
solutions	0	1	0	0	0	1	1	0

.

Bemerkung 4.1. *Wird das DCW genutzt, so besitzt jedes Zentrumswort eine individuelle Window-Size s, sodass die Co-Occurrence-Matrix dann in der Regel nicht mehr symmetrisch ist.*

Weiterhin ist sie für große Datenmengen dünn-besetzt, d.h. ein Wort w besitzt im Text eine geringe Anzahl an unterschiedlichen Nachbarn. Viele Elemente des Vokabulars treten somit nie im Kontext von w auf, was auch die Annahme bei der NCE, dass Noise-Muster deutlich häufiger (k-mal) vorhanden sind, unterstreicht.

Das *GloVe-Modell* (vgl. [36]) stellt einen weiteren populären Ansatz dar, um Wörter in einen Vektorraum via Unsupervised Learning einzubetten. Es verwendet kein Neuronales Netz, sondern minimiert eine Zielfunktion auf Basis der Co-Occurrence-Matrix, um Wortvektoren zu modellieren. Das "GloVe" steht dabei für "Global Vectors", da die verwendete Matrix eine globale Statistik über den gesamten Dokumentenkorpus darstellt.

4.1 Motivation der GloVe-Zielfunktion

Wir wollen nun die verwendete Zielfunktion dieses neuen Ansatzes mit Hilfe des bereits bekannten Skip-Gram-Modells motivieren (vgl. [36], Abschnitt 3.1). Den Ausgangspunkt bildet dabei das Lemma 3.1, in dem die äquivalente Formulierung der Softmax-Zielfunktion

$$\begin{aligned} \mathcal{J}_{\text{soft}}\left(w_1, \ldots, w_T, \boldsymbol{\theta}\right) &= \sum_{w_t \in \mathcal{V}} N_{w_t} \cdot H\left(\tilde{p}\left(. \mid w_t\right), p_{\boldsymbol{\theta}}\left(. \mid w_t\right)\right) \\ &= -\sum_{w_t \in \mathcal{V}} \sum_{w_{t+i} \in \mathcal{V}} N_{w_t} \cdot \tilde{p}\left(w_{t+i} \mid w_t\right) \cdot \log p_{\boldsymbol{\theta}}\left(w_{t+i} | w_t\right) \ , \end{aligned} \tag{4.2}$$

mit

$$\tilde{p}\left(w_{t+i} \mid w_t\right) = \frac{N_{w_t, w_{t+i}}}{N_{w_t}} \qquad \text{und} \qquad p_{\boldsymbol{\theta}}\left(w_{t+i} \mid w_t\right) = \frac{\exp\left(\mathbf{c}_{w_{t+i}}^T \mathbf{e}_{w_t}\right)}{\sum_{w_l \in \mathcal{V}} \exp\left(\mathbf{c}_{w_l}^T \mathbf{e}_{w_t}\right)} \tag{4.3}$$

gezeigt wurde und die es zu minimieren galt, wobei nun auf den Normierungsfaktor $\frac{1}{T}$ verzichtet wird.

Die Darstellung (4.2) kann als globale Formulierung der Zielfunktion des Skip-Gram-Ansatzes verstanden werden, da nicht mehr eine Summe über einzelne lokale Trainingsmuster, sondern eine Summe über das gesamte Vokabular genutzt wird. Allerdings besitzt die verwendete Kreuzentropie den Nachteile, dass sie für Wahrscheinlichkeitsverteilungen definiert ist, was in Bezug auf (4.3) wieder das Problem der Normierung von $p_{\boldsymbol{\theta}}$ hervorruft. Der GloVe-Ansatz nutzt nun als alternativen Abstandsbegriff zu der Kreuzentropie die quadrierte Differenz der nicht-normalisierten und logarithmierten Ausdrücke aus (4.3), d.h. wir erhalten die neue Zielfunktion

$$\begin{aligned} &\sum_{w_t \in \mathcal{V}} \sum_{w_{t+i} \in \mathcal{V}} N_{w_t} \cdot \left(\log N_{w_t, w_{t+i}} - \log \exp\left(\mathbf{c}_{w_{t+i}}^T \mathbf{e}_{w_t}\right)\right)^2 \\ &= \sum_{w_t \in \mathcal{V}} \sum_{w_{t+i} \in \mathcal{V}} N_{w_t} \cdot \left(\mathbf{e}_{w_t}^T \mathbf{c}_{w_{t+i}} - \log N_{w_t, w_{t+i}}\right)^2 \rightarrow \min \ . \end{aligned} \tag{4.4}$$

Der Vorfaktor N_{w_t} kann als Gewichtsterm des jeweils betrachteten Zentrumswortes aufgefasst werden.

Der Logarithmus in (4.4) sorgt dafür, dass die Werte $N_{w_t, w_{t+i}}$ in der Größenordnung verringert werden, da diese für gewisse Wortpaare durchaus groß werden können. Allerdings ist die Co-Occurrence-Matrix in der Regel dünn-besetzt, sodass der Logarithmusausdruck oftmals divergiert. Um diesem Problem entgegenzuwirken, wird N_{w_t} durch eine Gewichtsfunktion $f\left(N_{w_t, w_{t+i}}\right)$ ersetzt, die zudem vom Kontextwort w_{t+i} abhängt und die zusätzlich den Einfluss sehr häufig auftretender

Trainingspaare (w_t, w_{t+i}), d.h. Paare, für die der Wert $N_{w_t,w_{t+i}}$ recht groß ist, eindämmt. Darüber hinaus sollten aber auch seltene Muster mit Vorsicht behandelt werden, da sie nur in wenigen Szenarien auftreten und daher ggf. eine unzuverlässige Aussagekraft besitzen. Dementsprechend sollte die Abbildung f die folgenden drei Voraussetzungen erfüllen:

1) $f(0) = 0$ und im Grenzfall $x \to 0$ sollte die Funktion schnell genug verschwinden, wenn sie als kontinuierlich aufgefasst wird, sodass $\lim_{x\to 0} f(x) \cdot \log^2(x) = 0$. Zur Laufzeit sind die Funktionsargumente diskrete Werte, da die $N_{w_t,w_{t+i}}$ Häufigkeiten darstellen.

2) $f(x)$ ist monoton wachsend, sodass seltene Trainingspaare nicht überbewertet werden.

3) $f(x)$ nimmt für große Argumente x relativ kleine Werte an, sodass häufig auftretende Trainingspaare nicht überbewertet werden.

Es gibt eine Menge an Funktionen, die diese Eigenschaften erfüllen. Entsprechend der Empfehlung der Entwickler von GloVe definieren wir

$$f(x) := \begin{cases} \left(\frac{x}{N_{\max}}\right)^{\alpha} & \text{, falls } x < N_{\max} \\ 1 & \text{, sonst ,} \end{cases} \tag{4.5}$$

wobei der Parameter α auf $\frac{3}{4}$ und der Parameter $N_{\max}$ auf 100 gesetzt wird (vgl. Abbildung 4.1).

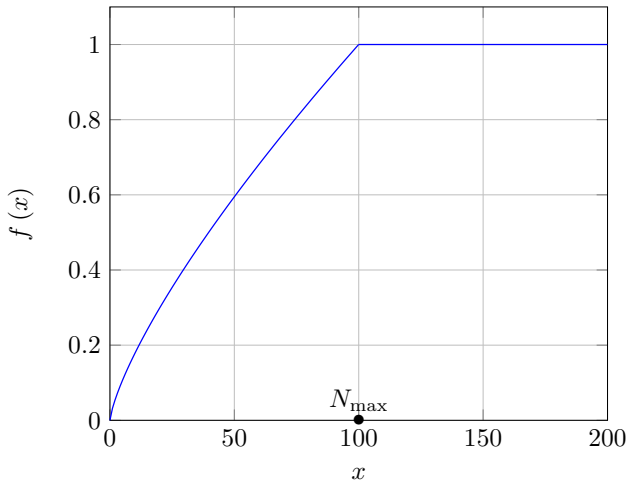


Abbildung 4.1: Verlauf der Gewichtsfunktion des GloVe-Modells

Bemerkung 4.2. *Die Voraussetzung 1) an die Wahl der Gewichtsfunktion f wird von (4.5) erfüllt, wie durch zweimaliges Anwenden der Regel von l'Hospital zu sehen ist*

$$\begin{aligned} \lim_{x\to 0} \left(\frac{x}{N_{max}}\right)^{\alpha} \cdot \log^2(x) &= \frac{1}{N_{\max}^{\alpha}} \cdot \lim_{x\to 0} \frac{\log^2(x)}{x^{-\alpha}} \\ &\overset{LH}{=} \frac{1}{N_{\max}^{\alpha}} \cdot \lim_{x\to 0} \frac{2 \cdot \log(x) \cdot \frac{1}{x}}{-\alpha \cdot x^{-\alpha-1}} = \frac{1}{N_{\max}^{\alpha}} \cdot \lim_{x\to 0} \frac{2 \cdot \log(x)}{-\alpha \cdot x^{-\alpha}} \\ &\overset{LH}{=} \frac{1}{N_{\max}^{\alpha}} \cdot \lim_{x\to 0} \frac{2 \cdot \frac{1}{x}}{-\alpha \cdot (-\alpha) \cdot x^{-\alpha-1}} = \frac{1}{N_{\max}^{\alpha}} \cdot \lim_{x\to 0} \frac{2 \cdot x^{\alpha}}{\alpha^2} \\ &= 0 \qquad \text{, } \textit{falls } \alpha > 0 \text{ .} \end{aligned}$$

Unter Verwendung dieser Funktion f folgt aus (4.4) die Formel

$$\sum_{w_t \in \mathcal{V}} \sum_{w_{t+i} \in \mathcal{V}} f\left(N_{w_t, w_{t+i}}\right) \cdot \left(\mathbf{e}_{w_t}^T \mathbf{c}_{w_{t+i}} - \log N_{w_t, w_{t+i}}\right)^2 \ . \tag{4.6}$$

Abschließend führen wir zusätzliche Bias-Terme ein und wir erhalten die Zielfunktion des GloVe-Modells

$$\begin{aligned} &\mathcal{J}_{\text{GloVe}}\left(w_1, \ldots, w_T, \boldsymbol{\theta}\right) \\ &\coloneqq \sum_{w_t \in \mathcal{V}} \sum_{w_{t+i} \in \mathcal{V}} f\left(N_{w_t, w_{t+i}}\right) \cdot \left(\mathbf{e}_{w_t}^T \mathbf{c}_{w_{t+i}} + b_{w_t}^e + b_{w_{t+i}}^c - \log N_{w_t, w_{t+i}}\right)^2 \to \min \ . \end{aligned} \tag{4.7}$$

Die Parametermenge $\boldsymbol{\theta}$ umfasst eine Einbettungsmatrix $E \in \mathbb{R}^{|\mathcal{V}| \times d}$ sowie eine Kontextmatrix $C \in \mathbb{R}^{|\mathcal{V}| \times d}$. Jedoch existieren nun weiterhin Bias-Werte $\mathbf{b}^e \in \mathbb{R}^{|\mathcal{V}|}$ bzgl. der Einbettungen und $\mathbf{b}^c \in \mathbb{R}^{|\mathcal{V}|}$ bzgl. der Kontexte für jedes einzelne Wort des Vokabulars $\mathcal{V}$. Mit Hilfe eines Gradientenabstiegsverfahrens können nun wieder diese Parameter verändert werden, um ein Minimum der GloVe-Zielfunktion zu erhalten.

Schlussendlich sind auch beim GloVe-Modell Verbesserungsmöglichkeiten wie in Abschnitt 3.3 anwendbar. So kann hier ebenfalls ein DCW, Subsampling und eine Addition der Kontextvektoren vorgenommen bzw. genutzt werden.

Bemerkung 4.3 (Wieso funktioniert GloVe?)**.** *Betrachten wir erneut die Formel (4.6), so ist es dort die Aufgabe, die Ausdrücke*

$$\left(\boldsymbol{e}_{w_t}^T \boldsymbol{c}_{w_{t+i}} - \log N_{w_t, w_{t+i}}\right)^2 \tag{4.8}$$

zu minimieren. Daher werden die Skalarprodukte $\boldsymbol{e}_{w_t}^T \boldsymbol{c}_{w_{t+i}}$ dazu angehalten, die logarithmierten Häufigkeiten des Auftretens der w_{t+i} in der Nachbarschaft der w_t zu kodieren. Da diese Skalarprodukte die Terme erster Art der Kosinus-Ähnlichkeit (3.56) darstellen, ist das GloVe-Modell also in der Lage, die semantische Bedeutung der Wörter sinnvoll zu erfassen.

Bemerkung 4.4. *Im Falle einer symmetrischen Co-Occurrence-Matrix, d.h. $N_{w_t, w_{t+i}} = N_{w_{t+i}, w_t}$, unterscheiden sich die trainierten Vektoren $\boldsymbol{e}_w$ und $\boldsymbol{c}_w$ nur durch die zufällige Initialisierung des Modells (4.7).*

Somit lässt sich zusammenfassend feststellen, dass das GloVe-Modell explizit versucht, Bedeutungen der Wörter über entsprechende Häufigkeiten zu kodieren, was beim Skip-Gram-Ansatz lediglich als "Nebenprodukt" abfällt. Weitere Unterschiede sind kurz im folgenden Abschnitt aufgelistet.

4.2 Unterschiede zwischen GloVe und Skip-Gram

Die vorangegangene Herleitung zeigt auf, dass die Predictive-based- und die Count-based-Methoden zur Berechnung von Wortvektoren nicht strikt voneinander zu trennen sind und durchaus Verbindungen zwischen diesen beiden Typen bestehen. Da die Zielfunktion (4.7) keine Wortvorhersagen mehr trifft und der betrachtete Text in Form der Einträge der Co-Occurrence-Matrix in das Modell einfließt, wollen wir GloVe zu den Count-based-Methoden zählen. Allerdings wurde diese Matrix auf Basis von Zählungen der Nachbarschaftswörter erstellt, sodass auch implizit Informationen über Wortvorhersagen enthalten sind.

Aufgrund der globalen Konstruktion von (4.7) muss pro Trainingsepoche jedes Wortpaar (w_t, w_{t+i}) höchstens einmal verarbeitet werden, da mögliche Mehrfachnennungen durch die Einträge der Matrix N abgedeckt werden. Tritt ein Wortpaar nicht im Textkorpus auf, d.h. $N_{w_t, w_{t+i}}$ nimmt den

Wert 0 an, muss dieses aufgrund der Gewichtsfunktion $f(.)$ überhaupt nicht betrachtet werden. Beim Word2Vec-Modell übergeben wir hingegen die einzelnen lokalen Trainingspaare nach und nach dem Netz, wobei ein und dasselbe Paar auch mehrfach herangezogen wird, wenn es mehrfach in den Dokumenten auftritt. Aufgrund dessen benötigt GloVe weniger Rechenzeit, da es Nutzen aus der globalen Zählung N zieht, deren Generierung allerdings auch eine gewisse Zeit in Anspruch nehmen kann, wobei es sich hierbei um einen einmaligen Vorgang handelt. Wird die Statistik zudem als Matrix abgespeichert, benötigt dies bei einem großen Vokabular sehr viel Speicherplatz ($|\mathcal{V}|^2$ Elemente). Dieses Problem kann dadurch eingedämmt werden, indem die Größe des Vokabulars beschränkt wird (es werden z.B. nur die 20000 häufigsten Wörter in $\mathcal{V}$ aufgenommen) oder indem nur die Werte $N_{w_t,w_{t+i}} \neq 0$ gespeichert werden, was wir in unseren Experimenten später machen werden.

