Semtimentanalyse von Kundenrezensionen mittels Word Embeddings und Convolutional Neural Networks

Markus Glas
Technische Hochschule Nürnberg Georg Simon Ohm
Fakultät Informatik
Hohfederstr. 40 90489 Nürnberg
Email: glasma60952@th-nuernberg.de

***Zusammenfassung*—Künstliche Neuronale Netze haben in den letzten Jahren große Fortschritte in der Bild- und Spracherkennung erreicht. Auch in der Textklassifikation finden Künstliche Neuronal Netze immer häufiger Anwendung und erzielten dort teils bessere Ergebnisse als klassische Machine-Lerning-Verfahren. In dieser Arbeit wird untersucht, wie mit Hilfe von Convolutional Neural Networks und pre-trained Word Embeddings eine Sentimentanalyse auf Kundenrezensionen durchgeführt werden kann. Dazu werden unterschiedliche Word-Embedding-Modelle verglichen und das Convolutional Neural Network für die Textklassifikation anhand eines Datensatzes mit Kundenrezensionen optimiert. Als Referenz dient eine Support Vector Machine mit unterschiedlichen Kernel-Konfigurationen.**

I. Einführung

Sentimentanalyse (auch Sentiment Detection, Sentiment Classification oder Opinion Mining) beschreibt die automatische Auswertung von Texten mit dem Ziel, die Meinungen, Emotionen und Haltungen der verfassenden Personen zu erkennen [1]. Üblicherweise gibt es dazu mehrere Möglichkeiten, von binären Entscheidern (positiv/negativ, gut/schlecht) bis zu mehrstufigen Bewertungen. Sentimentanalyse findet in Geschäftsbereichen Anwendung, um die Meinungen zu Produkten oder die Stimmungslage ganzer Käufergruppen zu ermitteln. Weitere Anwendungsfelder sind Empfehlungssysteme, Spam-Erkennung oder statistische Auswertung zu aktuellen Themen, wie Wahlen oder Umfragen. Gerade letzteres hat durch die Nutzung sozialer Netzwerke an Popularität gewonnen.

Zur Sentimentanalyse werden häufig Techniken der Computerlinguistik mit statistischen Methoden und Machine Learning kombiniert. Durch eine Vielzahl gut dokumentierter Softwarebibliotheken und leistungsfähigen Grafikprozessoren finden Künstliche Neuronale Netze (KNN) immer mehr Anwendung, auch in kleinen Unternehmen oder dem privaten Umfeld. KNN haben sich aufgrund guter Ergebnisse, gerade in der Bildklassifikation [2], [3], [4], [5], zu einem beliebten Machine-Learning-Ansatz gegenüber bewährten Ansätzen wie Support Vector Machine (SVM) oder Logistischer Regression entwickelt.

Convolutional Neural Networks (CNN) sind eine spezialisierte Version von Künstlichen Neuronalen Netzen, welche die mathematische Faltung (engl. convolution) anstatt der Matrix-Multiplikation in mindestens einer ihrer Schichten nutzen [6] und erreichen dadurch u. a. einen Geschwindigkeitsvorteil beim Training hochauflösender Bilder. Deep Convolutional Neural Networks wie etwa GoogLeNet (auch Inception) erreichten bereits 2014 bei der ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [7] eine Fehlerrate von unter 5%[1].

Ursprünglich für Bild-Erkennung entwickelt, finden CNN mittlerweile auch in traditionellen Natural Language Processing (NLP) Aufgaben, wie Part-of-Speech-Tagging oder Named Entitiy Recognition (NER) Anwendung [8], [9]. Collobert und Weston (2008) beschreiben wie Sätze in eine *Sentence-Matrix* transformiert werden, um darauf anschließend die Faltung mit linearen Filtern durchzuführen. Die Sentence-Matrix wird anschließend vom CNN ähnlich zu Bildrepräsentationen verarbeitet.

Auch bei der Textklassifikation konnten CNN sehr gute Ergebnisse erreichen [10], [11], [12]. Yoon Kim (2014) zeigte, wie man mit einem einfachen einschichtigen CNN vergleichbare Resultate zu mehrschichtigen CNN oder bewährten Ansätzen wie SVM erzielen kann. Ebenfalls fanden CNN bei der Sentimentanalyse Anwendung und konnten gute Ergebnisse erreichen [13], [14], [15]. Aufgrund der kürzeren Trainingszeiten, sind sie in manchen Fällen anderen Formen von Künstlichen Neuronalen Netzen, wie etwa den weitaus komplexeren Rekurrenten Neuronalen Netzen, zu bevorzugen.

