Transferaufgabe - Sentiment Analysis mit Word Embeddings

August 23, 2020

IDEEN / TODOS:

- Medium Artikel zum Dataset: https://towardsdatascience.com/sentiment-analysis-and-product-recommendation-on-amazons-electronics-dataset-reviews-part-1-6b340de660c2

LINKS: - hate speech: https://romanorac.github.io/machine/learning/2019/12/02/identifying-hate-speech-with-bert-and-cnn.html - sentiment tut: https://github.com/bentrevett/pytorch-sentiment-analysis/blob/master/5%20-%20Multi-class%20Sentiment%20Analysis.ipynb - Kim-CNN paper: https://arxiv.org/pdf/1904.12848.pdf

TODO: Besonderheiten beim Nutzerreview-Korpus sind die Kürze der Texte und die fehlerhafte Orthographie.

TODO: fragstellung - Word Embeddings (GloVe, FastText) mit CNN vs. BERT Transformer, die die Embeddings nutzen (SOTA) - Bert kann nicht so einfach mit CNN verwendet werden, da Modell mitgeliefert werden muss - interessant, wie CNN gegen BERT, welches eine Art Weiterentwicklung von RNN ist, funktioniert - verschiedene CNN Netze?

[43]:
```
import pandas as pd
from models import KimCNN

corpus = pd.read_csv("../corpora/alter_small_amazon_reviews_electronic.csv")
#print(corpus[corpus.length <= 20].sample(3).to_markdown())
```

Inhaltsverzeichnis

1 Einleitung

Die **Sentiment Analysis** ist ein wissenschaftliches Feld des Natural Language Processing, welches sich sich mit Texten befasst, die Meinungen, Stimmungen, Einschätzungen und Emotionen von Menschen beinhalten (Liu 2015, S. 1). Dazu gehören z.B. Filmkritiken, Produktreviews oder Twitterposts. In dieser Arbeit wird die Sentiment Analysis als eine besondere Form der Textklassifikation angesehen. Wichtig bei der Sentiment Analysis sind vor allem Schlüsselwörter oder -phrasen, die Auskunft über die Meinung, Stimmung oder Emotion des Textes gibt. In früheren Jahren wurden dafür zu Unterstützung der Textklassifikationstechniken "Stimmungslexika" verwendet, die den enstprechenden Wörter/Phrasen eine Stimmung (z.B. "gut", "schlecht", "neutral") zuordneten (Liu 2015, S. 10f.). Dadurch konnten jedoch Probleme wie die sich ändernde Semantik eines Wortes hinsichtlich des Kontextes nicht gelöst werden (Liu 2015, S. 10f.). In den letzten Jahren wurden daher immer häufiger die sich als sehr effektiv erweisenden **Word Embeddings** im Rahmen der Sentiment Analysis verwendet (Petrolito, Dell'Orletta 2019, S. 330), da sie beim Erstellen der Wortrepräsentation den Kontext eines Wortes berücksichtigen.

In dieser Arbeit wurde mithilfe eines Convolutional Neural Networks (CNN) eine Sentiment Analysis auf ein Korpus von Nutzerreviews des Onlineversandhändlers **Amazon** durchgeführt. Es sollte untersucht werden, welche Word Embeddings in Kombination mit welchen Parametern des Neuronalen Netzes die besten Ergebnisse liefern und sich so am besten für die Sentiment Analysis eignen. Es wurden dafür verschiedene CNN-Architekturen wie KimCNN oder DPCNN (TODO) ausprobiert (Kapitel 4.2 TODO). Als Word Embeddings wurden verschiedene vortrainierte Embeddings von **GloVe** und **FastText** verwendet (Kapitel 3.1 TODO). Die Ergebnisse der Sentiment Analysis in Kombination mit den Word Embeddings sollten mit einem weiteren Experiment verglichen werden, beim dem das Korpus mithilfe des BERT-Modells und **BERT**-Embeddings *fine-tuned* wurde, da dies für das Amazon Reviews Datensatz State-of-the-art ist (Kapitel 3.1)(FN: Siehe https://paperswithcode.com/sota/sentiment-analysis-on-amazon-review-full (abgerufen am 23.08.2020).

