

# Amazon Review Klassifizierung

## Projektbericht NLP

	Name	Martikel-Nr:	eMail-Präfix
Verfasser:	Thorsten Hilbradt	5034067	s182316
	Andreas Bernrieder	7876007	s180259
Git:	<a href="https://github.com/Phantomias3782/AmazonReviews">https://github.com/Phantomias3782/AmazonReviews</a>		
Kurs:	WWI18DSB		
Semester:	WS 2020/21		

# Inhaltsverzeichnis

<b>Inhaltsverzeichnis</b>	<b>2</b>
<b>Einleitung</b>	<b>3</b>
<b>Datenquellen</b>	<b>4</b>
<b>Datentransformation</b>	<b>5</b>
<b>Theorie des Lösungsansatzes</b>	<b>6</b>
<b>Eingesetzter Algorithmus</b>	<b>7</b>
K-Nearest Neighbors	7
Neuronales Feed Forward Netzwerk	7
<b>Implementierungsdetails</b>	<b>9</b>
KNN	9
NN	9
<b>Evaluierung der Umsetzung</b>	<b>11</b>
<b>Literaturverzeichnis</b>	<b>14</b>

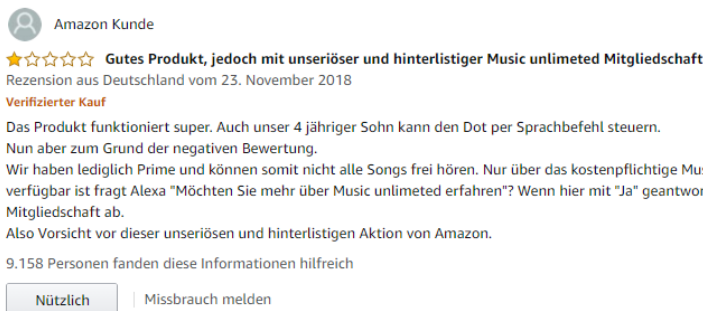
# Einleitung

Fürs das abschließende Projekt des Natural Language Processing Kurs im Wintersemester 2020/21 an der DHBW Mannheim haben wir, Thorsten Hilbradt und Andreas Bernrieder, uns dafür entschieden eine Sentiment Analyse von Amazon Reviews durchzuführen.

Das Ziel dieser Ausarbeitung ist es Reviews, welche von Kunden bei verschiedenen Produkten hinterlassen wurden, so zu klassifizieren, dass eine automatische Einordnung in die verschiedenen Kategorien, von ein Stern bis fünf Sterne, möglich ist. In diesem Prozess werden zwei verschieden Algorithmen und deren Ergebnisse verglichen und ausgewertet. Es wurde sich hier einmal für die Anwendung von einem K-Nearest Neighbour (KNN) Algorithmus entschieden und als zweites für ein Neuronales Netzwerk (NN).

Der wirtschaftliche Nutzen kann dabei wie folgt festgehalten werden.

Wie das [unten stehende Review](#) zeigt sind Kunden häufig eigentlich zufrieden mit einem Produkt, geben aber aufgrund anderer Faktoren (Lieferung, Unzufriedenheit mit einem Teilaspekt) direkt eine 1 Sterne Bewertung.



Amazon Kunde

☆☆☆☆ Gutes Produkt, jedoch mit unseriöser und hinterlistiger Music unlimited Mitgliedschaft

Rezension aus Deutschland vom 23. November 2018

Verifizierter Kauf

Das Produkt funktioniert super. Auch unser 4 jähriger Sohn kann den Dot per Sprachbefehl steuern.  
Nun aber zum Grund der negativen Bewertung.  
Wir haben lediglich Prime und können somit nicht alle Songs frei hören. Nur über das kostenpflichtige Music unlimited sind mehr Songs verfügbar. Wenn ein Song nicht frei verfügbar ist fragt Alexa "Möchten Sie mehr über Music unlimited erfahren"? Wenn hier mit "Ja" geantwortet wird erfährt man nicht nur mehr sondern schließt automatisch die Mitgliedschaft ab.  
Also Vorsicht vor dieser unseriösen und hinterlistigen Aktion von Amazon.

9.158 Personen fanden diese Informationen hilfreich

Nützlich | Missbrauch melden

Dies macht es für andere interessierte Kunden schwerer auf die Sternebewertungen zu vertrauen, was wiederum zu einem niedrigeren Kaufinteresse bei dem Produkt führen kann. Mit unserem Algorithmus soll es möglich sein eine objektive Sternebewertung auf Basis der textuellen Eingabe des Users zu geben.

Des Weiteren ist diese Funktion für die User eine Komfortfunktion. Die meisten nutzer verzichten aufgrund des Aufwandes auf das Bewerten von Produkten, aber wenn ihnen diese Aufgabe teilweise abgenommen wird, oder weil sie sich dafür interessieren wie das neue System funktioniert, könnten mehr Nutzer Wertungen vergeben, was wiederum zu höheren Verkaufsmöglichkeiten für die Händler führt.

# Datenquellen

Im Folgenden soll der Prozess der Datenauswahl dargelegt werden.

Im Zuge der Recherche nach einem passendem Thema für die Projektarbeit im Zuge des Natural Language Processing Kurses wurde auf kaggle ein Datensatz zur Sentiment Analyse von Amazon Review Daten gefunden ([siehe hier](#)).

Bei näherer Betrachtung der Daten wurden jedoch mehrere Hemmnisse auffällig. Die Daten liegen in einem, zumindest vordergründig, nicht schlüssigen Format vor, nämlich in einer txt Datei mit einem Eintrag pro Zeile im Format:

```
__label__[ReviewsScore] [ReviewText]
```

Es wurde als aufwändig erachtet die Daten in diesem Format einzulesen, weswegen ein weiterer Datensatz herangezogen wurde. Dieser wurde von Ni et al. [1] unter [dieser Domain](#) bereitgestellt und bietet im Vergleich mehrere Vorteile. Der Zugang zum Datensatz ist durch eine Anmeldung per WebForm möglich.

Der Datensatz besteht in seiner vollen Form aus etwa 233 Millionen Amazon Reviews aus verschiedenen Produktkategorien. Zusätzlich wird die Möglichkeit gegeben bei Bedarf auf Metadaten zurückzugreifen, die Produktbeschreibungen, Preise, etc. umfassen. Auf diese wird jedoch für dieses Projekt verzichtet.

Es wird jedoch auch die Möglichkeit geboten nur die Reviews einer speziellen Kategorie herunterzuladen, wodurch es möglich ist zu evaluieren, ob es Kategoriespezifische Wertungen gibt, also eine Textbewertung verschiedene Sterne in verschiedenen Kategorien zur Folge hat. Die auf die Kategorien aufgeteilten Reviews liefern dabei einen Datenumfang von ~150.000 Datenpunkten (Giftcards), bis hin zu ~21 Millionen Datenpunkten (Home and Kitchen).

Die Website bietet ferner die Möglichkeit noch kleinere Subsamples der Kategorien zum Experimentieren herunterzuladen, die nur wenige tausend Datenpunkte enthalten. Da festgestellt wurde, dass ein großer Datensatz deutlich schlechtere Ergebnisse liefert als mehrere kleine, wurde auf 6 kleinere Datensätze mit insgesamt knapp 29.000 Datenpunkten gearbeitet.

Die Daten liegen in jedem Fall in einem JSON Format vor. beispielsweise:

```
{"overall": 5.0, "verified": true, "reviewTime": "09 4, 2015", "reviewerID": "ALJ66O1Y6SLHA",  
"asin": "B000K2PJ4K", "style": {"Size": " Big Boys", "Color": " Blue/Orange"},  
"reviewerName": "Tonya B.", "reviewText": "Great product and price!", "summary": "Five  
Stars", "unixReviewTime": 1441324800} (aus Amazon Fashion Subset).
```

# Datentransformation

Für dieses Projekt werden die im JSON Format gespeicherten Daten via Pandas eingelesen. Im Folgenden sollen die durchgeführten Schritte der Datentransformation dargelegt werden. Die einzelnen Schritte können im beigefügten Jupyter Notebook "AmazonReviews\_SentimentAnalysis" nachvollzogen werden.

## 1. Reduzierung

Wie im vorangegangenen Kapitel beschrieben weisen die Daten 12 Spalten auf, wovon nur die Spalten "overall" und "reviewText" behalten werden. "overall" enthält dabei die Sternebewertung des Users als numerische Zahl zwischen 1 und 5, während in "reviewText" ein String mit der textuellen Bewertung des Users abgelegt ist.

Es werden zudem Reihen mit null Werten verworfen, welche beim verwendeten Subset jedoch nur einen geringen Anteil ausmachen (29 Reviews).

## 2. Balancing

Darauffolgend wurde eine nähere Betrachtung der Daten, bzw. ihrer Verteilung vorgenommen. Dabei wurde das Ergebnis gezogen, dass (zumindest) beim Fashion Subset ein deutlicher Überhang der 5 Sterne Reviews vorliegt. So befinden sich im Datensatz mehr als 15 mal so viele 5 Sterne Bewertungen, wie 2 Sterne Bewertungen. Ein solcher Überhang einer einzelnen Klasse führt häufig zu schlechteren Modellen, da diese schlechter generalisieren können und im Zweifel die übergewichtete Klasse ausgeben. Aus diesem Grund wird ein ausbalancierter Datensatz kreiert, der von jeder Klasse die selbe Anzahl an Datenpunkten enthält, also beim verwendeten Subset 1018 Datenpunkte pro Klasse.

## 3. Wort Vektoren

Im nächsten Schritt werden aus den Review Texten Wort Vektoren erstellt, um diese Computer verarbeitbar zu machen. Dafür wird auf Funktionen zurückgegriffen, die für die erste Hausaufgabe in diesem Kurs entwickelt wurden. Diese wären die Umwandlung von den Eingabestrings, hier also den Reviews in Wort Vektoren anhand von "term frequency" und "inverse document frequency". [2]

## 4. Train Test Split

Der nun gewonnene Datensatz wird nun in zwei unabhängige Datensätze geteilt, wobei darauf geachtet wird, dass beide Datensätze dieselbe Datenverteilung haben wie der ursprüngliche Datensatz. Die Aufteilung erfolgt in einem 80:20 Verhältnis, wobei in einem späteren Schritt erneut 5 % des Trainingsdatensatzes abgegriffen werden, um einen Validation Datensatz zu bilden.

# Theorie des Lösungsansatzes

Das Projekt zielt darauf ab anhand der UserReviews die jeweils zugehörige Sterne Bewertung zu predicten. Diese Prediction kann auf unterschiedliche Weise erfolgen.

Für dieses Projekt wurde sich entschieden neue UserReviews auf Basis der bisher vorhandenen zu bewerten. Dazu arbeitet der erste Ansatz mit dem Hintergrund der Ähnlichkeit. Es wird angenommen, dass Wort Vektoren, die einander ähnlich sind eine ähnliche Sternebewertung zur Folge haben. Beispielsweise seien hier "Sehr tolles Produkt. Würde es immer wieder kaufen." und "Tolles Produkt. Kaufen kann ich empfehlen" als Rezensionen genannt. Es ist erkennbar, dass beide mit dem Produkt zufrieden sind, und auch Schlüsselworte wie "Tolles" kommen in beiden Strings vor, in diesem Fall sogar in derselben Gewichtung. Für die Ähnlichkeit wird also davon ausgegangen, dass die aus diesen Strings erzeugten Wort Vektoren in einem (hier 7, aber allgemein)  $n$  Dimensionalen grob in dieselbe Richtung zeigen. Der Abstand zwischen den beiden ist also geringer als zu anderen Vektoren, wie z.B. "Ich hasse es. Wurde direkt zurück geschickt.". Mit diesem Prinzip sollen auch neue Vektoren in den  $n$ -Dimensionalen Raum gelegt werden, worauf der Abstand zu allen anderen Vektoren berechnet wird. Die Klasse wird dann anhand einer gewissen Anzahl von nahen Vektoren bestimmt.

Der zweite verwendete Ansatz ist der eines Neuronalen Netzes, auch hier dienen die Wort Vektoren als Input. Mithilfe verschiedener Layer des Netzwerkes sollen die einzelnen Bestandteile des Vektors gewichtet werden, sodass am Ende die Output Neuronen im richtigen Maß gewichtet werden für die bestimmten Klassen.

[Neuronales Netzwerk noch etwas Abstrakt beschreiben.]

# Eingesetzter Algorithmus

## K-Nearest Neighbors

k-nearest Neighbors (knn) ist eine Standardimplementierung des oben beschriebenen ersten Ansatzes.

Es handelt sich bei knn um einen sogenannten lazy learner, da ein knn Algorithmus nicht auf vorhandenen Daten vor trainiert wird um im späteren Produktivbetrieb ressourcenschonend und schnell eingesetzt zu werden. Vielmehr wird beim knn jedes mal der komplette Algorithmus mit all seinen Berechnungen ausgeführt, was zur Folge hat, dass für eine gute Performance einerseits der Datensatz nicht unendlich groß sein darf, andererseits der User eine ausreichende Rechenleistung vorweisen kann.

Wie bereits im vorigen Kapitel beschrieben, funktioniert der knn auf Basis von Ähnlichkeiten. Für jeden neuen Datenpunkt werden die Abstände zu allen bisherigen Datenpunkten berechnet. Für diese Distanzen gibt es verschiedene Metriken, die am weitesten verbreitete ist dabei die Euklidische Distanz. Anhand der nun erzeugten Entfernungen zu den Punkten werden nun nur noch die k nächsten Punkte betrachtet, bzw. ihre jeweilige Klasse. Dem neuen Datenpunkt wird nun die Klasse zugesprochen die bei den k nächsten Nachbarn am häufigsten auftritt. Verfeinerungen dieses Algorithmus' wären eine weitere Gewichtung der Klassen nach der Entfernung zum Datenpunkt, also dass z.B. die Klasse 1 schwerer wiegt als zweimal die Klasse 4, wenn der 1-er Datenpunkt bedeutend näher liegt.

## Neuronales Feed Forward Netzwerk

Das neuronale Feed Forward Netzwerk ist eine Variation des Neuronalen Netzes und ist ein Supervised Learning Algorithmus. Hier geht es im wesentlichen darum, dass mit einer Funktion  $y$  die werte der input parameter  $\Theta$  approximiert werden oder grob gesagt wollen wir den output von dem Input vorhersagen.[4]

Das Netzwerk besteht aus einem Input layer, einem Outputlayer und verschiedene neuronenschichten dazwischen. Die Schichten selber sind mit der jeweils nächsten Schicht voll verknüpft. Das bedeutet, dass z.B. jeder Input im Input-Layer mit jedem Neuron der ersten Schicht verbunden ist, so dann auch der erste neuron layer mit dem zweiten neuron layer usw..

Am Ende ist der letzte neuron layer voll verknüpft mit jedem Output im Output-Layer.

Dabei ist jede verbindung zwischen den einzelnen Neuronen gewichtet. Durch das Updaten dieser Gewichtungen "lernt" das NN.[5]

Das bedeutet, dass diese in der Berechnung mit dem Input einen bestimmten Wert erhalten welche dann einen vorher bestimmten Schwellwert überschreitet oder unterschreitet um das Neuron zu aktivieren oder nicht zu aktivieren.

Wenn das Neuron Aktiviert wird bedeutet das der errechnete wert des aus der Gewichtung und dem Input-Wert mit der Gewichtung zum nächsten Neuron oder Outpulayer verrechnet wird.



