Transferaufgabe - Sentiment Analysis mit Word Embeddings

August 25, 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?

[2]:
```
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 geben. In früheren Jahren wurden dafür zu Unterstützung der Textklassifikationstechniken sogenannte "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 wurden eine Sentiment Analysis auf ein Korpus von Nutzerreviews des Onlineversandhändlers **Amazon** in der Produktkategorie "Elektronik" durchgeführt. Für die Sentiment Analysis wurden mehrere Klassifizierungstechniken verwendet. Die zentrale Technik war ein Convolutional Neural Network (CNN), welches in Kombination mit vortrainierten Word Embeddings verwendet wurde. Als CNN-Architektur wurde das CNN von Yoon Kim verwendet, welches mit einer einfachen Neuronalen Netz Architektur seinerzeit mehrere State-of-the-art Klassifizierungsansätze übertreffen konnte, darunter auch einen Sentiment Analysis Datensatz (Kim, 2014). Kim verwendete in seinen Experimenten die Word2Vec Embeddings, in dieser Arbeit wurden stattdessen die neueren **GloVe** und **FastText** Embeddings verwendet. Während Kim nur ein votrainiertes Embedding verwendete, sollten in dieser Arbeit insgesamt fünf (TODO?) verschiedene Embeddings in Kombination mit dem CNN von Kim verwendet werden, zwei FastText und drei GloVe Embeddings. Es sollte untersucht werden, welche der fünf Embeddings sich am besten für die Sentiment Analysis des Nutzerreview-Korpus eigneten. Diese Ergebnisse wurden mit zwei weiteren Ansätzen verglichen: einer SVM-Klassifikation und einem *fine-tuned* BERT-Modell. Anders als CNN sind Support Vector Machines (SVM) keine neuronalen Netze, sondern gehören zu den maschinellen Lernverfahren und werden bereits seit Jahrzehnten für die Textklassifikation verwendet. Sie nutzen jedoch keine vortrainierten Gewichte von Word Embeddings, sondern lernen "from scratch", d.h. alle Informationen, die zur Klassifizierung verwendet werden, werden lediglich aus den Trainingsdaten gewonnen. Es sollte deshalb untersucht werden, ob die Nutzung von Word Embeddings einen Vorteil für die Sentiment Analysis bieten oder ob bereits ein einfaches Support Vector Verfahren ohne vortrainierte Gewichte ähnliche Resultate wie das CNN inklusive Word Embeddings erreichen konnte. Zuletzt sollten die Ergebnisse der CNNs und der SVM mit einem *fine-tuned* BERT-Modell verglichen werden. BERT wurde Ende 2018 von Devlin et. al. veröffentlicht und konnte mehrere State-of-the-art Ergebnisse in vielen Aufgaben des Natural Language Processing erzielen (Devlin et. al., 2018). Auch fast zwei Jahre nach seiner Veröffentlichung ist BERT noch sehr populär und wird bei vielen Aufgaben des Natural Language Processing verwendet. Wie das hier verwendete CNN benutzt BERT für die Klassifizierung vortrainierte Gewichte, die es aus eigenen bereitgestell-

ten Embeddings extrahiert werden. Für das gesamte Amazon-Review-Korpus gibt es eine Auflistung der State-of-the-art Verfahren, bei welcher BERT aktuell den ersten Platz belegt.(FN: Siehe https://paperswithcode.com/sota/sentiment-analysis-on-amazon-review-full (abgerufen am 23.08.2020).) An zweiter Stelle des Rankings befindet sich ein CNN. Es sollte deshalb untersucht werden, ob das neuere und komplexere BERT-Modell, welches weitaus hochdimensionalere Embeddings als GloVe und FastText verwendet, bessere Ergebnisse als die simple CNN Architektur von Kim erzielen konnte oder ob dieses oder die SVM für die Sentiment Analysis des Reviewkorpus in der ausgewählten Kategorie "Elektronik" besser geeignet war. Das vorwiegende Unterscheidungskriterium der drei Verfahren war die Klassifizierungsgenauigkeit, weitere Faktoren waren die Dauer des Trainings, die Dauer der Optimierung der Hyperparameter, die benötigte Rechenleistung und die Komplexität der Implementierung (TODO: wirklich?).