2 Das Korpus

Das verwendete Korpus ist ein Sammlung von englischsprachigen Nutzerreviews zu den Produkten des Onlineversandhändlers **Amazon** von Julian McAuley (Quelle). Der Zeitraum der Veröffentlichungsdaten der Reviews im originalen Korpus liegt zwischen dem Mai 1996 und dem Oktober 2018. Diese Zeitspanne umfasst ~ 233 Millionen Reviews aus 29 verschiedenen Produktkategorien. Zu jedem Produkt stehen die Bewertung in einer Skala von 1 bis 5 (sehr schlecht bis sehr gut) zur Verfügung, der Reviewtext, die Anzahl der "Nützlich"-Votierungen, eine Verifizierung von Amazon, die Produkt-Metadaten und weitere Links.

Für diese Arbeit wurde eine verkürzte Version des Korpus verwendet. Alle Produktreviews stammen aus der Kategorie "Elektronik" und aus dem Jahr 2018. Es wurden nur Reviews berücksichtigt, die zu jeder ausgewählten Metainformation ("Bewertung", "Nutzername", "Reviewtext", "Verifizierung", "Datum") Werte enthielten. Das Bewertungssystem von fünf Sternen ist problematisch, da diese oft eine bimodale Verteilung aufweisen hinsichtlich der Extremwertungen 1 und 5.(FN: Siehe http://www.lifewithalacrity.com/2006/08/using_5star_rat.html (abgerufen am 23.08.2020).) Vor allem die 2 und 4 Sterne lassen sich nicht immer ganz einfach von den 1 und 5

Sternen abgrenzen, da Nutzer individuelle Bewertungsrichtlinien festlegen. Für diese Arbeit sollen deshalb die 2 und 4 Sterne Bewertungen mit den 1 und 5 Sterne Bewertungen zusammengeführt werden, sodass es nur noch drei Bewertungseinheiten gibt (Tabelle 1): positiv (4 und 5 Sterne), neutral (3 Sterne), negativ (1 und 2 Sterne). Das resultiernde Korpus zeigte hinsichtlich der Klassenverteilung eine starke Unausgeglichenheit, weshalb mithilfe von zufälligem Downsampling zu jeder Klasse 15000 Nutzerreviews ausgewählt wurden, um ein ausgeglichenes Korpus zu erhalten (Größe: 45000 Reviews). Zusätzlich wurde die Spalte "length" hinzugefügt, welche die Länge der Reviews beinhaltet.

	rating	name	review	verified	vote	date	length
35382	positive	Joseph N. Matura	worked perfectly	True	0	19.03.2018	2
14209	negative	waltergreene	This camera tries to auto focus too much	True	0	11.03.2018	8
37948	positive	matjones94	Works great	True	0	15.01.2018	2

Tabelle 1: Das Amazon-Nutzerreview-Korpus in der Kategorie "Electronic" von 2018 (Ausschnitt).