# Implementierungsdetails

## KNN

Für den knn Algorithmus wurde weitestgehend auf die Verwendung externer Bibliotheken verzichtet. Er wurde also zum Größten teil selbst implementiert. So werden zunächst die einzelnen Vektoren des Trainings Datensatzes aus dem pandas Dataframe in einer list gespeichert. Daran anschließend wird mittels einer for Schleife für jeden Vektor der Abstand zum gegebenen neuen Datenpunkt, dem Testdaten Punkt berechnet. Dies erfolgt mithilfe einer scipy.spatial.distance Implementation der Euklidischen Distanz. Dieser Abstand wird in einem Dictionary gespeichert, wobei der Index des Trainingsdaten Punktes der key und der Abstand der value ist. Nach dem Abschluss der for Schleife wird das Dictionary nach den values sortiert, sodass an erster Stelle der geringste, bzw. an letzter Stelle der höchste Abstand zum Test Datenpunkt liegt. Daran anschließend verzweigen sich die Möglichkeiten. Wir haben dem User die Möglichkeit gegeben als k sowohl einen Integer (herkömmlich), als auch eine Liste von verschiedenen k's zu übergeben. Dies eröffnet die Möglichkeit ressourcenschonend verschiedene k's zu testen, da nur einmalig die Abstände berechnet werden müssen, bevor die Class Estimation folgt. Bei einer Liste von k's wird das folgende mittels einer for Schleife ausgeführt. Für ein einzelnes k werden die k ersten Einträge des Distanz Dictionaries ausgelesen. Dabei wird mithilfe des Indexes die zugehörige Klasse zum Datenpunkt ausgelesen. Beim erstmaligen Erkennen einer Klasse wird ein Eintrag in einem Identifikations Dictionary angelegt mit dem Value 1. Beim nächsten Mal wird der Value um eins hochgezählt. Am Ende wird der key mit dem höchsten Value zurückgegeben, also die Klasse, die am häufigsten in den k nächsten Nachbarn vorkommt.

## NN

Für die Implementierung des Neuronalen Netzwerkes wurde auf die Standardimplementierungen / Funktionen von keras zurückgegriffen. [3] Der erste Schritt ist eine weitere Datenvorverarbeitung. keras Modelle benötigen die Trainingsdaten in einem bestimmten Format, sodass zunächst die Daten aus dem pandas Dataframe in numpy arrays übertragen werden. Die Daten werden dabei in einem array of Lists gespeichert wobei jede Liste einen Datenpunkt, also einen Wort Vektor repräsentiert. In diesem Schritt wird auch die shape, also die Länge der Worte Vektoren abgefragt. Außerdem werden die Daten in float umgewandelt. Abschließend wird in diesem Schritt, wie bereits im Kapitel Datentransformation angedeutet, ein Anteil der Trainingsdaten extrahiert und zu

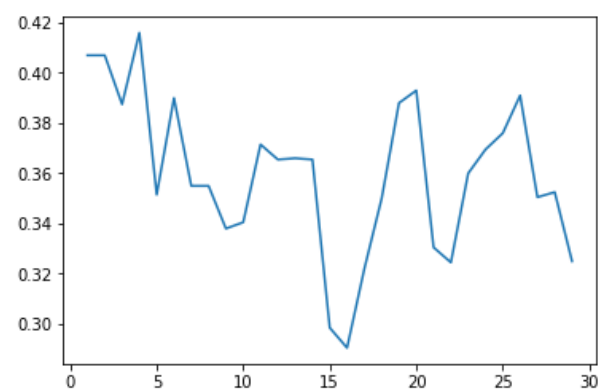
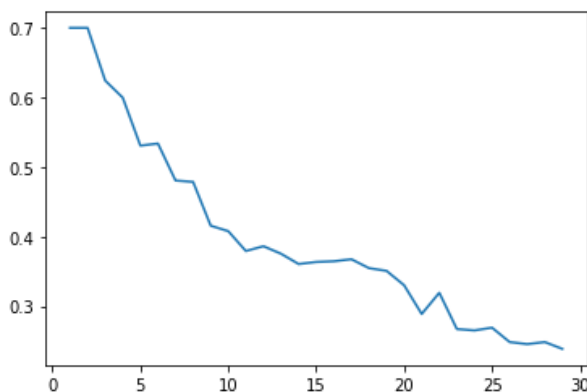
Validierungsdaten. Diese sollen während dem Training des Modells eingesetzt werden, um den Lernfortschritt anzuzeigen. Wir entnehmen standardmäßig 5 Prozent.

Im nächsten Schritt wird das Modell gebaut. Wir greifen dabei auf sogenannte [Dense Layer](#) zurück. Dabei handelt es sich um voll verknüpfte Layer, was bedeutet, dass jedes Neuron einer Schicht mit jedem Neuron der nachfolgenden verknüpft ist. In unserer Implementierung bieten wir dem Nutzer die Möglichkeit die Anzahl an Layern dynamisch zu verändern, dafür müssen Parameter der `build_model` Funktion angepasst werden. Der User bestimmt die Anzahl der Layer mit dem `neuron_layers` Parameter, der standardmäßig 1 ist. Die Parameter `neuron_count` und `activation_functions` erwarten jeweils als Input eine Liste. In dieser soll für jedes vorgesehene Layer die Anzahl der Neuronen, bzw. die Aktivierungsfunktion des jeweiligen Layers gegeben sein.

Nachdem das Model noch in derselben Funktion `compiled` wird kann in der nächsten Funktion das Training beginnen. Der User kann auch hier verschiedene Parameter, wie die Epochen Anzahl anpassen. Als Output erhält er das trainierte Modell, dass dann in weiteren Funktionen getestet, bzw. angewendet werden kann.

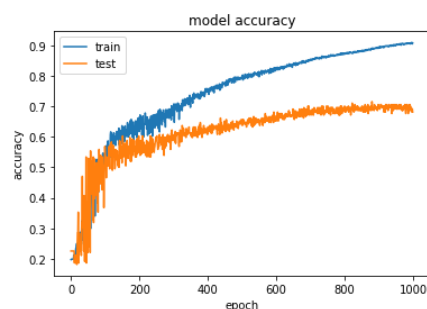
# Evaluierung der Umsetzung

Bei der Implementierung der beiden Algorithmen wurden verschiedene Auffälligkeiten vermerkt. Im allgemeinen wird erwartet, dass mit einem deutlich umfangreicheren Datensatz auch bessere Ergebnisse als mit einem begrenzten Datensatz erreicht wird. Diese Vermutung bestätigte sich in unserem Fall nicht, beim Vergleich des Amazon\_Fashion\_Full Datensatzes (etwa 800.000 Datenpunkte, davon 10.000 im balancierten Datensatz) mit der Kombination mehrerer kleiner Datensätze (etwa 29.000 Datenpunkte, davon etwa 5.000 im



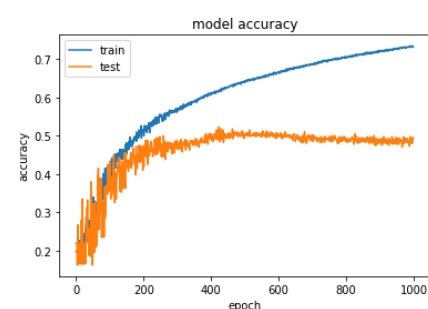
balancierten Datensatz) erreichte der große Datensatz sowohl beim knn, als auch beim Neuronalen Netzwerk bedeutend schlechtere Ergebnisse. Oben stehend ist der Vergleich zwischen den beiden Datensätzen beim knn Algorithmus dargelegt. Der linke Graph zeigt die Performance des kleineres Datensatzes, der rechte die des größeren. Während beim linken bei niedrigen k noch eine akzeptable Performance messbar ist, bewegt sich der

Fit model on training data  
Model Data with, epoch count of 1000,  
second layer = True, neuron number of 10,  
and an sigmoid main activation function



1018/1018 [=====] - 0s 18us/step  
test loss, test acc: [0.6991537325742905, 0.7524557709693909]

Fit model on training data  
Model Data with, epoch count of 1000,  
second layer = True, neuron number of 10,  
and an sigmoid main activation function



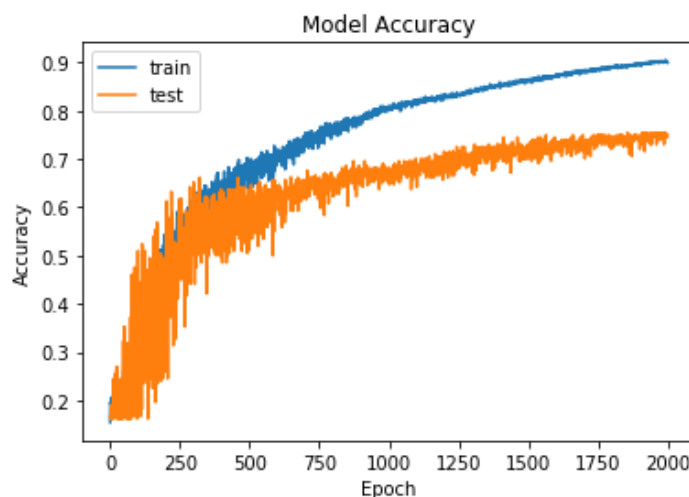
2000/2000 [=====] - 0s 21us/step  
test loss, test acc: [1.2424069805145264, 0.534500002861023]

rechte von Anfang an in einem sehr niedrigen Bereich. Auch beim Neuronalen Netzwerk ist ein deutlicher Qualitätsunterschied bemerkbar. Erneut ist links das Training mit dem kleineren, rechts das Training mit dem größeren Datensatz abgebildet. Hier erreicht der

größere Datensatz eine maximale Performance, die etwas 20 Prozentpunkte unter dem des kleineren liegt.

Aufgrund der vorgebrachten Ergebnisse wurde sich entschieden auf den kleineren Datensatz zurückzugreifen. Ebenso wurde aufgrund der etwas besseren Accuracy, aber hauptsächlich auf Basis der höheren Performance wurde sich dann für das Neuronale Netzwerk als Haupt Algorithmus entschieden. Mittels Ausprobieren wurden verschiedene Konfigurationen des Netzwerks getestet und von den Ergebnissen ausgehen die beste verwendet. Diese hat folgende Konfiguration: 1 Hidden Layer mit 20 Neuronen und einer sigmoid Aktivierungsfunktion, in einem Training mit 2.000 Epochen und einer Batch Größe von 64. Wie unten abgebildet erreicht es damit eine Accuracy von etwa 77 %.

```
1018/1018 [=====] - 0s 17us/step  
Model has a loss of 0.6665894604619219 and an accuracy of 0.7730844616889954
```



Im Abschlussteil des beigefügten Jupyter Notebooks hat der Nutzer die Möglichkeit einen Input zu verfassen, also eine Rezension. Diese wird dann in einem Wort Vektor umgewandelt und auf das nötige Format für das NN gebracht. Die Ausgabe ist dann die berechnete Klasse (Sterne Bewertung). In der nachfolgenden Grafik sind verschiedene Beispieltexte getestet worden. Es wurde ersichtlich dass Sätze die Ausdrücke wie "Terrible", oder "Hate" enthalten zu einer 1 Sterne Bewertung führen, während positive Worte, wie "Great" und buy again" augenscheinlich zu guten Bewertungen (5 Sterne folgen). In diesem Punkt sei auf die letzte Zelle des abgebildeten Notebooks verwiesen, hierfür wurde auf Amazon US eine Produktbewertung einer Kamera eingegeben. Es sei darauf verwiesen, dass sich diese Produktkategorie nicht unter den trainierten befindet, des NN also mit einer neuen Kategorie und komplett [neuen Review](#) konfrontiert wird. Dennoch sagt es die 4 Sterne Bewertung richtig voraus.

Hiermit wurde also bewiesen, dass unser entwickelter Algorithmus eingesetzt werden kann, um neue Reviews automatisch zu klassifizieren und so dem User Arbeit abzunehmen, bzw. bessere und verlässlichere Scores den Händlern zu bieten.

|| 🔍 | 📄 | 📁 | ⬆️ | ⬇️ | ▶️ Run | 🔄 | ▶️ | Code | 📄 |

```
In [51]: # get user input
input_string = input("Please enter your Product review: ")

# process string to word vector
word_vec = create_input_word_vector(input_string, idf_dict) # also usable for knn
input_vector = prepare_nn_input(word_vec, shape)

# predict review score
review_score = model_predict(input_vector, model)

# print user output
print(f"Your score is most likly: {review_score}")

Please enter your Product review: Terrible. Hate it
Your score is most likly: 1
```

```
In [22]: # get user input
input_string = input("Please enter your Product review: ")

# process string to word vector
word_vec = create_input_word_vector(input_string, idf_dict) # also usable for knn
input_vector = prepare_nn_input(word_vec, shape)

# predict review score
review_score = model_predict(input_vector, model)

# print user output
print(f"Your score is most likly: {review_score}")

Please enter your Product review: Great Product. love it buy again
Your score is most likly: 5
```

```
In [50]: # get user input
input_string = input("Please enter your Product review: ")

# process string to word vector
word_vec = create_input_word_vector(input_string, idf_dict) # also usable for knn
input_vector = prepare_nn_input(word_vec, shape)

# predict review score
review_score = model_predict(input_vector, model)

# print user output
print(f"Your score is most likly: {review_score}")

Please enter your Product review: quite average. there are okay points but also negative points. ist very good to
get an insight but he bad points are still present
Your score is most likly: 3
```

```
In [29]: # get user input
input_string = """I like to post a lot of stories on Instagram and this camera does help me with that.
Clarity is also very good and very handy as well. I use it with my tripod stand which makes it even
easier for making better videos. I will say battery backup could have been.
better and the overall quality could have been a little improved as well.
But for the price, I think it totally suffices.
"""

# process string to word vector
word_vec = create_input_word_vector(input_string, idf_dict) # also usable for knn
input_vector = prepare_nn_input(word_vec, shape)

# predict review score
review_score = model_predict(input_vector, model)

# print user output
print(f"Your score is most likly: {review_score}")

Your score is most likly: 4
```



Annie



**Nice to have for Vlogging**

Reviewed in the United States on November 22, 2020

Color: Black | **Verified Purchase**

I like to post a lot of stories on Instagram and this camera does help me with that. Clarity is also very good and very handy as well. I use it with my tripod stand which makes it even easier for making better videos. I will say battery backup could have been. better and the overall quality could have been a little improved as well. But for the price, I think it totally suffices.

4 people found this helpful

Helpful

Report abuse

# Literaturverzeichnis

- [1] **Justifying recommendations using distantly-labeled reviews and fined-grained aspects**  
Jianmo Ni, Jiacheng Li, Julian McAuley  
*Empirical Methods in Natural Language Processing (EMNLP)*, 2019
  
- [2] C. D. Manning, P. Raghavan & H. Schütze, Introduction to Information Retrieval, tf-idf, [Scoring, term weighting & the vector space model](#) Cambridge University Press, 2008
  
- [3] [https://www.tensorflow.org/api\\_docs/python/tf/keras/Model](https://www.tensorflow.org/api_docs/python/tf/keras/Model), 26.01.2021
  
- [4] Ian Goodfellow, Yoshua Bengio, & Aaron Courville (2016). *Deep Learning*. MIT Press.
  
- [5] Erb, R.J. Introduction to Backpropagation Neural Network Computation. *Pharm Res* 10, 165–170 (1993).