Es sei abschließend nochmals explizit angemerkt, dass das GloVe-Modell solche Trainingspaare, die nicht im Text erscheinen, auch nicht verarbeitet. Beim Skip-Gram-Ansatz geschieht dies jedoch über die Noise-Muster, die in der Zielfunktion zur Approximation der Softmax auftreten.

Bemerkung 4.5. *Prinzipiell sind die beiden in dieser Arbeit vorgestellten Wortvektor-Modelle auf jede beliebige Sprache anwendbar. Die folgenden Experimente basieren jedoch auf englischen Texten, da die meisten bereitgestellten Datensätze zu Evaluierungszwecken in dieser Sprache vorliegen.*

Sollen z.B. Wortvektoren für deutsche Wörter generiert werden, so ist zu beachten, dass die deutsche Sprache eine unweit größere Menge an unterschiedlichen Wörtern und Wortformen besitzt als das Englische. Dies basiert u.a. auf der Vielzahl an zusammengesetzten Substantiven oder der Konjugation von Verben. Dementsprechend sind die einzelnen Wörter in weniger Nutzungsszenarien zu beobachten, was ggf. die Erzeugung passender Vektoren erschweren kann.

Kapitel 5

Evaluierung von Wortvektoren

Nachdem wir die Wortvektoren unter Verwendung verschiedener Methoden erzeugt haben, stellt sich nun die Frage, auf Welche Art und Weise ihre Qualität beurteilt werden kann. Zwar verwenden die beschriebenen Modelle während des Trainings eine Zielfunktion, die für gewöhnlich zur Auswertung herangezogen wird, jedoch ist der durch sie berechnete Fehler nur von geringem Interesse. So ist beim Word2Vec-Modell das Trainingsszenario lediglich ein Mittel zum Zweck, um implizit die gewünschten Einbettungen zu erhalten, sodass es nun anderer Evaluierungsverfahren bedarf.

Es ist zu beachten, dass das Problem der Evaluierung von Wortvektoren nicht trivial ist. Daher wurden verschiedene Möglichkeiten entwickelt, um ihre Qualität sinnvoll einschätzen zu können (für eine Übersicht vgl. [3]).

5.1 Extrinsische und intrinsische Evaluierungsmethoden

Zum einen können Wortvektoren danach bewertet werden, wie gut sie die semantischen Bedeutungen der einzelnen Wörter erfasst haben. Auf der anderen Seite kann es auch von Vorteil sein, sie direkt unter Nutzung der nachgelagerten NLP-Tasks zu evaluieren, um die Performance auf den möglichen Anwendungsgebieten einschätzen zu können.

Auswertungen, bei denen die Wortvektoren als Feature-Vektoren für eine NLP-Task verwendet werden, heißen *extrinsische Evaluierungsmethoden.* Die Performance des NLP-Modells dient dann als Maß für die Qualität. Der Vorteil ist hierbei, dass eine Evaluierung auf einem echten Anwendungsproblem stattfindet. Jedoch ist unter Umständen viel Rechenzeit für die Auswertung notwendig (ggf. muss das nachgelagerte System mit den neuen Wortvektoren erneut trainiert werden) und da Wortvektoren immer nur einen bestimmten Baustein des NLP-Modells darstellen, ist es auch nicht immer klar, wie groß der tatsächliche Einfluss auf das Gesamtergebnis ist. Zudem ist es sehr schwer, einen globalen Evaluationswert mit Hilfe extrinsischer Methoden anzugeben (vgl. Bemerkung 5.1).

Bemerkung 5.1. *Eine gute Performance der Einbettungen auf einer NLP-Aufgabenstellung muss nicht zwangsläufig auch ein gute Performance auf einer anderen nach sich ziehen, d.h. die Ergebnisse auf unterschiedlichen NLP-Tasks korrelieren nicht miteinander (vgl. [42]). Dies liegt daran, dass bei unterschiedlichen Aufgaben auch völlig unterschiedliche Merkmale der Vektoren von Bedeutung sind. So ist es z.B. beim Part-of-speech-Tagging entscheidend, dass Wörter der gleichen Wortart auch als identisch aufgefasst werden, wobei ihre semantische Bedeutung dabei keine große Rolle spielt. Daher macht es Sinn, die Wortvektoren für jede NLP-Task speziell zu trainieren.*

Neben den extrinsischen Verfahren existieren weiterhin die *intrinsischen Evaluierungsmethoden*. Hierbei werden die Einschätzungen von Wortvektoren mit menschlichen Beurteilungen über verschiedene Wortrelationen verglichen. Der Vorteil besteht dabei darin, dass die Evaluierungswerte in der Regel schnell berechnet werden können. Weiterhin helfen die Ergebnisse, die Struktur der Wortvektoren besser zu verstehen, da die Semantik im Mittelpunkt der Betrachtung steht. Allerdings sind die menschlichen Einschätzungen, die die Grundlage der intrinsischen Verfahren bilden, sehr subjektiv geprägt. Zudem ist es nicht klar, inwiefern die Performance auf einer intrinsischen Aufgabenstellung mit der einer extrinsischen korreliert.

Im Folgenden sollen zwei intrinsische Methoden näher erläutert werden, mit denen wir anschließend unsere erzeugten Wortvektoren auch auswerten wollen.

5.2 Word-Similarity-Task

Die *Word-Similarity-Task* nutzt die zu Beginn der Arbeit erwähnte Eigenschaft von Wortvektoren, dass mit ihnen die Ähnlichkeit zwischen Wörtern unter Verwendung der Kosinus-Distanz bzw. der Kosinus-Ähnlichkeit gemessen werden kann (vgl. [3]). Sie zählt zu den populärsten Evaluierungsmethoden.

Bei der Auswertung wird eine Liste mit Wortpaaren der Länge n bereitgestellt. Zunächst werden die Ähnlichkeiten zwischen den Wörtern der einzelnen Paare durch Menschen in Form numerischer Werte x_i mit $i \in \{1, \dots n\}$ beurteilt. Dies geschieht unter der Maßgabe: Je größer der Wert, desto identischer sind die beiden Wörter. Anschließend werden die korrespondierenden Werte y_i mit Hilfe der Wortvektoren via Kosinus-Ähnlichkeit bestimmt. Die beiden sich ergebenen Messreihen $[x_1, \dots, x_n]$ und $[y_1, \dots, y_n]$ werden dann zur Einschätzung der Qualität der Wortvektoren genutzt.

Dabei ist zu beachten, dass wir später solche Wortpaare bei der Auswertung nicht mit in die Betrachtung einschließen können, bei denen für mindestens eines der beiden Wörter kein Wortvektor existiert, da dieses im Training auf Grundlage des betrachteten Textes $w_1, \dots, w_T$ nicht enthalten war.

5.2.1 Evaluierungsmaß

Zur Gegenüberstellung der beiden Messreihen wird der (*Spearman*) *Rangkorrelationskoeffizient* verwendet. Dieser unterscheidet sich vom gewöhnlichen (Pearson) Korrelationskoeffizienten dadurch, dass er nicht direkt die Werte x_i bzw. y_i der Messreihen verwendet, sondern deren Ränge $\operatorname{rg}(x_i)$ bzw. $\operatorname{rg}(y_i)$, die diese Werte innerhalb ihrer Messreihen annehmen, wobei der kleinste Wert den Rang 1 besitzt. Aufgrund dieser Rangbetrachtung sind wir in der Lage, allgemeinere monotone Zusammenhänge zu erfassen und nicht bloß lineare Beziehungen wie beim gängigen Korrelationskoeffizienten.

Definition 5.1 (Rangkorrelationskoeffizient)**.** *Der Rangkorrelationskoeffizient zwischen den beiden Messreihen $[x_1, \dots, x_n]$ und $[y_1, \dots, y_n]$ ist definiert durch*

$$\rho_s(x_1, \dots, x_n, y_1, \dots, y_n) := \frac{\sum_{i=1}^{n} \left(rg(x_i) - \overline{rg}_x\right) \cdot \left(rg(y_i) - \overline{rg}_y\right)}{\sqrt{\sum_{i=1}^{n} \left(rg(x_i) - \overline{rg}_x\right)^2} \cdot \sqrt{\sum_{i=1}^{n} \left(rg(y_i) - \overline{rg}_y\right)^2}}, \tag{5.1}$$

wobei $\overline{rg}_{\cdot}$ den Mittelwert der Ränge der jeweiligen Messreihe angibt.

Bemerkung 5.2. *Der Rangkorrelationskoeffizient liefert Ergebnisse im Intervall $[-1, +1]$, wobei ein Wert von -1 einen vollständigen negativen monotonen Zusammenhang und ein Wert von*

+1 *einen vollständigen positiven monotonen Zusammenhang anzeigt. In unserem Fall ist natürlich eine positive Korrelation zwischen den von den Wortvektoren gelieferten Ähnlichkeiten und den menschlichen Einschätzungen gewünscht. Oftmals wird* ρ_s *mit* 100 *multipliziert, sodass der angestrebte Zielwert bei der Evaluierung nun* 100 *ist.*

Es kann durchaus vorkommen, dass in einer Messreihe Werte doppelt auftreten, sodass deren entsprechender Rang nicht mehr eindeutig ist. In einem solchen Fall werden zunächst die Daten normal in Ränge ohne Dopplungen transformiert und anschließend bei gleichen Werten die Durchschnittsränge gebildet, die dann für die Berechnung von (5.1) verwendet werden. Zur Veranschaulichung dessen betrachten wir als Beispiel die Messreihe $[80, 2, 3, 3, 1, 2, 4, 2]$ und Tabelle 5.1.

Wert x_i	vorläufiger Rang	Durchschnitts-berechnung	Durchschnittsrang $\mathrm{rg}(x_i)$
80	8	$\frac{8}{1}$	8
2	2	$\frac{2+3+4}{3}$	3
3	5	$\frac{5+6}{2}$	5.5
3	6	$\frac{5+6}{2}$	5.5
1	1	$\frac{1}{1}$	1
2	3	$\frac{2+3+4}{3}$	3
4	7	$\frac{7}{1}$	7
2	4	$\frac{2+3+4}{3}$	3

Tabelle 5.1: Durchschnittsränge der Messreihe $[80, 2, 3, 3, 1, 2, 4, 2]$

Die Transformation der Messreihen in Ränge bringt immer einen Informationsverlust mit sich. Bei der Word-Similarity-Task stellt dies aber keine Einschränkung dar, da die menschlichen Beurteilungen als sehr subjektiv mit vielen Verzerrungen aufzufassen sind, sodass es vorteilhafter erscheint, nur die Ränge der Ähnlichkeiten zu betrachten, anstatt die exakten Werte heranzuziehen. Durch dieses Vorgehen reagiert der Rangkorrelationskoeffizient zudem recht robust auf Ausreißer in den Messreihen.

Allerdings induziert die Nutzung von Rängen, dass die Wortpaare in eine Reihenfolge gebracht werden können. Dies bedeutet demnach, dass völlig unterschiedliche Muster gegenübergestellt werden und nun zu entscheiden ist, ob das Paar ("dog", "cat") eine höhere Ähnlichkeit aufweist als ("banana", "apple"), was problematisch wirkt.

5.2.2 Kritik an der Word-Similarity-Task

Es bestehen diverse Kritikpunkte an dieser Form der Evaluierung von Wortvektoren, von denen wir einige kurz ansprechen wollen (vgl. [12]).

Einer von diesen ist die bereits erwähnte Subjektivität bei der Einschätzung von Ähnlichkeiten zwischen Wörtern durch menschliche Probanden. Weiterhin ist es sehr schwer, mit einem einzigen numerischen Wert die Vielzahl an semantischen Zusammenhängen zwischen Wörtern adäquat zu erfassen und wiederzugeben.

Darüber hinaus besteht ein Problem mit dem bisher ziemlich beliebig gewählten Begriff "Ähnlichkeit". Darunter können je nach Ansicht verschiedenen Bedeutungen gemeint sein. So sind z.B. die

beiden Wörter "house" und "building" sehr ähnlich, da sie gleiche Objekte mit identischen Funktionen beschreiben und daher als Synonyme genutzt werden können (*Semantische-Ähnlichkeit*). Allerdings können die beiden Wörter "building" und "roof" auch als ähnlich aufgefasst werden, da sie häufig im Kontext des jeweils anderen Wortes auftreten (*Semantische-Verbundenheit*). Die Semantische-Ähnlichkeit kann daher als ein strengeres Ähnlichkeitsverständnis interpretiert werden. Je nach Aufgabenstellung ist die eine oder die andere Auffassung von Bedeutung. So ist bei der Textklassifikation eher die Semantische-Verbundenheit von Interesse, während bei der Maschinellen Übersetzung die Semantische-Ähnlichkeit hilfreicher ist (vgl. [20]).

Abschließend bleibt festzustellen, dass es großteils unklar ist, wie gut der verwendete Evaluierungsdatensatz, der in der Regel nur einen sehr geringen Anteil an Wörtern des Vokabulars $\mathcal{V}$ enthält, die semantische Struktur einer ganzen Sprache abbildet. Dies ist eine Problematik, die alle intrinsischen Evaluierungsmethoden betrifft, da eine gute Performance auf den entsprechenden Datensätzen nicht zwangsläufig auf eine gute Einbettung des gesamten Vokabulars durch die Wortvektoren hindeuten muss. Es ist also zu berücksichtigen, welche Elemente ein repräsentativer Datensatz zur Evaluierung enthalten sollte.

5.2.3 Verwendetet Datensätze

Da die Word-Similarity-Task eine sehr gängige Methode ist, existieren auch viele Datensätze in englischer Sprache, die eine Liste von Wortpaaren mit den entsprechenden Ähnlichkeitswerten, die von Menschen bestimmt wurden, beinhalten. Diese Bewertungen besitzen je nach Datensatz unterschiedliche Größenordnungen. Dies stellt allerdings für uns kein Problem dar, da später Rangkorrelationen berechnet werden.

Im Folgenden werden die vier von uns verwendeten Datensätze aufgeführt, wobei zu beachten ist, dass diese vorrangig Substantive enthalten. In der Regel haben die Bewertungen mehrere Personen unabhängig voneinander durchgeführt, sodass in den Listen die entsprechenden Mittelwerte enthalten sind.

- **SimLex-999** (vgl. [20])

  Ein Datensatz bestehend aus 999 Wortpaaren, die sehr streng nach Semantischer-Ähnlichkeit mit Zahlen im Intervall 0 bis 10 bewertet wurden. Der Datensatz umfasst 666 Substantiv-Paare, 222 Verb-Paare und 111 Adjektiv-Paare.

- **MTurk-771** (vgl. [19])

  Ein Datensatz bestehend aus 771 Wortpaaren, die nach Semantischer-Verbundenheit mit Zahlen im Intervall 1 bis 5 bewertet wurden.

- **WordSim-353** (vgl. [13])

  Ein Datensatz bestehend aus 353 Wortpaaren, die nach Semantischer-Verbundenheit mit Zahlen im Intervall 0 bis 10 bewertet wurden.

- **MEN** (vgl. [8])

  Ein Datensatz bestehend aus 3000 Wortpaaren, die nach Semantischer-Verbundenheit mit ganzen Zahlen zwischen 0 und 50 bewertet wurden. Da nur ganze Zahlen verwendet werden, treten häufig Dopplungen auf, sodass später viele Durchschnittsränge zur Ermittlung des Korrelationskoeffizienten zu berechnen sind.