3 Theoretische Grundlagen

3.1 Word Embeddings

Word Embeddings sind eine besondere Art der distributiven Repräsentation von Wörtern (PILEHEVAR 2020, S. 27). Word Embeddings bauen auf der Idee der **Distributionellen Hypothese** von John Rupert Firth auf, die besagt, dass die Bedeutung eines Wortes durch sein Umfeld geprägt ist. Wörter, die einen ähnlichen Kontext besitzen, haben eine ähnliche Bedeutung. Word Embeddings konstruieren diese Wortrepräsentationen mithilfe von Neuronalen Netzen und basieren meist auf Sprachmodellierungstechniken, mithilfe derer nachfolgende oder fehlende Wörter vorausgesagt werden. In dieser Arbeit wurden die Word Embeddings **GloVe**, **FastText** und **BERT** verwendet. **GloVe** wurde 2014 von Pennigton et. al. veröffentlicht (Pennigton u.a. 2014). Anders als andere Word Embedding Verfahren verwendet GloVe für die Darstellung der Worthäufigkeiten keine Voraussagemodelle in Form von neuronalen Netzen, sondern eine Kookkurrenz-Matrix, die mithilfe einer Mischung aus maschinellem Lernen und statischen Verfahren aus den Texten gewonnen wird. GloVe hat den Nachteil, dass es nicht gut mit unbekannten Wörtern arbeiten kann (= *Out of vocabulary*-Fehler). Ein Verfahren, welches dieses Problem umgeht, ist das 2016 von Bojanowski et. al. veröffentlichte **FastText** (Bojanowski u.a. 2016). FastText löst das OOV-Problem, indem es während des Trainings anstatt ganzer Wörter Buchstaben N-Gramme lernt, aus denen unbekannte Wörter zusammengebaut werden können. Dies ist leider keine optimale Lösung, da Wörter zwar aus ähnlichen Buchstaben N-Gramm-Bestandteilen bestehen, sich aber semantisch trotzdem stark voneinander unterscheiden können. Eine bessere Lösung des OOV-Problems bietet das 2018 von Devlin et. al. veröffentliche **BERT** (Devlin u.a. 2018). Wie FastText auch lernt BERT keine ganzen Wörter, sondern Teilwörter, aus welchen es unbekannte Wörter zusammenbauen kann. Anders als FastText oder GloVe zählt BERT jedoch zu den contextualised Word Embeddings, was bedeutet, dass es den Kontext eines Wortes bei der Bildung des Embeddings berücksichtigt. Dies erreicht BERT durch den sogenannten **Attention**-Mechanismus des **Transformers**-Modell, der es erlaubt, relevanten Worten in einer Sequenz mehr Bedeutung als anderen Worten zuzuschreiben. Dabei betrachtet BERT vorhergehende und nachfolgende Wörter (unidirektionaler Ansatz). Da

sich durch diesen Ansatz Wörter jedoch "selber sehen" können, verwendet BERT zusätzlich noch die Konzepte **Next Sentence Prediction** (NSP) und **Masked Language Modeling** (MLM). Bei der Next Sentence Prediction überprüft BERT, ob der akutelle betrachtete Satz kontextuell zum nachfolgenden Satz passt. Beim Masked Language Modeling maskiert BERT nach einer gewissen Strategie Wörter, um diese mithilfe der umliegenden Wörter voraussagen zu können. Somit lernt BERT den Kontext von Wörtern, was es BERT erlaubt, zwischen mehrdeutigen Wörtern zu unterscheiden. Ein weiterer Unterschied von BERT zu GloVe und FastText ist, dass es keine **statische**, sondern eine **dynamische** Repräsentation der Wörter liefert. Worte, die die gleiche Schreibweise besitzen, können somit durch unterschiedliche Vektoren dargestellt werden, je nach Kontext und Reihenfolge. Dies bedeutet aber auch, dass auch nach dem Training des Modells dieses für die Benutzung der Embeddings obligatorisch ist. Bei den statischen Word Embeddings GloVe und FastText werden lediglich die Embeddings in Form von Wortvektoren benötigt.

3.2 Convolutional Neural Networks

Convolutional Neural Networks (CNN) sind eine bestimmte Form von neuronalen Netzen, die vorwiegend für die Klassifizierung von Bildern verwendet werden. Anders als Feedforward Netze lernt ein CNN keine globalen Muster in den Daten, sondern lokale Muster. Somit werden keine zufälligen, sondern aufeinanderfolgende und umliegende Merkmalskombinationen gelernt. Bei einem Bild sind das Ausschnitte der Bilder, die z.B. mithilfe eines quadratischen 3x3 Filters erzeugt werden. Das Lernen der Merkmalskombinationen geschieht in den namensgebenden **Convolutional Layern**, die vor den Dense Layern eines Neuronalen Netzes angefügt werden. Mithilfe der Convolutional Layer ist ein CNN in der Lage, wiederkehrende Muster an verschiedensten Stellen des Bildes zu erkennen. Weiterhin erlaubt eine Aneinandereihung mehrerer Convolutional Layer das Erkennen komplexerer und abstrakterer Merkmale, wobei zu Beginn eines Convolutional Layers eher kleinere Muster wie Kanten erkannt werden (Chollet, 2018, S. 123).