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. Zudem wurden die Zusammenfassung und der eigentliche Text eines Reviews zusammengeführt. Das Bewertungssystem wurde für diese Arbeit angepasst und von fünf Sterne auf drei Sterne reduziert. Bewertungssysteme mit fünf Sternen sind problematisch, da diese oft eine bimodale Verteilung hinsichtlich der Extremwertungen 1 und 5 aufweisen. (FN: Siehe http://www.lifewithalacrity.com/2006/08/using_5star_rat.html (abgerufen am 23.08.2020).) Vor allem die 2 und 4 Sterne-Bewertungen lassen sich nicht immer ganz offensichtlich von den 1 und 5 Sternen abgrenzen, da Nutzer für sich selbst individuelle Bewertungsrichtlinien festlegen. Für diese Arbeit wurden deshalb die 2 und 4 Sterne Bewertungen mit den 1 und 5 Sterne Bewertungen zusammengeführt, sodass es nur noch drei Bewertungseinheiten gab (siehe Korpusausschnitt in Tabelle 1): **positiv** (4 und 5 Sterne), **neutral** (3 Sterne) und **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 (Gesamtgröße: 45000 Reviews). Zusätzlich wurde die Spalte "length" hinzugefügt, welche die Länge der Reviews beinhaltet.

	rating	name	review	verified	vote	date	length
34664	positive	leo felix	great deal	True	0	18.01.2018	2
17394	neutral	Benjiboi666	does not work on xbox or ps	True	0	21.01.2018	7
11339	negative	Lost-creek	Stops recording after a few minutes	True	0	06.01.2018	6

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

3 Theoretische Grundlagen

3.1 Word Embeddings

Word Embeddings sind eine besondere Art der distributiven Repräsentation von Wörtern (PILHEVAR 2020, S. 27). Sie 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 können. 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 aktuell 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

3.2.1 Allgemeine Funktionsweise

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 (Chollet, 2016, S. 122). 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 (Géron, 2020, S. 451). 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.

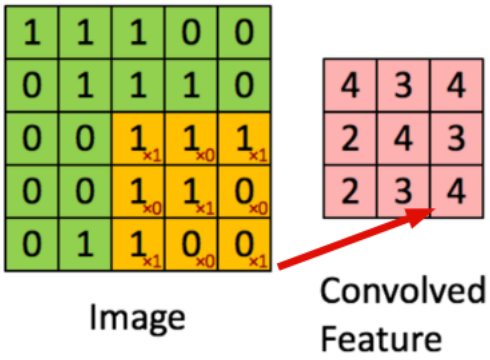

Abbildung 1. Die Abbildung wurde aus folgendem Artikel entnommen: An Intuitive Explanation of Convolutional Neural Networks.

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 (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 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 oft 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 das Netz 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). Laut Goldberg sind CNNs nützlich für Textklassikationsprobleme, bei denen es wichtige, lokale Hinweise in den Daten hinsichtlich der Klassenzugehörigkeit gibt, die jedoch an verschiedenen Stellen eines Dokuments auftauchen (Goldberg, 2015, S. 348). Damit eignen sich CNNs für die Entdeckung von relevanten, positionsunabhängigen Wortfolgen, welches bei der Sentiment Analysis z.B. aussagekräftige Wörter oder Phrasen sein können, die Bedeutungsträger von Meinungen, Stimmungen, Einschätzungen und Emotionen sein können.

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

3.2.2 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 State-of-the-art Ergebnisse im Bereich "Textklassifikation" bei 4 von 7 getesteten Datensätzen erreichen, darunter auch ein Sentiment Analysis Datensatz (Kim, 2014, S. 1746). Das originale KimCNN wurde nur auf Word2Vec Embeddings trainiert (KIM, 2014, S. 1748). In dieser Arbeit wurde das Modell in Kombination mit den Word2Vec-Nachfolgern **GloVe** und **FastText** verwendet. Zudem wurde nur die Version "CNN-non-static" verwendet, bei welcher die vortrainierten Word Embeddings für die entsprechende Aufgabe gefinetuned werden. Bei KimCNN werden die Convolutions (deutsch: Faltungen) wortweise über die Eingabevektoren berechnet, wobei unterschiedlich große Filterfenster (Werte: 3, 4, 5) für die gleichzeitige Verarbeitung von Wörtern verwendet wurden. Die resultierenden Feature-Maps wurden mit einem Max-Pooling-Layer verarbeitet, um die extrahierten Features zu verdichten oder zusammenzufassen. Das Modell sieht wie folgt aus:

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)
)
```

KimCNN verwendet folgende Parameter, die mithilfe von GridSearch für alle von Kim getesteten Datensätze die besten Ergebnisse lieferten:

- Aktivierunsfunktion: ReLU
- Filtergrößen: 3, 4, 5
- Anzahl der Filter: 100
- Dropoutrate: 0.5
- Batchgröße: 50
- Optimierungsverfahren: Adadelta

4 Experimente

4.1 Aufbau

Für die Experimente mit KimCNN wurde Stoppwörter beibehalten, um die Größe der im Durchschnitt sehr kurzen Reviews (~49 Tokens) nicht noch weiter zu verringern. Alle Wörter wurden durch kleingeschriebene Wörter dargestellt. Das Korpus wurde für die Experimente in einen Trainings-, Validierungs- und Testdatensatz aufgeteilt. Die Aufteilung erfolgte nach dem Pareto-Prinzip, d.h. 80% der Daten wurden als Trainingsdaten verwendet und jeweils 10% für die Validierungs- und Testdaten. Dies wurde dreimal durchgeführt, sodass eine dreifache Kreuzvalidierung durchgeführt werden konnte. Die Genauigkeit wurde bei jeder Kreuzvalidierung für den Testdatensatz berechnet und dann gemittelt. 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 eine Reihe von vortrainierten Embeddings verwendet (siehe Tabelle 2). Für jedes Embedding (außer den BERT Embeddings) wurde in Kombination mit KimCNN eine Hyperparameteroptimierung durchgeführt, bei der folgende Parameter und Werte optimiert wurden: - Batchgröße: 50 - Lernrate: 0.01, 0.001 - Maximale Anzahl an Features: 25000, 50000 - Maximale Anzahl an Epochen: 500

Die Batchgröße wurde einmal auf 50 gestellt, da dies bei den Experimenten von Kim die besten Ergebnisse lieferte (Kim, 2014, S. 1748). Die maximale Anzahl an Epochen wurde auf 500 gestellt, jedoch wurde Early Stopping mit einer Patience von 3 eingebaut, sodass das Netz aufhörte zu trainieren, sobald sich der Validierungs Loss drei Epochen lang nicht verringerte.

Embedding	Modell	Details	Link
BERT	bert-large-uncased	24-layer, 1024-hidden, 16-heads, 340 Millionen Parameter. Wurde auf kleingeschriebenen englischen Texten trainiert.	https://huggingface.co/tran (abgerufen am 23.08.2020).

Embedding	Modell	Details	Link
FastText	fasttext.en.300d	2 Million Wortvektoren wurden auf einem Wiki-Korpus von 2017, dem UMBC Korpus und dem statmt.org Nachrichtenkorpus erstellt. Die Sprache ist Englisch.	https://fasttext.cc/docs/en/vectors.html (abgerufen am 23.08.2020).
FastText	fasttext.simple.300d	1 Million Wortvektoren wurden auf einem Wiki-Korpus von 2017, dem UMBC Korpus und dem statmt.org Nachrichtenkorpus erstellt. Die Sprache ist einfaches Englisch.	https://fasttext.cc/docs/en/vectors.html (abgerufen am 23.08.2020).
GloVe	glove.6B.300d	300d Embeddings wurden auf einem Wiki-Korpus von 2014 und dem Gigaword Korpus trainiert. Es beinhaltet 6 Milliarden Tokens und ein Vokabular von 400000 Tokens.	https://nlp.stanford.edu/pro (abgerufen am 23.08.2020)
GloVe	glove.840B.300d	300d Embeddings wurden auf dem Common Crawl Korpus trainiert. Es beinhaltet 840 Milliarden Tokens und ein Vokabular von 2,2 Millionen Tokens.	https://nlp.stanford.edu/pro (abgerufen am 23.08.2020)
GloVe	glove.twitter.27B.200d	200d Embeddings wurden auf Tweets von Twitter trainiert. Es beinhaltet 2 Milliarden Tweetw, 27 Milliarden Tokens und ein Vokabular von 1,2 Millionen Tokens.	https://nlp.stanford.edu/pro (abgerufen am 23.08.2020)

Tabelle 2: Auflistung aller verwendeten Embeddings.

TODO: BERT

4.2 Ergebnisse

TODO: - Tabelle - interpretieren

TODO: ausfüllen

Embedding	Genauigkeit	Besten Parameter
bert-large-uncased		...
fasttext.en.300d		Lernrate: 0.01, 0.001Maximale Anzahl an Features: 25000, 50000
fasttext.simple.300d		Lernrate: 0.01, 0.001Maximale Anzahl an Features: 25000, 50000
glove.6B.300d		Lernrate: 0.01, 0.001Maximale Anzahl an Features: 25000, 50000
glove.840B.300d		Lernrate: 0.01, 0.001Maximale Anzahl an Features: 25000, 50000
glove.twitter.27B.200d		Lernrate: 0.01, 0.001Maximale Anzahl an Features: 25000, 50000

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.

GOLDBERG, Yoav, A Primer on Neural Network Models for Natural Language Processing, in: Journal Of Artificial Intelligence Research, Bd. 57 (2016), S. 345-420.

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} }

%GOLDBERG 2015

@article{goldberg 2016, author = {Yoav Goldberg}, title = {A Primer on Neural Network Models for Natural Language Processing}, journal = {Journal Of Artificial Intelligence Research}, volume = {57}, year = {2016}, pages = {345-420} }

%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} }

[ ]: