technische universität
dortmund

Masterarbeit

Semantisches Word Spotting

Oliver Tüselmann
Oktober 2019

Gutachter:
Prof. Dr.-Ing. Gernot A. Fink
Philipp Oberdieck, M.Sc.

Fakultät für Informatik
Technische Universität Dortmund
http://www.cs.uni-dortmund.de

INHALTSVERZEICHNIS

ABKÜRZUNGSVERZEICHNIS

AP	Average Precision
CBOW	Continous Bag-of-Words
CharLSTM	Character-Aware Neural Language Model
CharCNN	Character-Level Convolutional Neural Network
CNN	Convolution Neural Network
DCT	Discrete Cosine Transformation
DCToW	Discrete Cosine Transform of Words
DTW	Dynamic Time Warping
FP	False Positive
GW	George Washington
GWW	George Washington Writings
HMM	Hidden Markov Model
IAM-DB	IAM offline Handwriting Database
LSA	Latent Semantic Analysis
LSTM	Long Short-Term Memory
MAP	Mean Average Precision
MLP	Multi-Layer Perceptron
MSV	Mean Semantic Value
NLL	Negative Log-Likelihood
NLP	Natural Language Processing
NN	Neural Network
OCR	Optical Character Recognition
PHOC	Pyramidal Histogram of Characters
QbE	Query-by-Example
QbS	Query-by-String

ReLU	Rectified Linear Unit
ResNet	Residual Network
RNN	Recurrent Neural Network
RNN-LM	Recurrent Neural Network Language Model
SLV	Semantic List Value
SPP	Spatial Pyramid Pooling
SV	Semantic Value
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TP	True Positive
TPP	Temporal Pyramid Pooling

Typ	**Beispiel**	**Notation**
Konstante	A, B	Großbuchstaben
Variable, Skalar	a, b, a_i, b_i	Kleinbuchstaben
Vektor	$\mathbf{a}, \mathbf{b}$	Kleinbuchstaben, **fett**
Matrix	$\mathbf{A}, \mathbf{B}$	Großbuchstaben, **fett**
Parameter	α, η	Griechische Buchstaben, klein
Funktion	$f, g(x), h[x]$	Kleinbuchstaben, kursiv
Menge	$\mathcal{A}, \mathcal{B}$	Skriptbuchstaben
Matrixform	$m \times n$	Zeilen $\times$ Spalten
Zahlenbereiche, Koordinatenräume	$\mathbb{N}, \mathbb{R}, \mathbb{R}^2$	Doppelt gestrichen
Skalarprodukt	$\mathbf{a} \cdot \mathbf{b}$	$z = \mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{n} x_i \cdot y_i$ mit $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$
Matrix-Vektor-Produkt	$\mathbf{A} \cdot \mathbf{b}$	$\mathbf{z} = \mathbf{X} \cdot \mathbf{y} \rightarrow z_i = \sum_{j=1}^{m} x_{i,j} \cdot y_j$ für $i \in \{1, ..., n\}$ mit $\mathbf{X} \in \mathbb{R}^{n \times m}, \mathbf{y} \in \mathbb{R}^m$ und $\mathbf{z} \in \mathbb{R}^n$
Elementweise Multiplikation	$\mathbf{a} \odot \mathbf{b}$	$\mathbf{z} = \mathbf{x} \odot \mathbf{y} \rightarrow z_i = x_i \cdot y_i$ für $i \in \{1, ..., n\}$ mit $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^n$
Skalare Multiplikation	$a \cdot b$	$z = x \cdot y$
Frobenius Skalarprodukt	$\langle \mathbf{A}, \mathbf{B} \rangle$	$z = \langle \mathbf{X}, \mathbf{Y} \rangle = \sum_{i=1}^{m} \sum_{j=1}^{n} x_{i,j} \cdot y_{i,j}$ mit $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{m \times n}$
P-Norm	$\|\mathbf{a}\|_p, \|\mathbf{b}\|_p$	$z = \|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} \|x_i\|^p\right)^{1/p}$ mit $\mathbf{x} \in \mathbb{R}^n$, $p \in \mathbb{R}$ und $1 \leqslant p < \infty$

1 EINLEITUNG

Es existiert weltweit eine große Anzahl an nicht digitalisierten Dokumenten in Büchereien und Institutionen. Um die Informationen aus diesen Dokumenten zu sichern und maschinell verarbeitbar zu machen, werden diese digitalisiert und in Bibliotheken verwaltet. Für eine effiziente Verwendung dieser Bibliotheken wird eine geeignete Suchfunktion benötigt, sodass die Dokumente bezüglich einer Suchanfrage gefiltert und sortiert werden können. Einen intuitiven Ansatz bietet die *Optical Character Recognition (OCR)*. Hierbei wird eine Transkription eines digitalen Abbildes durchgeführt. Anschließend wird der erhaltene maschinen-kodierte Text mit Standardverfahren aus der Informatik weiterverarbeitet. Dabei erzielen die OCR Methoden sehr gute Ergebnisse auf modernen und maschinell erstellten Dokumentabbildern, mit einfachen Layouts und bekannten Schriftarten [HK16]. Diese Methoden scheitern jedoch häufig bei der automatischen Erkennung von schlecht gedruckten Texten, historischen Dokumenten und speziell bei handschriftlichen Dokumentabbildern [RM07] [RRF13]. Aus diesem Grund wird bei dieser Art von Dokumenten häufig auf das *Word Spotting* zurückgegriffen. Dies ist eine deutlich flexiblere Methode, bei der ein Benutzer eine Suchanfrage an das System stellen kann und eine sortierte Liste mit den Positionen der ähnlichsten Vorkommen in den Dokumenten erhält. Dabei basiert das System nicht auf einer Transkription der Dokumente, sondern eher auf den Ähnlichkeiten zwischen der Anfrage und den Dokumentausschnitten. Wilkinson et al. haben mit Ihrer Veröffentlichung [WB16], einen state-of-the-art Ansatz für das segmentierungsbasierte Word Spotting vorgestellt. Diese Arbeit stellt zudem, mit dem semantischen Word Spotting, eine Erweiterung des Ansatzes vor, bei der nicht nur visuell, sondern auch semantisch ähnliche Treffer bezüglich der Suchanfrage berücksichtigt werden. Dabei ist es das Ziel des semantischen Word Spottings, dass in der Ergebnisliste zunächst alle Wortabbilder des gesuchten Wortes aufgelistet werden und anschließend semantisch ähnliche Wortabbilder bezüglich der Anfrage folgen. Ein möglicher Anwendungsfall für das semantische Word Spotting ist die Suche in einem großen und unbekannten Dokumentenkorpus, wobei der Benutzer noch nicht genau weiß wonach er sucht. Wenn dieser zum Beispiel nach dem Wort *Sonntag* sucht, dann sollte das System zunächst alle Vorkommen des Wortes *Sonntag* auflisten und anschließend weitere Wortabbilder von zum Beispiel anderen Wochentagen.

Die Motivation dieser Arbeit besteht darin, dass System von Wilkinson et al. nachzubauen und die in der Veröffentlichung angegebenen Ergebnisse zu kontrollieren. Zudem ist eine interessante Fragestellung, ob die Ergebnisse für das semantische Word Spotting durch die Verwendung von bewährten Verfahren aus den natürlichsprachlichen Systemen verbessert werden können. Außerdem werden die semantischen Rückgabelisten in der Arbeit von Wilkinson et al. mit der *Mean Average Precision (MAP)* evaluiert. Dies ist jedoch ein fragwürdiges Bewertungsmaß für semantische Rückgabelisten, weshalb in dieser Arbeit ein geeigneteres Maß für die Evaluation dieser Listen ermittelt werden soll.

Diese Arbeit ist wie folgt aufgebaut. Im Kapitel 2 werden die für diese Arbeit benötigten Grundlagen vorgestellt. Dabei wird zunächst detailliert auf das Word Spotting Verfahren eingegangen. Anschließend werden sowohl die Grundlagen der künstlichen neuronalen Netzwerke als auch die in den Word Spotting Ansätzen verwendeten Netzwerkarchitekturen präsentiert. Zudem werden mit dem Word2Vec, FastText und Character-Aware Neural Language Model, die drei zu evaluierenden Worteinbettungen dieser Arbeit vorgestellt. Im Kapitel 3 wird mit dem Attribute-CNN von Sudholt et al. [SF18] und dem Word Spotting System von Wilkinson et al. [WB16] detailliert auf die verwandten Arbeiten eingegangen. Im Kapitel 4 werden zunächst die Implementierungsdetails für den Nachbau des Wilkinson Systems präsentiert und anschließend das Evaluationsverfahren vorgestellt. Zudem wird der Nachbau mit dem beschriebenen Verfahren evaluiert und die ermittelten Resultate mit den Werten aus der Veröffentlichung vergleichen sowie diskutiert. Im Kapitel 5 wird das im Wilkinson verwendete Character-Aware Neural Language Model, durch unterschiedlich trainierte Word2Vec und FastText Modelle ersetzt, evaluiert sowie deren Ergebnisse diskutiert. Dafür wird zunächst eine Analyse der Wortmodelle durchgeführt, um einen Eindruck über die Qualität der semantischen Beziehungen in den jeweiligen Attributräumen zu erhalten. Anschließend wird der Einfluss dieser Räume auf die Erzeugung der Rückgabelisten im Wilkinson System überprüft. Zudem werden die unterschiedlichen semantischen Modelle in diesem Kapitel bezüglich der MAP bewertet und die Ergebnisse dieser Evaluationen miteinander verglichen sowie diskutiert. Abschließend wird im Kapitel 6, aufgrund der nicht Berücksichtigung von semantischen Informationen bei der MAP, ein neues Bewertungsmaß für die Evaluation von den Rückgabelisten beim semantischen Word Spotting vorgestellt und evaluiert.

2 GRUNDLAGEN

In diesem Kapitel werden die grundlegenden Modelle und Methoden dieser Arbeit vorgestellt. Im Kapitel 2.1 wird zunächst die Verwendung der Word Spotting Methode motiviert. Außerdem werden in dem Kapitel die wichtigsten Arbeiten aus diesem Bereich beschrieben. Diese Arbeit beschäftigt sich intensiv mit zwei Word Spotting Systemen, welche auf neuronalen Netzwerken basieren. Aus diesem Grund wird im Kapitel 2.2 sowohl auf die Grundlagen von neuronalen Netzwerken als auch auf zwei für diese Arbeit wichtige Netzwerkarchitekturen eingegangen. Da in dieser Arbeit außerdem das semantische Word Spotting untersucht wird, werden in dem Kapitel 2.3, die für diese Arbeit relevanten, semantischen Modelle vorgestellt.

2.1 WORD SPOTTING

Wie bereits in der Einleitung beschrieben, wird bei der automatischen Erkennung von schlecht gedruckten Texten, historischen Dokumenten und speziell bei handschriftlichen Dokumentabbildern häufig auf das Word Spotting zurückgegriffen. Dies ist im Vergleich zur OCR eine flexiblere Methode, welche zuerst von [MHRC96] veröffentlicht wurde. Bei dieser Methode stellt ein Benutzer eine Suchanfrage an das Word Spotting System, welches anschließend die Ähnlichkeiten von der Anfrage zu allen gespeicherten Dokumentabbildern im System berechnet und abschließend eine sortierte Liste mit den ähnlichsten Wortabbildern bezüglich der Anfrage zurückgibt. Die Ähnlichkeit basiert dabei vollständig auf festgelegten beziehungsweise gelernten Merkmalen, wie zum Beispiel Farbe, geometrische Form oder Textur. Somit wird anstelle einer Transkription des Wortabbildes, eine nach der Ähnlichkeit zur Anfrage sortierte Liste, mit den gespeicherten Wortabbildern aus dem System zurückgegeben. Daher kann die Word Spotting Methode zu dem *Information Retrieval* Bereich zugeordnet werden.

Die publizierten Word Spotting Methoden unterscheiden sich bezüglich der Anwendung stark voneinander und können auf Grundlage mehrerer Kriterien gruppiert werden. Ein Kriterium ist, wie der Benutzer Anfragen an das System stellen kann. Dabei sind *Query-by-String (QbS)* und *Query-by-Example (QbE)* zwei viel verwendete Ansätze. Beim QbS Szenario gibt der Benutzer die Anfrage als Text-String an,

wohingegen beim QbE ein beispielhaftes Wortabbild des zu suchenden Wortes als Anfrage dient. Ein klarer Vorteil von QbS gegenüber QbE ist, dass der Anwender nicht erst ein Bild seiner Suchanfrage finden muss, bevor er danach suchen kann. Der Nachteil von QbS ist jedoch, dass eine Abbildung von Strings zu einer systemabhängigen Repräsentation benötigt wird. Einen interessanten Ansatz bieten dabei die *Attribut-Repräsentationen* [AGFV14]. Hierbei können aufgrund der gemeinsamen Darstellungsmöglichkeit von Wortabbildern und Zeichenketten, sowohl Anfragen mittels QbS als auch QbE gestellt werden. Des Weiteren kann beim Word Spotting zwischen segmentierungsfreien und segmentierungsbasierten Ansätzen unterschieden werden [GSGN17]. Beim segmentierungsbasierten Word Spotting arbeitet das System nicht auf dem Dokument als Ganzes, sondern auf vorsegmentierten Wortabbildern. Dazu wird in einem Vorverarbeitungsschritt jedes Dokumentenabbild in eine Liste von segmentierten Wortabbildern überführt. Ein weiteres Kriterium ist die Unterscheidung zwischen einer traditionellen und semantischen Suchanfrage. Bei der traditionellen Anfrage wird lediglich nach visuell ähnlichen Wortabbildern im Korpus gesucht. Eine Erweiterung bietet die semantische Suche, bei der nicht nur visuell, sondern auch semantisch ähnliche Treffer bezüglich der Suchanfrage berücksichtigt werden. Mit dieser Erweiterung beschäftigen sich bisher nur wenige Arbeiten [GAMP15] [WB16]. Eine der ersten Arbeiten zum segmentierungsbasierten Word Spotting stammt von [MHRC96]. Hierbei werden die Dokumente zunächst in Graustufenbilder überführt und auf diesen eine Binarisierung mittels Schwellwert angewendet. Die binarisierten Dokumentabbilder werden anschließend segmentiert. Mithilfe einer XOR-Operation wird zwischen dem Anfragebild und einem gespeicherten Wortabbild, jeweils eine Distanz berechnet. Abschließend werden die gespeicherten Wortabbilder nach der minimalen Distanz sortiert und zurückgegeben.

In weiteren Arbeiten zum Thema Word Spotting wurde die Ähnlichkeit zweier Wortabbilder auf Grundlage einer sequenz-basierten Methode berechnet. Dazu wird in [RM07] eine Verteilung der Schriftfarbe entlang einer Achse verwendet, um ein Wort-Profil zu bestimmen. Aufgrund der Variabilität der Handschrift, ist es jedoch sehr unwahrscheinlich, dass zwei Wort-Profile von demselben Wort übereinstimmen. Aus diesem Grund wird *Dynamic Time Warping (DTW)* [SK83] zur robusteren Anwendbarkeit des Ansatzes verwendet. In den Veröffentlichungen von [RSP09] und [FKFB10] konnte gezeigt werden, dass durch die Verwendung von *Hidden Markov Modellen (HMMs)* im Kontext des Word Spottings bessere Resultate erzielt werden konnten als durch das DTW. Weitere Arbeiten aus diesem Bereich beschäftigten sich mit dem Thema *holistische Repräsentationen* mit lokalen Deskriptoren. Dabei werden die Wortabbilder unter anderem als SIFT Merkmale [RATL11], geometrische Merkmale [RM07] oder HOG-basierte Deskriptoren [RATL11] repräsentiert. Eine fundamentale

Arbeit aus dem Bereich des segmentierungsbasierten Word Spottings ist das System aus [AGFV14]. Hierbei wird mit dem *Pyramidal Histogram of Characters (PHOC)* eine Attribut-Repräsentation für das Word Spotting verwendet. Dies ermöglicht eine Ähnlichkeitsbewertung von Strings und Wortabbildern, da diese in denselben Attributraum projiziert werden. Die Berechnung der PHOC-Repräsentation ist für Strings trivial, jedoch nicht für Wortabbilder. Diese werden zunächst mithilfe der *Fisher Vektoren* [PSM10] in einen Attributraum projiziert und anschließend mit mehreren *Support Vektor Maschinen (SVMs)* in die PHOC-Repräsentation überführt. Dafür müssen die SVMs einen Unterraum zwischen Fisher Vektoren und PHOC-Repräsentation lernen. Da die Wortabbilder und Strings in denselben Raum projiziert werden, ist die Berechnung der Ähnlichkeit mithilfe eines geeigneten Distanzmaßes trivial.

In den letzten Jahren haben neuronale Netze in vielen Bereichen der Informatik, speziell in der *Computer Vision*, zu zahlreichen state-of-the-art Ergebnissen geführt. Auch deshalb steigt die Anzahl der auf neuronalen Netzwerken basierenden Veröffentlichungen im Bereich des Word Spottings [KDJ18] [SF16] [WB16]. In [SF16] haben Sudholt et al. im Vergleich zu [AGFV14] die SVMs durch neuronale Netzwerke ersetzt. Wilkinson et al. haben mit [WB16] einen ähnlichen Ansatz zu [SF16] vorgestellt. Dessen System lernt die Attribut-Repräsentationen jedoch nicht nach dem Ende-zu-Ende Prinzip, sondern bestimmt diese in einem zweistufigen Prozess. Dabei werden zunächst Bilddeskriptoren für das eingehende Wortabbild mittels eines *Triplet-CNNs* [BJTM16] berechnet. Anschließend werden auf Grundlage dieser Deskriptoren die Attribut-Repräsentationen bestimmt.

Für eine detaillierte und umfassende Betrachtung der Word Spotting Methode siehe [GSGN17].

2.2 KÜNSTLICHE NEURONALE NETZWERKE

Wie im letzten Kapitel bereits beschrieben, haben künstliche neuronale Netzwerke in vielen Bereichen der Informatik, wie auch beim Word Spotting, zu zahlreichen state-of-the-art Ergebnissen geführt. Da die in dieser Arbeit verwendeten Word Spotting Systeme auf neuronale Netzwerke basieren, werden in dem Unterkapitel 2.2.1 zunächst die Grundlagen der künstlichen neuronalen Netzwerke vorgestellt. Darauf aufbauend wird in den Unterkapiteln 2.2.2 und 2.2.3 detailliert auf relevante Netzwerkarchitekturen für diese Arbeit eingegangen.

2.2.1 *Grundlagen*

Ein künstliches neuronales Netz ist bis zu einem gewissen Grad dem Aufbau des biologischen Gehirns nachempfunden [JMM96]. Es besteht aus einem abstrahierten Modell miteinander verbundener Neuronen. Der Aufbau und die Fähigkeiten eines einzelnen Neurons werden in dem Kapitel zu den *Perzeptrons* beschrieben. Es wird ersichtlich, dass ein einzelnes Neuron keine komplexen Aufgaben erledigen kann. Die Anordnung dieser Neuronen in hierarchische Schichten und deren Vernetzung, führt zu deutlich leistungsfähigeren Modellen. Diese Modelle werden als *Multi-Layer Perzeptrons (MLPs)* bezeichnet und im gleichnamigen Kapitel beschrieben. Abschließend wird das Training eines MLPs im Kapitel zum Gradientenabstieg vorgestellt.

Perzeptron

Der erste Ansatz für ein künstliches Neuron stammt aus dem Jahr 1943 von Warren McCulloch und Walter Pitts [MP43]. Dabei handelt es sich um ein simples mathematisches Modell, welches n binäre Signale als Eingabe erhält und diese in einem zweistufigen Prozess verarbeitet. Im ersten Schritt werden die binären Eingaben aufsummiert und anschließend in eine nicht lineare Funktion überführt, welche abhängig von einem festgelegten Schwellwert eine null oder eins ausgibt. Das McCulloch-Pitts Neuron mit dem Schwellwert θ und den Eingaben $x_1, x_2, ..., x_n$ kann formal mithilfe der folgenden Formel beschrieben werden:

$$f(x_1, x_2, ..., x_n) = \begin{cases} 1 & \text{wenn } \sum_{i=1}^{n} x_i \geqslant \theta \\ 0 & \text{sonst} \end{cases} \tag{2.2.1}$$

Dieses simple mathematische Modell ist in der Lage, die logischen Funktionen *AND*, *OR* und *NOT* zu realisieren. Damit kann gezeigt werden, dass jede logische Funktion $F : \mathbb{B}^n \to \mathbb{B}$ mit einem zweistufigen McCulloch-Pitts Netzwerk simuliert werden kann. Frank Rosenberg erweiterte in seiner Arbeit [Ros58] das McCulloch-Pitts Neuron um gewichtete Eingaben. Dieses Modell wird in der Literatur als Perzeptron bezeichnet und wurde von Minsky und Papert in [MP69] weiter generalisiert. Die fundamentalste Änderung im Vergleich zum Rosenberg Perzeptron ist die Verwendung

von reellwertigen Zahlen für die Eingabesignale. Das Perzeptron kann formal mithilfe der folgenden Formel beschrieben werden:

$$f(x_1, x_2, ..., x_n) = \begin{cases} 1 & \text{wenn } \sum_{i=1}^{n} w_i \cdot x_i + b > 0 \\ -1 & \text{sonst} \end{cases} \tag{2.2.2}$$

Dabei ist $w_i \in \mathbb{R}$ das Gewicht für Eingabe $x_i \in \mathbb{R}$ und $b \in \mathbb{R}$ der Schwellwert. Abbildung 2.2.1 zeigt eine Visualisierung des Perzeptrons.

In der Arbeit von Rosenberg [Ros58] wurde zudem ein Lern-Algorithmus für das Perzeptron vorgestellt. Das Hauptziel des Algorithmus ist das Finden von Gewichten, sodass die positiven und negativen Trainingsdaten vollständig voneinander getrennt werden können. Dabei haben die positiven Beispiele einen Wert von 1 und die negativen einen Wert von -1. Die generelle Vorgehensweise des Algorithmus wird im Folgenden beschrieben. Jedes Beispiel $\mathbf{x} = x_1, x_2, ..., x_n$ der Trainingsmenge und das zugehörige Label $y \in \{-1, 1\}$ wird von dem Perzeptron individuell und iterativ verarbeitet. Wenn für das Beispiel $\mathbf{x}$ gilt, dass

$$\left(\sum_{i=1}^{n} w_i \cdot x_i + b \right) \cdot y > 0 \tag{2.2.3}$$

erfüllt ist, werden die Gewichte nicht verändert und es wird mit dem nächsten Beispiel fortgefahren. Andernfalls wird das Gewicht $w_i^{(t)}$ im Iterationsschritt t wie folgt aktualisiert:

$$w_i^{(t)} \leftarrow w_i^{(t-1)} + y \cdot x_i \tag{2.2.4}$$

Der Algorithmus konvergiert, wenn alle Trainingsdaten korrekt klassifiziert wurden. Die Konvergenz des Algorithmus kann bei linear separierbaren Trainingsdaten garantiert werden [MP69][Ros58]. Damit ist ein Perzeptron in der Lage, ein Zwei-Klassenproblem zu lösen.

Multi-Layer Perzeptron

Minsky und Papert haben in Ihrer berühmten Veröffentlichung [MP69] gezeigt, dass ein einzelnes Perzeptron nicht in der Lage ist, die XOR Funktion zu realisieren. Außerdem ist es mit einem Perzeptron nicht möglich, nicht linear separierbare Daten zu verarbeiten. In [MP69] konnte jedoch auch gezeigt werden, dass diese Limitierungen durch eine Stapelung von mehreren Perzeptron Schichten umgangen werden kann.

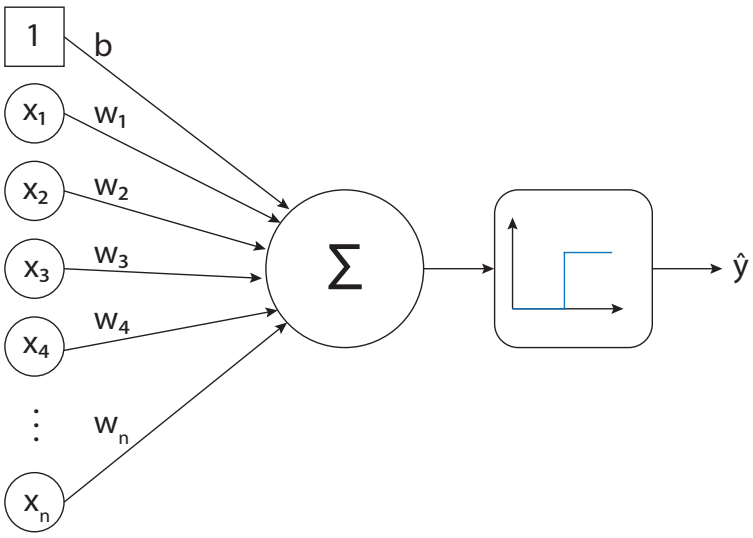

Abbildung 2.2.1: Das Perzeptron Modell nach Minsky und Papert [MP69]. Das Modell erhält einen reellwertigen Vektor als Eingabe und berechnet damit eine gewichtete Summe, welche abschließend mithilfe einer Treppenfunktion die Ausgabe $\hat{y}$ bestimmt. Quelle: [Sud18].

Diese Überlegungen führten zu dem *Multi-Layer Perzeptron*. Die Ausdruckskraft eines MLPs im Vergleich zu einem einzelnen Perzeptron kann anhand der Abbildung 2.2.2 nachvollzogen werden.

Ein MLP ist ein künstliches neuronales Netzwerk aus der Klasse der *feedforward* Netzwerke. Es besteht aus einer Eingabe-, mindestens einer Zwischen- sowie einer Ausgabeschicht. Die einzelnen Schichten enthalten jeweils Neuronen. Zwischen den Schichten ist jedes Neuron einer Schicht immer mit allen Neuronen der nächsten Schicht verbunden. Es gibt keine Verbindungen zur vorherigen Schicht und keine Verbindungen, die eine Schicht überspringen. Zudem sind die Verbindungen zwischen den Neuronen unidirektional und nicht zyklisch. Beginnend mit der Eingabeschicht fließen die Eingabedaten über eine oder mehrere Zwischenschichten bis hin zur Ausgabeschicht. Dabei ist die Ausgabe des einen Neurons, die Eingabe des nächsten. Abbildung 2.2.3 zeigt ein beispielhaftes MLP zur Lösung des XOR Problems.

Im Folgenden wird die Funktionsweise eines MLPs formal betrachtet. Die Berechnung der Ausgabe für die Schicht s in einem MLP ist ähnlich zu den Berechnungen in

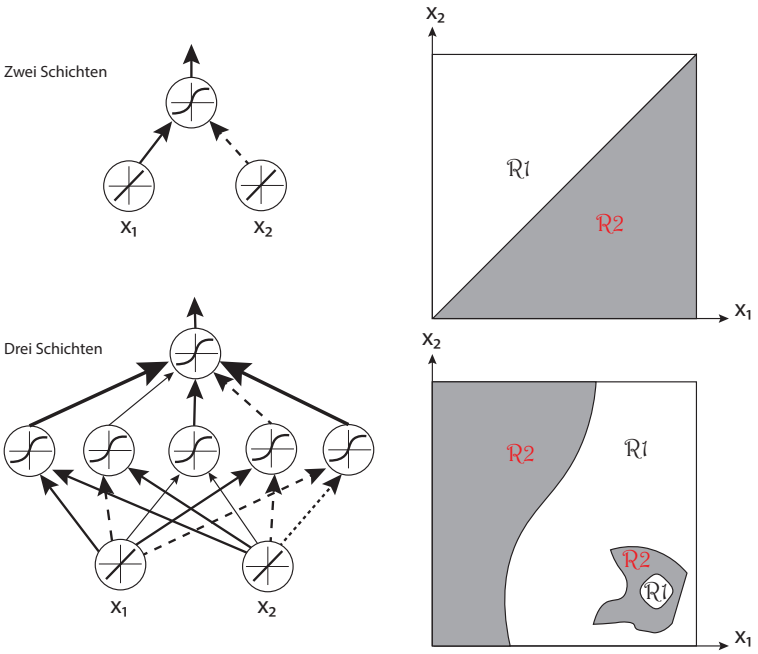

Abbildung 2.2.2: Ein einzelnes Perzeptron (obere Abbildung) kann lediglich eine lineare Trennebene simulieren. Ein Netzwerk mit einer angemessenen Anzahl an Zwischenschichten (untere Abbildung) kann hingegen jede Entscheidungsregel beliebig genau approximieren. Quelle: [DHSoo].

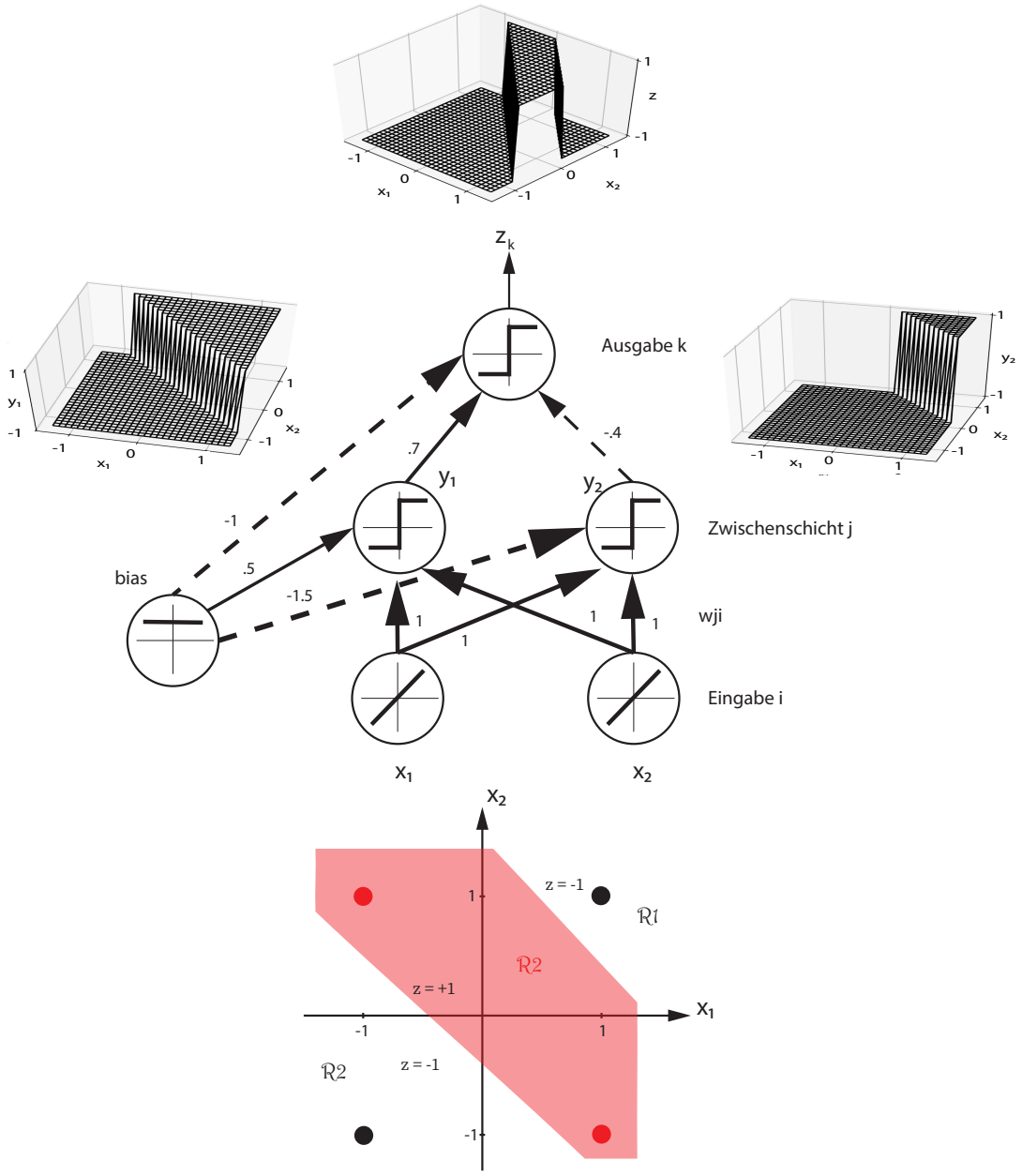

Abbildung 2.2.3: Beispielhafte Visualisierung eines minimalen Multi-Layer Perzeptrons zur Lösung des XOR Problems. In dem unteren Bereich der Grafik befindet sich der zweidimensionale Merkmalsraum mit den zu klassifizierenden Punkten. In der Mitte befindet sich das dreischichtige Netzwerk zur Lösung des XOR Problems. Dabei gibt es zwei Eingabeneuronen, zwei Neuronen in der Zwischenschicht und ein Ausgabeneuron. Die Pfeile zwischen den einzelnen Neuronen repräsentieren die Gewichte. Dabei stehen die gestrichelten Linien für negative und die durchgängigen Linien für positive Gewichte. Die Aktivierungsfunktion der Eingabeneuronen ist linear, wohingegen die Zwischen- und Ausgabeneuronen eine Stufenfunktion verwenden. Die Visualisierungen links und rechts neben dem Netzwerk repräsentieren jeweils die generierten Funktionen der Neuronen in der Zwischenschicht. Der 3D Plot im oberen Bereich der Grafik zeigt die gelernte Entscheidungsregel des Netzwerks. Quelle: [DHS00].

einem Perzeptron. Dazu wird zunächst eine gewichtete Summe der Eingabe $\mathbf{x}$ für die Schicht s berechnet:

$$\mathbf{i}^{(s)} = i^{(s)}\left(\mathbf{x}, \mathbf{W}^{(s)}, \mathbf{b}^{(s)}\right) = \mathbf{W}^{(s)} \cdot \mathbf{x} + \mathbf{b}^{(s)} \tag{2.2.5}$$

Dabei ist $\mathbf{W}^{(s)}$ die Gewichtsmatrix der Schicht s und hat die Form: *(Anzahl Neuronen in der Schicht* s $\times$ *Anzahl Neuronen in der Schicht* $s-1$*)*. Somit beschreibt $\mathbf{W}^{(s)}_{i,j}$ das Kantengewicht von dem Neuron j in Schicht $s-1$ zu dem Neuron i in Schicht s. Zudem repräsentiert $\mathbf{b}^{(s)}$ die Bias-Werte der Neuronen in Schicht s. Somit ist $\mathbf{b}^{(s)}_i$ der Bias-Wert für das Neuron i in Schicht s. Auf die gewichtete Summe der Eingabe für Schicht s, $\mathbf{i}^{(s)}$, wird anschließend eine nicht lineare, elementweise Aktivierungsfunktion θ angewendet

$$\mathbf{f}^{(s)} = f^{(s)}\left(\mathbf{x}, \mathbf{W}^{(s)}, \mathbf{b}^{(s)}\right) = \theta\left(i^{(s)}\left(\mathbf{x}, \mathbf{W}^{(s)}, \mathbf{b}^{(s)}\right)\right) \tag{2.2.6}$$

und damit die Ausgabe $\mathbf{f}^{(s)}$ der Schicht s berechnet. Dabei ist es fundamental, dass die Aktivierungsfunktion nicht linear ist, da das MLP ansonsten nicht ausdrucksfähiger als ein linearer Klassifikator wäre [DHS00, S.307]. Zudem wird in einem MLP die Stufenfunktion des Perzeptrons traditionell durch eine Sigmoid Funktion

$$\theta(x) = sigm(x) = \frac{1}{1 + exp(-x)} \tag{2.2.7}$$

ersetzt. Dies ist ein wichtiger Aspekt für das Training des MLPs, da diese Funktion überall differenzierbar ist. Die Ausgabe $\hat{\mathbf{y}}$ des MLPs wird mit dem sogenannten *forward pass* für die Eingabe $\mathbf{x}$ berechnet:

$$\hat{\mathbf{y}} = f^{(N_l)}\left(f^{(N_{l-1})}\left(...f^{(2)}\left(f^{(1)}\left(\mathbf{x}\right)\right)\right)\right). \tag{2.2.8}$$

Hierbei steht N_l für die Anzahl der Schichten im MLP. Die Gewichte und der Bias-Wert der einzelnen Schichten sind aufgrund der Übersichtlichkeit in der Formel nicht mit angegeben. Anhand der Berechnungen wird ersichtlich, dass für die Ausgabe einer Schicht nur auf die Ausgabe der vorherigen Schicht zurückgegriffen wird.

Mithilfe des *Universal Approximation Theorems* [HSW89] kann zudem bewiesen werden, dass mit einem MLP theoretisch alle reellwertigen multivariaten Funktionen beliebig genau approximiert werden können [Sch14]. Dazu reicht ein minimales MLP mit einer Zwischen- und einer Ausgabeschicht [HSW89]. Dieses Theorem ist in der Praxis jedoch unpraktikabel, da die Anzahl der verborgenen Neuronen gegen unendlich laufen müsste.

Gradientenabstieg

In dem Training für neuronale Netzwerke werden die Parameter des Netzes, also die Gewichte und Bias-Werte gesucht, welche die gewünschten Ausgaben $\mathbf{y}$ zu den Eingaben $\mathbf{x}$ liefern. Die Berechnung dieser Parameter ist jedoch nicht trivial und das Perzeptron Training aus dem vorherigen Kapitel auf MLPs nicht anwendbar. Aus diesem Grund wird ein geeignetes Verfahren zur Bestimmung der Parameter im MLP gesucht.

Die grundlegende Idee ist, dass mithilfe einer Kostenfunktion c die Differenz zwischen der aktuellen Netzausgabe $\hat{\mathbf{y}}$ und der gewünschten Ausgabe $\mathbf{y}$ berechnet wird. Eine intuitive und viel verwendete Kostenfunktion ist die *Euklidische Verlustfunktion*:

$$c(\hat{\mathbf{y}}, \mathbf{y}) = \sum_{i=1}^{d} \frac{1}{2}(\hat{y}_i - y_i)^2 \tag{2.2.9}$$

Dabei ist d die Dimensionalität der Netzwerksausgabe. Das Ziel ist die Ermittlung der Parameter des Netzes, welche die Funktion c minimieren. Leider können die optimalen Parameter im Normalfall und speziell bei tiefen Netzen nicht trivial bestimmt werden. Außerdem ist die Berechnung dieser Parameter aufwendig. Aus diesem Grund wird mit dem Gradientenabstieg, auf ein bekanntes Optimierungsverfahren zur iterativen Minimierung einer Funktion zurückgegriffen. Dieses Prinzip hat sich mittlerweile als Standardverfahren für das Training von neuronalen Netzwerken etabliert [DHSoo, S.288]. Der Algorithmus passt die aktuellen Parameter in jedem Iterationsschritt so an, dass sich einem lokalen Minimum der Kostenfunktion genähert wird. Dafür wird das Gewicht $w_{i,j}^{(s)}$ in Schicht s ein kleines Stück in Richtung des negativen Gradienten der Kostenfunktion, in Bezug zu $w_{i,j}^{(s)}$, verändert:

$$w_{i,j}^{(s)} \leftarrow w_{i,j}^{(s)} - \eta \frac{\partial c}{\partial w_{i,j}^{(s)}} \tag{2.2.10}$$

Dabei ist η die Lernrate und fundamental für den Optimierungsprozess. Wenn die Lernrate zu groß gewählt wird, kann die Optimierung oszillieren oder sogar divergieren. Wird sie jedoch zu gering gewählt, benötigt die Optimierung viele Iterationen für die Konvergenz von dem Algorithmus [DHSoo, S.225]. Die Berechnung der neuen Bias-Werte ist äquivalent zu der Berechnung der Gewichte im Netzwerk.

Die Formel 2.2.10 benötigt die Berechnung des Gradienten $\frac{\partial c}{\partial w_{i,j}^{(s)}}$. Dabei ist die Berechnung dieses Wertes für die Ausgabeschicht trivial, jedoch nicht für die Zwischenschichten. Aufgrund der Formulierung des neuronalen Netzes als Kombination

von Funktionen (siehe Formel 2.2.8), kann der Gradient der Kostenfunktion in Bezug zu dem Gewicht $w_{i,j}^{(s)}$, in Schicht s mithilfe der Kettenregel bestimmt werden:

$$\begin{aligned}\frac{\partial c}{\partial w_{i,j}^{(s)}} &= \frac{\partial c}{\partial f^{(N_l)}} \cdot \frac{\partial f^{(N_l)}}{\partial f^{(N_{l-1})}} \cdot \ldots \cdot \frac{\partial f^{(s+1)}}{\partial f^{(s)}} \cdot \frac{\partial f^{(s)}}{\partial i^{(s)}} \cdot \frac{\partial i^{(s)}}{\partial w_{i,j}^{(s)}} \\ &= \frac{\partial c}{\partial f^{(N_l)}} \cdot \left\{ \prod_{k=s+1}^{N_l} \frac{\partial f^{(k)}}{\partial f^{(k-1)}} \right\} \cdot \frac{\partial f^{(s)}}{\partial i^{(s)}} \cdot \frac{\partial i^{(s)}}{\partial w_{i,j}^{(s)}}\end{aligned} \tag{2.2.11}$$

Die ersten beiden Faktoren aus der letzten Zeile der Formel 2.2.11 beschreiben den Gradienten der Kostenfunktion in Bezug zu der vorherigen Schicht $s+1$. Dieser Gradient wird ausgehend von der Ausgabeschicht berechnet und endet bei der Schicht $s+1$. Dabei werden die Gradienten für die Gewichte der Ausgabeschicht direkt von der Kostenfunktion berechnet (siehe ersten Faktor der Formel 2.2.11). Dieser Fehler wird anschließend an die vorherigen Schichten zurück propagiert. Anhand dieser Formel wird ersichtlich, dass die Gewichte der Zwischenschicht s mithilfe des Gradienten der Schicht $s+1$ aktualisiert werden. Mit den letzten beiden Faktoren aus der letzten Zeile der Formel 2.2.11 wird der Anteil des Gewichtes für den Vorhersagefehler an Schicht s ermittelt.

Der Algorithmus für den Gradientenabstieg berechnet den Gradienten über allen Trainingsdaten. Die Berechnung kann für eine große Trainingsmenge und tiefe Netze lange dauern. Aus diesem Grund wird beim Training von neuronalen Netzen häufig auf den *stochastischen Gradientenabstieg* zurückgegriffen. Dabei wird nur eine kleinere Menge von Trainingsdaten für die Berechnung des Gradienten verwendet. Dies beschleunigt den Trainingsprozess und führt in den meisten Fällen zu guten Schätzwerten der Parameter.

Mit dem Gradientenabstieg kann die Ermittlung des globalen Minimums nicht garantiert werden. Es zeigt sich jedoch anhand der zahlreichen state-of-the-art Ansätze, dass dieses Verfahren in der Praxis zu guten Schätzwerten der Parameter führt.

2.2.2 *Convolutional Neural Network*

Die Verarbeitung von Bilddaten ist mit einem traditionellen feedforward Netzwerk meistens nicht praktikabel. Dies resultiert aus der großen Anzahl an Neuronen in der Eingabeschicht und der daraus folgenden Parameteranzahl im Netzwerk. Da in einem feedforward Netzwerk jedes Neuron aus der Eingabeschicht mit jedem Neuron aus der nachfolgenden Schicht verbunden ist, besitzt das Netzwerk bereits bei kleinen

Eingabebildern (z.B. 28x28 Pixeln) viele Parameter. Die hohe Anzahl an Parametern führt unter anderem zu einem längeren Training, einem erhöhten Speicherverbrauch und speziell zu einer hohen Overfitting Gefahr. Mit dem *Convolution Neural Network (CNN)* wurde eine effiziente Netzwerkarchitektur speziell für Bilder entwickelt. Das Modell besteht dabei aus einer Eingabe, Ausgabe sowie mehreren Zwischenschichten. Die Zwischenschichten sind traditionell Pooling, voll vernetzte oder Faltungsschichten.

Das Modell eines CNNs ist durch den biologischen Prozess des visuellen Cortex inspiriert und arbeitet bei der Erkennung von Objekten mit stufenweisen Filtern, welche Muster in dem Eingabebild lokalisieren können [KVP+18]. Dabei ist in der Praxis häufig zu beobachten, dass das CNN in den ersten Ebenen zunächst einfache Strukturen wie Linien, Farbtupfer oder Kanten erkennt. In den weiteren Ebenen lernt das CNN dann Kombinationen aus diesen Strukturen wie einfache Formen oder Kurven. Mit jeder Ebene lassen sich somit komplexere Strukturen identifizieren. Im letzten Schritt werden die Ergebnisse den zu erkennenden Klassen oder Objekten zugeordnet.

Die CNNs arbeiten sehr robust und sind gegenüber Verzerrungen oder anderen optischen Veränderungen unempfindlich. Dies wird auch anhand der state-of-the-art Modelle [KSH12] [ZZ14] in der Bildklassifikation ersichtlich. In den folgenden Unterkapiteln wird detaillierter auf die typischen Schichten in einem CNN eingegangen.

Faltungsschicht

Mithilfe einer Faltungsschicht sollen einzelne Merkmale in den Eingabedaten erkannt und extrahiert werden. Bei der Bildverarbeitung können dies Merkmale wie Linien, Kanten oder bestimmte Formen sein. Zur Erkennung dieser Merkmale wird eine Faltungsoperation zwischen der Eingabe und mehreren Filtern durchgeführt. Dabei signalisiert das Ergebnis einer Faltung zwischen einem Bildausschnitt und einem Filter, wie stark das vom Filter vorgegebene Merkmal in dem Bildbereich auftritt. Mithilfe der Verschiebung des Filters über das gesamte Bild, wird eine sogenannte Merkmalskarte ermittelt. Diese beschreibt die Position und die Stärke des Merkmals in der Eingabe.

Die diskrete Faltung von zwei kontinuierlichen Funktionen g und h kann bezüglich eines Zeitpunktes t formal mithilfe der Formel

$$f(t) = (g * h)(t) = \sum_{a=-\infty}^{\infty} g(t-a) \cdot h(a) \tag{2.2.12}$$

berechnet werden. Dabei kann g als Eingabebild, h als Filter und $f(t)$ als Indikator für das Auftreten des vom Filter vorgegebenen Merkmals im Bildausschnitt interpre-

tiert werden. Um dieses Konzept auf zweidimensionale Bildeingaben **B** und Filter **K** anwenden zu können, muss die Formel wie folgt umformuliert werden:

$$m(i,j) = \sum_m \sum_n \mathbf{B}[i-m, j-n] \cdot \mathbf{K}[m,n]. \tag{2.2.13}$$

Dabei ist die Verwendung von Filtern zur Detektion von Merkmalen in Bildern nicht neu [GM13]. Der große Unterschied ist jedoch, dass die Werte im Filter nicht wie bei einem CNN automatisch, sondern per Hand festgelegt werden müssen. Dadurch werden lediglich vorgegebene Merkmale im Bild gefunden und entscheidungsrelevante Merkmale in den meisten Fällen nicht modelliert. Bei den CNNs können die Filter hingegen im Training automatisch so angepasst werden, dass sie bezüglich der zu lösenden Aufgabe gute Merkmale repräsentieren.

Praktisch wird die Faltung mithilfe eines Skalarprodukts zwischen dem Bildausschnitt und dem Filter berechnet. Eine Visualisierung des Faltungsprozesses ist in Abbildung 2.2.4 ersichtlich. Der Bildausschnitt, welcher als Eingabe für das Neuron dient, wird als Fenster bezeichnet und über das Eingabebild geschoben. Dabei wird die Schrittbreite des Fensters über den sogenannten *stride* Parameter gesteuert. Zur Behandlung der Randregionen existieren verschiedene Padding-Methoden.

Pooling Schicht

Die Pooling Operation ist durch den biologischen Prozess der lateralen Hemmung im visuellen Cortex motiviert [Fro18, S.235]. Hierbei werden überflüssige Informationen verworfen und die Datenmenge so reduziert. Dadurch ergeben sich mehrere Vorteile, wie zum Beispiel eine geringere Gefahr für ein Overfitting, Reduzierung des Speicherplatzes sowie ein verringerter Berechnungsaufwand.

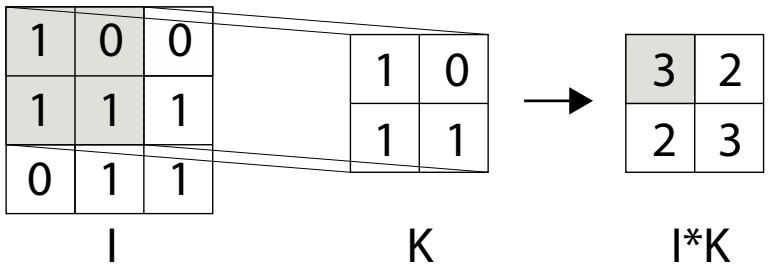

Abbildung 2.2.4: Visualisiert beispielhaft die Faltung zwischen einem Filter **K** und einer Eingabe **I**. Dabei ist die Verschiebung des Filters auf der Eingabe und der daraus resultierenden Merkmalskarte **I*K** zu beachten.

Es existieren mehrere Verfahren, mit dem das Pooling in einem CNN realisiert werden kann. Ein viel verwendeter Ansatz ist das *Max-Pooling*. Hierbei wird das Eingabebild in eine Menge von disjunkten Rechtecken unterteilt und für jedes dieser Rechtecke lediglich das Maximum weitergereicht. Dieser Prozess kann beispielhaft anhand der Abbildung 2.2.5 nachvollzogen werden.

Ein weiterer Vorteil der Pooling Schicht ist, dass sie die Abstraktionsfähigkeit von neuronalen Netzwerken erhöhen kann, indem lediglich die relationalen und keine absoluten Positionierungen der Merkmale im Bild gelernt werden. Dieser Prozess ist gerade bei der Objekterkennung hilfreich, da dort die exakte Position eines Merkmals im Bild nicht besonders relevant ist. Für die Erkennung ist es vollkommen ausreichend, wenn die ungefähre Position und das Verhältnis zu anderen Merkmalen im Bild erfasst wird.

Voll vernetzte Schicht

Die eigentliche Entscheidungslogik, wie zum Beispiel eine Klassifikation, wird am Ende eines CNNs mithilfe einer voll vernetzten Schicht realisiert. Der Aufbau der voll vernetzten Schicht folgt dem Prinzip eines MLPs und kann mithilfe der Eingabeschicht auf die extrahierten Merkmalskarten der vorgelagerten Schicht zurückgreifen. Diese Karten enthalten nach dem Training entscheidungsrelevante Merkmalsinformationen für das Eingabebild und ermöglichen damit eine effiziente Entscheidungsfindung. Die Ausgabe der letzten Schicht des CNNs wird in der Regel durch eine *Softmax* Funktion

$$softmax(\mathbf{x})_i = P(y = i|\mathbf{x}) = \frac{exp(\mathbf{x}^T \cdot \mathbf{w}_i)}{\sum_{j=1}^{K} exp(\mathbf{x}^T \cdot \mathbf{w}_j)}, \quad \text{mit } i \in \{1, ..., K\} \tag{2.2.14}$$

realisiert. Dabei berechnet diese Funktion für eine Eingabe $\mathbf{x}$ und K möglichen Klassen, eine Abbildung $\mathbb{R}^K \rightarrow \left\{\mathbf{o} \in \mathbb{R}^K | o_i \geqslant 0, \sum_{i=1}^{K} o_i = 1\right\}$. Zudem ist $\mathbf{w}_l$ die l-te Spalte der Gewichtsmatrix $\mathbf{W}$ von der letzten Schicht im CNN. Die Softmax Funktion weißt somit jeder festgelegten Klasse einen reellwertigen pseudo-Wahrscheinlichkeitswert zu und ermöglicht damit ein interpretierbares Ergebnis. Die Anzahl der Neuronen in der Ausgabeschicht ist dabei abhängig von den Klassen beziehungsweise Objekten, die das neuronale Netz unterscheiden soll.

2.2.3 *Residuales Netzwerk*

Wie bereits in den Grundlagen beschrieben, kann mithilfe eines zweischichtigen feedforward Netzwerks jede kontinuierliche Funktion beliebig genau approximiert

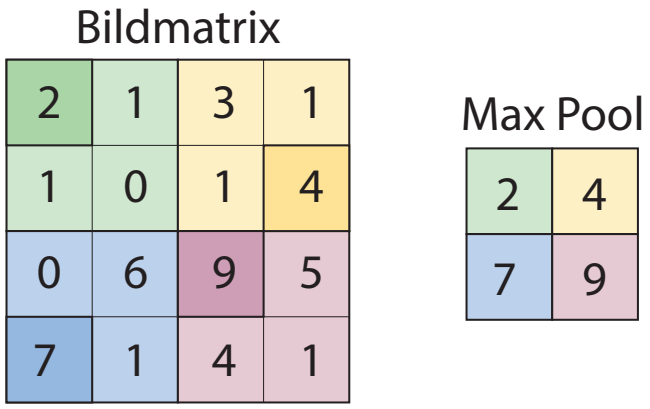

Abbildung 2.2.5: Eine beispielhafte Visualisierung einer Max-Pooling Operation. Dabei wird das Pooling auf die Bildmatrix mit der Größe 4×4 und einer Fenstergröße von 2×2 angewendet. Die einzelnen Fenster und das daraus resultierende Ergebnis (MaxPool) sind farblich hervorgehoben. Quelle: [Wal19].

werden. Jedoch führt die Verwendung von nur zwei Schichten in den meisten Fällen zu sehr breiten Netzwerken, welche zum Overfitting neigen. Aus diesem Grund geht der aktuelle Trend zu tieferen und schmaleren Netzwerken. Dies ist auch anhand der state-of-the-art Modelle [SZ14][HZRS15a] ersichtlich. Dabei führt das einfache Hinzufügen von weiteren Schichten, ab einer bestimmten Netzwerktiefe, zu keiner Leistungssteigerung mehr und kann sich sogar negativ auf die Leistung auswirken [HZRS15a]. Dieses Phänomen ist auf mehrere Probleme zurückzuführen, welche sich erst ab einer gewissen Netzwerktiefe zeigen.

Ein bekanntes Problem beim Training von tiefen neuronalen Netzwerken ist das *Vanishing Gradient Problem*. Dieses entsteht durch die Verwendung des Gradientenabstiegverfahrens in tiefen neuronalen Netzen und führt in den meisten Fällen zu einem sehr langsamen oder scheiternden Training. Hierbei werden die zurückgeführten Gradienten in den ersten Schichten eines Netzes so gering, dass sie gegen 0 gehen und somit zu einer sehr kleinen bis hin zu keiner Gewichtsveränderung in diesen Schichten führen. Auch wenn das Problem durch die Verwendung von Normalisierungen und der ReLU Aktivierungsfunktion verbessert werden konnte [IS15], sollte es bei der Optimierung von tiefen neuronalen Netzen nicht vernachlässigt werden.

Weitaus problematischer ist das *Degradation Problem*. Dieses beschreibt, dass kleinere Netzwerke einen kleineren Trainingsfehler als ihre tieferen Gegenstücke haben können [SGS15]. Dabei ist mit den tieferen Gegenstücken das kleinere Netzwerk gemeint, welches um weitere Schichten ergänzt wurde. Dieses Verhalten ist nicht intuitiv, da

das tiefere keinen größeren Trainingsfehler haben sollte als das kleinere Netz. Dies kann anhand der folgenden Überlegung nachvollzogen werden. Angenommen es gäbe ein kleines Netzwerk und ein tieferes Gegenstück. Dann könnte das tiefere Netz so konstruiert werden, dass alle Schichten aus dem kleineren Netz kopiert und die restlichen Schichten mithilfe von Identitätsfunktionen realisiert werden. Die Existenz von dieser konstruierte Lösung zeigt, dass ein tieferes Modell zu keinen höheren Trainingsfehler führen sollte. Doch es kommt in der Praxis häufig zu einem solchen Verhalten.

Eine mögliche Lösung für die oben genannten Probleme bieten die Residualen Netzwerke *(engl.: Residual Networks (ResNets))* [HZRS15a]. ResNets erweitern die tiefen Netzwerkarchitekturen um *Skip-Connections*. Mit diesen zusätzlichen Verbindungen ist es möglich, eine oder mehrere Netzwerkschichten zu überspringen und somit leichter eine Identitätsfunktion zu realisieren. Diese Erweiterung kann, wie in Abbildung 2.2.6 visualisiert, mithilfe eines feedforward Netzes umgesetzt werden. Dabei haben die Skip-Connections keine weiteren Parameter und führen auch zu keiner Erhöhung der Berechnungskomplexität.

Aufgrund der neuen Architektur müssen die einzelnen Schichten des Netzwerks nicht mehr direkt die gewünschte, sondern lediglich die residuale Abbildung lernen. Diese Idee wird im Folgenden weiter formalisiert. Angenommen eine oder mehrere Schichten eines feedforward Netzwerks sollen die gewünschte Abbildung $h(\mathbf{x})$ realisieren. Aufgrund der verwendeten Skip-Connection reicht es jedoch aus, wenn diese Schichten lediglich die residuale Abbildung

$$f(\mathbf{x}) = h(\mathbf{x}) - \mathbf{x}, \tag{2.2.15}$$

lernen. Mittels eines residualen Blocks kann die gewünschte Abbildung schließlich durch

$$f(\mathbf{x}) + \mathbf{x} \tag{2.2.16}$$

approximiert werden. In [HZRS15a] konnte experimentell nachgewiesen werden, dass die Formel 2.2.16 für tiefe neuronale Netze leichter zu optimieren ist, als die ursprüngliche Formulierung $h(\mathbf{x})$. Zudem führen die mit 0 initialisierten Gewichte eines Netzes, sowie die Verwendung von Skip-Connections, zur automatischen Realisierung einer Identitätsfunktion. Dadurch sind die einzelnen Schichten des Netzes in der Lage, die Identitätsabbildung einfacher zu lernen.

Durch die Verwendung der zusätzlichen Verbindungen ergibt sich zudem der Vorteil, dass die Gradienten beim Training schneller zu den ersten Schichten des Netzes zurückgeführt werden können. Dadurch kommt es zu einem deutlich schnelleren

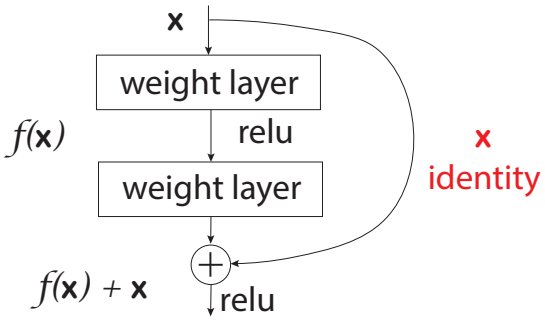

Abbildung 2.2.6: Die Visualisierung eines einzelnen residualen Blocks. Dabei ist besonders die Skip-Connection zur einfacheren Realisierung einer Identitätsabbildung zu beachten. Zudem wird die Berechnung der umformulierten und gesuchten Abbildung $f(\mathbf{x}) + \mathbf{x}$ visualisiert. Quelle: [HZRS15a].

Training und einer effizienten Verwendung des Gradientenabstiegverfahrens auch in tiefen Netzwerken. Obwohl in [HZRS15a] gezeigt werden konnte, dass extrem tiefe residuale Netzwerke mit mehr als tausend Schichten effizient optimiert werden können, führen diese in den meisten Anwendungsfällen zum Overfitting.

2.3 WORTEINBETTUNG

Die Worteinbettung *(engl.: Word Embedding)* bezeichnet eine Menge von Verfahren und Techniken aus den natürlichsprachlichen Systemen *(engl.: Natural Language Processing (NLP))*, welche Wörter auf reellwertige Vektoren abbilden. Die Vektoren sind dabei so aufgebaut, dass semantisch ähnliche Wörter im Vektorraum nah beieinander liegen.

Die ersten Modelle aus diesem Bereich analysierten die Trainingsdokumente und zählten dabei die Anzahl der Wortvorkommen in jedem Dokument, um die unterliegende Struktur der Dokumente erfassen zu können. Eines der erfolgreichsten Verfahren ist die latente semantische Analyse *(engl.: Latent Semantic Analysis (LSA))* [DDF+90]. Dieses Verfahren erhält eine Menge von Trainingsdokumenten sowie eine vorgegebene Anzahl an Kategorien als Eingabe und gibt eine Matrix mit den Zugehörigkeiten der Trainingswörter zu den gelernten Kategorien aus. Dafür wird zunächst eine normalisierte Matrix mit den Worthäufigkeiten pro Dokument bestimmt. Anschließend wird mithilfe der Singulärwertzerlegung *(engl.: Singular Value Decomposition (SVD))* [Kal96] eine Dimensionsreduktion durchgeführt. Diese Reduktion komprimiert alle relevanten Merkmale in einen kleineren Vektorraum und definiert dadurch die semantische Ähnlichkeit von Wörtern.

Auch im Bereich der Worteinbettung konnten durch die Verwendung des maschinellen Lernens bessere Ergebnisse erzielt werden [BDK14]. Eines der bekanntesten Modelle ist *Word2Vec* (siehe Kapitel 2.3.1). Dieses Modell besitzt jedoch die Limitierung, dass es die Struktur der Wörter bei der Bestimmung von Vektoren vernachlässigt und keine Vektoren für Wörter außerhalb des Vokabulars berechnen kann. Aus diesem Grund gibt es mit *FastText* (siehe Kapitel 2.3.2) eine Erweiterung von Word2Vec, welche die Limitierungen umgeht. Ein weiteres interessantes, neuronales Modell aus der NLP ist das *Character-Aware Neural Language Model*, welches im Kapitel 2.3.3 vorgestellt wird.

2.3.1 *Word2Vec*

Das Word2Vec Verfahren [MSC+13] lernt mithilfe eines neuronalen Netzwerks und einem unstrukturierten Trainingskorpus eine Abbildung von Wörtern zu reellwertigen Vektoren. Dabei werden die Vektoren so in den Vektorraum projiziert, dass sie semantische Beziehungen zwischen einzelnen Wörtern kodieren. Diese Beziehungen werden auf Grundlage der Verteilungshypothese [Saho8] automatisch aus den Trainingsdaten extrahiert. Die Hypothese stammt aus der Linguistik und besagt, dass Wörter, die in demselben Kontext vorkommen einen semantischen Zusammenhang haben.

Für das Training kann entweder das *Skip-gram* Modell oder das *Continous Bag-of-Words (CBOW)* Modell verwendet werden [MSC+13]. Beide Modelle sind schmale, zweischichtige neuronale Netzwerke (siehe Abbildung 2.3.1). In der CBOW Architektur wird das zentrale Wort mithilfe seiner Kontextwörter vorhergesagt, wohingegen beim Skip-gram Modell die Kontextwörter mithilfe des zentralen Wortes vorhergesagt werden. Dabei besteht der Kontext eines zentralen Wortes aus den Wörtern, die links und rechts von diesem liegen. In [Goo13] wird argumentiert, dass das Training mit der CBOW Architektur schneller ist als mit dem Skip-gram Modell, es jedoch bei nicht so häufig im Training vorkommenden Wörtern zu schlechteren Ergebnissen führt. Aus diesem Grund wird im Folgenden detaillierter auf das Skip-gram Modell eingegangen.

Mit dem Skip-gram Modell sollen Kontextwörter auf Grundlage des zentralen Wortes vorhergesagt werden. Dafür wird das Prinzip der *Maximum Likelihood* verwendet, bei der die Wahrscheinlichkeit von einem Kontextwort gegeben dem zentralen Wort maximiert wird. Eine formale Definition der Optimierungsfunktion für das Modell wird im Folgenden genauer betrachtet. Gegeben sei eine Liste von Trainingswörtern $[w_1, w_2, ..., w_n]$. Das Ziel ist die Maximierung der Optimierungsfunktion

$$\frac{1}{n}\sum_{t=1}^{n}\sum_{-c\leqslant j\leqslant c, j\neq 0} log\ P\left(w_{t+j}|w_t\right) \tag{2.3.1}$$

wobei c die Breite des Kontextfensters ist. Der Kontext eines Wortes w_t kann formal mithilfe eines Fensters der Größe $2 \cdot c$ wie folgt definiert werden:

$$kontext(w_t) = \{w_i | i \in \{1, ..., n\}, -c \leqslant t - i \leqslant c, t \neq i\} \tag{2.3.2}$$

Die Berechnung der bedingten Wahrscheinlichkeit $P\left(w_{t+j}|w_t\right)$ wird in einem Skip-gram Modell traditionell mithilfe der Softmax Funktion realisiert:

$$P\left(w_{t+j}|w_t\right) = \frac{exp(s(w_{t+j}, w_t))}{\sum_{k=1}^{n} exp(s(w_k, w_t))} = \frac{exp(\mathbf{v}_{w_{t+j}}^T \cdot \mathbf{v}_{w_t})}{\sum_{k=1}^{n} exp(\mathbf{v}_{w_k}^T \cdot \mathbf{v}_{w_t})} \tag{2.3.3}$$

Dabei ist $\mathbf{v}_w$ der Wortvektor für das Wort w und $s(w, v)$ die Ähnlichkeit der Wörter w und v. Die Berechnung dieser Softmax Funktion ist im Allgemeinen sehr rechenintensiv. Dies liegt daran, dass in jedem Trainingsschritt die Wahrscheinlichkeiten für alle Wörter in dem Korpus berechnet und normalisiert werden müssen. Aus diesem Grund wird die Softmax Funktion in den praktischen Implementierungen von Word2Vec mithilfe des *Negative Samplings* [GH12] und/oder dem hierarchischen Softmax [MB05] approximiert.

Die gelernten Repräsentationen kodieren zudem weitere semantische Beziehungen zwischen Wörtern, sodass mithilfe der Vektoren und arithmetischen Operationen Zusammenhänge wie zum Beispiel $vec(\texttt{germany}) + vec(\texttt{capital}) = vec(\texttt{berlin})$ erzeugt werden können. Auch wenn Word2Vec automatisch gute semantische Beziehungen zwischen Wörtern lernt, besitzt es jedoch die Limitierung, dass nur Wortvektoren für Wörter aus dem Vokabular berechnet werden können. Daher werden im Folgenden zwei weitere Modelle vorgestellt, welche diese Limitierung umgehen.

2.3.2 *FastText*

Es existieren viele Modelle aus dem Bereich der natürlichsprachlichen Systeme, unter anderem Word2Vec, welche einen individuellen Vektor für jedes Wort lernen und dabei die interne Struktur von Wörtern ignorieren. Dies ist speziell für Sprachen mit einem großen Vokabular und vielen selten vorkommenden Wörtern problematisch, da diese eine Vielzahl an Wörter enthalten, die selten oder gar nicht im Trainingskorpus vorkommen. Somit ist es für diese Wörter schwierig, eine gute Wort-Repräsentation zu lernen.

Das FastText Modell [BGJM16] kann als eine Erweiterung von Word2Vec aufgefasst werden und greift bei der Berechnung von Wortvektoren auf die internen Strukturen von Wörtern zurück. Dabei werden mit FastText speziell in morphologischen Sprachen

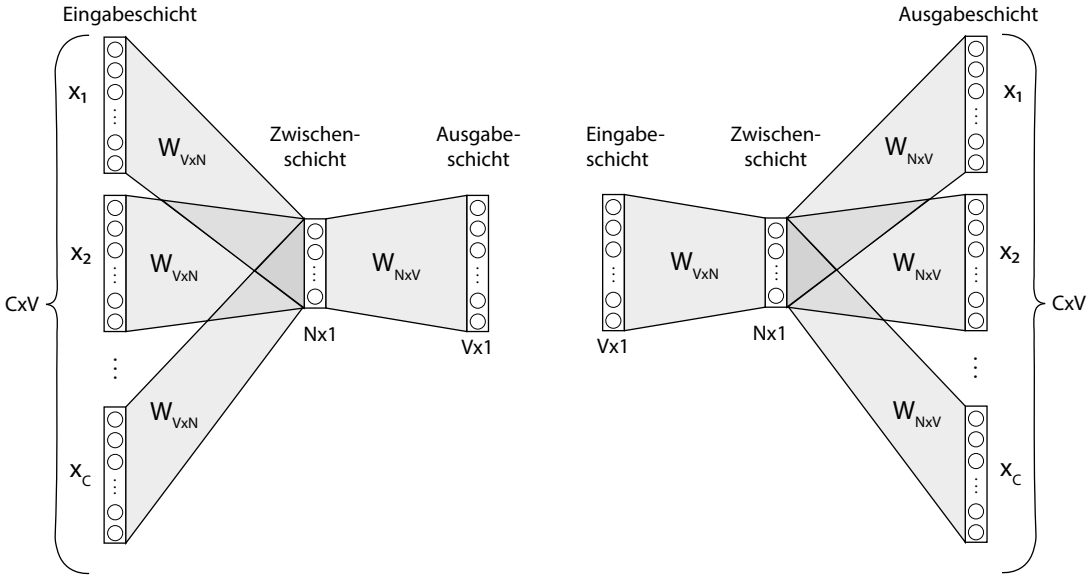

Abbildung 2.3.1: Visualisiert den Aufbau der Architekturen für ein Skip-gram (rechts) und CBOW (links) Modell. In dem Skip-gram Modell wird der Kontext aus dem zentralen Wort vorhergesagt. Dazu wird zunächst das zentrale Wort als One-Hot-Kodierung repräsentiert. Dies bedeutet, dass der Eingabevektor an der Stelle des zentralen Wortes eine eins und an allen anderen Stellen eine null hat. Dieser Eingabevektor wird im nächsten Schritt auf einen in der Regel niedrig-dimensionalen Wortvektor projiziert. Auf Grundlage des Wortvektors werden abschließend die Kontextwörter vorhergesagt. In der Abbildung befinden sich C Kontextwörter, welche jeweils mithilfe einer One-Hot-Kodierung repräsentiert werden. Der Aufbau des CBOW Modell ist analog zum Skip-gram Modell, außer dass mithilfe der Kontextwörter das zentrale Wort bestimmt wird. Quelle: [TN17].

gute Ergebnisse erzielt, da die Wörter in diesen Sprachen bestimmten Regeln folgen und diese bei der Bestimmung von Vektoren mitgelernt sowie berücksichtigt werden [BGJM16].