In den Convolutional Layern wird ein **Filter** (oder: Kernel) auf die Featurematrix angewandt, woraus eine **Feature Map** (auch: Activation Map) entsteht. Dies wird durch Abbildung 1 deutlich: Auf jedes mögliche Feature eines Eingabebilds, welches aus 25 (5x5) Pixeln bzw. Featuren besteht, wird ein 3x3 Filter angewandt. Dies ist bei allen Featuren außer den Features am Rand möglich, da dort der Filter nicht vollends angewandt werden kann. Das Ergebnis für jede Anwendung des Filters auf möglichen Features der Eingabematrix ist ein Wert, der ein *Neuron* bzw. ein *gelerntes Feature* darstellt (Weidman, 2020, S. 130). Mathematisch wird der Wert mit $w^T \cdot x + b$ berechnet, wobei w die zufällig initialisierte Filtermatrix, x die Eingabematrix und b ein typischer Bias eines Neuronalen Netzes ist. Aus allen so berechneten Werten bzw. gelernten Features ergibt sich die **Feature Map**. Dieser ganze Vorgang wird **Convolution** (deutsch: Faltung) genannt und ist die Kernoperation jedes CNNs.

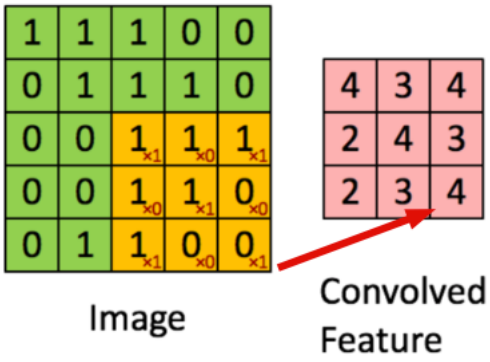

Abbildung 1. Die Abbildung wurde aus folgendem Artikel entnommen: An Intuitive Explanation of Convolutional Neural Networks.

Da jeder Filter in der Eingabematrix ein bestimmtes Muster oder Konzept erkennt, werden mehrere Filter in einem Convolutional Layer verwendet. Je mehr Filter verwendet werden, desto mehr Features können aus den Eingabedaten extrahiert werden (FN: Karn, 2016). Auf jede Ausgabe eines Convolutional Layer wird eine Aktivierungsfunktion wie z.B. die ReLU-Funktion angewandt. Bevor die Ausgabe eines vorangegangenen Convolution Layers an das nächste Convolution oder Dense Layer übergeben wird, wird sie oft einem **Pooling Layer** übergeben. Die Aufgabe eines Pooling Layers ist die Verkleinerung des Bildes mithilfe von Pooling-Filtern, um die Anzahl der Parameter, die Rechenlast und das Risiko von Overfitting zu verringern (Géron, 2020, S. 460). Es gibt verschiedene Arten von Pooling, das in dieser Arbeit verwendete Pooling ist das Max-Pooling, bei dem nur der größte Wert des Pooling Filters dem folgenden Layer übergeben wird.

Es ist jedoch auch möglich, CNNs für andere Dateitypen zu verwenden. Da in dieser Arbeit Textdaten verwendet werden, muss die CNN Architektur für diese Daten angepasst werden. Anstatt zweidimensionaler Convolutional Layer benutzt man bei Textdaten eindimensionale Convolutional Layer (Géron, 2020, S. 524). Beim eindimensionalen Convolutional Layer werden viele Filter über eine Textsequenz geschoben, womit für jeden Filter eine eindimensionale Feature Map erzeugt wird. Somit lernt bei jedem Filter ein einzelnes, sequenzielles Muster und die Filter können in Kombination mit Max-Pooling relevante N-Gramme entdecken (JACOVI, 2018, S. 56). Damit können eindimensionale CNNs eine schnellere Alternative zu Rekurrenten Neuronalen Netzen für simple Aufgaben wie die Textklassifikation sein (CHOLLET, 2018, S. 225).