  Im Gegensatz zu den vorangegangenen Datensätzen wurden hier bei der menschlichen Beurteilung die Paare nicht direkt mit einem Wert eingeschätzt, der angibt, wie stark die beiden Wörter semantisch miteinander verbunden sind, sondern es wurden einzelne Wortpaare gegenübergestellt und es sollte dann bestimmt werden, welches von ihnen identischer ist.

Um eine Vorstellung zu erhalten, inwiefern sich die Bewertungen in Bezug auf Semantische-Ähnlichkeit und Semantische-Verbundenheit voneinander unterscheiden, sind in Tabelle 5.2 einige Wortpaare mit ihren entsprechenden Einschätzungen dargestellt, die sowohl im SimLex- als auch im WordSim-Datensatz enthalten sind. Diese Werte sind miteinander vergleichbar, da sie sich im selben Intervall $[0, 10]$ bewegen.

Wort 1	Wort 2	SimLex-Score	WordSim-Score
coast	shore	8.83	9.1
aluminium	metal	7.25	7.83
closet	clothes	3.27	8.0
women	men	3.33	8.3

Tabelle 5.2: Gegenüberstellung ausgewählter Ähnlichkeitsbewertungen des SimLex- und des WordSim-Datensatzes

Die ersten beiden Wortpaare werden beide Male als ziemlich identisch bewertet, da u.a. "aluminium" und "metal" zum einen identische Dinge beschrieben und zum anderen häufig im gegenseitigen Kontext auftreten. Allerdings unterscheiden sich die Ansichten bei den anderen beiden Beispielmustern deutlich voneinander. So kommen "closet" und "clothes" häufig im gegenseitigen Kontext vor (daher ein hoher WordSim-Score), stellen aber völlig anderer Objekte mit anderen Funktionen dar (daher ein niedriger SimLex-Score).

5.3 Word-Analogy-Task

Die *Word-Analogy-Task* ist eine zweite und ebenfalls sehr populäre intrinsische Evaluierungsmethode (vgl. [3] und [29]). Diese basiert auf der überraschenden Beobachtung, dass die Wortvektoren nicht nur semantische Identitäten mit Hilfe der Kosinus-Ähnlichkeit erfassen können, sondern auch in der Lage sind, die Struktur der Sprache zu verstehen, sodass dann arithmetische Operationen mit ihnen möglich sind (vgl. [32], dort wird von *Linguistischen Regelmäßigkeiten* gesprochen).

Die Problemstellung besitzt folgende Gestalt: Seien die drei Wörter w_1, w_2 und w_3 gegeben. Dann ist nun dasjenige Wort w_4 gesucht, welches sich zu w_3 genauso verhält wie w_1 zu w_2, d.h

$$w_1 : w_2 \qquad \text{wie} \qquad w_3 : w_4 \ .$$

Kategorien solcher Analogien können von morphologischer Form sein mit Unterklassen wie: Numerus von Substantiven (Singular : Plural), Zeitformen von Verben (Präsens : Präteritum), Steigerungen von Adjektiven (Positiv : Komparativ) oder von semantischer Art mit Unterklassen wie: Genus von Substantiven (Maskulin : Feminin), Nationalitäten (Land : Sprache, Land : Hauptstadt). Betrachten wir das Beispiel

$$\text{"man"} : \text{"woman"} \qquad \text{wie} \qquad \text{"father"} : ? \ ,$$

so ist das gesuchte Wort in diesem Fall offenbar "mother".

Zur Veranschaulichung sei hier angenommen, dass wir im zweidimensionalen operieren. Dann können gut trainierte Wortvektoren solche Analogien mit Hilfe einer linearen Struktur wie z.B. in Abbildung 5.1 wiedergeben, wobei im höherdimensionalen Raum ein Wort mehrere Analogien kodieren kann. Die einzelnen Wörter sind in einer Art und Weise im Vektorraum $\mathbb{R}^2$ angeordnet, sodass die Relationsvektoren $\mathbf{t}, \mathbf{u}$ und $\mathbf{v}$ zwischen den maskulinen und den korrespondierenden femininen Substantiven eine identische Ausrichtung aufzeigen.

Angenommen, die Wortvektoren sind tatsächlich so im Raum angeordnet, dann liegt es nahe, dass gesuchte Wort "mother" dadurch zu ermitteln, indem derjenige Vektor bestimmt wird, der

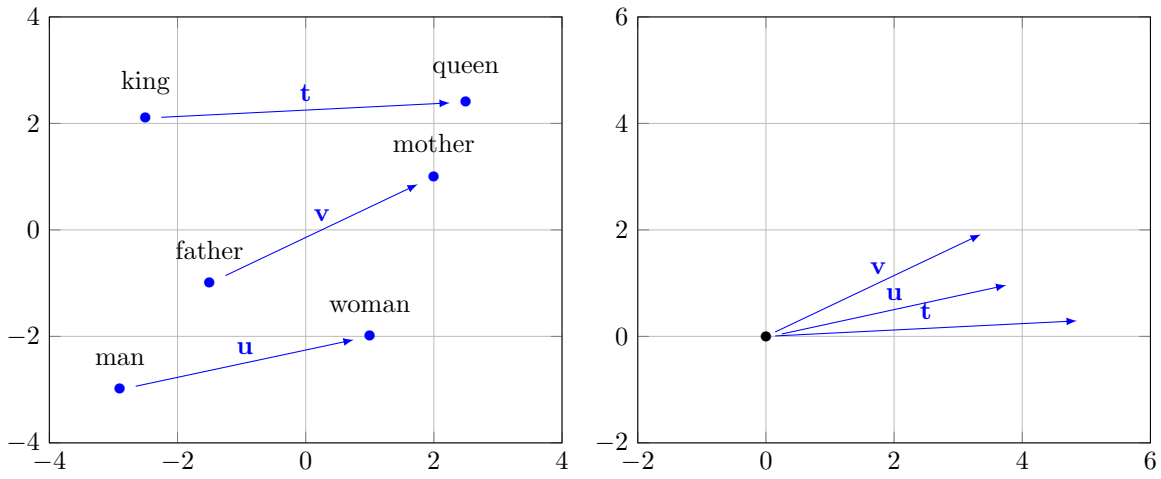


Abbildung 5.1: Möglicher Verlauf der Relationsvektoren (links) und deren Verschiebung in den Ursprung (rechts)

im Sinne der Kosinus-Distanz den geringsten Abstand (also maximale Kosinus-Ähnlichkeit) zum Vektor

$$\boldsymbol{w}_{\text{woman}} - \boldsymbol{w}_{\text{man}} + \boldsymbol{w}_{\text{father}}$$

aufweist. Durch eine solche Erfassung der Analogien einer Sprache können wir demnach arithmetische Operationen mit den Worteinbettungen durchführen, was auch in der Tat zu sinnvollen Ergebnissen führt (vgl. u.a. [32] und Abschnitt 6.2.5).

5.3.1 Evaluierungsmaß

Allgemein kann die Word-Analogy-Task wie folgt formuliert werden (vgl. [24] und [34]). Bei gegebenem Vokabular $\mathcal{V}$ und bei den gegebenen Wörtern w_1, w_2, w_3 ist dasjenige Wort w_4 gesucht, für welches gilt

$$\begin{aligned}
w_4 &= \underset{w \in \mathcal{V} \setminus \{w_2, w_3\}}{\operatorname{argmax}} \cos_{\text{sim}}\left(\boldsymbol{w}_2 - \boldsymbol{w}_1 + \boldsymbol{w}_3, \boldsymbol{w}\right) \\
&= \underset{w \in \mathcal{V} \setminus \{w_2, w_3\}}{\operatorname{argmax}} \frac{\left(\boldsymbol{w}_2 - \boldsymbol{w}_1 + \boldsymbol{w}_3\right)^T \boldsymbol{w}}{\|\boldsymbol{w}_2 - \boldsymbol{w}_1 + \boldsymbol{w}_3\|_2 \cdot \|\boldsymbol{w}\|_2} \\
&= \underset{w \in \mathcal{V} \setminus \{w_2, w_3\}}{\operatorname{argmax}} \frac{\boldsymbol{w}_2^T \boldsymbol{w} - \boldsymbol{w}_1^T \boldsymbol{w} + \boldsymbol{w}_3^T \boldsymbol{w}}{\|\boldsymbol{w}_2 - \boldsymbol{w}_1 + \boldsymbol{w}_3\|_2 \cdot \|\boldsymbol{w}\|_2} \\
&= \underset{w \in \mathcal{V} \setminus \{w_2, w_3\}}{\operatorname{argmax}} \boldsymbol{w}_2^T \boldsymbol{w} - \boldsymbol{w}_1^T \boldsymbol{w} + \boldsymbol{w}_3^T \boldsymbol{w} \\
&= \underset{w \in \mathcal{V} \setminus \{w_2, w_3\}}{\operatorname{argmax}} \cos_{\text{sim}}\left(\boldsymbol{w}_2, \boldsymbol{w}\right) - \cos_{\text{sim}}\left(\boldsymbol{w}_1, \boldsymbol{w}\right) + \cos_{\text{sim}}\left(\boldsymbol{w}_3, \boldsymbol{w}\right) \ .
\end{aligned} \tag{5.2}$$

Die vorletzte Gleichheit gilt aufgrund der Tatsache, weil wir normierte Wortvektoren betrachten und $1/\|\boldsymbol{w}_2 - \boldsymbol{w}_1 + \boldsymbol{w}_3\|_2$ einen konstanten Vorfaktor darstellt. Es ist also jenes Element aus $\mathcal{V}$ von Interesse, welches den beiden Wörtern w_2 und w_3 sehr stark ähnelt, sich jedoch möglichst deutlich von w_1 unterscheidet. Daher werden auch die Wörter w_2 und w_3 bei der argmax-Funktion in (5.2) explizit ausgeschlossen. Die Analogy-Task korrespondiert also in einem gewissen Maße mit der Similarity-Task.

Ein entsprechender Datensatz für diese Evaluierungsmethode enthält folglich eine Liste von Wörtern w_1, w_2, w_3 mit dem entsprechenden Zielwort w_4, welches durch die Vorgehensweise (5.2) mit Hilfe der Wortvektoren richtig zu prognostizieren ist. Je höher der Anteil der Übereinstimmungen, desto besser ist die Qualität der Einbettungen.

Um aber für ein einziges Beispielmuster aussagen zu können, ob auch das entsprechende Zielwort von den Wortvektoren vorhergesagt wird, müssen die Kosinus-Ähnlichkeiten (5.2) über das gesamte Vokabular $\mathcal{V}$ berechnet werden, was sehr viel Rechenzeit in Anspruch nimmt. Dieses Problem wird weiter dadurch verschärft, dass bei vier gegebenen Wörtern eines Analogiemusters auch insgesamt vier verschiedene Auswahlmöglichkeiten für das zu prognostizierende Wort existieren (neben w_4 können durch Umordnung auch w_1, w_2 oder w_3 prognostiziert werden). Diese sollten auch ausgeschöpft werden, da so die Qualität der Vektoren besser eingeschätzt werden kann. Da weiterhin nur exakte Übereinstimmungen als richtige Vorhersagen gewertet werden, liegt ein großer Informationsverlust vor. Ist der Zielvektor $\boldsymbol{w}_4$ nicht der direkte Nachbar von $\boldsymbol{w}_2 - \boldsymbol{w}_1 + \boldsymbol{w}_3$, sondern nur auf Platz zwei oder drei der "Nachbarschaftsliste", so wird dies überhaupt nicht honoriert, obwohl dies auch schon ein gutes Resultat darstellt, wenn sich wieder die große Mächtigkeit von $\mathcal{V}$ vergegenwärtigt wird.

Aufgrund dieser Probleme wollen wir eine alternative Variante zum originalen Word2Vec-Paper verwenden (vgl. [9]). In Abbildung 5.1 haben wir bereits gesehen, dass zu einer Analogie

$$w_1 : w_2 \qquad \text{wie} \qquad w_3 : w_4$$

zwei Relationsvektoren

$$\boldsymbol{w}_2 - \boldsymbol{w}_1 \qquad \text{und} \qquad \boldsymbol{w}_4 - \boldsymbol{w}_3$$

gehören. Besitzen die Wortvektoren eine gute Qualität, sollten diese beiden Relationsvektoren in dieselbe Richtung zeigen, was insbesondere bedeutet, dass sie einen kleinen Winkel einschließen (vgl. Abbildung 5.1 (rechts)). Demnach erscheint es logisch, die Kosinus-Ähnlichkeit dieser beiden Vektoren zu analysieren, d.h.

$$\frac{(\boldsymbol{w}_2 - \boldsymbol{w}_1)^T (\boldsymbol{w}_4 - \boldsymbol{w}_3)}{\|\boldsymbol{w}_2 - \boldsymbol{w}_1\|_2 \cdot \|\boldsymbol{w}_4 - \boldsymbol{w}_3\|_2}\,, \tag{5.3}$$

wobei später bei einer Liste von mehreren Analogiemustern die Mittelwerte berechnet werden. Folglich analysieren wir nun direkt die Relationsvektoren zwischen den gegebenen Wortpaaren des Evaluierungsdatensatzes, anstatt die nächsten Nachbarn im gesamten Vokabular zu suchen.

Bemerkung 5.3. *Die Kosinus-Ähnlichkeit zwischen den Relationsvektoren liefert wieder Ergebnisse im Intervall* $[-1, +1]$*, wobei ein Wert von* -1 *eine vollkommen schlechte Erfassung und ein Wert von* $+1$ *eine vollkommen gute Erfassung der Analogie anzeigt. Hier wird das Ergebnis ebenfalls mit* 100 *multipliziert, sodass der angestrebte Zielwert bei der Evaluierung erneut* 100 *ist.*

5.3.2 Verwendete Datensätze

Auch bei diesem Verfahren existieren verschiedene Datensätze, die zur Evaluierung herangezogen werden können. Diese sind in der Regel in mehrere Kategorien von Analogien aufgeteilt, die von Menschen zusammengestellte Beispiel enthalten.

- **BATS** (vgl. [16])
  Der Datensatz besteht aus zwei morphologischen und zwei semantischen Hauptkategorien, die jeweils in zehn Unterklassen eingeteilt sind. Jede dieser Klassen enthält 50 Wortpaare, die gegenübergestellt werden, wobei dabei deren Reihenfolge bei der Gegenüberstellung zu beachten ist.

Da wir ein Wortpaar nicht mit sich selbst gegenüberstellen wollen, erhalten wir insgesamt

$$4 \cdot 10 \cdot \frac{50!}{(50-2)!} = 98000$$

Evaluierungsmuster.

- **Google Analogy** (vgl. [29])
  Der Datensatz besteht aus einer morphologischen und einer semantischen Hauptkategorie mit insgesamt 14 Unterklassen. Jede dieser Klassen enthält zwischen 20 und 70 Wortpaare. Insgesamt sind 19544 Evaluierungsmuster enthalten.

  Der Datensatz ist allerdings recht unausgeglichen, da u.a. sehr viel Aufmerksamkeit auf die Relationen zwischen Ländern und ihren Städten bzw. zwischen Ländern und ihren Währungen gelegt wird, was ca. 40% aller Muster ausmacht.

Kapitel 6

Experimente

In diesem Kapitel der Masterarbeit sollen das Skip-Gram- und das GloVe-Modell auf Basis der vorangegangenen Ausführungen ausgewertet werden. Die Trainingsgrundlage für alle Experimente bildet der *One-Billion-Word-Benchmark* Textkorpus (vgl. [10]), der in englischer Sprache formuliert ist und vorrangig auf der Grundlage von Nachrichtenartikeln erzeugt wurde.

6.1 Wahl der Hyperparameter

Wir wollen zunächst eine Idee davon erhalten, inwiefern sich die Wahl der Hyperparameter auf die Qualität der erzeugten Wortvektoren auswirkt (für eine Übersicht über alle möglichen Hyperparameter vgl. Tabelle 6.1). Aufgrund des begrenzten Zeitumfangs wurden die folgenden Experimente

	Skip-Gram	GloVe
Window-Size s	✓	✓
Vektordimension d	✓	✓
Anzahl der Noise-Muster k	✓	✗
Lernrate η	✓	✓
Anzahl der Trainingsepochen	30	30
Batch-Size	64	64
Optimierungsalgorithmus	Adam (vgl. [22])	Adam
DCW	✓	✓
Subsampling	✓	✓
Noise-Verteilung	✓	✗
AKV (mit Koeffizient λ)	✓	✓

Tabelle 6.1: Hyperparameter und Verbesserungsmöglichkeiten von Skip-Gram und GloVe im Überblick

lediglich auf einem sehr kleinen Teildatensatz, der eine Million Wörter enthält, durchgeführt, um die Trainingszeiten zu reduzieren. Diesen wollen wir als *One-Million-Word-Korpus* bezeichnen. Dementsprechend sind die Ergebnisse unserer Vektoren auch nicht mit denen der Literatur vergleichbar, da diese dort auf viel größeren Datensätzen trainiert worden sind und dies in der Regel zu deutlich besseren Werten führt.

Die Vorverarbeitung des Textkorpus bestand darin, dass zunächst die einzelnen Sätze voneinander

getrennt und anschließend deren Elemente (Wörter, Ziffern, Sonderzeichen) extrahiert wurden. Dabei wurde alles in Kleinbuchstaben transformiert, wobei wir Sonderzeichen wie Punkte oder Kommas entfernt haben. Auf Grundlage der erhaltenen Wörter konnte dann das Vokabular $\mathcal{V}$ generiert werden, welches insgesamt 57493 Einträge besitzt. Die auftretenden Ziffern wurden während des Trainings einheitlich mit einem `<NUM>` Zeichen erfasst und werden daher alle durch denselben Wortvektor dargestellt.

Die im dritten Block der Tabelle 6.1 aufgelisteten Verbesserungsmöglichkeiten der Modelle werden stets angewendet. Die Auswahl der Gestalt der Noise-Verteilung (3.53) bei Skip-Gram, der Gewichtsfunktion (4.5) bei GloVe und der Entfernungswahrscheinlichkeit (3.52) beim Subsampling kann ebenfalls als Hyperparameter interpretiert werden. Wir folgen aber immer den angegebenen Empfehlungen der entsprechenden Autoren.

Durch das Separieren der einzelnen Textelemente und das Zusammenstellen des Vokabulars kann nun der Text Wort für Wort analysiert und so die Trainingsmuster für das Skip-Gram- und das GloVe-Modell erzeugt werden. Wir trainieren beide Modelle immer 30 Epochen lang, wobei eine Epoche durchlaufen ist, wenn alle vorhandenen Trainingselemente dem Modell präsentiert worden sind.

Für die Verarbeitung eines Evaluierungsmusters sind die Einbettungen für die im Muster enthaltenen Wörter notwendig. Wie wir schon erwähnt haben, können daher nachfolgend solche Muster, bei denen mindestens eines der auftretenden Wörter nicht im Vokabular des Modells enthalten ist, nicht betrachtet werden. Die Anzahlen der verbliebenen Elemente der Evaluierungsdatensätze sind in Tabelle 6.2 aufgelistet.

Datensatz	verbliebene Muster / Gesamtanzahl
SimLex	967 / 999
MTurk	745 / 771
WordSim	334 / 353
MEN	2751 / 3000
BATS	57134 / 98000
Google Analogy	14306 / 19544

Tabelle 6.2: Anzahlen der verbliebenen Muster der Evaluierungsdatensätze beim Training auf dem One-Million-Word-Korpus

Programmiert wurde unter Nutzung von `Python 3`. Zum Zweck der Modelloptimierung wurde weiter das Framework `TensorFlow` verwendet.

6.1.1 Auswertung des Skip-Gram-Modells mit NCE

In der Abbildung 6.1 und der Tabelle 6.3 sind die Ergebnisse des Skip-Gram-Modells dargestellt, wobei wir die Hyperparameter aus dem ersten Block der Tabelle 6.1 untersuchen und das Modell mittels NCE trainiert wurde.

Als *Standardparametrisierung* wurde $s = 16$, $d = 128$, $k = 64$ und $\eta = 0.0001$ gewählt, d.h. in jeder der vier Abbildungen wurde genau ein Parameter variiert, während die übrigen diese Standardwerte annehmen. Der Koeffizient λ zur Konvexkombination der Einbettungs- und der Kontextvektoren nimmt zudem stets den Wert 0.5 an, wobei wir anschließend die resultierenden Wortvektoren normieren. Die Ergebnisse der jeweiligen Task sind über alle verwendeten Datensätze gemittelt abgebildet.

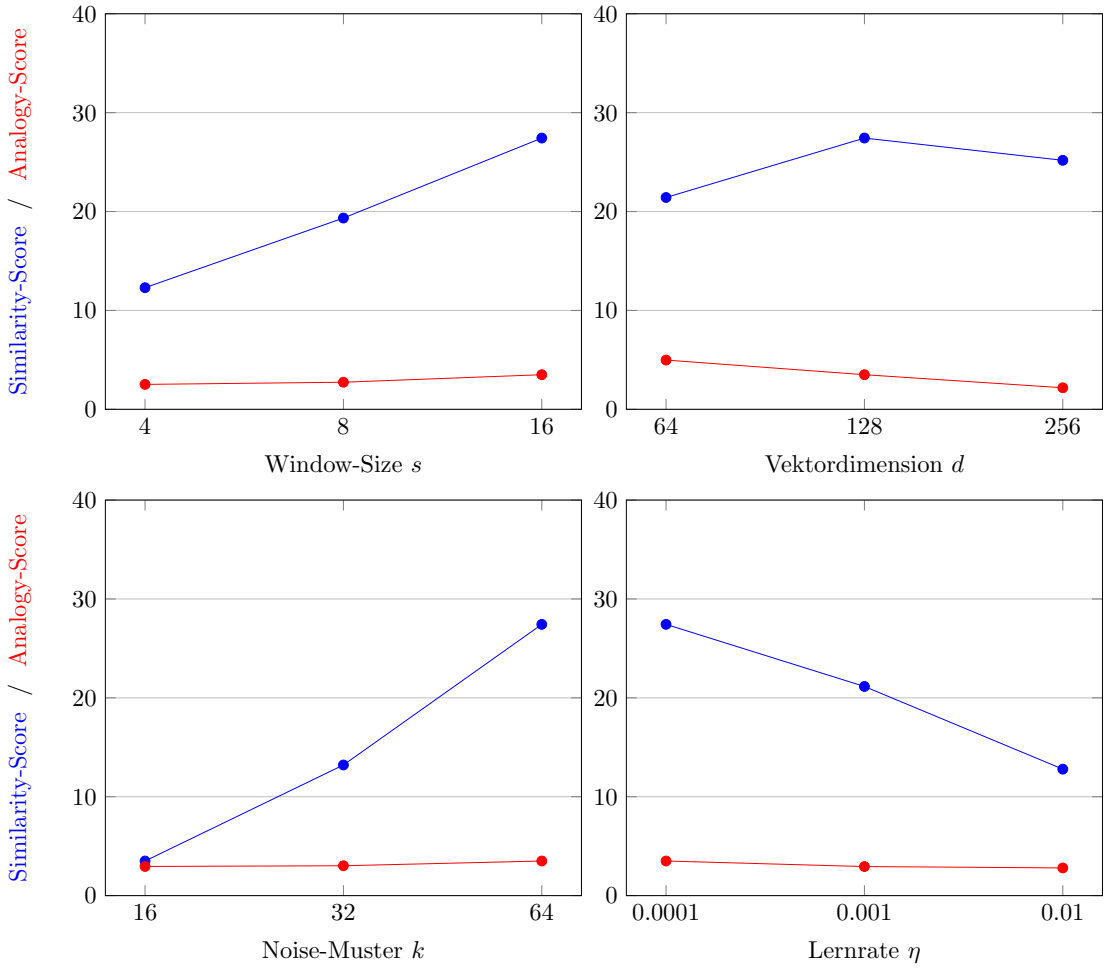


Abbildung 6.1: Ergebnisse des Skip-Gram-Modells mit NCE und der Standardparametrisierung $s = 16$, $d = 128$, $k = 64$, $\eta = 0.0001$ und $\lambda = 0.5$

s		d		k		η	
4	12.3 / 2.52	64	21.42 / **4.98**	16	3.5 / 2.95	0.0001	**27.43** / 3.5
8	19.34 / 2.74	128	**27.43** / 3.5	32	13.21 / 3.02	0.001	21.15 / 2.94
16	**27.43** / 3.5	256	25.18 / 2.18	64	**27.43** / 3.5	0.01	12.79 / 2.8

Tabelle 6.3: Skip-Gram-Resultate entsprechend Abbildung 6.1 mit Similarity-Scores / Analogy-Scores

Es sticht deutlich heraus, dass sich die Performance der Analogy-Task nur unwesentlich verändert. Dies liegt in erster Linie an unserer verwendeten Metrik, die die Kosinus-Ähnlichkeit zwischen den Relationsvektoren heranzieht, deren Richtungen im Optimalfall identisch sein sollen. In Anbetracht dessen, dass wir insbesondere hochdimensionale Räume vorliegen haben, ist dies jedoch ein sehr anspruchsvolles Unterfangen, welches nur geringfügig auf Parameteränderungen reagiert. Wie die Abbildung zeigt, werden die Resultate mit wachsendem d dementsprechend auch schlechter, obwohl intuitiv auch eine größere Dimension sinnvoll erscheinen würde, da dann die Vektoren in der Lage sind, mehr Informationen zu kodieren. In Kombination mit dem kleinen Trainingsdatensatz führt dies zu Ergebnissen, die zwar positiv sind, aber sich deutlich unterhalb des Zielwerts 100 bewegen.

Es ist daran zu erinnern, dass die blauen und die roten Werte der Grafik 6.1 nicht miteinander vergleichbar sind. Zum einen wurden sie mit Hilfe völlig unterschiedlicher Datensätze generiert und zum anderen stellen die Similarity-Scores einen Rangkorrelationskoeffizienten dar, während die Analogy-Scores Kosinus-Ähnlichkeiten wiedergeben.

Analysieren wir nun die Similarity-Task, so nimmt mit wachsender Anzahl an Noise-Mustern die Qualität der Vektoren zu, was mit unseren theoretischen Überlegungen aus Abschnitt 3.2.5 übereinstimmt. Weiterhin ist eine kleinere Lernrate empfehlenswert und eine große Vektordimension muss auch hier nicht zwangsläufig zu besseren Ergebnissen führen, was erfreulich ist, weil mit steigendem d auch die Rechenzeit zunimmt, da das Modell nun viel mehr Parameter besitzt, die im Laufe des Optimierungsprozesses aktualisiert werden müssen. Darüber hinaus führt eine größere Window-Size ebenfalls zu Verbesserungen. Dies liegt daran, dass durch die Betrachtung eines größeren Kontext vor allem die Semantische-Verbundenheit von Wörtern besser vom System erfasst werden kann und drei der vier hier verwendeten Datensätze gerade nach diesem Kriterium erstellt worden sind.

Wie unterschiedlich das System mit der Semantischen-Ähnlichkeit (SimLex-Datensatz) und der Semantischen-Verbundenheit (u.a. WordSim-Datensatz) umgeht, ist in der Grafik 6.2 abzulesen. Das linke Schaubild zeigt auf, dass das Modell besser mit dem weiter gefassten Identitätsbegriff um-

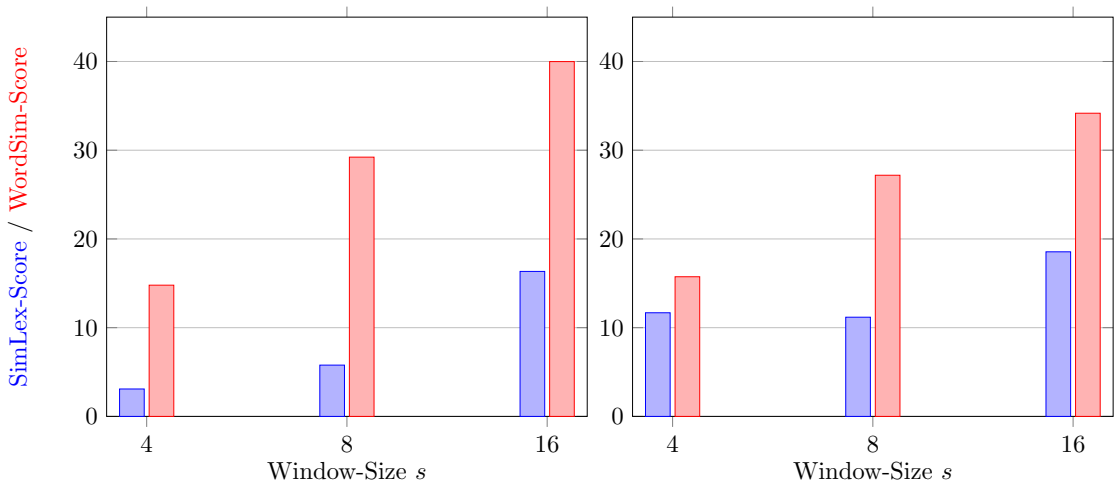


Abbildung 6.2: Einzelresultate des Skip-Gram-Modells mit NCE und der Standarparametrisierung auf dem SimLex- und dem WordSim-Datensatz unter Nutzung der gewöhnlichen Kosinus-Ähnlichkeit (links) und der Kosinus-Ähnlichkeit ohne Terme erster Ordnung (rechts)

gehen kann, da es in erster Linie Wörter als ähnlich auffasst, falls diese häufig gemeinsam im Text auftreten, und dies unabhängig davon, ob sie Synonyme sind oder nicht. Es ist daher für das Skip-Gram-Modell sehr schwer, zwischen Semantischer-Ähnlichkeit und Semantischer-Verbundenheit zu unterscheiden, sodass die Performance auf dem SimLex-Datensatz deutlich schlechter ist.

Interessant ist weiterhin das rechte Schaubild von 6.2. Es ist zu erkennen, dass die Resultate auf

dem SimLex-Datensatz nun deutlich besser geworden sind, wohingegen sich die WordSim-Scores verringert haben. Dies liegt an der Verwendung einer abgeänderten Kosinus-Ähnlichkeit, indem die Terme erster Ordnung in (3.56) entfernt wurden und demnach nur die Terme zweiter Ordnung verbleiben, welche stärker auf die Semantische-Ähnlichkeit abzielen

$$\cos_{\text{sim}}(\boldsymbol{w}_1, \boldsymbol{w}_2) = \frac{\lambda^2 \cdot \boldsymbol{e}_{w_1}^T \boldsymbol{e}_{w_2} + (1-\lambda)^2 \cdot \boldsymbol{c}_{w_1}^T \boldsymbol{c}_{w_2}}{\|\lambda \cdot \boldsymbol{e}_{w_1} + (1-\lambda) \cdot \boldsymbol{c}_{w_1}\|_2 \cdot \|\lambda \cdot \boldsymbol{e}_{w_2} + (1-\lambda) \cdot \boldsymbol{c}_{w_2}\|_2} \,. \tag{6.1}$$

Nutzen wir auf der anderen Seite ausschließlich die Terme erster Ordnung, so rutschen die Resultate auf dem SimLex-Datensatz sogar in den negativen Bereich ab, was wir hier allerdings nicht in einer Grafik festgehalten haben. Allerdings profitiert in diesem Fall der WordSim-Datensatz auch nicht, da er ebenfalls Synonyme mit hohen Ähnlichkeitswerten enthält und es nun schwerer ist, diese zu erkennen, da die Terme erster Art besser die Semantische-Verbundenheit abbilden.

Weiterhin wäre es plausibel, dass sich mit wachsender Window-Size die SimLex-Ergebnisse verschlechtern, da wir Synonyme registrieren wollen und dafür ein kleineres s vorteilhafter erscheint, da dann nicht zu viele bedeutungsfremde Kontextwörter verarbeitet werden. Dies ist hier jedoch nicht zu beobachten (vgl. dazu auch [20]).

Schlussendlich wollen wir noch einen Blick auf den Koeffizienten λ werfen, der bisher immer auf 0.5 gesetzt wurde. In Abbildung 6.3 ist zu erkennen, dass bei der Similarity-Task Vektoren, die

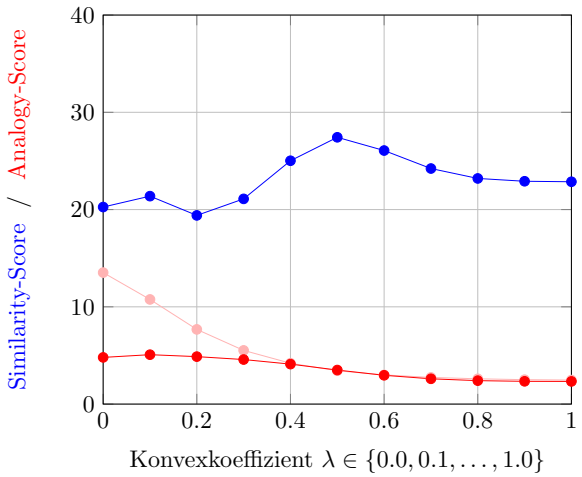


Abbildung 6.3: Ergebnisse des Skip-Gram-Modells mit NCE und der Standarparametrisierung und variierendem λ (im hellroten Fall wurden die Vektoren nicht normiert)

durch eine ausgeglichene Kombination der Einbettungs- und der Kontextvektoren erzeugt wurden, von Vorteil sind, da dann der gesamte Umfang an kodierten Informationen über das Wort selbst und dessen Kontext zur Verfügung steht. Bei der Analogy-Task scheint es hingegen besser zu funktionieren, vorrangig auf die Kontextvektoren zurückzugreifen, da sich für kleiner werdendes λ die Performance verbessert. Mit Hilfe dieser erkennt das Skip-Gram-Modell die Struktur innerhalb der Sprache besser. Dies kann damit motiviert werden, weil die Kontextvektoren vor allem versuchen, die Beziehungen zwischen Wörtern zu erfassen, da sie den Kontext, also eine Vielzahl an Elementen eines Zentrumswortes, kodieren. Daher sind sie auch geeigneter, um Strukturen wiederzugeben.

Überraschend ist das Resultat, welches mir der hellroten Linie dargestellt ist. Hier wurde auf eine abschließende Normierung der Wortvektoren verzichtet. Für die Similarity-Werte ist dies nicht bedeutsam, da die Kosinus-Ähnlichkeit sowieso nur die Richtungen der Vektoren berücksichtigt. Die Analogy-Scores nutzen hingegen die Relationsvektoren zwischen den Einbettungen, sodass zwischen einer Normierung und keiner Normierung der Wortvektoren sehr wohl ein Unterschied besteht. Dieser ist in der Regel nicht gravierend, wobei nun jedoch für kleine λ eine deutliche Performanceverbesserung auftritt. Im Fall $\lambda = 0$ scheint demnach eine Projektion der Wortvektoren,

die nun den Kontextvektoren entsprechen, auf den Rand der Einheitskugel dazu zu führen, dass die erfasste Struktur verzerrt wird und sich dies negativ auswirkt.

6.1.2 Auswertung des Skip-Gram-Modells mit NES

Die gleichen Hyperparameteruntersuchungen wie im vorangegangenen Abschnitt wurden auch für den Fall vorgenommen, bei dem das Training des Skip-Gram-Modells unter Nutzung des NES durchgeführt wurde. Die dazugehörigen Resultate sind in Abbildung 6.4 und Tabelle 6.4 festgehalten.

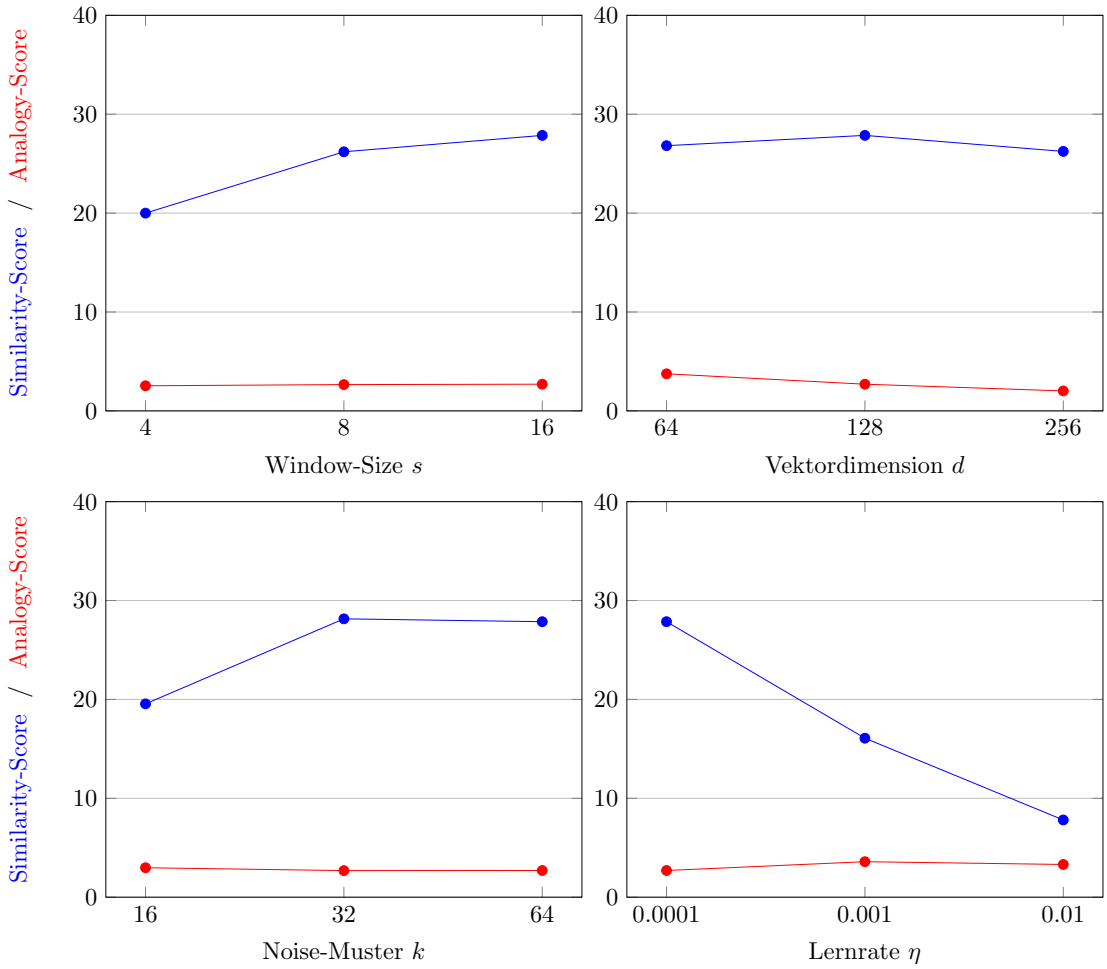


Abbildung 6.4: Ergebnisse des Skip-Gram-Modells mit NES und der Standardparametrisierung $s = 16$, $d = 128$, $k = 64$, $\eta = 0.0001$ und $\lambda = 0.5$

s		d		k		η	
4	20.0 / 2.54	64	26.82 / **3.75**	16	19.55 / 2.98	0.0001	27.86 / 2.7
8	26.2 / 2.66	128	27.86 / 2.7	32	**28.15** / 2.69	0.001	16.08 / 3.59
16	27.86 /2.7	256	26.24 / 2.02	64	27.86 / 2.7	0.01	7.81 / 3.31

Tabelle 6.4: Skip-Gram-Resultate entsprechend Abbildung 6.4 mit Similarity-Scores / Analogy-Scores

Es sind großteils dieselben Trends wie beim Training mit der NCE zu beobachten, auch wenn diese weniger stark ausgeprägt sind. Es kommt hier zu etwas besseren Resultaten bei der Similarity-Task

und zu schlechteren bei der Analogy-Task. Das bessere Ähnlichkeitsverständnis der Vektoren kann damit erklärt werden, weil die nun genutzte Zielfunktion direkt darauf ausgerichtet ist, die Terme erster Ordnung der Kosinus-Ähnlichkeit zu erfassen (vgl. Bemerkung 3.12).

Auch bei der Analyse des Koeffizienten λ ergibt sich ein identisches Bild (vgl. Abbildung 6.5). Die

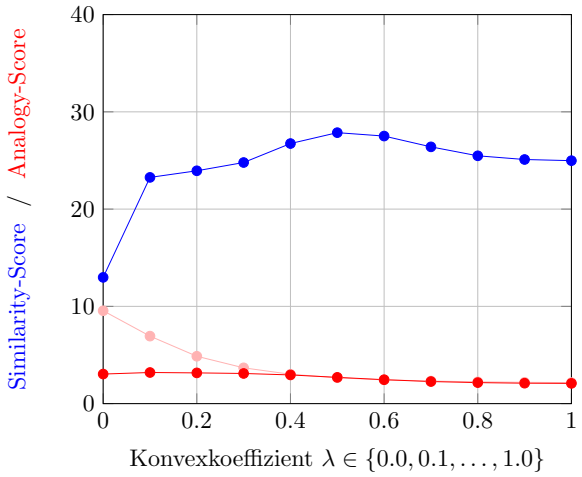


Abbildung 6.5: Ergebnisse des Skip-Gram-Modells mit NES und der Standarparametrisierung und variierendem λ (im hellroten Fall wurden die Vektoren nicht normiert)

Similarity-Task profitiert wieder von einer Kombination der Einbettungs- und der Kontextvektoren. Es tritt zudem ebenfalls das Phänomen auf, dass für kleiner werdendes λ die Ergebnisse der Analogy-Task beim Verzicht auf eine Normierung deutlich besser werden.

Bemerkung 6.1. *Es kann zusammengefasst werden, dass wir in unseren durchgeführten Experimenten keine gravierenden Unterschiede zwischen der NCE und dem NES feststellen können, da das NES eine grobe Näherung der NCE darstellt.*

6.1.3 Auswertung des GloVe-Modells

Bei der Auswertung des GloVe-Modells (vgl. Abbildung 6.6 und Tabelle 6.5) wurde die Standardparametrisierung $s = 16$, $d = 128$ und $\eta = 0.001$ gewählt. Der Koeffizient λ wird wieder zunächst auf 0.5 gesetzt.

s		d		η	
4	9.06 / 1.12	64	10.74 / **1.98**	0.0001	6.6 / 1.48
8	9.62 / 1.54	128	**11.07** / 1.55	0.001	**11.07** / 1.55
16	**11.07** / 1.55	256	10.84 / 1.2	0.01	2.53 / 0.44

Tabelle 6.5: GloVe-Resultate entsprechend Abbildung 6.6 mit Similarity-Scores / Analogy-Scores

Es sind ähnliche Tendenzen wie bei den Skip-Gram-Experimenten zu beobachten, wobei das Niveau, auf dem sich die einzelnen Ergebnisse bewegen, deutlich niedriger ist. Insbesondere ist die Performance auf der Similarity-Task schlechter geworden. Dies ist möglicherweise ein Resultat des sehr begrenzten Trainingsdatensatzes, da die Werte in der Co-Occurrence-Matrix und somit deren Informationsgehalt nicht aussagekräftig genug ist, um die Eigenschaften der Sprache adäquat zu beschreiben.

Eine größere Window-Size und eine nicht zu große Vektordimension führen hier zu den besten Resultaten. Weitere Analysen haben gezeigt, dass sich dieses System ebenfalls deutlich schwerer

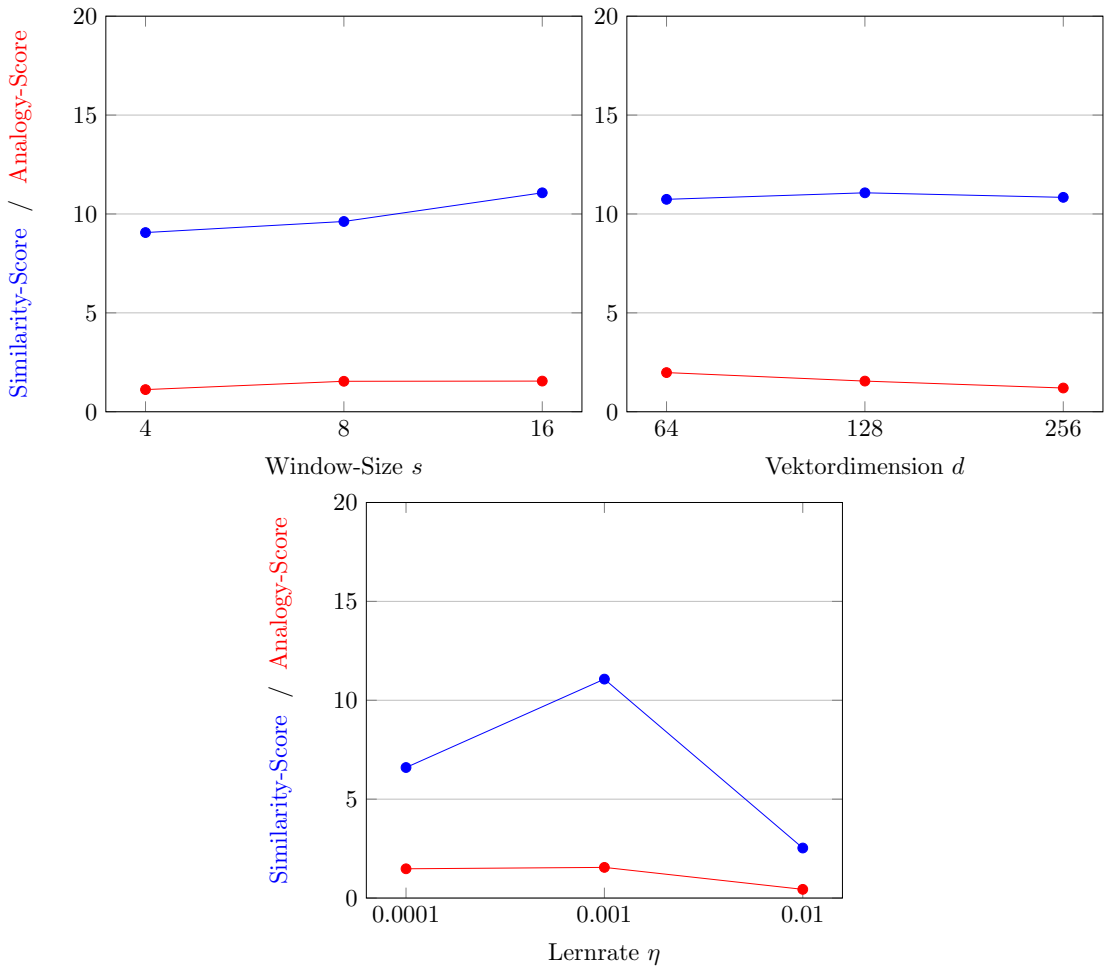


Abbildung 6.6: Ergebnisse des GloVe-Modells mit der Standardparametrisierung $s = 16$, $d = 128$, $\eta = 0.001$ und $\lambda = 0.5$

damit tut, gute Ergebnisse auf dem SimLex-Datensatz zu erzielen, die oftmals sogar negativ sind. Dies ist der Tatsache geschuldet, dass auch dieses Modell die Ähnlichkeit zwischen Wörtern davon ableitet, wie häufig sie gemeinsam im Text auftreten, was hier dem System in Form der Co-Occurrence-Matrix mitgeteilt wird.