FastText repräsentiert jedes Wort als eine Zusammensetzung aus n-grammen. Dabei wird mithilfe des Skip-gram Modells für jedes n-gram aus dem Trainingskorpus eine Vektor-Repräsentation gelernt und die Wörter anschließend als Summe ihrer n-gram Vektoren repräsentiert. Beim Training werden zudem die Symbole $<$ und $>$ am Anfang und Ende eines Wortes verwendet, wodurch eine Unterscheidung von Präfix und Suffix bezüglich anderer n-gramme ermöglicht wird. Außerdem kann das Training nicht nur auf die n-gramme des Wortes, sondern auch auf das Wort selbst zurückgreifen. Somit stehen dem Skip-gram Modell beispielsweise bei dem Wort *where* und $n = 3$ die Informationen: <*wh, whe, her, ere, re*> und <*where*> zur Verfügung.

In der Praxis werden alle n-gramme mit $3 \leqslant n \leqslant 6$ aus einem Wort extrahiert. Im Vergleich zum Word2Vec Training wird die Ähnlichkeitsbeziehung in der Softmax Funktion (siehe Formel 2.3.3) wie folgt angepasst. Es wird angenommen, dass eine Liste von n-grammen der Größe G existiert. Gegeben Wort w, sei $\mathcal{G}_w \subset \{1, ..., G\}$ die Menge der n-gramme die in w vorkommen. Dabei sei $\mathbf{z}_k$ die Vektor-Repräsentation von n-gram k und $\mathbf{v}_c$ die Repräsentation des Wortes c. Das Wort wird als Summe der Vektor-Repräsentationen seiner n-gramme repräsentiert, wodurch sich die folgende Formel zur Berechnung der Ähnlichkeit zweier Wörter ergibt:

$$s(w, c) = \left\{ \sum_{k \in \mathcal{G}_w} \mathbf{z}_k^T \right\} \cdot \mathbf{v}_c \tag{2.3.4}$$

Ein weiterer Vorteil von FastText ist, dass sowohl das Training als auch die praktische Anwendung dieses Modells, selbst auf einem großen Textkorpus schnell ist [BGJM16]. Zudem können Wortvektoren für Wörter bestimmt werden, die nicht im Training vorkamen.

2.3.3 *Character-Aware Neural Language Models*

Mit dem *Character-Aware Neural Language Model (CharLSTM)* [KJSR15] haben die Autoren ein Sprachmodell entwickelt, welches die interne Struktur der Wörter bei der Berechnung von Wortvektoren berücksichtigt. Somit ergibt sich wie beim FastText Modell der Vorteil, dass das Modell beispielsweise a priori für die Wörter: *eventful*, *eventfully*, *uneventful* und *uneventfully* strukturell ähnliche Wortvektoren bestimmt.

Mithilfe dieses Modells können sowohl semantische als auch traditionelle Wort-Repräsentationen gelernt werden. Der Aufbau und die Funktionsweise des Modells können anhand der Abbildung 2.3.2 nachvollzogen werden.

Im Training dieses Modells wird zunächst die interne Struktur von dem aktuellen Wort, mithilfe eines *Character-Level Convolutional Neural Networks (CharCNNs)* ermittelt und in eine Vektor-Repräsentation überführt. Diese kann als eigenständige Attribut-Repräsentation verwendet oder mittels eines *Highway Networks* in einen semantischen Raum transformiert werden. Danach wird mithilfe der ermittelten Wort-Repräsentation und einem *Recurrent Neural Network Language Model (RNN-LM)* eine Verteilung über das nächste Wort vorhergesagt. Abschließend wird der *Cross Entropy Loss* zwischen der vorhergesagten Verteilung und dem nächsten Wort aus dem Training minimiert. Im Folgenden wird detaillierter auf die einzelnen Bestandteile des Modells eingegangen.

Character-Level Convolutional Neural Network

Mit dem *Character-Level Convolutional Neural Network* wird die Struktur eines Eingabewortes extrahiert. Dafür wird das Wort zunächst in eine Matrix-Repräsentation überführt und darauf anschließend insgesamt s Filter mit unterschiedlichen Größen angewendet. Auf diese Ergebnisse wird noch eine *Max-Over-Time Pooling* Operation sowie eine nicht lineare Funktion ausgeführt und ergibt damit die Vektor-Repräsentation von dem Wort. Die Vorgehensweise des CharCNNs wird im Folgenden formal beschrieben. Dabei sei $\mathcal{V}$ das Vokabular von Wörtern mit einer festen Länge, $\mathcal{C}$ das Vokabular von Zeichen, d die Dimensionalität und $\mathbf{Q} \in \mathbb{R}^{d \times |\mathcal{C}|}$ die Matrix der Zeichen-Repräsentationen. Zudem sei $k \in \mathcal{V}$ durch die Zeichenkette $[c_1, ..., c_l]$ repräsentiert, wobei l die Länge des Wortes k ist. Dann ist die *Character-Level* Repräsentation des Wortes k durch die Matrix $\mathbf{C}^k \in \mathbb{R}^{d \times l}$ gegeben. Dabei entspricht die j-te Spalte der Matrix $\mathbf{C}^k$, der Zeichen-Repräsentation von c_j. Der Aufbau einer Matrix $\mathbf{C}^k$ kann beispielhaft anhand der Repräsentation für das Wort *absurdity* in Abbildung 2.3.3 nachvollzogen werden.

Auf Grundlage dieser Repräsentation eines Wortes wird eine Faltung zwischen $\mathbf{C}^k$ und einem Filter $\mathbf{H}_m \in \mathbb{R}^{d \times w}$ mit der Breite w und dem Index $m \in \{1, ..., s\}$ berechnet. Dieses Verfahren erzeugt eine Merkmalskarte $\mathbf{f}^k_m \in \mathbb{R}^{l-w+1}$, wobei das i-te Element von $\mathbf{f}^k_m$ mithilfe der Formel

$$\mathbf{f}^k_m[i] = \left\langle \mathbf{C}^k[*, i : i + w - 1], \mathbf{H}_m \right\rangle \tag{2.3.5}$$

berechnet wird. Dabei beschreibt $\mathbf{C}^k[*, i : i + w - 1]$ die Spalten i bis $i + w - 1$ der Matrix $\mathbf{C}^k$. Die Berechnung dieser Werte kann beispielhaft anhand der Abbildung 2.3.4 nachvollzogen werden.

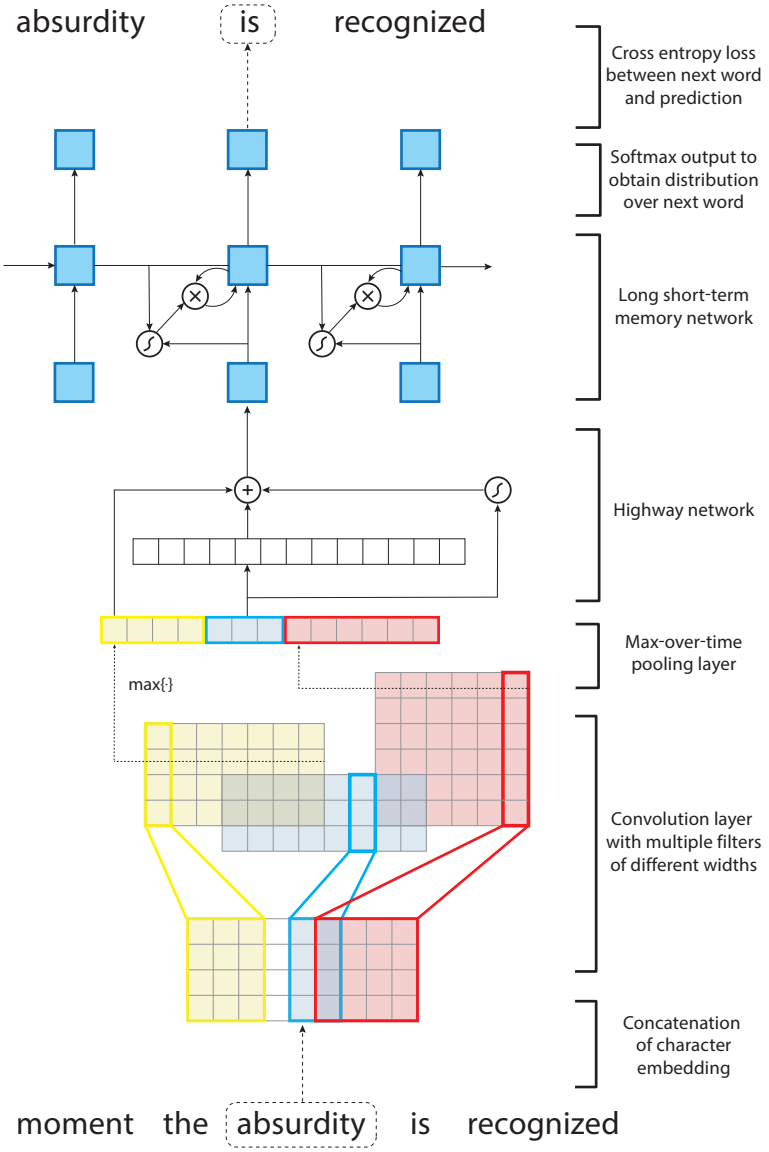

Abbildung 2.3.2: Visualisiert den Aufbau eines *Character-Aware Neural Language Models* anhand eines Beispielsatzes. Hierbei ist das Wort *absurdity* als aktuelle Eingabe gegeben. Diese Eingabe wird mit den vorherigen Wörtern kombiniert und damit das nächste Wort *is* vorhergesagt. In der ersten Schicht werden die Zeichen aus dem Eingabewort in ihre vierdimensionale Vektor-Repräsentation überführt und in einer Matrix $\mathbf{C}_k$ zusammengefasst. Anschließend werden Faltungsoperationen zwischen $\mathbf{C}_k$ und mehreren Filtern durchgeführt. In diesem Beispiel existieren zwölf Filter. Drei Filter der Breite zwei (blau), vier Filter der Breite drei (gelb) sowie fünf Filter der Breite vier (rot). Mithilfe der *Max-Over-Time Pooling* Operation wird eine fixe Größe für die Dimensionalität einer Wort-Repräsentation erreicht. Diese Repräsentation dient anschließend als Eingabe für das *Highway Network*. Die Ausgabe dieses Netzwerks ist wiederum die Eingabe eines mehrschichtigen LSTMs. Abschließend wird eine affinen Transformation gefolgt von einem Softmax auf die versteckten Schichten des LSTMs angewendet, wodurch sich eine Verteilung über das nächste Wort ergibt. Diese Verteilung wird zusammen mit dem aktuellen nächsten Wort, bezüglich des *Cross Entropy Loss* minimiert. Die elementweisen Additionen, Multiplikationen und Sigmoid Operationen werden mithilfe der Kreissymbole dargestellt. Zudem werden die affinen Transformationen mithilfe von durchgängigen Pfeilen repräsentiert. Quelle: [KJSR15].

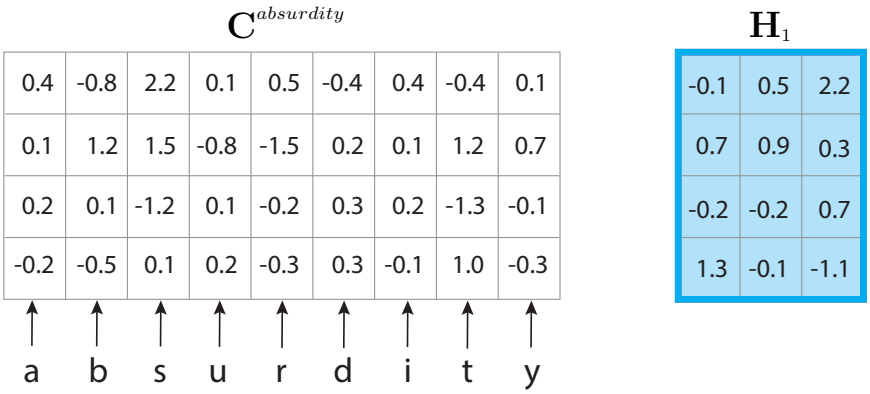

Abbildung 2.3.3: Zeigt eine beispielhafte Visualisierung für die Matrix-Repräsentation des Wortes *absurdity* ($\mathbf{C}^{absurdity}$) sowie einen beispielhaften Filter $\mathbf{H}_1$. Quelle: [KJSR16].

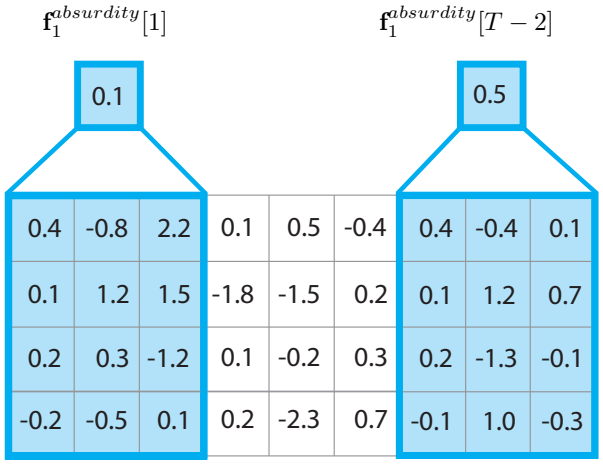

Abbildung 2.3.4: Visualisiert beispielhaft die Bestimmung der Werte $\mathbf{f}_1^{absurdity}[1] = \left\langle \mathbf{C}^{absurdity}[*, 1:3], \mathbf{H}_1 \right\rangle$ und $\mathbf{f}_1^{absurdity}[T-2] = \left\langle \mathbf{C}^{absurdity}[*, T-2:T], \mathbf{H}_1 \right\rangle$ mit der Wort-Repräsentation und dem Filter aus Abbildung 2.3.3. Quelle: [KJSR16].

Um eine Vektor-Repräsentation mit fester Länge zu erhalten, wird auf die Faltungsergebnisse jedes angewendeten Filters, eine *Max-Over-Time Pooling* Operation durchgeführt. Davor wird jedoch zunächst ein Bias-Wert b auf die ermittelten Resultate addiert sowie mit *tanh* eine nicht lineare Funktion auf diese Summe angewendet. Damit errechnet sich der Merkmalswert für den Filter $\mathbf{H}_m$ und dem Wort k mithilfe der Formel:

$$y_m^k = tanh\left(\max_i \left\{\mathbf{f}_m^k[i]\right\} + b\right) \tag{2.3.6}$$

Die Vorgehensweise bei der *Max-Over-Time Pooling* Operation kann beispielhaft anhand der Abbildung 2.3.5 nachvollzogen werden. Dabei ist die generelle Idee dieser Operation, dass es nicht relevant ist wo, sondern ob das vom Filter vorgegebene Muster im Bild vorkommt oder nicht. Die Filter sollen unter anderem n-gram Informationen kodieren. Dabei korrespondiert die Filtergröße zu der Breite des repräsentierten n-grams.

Dieser Prozess beschreibt die Berechnung von einem Merkmalswert mithilfe eines Filters. Zur Berechnung des Wortvektors für das Wort k werden in dem CharCNN jedoch mehrere Filter mit unterschiedlichen Breiten verwendet. Bei s Filtern $\mathbf{H}_1, ..., \mathbf{H}_s$ kann der Wortvektor für das Wort k formal durch $\mathbf{y}^k = \left[y_1^k, ..., y_s^k\right]$ beschrieben werden.

Highway Network

Das Modell kann mit den Ausgaben des CharCNNs ($\mathbf{y}^k$) bereits gute Wort-Repräsentationen berechnen. Jedoch können die Ergebnisse mithilfe eines *Highway Networks* weiter verbessert und die Eingabebilder mit diesem in einen semantischen Attributraum projiziert werden [KJSR15]. Im Folgenden wird der Aufbau und die Funktionsweise eines *Highway Networks* vorgestellt. Dabei kann die Ausgabe einer einzelnen Schicht in einem MLP, für eine Eingabe $\mathbf{x}$, einer Gewichtsmatrix $\mathbf{W}_H$, einem Bias-Vektor $\mathbf{b}_H$ und einer nicht linearen Funktion θ mithilfe der Formel

$$\mathbf{z} = \theta\left(\mathbf{W}_H \cdot \mathbf{x} + \mathbf{b}_H\right) \tag{2.3.7}$$

berechnet werden. Die Ausgabe einer Schicht im *Highway Network* unterscheidet sich hingegen wie folgt:

$$\mathbf{z} = \mathbf{t} \odot \theta\left(\mathbf{W}_H \cdot \mathbf{x} + \mathbf{b}_H\right) + (\mathbf{1} - \mathbf{t}) \odot \mathbf{x} \tag{2.3.8}$$

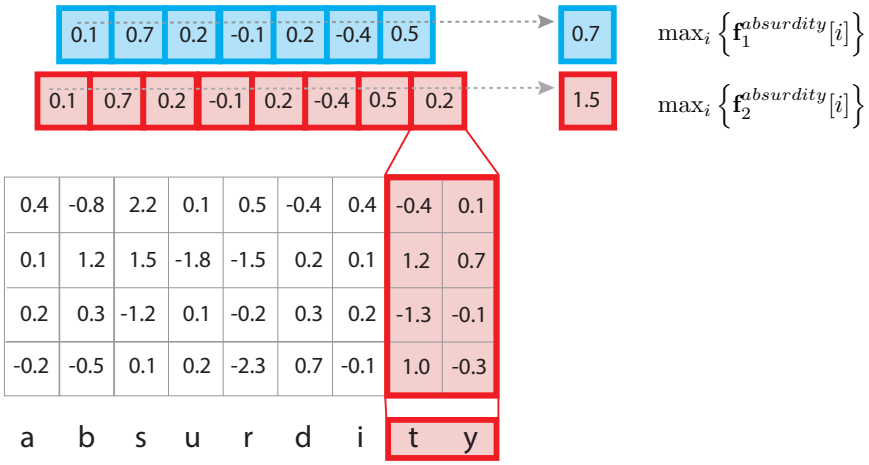

Abbildung 2.3.5: Visualisiert beispielhaft die *Max-Over-Time Pooling* Operation anhand der Resultate von zwei Filtern, welche auf die Repräsentation des Wortes *absurdity* angewendet wurden. Quelle: [KJSR16].

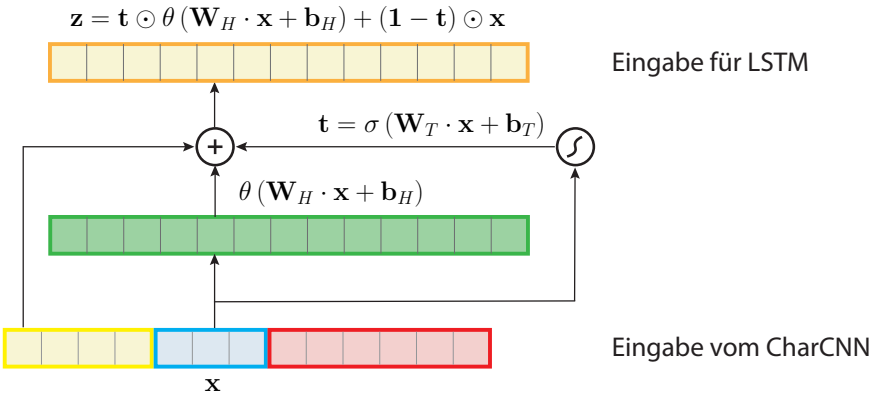

Abbildung 2.3.6: Visualisierung des *Highway Networks* im CharLSTM zur Verdeutlichung der Formel 2.3.8. Quelle: [KJSR16].

Hierbei ist $\mathbf{t} = \sigma(\mathbf{W}_T \cdot \mathbf{x} + \mathbf{b}_T)$ das sogenannte *Transform gate* und $(\mathbf{1} - \mathbf{t})$ das *Carry gate*. Die Idee und die Formel zu dem Netzwerk sind anhand der Visualisierung in Abbildung 2.3.6 nachvollziehbar. Dabei ermöglicht die *Highway* Schicht ein effizienteres Training von tiefen Netzen, da sie ein Teil der Eingabe direkt zur Ausgabe durchreicht. Mit dieser Architektur kann das Netzwerk entscheiden, ob es die k-te Eingabedimension direkt zur Ausgabe weitergibt ($t_k = 0$), oder ob es auf dieser vorher

noch eine nicht lineare Transformation anwendet ($t_k > 0$). Insgesamt kann festgestellt werden, dass das ResNet aus Kapitel 2.2.3 ein Spezialfall des *Highway Networks* ist und die dort erwähnten Vorteile auch auf diese Architektur zutreffen.

Recurrent Neural Network

Ein *Recurrent Neural Network (RNN)* ist eine neuronale Netzwerkarchitektur, die speziell für sequentielle Daten entwickelt wurde. In jedem Zeitschritt t nimmt das Netzwerk den Eingabevektor $\mathbf{x}_t \in \mathbb{R}^n$ sowie den versteckten Zustandsvektor $\mathbf{h}_{t-1} \in \mathbb{R}^s$ und produziert den nächsten Zustand $\mathbf{h}_t$ mithilfe der Formel:

$$\mathbf{h}_t = f(\mathbf{W} \cdot \mathbf{x}_t + \mathbf{U} \cdot \mathbf{h}_{t-1} + \mathbf{b}) \tag{2.3.9}$$

Dabei sind $\mathbf{W} \in \mathbb{R}^{s \times n}$, $\mathbf{U} \in \mathbb{R}^{s \times s}$ sowie $\mathbf{b} \in \mathbb{R}^s$ Parameter einer affinen Transformation und f eine elementweise nicht lineare Funktion. Ein *RNN* kann in der Theorie alle historischen Informationen bis zum Zeitpunkt t, mit dem Zustand $\mathbf{h}_t$ zusammenfassen. In der Praxis scheitert jedoch häufig das Lernen von langen Abhängigkeiten in einem RNN aufgrund des *Vanishing Gradient Problems* [Hoc98]. Um dieses Problem zu lösen, haben Hochreiter und Schmidhuber mit dem *Long Short-Term Memory (LSTM)* [HS97] eine Erweiterung zu den RNNs vorgestellt. Hierbei werden die RNNs in jedem Zeitschritt um eine Speicherzelle $\mathbf{c}_t \in \mathbb{R}^n$ erweitert. Somit erhält das LSTM in jedem Schritt t die Eingaben $\mathbf{x}_t$, $\mathbf{h}_{t-1}$ sowie $\mathbf{c}_{t-1}$ und berechnet damit die Werte $\mathbf{h}_t$ sowie $\mathbf{c}_t$ mithilfe der folgenden Formeln:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}^i \cdot \mathbf{x}_t + \mathbf{U}^i \cdot \mathbf{h}_{t-1} + \mathbf{b}^i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}^f \cdot \mathbf{x}_t + \mathbf{U}^f \cdot \mathbf{h}_{t-1} + \mathbf{b}^f) \\
\mathbf{o}_t &= \sigma(\mathbf{W}^o \cdot \mathbf{x}_t + \mathbf{U}^o \cdot \mathbf{h}_{t-1} + \mathbf{b}^o) \\
\mathbf{g}_t &= \mathit{tanh}(\mathbf{W}^g \cdot \mathbf{x}_t + \mathbf{U}^g \cdot \mathbf{h}_{t-1} + \mathbf{b}^g) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \mathit{tanh}(\mathbf{c}_t)
\end{aligned} \tag{2.3.10}$$

Dabei sind $\sigma(\cdot)$ und $\mathit{tanh}(\cdot)$ elementweise Sigmoid und Tangens hyperbolicus Funktionen. Die Vektoren $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$ werden im Kontext des LSTMs als *Input*, *Forget* und *Output gate* bezeichnet. Beim Start werden $\mathbf{c}_0$ und $\mathbf{h}_0$ mit $\mathbf{0}$ initialisiert. Die Berechnungen aus Formel 2.3.10 können anschaulich anhand der Visualisierung eines LSTM Blocks in Abbildung 2.3.7 nachvollzogen werden.

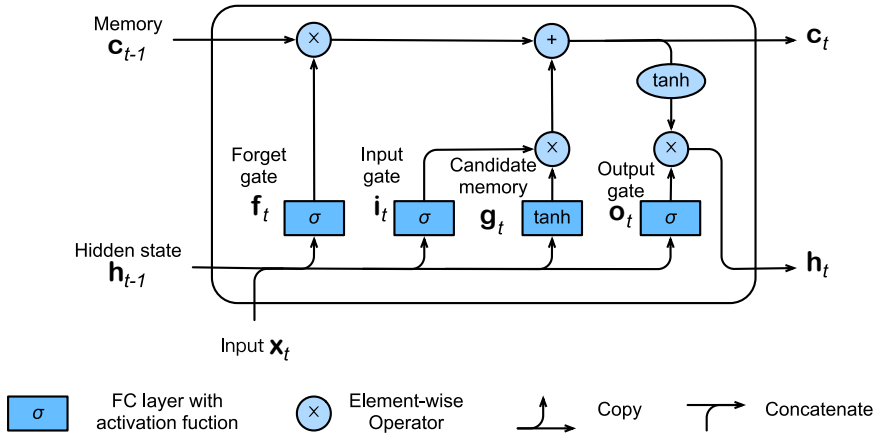

Abbildung 2.3.7: Visualisierung eines LSTM Blocks zur Verdeutlichung der Berechnungen aus Formel 2.3.10. Quelle: [ZLLS19].

Recurrent Neural Network Language Model

Das Sprachmodell berechnet mithilfe der vorherigen Wörter $[w_1, ..., w_t]$ eine Verteilung über das Wort w_{t+1}. Diese Verteilung wird vom *Recurrent Neural Network Language Model* berechnet, indem es eine affine Transformation auf die versteckte Schicht des LSTMs anwendet und anschließend die Softmax Funktion darauf ausführt:

$$\mathbf{P}\left(w_{t+1} = j | w_1, ..., w_t\right) = \frac{exp(\mathbf{h}_t \cdot \mathbf{p}^j + q^j)}{\sum_{j' \in \mathcal{V}} exp(\mathbf{h}_t \cdot \mathbf{p}^{j'} + q^{j'})} \tag{2.3.11}$$

Dabei ist $\mathbf{p}^j$ die j-te Spalte aus der Ausgabeeinbettung $\mathbf{P} \in \mathbb{R}^{s \times |\mathcal{V}|}$ und q^j ein Bias-Wert. Sei k der Index von dem Wort w_t, dann dient die Eingabeeinbettung $\mathbf{x}^k$ als Eingabe für das RNN-LM. Dabei ist $\mathbf{x}^k$ die k-te Spalte aus der Einbettungsmatrix $\mathbf{X} \in \mathbb{R}^{s \times |\mathcal{V}|}$, wobei diese durch die Ausgabe von dem CharCNN bestimmt wird. Bei dem Netzwerktraining wird die *Negative Log-Likelihood (NLL)* von der Wortsequenz $w_1, ..., w_T$:

$$\text{NLL} = -\sum_{t=1}^{T} \log \mathrm{P}\left(w_t | w_1, ..., w_{t-1}\right) \tag{2.3.12}$$

mithilfe der *Truncated Backpropagation Through Time* [Wer90] minimiert.

3 VERWANDTE ARBEITEN

In diesem Kapitel werden zwei aktuelle Ansätze aus dem Bereich des segmentierungsbasierten Word Spottings vorgestellt. Diese Ansätze sind fundamental für diese Arbeit und dienen dabei als Referenzmodell für das semantische Word Spotting. Im Kapitel 3.1 wird das von Sudholt et al. entwickelte Attribute-CNN vorgestellt. Dieser Ansatz liefert state-of-the-art Resultate für mehrere Benchmarks aus dem Bereich. Zudem wird im Kapitel 3.2 das von Wilkinson et al. entwickelte und im Vergleich zum Attribute-CNN sehr ähnliche Modell vorgestellt. In Ihrer Arbeit wurde die Idee des semantischen Word Spottings zum ersten Mal formuliert, sowie experimentell evaluiert.

3.1 ATTRIBUTE-CNN

Mit dem Attribute-CNN [SF18] haben Sudholt et al. einen CNN basierten Ansatz für das segmentierungsbasierte Word Spotting entwickelt. Der Ansatz basiert dabei auf der Arbeit von Almazán et al. [AGFV14], aber verwendet im Gegensatz dazu ein CNN zur Transformation der Wortabbilder in eine vorgegebene Attribut-Repräsentation. Somit ist mit diesem Modell sowohl eine Anfrage mittel QbS als auch QbE möglich. Wie in [SF18] und [PZG+16] beschrieben, führt das Attribute-CNN bei mehreren segmentierungsbasierten Word Spotting Benchmarks mit den Standard Evaluationsprotokollen zu state-of-the-art Ergebnissen. In Abbildung 3.1.1 wird das Word Spotting System visualisiert.

Mit diesem Modell können unterschiedliche Attribut-Repräsentationen verwendet werden. Die Autoren des Papers [SF18] favorisieren jedoch die Verwendung der *Pyramidal Histogram of Characters (PHOC)* - Repräsentation. Aus diesem Grund wird in dem gleichnamigen Kapitel detaillierter auf diese eingegangen. Zudem ist das Modell in der Lage, Bilder mit unterschiedlichen Eingabegrößen zu verarbeiten. Dies wird durch die Verwendung von einem *Spatial Pyramid Pooling Layer* beziehungsweise *Temporal Pyramid Pooling Layer* erreicht. Die Funktionsweise dieser Schichten wird in dem Kapitel *Variable Eingabegröße* vorgestellt. Abschließend wird im letzten Kapitel detailliert auf die Architektur eines Attribute-CNNs eingegangen.

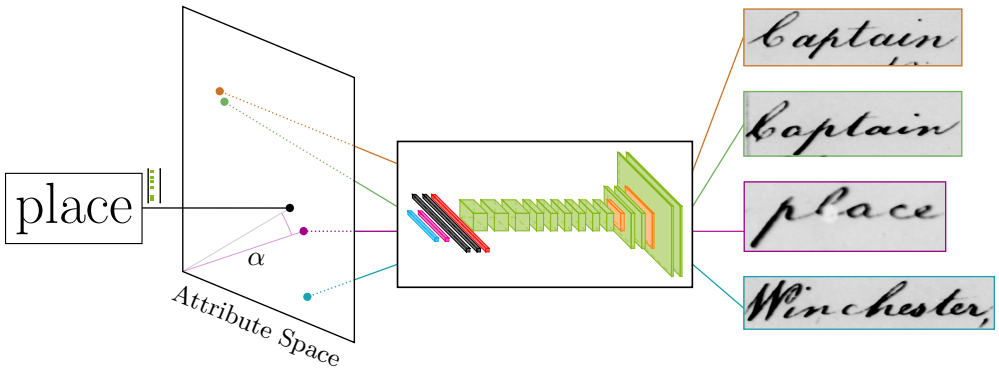

Abbildung 3.1.1: Visualisiert das Word Spotting System aus [SF18]. Bei der Verwendung von QbS wird die Attribut-Repräsentation, in dieser Abbildung die PHOC-Repräsentation, direkt für das Eingabewort berechnet. Für Wortabbilder wird die Attribut-Repräsentation hingegen mithilfe eines CNNs vorhergesagt. Somit werden Strings und Wortabbilder in denselben Attributraum projiziert und können mithilfe der Kosinus-Distanz, bezüglich der Ähnlichkeit, bewertet werden. Mit diesem System ist auch die Verwendung von QbE möglich, da die Ergebnisliste im Attributraum trivial berechnet werden kann. Quelle: [SF18].

Pyramidal Histogram of Characters

Mithilfe der PHOC-Repräsentation wird ein String in einen binären PHOC Vektor überführt. Dabei entsteht der Vektor durch die Verwendung von mehreren Histogrammen über die Zeichen eines Strings und einem festgelegten Vokabular. In der englischen Sprache besteht ein Histogramm aus 36 Dimensionen, welche sich aus dem englischen Alphabet und den Zahlen von 0 bis 9 ergeben. Dabei wird für jedes vorkommende Zeichen im Wort, die korrespondierende Positionen im Histogramm auf einen Wert von 1 gesetzt. Alle anderen Werte im Histogramm erhalten den Wert 0. Somit kann mithilfe eines Histogramms festgehalten werden, welche Zeichen im Wort vorkommen und welche nicht. Da jedoch bei einem einzelnen Histogramm mehrere Wörter dieselbe Repräsentation erhalten, werden mehrere Histogramme in einer pyramidalen Form verwendet. Dabei wird ein String mithilfe unterschiedlicher Ebenen in mehrere Bereiche zerlegt und für jeden dieser Bereiche ein binäres Histogramm berechnet. In der Ebene zwei wird der String in der Hälfte geteilt und in der Ebene drei gedrittelt. Dieses Prinzip wird für größere Splits analog angewendet. In der Abbildung 3.1.2 wird der Prozess für das Wort *place* und den Ebenen eins bis drei visualisiert. Die ermittelten Histogramme werden abschließend ebenenweise hintereinander geschrieben und die-

nen als PHOC-Repräsentation. In [SF18] werden die Ebenen 2, 3, 4 und 5 sowie die 50 häufigsten englischen Bigramme auf der zweiten Ebene verwendet. Dies ergibt einen 604-dimensionalen PHOC Vektor für die Repräsentation von Wörtern. Mithilfe dieser Kodierung ist es möglich, n-gram Informationen eines Wortes zu extrahieren und diese bei der Abbildung von Wortabbildern in PHOC Vektoren zu berücksichtigen.

Variable Eingabegröße

Aufgrund der stark unterschiedlichen Verhältnisse bei der Länge und Breite von Wortabbildern, kann es durch die Verwendung von Standardverfahren, wie dem Zuschneiden oder der Skalierung von Bildern, zur Beeinflussung der Netzwerkleistung kommen. Um dieses Problem zu umgehen, wird eine *Spatial Pyramid Pooling (SPP)* beziehungsweise *Temporal Pyramid Pooling (TPP)* Schicht für Eingabedaten mit variabler Größe verwendet. Eine dieser Schichten befindet sich dann direkt vor der voll vernetzten Schicht, da diese im Gegensatz zu den Pooling- und Faltungsschichten, keine Merkmalskarten mit variabler Größe verarbeiten kann. Dabei erzeugt eine SPP beziehungsweise TPP Schicht eine Ausgabe mit konstanter Größe. Im Folgenden wird der Aufbau und die Funktionsweise der SPP und TPP Schichten erläutert.

SPATIAL PYRAMID POOLING LAYER In einer SPP Schicht können Merkmalskarten mit variabler Größe zu Merkmalsvektoren mit konstanter Größe umgeformt werden. Dafür werden die Merkmalskarten in mehrere räumliche Bereiche unterteilt und für jeden dieser Bereiche ein Wert mithilfe der *Max-Pooling* Operation ermittelt. Dabei arbeitet dieses Verfahren mit mehreren Ebenen, welche pyramidal aufgebaut sind. In der ersten Ebene wird die Merkmalskarte als Ganzes betrachtet und auf dieser die *Max-Pooling* Operation angewendet. In der zweiten Ebene wird die Karte in jeder

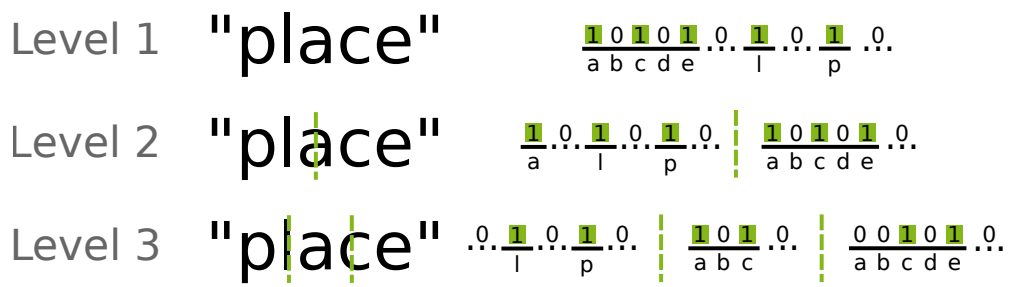

Abbildung 3.1.2: Visualisiert beispielhaft die Berechnung eines PHOC Vektors für das Wort *place* und den Ebenen eins bis drei. Quelle: [SF18].

Dimension gesplittet. Dadurch ergeben sich vier separate Bereiche, wobei jeder Bereich mithilfe der *Max-Pooling* Operation einen Wert zurückgibt. Dieses Prinzip wird für höhere Ebenen fortgesetzt, sodass in der dritten Ebene 16 separate Bereiche existieren. Mit diesem Verfahren und einer vorgegebenen Höhe der Ebenen, kann somit ein Merkmalsvektor mit konstanter Größe ermittelt werden.

TEMPORAL PYRAMID POOLING LAYER In [SF18] wurde zudem mit der TPP Schicht eine Anpassung der SPP Schicht für Wortabbilder vorgestellt. Da nach Aussagen der Autoren die Zerlegung eines Wortabbildes entlang der horizontalen Achse deutlich relevanter als die vertikale ist, werden die Merkmalskarten in einer TPP Schicht nur entlang der horizontalen Achse geteilt. Anschließend werden die separaten Bereiche, wie in der SPP Schicht, mithilfe der *Max-Pooling* Operation in einen Merkmalsvektor mit konstanter Größe überführt. Es konnte gezeigt werden, dass das System mit dieser Schicht vergleichbare beziehungsweise bessere Ergebnisse als die state-of-the-art Methode liefert [SF18]. Der Aufbau und die Funktionsweise einer TPP Schicht können beispielhaft anhand der Abbildung 3.1.3 nachvollzogen werden.

Modell

Die Architektur des Attribute-CNNs ist von der erfolgreichen VGG16 Architektur [SZ14] inspiriert und in Abbildung 3.1.4 visualisiert. Dabei wird wie in [SZ14], eine geringe Anzahl an Filtern in den ersten Faltungsschichten verwendet und die Anzahl der Filter in Abhängigkeit von der Netzwerktiefe erhöht. Diese Architektur zwingt das CNN wenige und damit generelle Merkmale in den frühen Schichten, sowie eine größere Anzahl an abstrakten Merkmalen in den späteren Schichten zu lernen. Hierbei werden lediglich Filter mit der Größe 3×3 in den Faltungsschichten verwendet. Zudem werden mithilfe von zwei Pooling Schichten die Berechnungskosten reduziert. Diese Schichten werden dabei relativ nah an den Eingabedaten platziert (siehe orangene Schicht in Abbildung 3.1.4). Wie bereits in dem Kapitel *Variable Eingabegröße* beschrieben, kann durch die Verwendung von Wortabbildern mit variabler Eingabegröße die Netzwerkleistung verbessert werden. Dafür wird eine TPP beziehungsweise SPP Schicht für Eingabedaten mit variabler Größe verwendet. In dieser Architektur befindet sich eine SPP Schicht direkt hinter der letzten Faltungsschicht im Netzwerk (siehe rote Schicht in Abbildung 3.1.4). Diese Schicht verbindet die Faltungen des Netzwerks mit dem MLP, welches aus zwei Zwischenschichten der Größe 4086 besteht. Das Lernen einer PHOC-Repräsentation entspricht einem Mehrklassenproblem mit mehreren aktiven Klassen. Jedoch wird anstelle einer Softmax Aktivierungsfunktion in der letzten Schicht, eine Sigmoid Aktivierungsfunktion zur Berechnung einer pseudo-

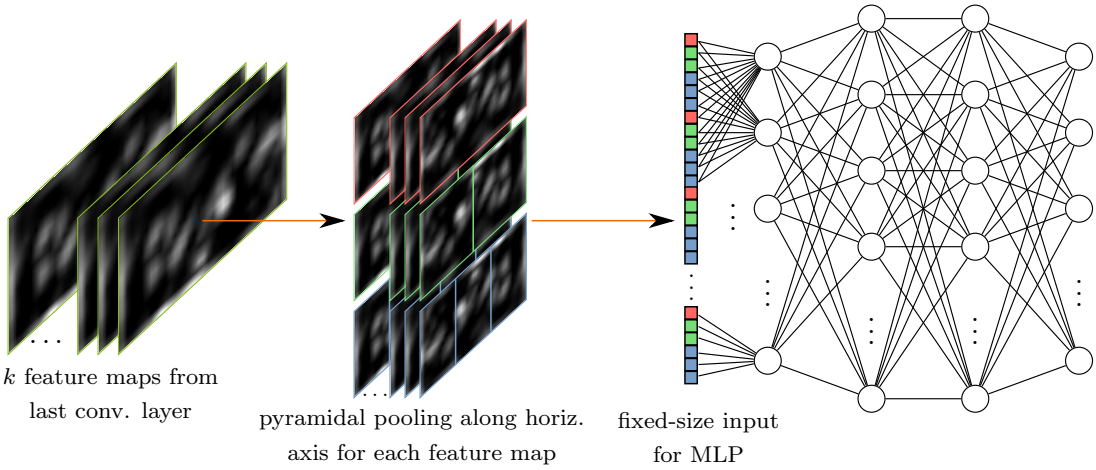

Abbildung 3.1.3: Die Abbildung visualisiert den beispielhaften Aufbau und die Funktionsweise einer TPP Schicht. Für jede Merkmalskarte in der letzten Faltungsschicht eines CNNs (links) wird ein Merkmalsvektor mit konstanter Größe extrahiert (Mitte). Dafür werden *Max-Pooling* Operationen auf pyramidal festgelegte Bereiche in der Merkmalskarte durchgeführt. In dieser Abbildung wird eine TPP Schicht mit den Ebenen eins bis drei visualisiert. Die Schicht erzeugt für jede Merkmalskarte sechs Ausgabewerte, welche mithilfe der festgelegten Bereiche innerhalb der Karte und der darauf angewendeten *Max-Pooling* Operation berechnet werden. Diese Werte werden in pyramidaler Form zusammengefasst und erzeugen damit, auch für Merkmalskarten mit variabler Größe, eine Repräsentation fester Größe. Diese Repräsentation dient abschließend als Eingabe für ein MLP (rechts). Quelle: [SF18].

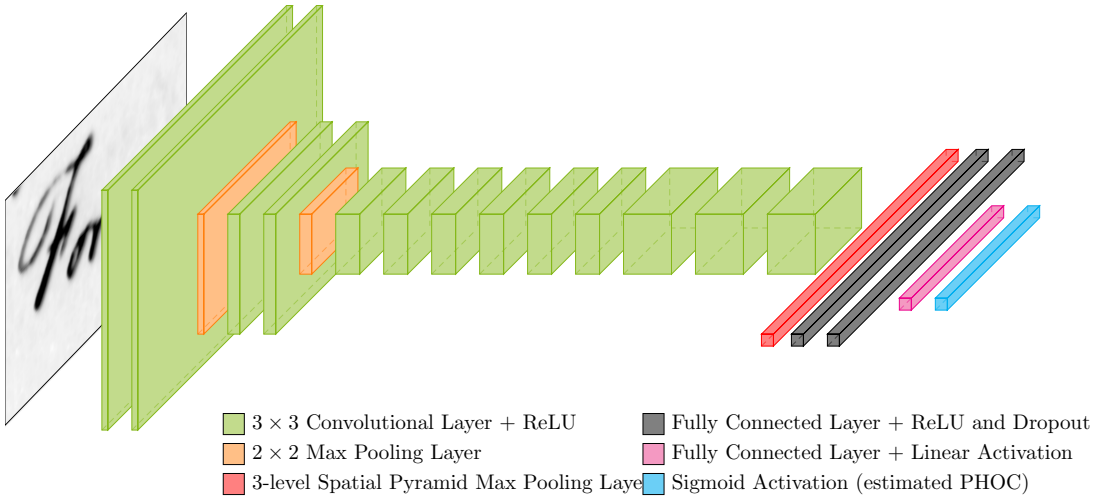

Abbildung 3.1.4: Visualisiert die Architektur eines Attribute-CNNs. Alle Faltungsschichten verwenden einen 3×3 Filter mit anschließender Anwendung einer ReLU Aktivierungsfunktion. Dabei wird auf die Eingaben in jeder Faltungsschicht, ein Padding von einem Pixel angewendet, um die Dimensionalität der Eingaben zu behalten. Mithilfe der Pooling Schichten wird die Größe der Eingabedaten anschließend reduziert. Dafür wird eine *Max-Pooling* Operation mit einer Fenstergröße von 2×2 und einem *stride* Wert der Größe zwei verwendet. Zudem wird im Training ein Dropout von 50% auf die Ausgaben der ersten beiden voll vernetzten Schichten angewendet. Quelle: [SF18].

Wahrscheinlichkeit verwendet. Diese pseudo-Wahrscheinlichkeit definiert die Zugehörigkeit einer Eingabe zu einer vorgegebenen Klasse. Für die Attribut-Repräsentation sagt dieser Wert voraus, ob ein Attribut in dem Eingabebild vorhanden ist oder nicht.

Mithilfe des Netzwerktrainings ist das Modell in der Lage, eine Abbildung zwischen Wortabbildern und PHOC-Repräsentationen zu lernen. Um mit diesem Modell Word Spotting zu betreiben, wird eine geeignete Vergleichsfunktion zur Erstellung einer Ergebnisliste benötigt. Dabei haben sich die Autoren des Attribute-CNNs für die im Word Spotting Bereich viel verwendete Kosinus-Distanz entschieden.

3.2 WILKINSON ET AL. MODELL

Das Word Spotting System von Wilkinson et al. [WB16] ist sehr ähnlich zu dem Attribute-CNN (siehe Kapitel 3.1). Einer der Hauptunterschiede ist jedoch, dass die Wortabbilder nicht nach dem Ende-zu-Ende Prinzip, sondern mithilfe eines zweistufigen Prozesses in den Attributraum projiziert werden. Dabei werden die Wortabbilder zunächst mithilfe eines Triplet-CNNs in einen Bilddeskriptor umgewandelt und anschließend mit einem kleinen MLP in den Attributraum projiziert. Diese Methode erzielt vergleichbare Resultate zu dem Attribute-CNN auf mehreren Datensätzen, wird aber generell von diesem übertroffen. Auch wenn beide Systeme sehr ähnlich zueinander sind, ist die direkte Abbildung von Wortabbildern zu Attribut-Repräsentationen im Allgemeinen zu favorisieren. Es kann jedoch generell festgestellt werden, dass die CNN basierten Ansätze deutliche Vorteile gegenüber traditionellen Verfahren haben.

Das System aus [WB16] besteht dabei aus drei verschiedenen Modulen, die in einem dreistufigen Prozess trainiert werden. Zunächst wird mithilfe eines Triplet Netzwerks [BJTM16] ein Bilddeskriptor für die Wortabbilder berechnet. Anschließend wird mithilfe verschiedener Repräsentationsmethoden, wie zum Beispiel der PHOC Methode, eine Wort-Repräsentation bestimmt. Abschließend werden mithilfe eines kleinen MLPs die Bilddeskriptoren in die Wort-Repräsentationen überführt. Der Aufbau und die Funktionsweise des Systems können anhand der Abbildung 3.2.1 nachvollzogen werden. Im Folgenden wird detaillierter auf die einzelnen Module des Word Spotting Systems eingegangen.

Bild-Repräsentation

Dieses Modul enthält ein Modell zur Ermittelung von Bilddeskriptoren für Wortabbilder. Dafür wird auf ein sogenanntes Triplet-CNN zurückgegriffen. Ein Triplet-CNN [BJTM16] ist ein genereller Ansatz zur Bestimmung von Vektor-Repräsentationen für Bilder. Dieser Ansatz besteht aus drei identischen CNNs, welche ihre Gewichte teilen.

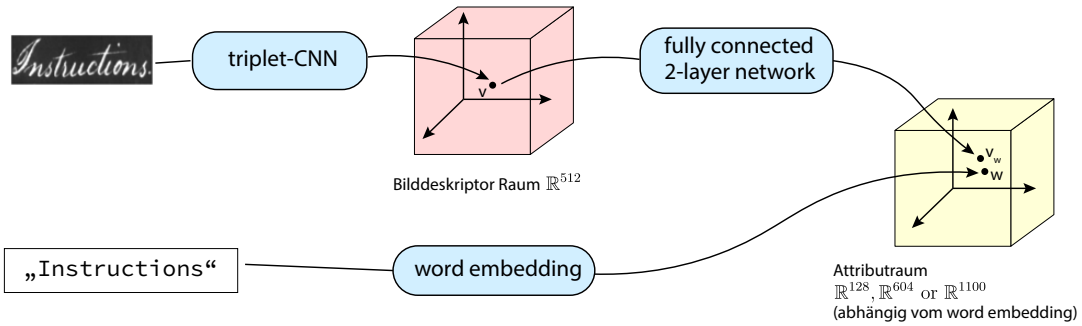

Abbildung 3.2.1: Visualisierung des Systems von Wilkinson et al. Ein Eingabebild wird mithilfe des Triplet-CNNs in einen Bilddeskriptor $\mathbf{v}$ überführt. Dieser Bilddeskriptor ist anschließend die Eingabe eines voll vernetzten, zweischichtigen neuronalen Netzwerks, welches $\mathbf{v}$ in die Attribut-Repräsentation $\mathbf{v}_w$ transformiert. Zudem wird der String *Instructions* mithilfe einer Repräsentationsmethode in die Attribut-Repräsentation $\mathbf{w}$ überführt. Quelle: [WB16].

Für die CNN Architektur wird das 34-schichtige ResNet aus [HZRS15b] verwendet. In jedem Trainingsschritt steht ein zufälliges Triplet $\tau = \{p_1, p_2, n\}$ aus 60×160 Pixel großen Wortabbildern zur Verfügung. Dabei besteht τ aus zwei Bildern von derselben Klasse (positive Beispiele p_1, p_2) sowie einem Bild einer anderen Klasse (negatives Beispiel n). Jedes Triplet enthält dabei zwei negative $\Delta(p_1, n)$, $\Delta(p_2, n)$ und eine positive Distanz $\Delta(p_1, p_2)$. Die geringere der beiden negativen Distanzen wird im Folgenden als weiche negative Distanz Δ^* bezeichnet. Das Ziel des Trainings ist, dass die Bilder derselben Klasse auf eine ähnliche und das Bild aus der negative Klasse auf eine unähnliche Vektor-Repräsentation abgebildet werden. Dabei ist die intuitive Idee, dass in einem Triplet die geringste Distanz zwischen einem positiven und negativen Beispiel größer sein sollte, als die Distanz zwischen den beiden positiven. Daher geht $\Delta(p_1, p_2)$ im Idealfall gegen 0 und $\Delta(p_1, n)$ sowie $\Delta(p_2, n)$ gegen $+\infty$. Eine solche Abbildung von Wortabbildern zu Bilddeskriptoren, wird durch die Minimierung der *SoftPN* Verlustfunktion [BJTM16] erreicht:

$$l(\tau) = \left(\frac{exp(\Delta(p_1, p_2))}{exp(\min(\Delta(p_1, n), \Delta(p_2, n))) + exp(\Delta(p_1, p_2))} \right)^2 + \left(\frac{exp(\min(\Delta(p_1, n), \Delta(p_2, n)))}{exp(\min(\Delta(p_1, n), \Delta(p_2, n))) + exp(\Delta(p_1, p_2))} - 1 \right)^2 \tag{3.2.1}$$

Dabei wird durch die Minimierung dieser Funktion, die Distanz zwischen den positiven Beispielen minimiert und die geringste Distanz zwischen den positiven

und dem negativen Beispiel maximiert. Sei $\Delta^* = \min(\Delta(p_1, n), \Delta(p_2, n))$ die weiche negative Distanz, dann wird mit der Verlustfunktion erzwungen, dass Δ^* größer als $\Delta(p_1, p_2)$ wird. Dadurch befinden sich die positiven Beispiele im Vektorraum nah beieinander und das negative Beispiel ist in dem Raum weit von den positiven entfernt. Durch dieses Verfahren wird ein sogenanntes *Soft-Negative-Mining* realisiert, was zur Folge hat, dass keine zusätzlichen schweren negativen Beispiel, wie beim *Hard-Negative-Mining*, im Netzwerktraining vorkommen müssen. Nach dem Training kann dann ein beliebiges CNN aus der Architektur verwendet werden, um 512-dimensionale Bilddeskriptoren für Wortabbilder zu bestimmen.

Wort-Repräsentation

Das Word Spotting System wird in [WB16] mit vier verschiedenen Wort-Repräsentationen evaluiert. Dabei wird zunächst die PHOC-Repräsentation (siehe Kapitel 3.1) mit denselben Parametereinstellungen wie in [SF16] verwendet. Zudem wird mit dem *Discrete Cosine Transform of Words (DCToW)* eine neue Repräsentationsmethode von den Autoren vorgestellt. Diese Methode konvertiert jedes Zeichen eines Strings s mit $|s| = m$ in einen One-Hot-Kodierten Vektor. Dies ergibt eine $K \times m$ Matrix, wobei K die Größe des verwendeten Alphabets ist. Aufgrund der variablen und von der Wortlänge abhängigen Breite der Matrix, wird zunächst eine diskrete Kosinustransformation *(engl.: Discrete Cosine Transformation (DCT))* auf jede Zeile der Matrix angewendet. Anschließend wird die Matrix verkleinert, indem lediglich die ersten R Komponenten nach der DCT Transformation behalten werden. Sollte $R > m$ sein, wird die Matrix mit Nullen aufgefüllt, sodass eine Größe von $K \times R$ garantiert werden kann. Diese Matrix wird anschließend in einen Vektor der Größe $K \cdot R$ transformiert, indem die Zeilen der Matrix hintereinander geschrieben werden. Dieser Vektor dient als Wort-Repräsentation für das Wort s. Die Autoren verwenden in Ihren Experimenten die Parameterbelegung $R = 3$ und $K = 36$, wodurch sich eine 108-dimensionale Wort-Repräsentation ergibt.

Die anderen beiden Wort-Repräsentationen werden mithilfe des *Character-Aware Neural Language Models* (siehe Kapitel 2.3.3) bestimmt. Hierbei können mit demselben Modell zwei konzeptionell unterschiedliche Repräsentationen für Wörter extrahiert werden. Die erste Repräsentation ist die Ausgabe des *Character-Level Convolutional Neural Networks* (siehe Kapitel 2.3.3). In diesem Modell wird eine Wort-Repräsentation mit fixer Länge berechnet. Dafür werden mehrere Filter mit einer Matrix gefaltet und auf den Resultaten jeweils eine *Max-Over-Time Pooling* Operation ausgeführt. Die Spalte i der Matrix wird dabei durch eine vektorielle Repräsentation für das Zeichen an der Stelle i im Eingabestring bestimmt. Dies wird im Folgenden als n-gram basierte Repräsentation bezeichnet.

In der letzten Repräsentationsmethode werden die Wörter in einen semantischen Raum transformiert. Hierbei signalisiert die Distanz zwischen zwei Wörtern in dem Raum, wie semantisch ähnlich die beiden Wörter zueinander sind. Diese Repräsentation wird im Folgenden als semantische Repräsentation bezeichnet. Diese Abbildung wird erreicht, indem die Ausgabe des CharCNNs durch eine *Highway* Schicht propagiert wird. Sowohl die n-gram als auch die semantische Repräsentation erzeugen eine 1100-dimensionale Vektor-Repräsentation.

Einbettungsnetzwerk

In diesem Modul wird ein Modell zur Abbildung von Bilddeskriptoren in den Attributraum vorgestellt. Dieses Modell ist ein kleines, voll vernetztes neuronales Netzwerk mit zwei Zwischenschichten der Größe 4096 und erhält den 512-dimensionalen Bilddeskriptor $\mathbf{v}$ als Eingabe. Die Größe der Ausgabeschicht des Netzwerks entspricht der Dimensionalität von der verwendeten Wort-Repräsentation. Auf die Ausgabe wird zudem eine l^2 Normalisierung durchgeführt. Für das Training des Netzwerks werden die Bilddeskriptoren von einem zufälligen Paar von Wortabbildern (x, y) und die zugehörigen Repräsentationen $(\mathbf{v}, \mathbf{w})$ verwendet. Das Netz wird mit dem *Cosine Embedding Loss* wie folgt optimiert. Gegeben seien $\mathbf{x} = (\mathbf{v}, \mathbf{w})$ und ein zugehöriges Label $y \in \{-1, 1\}$, welches signalisiert, ob $\mathbf{v}$ und $\mathbf{w}$ aus derselben Klasse stammen oder nicht. Dadurch ergibt sich die Optimierungsfunktion

$$l(x, y) = \begin{cases} 1 - \cos(\mathbf{v}, \mathbf{w}) & \text{wenn } y = -1 \\ max(0, \cos(\mathbf{v}, \mathbf{w}) - \gamma) & \text{wenn } y = 1 \end{cases} \tag{3.2.2}$$

wobei γ ein Hyperparameter ist. Die Optimierungsfunktion benötigt sowohl Paare von derselben ($y = 1$) als auch von unterschiedlichen Klassen ($y = -1$). Die Autoren von [WB16] wählen die Trainingsdaten in Ihren Experimenten so aus, dass die Hälfte von derselben und der Rest von unterschiedlichen Klassen stammen. In dem Training dieses Netzwerks werden zudem die Parameter im Triplet-CNN mit optimiert, wodurch sich eine deutliche Leistungssteigerung dieses Ansatzes ergibt [WB16].

4 EXPERIMENTELLE EVALUATION

Eine Motivation für diese Arbeit ist die Bestätigung der angegebenen Ergebnisse aus [WB16] von Wilkinson et al. Dafür wurde in dieser Masterarbeit das beschriebene Modell aus dieser Veröffentlichung nachimplementiert, trainiert sowie evaluiert. Im Kapitel 4.1 werden die relevanten Datensätze für das Training und die Evaluation des Systems angegeben. Eine detaillierte Beschreibung des Versuchsaufbaus ist in Kapitel 4.2 ersichtlich. Hierbei werden die Parametereinstellungen und die Trainingsdetails der einzelnen Module aus dem Wilkinson et al. Ansatz beschrieben. Für die abschließende Evaluation des Systems wird das in Kapitel 4.3 beschriebene Protokoll verwendet. Im Kapitel 4.4 werden die veröffentlichten Ergebnisse aus [WB16], mit den Resultaten von der Nachimplementierung verglichen und diskutiert.

4.1 DATENSÄTZE

Für die Evaluation des Word Spotting Systems aus [WB16] wird auf dem George Washington Datensatz und die IAM offline Handschrift Datenbank zurückgegriffen. Diese Datensätze sind prominente und viel verwendete Benchmarks im Word Spotting Bereich [Sud18] [AGFV14] [WB16] und werden zusammen mit dem CVL Datensatz für das Training der Modelle verwendet. Die drei Datensätze bieten jeweils vorsegmentierte sowie annotierte Wortabbilder mit variabler Größe und werden im Folgenden detaillierter beschrieben. Außerdem wird in diesem Kapitel der George Washington Writings Datensatz vorgestellt, welcher für das Training des CharLSTMs verwendet wird.

George Washington Datensatz

Der George Washington (GW) Datensatz ist einer der berühmtesten Datensätze für die Evaluation von Word Spotting Ansätzen. Der Datensatz umfasst lediglich 20 englischsprachige Dokumentenseiten aus den im Jahr 1755 verfassten Briefen des ehemaligen US Präsidenten George Washington. Dabei kann davon ausgegangen werden, dass diese historischen Dokumente von nur einer Person verfasst wurden. Die Daten sind bereits vorsegmentiert sowie annotiert. Dabei existieren insgesamt 4860 annotierte Wortabbilder mit 1124 unterschiedlichen Transkriptionen. Es gibt

keinen offiziellen Split für die Trainings- und Testdaten. Jedoch wird in den meisten Publikationen das Evaluationsverfahren aus [AGFV14] verwendet. Hierbei wird eine Kreuzvalidierung mit vier Splits durchgeführt, wobei ein viertel der Wortabbilder als Testdaten und der Rest als Trainingsdaten verwendet werden. Diese Evaluation wird insgesamt viermal durchgeführt, sodass jedes Mal ein anderes der vier Splits als Testdaten dient. Abschließend werden die Ergebnisse der einzelnen Evaluationen gemittelt und ergeben somit die Leistung des Systems.

IAM offline Handschrift Datenbank

Die IAM offline Handschrift Datenbank (IAM-DB) [MB02] ist ein speziell für die Handschrifterkennung entwickelter Datensatz, welcher in vielen Word Spotting Publikationen als Benchmark verwendet wird. Die Datenbank besteht aus 1539 Seiten, welche von 657 verschiedenen Personen mit moderner Handschrift und in der englischen Sprache erzeugt wurden. Diese 1539 Seiten enthalten insgesamt mehr als 13000 annotierte Textzeilen, welche wiederum aus 115320 Wortabbildern bestehen. Die Wortabbilder liegen als Graustufenbilder vor und weisen eine gute visuelle Qualität auf. Mit der *official writer independent text line recognition partition* existiert eine offizielle Partitionierung von Trainings-, Test- und Validierungsdaten. Zudem gibt es Informationen bezüglich dubioser beziehungsweise fehlerhafter Zeilentranskriptionen, sodass diese im ersten Evaluationsschritt von Almazán et al. [AGFV14] entfernt werden. Die IAM-DB ist ein anspruchsvoller Datensatz. Dies liegt vor allem an der hohen Anzahl an verschiedenen Verfassern und der daraus resultierenden großen Intra-Klassen-Varianz für Wortabbilder mit derselben Transkription. Eine weitere Schwierigkeit entsteht durch die schreiberunabhängige Partitionierung. Das bedeutet, dass alle Wortabbilder einer einzelnen Person entweder zu den Trainings- oder den Testdaten gehören.

CVL Datenbank

Die CVL Datenbank [KFDS13] ist eine öffentliche Datenbank, welche speziell für die Identifikation von Schreibern und das Word Spotting entwickelt wurde. Sie basiert auf einem deutschen und sechs englischen Texten. Diese Texte wurden von insgesamt 310 verschiedenen Personen handschriftlich abgeschrieben. Dabei haben 27 Personen jeweils alle sieben Texte verfasst und die restlichen 283 jeweils nur fünf. Die Texte stammen aus berühmten Literaturwerken, wie zum Beispiel *William Shakespeare - Mac Beth* oder *Johann Wolfgang von Goethe - Faust. Eine Tragödie*. Die handschriftlich verfassten Texte stehen als RGB kodierte Dokumentabbilder zur Verfügung. Zudem existiert eine Datenbank mit bereits vorsegmentierten und annotierten Wortabbildern.

George Washington Writings Datensatz

Der George Washington Writings (GWW) Datensatz besteht aus den automatischen Transkriptionen aller 40 Bänder von *The writings of George Washington from the original manuscript sources*, 1745 – 1799. Bei diesen Dokumenten handelt es sich um eine Sammlung von Briefen und anderen Dokumenten, die George Washington während seiner Zeit als Präsident verfasst hat. Die einzelnen Bänder umfassen jeweils mehrere hundert Seiten und wurden aus der öffentlichen Datenbank von `http://www.archive.org` am 06.05.2019 entnommen. Leider ist das Ergebnis der automatischen Transkription dieser Dokumente sehr schlecht. Jedoch ist dieser Datensatz gerade für das Training von Modellen mit vielen Parametern wichtig, da ansonsten wenig Trainingsdaten für den George Washington Datensatz existieren.

Die Transkriptionen der Bücher enthalten viele Sonderzeichen und andere Artefakte. Zudem sind mitten in den Texten die Buchseiten sowie Kapitelüberschriften zu finden. Auch die Tabellen werden fehlerhaft dargestellt und sorgen für einen hohen Aufwand bei der Säuberung der Dokumente. Außerdem enthält der Datensatz viele Abkürzungen sowie einige Rechtschreibfehler und Wortneubildungen. Aus diesem Grund ist eine aufwendige Vorverarbeitung unablässig. In der Vorverarbeitung wurden zunächst alle Sonderzeichen entfernt. Zudem wurden die überflüssigen und fehlerhaften Punkte in den Dokumenten beseitigt, sodass die tatsächlichen Sätze anhand der übrigen Punktzeichen einfacher aus dem Text extrahiert werden können. Wie bereits beschrieben, enthalten die Dokumente bestimmte Muster, wie z.B. Kapitelüberschriften oder Buchseiten, welche die Lesbarkeit der Dokumente erschweren und deshalb in den Daten eliminiert wurden. Außerdem gibt es häufig auftretende Fehler, wie zum Beispiel, dass das Wort of zu o1 transkribiert wurde. Diese falschen Transkriptionen wurden in der Vorverarbeitung ersetzt. Abschließend wurden die Texte noch einmal manuell aufwendig gesäubert und mithilfe von regulären Ausdrücken in Sätze untergliedert.

4.2 VERSUCHSAUFBAU

In diesem Kapitel wird der Versuchsaufbau für das Word Spotting Modell aus [WB16] beschrieben. Dafür wird im Folgenden detailliert auf die Parametereinstellungen und die Trainingsdetails der einzelnen Module eingegangen.

TRIPLET-CNN Das Triplet-CNN wird zunächst mithilfe der CVL Datenbank vortrainiert. Anschließend wird das vortrainierte Modell jeweils auf dem GW Datensatz und der IAM Datenbank spezialisiert. In [WB16] konnte gezeigt werden, dass dieses Verfahren das Overfitting des Modells reduzieren konnte. Bei dem Training des Triplet-

CNNs werden die Gewichte des Modells in jedem Trainingsschritt so angepasst, dass die SoftPN Verlustfunktion (siehe Formel 3.2.1) für ein Batch von zufällig generierten Triplets minimiert wird. Dabei werden so viele zufällige Triplets generiert, dass keins im Netzwerktraining doppelt vorkommt. Für das Training des Triplet-CNNs auf der CVL Datenbank, werden insgesamt 185000 Trainingsschritte (Iterationen) durchlaufen. Die Modelle für GW und IAM werden bis auf die genannten Kleinigkeiten nahezu identisch trainiert. Die Lernrate wird initial auf 0.01 gesetzt und alle 30000 (50000 für IAM) Iterationen mit 0.1 multipliziert. Bei dem Training für den GW Datensatz werden 100000 und für die IAM Datenbank 200000 Iterationen durchgeführt.

EINBETTUNGSNETZWERK Das Einbettungsnetzwerk wird für den GW Datensatz mit 30000 und die IAM Datenbank mit 50000 Iterationen trainiert. Wie bereits beim Training des Triplet-CNNs, werden genug zufällige Paare von Bilddeskriptoren und Attribut-Repräsentationen für das Netzwerktraining generiert, sodass das Netzwerk im Training kein Paar doppelt sieht. Beim Training des Einbettungsnetzwerks kann durch die Optimierung der Parameter im Triplet-CNN eine deutliche Leistungssteigerung des Systems erreicht werden. Der Hyperparameter γ aus der Verlustfunktion (siehe Formel 3.2.2) wird für GW auf 0.1 und für IAM auf 0.3 festgelegt. Sowohl das Triplet-CNN als auch das Einbettungsnetzwerk werden mit dem *mini-batch stochastic gradient descent mit Nesterov Momentum* trainiert. Dabei wird eine Batchgröße von 128, eine Gewichtsabnahme von 10^{-4}, ein Momentum von 0.9 und eine Lernrate von 0.01 verwendet.

CHARLSTM Das in [KJSR15] beschriebene CharLSTM wird mit den in dieser Veröffentlichung angegebenen Standardparametern und der Implementierung aus [Lee18] für den GW Datensatz und die IAM Datenbank trainiert. Dabei stammen die Trainings- sowie Testdaten für den GW Datensatz aus den angepassten Dokumenten des GWW Datensatzes. Hierbei wurde nämlich zunächst in einem Vorverarbeitungsschritt mit den insgesamt 20000 häufigsten Wörtern aus diesem Datensatz ein Vokabular bestimmt und damit die Dokumente bereinigt. Dabei wurden alle Wörter und Symbole aus den Dokumenten, die nicht im Vokabular vorkamen, durch ein Sonderzeichen ersetzt und zudem alle Zahlen mit einem speziellen Symbol substituiert. Für das Training des CharLSTMs auf die IAM Datenbank wurde auf die *Penn Treebank* [TMS03] zurückgegriffen.

4.3 EVALUATIONSPROTOKOLL

Die Evaluation folgt dem Protokoll aus [AGFV14] und verwendet mit der *Mean Average Precision (MAP)* ein Bewertungsmaß, mit dem die Leistung von Word Spotting Systemen bestimmt werden kann. Bei der Evaluation eines Word Spotting Systems wird eine Bewertung der Ergebnislisten vorgenommen. Aus diesem Grund wird im Folgenden nochmal detailliert beschrieben, wie die Ergebnislisten für das Modell aus [WB16] ermittelt werden.

Für die Berechnung der Ergebnislisten beim QbE Ansatz muss die Größe des Anfragebildes zunächst auf 60×160 Pixel geändert werden. Anschließend wird das Bild invertiert und mithilfe des trainierten CNNs in einen Bilddeskriptor umgewandelt. Dieser Deskriptor dient dann als Eingabe für das Einbettungsnetzwerk, welches diesen in die benötigte Attribut-Repräsentation überführt. Bei der Verwendung des QbS Ansatzes wird hingegen für das Anfragewort entweder die benötigte Attribut-Repräsentation mit dem CharLSTM vorhergesagt oder die PHOC bzw. DCToW-Repräsentation für dieses berechnet. Die im Word Spotting benötigte Datenbank besteht bei der Evaluation aus den Wortabbildern der Testmenge und variiert daher bezüglich der Datensätze. Diese Wortabbilder werden mit demselben Verfahren wie bei dem Anfragebild im QbE Ansatz, in den Attributraum projiziert, wodurch eine Ähnlichkeitsbewertung zwischen der Anfrage und den Wortabbildern der Datenbank ermöglicht wird. Anschließend werden in diesem Raum die Distanzen zwischen der Anfrage und allen in der Datenbank gespeicherten Wortabbilder bestimmt und in einer nach der minimalen Distanz sortierten Ergebnisliste zurückgegeben. Die Distanz zwischen der Anfrage und einem Element der Datenbank wird mithilfe der Kosinus-Distanz

$$d(\mathbf{u}, \mathbf{v}) = 1 - \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|_2 \|\mathbf{v}\|_2} \tag{4.3.1}$$

für die jeweils zugehörigen Attribut-Repräsentationen $\mathbf{u}$ und $\mathbf{v}$ bestimmt. Soweit nicht weiter angegeben, wird jedes Element aus der Testmenge bei der Evaluierung genau einmal als Anfrage verwendet.

Die Datenbanken werden sowohl mittels des QbE als auch QbS Ansatzes evaluiert. Für den QbE Ansatz werden jedoch nur die Wortabbilder als Anfragebilder berücksichtigt, die mehr als einmal in der Testmenge vorkommen. Zudem wird das Anfragebild aus der Ergebnisliste entfernt und geht dadurch nicht in die Bewertung ein. Es kann vorkommen, dass mehrere Wortabbilder mit derselben Transkription in der Testmenge vorhanden sind. Aus diesem Grund wird für den QbS Ansatz jede Transkription nur einmalig als Anfragewort verwendet. Wie in dem Standardprotokoll beschrieben,

werden zudem die offiziellen Stopwörter aus der Anfrageliste für die Evaluation der IAM Datenbank entfernt.

Die Resultate werden mithilfe der MAP Werte angegeben, wobei dies das aktuelle Standard Bewertungsmaß für den Vergleich von Word Spotting Methoden ist. Dieser Wert lässt sich aus dem Durchschnitt der *Average Precision (AP)* Werte für jede Anfrage bestimmen. Dabei wird der AP Wert mit der Formel

$$\mathrm{AP} = \frac{\sum_{k=1}^{l} p(k) \cdot r(k)}{t} \tag{4.3.2}$$

berechnet. Hierbei ist l die Länge der Ergebnisliste und $p(i)$ der Precision Wert für die ersten i Elemente der Liste. Dabei beschreibt der Precision Wert mit:

$$p(k) = \frac{\sum_{i=1}^{k} r(i)}{k} \tag{4.3.3}$$

den Anteil der für die Anfrage relevanten Elemente. Zudem ist $r(i) \in \{0, 1\}$ eine Indikatorfunktion, für die gilt:

$$r(i) = \begin{cases} 1 & \text{wenn } \mathit{trans}(i) = a \\ 0 & \text{sonst} \end{cases} \tag{4.3.4}$$

Dabei ist $\mathit{trans}(i)$ die Transkription des i-ten Elements der Ergebnisliste, für die Anfrage mit der Transkription a. Zudem ist $t = \sum_{i=1}^{l} r(i)$ und entspricht damit der Gesamtanzahl der Wortabbilder mit derselben Transkription in der Ergebnisliste.

Mithilfe dieses Evaluationsprotokolls können die verschiedenen Attribut-Repräsentationen mit den state-of-the-art Ansätzen aus dem segmentierungsbasierten Word Spotting verglichen werden.

4.4 ERGEBNISSE

In diesem Kapitel werden die ermittelten Ergebnisse von der Nachimplementierung des Systems aus [WB16] bezüglich des GW Datensatzes und der IAM Datenbank vorgestellt. Zudem werden die Ergebnisse mit den angegebenen Werten aus der Veröffentlichung verglichen sowie diskutiert.

Zunächst werden die Ergebnisse für den GW Datensatz betrachtet. Dabei wurden in der Arbeit die MAP Werte sowohl mit dem QbS als auch mit dem QbE Ansatz für insgesamt fünf verschiedene Wort-Repräsentationen evaluiert. Die Ergebnisse der

	Eigene Implementierung		*Wilkinson Paper*	
Modell	*QbE*	*QbS*	*QbE*	*QbS*
Triplet-CNN	91.31	—	93.61	—
PHOC	95.52	91.93	98.00	92.23
DCToW	96.55	95.41	97.98	93.69
CharLSTM (n-gram)	97.33	89.98	97.70	83.97
CharLSTM (semantisch)	97.17	81.93	96.91	69.81

Tabelle 4.4.1: Gegenüberstellung der Ergebnisse für den George Washington Datensatz, mit der Nachimplementierung dieser Arbeit und den veröffentlichten Werten aus [WB16]. Hierbei sind die Werte mit der MAP angegeben.

Nachimplementierung und die veröffentlichten Resultate aus [WB16] sind in der Tabelle 4.4.1 aufgelistet und werden im Folgenden diskutiert. Dabei fällt zunächst auf, dass die ermittelten Ergebnisse der Nachimplementierung sehr ähnlich zu den veröffentlichten Resultaten sind. Jedoch gibt es kleine Abweichungen, welche hauptsächlich auf die Generierung der Trainingsdaten zurückzuführen sind. Bei dem Training werden nämlich in jedem Trainingsschritt zufällige Triplet Daten generiert. Dafür wird zunächst ein zufälliges Wortabbild als Anchor ausgewählt. Für diesen Anchor wird dann wiederum zufällig ein positives und ein negatives Wortabbild ermittelt. Auf diese ausgewählten Wortabbilder wird dann anschließend noch jeweils eine Datenaugmentierung mit zufällig ausgewählten Parametern angewendet. Wie in mehreren Experimenten ermittelt werden konnte, haben sowohl die Triplets selbst, als auch ihre Sortierung einen hohen Einfluss auf das abschließende Ergebnis des zu trainierenden Netzwerks. Aus diesem Grund wird ersichtlich, dass durch die zufällige Auswahl der Triplets und deren Sortierung, die Ergebnisse von der Nachimplementierung und den veröffentlichten Resultaten aus [WB16] leicht voneinander abweichen können. Diese Abweichung ist gut anhand der QbE Werte des Triplet-CNNs sowie der PHOC und DCToW-Repräsentationen ersichtlich. Zudem kann festgestellt werden, dass der QbS Wert bei der PHOC-Repräsentation fast identisch für die beiden Implementierungen ist und bei der DCToW-Repräsentation sogar leicht bessere Werte aufweist. Die deutlichsten Unterschiede sind bei den beiden CharLSTMs zu finden. Hierbei hat die Nachimplementierung zu deutlich höheren QbS Werten geführt. Dies ist wahrscheinlich darauf zurückzuführen, dass das CharLSTM auf den gesäuberten Daten des GWW Datensatzes trainiert wurde und diese in der Nachimplementierung eine aufwendigere Vorverarbeitung durchlaufen haben. In [WB16] wurden leider keine Details

zur Vorverarbeitung dieser Daten genannt sondern lediglich, dass eine durchgeführt wurde.

Im Folgenden werden die Resultate der beiden Implementierungen für die IAM Datenbank verglichen. Dabei sind die Ergebnisse in der Tabelle 4.4.2 gegenübergestellt und werden im weiteren Verlauf des Kapitels diskutiert. Hierbei ist zunächst festzustellen, dass auch bei der IAM Datenbank fast dieselben Muster wie beim GW Datensatz zu sehen sind. Dabei zeigen sich auch hier kleinere Unterschiede bei den QbE Werten des Triplet-CNNs sowie bei den PHOC und DCToW-Repräsentationen. Die leichten Differenzen sind auch hier hauptsächlich mit den zufällig generierten Trainingsdaten erklärbar. Bei den Implementierungen von den genannten Modellen konnten bezüglich der QbS Resultate kaum Unterschiede festgestellt werden. Die deutlichsten Differenzen zwischen den Resultaten der Nachimplementierung und den veröffentlichten Werten des Wilkinson Papers wurden, wie beim GW Datensatz, bei den CharLSTM Ergebnissen erzielt. Die großen Unterschiede sind jedoch nicht genau erklärbar, da es nur wenige Informationen zum Training des CharLSTMs in [WB16] gibt. In der Veröffentlichung wird jedoch selbst an den vorgestellten Ergebnissen zum n-gram Modell gezweifelt und keine Erklärung für die niedrigen Werte angegeben. Bei der Nachimplementierung wurde, wie im Paper vorgestellt, das CharLSTM auf der Penn Treebank mit den Standardparametern aus [KJSR15] trainiert. Mit dem trainierten Modell konnten dann, mit demselben Verfahren wie bereits bei dem GW Datensatz, die genannten Ergebnisse erreicht werden. Diese Ergebnisse sind dabei deutlich intuitiver als die des Wilkinson Papers.

	Eigene Implementierung		*Wilkinson Paper*	
Modell	***QbE***	***QbS***	***QbE***	***QbS***
Triplet-CNN	66.45	—	70.81	—
PHOC	72.72	88.23	77.24	89.49
DCToW	71.75	83.20	76.98	85.33
CharLSTM (n-gram)	79.78	88.27	53.43	45.08
CharLSTM (semantisch)	81.98	84.45	81.58	75.74

Tabelle 4.4.2: Gegenüberstellung der MAP Evaluationsergebnisse für die IAM Datenbank, mit der Nachimplementierung dieser Arbeit und den veröffentlichten Werten aus [WB16].

Insgesamt kann festgestellt werden, dass die Werte aus [WB16], sowohl für den GW Datensatz als auch für die IAM Datenbank, mit der Nachimplementierung dieser Arbeit bestätigt werden konnten. Gerade bei den CharLSTM-Repräsentationen konnten die veröffentlichten Ergebnisse, für beide Datensätze, sogar deutlich übertroffen werden.

5 SEMANTISCHES WORD SPOTTING

Die bisherigen attributbasierten Word Spotting Systeme transformieren die Wörter und Wortabbilder so in den Attributraum, dass der Abstand zwischen den Wörtern im Raum näherungsweise der Editierdistanz entspricht. Dabei ist die Editierdistanz ein Maß für die Ähnlichkeit von zwei Wörtern und wird mithilfe der minimalen Anzahl an Operationen für die Umwandlung des einen Wortes in das andere bestimmt. Beim semantischen Word Spotting wird hingegen ein semantischer Attributraum verwendet, wobei die Distanz zwischen zwei Wörtern in dem Raum der semantischen Ähnlichkeit dieser beiden Wörter entspricht. Die Abbildung 5.0.1 visualisiert beispielhaft den Unterschied zwischen den beiden Attributräumen.

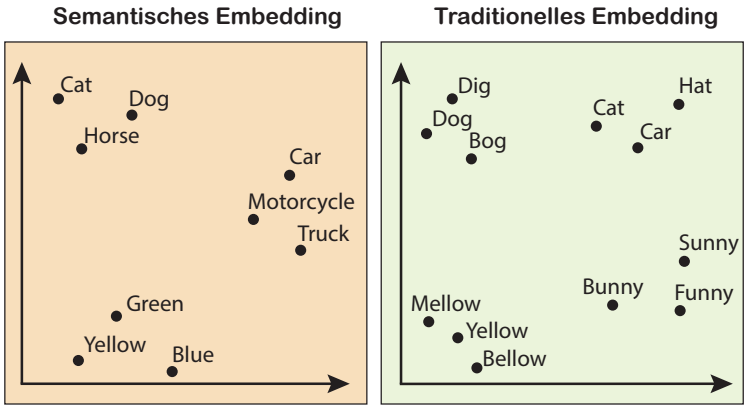

Abbildung 5.0.1: Beispielhafter Vergleich zwischen dem Aufbau eines semantischen und eines traditionellen Attributraums. Quelle: [WB16].

Dabei ist es das Ziel des semantischen Word Spottings, dass in der Ergebnisliste zunächst alle Wortabbilder des gesuchten Wortes aufgelistet werden und anschließend semantisch ähnliche Wortabbilder bezüglich der Anfrage folgen. Ein möglicher Anwendungsfall für das semantische Word Spotting ist die Suche in einem großen und unbekannten Dokumentenkorpus, wobei der Benutzer noch nicht genau weiß wonach er sucht. Wenn dieser zum Beispiel nach dem Wort *Sonntag* sucht, dann sollte

das System zunächst alle Vorkommen des Wortes *Sonntag* auflisten und anschließend weitere Wortabbilder von zum Beispiel anderen Wochentagen.

In der Arbeit von Wilkinson et al. [WB16] wurde die Idee des semantischen Word Spottings zum ersten Mal erwähnt und zudem ein Lösungsansatz vorgestellt. Dieser basiert auf dem im Kapitel 3.2 beschriebenen Word Spotting System sowie einem CharLSTM zur Bildung des semantischen Attributraums. Das System transformiert die Bilddeskriptoren dabei so in den Attributraum, dass semantisch ähnliche Wörter eine geringe Distanz und semantisch unähnliche Wörter eine große Distanz zueinander haben. Die Idee und das Modell können beispielhaft anhand der Abbildung 5.0.2 nachvollzogen werden. In Abbildung 5.0.3 befinden sich zudem beispielhafte Rückgabelisten, welche vom Wilkinson Modell erzeugt wurden. Hierbei wird ersichtlich, dass viele semantisch unähnliche Wortabbilder weit vorne in den Listen auftauchen. Aus diesem Grund wird in dem Kapitel 5.1 überprüft, ob das Verfahren aus [WB16] oder lediglich der vom CharLSTM generierte Attributraum für die fehlerhaften Resultate verantwortlich ist. Dafür wird zunächst der gelernte Attributraum des CharLSTMs für den GW Datensatz analysiert und mit den Attributräumen des Word2Vec und FastText Modells verglichen. Dabei sind dies zwei bekannte und viel verwendete Ansätze aus der NLP. In dem Kapitel 5.2 wird anschließend die Auswirkung der Attributräume auf die semantische Sortierung der Rückgabelisten im Word Spotting untersucht. Abschließend werden im Kapitel 5.3 die drei semantischen Modelle mit dem Wilkinson System evaluiert und die Ergebnisse diskutiert.

5.1 ANALYSE DER WORTMODELLE

Die Attributräume legen die semantischen Beziehungen zwischen einzelnen Wörtern fest und spielen wahrscheinlich eine fundamentale Rolle bei der Berechnung der Ergebnislisten für eine Word Spotting Anfrage. Deshalb wird in diesem Kapitel zunächst eine Analyse vom Attributraum des semantischen CharLSTMs durchgeführt, welches auf den GW Daten trainiert wurde. Anschließend wird dieser Attributraum mit den Attributräumen von unterschiedlich trainierten Word2Vec sowie FastText Modellen verglichen. Das Ziel dieser Wortmodellanalyse ist, dass die Qualität der semantischen Beziehungen in den jeweiligen gelernten Attributräumen untersucht wird und ein Eindruck über den Aufbau der Räume entsteht.

Bei der Analyse werden zu einer ausgewählten Menge von Anfragewörtern, jeweils die zehn semantisch ähnlichsten Wörtern aus einer großen Datenbank bestimmt. Die Datenbank besteht dabei aus den 20.000 häufigsten Wörtern des GWW Datensatzes. Bei einer vorherigen Analyse der Rückgabelisten für den George Washington Datensatz

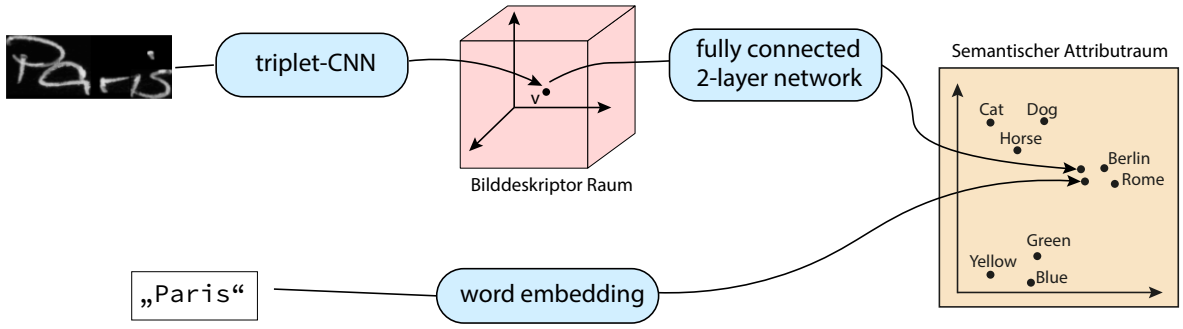

Abbildung 5.0.2: Die Abbildung visualisiert den Aufbau des semantischen Word Spotting Modells nach [WB16]. Hierbei werden sowohl die Wortabbilder als auch die Wörter selbst so in den Attributraum gemappt, dass semantisch ähnliche Wörter nah beieinander liegen und semantisch unähnliche Wörter weit voneinander entfernt sind. In diesem Beispiel ist die Distanz zwischen den Städten: *Paris*, *Berlin* und *Rom* gering, wodurch diese bei einer Städteanfrage weit vorne in der Ergebnisliste landen würden.

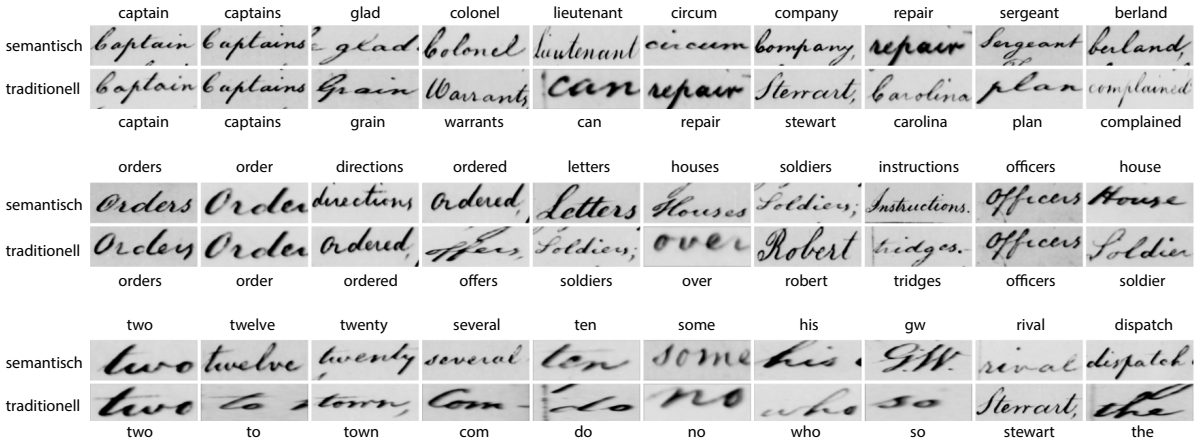

Abbildung 5.0.3: Vergleich zwischen der semantischen und PHOC-Repräsentation. Dabei werden jeweils die zehn ähnlichsten Ergebnisse für die Anfragen: *captain*, *orders* und *two* visualisiert, wobei jedes Wort nur einmal in der Ergebnisliste vorkommt. Die oberen Reihen gehören zur semantischen Repräsentation und die unteren zur PHOC-Repräsentation. Quelle: [WB16].

zeigte sich, dass die Wörter: *captain*, *sent*, *two* und *copy* die Vor- und Nachteile der semantischen Modelle sehr gut repräsentieren. Aus diesem Grund werden diese vier Wörter in der Analyse als Anfragen verwendet.

In den folgenden Unterkapiteln werden die Tabellen mit den zehn ähnlichsten Wörtern zu den jeweiligen Anfragen vorgestellt und die Ergebnisse diskutiert. Dabei stehen die eingeklammerten Werte hinter einem Wort für dessen Kosinusähnlichkeit bezüglich der Anfrage. Die Anfragewörter sind dabei repräsentativ für die jeweiligen Ergebnisse mit den Modellen.

5.1.1 *CharLSTM*

Das CharLSTM wurde mit demselben Verfahren wie in [WB16] beschrieben, auf dem GWW Datensatz trainiert. Die Ergebnisse für die oben genannten Anfragewörter befinden sich in der Tabelle 5.1.1. Da kein vortrainiertes Modell verfügbar war, kann leider kein Vergleich zwischen einem vortrainierten und einem spezialisierten Modell durchgeführt werden.

captain (1.0)	**sent** (1.0)	**two** (1.0)	**copy** (1.0)
captains (0.75)	nent (0.53)	three (0.41)	copying (0.59)
captaincy (0.67)	senting (0.53)	twenty (0.37)	cope (0.57)
captlieut (0.59)	tent (0.51)	twelve (0.34)	copies (0.50)
captainlieutenant (0.59)	went (0.51)	twentyeight (0.33)	cupy (0.50)
capt (0.55)	sents (0.50)	twentyfive (0.33)	copyist (0.47)
captn (0.53)	ient (0.50)	several (0.32)	scope (0.45)
capn (0.51)	lent (0.50)	seventeen (0.31)	cory (0.44)
captt (0.51)	dent (0.50)	few (0.31)	coty (0.43)
capts (0.49)	sert (0.43)	twentyone (0.31)	copied (0.43)

Tabelle 5.1.1: Die jeweils zehn ähnlichsten Wörter zu den Anfragen: *captain*, *sent*, *two* und *copy* mit dem CharLSTM.

Anhand der Ergebnisse wird ersichtlich, dass dieses Modell stark auf den Resultaten des n-gram Modells basiert und einen schlechten semantischen Attributraum lernt. Dabei sind die Ergebnisse repräsentativ für alle Wörter aus dem GW Datensatz. Es existieren ein paar wenige Wörter wie zum Beispiel: *captain* und *two* für die das Modell einigermaßen gute semantische Beziehungen lernt. Jedoch gilt für die meisten Wörter,

dass sie wenige bis hin zu keine semantischen Beziehungen lernen und die Distanzen in dem Raum der Editierdistanz entsprechen. Die Wörter *sent* und *copy* sind gute Beispiele dafür.

Bei dem Wort *captain* werden gute semantische Beziehungen bezüglich des George Washington Datensatzes gelernt. Dabei existieren viele militärische Abkürzungen in den Trainingsdaten, die semantisch ähnlich zu dem Anfragewort sind und auch in der Liste auftauchen. Jedoch sind die Ähnlichkeitswerte sehr gering. Gerade für die Wörter *captn*, *capt* und *capn* sollten die Werte deutlich größer sein, da sie in den Trainingsdaten im selben Kontext auftreten. Auch die semantische Bewertung des Wortes *captaincy* ist fraglich. Zudem kommt es zur Bildung neuer Wörter, da zum Beispiel das Wort *captain-lieutenant* nach der Vorverarbeitung zu *captainlieutenant* wird.

Die Liste mit den semantisch ähnlichsten Wörter bezüglich des Wortes *sent* ist sehr schlecht. Dabei erinnert diese Liste, aufgrund der geringen Editierdistanz zwischen dem Anfragewort und den Wörtern aus der Liste, eher an den Attributraum des n-gram Modells. Alle Wörter aus der Liste haben bis auf das Wort *sents* keine semantische Relevanz für das Anfragewort, dafür aber eine geringe Editierdistanz zu diesem. Zudem fehlen semantische Wörter wie zum Beispiel *forwarded* in der Liste, obwohl diese in den Trainingsdaten im selben Kontext wie *sent* auftauchen. Außerdem ist zu beachten, dass keines dieser Wörter durch eine fehlerhafte Transkription des Anfragewortes entstanden ist. Dabei stammen alle nicht korrekten englischen Wörter, aus der Worttrennung am Ende einer Zeile. Hierbei entsteht zum Beispiel das Wort *nent* aus den Wörtern *conti-nent* und *perma-nent* sowie das Wort *ient* aus dem Wort *conven-ient*.

Die Ergebnisse zu dem Wort *copy* sind ähnlich schlecht wie die von dem Wort *sent*. Hierbei haben die meisten Wörter der Liste keine semantische Relevanz für die Anfrage, jedoch eine geringe Editierdistanz zu dieser. Auch hier sind die Wörter keine Rechtschreibfehler, sondern lediglich Resultate aus einer Worttrennung. Hierbei stammt zum Beispiel das Wort *cupy* aus dem Wort *oc-cupy*. Ein weiterer negativer Aspekt dieser Liste ist, dass die semantisch ähnlichen Wörter *copied* und *copies* nur einen geringen Ähnlichkeitswert haben.

Die Ergebnisliste für das Wort *two* ist auf dem ersten Blick sehr gut, da sich in der Liste nur semantisch relevante Treffer befinden und auch Wörter wie *several* und *few* weit oben stehen. Jedoch ist auch hier das Muster aus dem n-gram Modell ersichtlich, da die meisten dieser Wörter mit den Buchstaben *tw* anfangen. Zudem haben die Wörter aus der Liste, trotz ihrer hohen semantischen Relevanz, nur einen geringen Ähnlichkeitswert. Dadurch ist es der Fall, dass sie sich nur geringfügig von semantisch unähnlichen Wörtern absetzen und dies zu einer schlecht sortierten Rückgabeliste im Word Spotting führen kann.

Insgesamt ist festzustellen, dass dieses Modell aufgrund der Nähe zum n-gram Modell und der geringen Größe der Trainingsdaten, die semantischen Beziehungen zwischen den meisten Wörtern nicht gut repräsentieren kann. Im nachfolgenden Kapitel wird ersichtlich, dass Word2Vec und FastText mithilfe von vortrainierten Modellen deutlich bessere semantische Attributräume lernen können.

5.1.2 *Word2Vec*

Für die Analyse des Word2Vec Modells werden insgesamt drei unterschiedlich trainierte Modelle betrachtet. Hierbei wird zunächst der Attributraum von einem auf dem Google News Datensatz vortrainierten Word2Vec Modell untersucht. Anschließend wird ein Word2Vec Modell nur auf dem GWW Datensatz trainiert und analysiert. Abschließend wird der erzeugte Attributraum von einer Spezialisierung des vortrainierten Modells auf dem GWW Datensatz betrachtet.

Das vortrainierte Word2Vec Modell stammt von [Goo13], hat ein Vokabular von drei Millionen Wörtern und wurde auf ungefähr 100 Milliarden Wörtern aus dem Google News Datensatz trainiert. Zudem haben die Attributvektoren eine Dimensionalität von 300. Die zehn ähnlichsten Wörter zu den repräsentativen Anfragen befinden sich in der Tabelle 5.1.2.

captain (1.0)	**sent** (1.0)	**two** (1.0)	**copy** (1.0)
skipper (0.77)	sending (0.76)	three (0.93)	copies (0.71)
captains (0.71)	send (0.74)	four (0.90)	copied (0.51)
captaincy (0.60)	sends (0.65)	five (0.84)	read (0.46)
coach (0.53)	forwarded (0.62)	six (0.82)	reprint (0.45)
squad (0.50)	despatched (0.56)	seven (0.81)	document (0.44)
mate (0.46)	dispatched (0.54)	eight (0.80)	printed (0.43)
team (0.45)	circulated (0.50)	nine (0.75)	copying (0.42)
skippers (0.45)	delivered (0.49)	several (0.71)	perusal (0.42)
commodore (0.45)	received (0.49)	couple (0.68)	manuscript (0.42)

Tabelle 5.1.2: Die jeweils zehn ähnlichsten Wörter zu den Anfragen: *captain*, *sent*, *two* und *copy* mit dem vortrainierten Word2Vec Modell.

Dieses Modell liefert bereits sehr gute Ähnlichkeitsbeziehungen zu den einzelnen Wörtern. Gerade die Listen zu den beiden Anfragewörter *sent* und *two* beinhalten

nur semantisch ähnliche Wörter. Die Liste von dem Wort *copy* ist auch gut, da die semantisch ähnlichen Wörter: *copies*, *copied* und *reprint* weit oben in der Liste stehen und der Rest der Wörter aus dem Dokumentenbereich stammt. Dabei ist allgemein positiv zu beachten, dass die semantischen Beziehungen der Wörter gut durch die Ähnlichkeitswerte repräsentiert werden. In der Liste von *sent* ist das Wort *send* zum Beispiel deutlich ähnlicher als *forwarded* oder *delivered*. Auch bei den Zahlen ist dies zu beobachten, da es einen starken Zusammenhang zwischen dem mathematischen Abstand der Zahlen und den Ähnlichkeitswerten gibt. Das Problem ist jedoch, dass dieses Modell gerade für das Anfragewort *captain* nicht auf die Wörter des GW Datensatzes angepasst ist. Hierbei sollten eher Wörter mit Bezug zum Militär ähnlicher sein als Wörter aus der Schiffsfahrt (*skipper*, *skippers*) oder dem Mannschaftssport (*coach*, *squad*, *mate*, *team*). Aus diesem Grund werden in Tabelle 5.1.3 zunächst die zehn ähnlichsten Wörter mit einem Word2Vec Modell angegeben, dass lediglich auf dem GWW Datensatz trainiert wurde.

captain (1.0)	**sent** (1.0)	**two** (1.0)	**copy** (1.0)
capt (0.87)	forwarded (0.79)	three (0.85)	duplicate (0.75)
ensign (0.84)	delivered (0.77)	four (0.84)	inclosed (0.71)
col (0.79)	brought (0.72)	six (0.75)	transcript (0.69)
colonel (0.78)	send (0.71)	five (0.73)	extract (0.68)
lieutenant (0.77)	deliverd (0.67)	seven (0.70)	copies (0.66)
lieut (0.75)	returned (0.67)	eight (0.68)	enclosed (0.66)
lieutt (0.74)	dispatched (0.67)	ten (0.65)	memorial (0.64)
cap (0.72)	drawn (0.66)	thirteen (0.64)	petition (0.63)
colo (0.72)	transmitted (0.64)	twelve (0.64)	original (0.63)

Tabelle 5.1.3: Die jeweils zehn ähnlichsten Wörter zu den Anfragen: *captain*, *sent*, *two* und *copy* mit dem Word2Vec Modell, welches nur auf den GW Daten trainiert wurde.

Zunächst kann festgestellt werden, dass der Attributraum die semantischen Beziehungen der Wörter im Vergleich zum CharLSTM deutlich besser repräsentiert. Dabei enthalten die Listen fast ausschließlich semantisch ähnliche Wörter bezüglich der Anfragen und weisen zudem gute Ähnlichkeitsbewertungen auf. Gerade bei dem Wort *captain* ist zu beachten, dass alle militärischen Abkürzungen und Ränge, die für das Wort relevanten sind, in der Liste vorkommen. Auch bei dem Wort *sent* befinden sich bis auf das Wort *drawn* ausschließlich semantisch ähnliche Wörter in der Liste. Für das Wort *copy* ist die Liste schon etwas schlechter, da zwar semantisch ähnliche Wörter wie

zum Beispiel: *duplicate*, *copies* und *transcript* weit oben in der Liste stehen, aber Wörter wie *inclosed*, *enclosed* und *petition* semantisch zu hoch bewertet werden. Insgesamt ist dies aber ein sehr guter semantischer Attributraum für den GW Datensatz, da die Trainingsmenge ziemlich klein war und trotzdem die wichtigsten Beziehungen für die Anfragewörter ermittelt werden konnten.

Im Folgenden wird analysiert, ob die Spezialisierung des vortrainierten Word2Vec Modells auf dem GWW Datensatz zu einem besseren semantischen Attributraum führt. Dafür wurde das vortrainierte Modell mit dem benutzerfreundlichen Python Paket *gensim* und den vorgegebenen Standardparametern spezialisiert. Die Ergebnisse befinden sich in der Tabelle 5.1.4.

captain (1.0)	**sent** (1.0)	**two** (1.0)	**copy** (1.0)
capt (0.89)	brought (0.82)	three (0.93)	copies (0.76)
colonel (0.89)	delivered (0.81)	four (0.90)	yours (0.75)
lieutenant (0.86)	forwarded (0.79)	five (0.85)	list (0.74)
col (0.86)	send (0.78)	six (0.84)	inclosed (0.73)
lieut (0.85)	returned (0.78)	seven (0.83)	writing (0.72)
major (0.85)	ordered (0.77)	eight (0.81)	extract (0.72)
ensign (0.83)	gone (0.72)	ten (0.80)	letter (0.71)
colo (0.83)	transmitted (0.72)	several (0.80)	photostat (0.71)
doctor (0.82)	left (0.71)	few (0.78)	note (0.71)

Tabelle 5.1.4: Die jeweils zehn ähnlichsten Wörter zu den Anfragen: *captain*, *sent*, *two* und *copy* mit dem spezialisierten Word2Vec Modell.

Hierbei ist zunächst festzustellen, dass die Ergebnisse für die Anfragewörter *sent* und *copy* deutlich schlechter geworden sind. Dabei sind die semantisch unähnlichen Wörter *gone* und *left* in die Liste des Wortes *sent* gerutscht und die semantisch sehr ähnlichen Wörter *sending* und *sends* nicht mehr vorhanden. Außerdem befinden sich nun die semantisch unähnlichen Wörter *yours*, *list*, *photostat* und *note* in der Liste des Wortes *copy*. Es wird jedoch ersichtlich, dass die Resultate für die beiden Wörter *captain* und *two* weiterhin gut sind. Dabei enthält die Liste für das Wort *captain* viele Wörter zu Militärrängen sowie die wichtigsten Abkürzungen aus dieser Kategorie. In der Liste des Wortes *two* befinden sich die Zahlen in aufsteigender Reihenfolge sowie die Wörter *several* und *few*. Hierbei sind die gesteigerten Ähnlichkeitswerte, sowie die Abwertung des Wortes *nine* zu beachten. Zudem ist es positiv, dass für das Wort *captain* zwar geläufige Abkürzungen wie zum Beispiel *capt*, *col*, *lieut* und *colo* in der

Liste vorkommen, aber die ungewöhnlichen Abkürzungen wie *captt* oder *capn* nicht vorhanden sind.

5.1.3 *FastText*

Wie bereits bei der Analyse des Word2Vec Modells, werden auch für das FastText Modell insgesamt drei Attributräume von unterschiedlich trainierten Modellen analysiert. Dafür wird zunächst der Attributraum eines FastText Modells untersucht, welches sowohl auf den Daten des *Common Crawl Datensatzes* sowie der *Wikipedia* vortrainiert wurde. Zudem wird das auf dem GWW Datensatz trainierte Modell und das auf diesen Daten spezialisierte Modell analysiert.

Das vortrainierte FastText Modell stammt aus [MGB+18] und wurde mit zwei großen Datensätzen vortrainiert. Der erste Datensatz ist der *Common Crawl Datensatz*. Dieser besteht aus mehreren Petabytes von Daten, die über acht Jahre beim Web-Crawling gesammelt wurden. Der zweite Datensatz besteht aus den Daten der englischen Wikipedia von Juni 2017 und umfasst insgesamt einen Textkorpus mit über neun Milliarden Wörtern. Die Attributvektoren des FastText Modells haben wie auch bei Word2Vec eine Dimensionalität von 300. Die zehn ähnlichsten Wörter zu den oben genannten Anfragen befinden sich in der Tabelle 5.1.5.

captain (1.0)	**sent** (1.0)	**two** (1.0)	**copy** (1.0)
captains (0.78)	send (0.74)	three (0.94)	copies (0.73)
skipper (0.71)	sending (0.71)	four (0.90)	copied (0.54)
captaincy (0.65)	forwarded (0.68)	five (0.82)	copying (0.52)
commander (0.62)	sends (0.64)	six (0.82)	reprint (0.52)
lieutenant (0.62)	received (0.63)	eight (0.79)	facsimile (0.51)
sailor (0.56)	returned (0.57)	seven (0.79)	paste (0.50)
admiral (0.56)	delivered (0.55)	several (0.77)	print (0.49)
boatswain (0.56)	despatched (0.55)	nine (0.75)	manuscript (0.49)
midshipman (0.54)	dispatched (0.55)	couple (0.71)	duplicate (0.49)

Tabelle 5.1.5: Die jeweils zehn ähnlichsten Wörter zu den Anfragen: *captain*, *sent*, *two* und *copy* mit dem vortrainierten FastText Modell.

Mit dem vortrainierten FastText Modell wird ein Attributraum erzeugt, welcher gute semantische Beziehungen für den GW Datensatz kodiert. Dabei enthält die Liste

von dem Anfragewort *captain* mehrere Wörter aus dem Militärbereich, wie *commander*, *lieutenant* und *admiral*. Jedoch sind in dieser Liste mit *skipper* und *sailor* auch Wörter aus dem Bereich der Schiffsfahrt vorhanden und semantisch hoch bewertet. Die semantischen Listen für die Wörter *sent* und *two* sind sehr gut, da sie ausschließlich semantisch ähnliche Wörter enthalten und diese passende semantische Beziehungswerte haben. Die Liste des Wortes *copy* kodiert die semantischen Beziehungen der Wörter auch sehr gut, da die Wörter: *copies*, *copied* und *copying* eine starke semantische Ähnlichkeit bezüglich der Anfrage haben und die restlichen Wörter aus dem Dokumentenbereich stammen. Zudem ist es erwähnenswert, dass ein so seltenes Wort wie *facsimile* korrekterweise in der Liste auftaucht. Ein negativer Punkt der Liste ist jedoch, dass es einen ziemlich großen Sprung der Ähnlichkeitsbewertungen bei den Wörtern *copies* und *copied* gibt und dass die Bewertung der Wörter *copied* und *paste* fast identisch ist. Wie bereits beim Word2Vec Modell, wird auch beim FastText Modell überprüft, wie gut die semantischen Beziehungen im Attributraum sind, wenn das Modell nur auf dem GWW Datensatz trainiert wurde. Die Ergebnisse dieser Analyse befinden sich in der Tabelle 5.1.6.

captain (1.0)	**sent** (1.0)	**two** (1.0)	**copy** (1.0)
captaincy (0.95)	sented (0.96)	three (0.86)	copyist (0.86)
captainlieutenant (0.92)	sentrys (0.91)	six (0.80)	cope (0.86)
captains (0.91)	sentinel (0.89)	four (0.78)	copies (0.86)
captt (0.90)	sen (0.87)	twelve (0.75)	transcript (0.83)
capt (0.89)	delivered (0.86)	twothirds (0.72)	copper (0.83)
captlieut (0.88)	sentinels (0.84)	seven (0.71)	enclosd (0.83)
captn (0.85)	senr (0.81)	sixteen (0.71)	inclosd (0.83)
captns (0.84)	sentry (0.80)	five (0.71)	enclos (0.82)
lieutenantcolonel (0.83)	seneka (0.79)	fifteen (0.71)	inclos (0.82)

Tabelle 5.1.6: Die jeweils zehn ähnlichsten Wörter zu den Anfragen: *captain*, *sent*, *two* und *copy* mit dem FastText Modell, welches nur auf den GW Daten trainiert wurde.

Diese Tabelle erinnert stark an die Ergebnisse von dem CharLSTM. Auch hier werden die Artefakte der Trainingsdaten ersichtlich.

Das vortrainierte FastText Modell, wird wie bereits bei Word2Vec, mithilfe von *gensim* und den vorgegebenen Standardparametern auf dem GWW Datensatz spezialisiert. Die Ergebnisse für die Analyse des Attributraums befinden sich in der Tabelle 5.1.7.

captain (1.0)	**sent** (1.0)	**two** (1.0)	**copy** (1.0)
captaincy (0.98)	sented (0.89)	three (0.96)	copyist (0.93)
captains (0.97)	delivered (0.89)	four (0.95)	copies (0.86)
capts (0.97)	forwarded (0.88)	eight (0.94)	transcript (0.86)
captt (0.97)	sentrys (0.88)	eighteen (0.94)	transcripts (0.85)
captn (0.97)	boarded (0.87)	twentyeight (0.93)	memorandum (0.85)
captns (0.97)	fitted (0.87)	eighty (0.93)	copied (0.85)
captainlieutenant (0.95)	handed (0.87)	fifteen (0.93)	enclosd (0.84)
captlieut (0.95)	warded (0.86)	nineteen (0.93)	memoranda (0.84)
capt (0.94)	forwards (0.86)	thirteen (0.93)	memoran (0.84)

Tabelle 5.1.7: Die jeweils zehn ähnlichsten Wörter zu den Anfragen: *captain*, *sent*, *two* und *copy* mit dem spezialisierten FastText Modell.

Auch das FastText Modell ist durch die Spezialisierung auf dem GWW Datensatz schlechter geworden. Zwar konnten für das Wort *captain* die in den George Washington Daten verwendeten Abkürzungen, im Gegensatz zum vortrainierten Modell, höher bewertet werden, jedoch haben sich für die Wörter *sent* und *copy* die Resultate verschlechtert. Gerade bei dem Anfragewort *sent* sind viele semantisch unähnliche Wörter, wie *sentrys*, *boarded*, *fitted* und *warded* zur Liste hinzugekommen und die semantisch ähnlichen Wörter, wie *sending*, *despatched*, *dispatched* und *send* schlechter bewertet worden. Für das Wort *two* sind die semantischen Beziehungen aber weiterhin sehr gut.

Insgesamt kann festgehalten werden, dass sowohl das vortrainierte Word2Vec als auch das vortrainierte FastText Modell einen sehr guten semantischen Attributraum erzeugen und diese auch für den George Washington Datensatz verwendet werden können. Die Spezialisierungen dieser vortrainierten Modelle führt für die meisten Anfragewörter zu einer Verschlechterung der semantischen Beziehungen im Attributraum. Dies liegt jedoch voraussichtlich an der schlechten Qualität der Trainingsdaten. Zudem wird durch diese Analyse ersichtlich, dass das semantische CharLSTM, im Vergleich zum FastText und Word2Vec Modell, einen deutlich schlechteren semantischen Attributraum aufweist und die Beziehungen der Wörter in diesem Raum einen starken Bezug zum n-gram Modell haben. Aus diesem Grund wird in dem nachfolgenden Kapitel der Einfluss von den Attributräumen auf die Berechnung der Ergebnislisten mit dem Word Spotting Ansatz von Wilkinson untersucht.

5.2 ANALYSE DER RÜCKGABELISTEN

In diesem Kapitel wird überprüft, in wie fern die semantischen Attributräume einen Einfluss auf die Berechnungen der Ergebnislisten im Wilkinson System haben und ob die schlechten semantischen Beziehungen in dem Attributraum des CharLSTMs alleine für die Resultate aus Abbildung 5.0.3 verantwortlich sein könnten. Dafür werden zunächst mit dem Wilkinson Ansatz die Rückgabelisten für die Anfragewörter und die semantischen Modelle aus dem vorherigen Kapitel ermittelt und die Ergebnisse diskutiert. Dabei handelt es sich bei den Ergebnislisten nicht um die traditionellen Rückgabelisten beim Word Spotting, sondern sie enthalten lediglich die zehn ähnlichsten Wortabbilder bezüglich der Anfrage und sorgen dafür, dass jede Transkription nur maximal einmal darin auftaucht. Die traditionelle Sortierung der Ergebnislisten wird jedoch berücksichtigt und für jede vorkommende Transkription nur das Wortabbild mit der höchsten Positionierung in der Ergebnisliste behalten. Für die Bewertung ist zudem interessant, dass die Anfragewörter *sent*, *captain* und *two* im Training des Einbettungsnetzwerks vorkamen und das Wort *copy* nicht. Außerdem besteht die Datenbank aus den Testdaten des ersten Kreuzvalidierungsblocks vom GW Datensatz.

Um den Einfluss der Attributräume auf die Ergebnislisten zu testen, werden im Folgenden, für jedes der semantischen Modelle aus dem vorherigen Kapitel, die Ergebnislisten der vier repräsentativen Anfragewörter mit dem Wilkinson Ansatz bestimmt. Anschließend werden diese Listen mit den zugehörigen Tabellen aus dem vorherigen Kapitel in Beziehung gesetzt und analysiert. Dabei wird überprüft, in wie fern sich die vorhandenen Muster aus den Tabellen, in den Ergebnislisten widerspiegeln.

Abbildung 5.2.1: Die ermittelten Rückgabelisten für die repräsentativen Anfragewörter. Dabei wird der semantische Attributraum vom semantischen CharLSTM erzeugt.

5.2.1 *CharLSTM*

In Abbildung 5.2.1 sind die ermittelten Ergebnislisten für das semantische CharLSTM visualisiert. Dabei wird ersichtlich, dass sich die Muster aus der Tabelle 5.1.1 auf die Ergebnislisten übertragen haben. Somit wird auch hier die Nähe des semantischen CharLSTMs zum n-gram Modell ersichtlich. Gerade die beiden Rückgabelisten der Wörter *copy* und *sent* enthalten mehrere Ergebnisse mit einer geringen Editierdistanz zur Anfrage. Aber auch bei dem Anfragewort *two* ist erkennbar, dass zwar viele semantisch ähnliche Wortabbilder in der Liste zu finden sind, jedoch mehrere mit dem Wort *tw* anfangen. Diese Wörter wurden auch schon bei der Analyse dieses Modells und dem Anfragewort *two* hoch bewertet. Für die Ergebnisliste des Wortes *captain* ist die Beziehung zur Analyse des vorherigen Kapitels nicht möglich, da die Tabelle lediglich Abkürzungen für dieses Anfragewort enthält und daher kein Muster extrahiert werden konnte. Insgesamt kann für das semantische CharLSTM festgestellt werden, dass auch bei den Ergebnislisten die Nähe zum n-gram Modell existiert und daher der Attributraum einen großen Einfluss auf die Berechnungen der Rückgabelisten mit dem Wilkinson Ansatz und dem CharLSTM hat.

5.2.2 *Word2Vec*

In den Abbildungen 5.2.2, 5.2.3 und 5.2.4 werden jeweils die Ergebnislisten für das vortrainierte, nur auf dem GWW Datensatz trainierte und das spezialisierte Word2Vec Modell vorgestellt. Auch hierbei ist der Zusammenhang zwischen dem gelernten Attributraum und den erzeugten Ergebnislisten ersichtlich. Dafür steht beispielhaft die Rückgabeliste für das Anfragewort *two* beim vortrainierten Word2Vec Modell in Abbildung 5.2.2. Hierbei enthält die Tabelle 5.1.2 für diese Anfrage, im Gegensatz zu den anderen Modellen, nicht nur Wörter von Zahlen sondern auch Wörter wie *several* und *couple*. Dies führt dazu, dass die semantisch ähnlichen Wörter *pair*, *all* und *some* weit oben in der Ergebnisliste für das Anfragewort *two* stehen. Generell bestätigen die Analysen den Trend, dass Modelle mit einem guten semantischen Attributraum auch gute semantische Ergebnislisten produzieren und das Modelle mit schlechten semantischen Attributräumen zu schlechten semantischen Ergebnislisten führen.

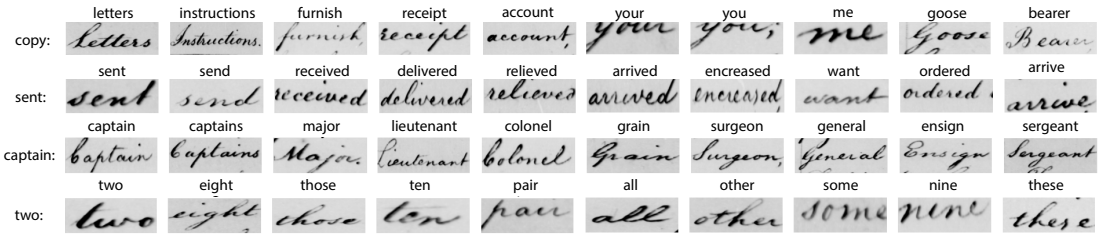

Abbildung 5.2.2: Die ermittelten Rückgabelisten für die repräsentativen Anfragewörter. Dabei wird der semantische Attributraum vom vortrainierten Word2Vec Modell erzeugt.

Abbildung 5.2.3: Die ermittelten Rückgabelisten für die repräsentativen Anfragewörter. Dabei wird der semantische Attributraum vom Word2Vec Modell erzeugt, welches lediglich auf den Daten des GWW Datensatzes trainiert wurde.

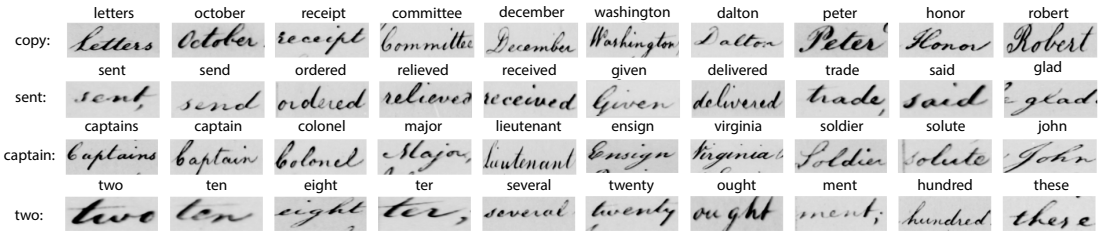

Abbildung 5.2.4: Die ermittelten Rückgabelisten für die repräsentativen Anfragewörter. Dabei wird der semantische Attributraum vom spezialisierten Word2Vec Modell erzeugt.

5.2.3 *FastText*

In den Abbildungen 5.2.5, 5.2.6 und 5.2.7 werden jeweils die Ergebnislisten für das vortrainierte, nur auf dem GWW Datensatz trainierte und das spezialisierte FastText Modell vorgestellt. Dabei zeigen sich dieselben Erkenntnisse wie bereits beim

Word2Vec Modell. Hierbei fällt jedoch auf, dass das vortrainierte FastText Modell die subjektiv besten semantischen Rückgabelisten für diese Anfragewörter liefert. Jedoch wird auch bei dem FastText Modell ersichtlich, dass bei den nicht im Training vorkommenden Anfragewörtern, die Wortabbilder mit derselben Transkription wie die Anfrage weit hinten in der Ergebnisliste auftauchen.

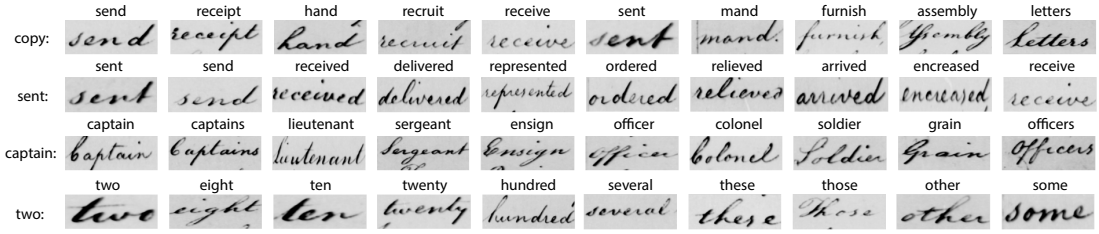

Abbildung 5.2.5: Die ermittelten Rückgabelisten für die repräsentativen Anfragewörter. Dabei wird der semantische Attributraum vom vortrainierten FastText Modell erzeugt.

Abbildung 5.2.6: Die ermittelten Rückgabelisten für die repräsentativen Anfragewörter. Dabei wird der semantische Attributraum vom FastText Modell erzeugt, welches lediglich auf den Daten des GWW Datensatzes trainiert wurde.

Abbildung 5.2.7: Die ermittelten Rückgabelisten für die repräsentativen Anfragewörter. Dabei wird der semantische Attributraum vom spezialisierten FastText Modell erzeugt.

5.2.4 *Fazit*

Insgesamt kann festgestellt werden, dass es einen starken Zusammenhang zwischen den gelernten Attributräumen der Modelle und der mit diesen Modellen erzeugten Ergebnislisten gibt. Dabei wirkt sich die Qualität der gelernten semantischen Beziehungen in dem Raum stark auf die Platzierung der Wortabbilder in einer Ergebnisliste aus. Jedoch wird auch ersichtlich, dass nicht nur der Attributraum für die Qualität eines semantischen Word Spotting Systems verantwortlich ist, sondern auch die Abbildung der Wortabbilder in den Attributraum eine entscheidende Rolle spielt. Dies wird auch anhand der wichtigen Erkenntnis aus dieser Analyse ersichtlich. Hierbei werden nämlich im semantischen CharLSTM, aufgrund der Nähe zum n-gram Modell, die nicht im Training vorgekommenen Wörter zwar weit vorne in der Ergebnisliste platziert, jedoch folgen darauf nur wenige bis keine semantischen Wortabbilder. Bei den FastText und Word2Vec Modellen ist es hingegen so, dass die Wortabbilder von nicht im Training vorkommenden Wörtern weit hinten in den Ergebnislisten platziert werden und die semantische Qualität der Rückgabelisten dann stark von der Qualität des gelernten Attributraums abhängt. Wie sich diese Erkenntnis auf die Bewertung der analysierten Modelle auswirkt, wird im nachfolgenden Kapitel für den GW Datensatz und die IAM Datenbank evaluiert.

5.3 EVALUATION DER SEMANTISCHEN MODELLE

In diesem Abschnitt werden die unterschiedlich trainierten FastText und Word2Vec Modelle auf dem GW Datensatz und der IAM Datenbank evaluiert und die Ergebnisse anschließend diskutiert sowie mit den Resultaten des semantischen CharLSTMs verglichen.

Die Evaluationsergebnisse für die semantischen Modelle auf dem GW Datensatz befinden sich in der Tabelle 5.3.1. Hierbei fällt zunächst auf, dass die trainierten Word2Vec und FastText Modelle bezüglich der QbE Ergebnisse zwar vergleichbare MAP Werte zum CharLSTM liefern, deren Resultate bei dem QbS Ansatz jedoch deutlich geringer ausfallen. Beim FastText Modell werden dabei die besten Resultate mit dem nur auf dem GWW Datensatz trainierten Modell erzielt, wohingegen beim Word2Vec Ansatz das vortrainierte Modell am besten funktioniert. Insgesamt muss jedoch festgestellt werden, dass die Ergebnisse von allen Word2Vec und FastText Modellen für diesen einfachen Datensatz zu gering ausfallen und bezüglich der MAP Werte deutlich schlechter abschneiden als das semantische CharLSTM.

Modell	*Trainingsdaten*	*QbE*	*QbS*
CharLSTM (semantisch)	**GWW**	97.02	81. 83
Word2Vec	**GWW**	92.31	53.08
	vortrainiert	93.99	59.59
	vortrainiert + GWW	90.86	53.36
FastText	**GWW**	95.36	66.44
	vortrainiert	94.21	58.73
	vortrainiert + GWW	87.57	43.80

Tabelle 5.3.1: Die MAP Werte für die Evaluation der unterschiedlich trainierten semantischen Modelle auf dem GW Datensatz.

Wie in der Tabelle 5.3.2 zu sehen, werden bei der Evaluation der unterschiedlich trainierten Word2Vec und FastText Modelle auf der IAM Datenbank, die Differenzen zwischen den MAP Werten dieser Modelle und denen des CharLSTMs deutlich größer. Dabei muss festgestellt werden, dass die trainierten Word2Vec und FastText Modelle, bezüglich der MAP Werte, auf diesem schwierigeren Datensatz versagen und somit für dieses Bewertungsmaß keine Alternative zum CharLSTM bieten können. Gerade die Modelle, die auf der Penn Treebank trainiert oder spezialisiert wurden, platzieren die gesuchten Wortabbilder bei fast allen Anfragewörtern weit hinten in den Ergebnislisten und führen damit zu einem geringen MAP Wert. Lediglich die vortrainierten Word2Vec und FastText Modelle konnten die gesuchten Wortabbilder für deutlich mehr Anfragewörter weiter vorne in den Ergebnislisten plazieren und damit einen höheren, aber immer noch im Vergleich zum CharLSTM geringen QbS Wert erreichen.

Modell	*Trainingsdaten*	*QbE*	*QbS*
CharLSTM (semantisch)	**Penn Treebank**	81.98	84.45
Word2Vec	**Penn Treebank**	1.45	0.33
	vortrainiert	44.08	19.61
	vortrainiert + Penn Treebank	32.56	0.08
FastText	**Penn Treebank**	22.15	0.05
	vortrainiert	46.71	20.37
	vortrainiert + Penn Treebank	3.08	0.44

Tabelle 5.3.2: Die MAP Werte für die Evaluation der unterschiedlich trainierten semantischen Modelle auf der IAM Datenbank.

Im Folgenden wird eine Intuition dafür gegeben, wieso die Ergebnisse für die Word2Vec und FastText Modelle im Gegensatz zum semantischen CharLSTM, beim Word Spotting mit dem Wilkinson Ansatz, so schlecht abschneiden, obwohl im Kapitel 5.2 beispielhaft gezeigt werden konnte, dass deren Ergebnislisten für die vorgestellten Anfragewörter meist deutlich bessere semantische Resultate beinhalteten. Da es für das Scheitern der Word2Vec und FastText Modelle unterschiedliche Ursachen gibt, werden diese im Folgenden separat vorgestellt.

Bei der Verwendung des Word2Vec Modells im Wilkinson Ansatz, ist das Modell selbst für die schlechten Resultate beim semantischen Word Spotting verantwortlich. Dabei ist das Hauptproblem, dass das Modell für die Abbildung eines Wortes zu einem Attributvektor, das Wort als Ganzes betrachtet und nicht auf einem n-gram Verfahren oder ähnlichem basiert. Das Modell erhält nämlich einen One-Hot-Kodierten Vektor als Eingabe und bestimmt damit die zugehörige Attribut-Repräsentation. Hierbei wird der Eingabevektor so konstruiert, dass die zum Wort korrespondierende Stelle im Vektor eine 1 erhält und der Rest des Vektors aus 0 Werten besteht. Dadurch wird lediglich eine 1:1 Abbildung zwischen einem Wort und dessen Attribut-Repräsentation gelernt. Jedoch ist es angesichts dieser Extrapolation generell ausgeschlossen, den korrekten Attributvektor für ein nicht im Training vorkommendes Wort zu bestimmen. Aufgrund der nicht vorhandenen Muster in der Abbildungsfunktion, kann auch das Einbettungsnetzwerk nur eine 1:1 Abbildung für die Trainingswörter lernen und damit keinen korrekten Attributvektor für die nicht im Training vorkommenden Wörter bestimmen. Somit wird ersichtlich, wieso das semantische Word Spotting mit dem Wilkinson System und dem Word2Vec Modell auf den beiden Datensätzen so schlecht abgeschnitten hat. Dabei resultiert der geringe MAP Wert von circa 53 beim QbS

Ansatz auf dem GW Datensatz daraus, dass für im Training vorkommende Wörter gute Resultate bezüglich der MAP Werte erzielt werden können und für nicht im Training vorkommende Wörter diese Werte bei näherungsweise 0 liegen. Somit ist das Scheitern des Word2Vec Ansatzes für das Word Spotting System von Wilkinson auf dem Ansatz selbst und nicht auf das System zurückzuführen. Wenn jedoch alle in der Praxis relevanten Wörter im Training vorkommen würden, dann könnte der Word2Vec Ansatz im Zusammenhang mit dem Wilkinson System trotzdem funktionieren. Dies ist allerdings für die meisten Szenarien unrealistisch und der Word2Vec Ansatz deshalb auch nicht besonders geeignet für das Word Spotting System von Wilkinson et al.

Beim FastText Ansatz hat das Einbettungsnetzwerk aus dem Wilkinson System, im Gegensatz zum Word2Vec Ansatz, gute Voraussetzungen, um die Abbildungen von nicht im Training vorkommenden Wörtern zu deren korrekten Attributvektoren zu lernen. Dies liegt daran, dass das FastText Modell auf n-grammen basiert und den Attributvektor eines Wortes durch die Summe seiner kodierten n-gramme berechnet. Somit existiert ein lernbares Muster für die Abbildung von Wörtern zu deren Attribut-Repräsentationen. Der Grund, wieso das Einbettungsnetzwerk trotzdem keine korrekte Abbildung zwischen den nicht im Training vorhandenen Wörtern und deren Attributvektoren lernt, liegt vermutlich an dem zweistufigen System des Wilkinson Ansatzes. Hierbei wird nämlich für ein Wortabbild zunächst ein Bilddeskriptor berechnet und mithilfe dieses Bilddeskriptors dessen Attribut-Repräsentation vorhergesagt. Die Bilddeskriptoren sind zwar gut darin, Bilder von unterschiedlichen Transkription im Bilddeskriptor-Raum voneinander zu trennen und gleichzeitig Wortabbilder mit derselben Transkription in dem Raum nah beieinander zu platzieren, jedoch ist es unwahrscheinlich, dass sie dabei Informationen über die Struktur der Wörter berücksichtigen. Somit kann anhand eines Bilddeskriptors im Normalfall kein Rückschluss auf die enthaltenen, und für den FastText Ansatz wichtigen, n-gramme des zugehörigen Wortabbildes erlangt werden. Dadurch ist es für das Einbettungsnetzwerk schwierig bis unmöglich, aus dem im Training gesehenen Bilddeskriptoren die korrekten n-gram Kodierungen des FastText Modells zu extrahieren. Daher ist es deutlich wahrscheinlicher, dass das Netzwerk nicht die korrekte Abbildungsfunktion lernt, sondern ein Overfitting entsteht und die Abbildungen auswendig gelernt werden. Das führt dazu, dass die Wörter, die nicht im Training des Einbettungsnetzwerks vorkamen, auf einen FastText Vektor projiziert werden, welcher eine große Distanz zur korrekten Repräsentation des Wortes hat. Damit gilt für jedes dieser Wörter, dass die Wortabbilder mit dieser Transkription, weit hinten in den Ergebnislisten platziert und dadurch, wie bereits beim Word2Vec Ansatz, MAP Werte von näherungsweise 0 erzielt werden. Für im Training vorkommende Wörter, werden hingegen die korrekten Abbildungen vorhergesagt und dadurch immerhin noch ein MAP Wert von circa 66

mit dem vortrainierten FastText Modell auf dem GW Datensatz erreicht. Das beschriebene Problem mit dem FastText Ansatz kann beispielhaft anhand der Visualisierung in Abbildung 5.3.1 nachvollzogen werden.

Die hohen MAP Werte beim Word Spotting mit dem Wilkinson Ansatz und dem CharLSTM resultieren daraus, dass die meisten Beziehungen im semantischen Attributraum sehr ähnlich zu denen im n-gram Modell sind. Dabei gibt es in diesem Attributraum jedoch auch gute semantische Beziehungen für manche Wörter, wie zum Beispiel für *captain* und *two*. Dies ist beispielhaft anhand des semantischen Raumes in der Abbildung 5.3.2 ersichtlich. Dabei lernt das Einbettungsnetzwerk auch für das CharLSTM nur eine 1:1 Abbildung von Bilddeskriptoren zu Attributvektoren und findet nicht die zugrundeliegende Abbildungsfunktion des Modells. Da jedoch die Transkriptionen von ähnlichen Wortabbildern meistens eine geringe Editierdistanz zueinander besitzen, haben sie sowohl einen ähnlichen Bilddeskriptor als auch eine geringe Distanz im Attributraum. Kommt nun im Training des Einbettungsnetzwerks ein Wortabbild w_1 vor, welches einen ähnlichen Bilddeskriptor wie das Wortabbild w_2 hat und w_2 eine nicht im Training vorkommende Transkription beinhaltet, dann wird dieses Wortabbild, aufgrund der Nähe zum n-gram Modell, für die meisten Wörter in die Nähe des Attributvektors von w_1 platziert. Somit ist es sehr häufig der Fall, dass w_2 in der Nähe des korrekten CharLSTM Vektors zu finden ist und damit bei einer QbS Anfrage mit dessen Transkription weit oben in der Ergebnisliste angezeigt wird. Jedoch muss beachtet werden, dass die Resultate, wie bereits in Kapitel 5.2 gezeigt, wenig mit einem semantischen Word Spotting Ergebnis zu tun haben und die hohen MAP Werte dadurch irreführend sind. Würde der Attributraum des CharLSTMs die korrekten semantischen Beziehungen der Wörter enthalten, dann würde es vergleichbare MAP Werte zum FastText Modell liefern.

Auch das im Gegensatz zum GW Datensatz schlechte abschneiden der Word2Vec und FastText Modelle auf der IAM Datenbank ist mithilfe der nachfolgenden Überlegungen gut nachvollziehbar. Dies liegt nämlich zum einen daran, dass der Anteil der nicht im Training vorkommenden Wörter bei der IAM Datenbank deutlich höher ist als beim GW Datensatz. Aber zum anderen liegt es vor allem an der hohen Intra-Klassen-Varianz der Wortabbilder in der IAM Datenbank. Bei dem GW Datensatz gibt es nur einen Schreiber, weshalb die Wortabbilder mit derselben Transkription sehr ähnlich zueinander sind. Dies wird auch anhand des Beispiels in der Abbildung 5.3.3 sowie dem hohen MAP Wert von ca 92 beim QbE Ansatz für das Triplet-CNN, siehe Tabelle 4.4.1, ersichtlich. Dieser hohe Wert bedeutet nämlich, dass die Wortabbilder mit derselben Transkription im Bilddeskriptor-Raum nah beieinander liegen und die Wortabbilder mit unterschiedlichen Transkriptionen weiter voneinander entfernt sind. Bei der IAM Datenbank ist es hingegen der Fall, dass die Wortabbilder von insgesamt 632 Personen

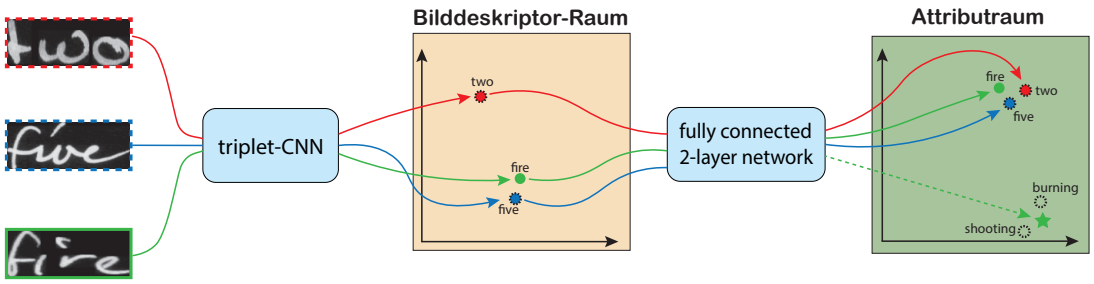

Abbildung 5.3.1: Diese Abbildung zeigt beispielhaft das Problem vom FastText Modell im Wilkinson Ansatz. Hierbei werden die Wortabbilder mit den Transkriptionen *two*, *five* und *fire* mithilfe dieses Ansatzes in den Attributraum des FastText Modells projiziert. Dabei sind die mit einem gestrichelten Rechteck umrandeten Wortabbilder bereits im Training des Einbettungsnetzwerks vorgekommen. Das Problem ist nun, dass das Wortabbild mit der Transkription *fire* nicht im Training vorkam und daher dessen korrekter Attributvektor für das Einbettungsnetzwerk nicht bekannt ist. Da es jedoch im Bilddeskriptor-Raum nah an dem Wort *five* liegt, wird es fälschlicherweise vom Einbettungsnetzwerk, auch im Attributraum nah an diesem platziert. Da es sich jedoch um einen semantischen Attributraum handelt, ist die Entfernung zur eigentlichen FastText-Repräsentation des Wortes, siehe grüner Stern, sehr groß. Aufgrund dieser großen Distanz landet das Wortabbild für die QbS Anfrage mit dem Wort *fire* ziemlich weit hinten in der Ergebnisliste. Mit diesem Beispiel wird verdeutlicht, dass das Einbettungsnetzwerk nicht die korrekte Abbildungsfunktion des FastText Modells lernt, sondern lediglich eine 1:1 Abbildung der im Training vorkommenden Wortabbilder zu deren Attributvektoren. Daher werden die Attributvektoren für nicht im Training gesehen Wörter, auf Grundlage ihrer Bilddeskriptoren und dem Abstand zu den im Training gesehenen Bilddeskriptoren bestimmt. Dieser Ansatz funktioniert zwar für einen traditionellen Word Spotting Ansatz, jedoch nicht für einen semantischen.

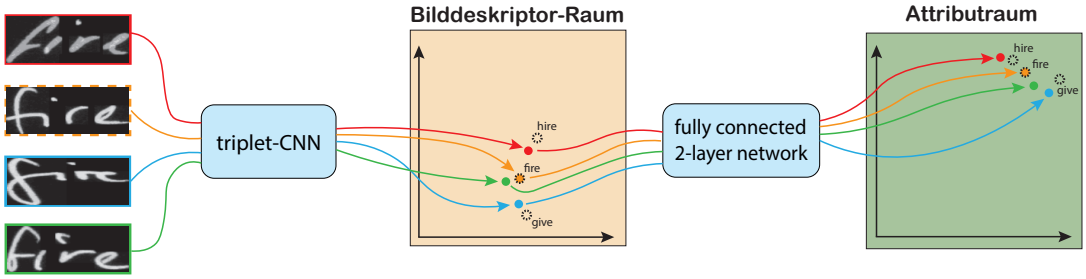

Abbildung 5.3.2: Anhand dieser Abbildung wird verdeutlicht, wieso das CharLSTM mit dem Wilkinson Ansatz so hohe MAP Werte für das semantische Word Spotting erzielen kann. Hierbei haben alle Anfragewörter die Transkription *fire*. Zudem hat das Einbettungsnetzwerk bereits die Wortabbilder mit den Transkriptionen *hire*, *fire* und *give* im Training gesehen. Da das rot markierte Wortabbild im Bilddeskriptor-Raum näher an dem Wortabbild von *hire* als zu *fire* ist, wird dieses, aufgrund der gelernten 1:1 Abbildungen des Einbettungsnetzwerks, auch im Attributraum nah an die Repräsentation des Wortes *hire* positioniert. Aus dem äquivalenten Grund wird auch das blau markierte Wortabbild im Attributraum nah an *give* platziert. Da die Wörter jedoch semantisch sehr unähnlich zueinander sind, würden diese in einem guten semantischen Attributraum weit voneinander entfernt liegen und deshalb bei einer QbS Anfrage mit dem Wort *fire* weit hinten in dessen Ergebnisliste auftauchen. Da der semantische Attributraum des CharLSTMs jedoch eher dem eines n-gram Modells entspricht, ist die Distanz zwischen diesen Wörtern im Raum sehr gering. Daher werden aufgrund dieses Attributraums, die aufgelisteten Wortabbilder bei einer QbS Anfrage für das Wort *fire* weit oben in der Ergebnisliste positioniert.

verfasst wurden und alle Wortabbilder eines Schreibers entweder in den Trainings- oder Testdaten liegen. Das führt dazu, dass eine Abbildung von Wortabbildern mit derselben Transkription zu einem sehr ähnlichen Bilddeskriptor schwieriger wird und diese bei einer praktischen Anwendung im Bilddeskriptor-Raum breiter gestreut liegen. Dies kann beispielhaft anhand der Abbildung 5.3.4 sowie dem geringen MAP Wert für den QbE Ansatz des Triplet-CNNs, siehe Tabelle 4.4.2, nachvollzogen werden. Aus diesen schlechten Bilddeskriptoren und den im Einbettungsnetzwerk gelernten 1:1 Abbildungen zwischen Bilddeskriptoren und Attribut-Repräsentationen resultiert, dass auch die Wortabbilder von im Training vorkommenden Wörtern nicht mehr auf ihre korrekten Attributvektoren abgebildet werden. Im GW Datensatz funktionierte das Prinzip noch gut für diese Wörter, da dort die Wortabbilder mit derselben Transkription sehr ähnlich zueinander sind.

Insgesamt kann also festgestellt werden, dass auch wenn das Word2Vec Modell selbst für das Scheitern verantwortlich ist, das System von Wilkinson für das semantische Word Spotting im Allgemeinen ungeeignet ist. Das liegt vor allem daran, dass ein Bilddeskriptor im Normalfall nur wenige bis keine Informationen über die Transkription eines Wortabbildes hat, sondern lediglich in der Lage ist, eine Trennung der Wortabbilder auf Grundlage der Transkriptionen zu ermöglichen. Damit entsteht zwar ein Bilddeskriptor-Raum, mit dem Wortabbilder bezüglich ihrer Transkription voneinander unterschieden werden können, jedoch kann anhand eines Bilddeskriptors nicht ermittelt werden, welche n-gramme in der Transkription des kodierten Wortabbildes vorhanden waren. Diese Information ist jedoch für die meisten semantischen Modelle aus der NLP fundamental und wird für die Bestimmung der korrekten Attribut-Repräsentationen von nicht im Training vorkommenden Wörtern benötigt. Aufgrund dieser nicht vorhandenen Muster in den Bilddeskriptoren, lernt das Einbettungsnetzwerk lediglich eine 1:1 Abbildung zwischen den Deskriptoren und deren Attribut-Repräsentationen und kann daher, im Normalfall, keinen korrekten Attributvektor für nicht im Training gesehene Wörter bestimmen. Zudem offenbart der Bilddeskriptor Ansatz große Probleme bei Wortabbildern mit einer hohen Intra-Klassen-Varianz. Auch das CharLSTM ist von diesen Problemen betroffen, erzielt aber trotzdem hohe MAP Werte für die beiden Datensätze, da die Beziehungen für die meisten Wörter im semantischen Attributraum sehr nah an denen im n-gram Modell sind.

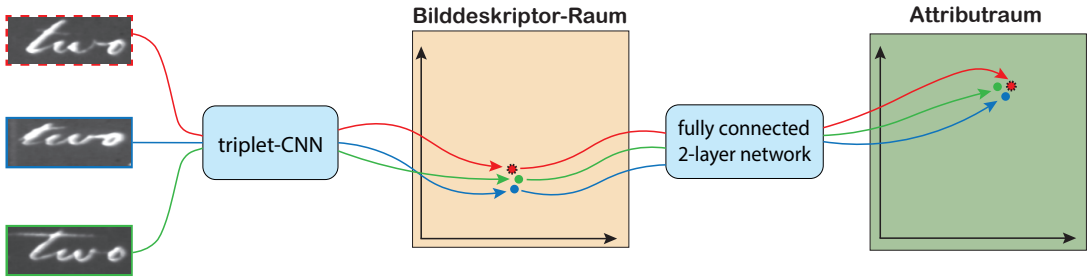

Abbildung 5.3.3: Mit dieser Abbildung soll verdeutlicht werden, dass die Wortabbilder mit derselben Transkription, im GW Datensatz sehr ähnlich zueinander sind und deshalb auch im Attributraum nah beieinander platziert werden. Dies wird beispielhaft anhand der drei Wortabbilder, aus dem GW Datensatz, mit der Transkription *two* visualisiert. Aufgrund ihrer visuellen Ähnlichkeit liegen diese Wortabbilder im Bilddeskriptor-Raum nah beieinander. Wegen dieser Nähe und des bereits im Training des Einbettungsnetzwerks vorgekommenen rot markierten Wortabbildes, werden auch für alle anderen Wortabbilder mit dieser Transkription, die korrekten Attributvektoren vorhergesagt.

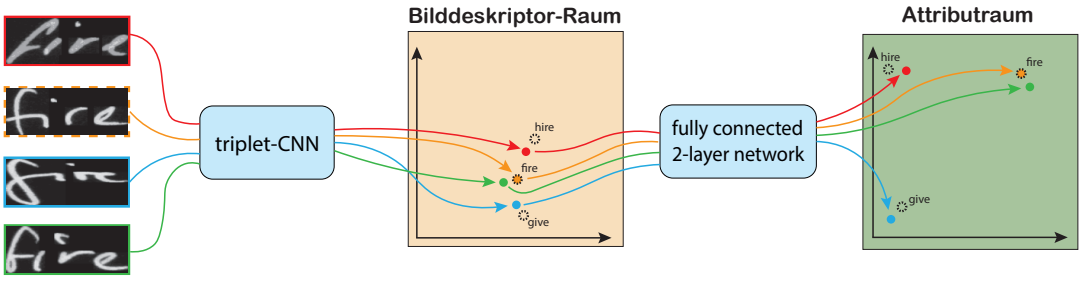

Abbildung 5.3.4: Diese Abbildung visualisiert, wieso die Modelle auf der IAM Datenbank, im Vergleich zum GW Datensatz, so schlechte Ergebnisse erzielen. Dabei haben alle Wortabbilder, mit *fire*, dieselbe Transkription. Zudem repräsentieren die gestrichelten Kreise im Bilddeskriptor- und Attributraum die Wortabbilder, die im Training des Einbettungsnetzwerks vorkamen. Aufgrund der hohen Intra-Klassen-Varianz von den Wortabbildern mit derselben Transkription in der IAM Datenbank, wurden die gezeigten Wortabbilder nicht auf einen gemeinsamen Bilddeskriptor abgebildet, sondern streuen stark im Bilddeskriptor-Raum. Das führt dazu, dass das rot markierte Wortabbild in dem Raum näher zu *hire* und das blau markierte näher zu *give* ist. Daher werden diese, aufgrund der schlechten Abbildungsfunktion des Einbettungsnetzwerks, auch im Attributraum nah beieinander platziert. Wegen der großen Distanz dieser Wörter im semantischen Attributraum, werden dieses beiden Wortabbilder bei einer QbS Anfrage des Wortes *fire* weit hinten in der Ergebnisliste aufgelistet. Damit verringert sich auch der AP Wert für im Training vorkommende Wörter drastisch und sorgt für die schlechten MAP Werte der FastText und Word2Vec Modelle, bei der Evaluation auf der IAM Datenbank.

6 BEWERTUNG VON ERGEBNISLISTEN BEIM SEMANTISCHEN WORD SPOTTING

Für die Bewertung von Ergebnislisten wird im Word Spotting traditionell auf die *Mean Average Precision* als Bewertungsmaß zurückgegriffen. Dieses Maß ist sehr gut geeignet für einen traditionellen Ansatz, da es bei diesem lediglich darauf ankommt, dass die Wortabbilder mit derselben Transkription wie die Anfrage, weit vorne in der Ergebnisliste stehen. Beim semantischen Word Spotting liegt hingegen der Fokus nicht nur auf die vordere Positionierung dieser Wortabbilder, sondern auch darauf, dass die zweiten Treffer in der Ergebnisliste absteigend nach der semantischen Ähnlichkeit sortiert sind. Leider werden die semantischen Ähnlichkeiten der einzelnen Treffer in der Liste, bei der Berechnung des MAP Wertes nicht mitberücksichtigt. Somit ist dieses Maß für die Bewertung von Rückgabelisten im semantischen Word Spotting ungeeignet. Dies kann beispielhaft anhand der Abbildung 6.0.1 nachvollzogen werden. Die fehlerhafte Verwendung des MAP Wertes zur Bewertung von semantischen Rückgabelisten, wird auch bei der Evaluation der semantischen Modelle im Kapitel 5.1 ersichtlich. Hierbei weißt das CharLSTM sowohl für den GW Datensatz als auch für die IAM Datenbank einen hohen MAP Wert auf, obwohl die zweiten Treffer meistens semantisch unähnlich bezüglich des Anfragewortes sind. Für Word2Vec und FastText sind die MAP Werte hingegen sehr gering, obwohl die zweiten Treffer in den meisten Rückgabelisten eine hohe semantische Ähnlichkeit bezüglich der Anfrage aufweisen. Aus diesem Grund wird in dem Kapitel 6.1 ein geeigneteres Maß für die Bewertung von Rückgabelisten beim semantischen Word Spotting vorgestellt, sowie im Kapitel 6.2 evaluiert.

6.1 BEWERTUNGSMASS

Wie bereits in der Einleitung des Kapitels beschrieben, wird für die Bewertung von semantischen Rückgabelisten im Word Spotting, eine geeignete Alternative zur *Mean Average Precision* benötigt. Dabei hat die MAP im Allgemeinen sehr nützliche Eigenschaften für die Bewertung von Rückgabelisten, ignoriert jedoch die semantischen Beziehungsinformationen zwischen dem Anfragewort und den Elementen aus der Liste. Aus diesem Grund baut das in diesem Kapitel vorgestellte Bewertungsmaß auf

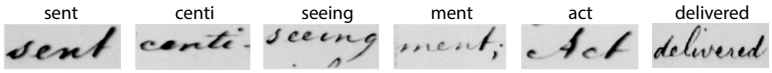

Abbildung 6.0.1: In dieser Abbildung wird beispielhaft die Rückgabeliste für das Anfragewort *sent* visualisiert. Die Werte unterhalb der Wortabbilder repräsentieren jeweils die semantische Ähnlichkeit zwischen der Anfrage und der Transkription des Wortabbildes. Hierbei wird ersichtlich, dass diese Liste einen MAP Wert von 1.0 erhält, obwohl das semantisch ähnliche Wort *delivered* am Ende der Liste vorkommt. Somit wird deutlich, dass lediglich die Positionierungen von den Wortabbildern mit derselben Transkription, bei der Berechnung des MAP Wertes berücksichtigt werden. Aus diesem Grund sollte die *Mean Average Precision* nicht für die Bewertung von semantischen Rückgabelisten verwendet werden.

der grundlegenden Idee der MAP auf und erweitert diese für die Bewertung von semantischen Ergebnislisten.

Für die Bewertung einer semantischen Liste, werden die semantischen Ähnlichkeiten zwischen dem Anfragewort und den jeweiligen Elementen dieser Liste benötigt. Dabei ist eine fundamentale Frage, wo die korrekten semantischen Beziehungsinformationen zwischen den Wörtern herkommen. Eine Idee, die auch in diesem Bewertungsmaß benutzt wird, ist die Verwendung eines semantischen Wortmodells, wie z.B. FastText, Word2Vec oder das CharLSTM, um die semantischen Ähnlichkeiten zwischen den Wörtern bestimmen zu können. Dafür ist es zunächst erforderlich, dass es ein Modell gibt, welches die semantischen Beziehungsinformationen zwischen den Wörtern aus der Datenbank korrekt repräsentiert. Dabei kann ein beliebiges Modell zur Bewertung der semantischen Ähnlichkeiten zweier Wörter verwendet werden. Für das verwendete Modell ist lediglich wichtig, dass es die korrekten semantischen Beziehungen zwischen allen relevanten Wörtern der Datenbank liefern kann.

Mithilfe der *Average Precision (AP)* und dem Modell zur Bestimmung von semantischen Ähnlichkeiten, wird im Folgendem ein für die semantische Bewertung geeignetes Maß hergeleitet. Die *Average Precision* ist dabei ein Maß zur Bewertung der Sortierung einer Rückgabeliste und kann für eine Liste mit der Länge n, wie bereits in Kapitel 4.3 beschrieben, formal mithilfe der Formel

$$AP = \frac{\sum_{k=1}^{n} p(k) \cdot r(k)}{t} \tag{6.1.1}$$

berechnet werden. Hierbei ist $r(k) \in \{0, 1\}$ eine Indikatorfunktion, mit

$$r(i) = \begin{cases} 1 & \text{wenn } \mathit{trans}(i) = a \\ 0 & \text{sonst} \end{cases} \tag{6.1.2}$$

Dabei ist $\mathit{trans}(i)$ die Transkription des i-ten Elements von der Ergebnisliste des Anfragewortes a. Zudem ist $t = \sum_{i=1}^{n} r(i)$ die Anzahl der Wortabbilder mit derselben Transkription wie das Anfragewort in der Liste. Mithilfe der Funktion

$$p(k) = \frac{TP}{TP + FP} = \frac{\sum_{i=1}^{k} r(i)}{k} \tag{6.1.3}$$

wird zudem der Precision Wert für die Rückgabeliste bis zur Position k bestimmt. Dieser Wert beschreibt das Verhältnis zwischen der Anzahl der relevanten und irrelevanten Elemente in der Liste. Dieses Verhältnis wird mithilfe der *True Positive (TP)* und *False Positive (FP)* Werte definiert. Hierbei geben TP und FP jeweils die Anzahl der Elemente bis zur Position k an, welche dieselbe bzw. eine unterschiedliche Transkription wie die Anfrage haben. Dabei kann der TP Wert aufgrund der Indikatorfunktion $r(\cdot)$, trivial mithilfe einer Summation der Werte $r(1)$ bis $r(k)$ berechnet werden. Da jedes Element der Liste exklusiv zu einer der beiden Klassen TP bzw. FP gehört, ist die Summe dieser Werte, gleich der Länge dieser Liste. Damit ergibt sich die in Formel 6.1.3 angegebene Berechnungsvorschrift für den Precision Wert, mit welcher die Sortierung der Rückgabeliste bis zur Position k bewertet werden kann. Dabei ist dieser Wert maximal, wenn lediglich relevante und minimal, wenn nur irrelevante Elemente in der Liste vorkommen. Generell gilt, dass umso mehr irrelevante Elemente vor einem relevanten Treffer stehen, der Precision Wert immer kleiner wird. Das liegt daran, dass mit jedem irrelevanten Treffer, der FP Wert um eins erhöht wird und gleichzeitig der TP Wert unverändert bleibt.

Diese Idee zur Bewertung der Sortierung einer Rückgabeliste wird im Folgenden auf die Bewertung von semantischen Rückgabelisten im Word Spotting angepasst. Dafür wird zunächst ersichtlich, dass es im Gegensatz zur AP keine 1 oder 0 Entscheidung darüber gibt, ob ein Wort für die Anfrage relevant ist oder nicht. Es ist eher so, dass jedes Wort der Liste eine semantische Beziehungsinformation bezüglich des Anfragewortes hat und damit alle Elemente der Liste bei der Bewertung relevant sind. Aus diesem Grund wird die Indikatorfunktion $r(i)$ durch die Funktion $\mathit{sim}(a, i)$ ersetzt. Diese Funktion bestimmt dabei, mithilfe eines semantischen Modells, die semantische Ähnlichkeit zwischen dem Anfragewort a und dem Wort an der Position i in der

Rückgabeliste. Mit dieser Änderung in der Precision Formel ergibt sich der sogenannte *similarity* oder kurz *s* Wert, der formal mithilfe der Formel

$$s(a,k) = \frac{\sum_{i=1}^{k} sim(a,i)}{k} \tag{6.1.4}$$

berechnet werden kann. Anstelle der Precision $p(k)$ wird mit dem $s(a,k)$ Wert die Sortierung der semantischen Ähnlichkeiten, bezüglich der Anfrage a, bis zur Position k in der Liste bestimmt. Der *s* Wert wird anschließend jeweils für die Positionen $k \in \{1,...,n\}$ berechnet und mit dem zugehörigen semantischen Ähnlichkeitswert des k-ten Elements der Rückgabeliste multipliziert. Abschließend werden diese Werte aufsummiert und ergeben den sogenannten *SemanticListValue(a,l)* oder kurz *SLV(a,l)*, welcher für eine Liste l mit der Größe n und einem Anfragewort a formal mithilfe der Formel

$$SLV(a,l) = \sum_{k=1}^{n} s(a,k) \cdot sim(a,k) = \sum_{k=1}^{n} \frac{\sum_{i=1}^{k} sim(a,i)}{k} \cdot sim(a,k) \tag{6.1.5}$$

bestimmt werden kann.

Leider existiert bei einer Bewertung mit dieser Formel ein Problem, das gut anhand des folgenden Beispiels ersichtlich wird. Gegeben sei die Rückgabeliste für das Anfragewort *captain* in Abbildung 6.1.1. Wie auch bei allen anderen Abbildungen von Rückgabelisten in diesem Kapitel, repräsentieren die Werte unterhalb der Wortabbilder jeweils die semantische Ähnlichkeit zwischen der Anfrage und der Transkription des Wortabbildes. Wie anhand der nachfolgenden Berechnung nachvollzogen werden kann, führt die Vertauschung der bezüglich des Anfragewortes semantisch unähnlichen Wörter *two* und *eight* in dieser Liste, zu einer veränderten Bewertung. Dabei wird mit der Berechnung 6.1.6, der SLV für die ursprüngliche Rückgabeliste und in Berechnung 6.1.7, der SLV für die Rückgabeliste mit den vertauschten Wortabbildern der Transkriptionen *two* und *eight* ermittelt.

$$\text{SLV} = \frac{1.0}{1} \cdot 1.0 + \frac{2.0}{2} \cdot 1.0 + \frac{2.10}{3} \cdot 0.1 + \frac{2.25}{4} \cdot 0.15 + \frac{2.87}{5} \cdot 0.62 + \frac{3.4}{6} \cdot 0.53 \approx 2.81 \tag{6.1.6}$$

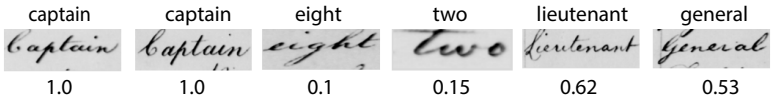

Abbildung 6.1.1: Visualisiert eine beispielhafte Rückgabeliste für das Anfragewort *captain*. Anhand dieser Rückgabeliste wird das Problem mit der Vertauschung von semantisch unähnlichen Wortabbildern in der Liste beschrieben.

$$\text{SLV} = \frac{1.0}{1} \cdot 1.0 + \frac{2.0}{2} \cdot 1.0 + \frac{2.15}{3} \cdot 0.15 + \frac{2.25}{4} \cdot 0.1 + \frac{2.87}{5} \cdot 0.62 + \frac{3.4}{6} \cdot 0.53 \approx 2.82 \quad (6.1.7)$$

Eine veränderte Bewertung bei einer Vertauschung von semantisch unähnlichen Wörtern in der Liste ist jedoch nicht erwünschenswert, da die beiden Wörter *two* und *eight* bezüglich des semantischen Modells zwar eine leicht unterschiedliche Ähnlichkeit zu der Anfrage haben, aber diese in der Praxis nicht existiert. Somit sollte berücksichtigt werden, dass die Vertauschung von semantisch unähnlichen Wörtern keinen Einfluss auf die Bewertung einer Rückgabeliste haben sollte. Dabei sollte zum Beispiel irrelevant sein, ob für das Anfragewort *captain*, die Wörter *eight* und *two* an der Stelle 20 und 21 in der Ergebnisliste stehen oder umgekehrt. Aus diesem Grund wird mithilfe der Formel

$$sim(a, i) \leftarrow \begin{cases} sim(a, i) & \text{wenn } sim(a, i) > \alpha \\ 0 & \text{sonst} \end{cases} \quad (6.1.8)$$

eine Anpassung der Ähnlichkeitswerte vorgenommen. Hierbei werden alle Ähnlichkeitswerte, die kleiner oder gleich einem Parameter α sind, auf den Wert 0 abgebildet. Somit existiert für Wörter, die bezüglich einer Anfrage semantisch unähnlich sind, keine Differenz mehr zwischen den Ähnlichkeitsbewertungen, wodurch die Vertauschung von diesen Wörtern in der Rückgabeliste keinen Einfluss mehr auf die Bewertung der Liste hat. Der α Wert ist ein Parameter, der vom Benutzer festgelegt werden muss. Für den George Washington Datensatz und dem vortrainierten FastText Modell ergibt sich für einen α Wert von 0.25 eine gute subjektive Grenze, mit der die semantisch unähnlichen Wörter einer Anfrage bestimmt werden können.

Ein weiterer wichtiger Bestandteil eines Bewertungsmaßes ist die Normalisierung der Werte auf das Intervall $[0, 1]$. Denn ohne diese Normalisierung wären die Ergebnisse nicht interpretierbar, wie anhand des SLVs für die Rückgabeliste aus Abbildung 6.1.1 nachvollzogen werden kann. Hierbei wurde für die Sortierung dieser Liste ein SLV von 2.81 berechnet. Jedoch ist anhand dieses Wertes nicht erkennbar, ob die Liste bezüglich der semantischen Ergebnisse gut oder schlecht ist. Bei der Bewertung sollte aufgrund der Interpretierbarkeit der Ergebnisse, für die bestmögliche semantische Rückgabeliste eine 1 zurückgegeben werden und für die schlechteste ein Wert der gegen 0 geht. Bei der AP wird die Normalisierung durch die Division der Bewertung einer Liste mit dem Wert t erreicht. Dieser Wert repräsentiert dabei die Bewertung der für dieses Bewertungsmaß bestmöglichen Sortierung dieser Liste. Dabei gilt für die AP, dass die bestmögliche Sortierung einer Liste alle Wortabbilder mit derselben

Transkription wie die Anfrage ganz vorne in der Liste platziert und die Wortabbilder mit den unterschiedlichen Transkriptionen dahinter aufgelistet werden. Dadurch ergibt sich für diese Ergebnisliste mit der Länge n und insgesamt w Wortabbilder mit derselben Transkription wie die Anfrage, eine Bewertung von

$$\sum_{k=1}^{n} \frac{\sum_{i=1}^{k} r(i)}{k} \cdot r(k) = \frac{1}{1} \cdot 1 + \frac{2}{2} \cdot 1 + ... + \frac{w}{w} \cdot 1 + \frac{w}{w+1} \cdot 0 + ... + \frac{w}{n} \cdot 0 = \sum_{k=1}^{n} r(k) = t = w. \tag{6.1.9}$$

Die Normalisierung des SLVs für eine Liste ist jedoch nicht so einfach mit $t = \sum_{k=1}^{n} sim(\cdot, k)$ möglich. Dafür muss zunächst die Bewertung für die semantisch bestmögliche Sortierung dieser Liste berechnet werden. Dabei ergibt sich die bestmögliche semantische Sortierung einer Liste durch die absteigende Sortierung der enthaltenen Wortabbilder nach ihrer semantischen Ähnlichkeit zur Anfrage. Anschließend kann für diese Liste der SLV Wert berechnet und mit der Bewertung der Rückgabeliste in Beziehung gesetzt werden. Aus dieser Überlegung folgt die endgültige Definition des *Semantic Values (SVs)*. Dieser Wert wird mithilfe der Formel

$$SV(a, l) = \frac{SLV(a, l)}{SLV(a, best)} \tag{6.1.10}$$

berechnet. Hierbei wird die semantische Bewertung der Rückgabeliste l für das Anfragewort a, mit der Bewertung für die semantisch bestmögliche Sortierung dieser Liste $(best)$, in Beziehung gesetzt. Die SVs werden für alle Anfragewörter einer Testmenge berechnet und abschließend gemittelt. Dies ergibt den *Mean Semantic Value (MSV)*.

Im Folgenden werden die Berechnungen für die Ermittlung des *Semantic Values*, anhand eines anschaulichen Beispiels demonstriert. Außerdem wird auf Grundlage mehrerer Rückgabelisten die Anwendbarkeit dieses Bewertungsmaßes getestet und diskutiert. In diesem Beispiel existiert eine Datenbank mit sechs Wortabbildern, welche die Transkriptionen *captain*, *captain*, *lieutenant*, *general*, *two* und *eight* haben. Als Anfragewort dient das Wort *captain*. Für die Ermittlung der semantischen Beziehungsinformationen wird das vortrainierte FastText Modell verwendet. Die semantischen Ähnlichkeitswerte zwischen dem Anfragewort und den Transkriptionen der Datenbank befinden sich in der Tabelle 6.1.1.

Für die Bewertung einer Rückgabeliste mit dem *Semantic Value* ist es zunächst erforderlich, dass der *SLV* für die semantisch bestmögliche Sortierung der Rückgabeliste bestimmt wird. Dafür werden die Elemente der Datenbank bezüglich der Ähnlichkeitswerte zum Anfragewort *captain* absteigend sortiert und erzeugen damit die in

Wortabbild	Ähnlichkeit zu Anfragewort bzgl. FastText Modell
captain	1.0
captain	1.0
lieutenant	0.62
general	0.53
two	0.15
eight	0.10

Tabelle 6.1.1: Die semantischen Ähnlichkeitswerte zwischen dem Anfragewort und den enthaltenen Transkriptionen in der Datenbank. Die Ähnlichkeitswerte wurden mit dem vortrainierten FastText Modell aus dem Kapitel 5.1 berechnet.

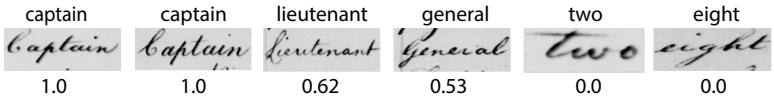

Abbildung 6.1.2: Die Abbildung der semantisch bestmöglichen Rückgabeliste für das beschriebene Szenario. Dabei wurden die Bewertungen mit $(\alpha = 0.25)$ angepasst.

der Abbildung 6.1.2 visualisierte Rückgabeliste. Auf Grundlage dieser sortierten Liste, wird der *SLV* mit

$$\text{SLV} = \frac{1.0}{1} \cdot 1.0 + \frac{2.0}{2} \cdot 1.0 + \frac{2.62}{3} \cdot 0.62 + \frac{3.15}{4} \cdot 0.53 + \frac{3.15}{5} \cdot 0.0 + \frac{3.15}{6} \cdot 0.0 \approx 2.96 \quad (6.1.11)$$

ermittelt. Mithilfe dieses Wertes werden die *Semantic List Values*, von den im Folgenden vorgestellten und beispielhaften Rückgabelisten normalisiert.

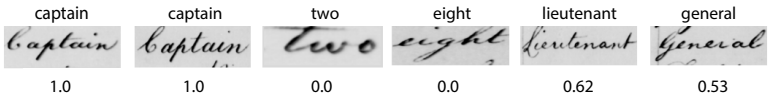

Abbildung 6.1.3: Visualisierung einer beispielhaften Rückgabeliste für das Anfragewort *captain*, um das Verhalten des *Semantic Values* bei einer fehlerhaften Sortierung der semantischen Ergebnisse zu untersuchen.

Mit der in Abbildung 6.1.3 visualisierten Ergebnisliste wird analysiert, wie sich die Bewertung bezüglich der bestmöglichen Rückgabeliste ändert, wenn die beiden semantisch unähnlichen Wörter *two* und *eight*, vor den semantisch ähnlichen Wörtern *lieutenant* und *general* in der Liste stehen. Dafür wird der SV dieser Liste mit

$$\text{SLV} = \frac{1.0}{1} \cdot 1.0 + \frac{2.0}{2} \cdot 1.0 + \frac{2.0}{3} \cdot 0.0 + \frac{2.0}{4} \cdot 0.0 + \frac{2.62}{5} \cdot 0.62 + \frac{3.15}{6} \cdot 0.53 \approx 2.6 \quad (6.1.12)$$

$$\text{SV} = \frac{2.6}{2.96} \approx 0.88$$

berechnet. Hierbei wird ersichtlich, dass die fehlerhafte Platzierung von semantisch unähnlichen Wörtern, vor semantisch ähnlichen Wörtern in der Liste, stark sanktioniert wird. Somit sinkt die Bewertung auf einen Wert von 0.88. Diese Bewertung ist angemessen, da die eigentlichen Treffer ganz vorne in der Liste stehen, aber die semantisch ähnlichen Treffer hinter semantisch irrelevanten Wortabbildern aufgelistet sind.

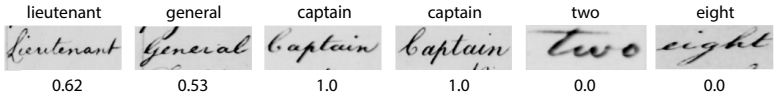

Abbildung 6.1.4: Visualisierung einer beispielhaften Rückgabeliste für das Anfragewort *captain*, um das Verhalten des *Semantic Values* bei einer fehlerhaften Platzierung von Wortabbildern mit derselben Transkription wie die Anfrage zu untersuchen.

Das Bewertungsmaß sollte zudem die fehlerhafte Platzierung von Wortabbildern mit derselben Transkription wie die Anfrage negativ bewerten. Dies wird anhand der Rückgabeliste aus Abbildung 6.1.4 untersucht. Hierbei stehen die beiden Wortabbilder mit der Transkription *captain*, hinter den semantisch ähnlichen Wörtern *lieutenant* und *general*. Der SV Wert wird für diese Rückgabeliste wie folgt bestimmt:

$$\begin{aligned} \text{SLV} &= \frac{0.62}{1} \cdot 0.62 + \frac{1.15}{2} \cdot 0.53 + \frac{2.15}{3} \cdot 1.0 + \frac{3.15}{4} \cdot 1.0 + \frac{3.15}{5} \cdot 0.0 + \frac{3.15}{6} \cdot 0.0 \approx 2.19 \\ \\ \text{SV} &= \frac{2.19}{2.96} \approx 0.74 \end{aligned} \tag{6.1.13}$$

Das Bewertungsmaß bestraft korrekterweise die falsche Platzierung der Wörter *captain* und *captain*. Ein Wert von 0.78 erscheint aufgrund der hohen semantischen Platzierung von den Wörtern *lieutenant* und *general* auf den ersten Blick zu gering. Jedoch wird die Korrektheit dieses Wertes bei der Betrachtung von dem Ziel des semantischen Word Spottings ersichtlich. Hierbei ist es nämlich so, dass nach den eigentlichen Treffern, auch semantisch ähnliche Ergebnisse in der Rückgabeliste angezeigt werden sollen. Somit liegt der Fokus zunächst auf der korrekten Platzierung der eigentlichen Treffer und danach wird erst auf die semantischen Ergebnisse eingegangen.

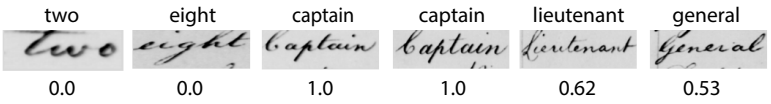

Abbildung 6.1.5: Visualisierung einer beispielhaften Rückgabeliste für das Anfragewort *captain*, um das Verhalten des *Semantic Values* bei einer fehlerhaften Platzierung von semantisch unähnlichen Wortabbildern vor den eigentlichen Treffern zu untersuchen.

Die Bewertung für die Rückgabeliste in Abbildung 6.1.5 sollte noch geringer werden, da die beiden unähnlichen Wörter *two* und *eight* nun ganz vorne in der Liste stehen. Die eigentlichen Treffer befinden sich dabei nur an den Positionen drei und vier. Mithilfe der folgenden Berechnungen:

$$\begin{aligned} \text{SLV} &= \frac{0}{1} \cdot 0.0 + \frac{0}{2} \cdot 0.0 + \frac{1.0}{3} \cdot 1.0 + \frac{2.0}{4} \cdot 1.0 + \frac{2.62}{5} \cdot 0.62 + \frac{3.15}{6} \cdot 0.53 \approx 1.44 \\ \\ \text{SV} &= \frac{1.44}{2.96} \approx 0.49 \end{aligned} \tag{6.1.14}$$

wird der SV für die Rückgabeliste bestimmt. Hierbei wird deutlich, dass die fehlerhafte Positionierung von semantisch unähnlichen Wortabbildern, vor den eigentlichen sowie den semantisch ähnlichen Treffern, korrekterweise zu einer starken Bestrafung seitens des Bewertungsmaßes führt.

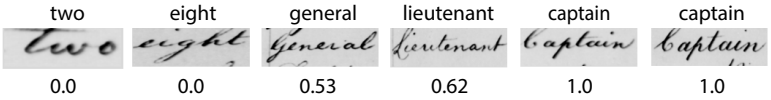

Abbildung 6.1.6: Visualisierung einer beispielhaften Rückgabeliste für das Anfragewort *captain*, um das Verhalten des *Semantic Values* bei der schlechtesten Rückgabeliste für dieses Szenario zu untersuchen.

In der Abbildung 6.1.6 ist die schlechteste Rückgabeliste für dieses Szenario abgebildet. Das Bewertungsmaß sollte für diese Liste den geringsten Wert von allen bislang getesteten Rückgabelisten liefern. Die Bewertung der Liste wird mit dem SV, wie folgt bestimmt:

$$\begin{aligned} SLV =& \frac{0}{1} \cdot 0.0 + \frac{0}{2} \cdot 0.0 + \frac{0.53}{3} \cdot 0.53 + \frac{1.15}{4} \cdot 0.62 + \frac{2.15}{5} \cdot 1.0 + \frac{3.15}{6} \cdot 1.0 \approx 1.23 \\ SV =& \frac{1.23}{2.96} \approx 0.42 \end{aligned} \tag{6.1.15}$$

Diese Liste führt korrekterweise zu der bislang schlechtesten Bewertung. In diesem Spielbeispiel hat die Liste zwar eine Bewertung, welche größer als 0 ist, jedoch ist dies in der Praxis normalerweise nicht der Fall. Das liegt zum einen an der deutlich größeren Datenbank und zum anderem an der normalerweise deutlich größeren Anzahl an semantisch unähnlichen Wörtern in dieser. Damit geht der SV für die schlechteste Liste und einer praxisnahen Datenbank gegen 0. Es wurde zudem überlegt eine Normierung der Bewertungen mit

$$\mathrm{normalized\_SV}(a, \mathrm{return}) = \frac{SV(a, \mathrm{return}) - SV(a, worst)}{SV(a, best) - SV(a, worst)} \tag{6.1.16}$$

vorzunehmen. Jedoch ist diese Form der Normalisierung, wie gerade erwähnt, in der Praxis nicht relevant und steigert zudem den Berechnungsaufwand. Dabei ist *return* die zu bewertende Rückgabeliste, *worst* die schlechteste sowie *best* die beste Rückgabeliste für eine vorgegebene Datenbank und Anfrage. Zudem ist $\mathrm{normalized\_SV}(a, \mathrm{return})$ die normalisierte Bewertung von der Rückgabeliste return, sodass die schlechteste Liste einen Wert von 0 und die beste einen Wert von 1 erhält.

Abbildung 6.1.7: Visualisierung einer beispielhaften Rückgabeliste für das Anfragewort *captain*, um zu überprüfen, ob die Vertauschung von semantisch unähnlichen Wörtern in der Liste, einen Einfluss auf die Bewertung hat.

Mit der Rückgabeliste aus Abbildung 6.1.7 soll überprüft werden, ob die Vertauschung von semantisch unähnlichen Wörtern in der Liste, einen Einfluss auf die Bewertung hat. Dafür wurden im Vergleich zur Rückgabeliste 6.1.3 die Wortabbilder mit den Transkriptionen *two* und *eight* vertauscht. Die Bewertung dieser Liste mit dem SV wird mit

$$\begin{aligned} SLV &= \frac{1.0}{1} \cdot 1.0 + \frac{2.0}{2} \cdot 1.0 + \frac{2.0}{3} \cdot 0.0 + \frac{2.0}{4} \cdot 0.0 + \frac{2.62}{5} \cdot 0.62 + \frac{3.15}{6} \cdot 0.53 \approx 2.6 \\ SV &= \frac{2.6}{2.96} \approx 0.88 \end{aligned} \tag{6.1.17}$$

bestimmt. Hierbei wird ersichtlich, dass sich die Bewertung dieser Rückgabeliste im Vergleich zur Bewertung der Liste in Abbildung 6.1.3 nicht geändert hat. Damit konnte beispielhaft gezeigt werden, dass die Vertauschung von semantisch unähnlichen Wörtern in der Liste keinen Einfluss auf die Bewertung hat.

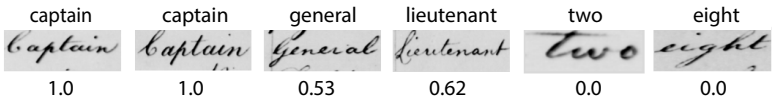

Abbildung 6.1.8: Visualisierung einer beispielhaften Rückgabeliste für das Anfragewort *captain*, um zu überprüfen, ob die Vertauschung von semantisch ähnlichen Wörtern mit unterschiedlichen Ähnlichkeiten in der Liste, einen Einfluss auf die Bewertung hat.

Die Vertauschung von semantisch ähnlichen Wortabbildern mit einer unterschiedlichen Ähnlichkeit, sollte jedoch einen Einfluss auf die Bewertung der Liste haben. Aus diesem Grund wird mit der Rückgabeliste aus Abbildung 6.1.8 überprüft, ob sich die Bewertung im Vergleich zur Rückgabeliste aus 6.1.2 ändert. Hierbei wurden die bezüglich der Anfrage semantisch ähnlichen Wortabbilder *lieutenant* und *general*,

in der bestmöglichen Rückgabeliste miteinander vertauscht. Die Berechnung für die Bewertung dieser Liste wird wie folgt ermittelt:

$$\begin{aligned} \text{SLV} &= \frac{1.0}{1} \cdot 1.0 + \frac{2.0}{2} \cdot 1.0 + \frac{2.53}{3} \cdot 0.53 + \frac{3.15}{4} \cdot 0.62 + \frac{3.15}{5} \cdot 0.0 + \frac{3.15}{6} \cdot 0.0 \approx 2.94 \\ \text{SV} &= \frac{2.94}{2.96} \approx 0.99 \end{aligned} \tag{6.1.18}$$

Dabei kann festgestellt werden, dass sich der Wert, wenn auch nur marginal, verschlechtert hat und somit die Vertauschung von semantisch ähnlichen Wortabbildern, mit unterschiedlichen Ähnlichkeiten, in der Liste, von dem Bewertungsmaß berücksichtigt wird.

6.2 EVALUATION DES BEWERTUNGSMASSES

In diesem Kapitel werden mit dem neu definierten Bewertungsmaß, die semantischen Ergebnisse für den QbS Ansatz des Wilkinson Systems bewertet und mit den korrespondierenden MAP Werten verglichen. Dabei wird der Vergleich sowohl für das CharLSTM als auch für das vortrainierte FastText und das spezialisierte Word2Vec Modell durchgeführt. Die Evaluation findet auf dem GW Datensatz und der IAM Datenbank statt. Für die beiden Datensätze wird der α Parameter des *Semantic Values* auf 0.25 festgelegt. Außerdem wird das vortrainierte FastText Modell zur Festlegung der semantischen Ground-Truth Beziehungsinformationen verwendet. Die MAP und MSV Bewertungen für die unterschiedlichen semantischen Modelle, befinden sich bezüglich des GW Datensatzes, in der Tabelle 6.2.1. Anhand der Ergebnisse aus dieser Tabelle, können mehrere Erkenntnisse für das neu definierte Bewertungsmaß gewonnen werden. Dabei fällt zunächst auf, dass der MSV Wert für das CharLSTM bezüglich der MAP Bewertung stark gesunken ist, aber immer noch im Vergleich zu den Word2Vec und FastText Ergebnissen hoch ausfällt. Das liegt vor allem daran, dass für viele Anfragewörter, die eigentlichen Treffer weit vorne in den Rückgabelisten stehen und es für diese Anfragen keine semantisch ähnlichen Wörter in der Datenbank gibt. Somit sind die Rückgabelisten für diese Wörter perfekt sortiert und ergeben eine hohe Bewertung bezüglich des *Semantic Values*. Befinden sich zu einem Anfragewort jedoch semantisch ähnliche Wörter in der Datenbank, ist der SV deutlich geringer, da diese weiter hinten in den Ergebnislisten platziert werden. Auch für das Word2Vec Modell sinken die MSVs. Dieses Verhalten ist dadurch erklärbar, da dieses Modell

Modell	*Trainingsdaten*	*MAP*	*MSV*
CharLSTM (semantisch)	**GWW**	81.93	55.05
Word2Vec	**GWW**	53.08	36.59
	vortrainiert	59.59	35.72
	vortrainiert + GWW	53.36	31.30
FastText	**GWW**	66.44	33.57
	vortrainiert	58.73	70.66
	vortrainiert + GWW	43.80	34.05

Tabelle 6.2.1: Die Bewertungsergebnisse für das Word Spotting System von Wilkinson mit den unterschiedlichen semantischen Modelle auf dem GW Datensatz. Dabei werden die MAP und MSV Ergebnisse der semantischen Modelle gegenübergestellt.

zum einen hohe *Semantic Values* für Anfragen liefert, die semantisch ähnliche Wörter in der Datenbank haben und wobei diese auch im Training des Einbettungsnetzwerks vorkamen. Und zum anderen erzeugt es hohe SVs für Anfragen, die keine oder nur wenige semantisch ähnliche Wörter in der Datenbank haben und wo das Anfragewort in dem Training des Einbettungsnetzwerks vorkam. Für alle anderen Szenarien gehen, aufgrund der schlechten Abbildungen von nicht im Training vorkommenden Wörtern, die SVs gegen 0. Die Ergebnisse für das FastText Modell sind Äquivalent zum Word2Vec Modell, außer dass der MSV für das vortrainierte Modell sehr hoch ist. Dies ist jedoch kein fairer Vergleichswert, da die semantischen Beziehungsinformationen von diesem Modell festgelegt werden. Damit wird auch ersichtlich, dass die Wahl des semantischen Modells zur Festlegung der Beziehungsinformationen zwischen Wörtern, einen hohen Einfluss auf die Bewertungen hat. Daher sollte für das verwendete Modell sichergestellt werden, dass es die Beziehungsinformationen für die anwendungsrelevanten Wörter korrekt repräsentiert. Die große Differenz zwischen den MAP Werten des nur auf dem GWW trainierten und dem spezialisierten FastText Modell und den gleichzeitig identischen MSVs dieser beiden Modelle, ist dadurch erklärbar, da die Bewertungen von den semantischen Ähnlichkeiten zwischen Wörtern, für die Modelle verschieden sind. Dabei bewertet zum Beispiel das vortrainierte Modell die semantische Ähnlichkeit der Wörter *by* und *as* mit einer 0.4 und das nur auf dem GWW trainierte Modell mit einer Bewertung nahe 0. Dadurch sortiert das auf dem GWW trainierte Modell, die Wortabbilder mit der Transkription *as* für das Anfragewort *by*, weit hinten in der Ergebnisliste ein und führt damit zu einer Reduzierung des SVs für diese Liste. Das spezialisierte Modell hat hingegen ähnlichere semantische

Bewertungen zum vortrainierten FastText Modell und sortiert die bezüglich des zu bewertenden Systems ähnlichen Wortabbilder weiter vorne in der Liste ein. Dies führt zu einer höheren semantischen Bewertung der Liste, obwohl die eigentlichen Treffer weiterer hinten zu finden sind. Zudem wirkt sich die Nähe von semantischen Modellen zum n-gram Modell für manche Anfragewörter negativ auf den SV aus.

Die MAP und MSV Bewertungen für die unterschiedlichen semantischen Modelle, befinden sich bezüglich der IAM Datenbank, in der Tabelle 6.2.2. Hierbei sind ähnliche Muster wie beim GW Datensatz zu erkennen. Jedoch sind die MSVs deutlich geringer. Dies liegt vor allem daran, dass die IAM Datenbank deutlich mehr semantische Wörter bezüglich einer Anfrage enthält und es daher nicht so häufig den Fall gibt, dass es zu einem Anfragewort keine semantischen Wörter in der Datenbank gibt. Dieses Szenario kam vergleichsweise häufig im GW Datensatz vor und führte dadurch zu einem hohen SV für diese Anfragewörter. Zudem ist es aufgrund der schlechten Abbildungen von nicht im Training vorkommenden Wörtern so, dass zwar für im Training vorkommende Anfragewörter, die eigentlichen Treffer weit vorne in deren Ergebnislisten stehen, aber viele ihrer semantisch ähnlichen Wörter nicht vorne einsortiert werden. Außerdem kommt aufgrund der hohen Intra-Klassen-Varianz erschwerend hinzu, dass auch für im Training vorkommende Wörter, die korrekte Attribut-Repräsentation nicht ermittelt werden kann. Eine weitere Auffälligkeit der Ergebnisse ist, dass alle Word2Vec Modelle ähnlich geringe MSVs erhalten, obwohl der MAP Wert vom vortrainierten Modell deutlich größer ist. Das liegt vermutlich daran, dass die Wortabbilder von semantisch ähnlichen Wörtern über die Rückgabeliste verteilt sind. Das bewirkt zum einen, dass Modelle mit geringen MAP Werten, zwar die eigentlichen Treffer am Ende der Liste haben, aber durch die Platzierung der semantisch ähnlichen Treffer in der Mitte der Ergebnisliste, nicht der schlechtesten Sortierung entsprechen und damit ein MSV um den Wert 15 angemessen erscheint. Zum anderen ist es für Modelle mit höheren MAP Werten der Fall, dass die mittig platzierten semantischen Wortabbilder in der Ergebnisliste, zu einem geringeren SV führen.

Modell	*Trainingsdaten*	*MAP*	*MSV*
CharLSTM (semantisch)	**Penntree**	84.45	25.54
Word2Vec	**Penntree**	0.33	14.96
	vortrainiert	19.61	15.34
	vortrainiert + Penntree	0.08	10.72
FastText	**Penntree**	0.05	18.30
	vortrainiert	20.37	31.71
	vortrainiert + Penntree	0.44	16.51

Tabelle 6.2.2: Die Bewertungsergebnisse für das Word Spotting System von Wilkinson mit den unterschiedlichen semantischen Modelle auf der IAM Datenbank. Dabei werden die MAP und MSV Ergebnisse der semantischen Modelle gegenübergestellt.

7 ZUSAMMENFASSUNG

In dieser Arbeit wurde das semantische Word Spotting System von Wilkinson et al. aus [WB16] untersucht. Dabei wurde zunächst das beschriebene System nachgebaut sowie mit dem angegebenen Evaluationsprotokoll evaluiert. Auf Grundlage dieser Nachimplementierung konnten die vorgestellten MAP Werte aus der Veröffentlichung bestätigt werden. Dabei wurden zum Teil deutlich höhere MAP Werte bei den Evaluationen der CharLSTM-Repräsentationen erreicht. Bei den restlichen Attribut-Repräsentationen waren die Evaluationsergebnisse im Vergleich zu den veröffentlichten Werten geringfügig schlechter, was voraussichtlich auf die zufällig generierten Trainingsdaten zurückzuführen ist. Außerdem wurde in dieser Arbeit evaluiert, ob durch die Ersetzung des CharLSTMs mit den FastText und Word2Vec Modellen im Wilkinson Ansatz, eine Verbesserung der semantischen Ergebnislisten erreicht werden kann. Hierbei wurde jedoch ersichtlich, dass diese bewährten Ansätze aus den natürlichsprachlichen Systemen auch für unterschiedlich trainierte Modelle zu keiner Verbesserung bezüglich der MAP Werte führen. Anhand der Wortmodellanalysen für diese Modelle konnte jedoch beispielhaft gezeigt werden, dass die semantischen Beziehungen in den vom vortrainierten Word2Vec und FastText Modell gelernten Attributräumen deutlich besser sind als die des CharLSTMs. Zudem wurde in der Analyse ersichtlich, dass der semantische Attributraum des CharLSTMs starke Ähnlichkeiten zu dem Raum des n-gram Modells hat und nur deshalb so gute MAP Werte bei den Evaluationen erzielen konnte. Dabei weißt dieses Modell nur für sehr wenige Wörter gute semantische Beziehungen auf. In der Veröffentlichung wurden genau zu diesen Wörtern die Ergebnislisten angegeben, wodurch zusammen mit dem hohen MAP Wert, ein fehlerhafter Eindruck von der Qualität des Ansatzes entsteht. Bei der Evaluation der Word2Vec Modelle stellte sich heraus, dass dieses Verfahren für die Verwendung im Wilkinson Ansatz ungeeignet ist. Dies liegt an den nicht lernbaren Strukturen der Word2Vec Attributvektoren, weshalb die Attributvektoren von nicht im Training vorkommenden Wörtern falsch vorhergesagt werden. Das FastText Modell ist hingegen für das Wilkinson System besser geeignet, da es eine lernbare Abbildungsfunktion gibt, mit der auch die Attributvektoren von nicht im Training vorkommenden Wörtern korrekt vorhergesagt werden können. Jedoch scheitert die Verwendung des FastText Modells im Wilkinson System an dem Abbildungsverfahren von Wortabbildern zu Attributvektoren. Dabei wird zunächst für das Wortabbild ein Bilddeskriptor berechnet und auf Grundlage dieses

Deskriptors dessen Attribut-Repräsentation vorhergesagt. Für die korrekte Vorhersage des FastText Vektors werden die strukturellen Informationen aus der Transkription des Wortabbildes benötigt. Dabei ist es jedoch sehr unwahrscheinlich, dass die Bilddeskriptoren die strukturellen Informationen aus einem Wortabbild extrahieren können, da sie lediglich so trainiert werden, dass Wortabbilder mit unterschiedlichen Transkriptionen voneinander unterscheidbar sind. Damit ist es für das neuronale Netzwerk aus dem Wilkinson Ansatz schwierig bis unmöglich, aus den strukturell nicht informativen Bilddeskriptoren, die korrekte Abbildungsfunktion eines semantischen Modells zu lernen. Aus diesem Grund kommt es zur fehlerhaften Vorhersage von Attributvektoren für die Wortabbilder mit nicht im Training vorkommenden Transkriptionen. Da die strukturellen Informationen eines Wortes jedoch für alle bekannten semantischen Wortmodelle relevant sind, ist das System von Wilkinson für ein semantisches Word Spotting im Allgemeinen ungeeignet. Eine weitere Aufgabe dieser Arbeit war die Ermittlung eines geeigneteren Maßes zur Bewertung von Ergebnislisten beim semantischen Word Spotting, da Aufgrund der nicht Berücksichtigung von semantischen Informationen bei der MAP ersichtlich wird, dass dieses Maß zur Bewertung dieser Listen ungeeignet ist. Daher wurde, mit dem *Mean Semantic Value (MSV)* eine Erweiterung der MAP entwickelt, bei der auch die semantischen Informationen mit in die Bewertung der Liste einfließen. Bei der Bewertung der semantischen Rückgabelisten mit dem neuen Bewertungsmaß fällt für die in dieser Arbeit vorkommenden semantischen Modelle auf, dass die MSV Werte für das CharLSTM deutlich geringer als die zugehörigen MAP Werte sind. Dies ist jedoch korrekt, da aufgrund der Nähe zum n-gram Modell, die semantisch ähnlichen Wörter bezüglich der Anfrage weit hinten in deren Ergebnislisten stehen. Zum anderen fällt auf, dass aufgrund der nicht lernbaren beziehungsweise nicht korrekt gelernten Abbildungsfunktionen der Word2Vec und FastText Modelle, fehlerhafte Attributvektoren für nicht im Training vorkommende Wörter bestimmt werden. Dies führt zu einer falschen Positionierung von diesen Wörtern in der Ergebnisliste, wodurch sich im Allgemeinen für die MSV Bewertung der Word2Vec und FastText Modelle auf dem GW Datensatz geringere und für die IAM Datenbank höhere Werte bezüglich der korrespondierenden MAP Werte ergeben. In weiterführenden Arbeiten zu diesem Thema könnte evaluiert werden, ob mit dem Ende-zu-Ende Ansatz vom Attribute-CNN (siehe Kapitel 3.1) und dem FastText Modell bessere Ergebnisse für das semantische Word Spotting erzielt werden können. Außerdem ist eine durchaus vielversprechende Idee, anstelle eines Bilddeskriptors die vorhergesagte PHOC-Repräsentation des Wortabbildes zu verwenden, da diese die benötigten n-gram Informationen aus dem Wortabbild enthält. Damit wäre es für das Einbettungsnetzwerk letztlich deutlich leichter, die korrekte Abbildungsfunktion zu finden.

ABBILDUNGSVERZEICHNIS

LITERATURVERZEICHNIS

[AGFV14] Almazán, Jon ; Gordo, Albert ; Fornes, Alicia ; Valveny, Ernest: Word Spotting and Recognition with Embedded Attributes. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 36 (2014), 12, S. 2552–2566. http://dx.doi.org/10.1109/TPAMI.2014.2339814. – DOI 10.1109/TPAMI.2014.2339814

[BDK14] Baroni, Marco ; Dinu, Georgiana ; Kruszewski, Germán: Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014, S. 238–247

[BGJM16] Bojanowski, Piotr ; Grave, Edouard ; Joulin, Armand ; Mikolov, Tomas: Enriching Word Vectors with Subword Information. In: *arXiv preprint arXiv:1607.04606* (2016)

[BJTM16] Balntas, Vassileios ; Johns, Edward ; Tang, Lilian ; Mikolajczyk, Krystian: PN-Net: Conjoined Triple Deep Network for Learning Local Image Descriptors. In: *CoRR* abs/1601.05030 (2016)

[DDF+90] Deerwester, Scott ; Dumais, Susan T. ; Furnas, George W. ; Landauer, Thomas K. ; Harshman, Richard: Indexing by latent semantic analysis. In: *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE* 41 (1990), Nr. 6, S. 391–407

[DHS00] Duda, Richard O. ; Hart, Peter E. ; Stork, David G.: *Pattern Classification (2Nd Edition)*. New York, NY, USA : Wiley-Interscience, 2000. – ISBN 0471056693

[FKFB10] Fischer, Andreas ; Keller, Andreas ; Frinken, Volkmar ; Bunke, Horst: HMM-based Word Spotting in Handwritten Documents Using Subword Models. In: *Proceedings of the 2010 20th International Conference on Pattern Recognition*. Washington, DC, USA : IEEE Computer Society, 2010 (ICPR '10). – ISBN 978–0–7695–4109–9, 3416–3419

[Fro18] Frochte, Jörg: *Maschinelles Lernen: Grundlagen und Algorithmen in Python*. 2018. http://dx.doi.org/10.3139/9783446457058. http://dx.doi.org/10.3139/9783446457058. – ISBN 978–3446452916

[GAMP15] Gordo, Albert ; Almazán, Jon ; Murray, Naila ; Perronnin, Florent: LEWIS: Latent Embeddings for Word Images and their Semantics. In: *CoRR* abs/1509.06243 (2015)

[GH12] Gutmann, Michael U. ; Hyvärinen, Aapo: Noise-contrastive Estimation of Unnormalized Statistical Models, with Applications to Natural Image Statistics. In: *J. Mach. Learn. Res.* 13 (2012), Februar, Nr. 1, 307–361. http://dl.acm.org/citation.cfm?id=2503308.2188396. – ISSN 1532–4435

[GM13] Gupta, Samta ; Mazumdar, Susmita G.: Sobel edge detection algorithm. (2013)

[Goo13] Google: *Google Code Archive*. https://code.google.com/archive/p/word2vec/, 2013. – Aufruf: 2019-08-14

[GSGN17] Giotis, Angelos P. ; Sfikas, Giorgos ; Gatos, Basilis ; Nikou, Christophoros: A survey of document image word spotting techniques. In: *Pattern Recognition* 68 (2017), 310 - 332. http://dx.doi.org/https://doi.org/10.1016/j.patcog.2017.02.023. – DOI https://doi.org/10.1016/j.patcog.2017.02.023. – ISSN 0031–3203

[HK16] Hamad, Karez ; Kaya, Mehmet: A Detailed Analysis of Optical Character Recognition Technology. In: *International Journal of Applied Mathematics, Electronics and Computers* 4 (2016), 12, S. 244–244. http://dx.doi.org/10.18100/ijamec.270374. – DOI 10.18100/ijamec.270374

[Hoc98] Hochreiter, Sepp: The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. In: *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* 6 (1998), April, Nr. 2, 107–116. http://dx.doi.org/10.1142/S0218488598000094. – DOI 10.1142/S0218488598000094. – ISSN 0218–4885

[HS97] Hochreiter, Sepp ; Schmidhuber, Jürgen: Long Short-Term Memory. In: *Neural Comput.* 9 (1997), November, Nr. 8, 1735–1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735. – DOI 10.1162/neco.1997.9.8.1735. – ISSN 0899–7667

[HSW89] Hornik, K. ; Stinchcombe, M. ; White, H.: Multilayer Feedforward Networks Are Universal Approximators. In: *Neural Netw.* 2 (1989), Juli, Nr. 5, 359–366. http://dx.doi.org/10.1016/0893-6080(89)90020-8. – DOI 10.1016/0893–6080(89)90020–8. – ISSN 0893–6080

[HZRS15a] He, Kaiming ; Zhang, Xiangyu ; Ren, Shaoqing ; Sun, Jian: Deep Residual Learning for Image Recognition. In: *CoRR* abs/1512.03385 (2015). http://arxiv.org/abs/1512.03385

[HZRS15b] He, Kaiming ; Zhang, Xiangyu ; Ren, Shaoqing ; Sun, Jian: Deep Residual Learning for Image Recognition. In: *CoRR* abs/1512.03385 (2015)

[IS15] Ioffe, Sergey ; Szegedy, Christian: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: *CoRR* abs/1502.03167 (2015). http://arxiv.org/abs/1502.03167

[JMM96] Jain, Anil K. ; Mao, Jianchang ; Mohiuddin, K.: Artificial Neural Networks: A Tutorial. In: *IEEE Computer* 29 (1996), S. 31–44

[Kal96] Kalman, Dan: A singularly valuable decomposition: The SVD of a matrix. In: *College Math Journal* 27 (1996), S. 2–23

[KDJ18] Krishnan, Praveen ; Dutta, Kartik ; Jawahar, C. V.: Word Spotting and Recognition Using Deep Embedding. In: *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)* (2018), S. 1–6

[KFDS13] Kleber, Florian ; Fiel, Stefan ; Diem, Markus ; Sablatnig, Robert: CVL-DataBase: An Off-Line Database for Writer Retrieval, Writer Identification and Word Spotting. In: *Proceedings of the 2013 12th International Conference on Document Analysis and Recognition*. Washington, DC, USA : IEEE Computer Society, 2013 (ICDAR '13). – ISBN 978–0–7695–4999–6, 560–564

[KJSR15] Kim, Yoon ; Jernite, Yacine ; Sontag, David ; Rush, Alexander M.: Character-Aware Neural Language Models. In: *CoRR* abs/1508.06615 (2015). http://arxiv.org/abs/1508.06615

[KJSR16] Kim, Yoon ; Jernite, Yacine ; Sontag, David ; Rush, Alexander M.: *Lecture notes: Character-Aware Neural Language Models*. https://nlp.seas.harvard.edu/slides/aaai16.pdf, 2016

[KSH12] Krizhevsky, Alex ; Sutskever, Ilya ; Hinton, Geoffrey E.: ImageNet Classification with Deep Convolutional Neural

Networks. Version: 2012. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf. In: Pereira, F. (Hrsg.) ; Burges, C. J. C. (Hrsg.) ; Bottou, L. (Hrsg.) ; Weinberger, K. Q. (Hrsg.): *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012, 1097–1105

[KVP+18] Kuzovkin, Ilya ; Vicente, Raul ; Petton, Mathilde ; Lachaux, J. P. ; Baciu, Monica ; Kahane, Philippe ; Rheims, Sylvain ; Vidal, Juan Ricardo M. ; Aru, Jaan: Activations of deep convolutional neural networks are aligned with gamma band activity of human visual cortex. In: *Communications Biology*, 2018

[Lee18] Lee, Jinhyuk: *charnlm-pytorch*. https://github.com/jhyuklee/charnlm-pytorch, 2018

[MB02] Marti, U.-V. ; Bunke, H.: The IAM-database: an English sentence database for offline handwriting recognition. In: *International Journal on Document Analysis and Recognition* 5 (2002), Nov, Nr. 1, 39–46. http://dx.doi.org/10.1007/s100320200071. – DOI 10.1007/s100320200071. – ISSN 1433–2833

[MB05] Morin, Frederic ; Bengio, Yoshua: Hierarchical probabilistic neural network language model. In: *AISTATS'05*, 2005, S. 246–252

[MGB+18] Mikolov, Tomas ; Grave, Edouard ; Bojanowski, Piotr ; Puhrsch, Christian ; Joulin, Armand: Advances in Pre-Training Distributed Word Representations. In: *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018

[MHRC96] Manmatha, R. ; Han, Chengfeng ; Riseman, E. M. ; Croft, W. B.: Indexing Handwriting Using Word Matching. In: *Proceedings of the First ACM International Conference on Digital Libraries*. New York, NY, USA : ACM, 1996 (DL '96). – ISBN 0–89791–830–4, 151–159

[MP43] McCulloch, Warren S. ; Pitts, Walter: A logical calculus of the ideas immanent in nervous activity. In: *The bulletin of mathematical biophysics* 5 (1943), Dec, Nr. 4, 115–133. http://dx.doi.org/10.1007/BF02478259. – DOI 10.1007/BF02478259. – ISSN 1522–9602

[MP69] Minsky, Marvin ; Papert, Seymour: *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA, USA : MIT Press, 1969

[MSC+13] Mikolov, Tomas ; Sutskever, Ilya ; Chen, Kai ; Corrado, Greg ; Dean, Jeffrey: Distributed Representations of Words and Phrases and Their Compositionality. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. USA : Curran Associates Inc., 2013 (NIPS'13), 3111–3119

[PSM10] Perronnin, Florent ; Sánchez, Jorge ; Mensink, Thomas: Improving the Fisher Kernel for Large-scale Image Classification. In: *Proceedings of the 11th European Conference on Computer Vision: Part IV*. Berlin, Heidelberg : Springer-Verlag, 2010 (ECCV'10). – ISBN 3–642–15560–X, 978–3–642–15560–4, 143–156

[PZG+16] Pratikakis, I. ; Zagoris, K. ; Gatos, B. ; Puigcerver, J. ; Toselli, A. H. ; Vidal, E.: ICFHR2016 Handwritten Keyword Spotting Competition (H-KWS 2016). In: *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016. – ISSN 2167–6445, S. 613–618

[RATL11] Rusinol, Marcal ; Aldavert, David ; Toledo, Ricardo ; Llados, Josep: Browsing Heterogeneous Document Collections by a Segmentation-Free Word Spotting Method. In: *Proceedings of the 2011 International Conference on Document Analysis and Recognition*. Washington, DC, USA : IEEE Computer Society, 2011 (ICDAR '11). – ISBN 978–0–7695–4520–2, 63–67

[RM07] Rath, Tony M. ; Manmatha, R.: Word spotting for historical documents. In: *International Journal of Document Analysis and Recognition (IJDAR)* 9 (2007), Apr, Nr. 2, 139–152. http://dx.doi.org/10.1007/s10032-006-0027-8. – DOI 10.1007/s10032–006–0027–8. – ISSN 1433–2825

[Ros58] Rosenblatt, F.: The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain. In: *Psychological Review* (1958), S. 65–386

[RRF13] Rothacker, L. ; Rusinol, M. ; Fink, G. A.: Bag-of-Features HMMs for Segmentation-Free Word Spotting in Handwritten Documents. In: *2013 12th International Conference on Document Analysis and Recognition*, 2013. – ISSN 1520–5363, S. 1305–1309

[RSP09] Rodríguez-Serrano, José A. ; Perronnin, Florent: Handwritten Word-spotting Using Hidden Markov Models and Universal Vocabularies. In:

Pattern Recogn. 42 (2009), September, Nr. 9, 2106–2116. http://dx.doi.org/10.1016/j.patcog.2009.02.005. – DOI 10.1016/j.patcog.2009.02.005. – ISSN 0031–3203

[Saho8] Sahlgren, Magnus: The distributional hypothesis. In: *Italian Journal of Linguistics* 20 (2008), 01

[Sch14] Schmidhuber, Jürgen: Deep Learning in Neural Networks: An Overview. In: *CoRR* abs/1404.7828 (2014). http://arxiv.org/abs/1404.7828

[SF16] Sudholt, Sebastian ; Fink, Gernot A.: PHOCNet: A Deep Convolutional Neural Network for Word Spotting in Handwritten Documents. In: *ICFHR*, IEEE Computer Society, 2016, S. 277–282

[SF18] Sudholt, Sebastian ; Fink, Gernot A.: Attribute CNNs for Word Spotting in Handwritten Documents. In: *Int. J. Doc. Anal. Recognit.* 21 (2018), September, Nr. 3, 199–218. http://dx.doi.org/10.1007/s10032-018-0295-0. – DOI 10.1007/s10032–018–0295–0. – ISSN 1433–2833

[SGS15] Srivastava, Rupesh K. ; Greff, Klaus ; Schmidhuber, Jürgen: Highway Networks. In: *CoRR* abs/1505.00387 (2015). http://arxiv.org/abs/1505.00387

[SK83] Sankoff, David (Hrsg.) ; Kruskal, Joseph B. (Hrsg.): *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Reading, MA : Addison-Wesley, 1983. – 1–44 S.

[Sud18] Sudholt, Sebastian: *Learning attribute representations with deep convolutional neural networks for word spotting*, TU Dortmund, Diss., 2018

[SZ14] Simonyan, Karen ; Zisserman, Andrew: Very Deep Convolutional Networks for Large-Scale Image Recognition. In: *Proceedings of the International Conference on Learning Representations* (2014), 09

[TMS03] Taylor, Ann ; Marcus, Mitchell ; Santorini, Beatrice: The Penn Treebank: An overview. (2003), 01

[TN17] Tutubalina, Elena ; Nikolenko, Sergey: Demographic Prediction Based on User Reviews about Medications. In: *Computacion y Sistemas* 21 (2017), 06, S. 227–241. http://dx.doi.org/10.13053/CyS-21-2-2736. – DOI 10.13053/CyS–21–2–2736

[Wal19] WALTERS, Austin G.: *Convolutional Neural Networks (CNN) to Classify Sentences*. https://austingwalters.com/convolutional-neural-networks-cnn-to-classify-sentences/, 2019. – Accessed: 2019-08-21

[WB16] WILKINSON, T. ; BRUN, A.: Semantic and Verbatim Word Spotting Using Deep Neural Networks. In: *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016. – ISSN 2167–6445, S. 307–312

[Wer90] WERBOS, P. J.: Backpropagation through time: what it does and how to do it. In: *Proceedings of the IEEE* 78 (1990), Oct, Nr. 10, S. 1550–1560. http://dx.doi.org/10.1109/5.58337. – DOI 10.1109/5.58337. – ISSN 0018–9219

[ZLLS19] ZHANG, Aston ; LIPTON, Zachary C. ; LI, Mu ; SMOLA, Alexander J.: *Dive into Deep Learning*. 2019. – http://www.d2l.ai

[ZZ14] ZHANG, C. ; ZHANG, Z.: Improving multiview face detection with multi-task deep convolutional neural networks. In: *IEEE Winter Conference on Applications of Computer Vision*, 2014. – ISSN 1550–5790, S. 1036–1041