4 Experimente

4.1 Aufbau

Für die Experimente wurde das Korpus in einen Trainings-, Validierungs- und Testdatensatz aufgeteilt. Die Aufteilung erfolgte nach dem Pareto-Prinzip, 80% der Daten wurden für als Trainingsdaten verwendet und jeweils 10% für die Validierungs- und Testdaten. Stoppwörter wurden beibehalten, um die Größe der im Durchschnitt sehr kurzen Reviews (~49 Tokens) nicht noch weiter zu verringern. Für die Messung der Genauigkeit wurde die **Categorical Accuracy** verwendet, welche mit der folgenden Formel berechnet wird:

$$\text{Categorical Accuracy} = \frac{\text{Anzahl der korrekten Voraussagen}}{\text{Gesamtanzahl aller Voraussagen}}$$

Für die unterschiedlichen Embeddings wurden folgende vortrainierte Embeddings verwendet:

Embedding	Modell	Details
BERT	bert-large-uncased	24-layer, 1024-hidden, 16-heads, 340 Millionen Parameter. Wurde auf kleingeschriebenen englischen Texten trainiert. Siehe https://huggingface.co/transformers/pretrained_models.html (abgerufen am 23.08.2020).
FastText	fasttext.en.300d	Eine Million Wortvektoren wurden auf einem Wiki-Korpus von 2017, dem UMBC Korpus und dem statmt.org Nachrichtenkorpus erstellt. Die Sprache ist Englisch. Siehe https://fasttext.cc/docs/en/english-vectors.html (abgerufen am 23.08.2020).
FastText	fasttext.simple.300d	Eine Million Wortvektoren wurden auf einem Wiki-Korpus von 2017, dem UMBC Korpus und dem statmt.org Nachrichtenkorpus erstellt. Die Sprache ist einfaches Englisch. Siehe https://fasttext.cc/docs/en/english-vectors.html (abgerufen am 23.08.2020).
GloVe	glove.6B.300d	Diese Embeddings wurden auf einem Wiki-Korpus von 2014 und dem Gigaword Korpus trainiert. Es beinhaltet 6 Milliarden Tokens und ein Vokabular von 400000 Tokens. Siehe https://nlp.stanford.edu/projects/glove/ (abgerufen am 23.08.2020)
GloVe	glove.840B.300d	Diese Embeddings wurden auf dem Common Crawl Korpus trainiert. Es beinhaltet 840 Milliarden Tokens und ein Vokabular von 2,2 Millionen Tokens. Siehe https://nlp.stanford.edu/projects/glove/ (abgerufen am 23.08.2020)
GloVe	glove.twitter.27B.200d	Diese Embeddings wurden auf Tweets von Twitter trainiert. Es beinhaltet 2 Milliarden Tweetw, 27 Milliarden Tokens und ein Vokabular von 1,2 Millionen Tokens. Siehe https://nlp.stanford.edu/projects/glove/ (abgerufen am 23.08.2020)

4.2 KimCNN

2014 veröffentlichte Yoon Kim das Paper “Convolutional Neural Networks for Sentence Classification”, in welchem er ein CNN Modell vorstellte (KIM, 2014), welches im Verlauf dieser Arbeit “KimCNN” genannt wird. Zu dem Zeitpunkt konnte KimCNN in Kombination mit Word Embeddings 4 von 7 State-of-the-art Textklassifikationsmodelle übertreffen, darunter auch ein Sentiment Analysis Datensatz (Kim, 2014, S. 1746).