Die Vektorrepräsentation der Sätze können durch Sparse-Vektoren (One-Hot-Vektoren) dargestellt werden. Moderne Ansätze verwenden jedoch meist Dense-Vektor-Repräsentationen, genannt *Word Embeddings*, um zum einen die Dimension des Vektorraums zu reduzieren und zum anderen die Wörter gemäß ihrer semantischen Bedeutung im Vektorraum abzubilden [16]. Populäre Werkzeuge sind *Word2vec*[2] [17] und *GloVe*[3] [18].

[1] http://www.image-net.org/challenges/LSVRC/2014/results
[2] code.google.com/p/word2vec
[3] https://nlp.stanford.edu/projects/glove/

In dieser Arbeit wird untersucht, wie mittels Word Embeddings und Convolutional Neural Networks eine Sentimentanalyse auf Kundenrezensionen durchgeführt werden kann. Dabei wird als Basis ein Modell mit lediglich einer Convolutions-Schicht in Anlehnung an Kim's [10] und Collobert's [9] Ansatz verwendet. Es werden zwei unterschiedliche pre-trained Word-Embeddings-Modelle verglichen. Als Referenz und zur Beurteilung der Leistungsfähigkeit wird eine SVM genutzt.

II. Methodik und Werkzeuge

A. Datensatz

Als Datensatz dient die CSV-Datei *Women's E-Commerce Clothing Reviews*[4] von kaggle. Dieser enthält 23486 Zeilen mit je einem Rezensionseintrag pro Zeile. Jeder Rezensionseintrag enthält die zehn Merkmale *Clothing ID, Age, Title, Review Text, Rating, Recommended IND, Positive Feedback Count, Division Name, Department Name, Class Name*, welche die Spalten des Datensatzes darstellen. Der Datensatz basiert auf echten Rezensionen wurde jedoch anonymisiert, d. h. die Kunden- und Firmennamen wurden entfernt bzw. ersetzt.

B. Verwendete Softwarebibliotheken

Als Machine-Learning-Bibliothek wird die High-Level-API Keras[5] mit Tensorflow als Backend eingesetzt. Die Referenzimplementierung der SVM wird mit der Machine-Learning-Bibliothek *scikit-learn*[6] umgesetzt. Im Zuge der Datenvorverarbeitung werden u. a. die Python-Bibliotheken *Pandas*[7] und *Natural Language Toolkit*[8] (NLTK) genutzt.

C. Datenanalyse

Zur Sentimentanalyse werden die Merkmale *Title, Review Text* und *Rating* genutzt. Tabelle I zeigt den Werttyp sowie den Inhalt der einzelnen Merkmale. Der eigentliche Rezensionstext befindet sich im Merkmal *Review Text* und stellt ein Freitextfeld dar, welches mit mehreren Sätzen gefüllt sein kann. Da es sich um optionale Felder handelt, können diese jedoch auch leer sein.

Tabelle I
Inhalt der Merkmale

Name	Werttyp	Inhalt
Title	Text	Titel der Rezension
Review Text	Text	Rezensionstext der Kundin
Rating	Integer	Positive Nummer für die Bewertung des Produkts. Von 1 schlechteste, bis 5 beste Bewertung.

Die Sentimentanalyse wird in dieser Arbeit als Zweiklassen-Problem betrachtet, wobei zwischen den Klassen *positiv* und *negativ* unterschieden wird. Dazu werden alle Rezensionen mit einem Rating größer oder gleich 3 als positiv annotiert, Rezensionen mit kleineren Ratings (1 und 2) entsprechend als negativ. Mit Hilfe des Datenanalyse-Tools *Pandas* werden die Spalten zunächst aus der CSV-Datei ausgelesen und die Rezensionseinträge den entsprechenden Klassen zugeordnet. Dies ergibt eine Aufteilung von 21079 positiven und 2407 negativen Rezensionen. Entfernt man anschließend die Zeilen, welche keinen Rezensionstext oder Rating enthalten, so erhält man 22641 Zeilen und eine Aufteilung wie in Tabelle II dargestellt.