In Abbildung 6.7 sind die Ergebnisse bei einer Variation des Koeffizienten λ dargestellt. Es tritt

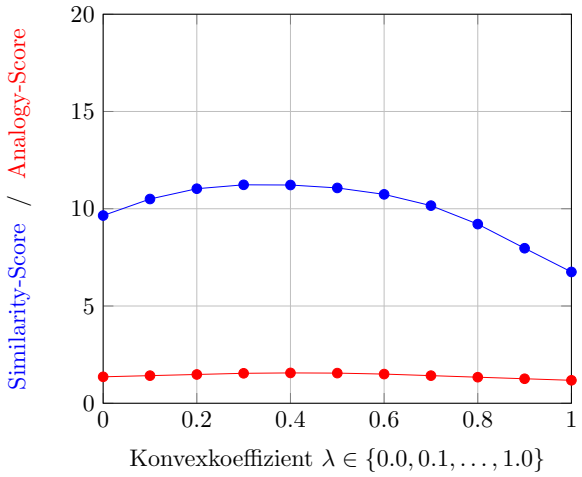


Abbildung 6.7: Ergebnisse des GloVe-Modells mit der Standarparametrisierung und variierendem λ

erneut der Fall ein, dass es bei der Similarity-Task sinnvoller ist, auf alle verfügbaren Modellvektoren zuzugreifen, indem diese kombiniert werden. Bei der Analogy-Task hingegen sind keine nennenswerten Unterschiede auszumachen. Dies kann mit der Bemerkung 4.4 erklärt werden. Unsere Co-Occurrence-Matrix ist aufgrund des verwendeten DCW zwar nicht symmetrisch, aber einem Eintrag $N_{w_t,w_{t+i}}$ steht in der Regel ein Wert N_{w_{t+i},w_t} in ähnlicher Größenordnung gegenüber, sodass die Einbettungs- und die Kontextvektoren identische Informationen über die Struktur der Sprache erfassen. Darüber hinaus ist hier nicht das Phänomen aufgetreten, dass sich bei einem Verzicht auf die Normierung die Resultate zum Teil deutlich verändern.

6.2 Ten-Million-Word-Korpus

Abschließend wurden sowohl das Skip-Gram-Modell mit NCE als auch das GloVe-Modell unter Verwendung ihrer jeweiligen Standardparametrisierungen auf einem größeren Datensatz, welcher aus zehn Millionen Wörtern besteht (*Ten-Million-Word-Korpus*), trainiert. Dieser beinhaltet den zuvor genutzten One-Million-Word-Korpus. Im Folgenden wurde λ immer auf 0.5 gesetzt und die Wortvektoren sind alle normiert. Die Anzahl der Trainingsepochen betrug hier 20, wobei das Vokabular nun insgesamt 194059 Elemente enthält. Die entsprechenden Ergebnisse sind in Tabelle 6.6 abzulesen.

	Skip-Gram 1 Mio	Skip-Gram 10 Mio	GloVe 1 Mio	GloVe 10 Mio
Similarity-Score	27.43	25.97	11.07	13.44
Analogy-Score	3.5	6.72	1.55	1.42

Tabelle 6.6: Skip-Gram-Resultate mit NCE und GloVe-Resultate mit ihren jeweiligen Standarparametrisierungen und $\lambda = 0.5$ beim Training auf dem One-Million- und dem Ten-Million-Word-Korpus im Vergleich

Die Tabelle zeigt auf, dass die Analogy-Task beim Skip-Gram-Modell von einer größeren Trainingsmenge profitiert, während sich die Performance auf der Similarity-Task leicht verschlechtert. Beim GloVe-Ansatz zeigt sich der umgekehrte Trend. Es kann also nicht immer prinzipiell gesagt werden, dass ein größerer Textkorpus auch immer zu besseren Ergebnissen führt. In unseren Experimenten liegen nun zwar mehr Informationen vor, da aber die Vektordimension von 128 gleich geblieben ist, können die Modelle Schwierigkeiten besitzen, diese mit Hilfe der Vektoreinträge zu kodieren. Die Wahl einer größeren Vektordimension, um dieses Problem zu lösen, würde jedoch die Rechenkomplexität noch weiter erhöhen.

6.2.1 Unterklassen des BATS- und des SimLex-Datensatzes

BATS-Datensatz Aufgrund der größeren Trainingsmenge existieren für viel mehr Wörter die entsprechenden Wortvektoren. Daher können nun von den 98000 Mustern des BATS-Datensatzes zur Evaluierung der Analogy-Task 84472 verarbeitet werden. Die Ergebnisse für das Skip-Gram- und das GloVe-Modell sind in der Grafik 6.8 nach den einzelnen Unterklassen aufgeschlüsselt dargestellt.

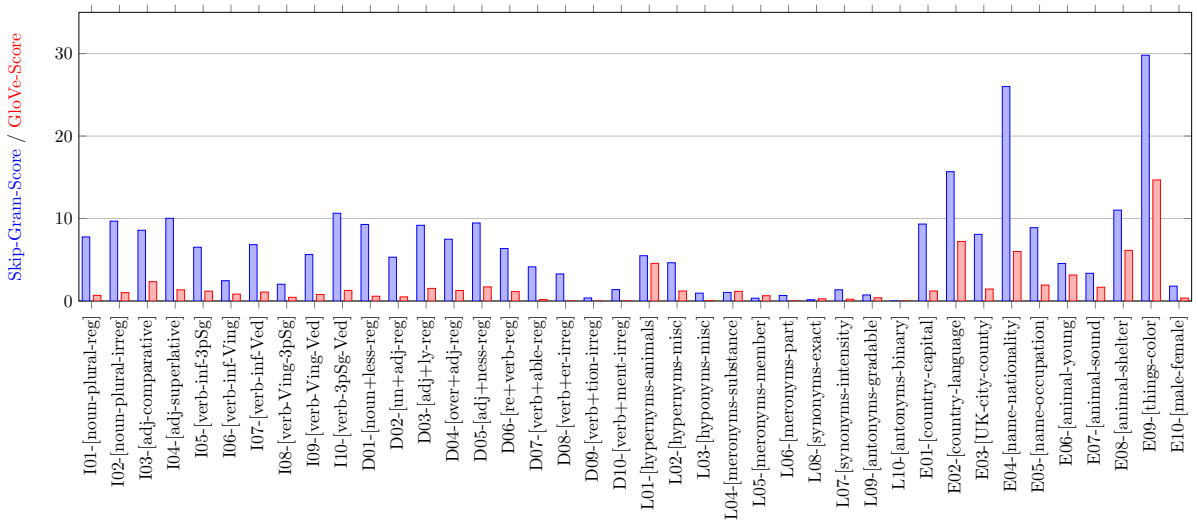


Abbildung 6.8: Einzelergebnisse von Skip-Gram (Durchschnitts-Score = 6.35) und GloVe (Durchschnitts-Score = 1.73) auf den 40 Unterklassen (morphologische Kategorien I und D, semantische Kategorien L und E) des BATS-Datensatzes

Beide Methoden weisen insbesondere auf der semantischen L-Kategorie keine guten Resultate auf. Dies kann dadurch verursacht sein, weil diese Kategorie ein tieferes Verständnis der Sprache erfordert. So sind hier u.a. Relationen zwischen Synonymen (L08-[synonyms-exact] z.B. ("rock" : "stone")) oder zwischen Gegensätzen (L09-[antonyms-gradable] z.B. ("beautiful" : "ugly")) von Bedeutung. Bei den morphologischen Kategorien werden hingegen in der Regel lediglich Wortendungen verändert (I02-[noun-plural-irreg] z.B. ("family" : "families") oder D07-[verb+able-reg] z.B. ("replace" : "replaceable")), was von den Vektoren leichter zu erfassen ist.

Die positiven Ausreißer in der E-Kategorie in Abbildung 6.8 treten auf, da diese Zusammenhänge beinhaltet, die mit Hilfe von Informationen wie aus einer Enzyklopädie zu erfassen sind (E04-[name-nationality] z.B. ("marx" : "german")), sodass die Ergebnisse sehr stark von den verwendeten Trainingstexten und deren Thematiken abhängen. Ein Training auf z.B. Wikipedia-Einträgen könnte hier zu Verbesserungen führen, weil unser Dokumentenkorpus vorrangig auf Nachrichtenartikeln basiert.

Darüber hinaus besteht eine Korrelation zwischen den Ähnlichkeiten der Wörter des Evaluierungs-

musters und dem Analogy-Score. Gegeben sei die Analogie

$$w_1 : w_2 \qquad \text{wie} \qquad w_3 : w_4 \ ,$$

so führt eine hohe Ähnlichkeit zwischen den Vektoren $\boldsymbol{w}_1$ und $\boldsymbol{w}_3$ bzw. zwischen $\boldsymbol{w}_2$ und $\boldsymbol{w}_4$, d.h. die Skalarprodukte $\boldsymbol{w}_1^T\boldsymbol{w}_3$ und $\boldsymbol{w}_2^T\boldsymbol{w}_4$ sind groß, und eine geringe Ähnlichkeit zwischen $\boldsymbol{w}_1$ und $\boldsymbol{w}_4$ bzw. zwischen $\boldsymbol{w}_2$ und $\boldsymbol{w}_3$, d.h. die Skalarprodukte $\boldsymbol{w}_1^T\boldsymbol{w}_4$ und $\boldsymbol{w}_2^T\boldsymbol{w}_3$ sind klein, auch zu einer besseren Performance, da dann die Relationsvektoren

$$\boldsymbol{w}_2 - \boldsymbol{w}_1 \qquad \text{und} \qquad \boldsymbol{w}_4 - \boldsymbol{w}_3$$

mit der Kosinus-Ähnlichkeit

$$\frac{(\boldsymbol{w}_2 - \boldsymbol{w}_1)^T (\boldsymbol{w}_4 - \boldsymbol{w}_3)}{\|\boldsymbol{w}_2 - \boldsymbol{w}_1\|_2 \cdot \|\boldsymbol{w}_4 - \boldsymbol{w}_3\|_2} = \frac{\boldsymbol{w}_2^T\boldsymbol{w}_4 - \boldsymbol{w}_2^T\boldsymbol{w}_3 - \boldsymbol{w}_1^T\boldsymbol{w}_4 + \boldsymbol{w}_1^T\boldsymbol{w}_3}{\|\boldsymbol{w}_2 - \boldsymbol{w}_1\|_2 \cdot \|\boldsymbol{w}_4 - \boldsymbol{w}_3\|_2}$$

eher in dieselbe Richtung zeigen. So ergibt sich z.B. für die Vektoren des Skip-Gram-Modells ein Rangkorrelationskoeffizient zwischen der Summe der Skalarprodukte $\boldsymbol{w}_1^T\boldsymbol{w}_3$ und $\boldsymbol{w}_2^T\boldsymbol{w}_4$ und dem Analogy-Score von 26.34 (positive Korrelation; bei GloVe: 49.42) und zwischen der Summe von $\boldsymbol{w}_1^T\boldsymbol{w}_4$ und $\boldsymbol{w}_2^T\boldsymbol{w}_3$ und dem Analogy-Score von -20.96 (negative Korrelation; bei GloVe: -45.52) auf dem BATS-Datensatz. Wie wir schon mit (5.2) einsehen konnten, ist die Analogy-Task also darauf angewiesen, dass das Modell auch die Ähnlichkeiten zwischen den Wörtern gut erkennt (in [37] wurden identische Experimente durchgeführt, die den Zusammenhang zwischen der Ähnlichkeit der Wörter der Analogie und dem Analogy-Score untersuchen).

SimLex-Datensatz Abbildung 6.9 illustriert die Performance der Similarity-Task der beiden Modelle auf dem SimLex-Datensatz, wobei die 111 Adjektiv-Paare, 222 Verb-Paare und die 666 Substantiv-Paare hier jeweils einzeln analysiert wurden. Es ist herauszustellen, dass die Vektoren

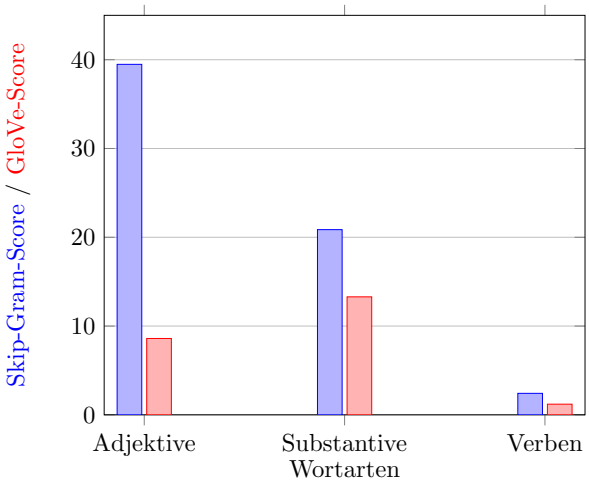


Abbildung 6.9: Einzelergebnisse von Skip-Gram und GloVe auf den verschiedenen Wortarten des SimLex-Datensatzes

die Beziehungen zwischen Verben deutlich schlechter erfassen als bei den anderen beiden Wortarten (vgl. dazu auch [20]).

6.2.2 Similarity-Task mit der Kosinus-Ähnlichkeit und der Euklid-Ähnlichkeit

Seit Beginn der Arbeit nutzen wir die Kosinus-Ähnlichkeit, wobei für die Similarity-Task auch ein Ähnlichkeitsmaß verwendet werden kann, welches mit Hilfe der Euklid-Norm berechnet wird.

Definition 6.1 (Euklid-Ähnlichkeit, vgl. [14])**.** *Die Euklid-Ähnlichkeit zweier Vektoren $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ ist definiert durch*

$$\mathrm{euk}_{\mathrm{sim}}(\boldsymbol{x}, \boldsymbol{y}) \coloneqq -\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \quad \leq 0 \ , \tag{6.2}$$

wobei ein Wert von 0 eine hohe und ein großer negativer Wert eine geringe Ähnlichkeit anzeigt.

Bemerkung 6.2. *Für normierte Vektoren $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ ist die Euklid-Ähnlichkeit proportional zur Kosinus-Ähnlichkeit, da*

$$\mathrm{euk}_{\mathrm{sim}}(\boldsymbol{x}, \boldsymbol{y}) = -\left(\boldsymbol{x} - \boldsymbol{y}\right)^T \left(\boldsymbol{x} - \boldsymbol{y}\right) = -\boldsymbol{x}^T\boldsymbol{x} - \boldsymbol{y}^T\boldsymbol{y} + 2\boldsymbol{x}^T\boldsymbol{y} = 2 \cdot \cos_{\mathrm{sim}}\left(\boldsymbol{x}, \boldsymbol{y}\right) - 2 \ . \tag{6.3}$$

Normierte Wortvektoren liefern bei der Similarity-Task unter Nutzung der Kosinus-Ähnlichkeit und der Euklid-Ähnlichkeit dieselben Ergebnisse, da bei dieser der Rangkorrelationskoeffizient ermittelt wird. Wie in der vorangegangenen Bemerkung zu sehen war, verhält sich die Euklid-Ähnlichkeit eines Wortpaares proportional zu dessen Kosinus-Ähnlichkeit, sodass sich dieselben Ränge innerhalb der jeweiligen Messreihen ergeben und sich daher die Resultate gleichen.

Für den Fall nicht-normierter Wortvektoren sind die Ergebnisse in Tabelle 6.7 gegenübergestellt. Es ist deutlich zu erkennen, dass die Euklid-Ähnlichkeit die Beziehungen zwischen den Wörtern

	Skip-Gram 1 Mio	Skip-Gram 10 Mio	GloVe 1 Mio	GloVe 10 Mio
Kosinus-Score	27.43	25.97	11.07	13.44
Euklid-Score	21.93	20.53	2.47	0.8

Tabelle 6.7: Skip-Gram-Resultate mit NCE und GloVe-Resultate mit ihren jeweiligen Standardparametrisierungen und $\lambda = 0.5$ bei der Similarity-Task mit der Kosinus- und der Euklid-Ähnlichkeit im Vergleich, wobei die Wortvektoren nicht normiert wurden

schlechter erfasst, was die zu Beginn der Arbeit getroffenen Annahme, dass die Richtungen der Einbettungen mehr Aussagekraft besitzen als deren Längen, untermauert.

In den Abbildungen 6.3 und 6.5 haben wir jedoch auch gesehen, dass die Länge der Vektoren durchaus entscheidend sein kann, da es durch eine Normierung zu deutlichen Performanceeinbußen bei der Analogy-Task gekommen ist.

6.2.3 Beziehung zwischen den Einbettungs- und den Kontextvektoren

Weiterhin wollen wir uns die Beziehung zwischen den generierten Einbettungs- und Kontextvektoren ansehen. Wir untersuchen dies zunächst anhand der Kosinus-Ähnlichkeit zwischen dem Einbettungs- und dem Kontextvektor desselben Wortes. Bilden wir den Durchschnitt dieser Ähnlichkeiten

$$\frac{\mathbf{e}_w^T\mathbf{c}_w}{\|\mathbf{e}_w\|_2 \cdot \|\mathbf{c}_w\|_2} \tag{6.4}$$

über alle Wörter $w \in \mathcal{V}$, so ergibt sich beim Skip-Gram-Ansatz ein Wert von -0.31. Dies erscheint auch sinnvoll, da der Zähler von (6.4) den Term erster Art von (3.56) darstellt. In diesem Fall gibt er die Tendenz an, dass ein Wort im eigenen Kontext auftritt. Dies ist jedoch recht unwahrscheinlich, da ein und dasselbe Wort in der Regel sehr selten mehrmals kurz hintereinander in einem Satz verwendet wird. Das GloVe-Modell erkennt diesen Zusammenhang hingegen nicht so gut, da wir hier einen Durchschnittswert von 0.07 erhalten.

Angelehnt an [33] können wir dieses Resultat allgemeiner untersuchen, indem wir den Mittelwert über alle Einbettungsvektoren des Vokabulars

$$\overline{\mathbf{e}} = \frac{1}{|\mathcal{V}|} \cdot \sum_{w \in \mathcal{V}} \mathbf{e}_w \tag{6.5}$$

in beiden Modellen berechnen. Betrachten wir die durchschnittliche Kosinus-Ähnlichkeit der Einbettungsvektoren mit $\overline{\mathbf{e}}$ und der Kontextvektoren mit $\overline{\mathbf{e}}$ (vgl. Tabelle 6.8), so ist zu erkennen, dass die Einbettungsvektoren bei Skip-Gram im Mittel eine hohe Ähnlichkeit mit $\overline{\mathbf{e}}$ aufweisen, d.h. sie zeigen alle tendenziell in dieselbe Richtung. Beim GloVe-Modell ist dies jedoch nicht zu beobachten.

	Skip-Gram	GloVe
$\varnothing \cos_{\text{sim}}$ der $\mathbf{e}_w$ mit $\overline{\mathbf{e}}$	0.648	0.092
$\varnothing \cos_{\text{sim}}$ der $\mathbf{c}_w$ mit $\overline{\mathbf{e}}$	-0.609	0.086

Tabelle 6.8: Durchschnittliche Kosinus-Ähnlichkeiten der Einbettungs- und der Kontextvektoren des Skip-Gram- und des GloVe-Modells mit $\overline{\mathbf{e}}$

Wird nun hingegen der Kosinus der Winkel zwischen den Kontextvektoren und $\overline{\mathbf{e}}$ ermittelt, so zeigen diese beim Skip-Gram-Modell im Schnitt genau in die entgegengesetzte Richtung wie $\overline{\mathbf{e}}$ und somit auch in die entgegengesetzte Richtung zum Großteil der Einbettungsvektoren. Beim GloVe-Ansatz weisen sie im Schnitt eine identische Beziehung zu $\overline{\mathbf{e}}$ auf wie die Einbettungsvektoren, was nochmals den (annähernd konstanten) Verlauf der Ergebnisse bzgl. der Analogy-Task in Abbildung 6.7 erklärt, da die Einbettungs- und die Kontextvektoren im GloVe-Modell eine geringerer Menge an unterschiedlichen Informationen kodieren als beim Skip-Gram-Ansatz.

6.2.4 Beziehung zwischen der Häufigkeit eines Wortes und der Häufigkeiten von dessen Nachbarn

In Abbildung 6.10 ist der Zusammenhang illustriert, inwiefern die Häufigkeit eines Wortes w im Dokumentenkorpus $w_1, \ldots, w_T$, d.h. $\text{count}(w)$, mit der durchschnittlichen Häufigkeit von dessen 1000 nächsten Nachbarn bzgl. der Kosinus-Ähnlichkeit korreliert. Zu diesem Zweck wurde das

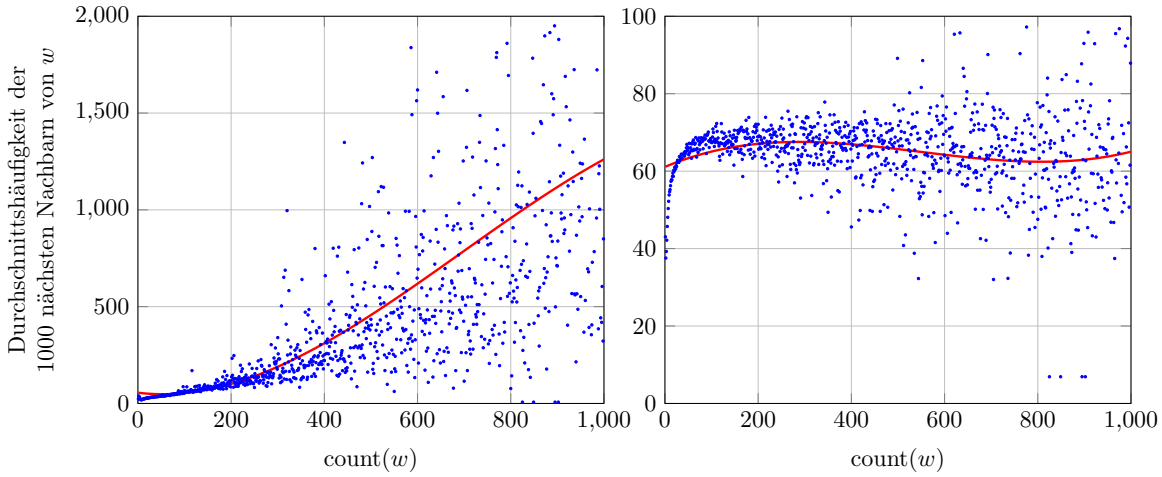


Abbildung 6.10: Zusammenhang zwischen der Häufigkeit eines Wortes und der Durchschnittshäufigkeit der 1000 nächsten Nachbarn (Skip-Gram links, GloVe rechts)

gesamte Vokabular durchlaufen und da natürlich mehrere Wörter z.B. 20-mal auftreten können,

wurde in solchen Fällen der Mittelwert der Durchschnittshäufigkeiten berechnet und als blauer Punkt dargestellt. Deshalb nimmt die Streuung auch mit wachsendem count (w) zu, weil nur wenige Wörter mit größeren Frequenzen im Korpus vertreten sind.

Beim Skip-Gram-Modell ist eine ähnliche Beobachtung wie in [42] zu machen. Häufige Wörter haben in der Regel Nachbarn, die ihrerseits ebenfalls wieder oft auftreten. Die Kosinus-Ähnlichkeit wird somit von den Häufigkeiten verzerrt, was problematisch sein kann, da nun Wörter eher in Umgebungen anderer Elemente gelangen, als ihre Bedeutungen unter Umständen zulassen würden. Beim GloVe-Ansatz ist hingegen ein solcher Zusammenhang nicht abzulesen.

6.2.5 Grafische Darstellung der Wortvektoren

Um schließlich eine Veranschaulichung der generierten Wortvektoren zu erhalten, sollen diese im Folgenden auf Grundlage der Skip-Gram-Methode grafisch dargestellt werden. Dazu muss die Dimension von 128 z.B. auf 2 reduziert werden, was wir mit Hilfe einer *Hauptkomponentenanalyse* realisiert haben.

In Abbildung 6.11 ist sehr gut zu sehen, dass die Vektoren sinnvoll erkannt haben, inwiefern einzelne Wörter miteinander gruppiert werden können. Die Tatsache, dass die Themenkomplexe im oberen Fall sehr deutlich voneinander separiert werden können, begründet sich darin, weil sie keine inhaltlichen Schnittmengen miteinander besitzen. Demgegenüber sind die Bereiche Politik und Wirtschaft eng miteinander verflochten, sodass die zugehörigen Einbettungen nah beieinander liegen.

Weiterhin sind in Tabelle 6.9 einige Beispiele aufgelistet, bei denen durch Nutzung der Wortvektoren und der Kosinus-Ähnlichkeit die identischsten Elemente im Vokabular $\mathcal{V}$ zum Wort $\boldsymbol{w}^*$ berechnet worden sind. Diese folgen der Intuition. Interessant ist, dass das Stopword "of" vom Sys-

$\boldsymbol{w}^*$	
beautiful	amazing, gorgeous, seemingly, wonderful, enormous
gay	marriage, lesbian, samesex, marry, heterosexual
january	march, february, month, next, last
of	in, a, the, and, for
merkel	angela, cdu, steinmeier, spd, chancellor
apple	ipod, iphone, touch, developers, laptops

Tabelle 6.9: Fünf nächsten Nachbarn bzgl. der Kosinus-Ähnlichkeit des Wortes $\boldsymbol{w}^*$ in absteigender Reihenfolge

tem mit anderen Stopwords in Verbindung gebracht worden ist, sodass in der nächsten Umgebung Wörter wie "and" oder "the" zu finden sind. Darüber hinaus assoziieren die Vektoren das Wort "apple" mit dem Technologieunternehmen und nicht mit dem Obst. Dies ist ein Resultat unseres Trainingsdatensatzes, da die Nachrichtentexte in erster Linie mit dem Wort "apple" das Unternehmen ansprechen. Daher können Wörter mit mehreren Bedeutungen je nach Korpus vorrangig in die eine oder in die anderer Richtung interpretiert werden

Neben der Ähnlichkeit von Wörtern können wir ebenfalls die Analogien mit Hilfe von Grafiken gut veranschaulichen (vgl. Abbildung 6.12). Die Grafik bestätigt den Sachverhalt, welchen wir bereits in Abbildung 5.1 angenommen bzw. behauptet haben. Die Wortvektoren haben die Struktur der Sprache in einer solchen Art und Weise erfasst, sodass die Relationsvektoren einer Analogieklasse in der Tat in identische Richtungen zeigen. Somit sind auch arithmetische Operationen dieser Vektoren möglich, die sinnvolle Ergebnisse liefern können. Allerdings ist diese Annahme nicht

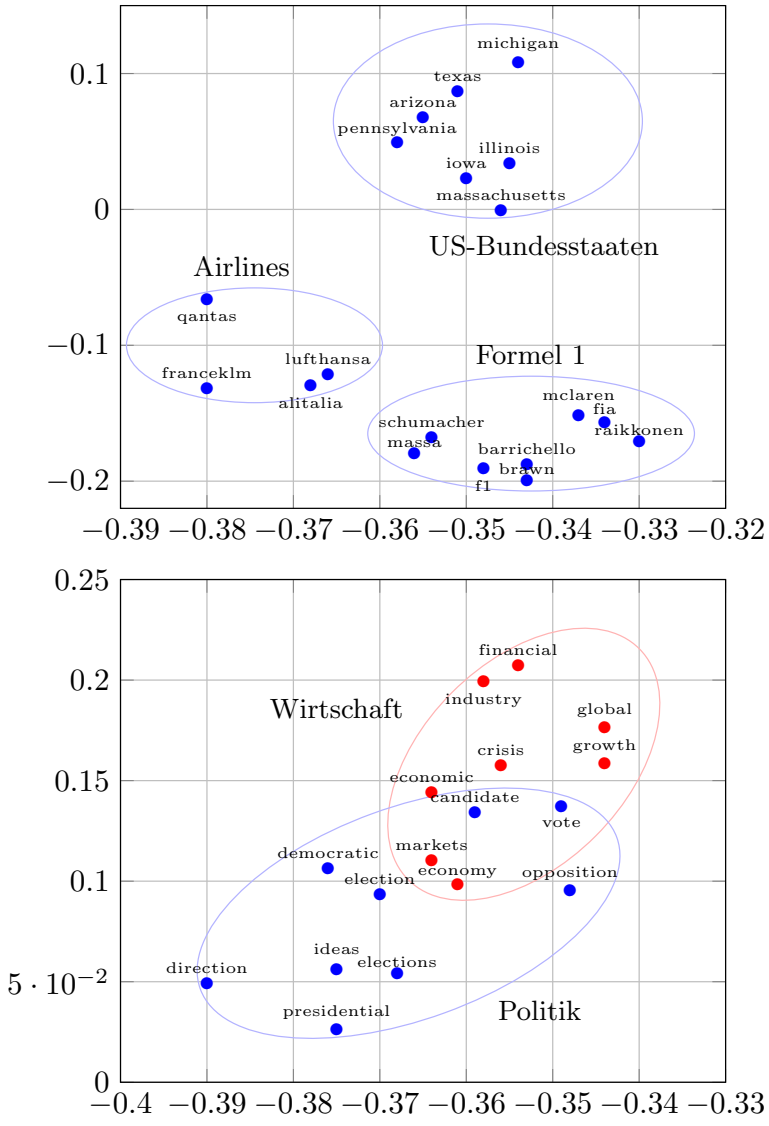


Abbildung 6.11: Zweidimensionale Darstellung von Wortvektoren bestimmter Themenbereiche

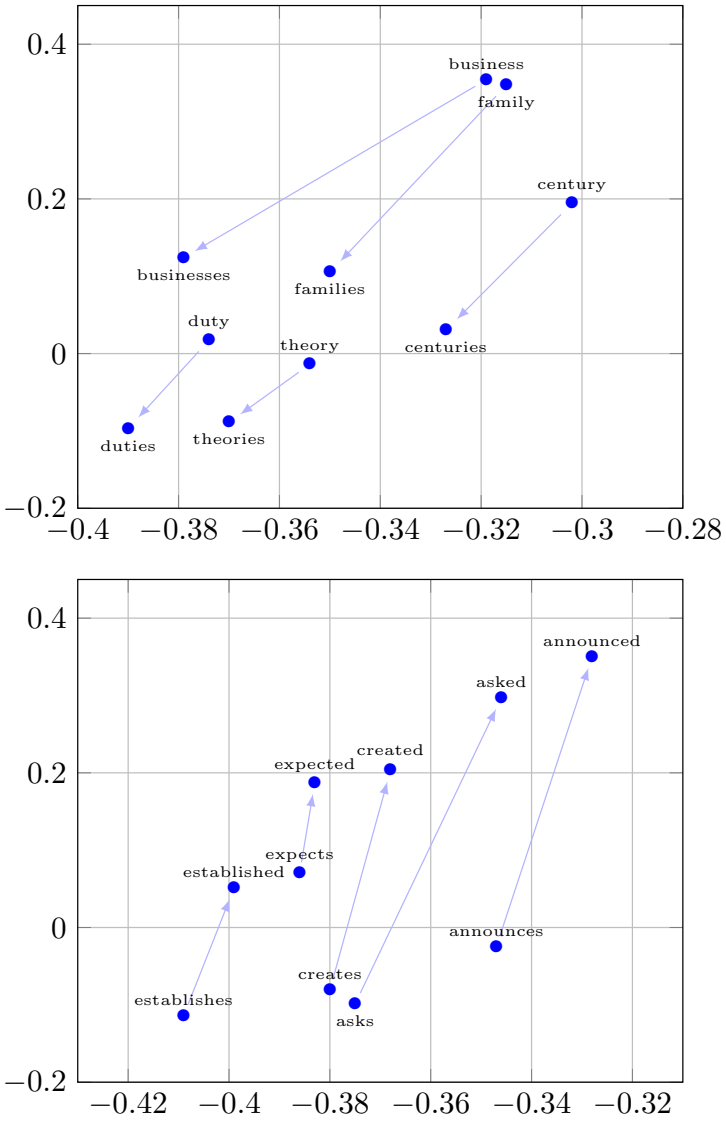


Abbildung 6.12: Zweidimensionale Darstellung von Wortvektoren bestimmter Analogiekategorien (I02-[noun-plural-irreg] oben und I10-[verb-3pSg-Ved] unten)

für alle Klassen gleich gut geeignet, wie wir u.a. bei den Synonymen oder den Gegensätzen des BATS-Datensatzes gesehen haben.

Ein Wort kann natürlich auch mehrere Analogien kodieren, was in Grafik 6.13 zu beobachten ist. Während die Beziehungen zwischen den femininen und den maskulinen Substantiven durch die

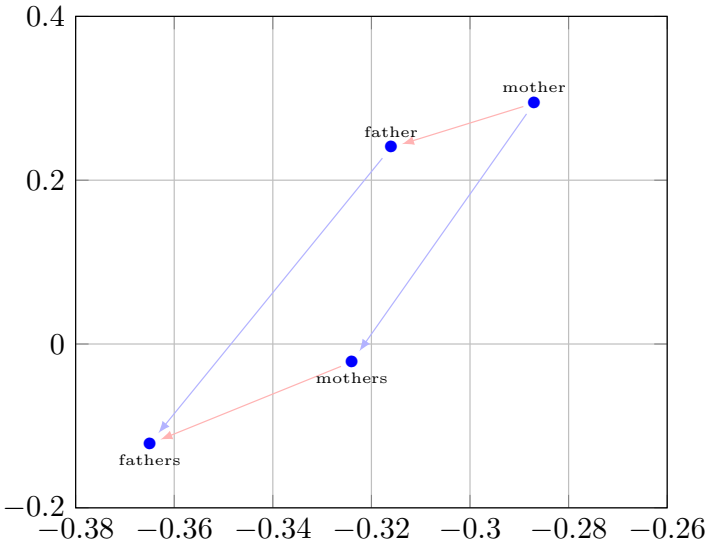


Abbildung 6.13: Zweidimensionale Darstellung von Wortvektoren, die zwei Analogiearten kodieren

roten Pfeile gekennzeichnet sind, werden die Zusammenhänge zwischen Singular und Plural durch die blauen Pfeile beschrieben.

Beispiele arithmetischer Berechnungen mit Wortvektoren entsprechend (5.2) sind in Tabelle 6.10 aufgelistet, wobei wieder die fünf nächsten Nachbarn angegeben worden sind. Das gesuchte Zielwort sollte sich dann möglichst weit oben auf der Nachbarschaftsliste befinden. Auch hier stimmen die

Analogie	
king : queen ; father :	**mother**, friend, daughter, husband, became
is : was ; are :	**were**, in, their, by, when
berlin : germany ; paris :	**france**, less, yet, standard, at
merkel : germany ; putin :	vladimir, **russia**, britain, country, russian
song : songs ; friend :	**friends**, lover, why, people, someone

Tabelle 6.10: Fünf nächsten Nachbarn beim Rechnen mit Wortvektoren entsprechend (5.2) in absteigender Reihenfolge

Ergebnisse mit den Erwartungen überein.

Kapitel 7

Zusammenfassung und Ausblick

In der vorliegenden Masterarbeit haben wir uns ausführlich mit der Thematik der Wortvektoren auseinandergesetzt. Es wurden zwei Modelle eingeführt und in entsprechenden Experimenten gegenübergestellt und ausgewertet. Dabei haben wir gesehen, dass bereits ein Training auf vergleichsweise kleinen Textmengen recht gute und intuitiv sinnvolle Ergebnisse liefert. Es ist also möglich, die semantische Bedeutung von einzelnen Wörtern mit einer Menge von Zahlen zu erfassen. Weiterhin wurde die Approximation der Softmax-Funktion durch Nutzung der NCE mathematisch detailliert ausgeführt.

Es existieren jedoch noch eine Reihe an offener Punkte und weiterer Probleme. So haben wir lediglich zwei Wortvektor-Modelle verwendet. Neben diesen bestehen allerdings noch weiter Möglichkeiten, um den Sinn von Wörtern zu kodieren. Es hat sich gezeigt, dass frühere Count-based-Modelle, die z.B. die Einbettungen mit Hilfe einer *Singulärwertzerlegung* der Co-Occurrence-Matrix generieren, nicht prinzipiell gegenüber unseren Systemen, die ein Optimierungsproblem lösen, benachteiligt sind (vgl. [20] und [26]).

Weiterhin fand die Evaluierung im Rahmen dieser Arbeit nur auf zwei intrinsischen Tasks statt, die ihrerseits eine Reihe an Kritikpunkten besitzen. Inwiefern sich die Performance der Vektoren auf einer nachgelagerten Aufgabestellung des NLP verhält, haben wir hier außer Acht gelassen, was aber gerade in Bezug auf explizite Anwendungen von besonderem Interesse ist.

Zudem ist noch die Frage offen geblieben, wie eine formelle Begründung aussieht, die darlegt, weshalb die Modelle eigentlich funktionieren und tatsächlich nützliche Vektoren liefern. Es gibt eine Vielzahl an Autoren, die versucht haben, sich einer entsprechenden Antwort darauf anzunähern (vgl. [1], [2], [21], [25] und [28]). So ist es insbesondere sehr rätselhaft, wieso die Vektoren eine gewisse lineare Struktur der Sprache erfasst haben, mit der sich dann Analogien lösen lassen. Der Versuch einer mathematischen Erklärung dieses Phänomens wurde u.a. in [15] unternommen.

Aufgrund der Popularität des Skip-Gram-Modells wurden in den letzten Jahren mehrere Erweiterungen und Verbesserungen entwickelt. Es bildet z.B. die Grundlage für ein System (vgl. [7]), welches die Einbettungen auf Basis von N-Grammen auf Buchstabenebene erstellt. Dies ermöglicht es dann, Vektoren für Wörter zu berechnen, die nicht in der Trainingsmenge enthalten waren, die sich aber aus betrachteten N-Grammen zusammensetzen. Dies ist vor allem für Sprachen mit einem großen Vokabular vorteilhaft (z.B. für die deutsche Sprache).

Wenn wir in der Lage sind, Vektoren für einzelne Wörter zu erzeugen, stellt sich abschließend die Frage, ob dies auch für ganze Sätze oder gar Textabschnitte möglich ist. Dies ist die direkt an den Wortvektoren anschließende Problemstellung, welche in der Vergangenheit bereits sehr vielversprechend untersucht wurde (vgl. [23]).

Literaturverzeichnis

[1] C. Allen, I. Balažević und T. Hospedales, *What the Vec? Towards Probabilistically Grounded Embeddings*, arXiv preprint arXiv:1805.12164, 2018

[2] S. Arora, Y. Li, Y. Liang, T. Ma und A. Risteski, *A latent variable model approach to pmi-based word embeddings*, Transactions of the Association for Computational Linguistics 4: 385-399, 2016

[3] A. Bakarov, *A Survey of Word Embeddings Evaluation Methods*, arXiv preprint arXiv:1801.09536, 2018

[4] M. Baroni, G. Dinu und G. Kruszewski, *Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors*, Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Vol. 1, 2014

[5] Y. Bengio, R. Ducharme, P. Vincent und C. Jauvin, *A neural probabilistic language model*, Journal of machine learning research: 1137-1155, 2003

[6] Y. Bengio und J.-S. Senécal, *Quick Training of Probabilistic Neural Nets by Importance Sampling*, AISTATS, 2003

[7] P. Bojanowski, E. Grave, A. Joulin und T. Mikolov, *Enriching word vectors with subword information*, arXiv preprint arXiv:1607.04606, 2016

[8] E. Bruni, N. K. Tran und M. Baroni, *Multimodal distributional semantics*, Journal of Artificial Intelligence Research 49: 1-47, 2014

[9] X. Che, N. Ring, W. Raschkowski, H. Yang und C. Meinel, *Traversal-free word vector evaluation in analogy space*, Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP, 2017

[10] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn und T. Robinson, *One billion word benchmark for measuring progress in statistical language modeling*, arXiv preprint arXiv:1312.3005, 2013

[11] C. Dyer, *Notes on noise contrastive estimation and negative sampling*, arXiv preprint arXiv:1410.8251, 2014

[12] M. Faruqui, Y. Tsvetkov, P. Rastogi und C. Dyer, *Problems with evaluation of word embeddings using word similarity tasks*, arXiv preprint arXiv:1605.02276, 2016

[13] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman und E. Ruppin, *Placing Search in Context: The Concept Revisited*, ACM Transactions on Information Systems, 20(1):116-131, 2002

[14] B. J. Frey und D. Dueck, *Clustering by passing messages between data points*, science 315.5814: 972-976, 2007

[15] A. Gittens, D. Achlioptas und M. W. Mahoney, *Skip-gram-zipf+ uniform= vector additivity*, Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2017

[16] A. Gladkova, A. Drozd und S. Matsuoka, *Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn't*, Proceedings of the NAACL Student Research Workshop, 2016

[17] J. Goldberger und O. Melamud, *Self-Normalization Properties of Language Modeling*, arXiv preprint arXiv:1806.00913, 2018

[18] M. Gutmann und A. Hyvärinen, *Noise-contrastive estimation: A new estimation principle for unnormalized statistical models*, Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010

[19] G. Halawi, G. Dror, E. Gabrilovich und Y. Koren, *Large-scale learning of word relatedness with constraints*, Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2012

[20] F. Hill, R. Reichart und A. Korhonen, *Simlex-999: Evaluating semantic models with (genuine) similarity estimation*, Computational Linguistics 41.4: 665-695, 2015

[21] S. S. Keerthi, T. Schnabel und R. Khanna, *Towards a better understanding of predict and count models*, arXiv preprint arXiv:1511.02024, 2015

[22] B. Laasch, *Vergleich von Gradientenabstiegsverfahren zum Training von Neuronalen Netzen*, Universität Rostock, Bachelorarbeit, 2016

[23] Q. Le und T. Mikolov, *Distributed representations of sentences and documents*, International Conference on Machine Learning, 2014

[24] O. Levy und Y. Goldberg, *Linguistic regularities in sparse and explicit word representations*, Proceedings of the eighteenth conference on computational natural language learning, 2014

[25] O. Levy und Y. Goldberg, *Neural word embedding as implicit matrix factorization*, Advances in neural information processing systems, 2014

[26] O. Levy, Y. Goldberg und I. Dagan, *Improving distributional similarity with lessons learned from word embeddings*, Transactions of the Association for Computational Linguistics 3: 211-225, 2015

[27] C. McCormick, *Word2Vec Tutorial - The Skip-Gram Model*, Retrieved from http://www.mccormickml.com, 2016

[28] O. Melamud und J. Goldberger, *Information-theory interpretation of the skip-gram negative-sampling objective function*, Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 2017

[29] T. Mikolov, K. Chen, G. Corrado und J. Dean, *Efficient estimation of word representations in vector space*, arXiv preprint arXiv:1301.3781, 2013

[30] T. Mikolov, Q. V. Le und I. Sutskever, *Exploiting similarities among languages for machine translation*, arXiv preprint arXiv:1309.4168, 2013

[31] T. Mikolov, I. Sutskever, K. Chen, G. Corrado und J. Dean, *Distributed representations of words and phrases and their compositionality*, Advances in neural information processing systems, 2013

[32] T. Mikolov, W. Yih und G. Zweig, *Linguistic regularities in continuous space word representations*, Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2013

[33] D. Mimno und L. Thompson, *The strange geometry of skip-gram with negative sampling*, Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017

[34] A. Mnih und K. Kavukcuoglu, *Learning word embeddings efficiently with noise-contrastive estimation*, Advances in neural information processing systems, 2013

[35] A. Mnih und Y. W. Teh, *A fast and simple algorithm for training neural probabilistic language models*, arXiv preprint arXiv:1206.6426, 2012

[36] J. Pennington, R. Socher und C. Manning, *Glove: Global vectors for word representation*, Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014

[37] A. Rogers, A. Drozd und B. Li, *The (Too Many) Problems of Analogical Reasoning with Word Vectors*, Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (* SEM 2017), 2017

[38] S. Ruder, *On word embeddings - Part 1*, http://ruder.io/word-embeddings-1, 2016

[39] S. Ruder, *On word embeddings - Part 2: Approximating the Softmax*, http://ruder.io/ word-embeddings-softmax, 2016

[40] S. Ruder, *On word embeddings - Part 3: The secret ingredients of word2vec*, http://ruder.io/secret-word2vec, 2016

[41] M. Sahlgren, *The distributional hypothesis*, Italian Journal of Disability Studies 20: 33-53, 2008

[42] T. Schnabel, I. Labutov, D. Mimno und T. Joachims, *Evaluation methods for unsupervised word embeddings*, Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015

[43] L. Weng, *Learning Word Embedding*, https://lilianweng.github.io/lil-log/2017/10/15/ learning-word-embedding.html, 2017

[44] B. Zoph, A. Vaswani, J. May und K. Knight, *Simple, fast noise-contrastive estimation for large rnn vocabularies*, Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016

Abbildungsverzeichnis

Tabellenverzeichnis

Selbstständigkeitserklärung

Hiermit erkläre ich, die vorliegende Masterarbeit selbstständig und nur unter Verwendung der angegebenen Literatur angefertigt zu haben.

Rostock, den 3. September 2018

Bastian Marc Laasch