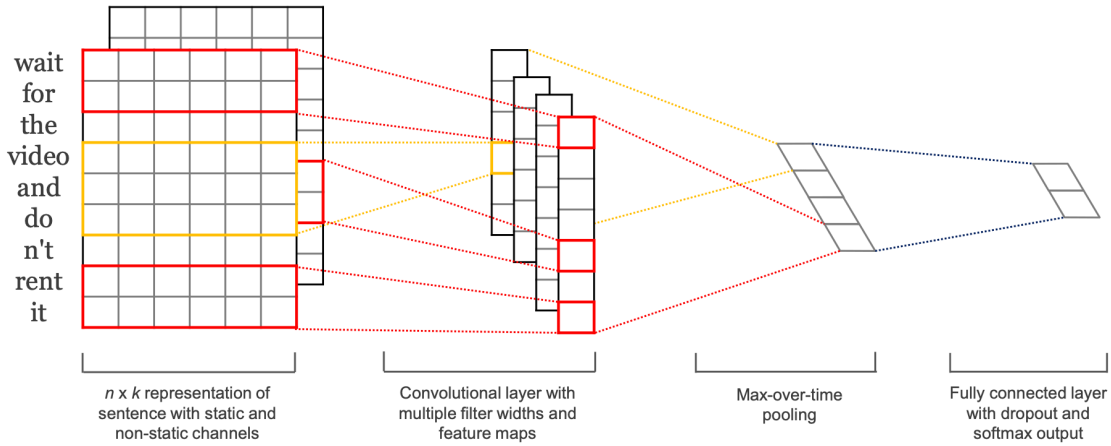

Abbildung 2. Die Grafik wurde aus Kims Aufsatz "Convolutional Neural Networks for Sentence Classification" entnommen und zeigt die grobe Architektur von KimCNN (Kim, 2014, S. 1747)

TODO

```
KimCNN(
  (word_emb): Embedding(44090, 300, padding_idx=1)
  (conv_3): Conv1d(1, 100, kernel_size=(900,), stride=(300,))
  (conv_4): Conv1d(1, 100, kernel_size=(1200,), stride=(300,))
  (conv_5): Conv1d(1, 100, kernel_size=(1500,), stride=(300,))
  (fc): Linear(in_features=300, out_features=5, bias=True)
)
```

TODO weiter: - Blog: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/ - CNN für Text erklären - Warum CNN? https://machinelearningmastery.com/best-practices-document-classification-deep-learning/ - KimCNN erklären - Paper: https://www.aclweb.org/anthology/D14-1181.pdf

[ ]:

5 Schlussbetrachtung

TODO

6 Literaturverzeichnis

BOJANOWSKI, Piotr, GRAVE, Edouard, JOULIN, Armand, MIKOLOV, Tomas, "Enriching Word Vectors with Subword Information", in: Transactions of the Association for Computational Linguistics, Bd. 5, Juli 2016, S. 135-146.

CHOLLET, Francois, Deep learning with python, 2018.

DEVLIN, Jacob, CHANG, Ming-Wei, LEE, Kenton, TOUTANOVA, Kristin, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", in: Proceedings of the NAACL-HLT Conference, S. 4171–4186.

GéRON, Aurélien, Praxiseinstieg Machine Learning mit Scikit-Learn, Keras und TensorFlow, übers. v. Kristian Rother und Thomas Demmig, ^{2}2020.

KARN, Ujjwal, An Intuitive Explanation of Convolutional Neural Networks, https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/ (abgerufen am 20.08.2020).

KIM, Yoon, Convolutional Neural Networks for Sentence Classification, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (Oktober 2014), S. 1746-1751.

JACOVI, Alon, SHALOM, Oren Sar, GOLDBERG, Yoav, Understanding Convolutional Neural Networks for Text Classification, in: Proceedings of the 2018 EMNLP Workshop BlackboxNLP. Analyzing and Interpreting Neural Networks for NLP (Januar 2018), S. 56-65.

LIU, Bing, Sentiment analysis. Mining opinions, sentiments, and emotions, Cambridge 2015.

PENNIGTON, Jeffrey, SOCHER, Richard, MANNING, Christopher D., "GloVe: Global Vectors for Word Representation", in: EMNLP (Januar 2014), S. 1532-1533.