Tabelle II
Anzahl der Rezensionen nach der Aufteilung in zwei Klassen

Klasse	Anzahl
Positive Rezensionen	20271
Negative Rezensionen	2370

III. Datenvorverarbeitung

Die Rezensionseinträge, vor allem im Freitextfeld *Review Text*, enthalten zum Teil Sonderzeichen, welche vorab entfernt werden sollen, um die Merkmalsextraktion zur Textklassifkation eindeutiger zu gestalten. Nachfolgend werden Ausschnitte dreier Rezensionstexte aufgezeigt, welche alle die Zeichenfolge *'...'* jedoch unterschiedliche Ratings enthalten.

Tabelle III
Sonderzeichen in Rezensionen

Review Text	Rating
bought this item from online... the fit	4
This sweater is perfect for fall...it's roomy	5
If only it looked like the photo...	1

Zur Vereinheitlichung der Daten und der Entfernung wenig aussagekräftiger Zeichen, etwa Sonderzeichen oder Interpunktionen, findet vor dem Training eine Bereinigung mittels regulärer Ausdrücke statt. Dazu werden die zwei nachfolgenden regulären Ausdrücke definiert:

1) `[^a-zA-Z0-9?!.]|[\.]{2,}`
2) `\s{2,}`

Ausdruck 1) wird genutzt, um alle nicht-alphanumerischen Zeichen, mit Ausnahme der Zeichen *'?.!'*, oder mehr als zwei aufeinanderfolgende Punkte zu entfernen. Alle Zeichen, welche diesem Ausdruck nicht entsprechen, werden durch ein Leerzeichen ersetzt. Ausdruck 2) entfernt anschließend mehr als zwei aufeinanderfolgende Leerzeichen durch ein einzelnes.

A. Stoppwörter

Stoppwörter bezeichnen extrem häufig auftretende Wörter in Texten, welche jedoch keinen Mehrwert hinsichtlich der Einordnung der Texte schaffen [19]. Dazu zählen beispielsweise Artikel oder Konjunktionen. Im vorliegenden Datensatz besteht ein großer Teil der häufigsten Wörter daraus. Abbildung 1 zeigt die zehn häufigsten Wörter der beiden Klassen. Zu sehen ist, dass Worte wie 'the' und 'and' einen großen Anteil in beiden Klassen besitzen. Aus diesem Grund wird im Zuge der Datenvorverarbeitung eine Stoppwortbereinigung anhand einer vordefinierten Liste durchgeführt.

[4]https://www.kaggle.com/nicapotato/womens-ecommerce-clothing-reviews
[5]https://keras.io/
[6]http://scikit-learn.org
[7]https://pandas.pydata.org/
[8]https://www.nltk.org/

Dazu wird die Stoppwort-Liste aus der Python-Bibliothek NLTK genutzt. Zudem werden die fünf häufigsten Wörter nach der Stoppwortbereinigung, welche in beiden Klassen existieren, entfernt. Diese sind *dress, size, top, fit* und *like*.

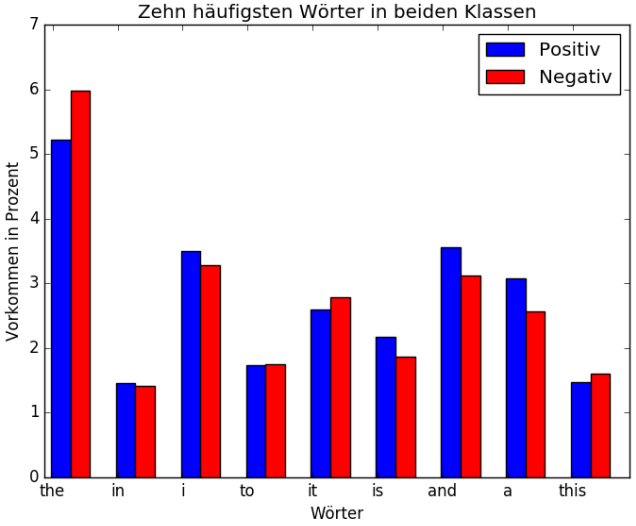

Abbildung 1. Die zehn häufigsten Wörter beider Klassen

B. Vektorrepräsentationen

Um die Texte im CNN nutzen zu können, müssen diese vorab in numerische Werte umgerechnet werden. In der Textklassifikation hat sich die Repräsentation von Wörtern oder Sätzen als Vektoren in einem *k*-dimensionalen Vektorraum $\mathbb{R}^k$ bewährt (siehe *Vector Space Model* [20]). Prinzipiell existieren zur Berechnung der Vektoren mehrere Möglichkeiten, von einfachem Zählen der Vorkommen bis zur Gewichtung in Relation zur gesamten Textmenge. In dieser Arbeit werden Vektorrepräsentationen aus vortrainierten (pre-trained) Modellen gewählt, d. h. die Vektor-Repräsentationen werden nicht berechnet, sondern für das entsprechende Wort nachgeschlagen und ausgewählt.

Zu Zwecken der Zuordnung von Vektor-Repräsentationen und den eindeutigen Wörter in der Textmenge (dem Vokabular) wird ein Wortindex erstellt. Jedes eindeutige Wort erhält eine Skalar woraufhin sich eine ganzzahlige Repräsentation der Sätze bilden lässt, wie nachfolgend an einem Beispiel zu sehen:

Rezensionstext:	I love, love, love this jumpsuit
Indexrepräsentation:	[4, 1, 1, 1, 2, 20]

Durch unterschiedliche Längen der Rezensionstexte kann im späteren Vektorraum ein signifikanter Längenunterschied entstehen und zu hohen Abständen im Raum trotz ähnlicher Wortfolge führen. Um dem vorzubeugen, werden alle Rezensionen auf dieselbe Länge vereinheitlicht. Dazu wird die längste Wortsequenz ermittelt und kürzere Sequenzen angepasst, d. h. mit Nullen aufgefüllt.

C. Word Embeddings

Die Vektorrepräsentationen in den vortrainierten Modellen entsprechen *Word Embeddings*. Dieser spezielle Typ von Wort-Vektor-Repräsentationen bildet die Wörter gemäß ihrer semantischen Bedeutung im Vektorraum ab. Somit werden Wörter mit ähnlicher Bedeutung, etwa *nice* und *sweet*, im Vektorraum mit geringem Abstand dargestellt. Wohingegen unähnliche Wörter, wie *love* und *hate*, einen größeren Abstand zueinander besitzen.

Die Repräsentation lässt sich wie folgt formal definieren. Sei $x_i \in \mathbb{R}^d$ ein *d*-dimensionaler Wortvektor, wobei *i* die Position des Wortes im Satz bestimmt. Somit ist ein Satz der Länge *s* durch $x_{1:s} = x_1 \oplus x_2 \oplus ... \oplus x_s$ definiert, wobei $\oplus$ die Konkatenation darstellt.

D. Vortrainierte Modelle

In dieser Arbeit werden zwei pre-trained Modelle mit zugrundeliegenden unterschiedlichen Verfahren und verschiedenen Trainings-Korpora verwendet. Nachfolgend werden zunächst beide Verfahren kurz erläutert.

Word2Vec[17] nutzt zur Erstellung der semantischen Wort-Vektoren ein drei-schichtiges Feedforward Neural Network. Die semantischen Zusammenhänge werden anhand der Nachbarwörter vorhergesagt, weshalb Word2Vec zu den *Prognosegestützen*-Modellen (prediction based models) gezählt wird. Die Anzahl der Nachbarwörter kann durch eine Fenstergröße festgelegt werden.

GloVe[18] zählt hingegen zu den *Zählbasierten*-Modellen (count based models) und erstellt zuerst eine Matrix X jedes Wortes im Vokabular zusammen mit seinem Kontext. So repräsentiert jedes Element X_{ij} der Matrix wie oft ein Wort *i* im Kontext des Wortes *j* auftritt. Die so entstandene Matrix wird anschließend mittels Matrix-Faktorisierung auf einen niedrigere Dimension projiziert, welche die Vektorrepräsentation eines jeden Wortes als Zeile enthält.

Die beiden verwendeten pre-trained Modelle sind folgende:

1) ***Word2vec* Google News Modell**[9]: Trainiert auf 100 Milliarden Wörtern aus Google News mit 300-dimensionalen Vektoren für drei Millionen Wörter und Sätze.
2) ***GloVe* Wikipedia Modell**[10]: Trainiert auf 400.000 Wörtern aus der englischsprachigen Wikipedia mit 100-dimensionalen Vektoren.

IV. Architektur und Modellbildung

Die Architektur des Convolutional Neural Networks, dargestellt in Abbildung 2, setzt sich aus einer Embedding-Schicht, einer Convolution-Schicht, gefolgt von Filtern, einer Max-Pooling-Schicht und dem Klassifikator mit Sigmoid-Funktion zusammen. Da andere Arbeiten bereits zeigten, dass CNN mit lediglich einer Convolutions-Schicht bei Textklassifikationen sehr gute Ergebnisse lieferten [9], [10], wird auf eine komplexere Architektur verzichtet. Dadurch reduziert sich die

[9]https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM
[10]http://nlp.stanford.edu/data/glove.6B.zip

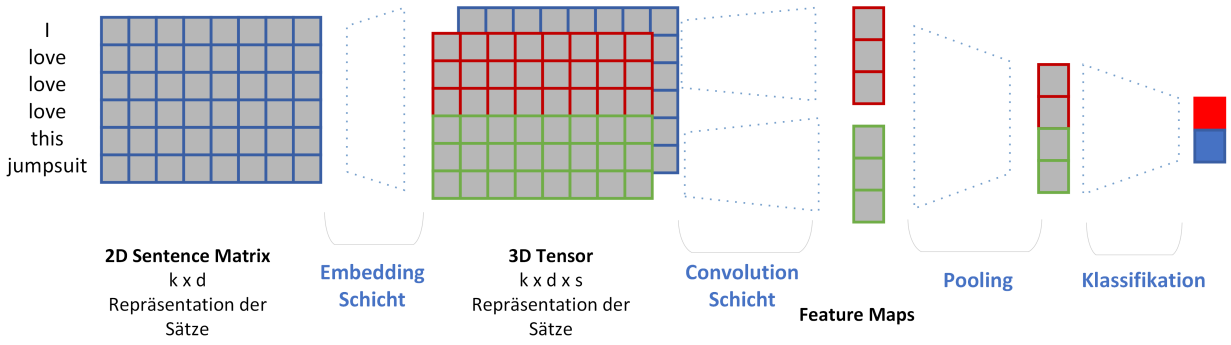

Abbildung 2. Schematische Darstellung der Architektur des CNN

Trainingszeit und die Optimierung wird durch eine geringere Anzahl an Parametern einfach gehalten.

Die Eingabe der Rezensionen in das CNN erfolgt über eine zweidimensionale Sentence-Matrix in Anlehnung an [8]. Diese wird vorab durch die Zuordnung der entsprechenden Word-Embedding-Repräsentation zu den Wörtern im Satz erstellt.

Die Breite d der Matrix entspricht der jeweiligen Vektorlänge (Word2Vec: d=300, GloVe: d=100). Wird für ein Wort keine entsprechende Wort-Embedding-Repräsentation in einem der Modelle gefunden, so wird dieses mit einem Null-Vektor dargestellt. Die Embedding-Schicht[11] übernimmt zunächst die Aufgabe, die 2D Repräsentation der Embedding-Matrix der Größe $k \times d$ in einen 3D Tensor der Größe $k \times d \times s$ zu wandeln, wobei s die Länge der Wortsequenz darstellt.

Die nachfolgende Convolutions-Schicht besitzt mehrere Filtern (auch Kernel genannt), welche die Faltung auf den Vektorrepräsentationen durchführt. Diese besitzen eine Breite entsprechend der Vektorlänge und eine feste Höhe von drei Vektoren. Die Filter werden gleichzeitig über die Sentence-Matrix bewegt und die Ausgabe auf neue Matrizen, den sogenannten *Feature-Maps*, projiziert. Die Feature-Maps werden dann in der Pooling-Schicht gesammelt, um die Dimensionen zu reduzieren. Die Ausgabe wird dann einer Fully-Connected-Schicht übergeben, um die Klassifikation mittels Sigmoid-Funktion durchzuführen.

Zur Validierung wird der Datensatz in 80% Trainingsdaten und 20% Validierungsdaten aufgeteilt.

V. Resultate

A. Support Vector Machine

Als Referenzimplementierung dient eine SVM mit jeweils linearem und RBF-Kernel (Radial Basis Function) in Anlehnung an Zhang und Wallace [21]. Die Merkmalsextraktion wird zunächst mit *Bag-of-words* (BOW), d. h. einem einfachen Zählen der auftretenden Wörter im Vokabular, durchgeführt. Des Weiteren wird die gewichtete Methode *Term-Frequency Inverse-Document Frequency* (TF-IDF) verwendet, welche die Termhäufigkeit in Relation zur Dokumentenhäufigkeit setzt. In beiden Fällen wird ebenfalls die Stoppwort-Liste aus dem NLTK verwendet.

[11]https://keras.io/layers/embeddings/

Die Validierung erfolgt auf 20% des Datensatzes mit Hilfe der Korrektklassifizierungsrate (Accuracy). Ein Vergleich der unterschiedlichen Konfigurationen (siehe Tabelle IV) zeigt, dass die SVM mit RBF-Kernel und TF-IDF die höchste Accuracy auf dem vorliegenden Datensatz erreicht.

Tabelle IV
Resultate der SVM

Verfahren	Accuray
SVM (BOW+Linear)	91,02%
SVM (TFIDF+Linear)	91,38%
SVM (BOW+RBF)	91,46%
SVM (TFIDF+RBF)	**92,02%**

B. Convolutional Neural Network

Das CNN wird mit der in Tabelle V gezeigten Basiskonfiguration in Anlehnung an Zhang und Wallace [21] initialisiert. Mit Hilfe einer Rastersuche werden die Hyperparameter *Dropout-Rate* und *Batch-Size* optimiert. Diese ergab eine optimale Dropout-Rate von 0,5 und eine Batch-Size von 32.

Tabelle V
Basiskonfiguration des CNN

Parameter	Wert
Kernelgröße	3
Feature-Map-Größe	300
Aktivierungsfunktion	ReLu
Pooling	1-max-pooling
Dropout-Rate	0,0 - 0,5
Klassifikationsfunktion	Sigmoid
Optimizer	Adam
Epochen	50
Batch-Size	16, 32, 64

Die Anzahl der Epochen wird großzügig gewählt, jedoch nie ausgenutzt da das Training mit Hilfe eines *EarlyStopping*[12] durchgeführt wird. Somit wird das Training beendet, sobald bei fünf aufeinanderfolgen Epochen keine Verringerung im *Loss*, also der Kompatibilität zwischen der Klassenvorhersage und den wahren Klassen, stattfindet. Mit beiden pre-trained Modellen fand bereits nach unter zehn Epochen keine Verbesserung statt. Des Weiteren wurde in beiden Variante eine Klassifikation mit und ohne Stoppwortbereinigung evaluiert.

[12]https://keras.io/callbacks/#earlystopping

Die Ergebnisse des CNN sind in Tabelle VI dargestellt. Die Accuracy erreicht in beiden Fällen knapp bessere Ergebnisse als die SVM mit RBF-Kernel. Zu beachten ist jedoch auch der hohe Loss, welcher in beiden Varianten bei über 15% liegt. Ebenfalls anzumerken ist, dass Word2Vec ohne Stoppwortbereinigung bessere Ergebnisse erzielt. Beim Einsatz von GloVe werden jedoch mit vorheriger Stoppwortbereinigung bessere Ergebnisse erzielt.

Tabelle VI
RESULTATE DES CNN

Verfahren	**Accuray**	**Loss**
CNN (Word2Vec) ohne Stoppwortbereinigung	**92,37%**	**16,57%**
CNN (Word2Vec) mit Stoppwortbereinigung	91,72%	19,50%
CNN (GloVe) ohne Stoppwortbereinigung	91,65%	19,02%
CNN (GloVe) mit Stoppwortbereinigung	**92,18%**	**17,82%**

Gründe für die hohe Fehlerrate können in einer zu komplexen Modellwahl für den vorliegenden Datensatz liegen. Gerade die Klasse mit negativen Rezensionen besitzt mit 2370 relativ wenige Trainingsbeispiele. Als weitere Ursache können die teils umgangssprachlichen Formulierung in einigen Rezensionstexten darstellen. Die Wortwahl entspricht teils nicht dem Vokabular der englischsprachigen Wikipedia oder Nachrichtenartikeln entsprechen. Ein vorgelagerter Spell-Checker oder eine Abbildung auf den Wortstamm (Stemming) könnte dabei Abhilfe schaffen. Ebenfalls können noch weitere Hyperparameter-Optimierungen durchgeführt werden, welche möglicherweise bessere Ergebnisse liefern.

VI. ZUSAMMENFASSUNG

In dieser Arbeit wurde gezeigt, wie eine Sentimentanalyse mit Hilfe eines Convolutional Neural Network durchgeführt werden kann. Dabei wurden Word-Embeddings zur Vektorisierung der Texte eingesetzt, um auch die Semantik zu beachten. Die zwei verschiedenen eingesetzten Word-Embedding-Tools, *Word2Vec* und *GloVe*, zeigten in den Endresultaten keine großen Unterschiede, auch wenn das pre-trained Modell von *GloVe* auf deutlich weniger Texten trainiert wurde. Gegenüber der Referenzimplementierung wurden leichte Verbesserungen in der Accuracy erreicht.

Die gewählten Vorverarbeitungsschritte sind so oder in abgewandelter Form auch auf andere Textklassifikationen anwendbar. Jedoch zeigen die Resultate des CNN, dass diese in Abhängigkeit zur eingesetzten Vektorisierung steht und deshalb eine Evaluation verschiedener Vorverarbeitungen obligatorisch ist.

LITERATUR

[1] B. Liu, "Sentiment Analysis and Opinion Mining," *Synthesis Lectures on Human Language Technologies*, vol. 5, no. 1, pp. 1–167, May 2012.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3098997.3065386

[3] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *arXiv:1512.00567 [cs]*, Dec. 2015, arXiv: 1512.00567. [Online]. Available: http://arxiv.org/abs/1512.00567

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," *arXiv:1409.4842 [cs]*, Sep. 2014, arXiv: 1409.4842. [Online]. Available: http://arxiv.org/abs/1409.4842

[5] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *arXiv:1502.03167 [cs]*, Feb. 2015, arXiv: 1502.03167. [Online]. Available: http://arxiv.org/abs/1502.03167

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, ser. mitp Business. MITP Verlags GmbH, 2018. [Online]. Available: https://books.google.de/books?id=rtHMswEACAAJ

[7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *arXiv:1409.0575 [cs]*, Sep. 2014, arXiv: 1409.0575. [Online]. Available: http://arxiv.org/abs/1409.0575

[8] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 160–167.

[9] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural Language Processing (Almost) from Scratch," *Journal of Machine Learning Research*, vol. 12, no. Aug, pp. 2493–2537, 2011. [Online]. Available: http://www.jmlr.org/papers/v12/collobert11a.html

[10] Y. Kim, "Convolutional Neural Networks for Sentence Classification," *arXiv:1408.5882 [cs]*, Aug. 2014, arXiv: 1408.5882. [Online]. Available: http://arxiv.org/abs/1408.5882

[11] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A Convolutional Neural Network for Modelling Sentences," *arXiv:1404.2188 [cs]*, Apr. 2014, arXiv: 1404.2188. [Online]. Available: http://arxiv.org/abs/1404.2188

[12] P. Wang, J. Xu, B. Xu, C. Liu, H. Zhang, F. Wang, and H. Hao, "Semantic Clustering and Convolutional Neural Network for Short Text Categorization," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 352–357. [Online]. Available: http://www.aclweb.org/anthology/P15-2058

[13] X. Ouyang, P. Zhou, C. H. Li, and L. Liu, "Sentiment Analysis Using Convolutional Neural Network," in *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, Oct. 2015, pp. 2359–2364.

[14] S. Liao, J. Wang, R. Yu, K. Sato, and Z. Cheng, "CNN for situations understanding based on sentiment analysis of twitter data," *Procedia Computer Science*, vol. 111, pp. 376–381, Jan. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050917312103

[15] C. dos Santos and M. Gatti, "Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts," in *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin, Ireland: Dublin City University and Association for Computational Linguistics, Aug. 2014, pp. 69–78. [Online]. Available: http://www.aclweb.org/anthology/C14-1008

[16] D. Jurafsky and J. H. Martin, *Speech and Language Processing*. Pearson, 2014.

[17] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," p. 9, 2014.

[18] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[19] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. New York: Cambridge University Press, 2008, oCLC: ocn190786122.

[20] G. Salton, A. Wong, and C. S. Yang, "A Vector Space Model for Automatic Indexing," *Commun. ACM*, vol. 18, no. 11, pp. 613–620, Nov. 1975. [Online]. Available: http://doi.acm.org/10.1145/361219.361220

[21] Y. Zhang and B. Wallace, "A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification," *arXiv:1510.03820 [cs]*, Oct. 2015, arXiv: 1510.03820. [Online]. Available: http://arxiv.org/abs/1510.03820