PETROLITO, Ruggero, DELL'ORLETTA, Felice, Word Embeddings in Sentiment Analysis, in: Proceedings of the Fifth Italian Conference on Computational Linguistics CLiC-it (Januar 2018), S. 330-334.

PILEHVAR, Mohammad Taher, CAMACHO-COLLADOS, Jose, "Embeddings in Natural Language Processing. Theory and Advances in Vector Representation of Meaning", 2020.

WEIDMAN, Seth, Deep Learning. Grundlagen und Implementierung, übers. v. Jorgen W. Lang, 2020.

7 BibText

%BOJANOWSKI 2016

@article{bojanowski2016, author = {Bojanowski, Piotr and Grave, Edouard and Joulin, Armand and Mikolov, Tomas}, year = {2016}, month = {07}, pages = {135-146}, title = {Enriching Word Vectors with Subword Information}, volume = {5}, journal = {Transactions of the Association for Computational Linguistics}, doi = {10.1162/tacl_a_00051} }

%CHOLLET 2018

@book{chollet2018, author = {Chollet, Francois}, title = {Deep learning with Python}, year = {2018} }

%DEVLIN 2018

@article{devlin2018, author = {{Devlin}, Jacob and {Chang}, Ming-Wei and {Lee}, Kenton and {Toutanova}, Kristin}, title = {BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding}, journal = {Proceedings of the NAACL-HLT Conference}, year = {2019}, pages = {4171-4186} }

%GERON 2020

@book{geron2020, author = {Géron, Aurélien}, title = {Praxiseinstieg Machine Learning mit Scikit-Learn, Keras und TensorFlow}, year = {2020}, edition = {2}, translator = {Rother, Kristian and Demmig, Thomas} }

%KARN 2016

@online{karn2016, title = {An Intuitive Explanation of Convolutional Neural Networks}, year = {2016}, url = {https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/}, urldate = {2020-08-20} }

%KIM 2014

@inproceedings{kim2014, title = {Convolutional Neural Networks for Sentence Classification}, author = {Kim, Yoon}, booktitle = {Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing ({EMNLP})}, month = {10}, year = {2014}, pages = {1746-1751}, }

%JACOVI 2018

@article{jacovi2018, author = {Jacovi, Alon and Shalom, Oren Sar and Goldberg, Yoav}, title = {Understanding Convolutional Neural Networks for Text Classification}, journal = {Proceedings of the 2018 EMNLP Workshop BlackboxNLP. Analyzing and Interpreting Neural Networks for NLP}, month = {01}, pages = {56-65}, year = {2018}, }

%LIU 2015

@book{liu2015, author = {Liu, Bing}, title = {Sentiment analysis. Mining opinions, sentiments, and emotions}, year = {2015} }

%PENNIGTION 2014

@inproceedings{pennington2014, author = {Pennington, Jeffrey and Socher, Richard and Manning, Christoper}, year = {2014}, month = {01}, pages = {1532-1543}, title = {Glove: Global Vectors for Word Representation}, volume = {14}, journal = {EMNLP}, doi = {10.3115/v1/D14-1162} }

%PETROLITO 2018

@article{petrolito2018, author = {Petrolito, Ruggero, and Dell'Orletta, Felice}, title = {Word Embeddings in Sentiment Analysis}, journal = {Proceedings of the Fifth Italian Conference on Computational Linguistics CLiC-it}, month = {01}, year = {2018}, pages = {330-334} }

%PILEHVAR 2020

@book{pilehvar2020, author = {Pilehvar, Mohammad Taher and Camacho-Collados, Jose}, title = {Embeddings in Natural Language Processing. Theory and Advances in Vector Representation of Meaning} year = {2020} }

%WEIDMAN 2020

@book{weidman2020, author = {Weidman, Seth}, title = {Deep Learning. Grundlagen und Implementierung}, translator = {Lang, Jorgen W.}, year = {2020} }

[ ]: