TECHNISCHE UNIVERSITÄT WIEN

DIPLOMARBEIT

Stock Market Forecasting

mittels klassischen statistischen Verfahren und Text Mining

ausgeführt am

Institut für
Stochastik und Wirtschaftsmathematik
TU Wien

unter der Anleitung von

Dipl.-Ing. Wolfgang Ecker-Lala und
Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Peter Grandits

durch

Sina Wiesinger BSc
Matrikelnummer: 1526074
[redacted]

Wien, am 14. Dezember 2021

Kurzfassung

Die richtige Vorhersage von Aktienkursen ist für viele Personen und Institutionen erstrebenswert. Allerdings ist dieses Ziel, auch wegen der vielen Einflussfaktoren auf die Aktienpreise, nicht einfach zu erreichen. Die richtige Vorhersage hilft frühzeitig auf etwaige Kursschwankungen reagieren zu können und verspricht somit einen monetären Vorteil. Eine Möglichkeit den Kurs der Aktien vorherzusagen ist mittels neuronaler Netze, die aus vergangenen Aktienpreisen bestimmte Muster erkennen und die zukünftigen Aktienpreise vorhersagen sollen. Der Schwerpunkt dieser Arbeit liegt darin dieses Verfahren mit der Methode der Sentiment Analyse zusammenzubringen. Bei der Sentiment Analyse versucht man Texten einen bestimmten Wert zwischen minus eins und eins zuzuordnen, je nachdem ob diese positiv oder negativ sind. Für die Vorhersage der Aktienpreise sollen Tweets über ein Unternehmen das Stimmungsbild der Community widerspiegeln und Artikel aus der New York Times die Meinung von Experten, die die breite Masse beeinflussen. Die Vorhersage der Aktienpreise erfolgt mit zwei verschiedenen Arten von neuronalen Netzwerken: den feedforward neuronalen Netz und das Long Short-Term Memory (LSTM) Netzwerk. Die Modelle werden für drei verschiedene Unternehmen Apple Inc., Tesla Inc. und Biontech SE mit verschiedenen Parametern durchlaufen und die Ergebnisse werden gegenübergestellt. Dabei stellt sich heraus, dass die Güte des Modells maßgeblich von der Datenlage und der Datenhistorie beeinflusst wird. Ist allerdings eine ausreichende Datenlage gegeben, liefert das feedforward neuronale Netzwerk bei diesem Versuchsaufbau die durchschnittlich besseren Ergebnisse.

Abstract

The correct prediction of share prices is desirable for many individuals and institutions. However, this goal is not easy to achieve because of the many factors that influence shares. The correct prediction helps to be able to react early to any price fluctuations and thus promises a monetary advantage. One way of predicting share prices is by the using of neural networks, which are supposed to recognise certain patterns from past share prices and predict future share prices. The focus of this paper is to combine this method with sentiment analysis. Sentiment analysis tries to assign a certain value between minus one and one to texts, depending on whether they are positive or negative. For the prediction of share prices, tweets about a company should reflect the sentiment of the community as a whole and articles from the New York Times should reflect the opinion of experts who influence the masses. The prediction of stock prices is done using two different types of neural networks: the feedforward Neural Network and the Long Short-Term Memory (LSTM) network. The models are run for three different companies Apple Inc, Tesla Inc and Biontech SE with different parameters and the results are compared. It turns out that the quality of the model is significantly influenced by the data situation and the data history. However, if there is sufficient data, the feedforward Neural Network delivers the better results on average in this test set-up.

Danksagung

Ich möchte mich bei vor allem bei meinen Eltern und Großeltern bedanken, die mir das Studium ermöglicht haben. Sie haben mich nicht nur finanziell sondern auch seelisch unterstützt. Außerdem gilt mein Dank meinem Freund, der immer für mich da ist und war. Des weiteren will ich mich bei meiner Lerngruppe für die schönen letzten Jahre bedanken. Ihr habt mich in allen Lagen unterstützt und die Studienzeit zu einer einzigartigen Erfahrung gemacht!
Zu guter Letzt möchte ich mich bei meinen Betreuern Herrn Dipl.-Ing. Wolfgang Ecker-Lala und Herrn Professor Grandits bedanken. Sie sind mir bei Fragen bei der Diplomarbeit immer mit Rat und Tat bei Seite gestanden.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Wien, am 14. Dezember 2021

Sina Wiesinger

Inhaltsverzeichnis

1 Einführung

Die Vorhersage von Aktienpreisen ist durch die verschiedenen Einflussfaktoren, die bei der Preisentwicklung einer Aktie einwirken, seit geraumer Zeit ein wichtiges Forschungsfeld. Zu diesen Einflussflussfaktoren zählen zum Beispiel Politik, die ökonomische Situation aber auch die Psychologie von den Investoren. In den letzten beiden Jahrzehnten entwickelte sich die Idee, die Vorhersage mittels neuronalen Netzwerken durchzuführen. Diese Netzwerke sollen die dynamischen und nicht lineare Beziehungen erkennen. Die Idee ist, dass sich Historie wiederholt und die gelernten Muster wieder angewendet werden können. [14]

Die reinen Aktienpreise reflektieren allerdings nicht unbedingt die Meinung der Allgemeinheit über ein bestimmtes Unternehmen, allerdings kann auch diese einen Einfluss auf die Preise haben. Um auch diese Meinung in das Vorhersagemodell einzubringen, wird auf die Technik der Sentiment Analyse zurückgegriffen. Besonders seitdem Soziale Netzwerke eine immer größeren Stellenwert bekommen, ist es einfach, die Stimmung vieler einzufangen. Die Kombination der historischen Aktienpreise mit der Stimmung der Gesamtbevölkerung soll eine bessere Vorhersage ermöglichen. [28]

In [30] wird gezeigt, dass es eine Korrelation zwischen den Sentiment Werten, die aus Tweets über ein Unternehmen generiert werden und den zugehörigen Änderungen des Aktienpreises gibt. In dieser Arbeit wollen wir nun die Aktienpreise für drei Unternehmen voraussagen. Bei den drei behandelten Unternehmen handel es sich um Apple Inc., Tesla Inc. und Biontech SE. Die Vorhersage geschieht mit zwei verschiedenen Arten von neuronalen Netzwerken, den feedforward Netzwerk und dem LSTM Netzwerk und die Ergebnisse dieser sollen verglichen werden. Für die Vorhersage der Aktienpreise werden die Sentiment Werte von Twitter ermittelt, um die Meinung der Community einzubeziehen. Des Weiteren werden noch die Sentiment Werte von Zeitungsartikel ermittelt, um auch eine Expertenmeinung in das Modell aufzunehmen. Außerdem fließen die vergangenen Aktienpreise in das Modell ein.

Die wichtigsten Terminologien der Finanzwelt und die zwei verschiedenen Arten der Vorhersage von Aktienpreisen - die technische und die fundamentale Analyse - werden in Kapitel 2 erklärt.

In Kaptiel 3 werden die verschieden verwendeten Techniken des Machine Learnings behandelt. Zuerst werden das feedforward neuronale Netzwerk und das Rekurrente neuronale Netzwerk mit ihren Bestandteilen und Hürden erklärt. Danach wird die Sentiment Analyse und ihre besonderen Hindernisse bei der Anwendung bei Sozialen Netzwerken dargelegt.

Der Aufbau des Experiments wird in Kapitel 4 besprochen. Dabei wird zuerst die Beschaffung und Speicherung der Daten behandelt und danach werden die Daten aus den Sozialen Netzwerken und den Zeitungsartikel nach ihrer Stimmung analysiert.

Im vorletzten Kapitel erfolgt der Vergleich der Ergebnisse. Die Modelle werden mit verschiedenen Parametern durchlaufen, um die bestmögliche Konfiguration herauszufinden. Dabei ist ersichtlich, dass eine ausreichende Datenhistorie von zentraler Bedeutung ist. Außerdem zeigt sich, dass die Ergebnisse mit dem feedforward neuronalen Netz bei diesen Versuchen bessere Ergebnisse erzielen als mit dem neuronalen Netzwerk.

2 Vorhersage von Aktienpreisen

In unserer kapitalistischen Welt haben Aktien einen sehr großen Stellenwert, nicht nur auf den ökonomischen sonderen auch auf die soziale Organisation eines Landes. Deswegen beschäftigt die Vorhersage der Aktienpreise seit Jahren Investoren, Analysten und die Wissenschaft. Mithilfe von Modellen für die Prognose, können Entscheidungen nicht nur nach persönlichen Empfinden sondern auch aufgrund von Daten getroffen werden. Das resultiert in einem geringeren Investitionsrisiko. [16]

Zeitreihen von Aktienpreisen können als rauschende, nicht-parametrische, volatile, komplexe, nicht-lineare, dynamische Prozesse aufgefasst werden. Diese schwierigen Eigenschaften machen die Vorhersage von Aktienpreise zu einem zentralen Thema im Finanzbereich. Die richtige Vorhersage ist jedoch von Bedeutung, denn die Preisentwicklung von Aktien hat fundamentale Auswirkungen auf die ökonomische Entwicklung vieler Länder und mit der richtigen Vorhersage kann frühzeitig auf Änderungen reagiert werden. [25]

Aktienmärkte werden von vielen verschiedenen makroökonomischen Faktoren beeinflusst. Der Preis ist also nicht nur von den bisherigen Kursen abhängig, sondern auch davon, wie die Allgemeinheit über das Unternehmen denkt. Im Zeitalter von Social Media kann man leicht das Empfinden über ein Thema oder ein Unternehmen herausfinden, denn die Gedanken werden von den einzelnen Personen selbst veröffentlicht. Diese große Menge an Daten soll nun für die bessere Vorhersage gesammelt und so analysiert werden, sodass man die richtigen Schlüsse auf das zukünftige Verhalten ziehen kann. [29]

Die Vorhersage von Aktienmärkten ist kein einfaches Unterfangen. Wie bereits erwähnt spielen viele verschiedene Einflussfaktoren eine Rolle, wie zum Beispiel die politische oder wirtschaftliche Lage. Allerdings ist die möglichst wahrheitsgetreue Vorhersage auch ein sehr lukratives Geschäft. Werden die Preisentwicklungen richtig vorhergesagt, kann man sehr stark profitieren. [12]

2.1 Begriffserklärungen

Im folgenden werden die wichtigsten Terminologien aus der Finanzwelt erklärt.

- **Aktie:** Bei einer Aktie handelt es sich um einen Anteil an einem Unternehmen. Die

Eigentümer der Aktien, die Geld in die Aktiengesellschaft investiert haben, werden Aktionäre genannt. Die Haftung der Aktionäre ist mit der eingelegten Summe begrenzt. Aktionäre haben dabei ein Recht auf einen Gewinnanteil, die Dividende, die in Prozenten des Grundkapitals oder pro Aktie ausgezahlt wird. Des Weiteren haben Aktionäre auch ein Stimmrecht bei der Jahreshauptversammlung der Aktiengesellschaft, wobei jede Aktie das Recht einer Stimme entspricht. Bei der Hauptversammlung wird über die Verwendung der Gewinne und über die Höhe der Dividende abgestimmt. Außerdem wählt die Hauptversammlung den Aufsichtsrat. [18] Der Vorteil einer Aktie im Gegensatz zu anderen Finanzinstrumenten, wie zum Beispiel Anleihen, ist, dass sie eine höhere Rendite hat, mit dem Nachteil eines höheren Risikos. [25]

- **Aktienmarkt:** Beim Aktienmarkt handelt es sich um einen Finanzmarkt, an dem Unternehmenswertpapiere - die Aktien - zu einem bestimmten Preis gehandelt werden. Der Ort, an dem Aktien gehandlet werden, wird Börse genannt. Der Preis von Aktien wird mithilfe von Angebot und Nachfrage bestimmt. Es steigt also der Preis, wenn die Nachfrage steigt und vice versa. [25] Durch das Angebot- und Nachfrage-Prinzip bewerten die Aktionäre also die Erfolgsaussichten des Unternehmens. Denken sie, dass das Unternehmen gute Zukunftsaussichten hat, wird mehr in das Unternehmen investiert und der Kurs steigt. Prognostizieren die Anleger allerdings eine schwächere Zukunft, sinkt die Nachfrage. Dies hat dann einen sinkenden Aktienkurs zur Folge. [18]

- **Börse:** Bei der Börse handelt es sich um den Ort, an dem Aktien gehandelt werden. Die Aktien kommen von einem Unternehmen mittels eines öffentlichen Erstangebots auf den Markt, danach können sie direkt an der Börse gehandelt werden. Die beiden größten Börsen der Welt sind die New York Stock Exchange (NYSE) und die National Association of Securities Dealers Automated Quotations (NASDAQ). [25] Beide haben ihren Sitz in New York City. Bei der NASDAQ handelt es sich um eine elektronische Börse und ist ein Händlermarkt. Das heißt, dass Personen direkt mit dem eigenen Konto die Finanzinstrumente kaufen und verkaufen können. Im Gegensatz dazu handelt es sich bei der NYSE um einen Auktionsmarkt. Hier ist ein Händler, wie ein Spezialist der NYSE beteiligt, der Gebote entgegennimmt, um für die Liquidität zu sorgen. [22] Um Aktien von einem Unternehmen an einer bestimmten Börse kaufen zu können, müssen die Unternehmen an genau dieser Börse gelistet sein. [25]

- **Aktienindex:** Aktienindizes sind ein statistisches Maß für eine Gruppe von Unternehmen. Genau wie bei einzelnen Unternehmensaktien hat ein Aktienindizes einen Kurs. Beispiele für Aktienindizes sind zum Beispiel der Dow Jones Industiral Average (DJIA) oder der Austrian Trade Index (ATX). Genau wie bei Aktienkursen von einzelnen Unternehmen, haben auch Aktienindizies eigene Kennzahlen, wie der Eröffnungskurs, der Schlusskurs, das durchschnittliche Volumen oder die Anzahl der

gehandelten Aktien. [25]

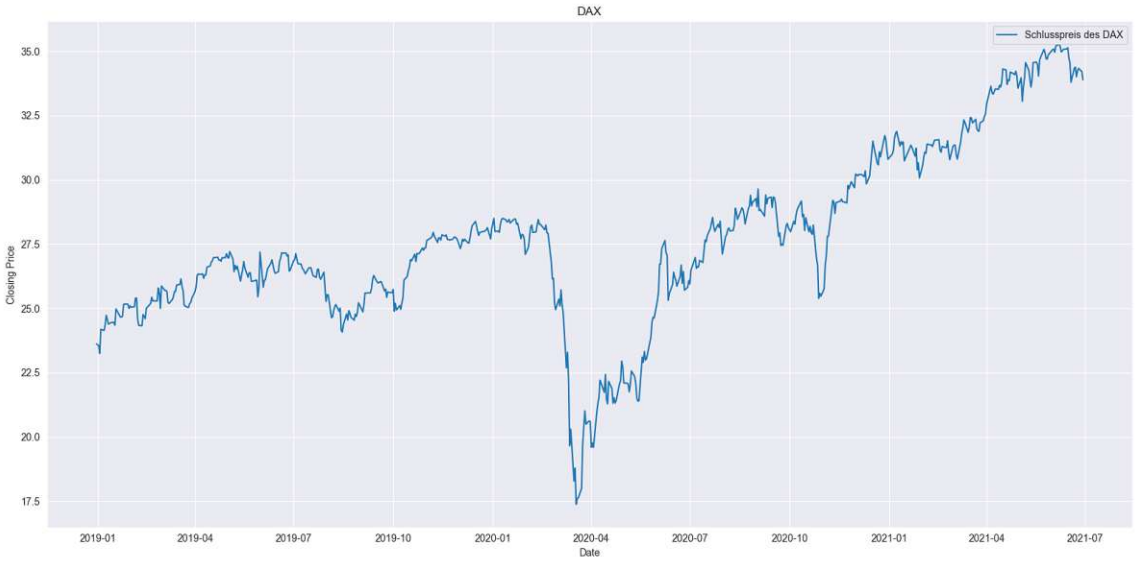


Abbildung 2.1: Schlusspreise des Deutschen Aktienindex (DAX) vom 01.01.2019 bis zum 30.6.2021

- **Vorhersage von Aktienpreisen:** Hier handelt es sich um das Prognostizieren von den zukünftigen Preisen, mithilfe von historischen Daten. Die Vorhersage von Aktienpreise wird meist von Investoren, Analysten und Wissenschaftlern getätigt und wird wegen des höheren Risikos beim Kauf einer Aktie im Gegensatz zu beispielsweise Anleihen durchgeführt, um das Risiko zu minimieren. [25]

2.2 Technische und Fundamentale Analyse von Aktienpreisen

Bei der Vorhersage von Aktienpreisen unterscheidet man zwischen zwei verschiedenen Techniken: die fundamentale Analyse und die technische Analyse. Bei der technischen Analyse werden vor allem historische quantitative Daten analysiert, um die zukünftigen vorherzusagen. Im Gegensatz dazu werden bei der fundamentalen Analyse textuelle Daten studiert, um eine Prognose zu erstellen. [12]

Die am weitesten verbreitete Methode in der Literatur ist die technische Analyse. Hier werden nur Kennzahlen und vergangene Aktienpreise für die Vorhersage hergenommen. Es wird von den Analysten, die diese Methode anwenden, argumentiert, dass die textuellen Daten, wie Zeitungsartikel oder Posts aus sozialen Netzwerke, bereits in den vergangenen Aktienpreisen enthalten sind. Deswegen sollten diese Daten nicht noch einmal zusätzlich ins Modell einbezogen werden. Mithilfe der Kennzahlen wird beurteilt, wie gut oder schlecht es

einem Unternehmen wirtschaftlich geht. Kennzahlen sind für die technische Analyse sehr genau studiert worden, um vorherzusehen, ob ein Kauf oder Verkauf einer Aktie lukrativ ist. Allerdings haben einige Studien, die eine technische Analyse anwenden, nur limitierte Erfolge erzielt. [12] Auch [25] vergleichen verschiedene Studien und Ansätze und kommen zu dem Ergebnis, dass technische Indikatoren eine wichtige Rolle bei der Vorhersage von Aktienpreisen spielen, aber die korrekte Auswahl dieser Indikatoren ein Problem sein kann. Um diese Schwierigkeit zu lösen, erzielt künstliche Intelligenz laut [25] gute Ergebnisse und hier sind hybride Modelle am häufigsten.

Weil der Modellaufbau, der die Schwankungen von Aktien ausreichend gut beschreibt, bei der fundamentalen Analyse komplizierter ist, ist diese Analyse in der Literatur nicht so weit verbreitet. Die meisten Paper verwenden bei der fundamentalen Analyse makroökonomische Zeitreihen. Außerdem werden zum Beispiel Umsätze, Jahres- und Quartalsberichte, Gewinn- und Verlustrechnungen und Bilanzen analysiert. Die zukünftigen Aktienentwicklungen werden durch bereits vergangene Entwicklungen von den vorher aufgezählten Ze abgeleitet, da man auf eine Wiederholung von vergangenen Ereignissen setzt. [25]

Ein weiterer möglicher Bestandteil bei der fundamentalen Analye sind auch Artikel aus fachlichen Zeitschriften. Hier ist der unstrukturierte Aufbau und der Fakt, dass die Veröffentlichung von Artikel nicht stetig geschieht, ein Hindernis. Um dieses Hindernis bewältigen zu können, werden Methoden des Text Minings verwendet. Erst vor Kurzem wurde festgestellt, dass auch Daten aus den sozialen Netzwerken, mit Sentiment Analyse (siehe Kapitel **??**), ein leistungsstarkes Mittel sind, um Aktienpreise vorhersagen zu können. [16] Die fundamentale Analyse wird oft von Langzeitinvestoren benützt. [25] In dieser Arbeit sollen nun die Aktien mithilfe von den vergangenen Aktienpreisen, Zeitungsartikel und Posts aus Sozialen Netzwerken vorhergesagt werden.

3 Machine Learning

Die Fähigkeit, dass Computer selbst aus strukturierten oder unstrukturieren Daten mithilfe eines Alogrithmus ein statistisches Modell aufbauen, wird Machine Learning genannt. Unter Lernen versteht man hier, in den Rohdaten bestimmte Strukturen oder Zusammenhänge zu finden, aufgrund dieser dann zukünftig Daten klassifiziert oder berechnet können. [31] Machine Learning selbst ist ein Teilbereich der künstlichen Intelligenz. [36]

Es werden verschiedene Arten von Machine Learning unterschieden:

1. **Überwachtes Lernen (supervised learning)**: Die Überwachung bezieht sich hier auf die Vorgehensweise, wie mit den Daten umgegangen wird, um den Algorithmus zu erstellen. [33] Es wird eine Beziehung zwischen den übergebenen Daten X und den Daten, die berechnet werden sollen Y, gesucht. Beide dieser Datensätze an den Algorithmus übergeben werden. Mithilfe der gefundenen Beziehung will man auch bei zukünftigen ähnlichen Problemen die richtige Lösung finden. [36] In den meisten Fällen kann man bei supervised learning davon ausgehen, dass $X \in \mathbb{R}^d$ mit geeignetem $d \in \mathbb{N}$ ist. Die Outputdaten sind meist $Y \in \mathbb{R}$ im Rahmen eines Regressionsproblems oder in $Y \in \{1, ..., K\}$ mit $K \in \mathbb{N}$ mit einem Klassifikationsproblem in K Klassen. Das Ziel des supervised learinings ist es einen Zusammenhang zwischen X und Y herzustellen, sodass eine Funktion idealerweise $Y = f(X)$ erfüllt. [34] Wenn man Vorhersagen über die Zukunft treffen und bestimmte Parameter vorhersagen will, verwendet man das überwachte Lernen.[33]

2. **Nicht überwachtes Lernen (unsupervised learning)**: Bei dem nicht überwachten Lernen gibt es im Gegensatz zum überwachten Lernen keine Outputdaten Y. Deswegen bezeichnet man hier die Daten als nicht gelabelt. Hier besteht meist die Herausforderung Strukturen festzustellen und dann die Daten in Gruppen einzuteilen.[33] Beim unsuperviesed learning sind Trainingsdaten $X_i \in \mathcal{X} \subset \mathbb{R}^d$ gegeben, für die versucht wird, bestimmte Muster zu erkennen. [34]

3. **Halb überwachtes Lernen (semi-supervised Learning)**: Dieser Punkt ist eine Zusammenführung der zwei vorigen Punkte. Es werden eine kleine Menge von markierten Daten und eine große Menge von nicht markierten Daten übergeben. [36]

4. **Bestärkendes Lernen (reinforcement learning)**: Hier will man ein System entwickeln, das sich aufgrund der Interaktion mit der Umgebung weiterentwickelt. Es

gibt meist keinen absolut richtigen Wert, sondern die Güte wird durch eine Belohnungsfunktion gegeben. Es wird also eine Strategie entwickelt, welche das Ziel hat, eine Belohnung zu maximieren. Eine typische Anwendung des bestärkenden Lernens ist zum Beispiel ein Machine Learing Algorithmus für ein Schachspiel. Man kann anhand der nacheinander ausgeführten Züge am Ende des Spiels feststellen, wie erfolgreich der Algorithmus ist, je nachdem ob das Spiel gewonnen oder verloren wurde. [33]

5. **Transfer Lernen**: Oft sind die Daten auf Grundlage dessen der Algorithmus entwickelt wird, veraltet oder die Datenverteilung ändert sich im Laufe der Zeit. Deshalb muss hier der Transfer des Wissens von der Quelldomäne zur Zieldomäne erfolgen. Das Wissen von einem Algorithmus wird weitergegeben, um ein anderes Problem zu lösen. [36]

Im folgenden wollen wir neuronale Netze mit verschiedenen Schichten, sogenannten Layern, betrachten, um Machine Learning zu implementieren. Dabei betrachten wir die verschiedenen Arten von neuronalen Netzen und am einfachsten Beispiel, dem feedforward neuronalen Netz wird die generelle Vorgehensweise erklärt.

3.1 Feed Forward Neuronale Netze

Unter einem neuronalen Netzwerk versteht man ein Modell, das aus einer bestimmte Anzahl an Knoten, einer bestimmte Anzahl an Layern und einer Verbindung zwischen den Layern besteht. Das bekannteste neuronale Netzwerk ist das feedforward neuronale Netzwerk.
Dieses besteht aus

- **einem Input Layer:** Der Input-Layer ist der erste Layer des Netwerks und hat typischerweise so viele Knoten, wie Eingabewerte, da mithilfe des Input Layers die Eingabewerte ins Netzwerk eingeführt werden. Nach dem Input Layer folgen
- **ein oder mehreren Hidden-Layer:** Durch die Gewichtung der Verbindungen zwischen den Knoten der einzelnen Hidden-Layern lernt das neuronale Netzwerk. Dabei ist jeder Knoten eines Layers mit allen Knoten des vorigen Layers und des nächsten Layers verbunden.
- **einem Output Layer:** Vom Output-Layer bekommen wir das Ergebnis geliefert. Um dieses zu bekommen, wird meist eine Sigmaoid-Aktivierungsfunktion (siehe Kapitel 3.1.1) verwendet.

Jeder Layer ist vollständig mit seinem vorherigen Layer verknüpft und kann aus verschieden vielen Knoten bestehen. Die Gewichtung der einzelnen Verbindungen wird stets weiterentwickelt. Dieser Lernmechanismus heißt Rückwärtspropagation. Den Namen erhält das

Modell vom biologischen Gehirn. Auch dieses hat eine große Menge an Neuronen (wie die Knoten im Modell), die durch einen Input angeregt werden. Die Verbindungen zwischen den Neuronen wird durch das stetige Lernen weiterentwickelt. Im menschlichen Körper geht man von mehr als 500 Billionen Verbindungen zwischen den einzelnen Neuronen aus. Diese Größenordnung erreicht bisher kein künstliches neuronales Netzwerk. Je mehr Hidden-Layer unser neuronales Netzwerk hat, desto tiefer ist es. [31]

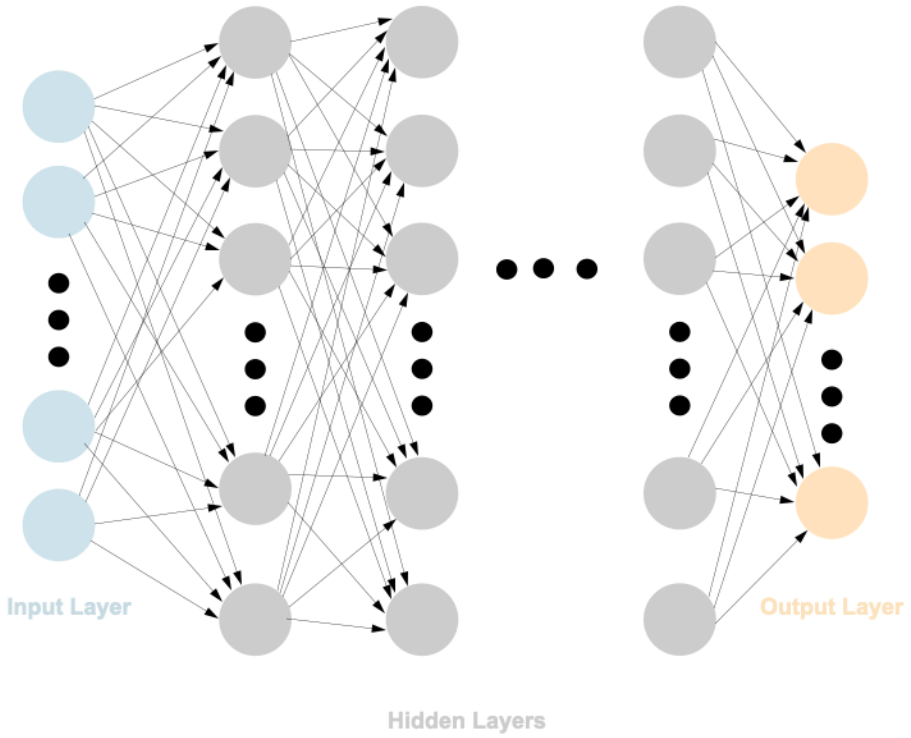


Abbildung 3.1: Aufbau des neuronalen Netzwerks

Mathematisch ist ein neuronales Netz laut [15] folgendermaßen definiert:

Definition 1 (neuronales Netz)**.** *Es sei* $L, N_0, N_1, ..., N_L \in \mathbb{N}$*, mit* $L \geq 2$ *und* $\sigma : \mathbb{R} \to \mathbb{R}$ *eine Aktivierungsfunktion. Bezeichne* $N = (N_0, ..., N_L)^\top$*. Außerdem sei für alle* $\ell = 1, ..., L$ *sei* $W_\ell : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_l}$ *eine affine Funktion. Dann heißt eine Funktion* $F : \mathbb{R}^{N_0} \to \mathbb{R}^{N_L}$*, die definiert ist als*

$$F(x) = W_L \circ F_{L-1} \circ ... \circ F_1 \text{ mit } F_\ell = \sigma \circ W_\ell \text{ für } \ell = 1, ..., L-1$$

(feedforward) neuronales Netz. Wobei hier die Aktivierungsfunktion σ *komponentenweise*

angewendet wird. Die Anzahl der Layer wird als L bezeichnet und die Dimensionen der Hidden-Layers als $N_1, ..., N_{L-1}$. Der Input-Layer ist N_0 und der Output-Layer N_L. Für alle $\ell = 1, ..., L$ ist die affine Transformation W_ℓ gegeben als $W_\ell(x) = \omega^\ell x + b^\ell$ für ein $\omega^\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ und $b^\ell \in \mathbb{R}^{N_\ell}$. Für alle $i = 1, ..., N_\ell$ und $j = 1, ..., N_{\ell-1}$ ist ω^ℓ_{ij} das Gewicht der Kanten, die den Knoten i vom Layer $\ell - 1$ mit dem Knoten j vom Layer ℓ verbindet. Die Anzahl der der Gewichte ungleich 0 ist die Summer der Einträge der Matrizen $\omega^\ell, \ell = 1, ..., L$ und Vektoren $b^\ell, \ell = 1, ..., L$, die ungleich 0 sind. Wir schreiben $\mathcal{F}(L, N)$ für die Menge aller solcher Funktionen F.

Bei der Backpropagation werden die Trainingsdaten nacheinander in das neuronale Netz eingespeist. Das Ergebnis, das vom Modell berechnet wird, wird dann mit dem wirklichen Ergebnis verglichen. Stimmen diese beiden überein passiert nichts. Stimmen diese beiden nicht überein, werden die Gewichte des Modells so verändert, sodass der Fehler minimiert wird. [31]

3.1.1 Aktivierungsfunktion

Aktivierungsfunktionen werden verwendet, um Werte von den Knoten von einem Layer zu den Knoten des nächsten Layers zu übergeben. Mithilfe der Aktivierungsfunktion wird ein Knoten angeleitet, wie er sich zu verhalten hat. In den Hidden Layern wird die Aktivierungsfunktion angewendet, um das gesamte Modell zu entlinearisieren. Auch beim Output Layer wird die Aktivierungsfunktion angewendet, um das Ergebnis auszugeben. [31]

Sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Die Sigmoid-Funktion wird meist im Output Layer verwendet, da sie im Hidden Layer zum Problem der verschwindenden Gradienten führen kann. [19] Für sehr große oder sehr kleine x ist die Geschwindigkeit, mit der das Modell lernt, sehr langsam, da der dazugehörende Gradient verschwindet. Dies führt besonders bei sehr tiefen neuronalen Netzen zu Problemen, da die Verlangsamung die Konvergenz alle nachfolgenden Layer betrifft. [36] Genaueres dazu siehe in Kapitel 3.1.4. Die Ergebnisse der Sigmoid-Funktion liegen zwischen 0 und 1. [31]

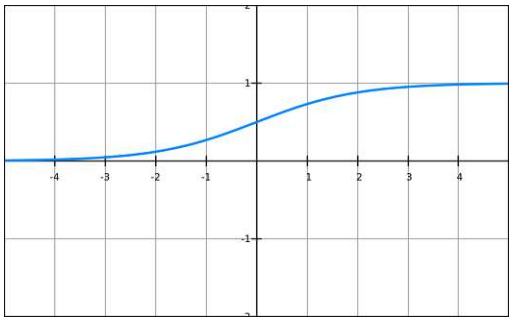


Abbildung 3.2: Sigmoid Aktivierungsfunktion [31]

Hyperpolische Tangente (tanh):

$$\sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Im Gegensatz zur Sigmoid Funktion liegt die Ausgabe zwischen -1 und 1 und ist um 0 zentriert. [19]

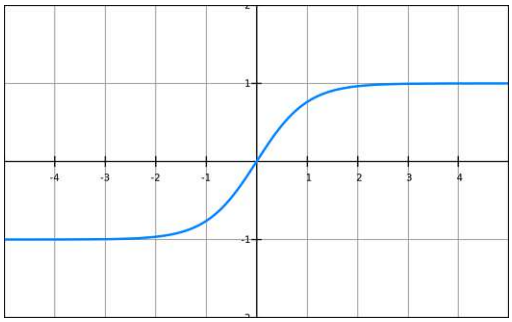


Abbildung 3.3: Tanh Aktivierungsfunktion [31]

Softmax:

$$\sigma(x) = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}}$$

Bei der Softmax-Aktivierungsfunktion wird ein K-elementiger Vektor in eine Wahrscheinlichkeitsverteilung umverwandelt. [19] Diesen Typ von Aktivierungsfunktion benötigt man, wenn es keine binäre Klassifizierung eines Problems gibt. [36]

Rectified Linear Unit (ReLU):

$$\sigma(x) = max(0, x)$$

Bei ReLU wird ein Knoten nur dann aktiviert, wenn die Eingabe über einem bestimmten Wert, meist 0, ist. Ist die Eingabe über 0, ist die Funktion linear. Diese Funktion wird sehr häufig verwendet, da sie in vielen verschiedenen Situationen gute Ergebnisse erzielt. Im Gegensatz zu der Sigmoid und der hyperbolischen Tangente hat die ReLU kein Problem mit verschwindenden Gradienten. [31] Des Weiteren ist die Verarbeitungsgeschwindigkeit um einiges schneller, als bei der hyperbolischen Tangente und der Sigmoid-Funktion. [36] Allerdings ist die ReLU Funktion nicht um null zentriert und falls der Input negativ ist, ist ReLU inaktiv. Dies kann beim Backpropagation-Prozess zu Problemen führen, da der Gradient komplett mit 0 übereinstimmt. Letzteres wird auch Dying ReLU Problem genannt. [36]

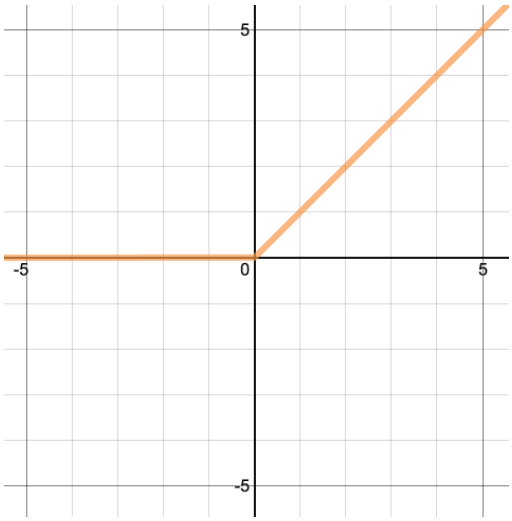


Abbildung 3.4: ReLU Aktivierungsfunktion [31]

Leaky ReLU:

$$\sigma(x) = \begin{cases} 0,01x & \text{wenn } x < 0 \\ x & \text{wenn } x \geq 0 \end{cases}$$

Im Gegensatz zu der ReLU sind bei den Leaky ReLU die Werte kleiner als 0 nicht konstant 0, sondern sind ganz leicht negativ. Mithilfe der Leaky ReLU kann man das Dying ReLU Problem lösen. Dieses tritt auf, wenn alle Inputwerte unter 0 sind und das neuronale Netz deswegen nicht lernen kann. [19]

Softplus:

$$f(x) = ln(1 + exp(x))$$

Bei der Softplus Aktivierungsfunktion handelt es sich um eine glatte Variante der ReLU Aktivierungsfunktion. Es wird versucht, eine ähnliche Gestalt wie die ReLU Aktivierungs-

funktion zu erzeugen, nur, dass die erste Ableitung überall existieren und von Null verschieden sein soll. [31]

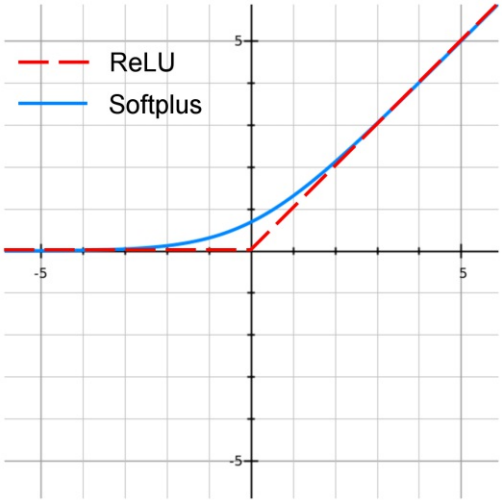


Abbildung 3.5: Softplus Aktivierungsfunktion [31]

Parametric ReLU:

$$\sigma(x) = \begin{cases} \alpha x & \text{wenn } x < 0 \\ x & \text{wenn } x \geq 0 \end{cases}$$

Der parametrischen ReLU ist ähnlich wie das Leaky ReLU. Die Verbesserungen zu der ReLU sind vergleichbar mit jenen des Leaky ReLU zur ReLU. Im Gegensatz dazu können die Parameter aber lernen. [19]

Exponential Linear Unit:

$$\sigma(x) = \begin{cases} \alpha(e^x - 1) & \text{wenn } x < 0 \\ x & \text{wenn } x \geq 0 \end{cases}$$

Eine andere Erweiterung des Leaky ReLU ist die Exponential Linear Unit. [19]

Lineare Funktion: Die lineare Funktion ist die Identitätsfunktion, was bedeutet, dass das Signal unverändert bleibt. Die Lineare Funktion wird im Input-Layer verwendet. [31]

$$\sigma(x) = x$$

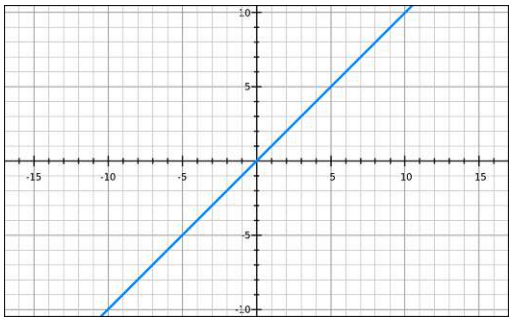


Abbildung 3.6: Lineare Aktivierungsfunktion [31]

3.1.2 Verlustfunktion

Mithilfe der Verlustfunktion wird der Fehler nach dem Durchlauf des Netzwerkes eines Trainingsdatensatzes analysiert. Der Wert, der übergeben wird, $y = (y_1, ..., y_N)$ wird mit dem berechneten Wert $\hat{y} = (\hat{y}_1, ..., \hat{y}_N)$ verglichen. N bezeichnet die Anazhl der Werte in den Trainingsdaten. Man unterscheidet hier zwischen lokalen und globalen Fehlern. Der lokale Fehler bezieht sich auf die Abweichung jedes Knotens. Der globale Fehler ist im Gegensatz dazu die Summe der lokalen Fehlern. Letzterer zeigt auch an, wie erfolgreich das gesamte neuronale Netzwerk ist. [19] Die Güte des Netzwerkes wird mit dem globalen Fehler bestimmt und kann mit den folgenden Funktionen berechnet werden:

Mittlerer absoluter Fehler (MAE): Beim mittleren absoluten Fehler wird die Summe der absoluten Fehler durch die Anazhl der Werte in den Trainingsdaten dividiert. So wird die absolute Distanz zwischen der Vorhersage und dem wahren Wert ermittelt. [19]

$$MAE = \frac{\sum_{i=1}^{N} |\hat{y}_i - y_i|}{N}$$

Mittlerer quadratischer Fehler (MSE): Beim mittleren quadratischen Fehler wird die gemittelte Abweichung der einzelnen Daten herangezogen. Für die Definition wird die Euklidische Norm $\|x\|_2 := \sqrt{\sum_{i=1}^{n} x_i^2}$ für $x = (x_1, ..., x_n) \in \mathbb{R}^n$ benötigt.

$$MSE = \frac{1}{N} \sum_{i=1}^{N} \|\hat{y}_i - y_i\|_2^2$$

Der Vorteil des mittleren quadratischen Fehler im Gegensatz zum mittleren absoluten Fehler ist, dass man durch das Quadrieren des Abstandes zwischen der Vorhersage und dem richtigen Wert die Fehler leichter in kleine beziehungsweise große Fehler einteilen kann. [19]

Die Verlustfunktion nach Huber (Huber Loss): Die Verlustfunktion nach Huber hat

die folgende Gestalt, wobei ϵ ein konstanter zu wählender Wert ist:

$$HL = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{wenn } |\hat{y} - y| \leq \epsilon \\ \epsilon|\hat{y} - y| - \frac{\epsilon^2}{2} & \text{sonst} \end{cases}$$

Falls ϵ sehr klein ist, stimmt der Huber Loss mit dem MAE überein und falls ϵ sehr groß ist, sind der MSE und der Huber Loss identisch. [19]

3.1.3 Backpropagation

Dieses Unterkapitel wurde auf Grundlage von [10] beschrieben. Beim Backpropagation Prozess handelt es sich um einen Spezialfall des Gradientenverfahrens, bei dem eine differenzierbare Funktion minimiert werden soll. Hat ein neuronales Netzwerk nur einen Layer, ist die Berechnung des Gradienten einfach, denn dieser kann aus den Gewichten der einzelnen Knoten gebildet werden. Mit diesem erhält man dann auch die Fehler. Nun soll allerdings ein neuronales Netzwerk mit mehreren Layern trainiert werden. Dies ist nicht mehr so einfach, da sich der Verlust aus der Zusammensetzung der Gewichte der einzelnen Knoten in den verschiedenen Layern ergibt. Um das Netzwerk trotzdem bestmögich einzustellen, wird der Backpropagation Algorithmus angewendet. Dieser besteht aus 2 Phasen:

1. **Vorwärtsphase:** In der Vorwärtsphase wird ein Datensatz von Eingabedaten in das neuronale Netzwerk gespeist. Dieses wird mit den aktuellen Gewichten durchlaufen und der Output wird mit dem Validierungsoutput verglichen. Den Validierungsoutput kennen wir, da für die Trainingsdaten nicht nur der Input, sondern auch der Output übergeben wird. Die Werte der einzelnen Knoten werden danach auch in der Rückwärtsphase benötigt.

2. **Rückwärtsphase:** In der Rückwärtsphase soll das neuronale Netzwerk aus dem aktuellen Fehler lernen. Dazu soll der Gradient der Verlustfunktion bezüglich der einzelenen Gewichte minimiert werden, indem die Gewichte aktualisiert werden. Diese Phase wird Rückwärtsphase genannt, weil die Aktualisierung der Knoten mit den Knoten des Output-Layers beginnt und dann rückwärts weiter abgewickelt wird.

Der Algorithmus macht sich die Kettenregel zu Nutze und berechnet den Gradienten mittels der Summe des Produktes der lokalen Gradienten. Dies lässt sich mit dynamischer Programmierung sehr einfach umsetzen.

Seien $h_1, ..., h_k$ die Knoten des Hidden-Layers und sei o der Rückgabewert des neuronalen Netzwerkes. Mithilfe von o und der Verlustfunktion wird dann der Fehler des neuronalen Netzwerkes berechnet. Angenommen es existiert nur ein Pfad vom Knoten h_1 zu unserer Ausgabe o und bezeichne $\omega_{(h_{r-1},h_r)}$ das Gewicht der Verbindung zwischen den Knoten h_{r-1} und h_r. Dann kann der Gradient der Verlustfunktion L aus

$$\frac{\partial L}{\partial \omega_{(h_{r-1},h_r)}} = \frac{\partial L}{\partial o} \cdot \Big[\frac{\partial o}{\partial h_k} \prod_{i=r}^{k-1} \frac{\partial h_{i+1}}{\partial h_i}\Big] \frac{\partial h_r}{\partial \omega_{(h_{r-1},h_r)}} \quad \forall r \in 1, ..., k \tag{3.1}$$

berechnet werden. Existiert mehr als ein Pfad vom Knoten h_1 zu o, muss eine Komposition für jeden Pfad von h_1 zu o gemacht werden. Für das nächste Lemma benötigt man die Grundlagen der Graphentheorie, für die auf [11] verwiesen wird.

Lemma 1 (Lemma der pfadweisen Aggregation)**.** *Betrachte einen gerichteten azyklischen Graphen. Die Knoten des Graphen beinhalten die Variablen $h(i)$. Die lokale Ableitung $z(i,j)$ des gerichteten Pfades (i,j) ist definiert als $z(i,j) = \frac{\partial h(j)}{\partial h(i)}$. Sei $\mathcal{P}$ eine nichtleere Menge der Pfade, die vom Knoten ω zum Knoten des Output-Layers o führen. Dann ist der Wert von $\frac{\partial o}{\partial w}$ gegeben durch das Produkt der lokalen Gradienten entlang jedes Pfades in $\mathcal{P}$ und der Summe über alle Pfade.*

$$\frac{\partial o}{\partial \omega} = \sum_{P \in \mathcal{P}} \prod_{(i,j) \in P} z(i,j)$$

Beweis. Die Aussage folgt direkt aus der rekursiven Anwendung der multivariaten Kettenregel. □

Wendet man nun das Lemma der pfadweisen Aggregation auf das neuroneale Netzwerk an, erhält man mit 3.1

$$\frac{\partial L}{\partial \omega_{(h_{r-1},h_r)}} = \underbrace{\frac{\partial L}{\partial o} \cdot \Big[\sum_{[h_r,h_{r+1},...,h_k,o] \in \mathcal{P}} \frac{\partial o}{\partial h_k} \prod_{i=r}^{k-1} \frac{\partial h_{i+1}}{\partial h_i}\Big]}_{=\frac{\partial L}{\partial h_r} =: \Delta(h_r,o)} \frac{\partial h_r}{\partial \omega_{(h_{r-1},h_r)}} \quad \forall r \in 1, ..., k.$$

Der Term $\Delta(h_r, o)$ wird durch Backpropagation berechnet, wobei die für die Berechnung verwendete Anzahl der Pfade in jedem Schritt exponentiell wächst. Zuerst muss natürlich der Pfad vom nächsten zum Output gelegenen Knoten berechnet werden. Die Berechnung der folgenden Knoten erfolgt dann rekursiv. Mithilfe der multivariaten Kettenregel kann die Rekursion folgendermaßen beschrieben werden

$$\Delta(h_r, o) = \frac{\partial L}{\partial h_r} = \sum_{h:h_r \Rightarrow h} \frac{\partial L}{\partial h} \frac{\partial h}{\partial h_r} = \sum_{h:h_r \Rightarrow h} \frac{\partial h}{\partial h_r} \Delta(h, o).$$

Dies folgt mit obiger Definition, da $\Delta(h,o) = \frac{\partial L}{\partial h}$. Aufgrund der Rekursion ist $\Delta(h,o)$ bereits berechnet. Sei $w_{(h_r,h)}$ das Gewicht zwischen den Knoten h_r und h. Des Weiteren sei

a_h der Wert des Knotens h vor der Anwendung der Aktivierungsfunktion $\sigma(\cdot)$. Dann ist

$$\frac{\partial h}{\partial h_r} = \frac{\partial h}{\partial a_h} \cdot \frac{\partial a_h}{\partial h_r} = \frac{\partial \sigma(a_h)}{\partial a_h} \cdot \omega_{(h_r,h)} = \sigma'(a_h) \cdot \omega_{(h_r,h)}.$$

Daraus folgt

$$\Delta(h_r, o) = \sum_{h:h_r \Rightarrow h} \sigma'(a_h) \cdot \omega_{(h_r,h)} \cdot \Delta(h, o).$$

Dies bedeutet, dass jeder Knoten nur mehr einmal im Laufe eines Backpropagation Durchlaufes bearbeitet wird. Nun muss nur noch

$$\frac{\partial h_r}{\partial \omega_{(h_{r-1},h_r)}} = h_{r-1} \cdot \sigma'(a_{h_r})$$

berechnet werden. Dieser Vorgang muss solange wiederholt werden, indem man die Trainingsdaten durchläuft, bis das Ergebnis konvergiert.

Die bisher vorgestellte Vorgehensweise führt zu folgendem Algorithmus

1. Berechne die einzelnen Werte h(i) für jeden Knoten i in der Vorwärtsphase
2. Berechne die lokalen Ableitungen $z(i,j) = \frac{\partial h(j)}{\partial h(i)}$ für jede Kante des Graphens
3. Sei $\mathcal{P}$ die Menge aller Pfade, die vom Eingabewert zum Ausgabewert gehen. Berechne für jeden Pfad $P \in \mathcal{P}$ das Produkt der lokalen Ableitung z(i,j).
4. Füge alle diese Werte für alle Pfade in $\mathcal{P}$ zusammen

Allerdings wächst die Anzahl der Pfade exponentiell mit der Tiefe des Graphen an. Um mit dieser Tiefe umzugehen, hilft die dynamische Programmierung, das typischerweise bei azyklischen Graphen angewendet wird. Bei der dynamischen Programmierung handelt es sich genau um die multivariate Kettenregel, die, die Pfade von hinten nach vorne durchgeht. Der einzige Unterschied ist, dass man den Graphen in einer bestimmten Reihenfolge durchläuft, um die Anzahl der Berechnungen zu minimieren. Man kann sogar die Variablen vor oder nach der Anwendung der Aktivierungsfunktion betrachten. Beide Varianten führen zum selben Ergebnis.

Backpropagation mit der Softmax Aktivierungsfunktion

Die Softmax-Aktivierungsfunktion unterscheidet sich zu den anderen Aktivierungsfunktionen, weil sie mehrere Input-Werte hat und nicht nur einen. Deswegen ist der Update-Prozess nicht genau gleich wie bei den anderen Aktivierungsfunktionen und muss hergeleitet werden. Wie bereits in Kapitel 3.1.3 beschrieben, werden die Input-Werte $x_1, ..., x_k$ in die

Ausgabe-Werte $o_1, ..., o_k$

$$o_i = \frac{exp(x_i)}{\sum_{j=1}^{k} exp(v_j)}$$

umgerechnet. Die Softmax Aktivierungsfunktion verwandelt dafür die Vorhersagen $v_1, ..., v_k$ in den Output $o_1, ..., o_k$. Falls wir die Backpropagation anwenden wollen, müssen wir die Kettenregel für die Ableitung für L für alle $\frac{\partial L}{\partial o_i}$ und alle $\frac{\partial o_i}{\partial v_j}$ berechnen. Diese Backpropagation können wir sehr stark vereinfachen, weil die Softmax-Aktivierungsfunktion oftmals im Output Layer verwendet wird und meist mit dem Kreuzentropie Verlust auftritt. Der Kreuzentropie Verlust ist für die Outputs $y_1, ..., y_k \in \{0, 1\}$ definiert als

$$L = -\sum_{i=1}^{k} y_i \log(o_i)$$

und ist ein Maß für die Qualität des neuronalen Netzwerks. Der Wert $\frac{\partial L}{\partial v_i}$ ist dabei für die Softmax-Aktivierungsfunktion einfach zu berechnen

$$\frac{\partial L}{\partial v_i} = \sum_{j=1}^{k} \frac{\partial L}{\partial o_j} \cdot \frac{\partial o_j}{\partial v_i} = o_i - y_i$$

Zuerst wird also der Gradient der Output von dem Layer berechnet, der $v_1, ..., v_k$ enthält. Danach kann die Backpropagation wie zuvor beschrieben durchgeführt werden. Wir haben also die Backpropagation von der Softmax Aktivierungsfunktion von der Backpropagation vom Netzwerk getrennt. [10]

3.1.4 Die Probleme der verschwindenden und explodierenden Gradienten

Auch beim Lernen von neuronalen Netzwerken ist man von Problemen nicht gefeit. Wird die Dichte des Netzwerkes vergrößert, wird meist die Anzahl der Parameter verringert, aber diese Vertiefung führt meist auch zu praktischen Problemen.

Ein Problem, das bei besonders tiefen neuronalen Netzwerken auftritt, heißt das Problem der verschwindenden Gradienten. Wir betrachten nun also ein tiefes neuronales Netzwerk mit (m+1) Layern. Dieses Netzwerk hat dann die Gewichte $\omega_1, ..., \omega_m$. Als Aktivierungsfunktion wird bei diesem neuronalen Netzwerk die Sigmoid-Funktion hergenommen. Wir verwenden den Backpropagation Algorithmus, wie zuvor beschrieben. Weiters nehmen wir an, dass zu Beginn die Gewichte mithilfe der Standardnormalverteilung initialisiert werden. Daraus folgt, dass die erwartete Größe jedes Gewichtes ω_t 1 ist.

Die Ableitung eines Sigmoids mit dem Ergebnis $f \in (0, 1)$ ist gegeben durch $f(1-f)$. Daraus

folgt, dass das maximale Ergebnis $f = 0,5$ und dass das Maximum kleiner als $\sigma'(h_t) = 0,25$ ist. Wir können mithilfe der Resultate, die wir beim Backpropagation Algorithmus herausbekommen haben herleiten, dass

$$\frac{\partial L}{\partial h_t} = \sigma'(h_{t+1}) \cdot \omega_{t+1} \cdot \frac{\partial L}{\partial h_{t+1}}$$

ist. Wegen der Initialisierung mit der Standardnormalverteilung soll nun der erwartete Wert von ω_{t+1} wieder 1 sein. Dadurch wird durch die Aktualisierung der Gewichte, der Wert von $\frac{\partial L}{\partial h_t}$ kleiner als 0,25 mal $\frac{\partial L}{\partial h_{t+1}}$ sein. Nach x Layern ist dann der Wert typischerweise kleiner als $0,25^x$. Die ersten Layer des neuronalen Netzwerkes bekommen also im Gegensatz zu den letzten Layern wegen der Backpropagation nur einen kleines Update.

Um diesem Problem entgegenzuwirken, könnte man eine Aktivierungsfunktion mit größerem Gradienten und und größeren initialisierten Gewichten benutzen. Hier besteht allerdings genau das gegenteilige Problem. Werden die Gradienten und Gewichte zu groß gewählt, besteht das Problem der explodierenden Gradienten.

Der erste Schritt, um dem Problem des verschwindenden Gradienten auszuweichen, ist die richtige Aktivierungsfunktion zu wählen. Wie bereits erwähnt, hat die Sigmoid Aktivierungsfunktion nie einen größeren Gradienten als 0,25, weshalb die Sigmoid Aktivierungsfunktion sehr anfällig für verschwindende Gradienten ist. Eine weitere Aktivierungsfunktion bei der die Gradienten oft verschwinden, ist die tanh Aktivierungsfunktion, da die Steigung der Funktion sehr schnell abflacht. Eine gute Alternative zu diesen zwei anfälligen Aktiverungsfunktionen ist die ReLU. Der Vorteil der ReLU Akrivierungsfunktion ist, dass ihre Ableitung für positive Werte immmer 1 ist und somit nicht verschwinden.

Andere Verfahren, um dem Problem auszuweichen, sind zum Beispiel adaptierte Lernraten und konjugierte Gradientenmethoden. [10]

3.1.5 Lernrate und Momentummethode

Der bisher besprochene Backpropagation Algorithmus wird das steepest-descent Gradientenverfahren genannt. Hier kann es allerdings zu dem Problem kommen, dass nicht immer die richtige Richtung beim Verbessern der Gradienten verwendet wird. Deswegen benötigt man oft viele Korrekturschritte, was zu einem Zickzackkurs und Oszillationen bei den einzelnen Gradienten führen kann. Um diesen Problemen vorzubeugen, führt man eine Lernrate ein, die für die Geschwindigkeit der Fehlerminimierung in jedem Schritt verantwortlich ist. Diese Lernrate richtig zu wählen, ist nun die Problemstellung. Einerseits ist eine konstante Lernrate nicht optimal, da diese dazu führt, dass sie, wenn sie zu klein gewählt wird, anfangs zu einem zu langsamen Algorithmus führt. Aber falls sie zu groß gewählt wird, sich zwar

schnell einer guten Lösung annähert, aber dann um das optimale Ergebnis oszilliert. Deswegen verkleinert man die Lernrate mit der Zeit, um ein optimales Ergebnis zu erzielen. [10]

Die Verkleinerung der Lernrate wird mit einer Dämpfungsfunktion erzielt. Die zwei meist verwendeten Dämpfungsfunktionen sind

- Exponentieller Abfall: $\alpha_t = \alpha_o exp(-k \cdot t)$
- Inverser Abfall: $\frac{\alpha_0}{1+k \cdot t}$

Wobei α_t die Lernrate und α_0 die initiale Abfallsrate bezeichnet. t bezeichnet die vergangen Zeitepochen und k die Rate des Abfalls. Ein anderer Ansatz ist, dass die Lernrate alle paar Schritte verringert wird. [10]

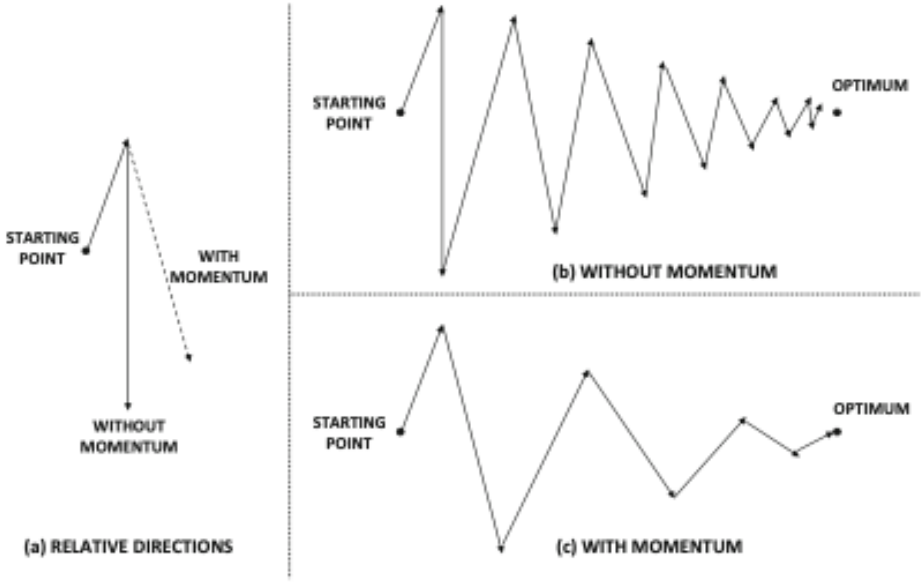


Abbildung 3.7: Einfluss der Momentummethode auf den Zickzackkurs bei den Gradientenupdates [10]

Eine andere Strategie um die richtigen Gewichte schnell zu erreichen ist die Momentummethode. Diese erkennt die Zickzacklinie aufgrund von sich aufhebenden Schritten. Die Methode ermittelt die gemittelte Richtung der letzten paar Schritte und versucht so, die Zickzacklinie zu glätten. Sei dafür $\overline{W}$ der Parametervektor, aufgrund dessen der Gradientenabstieg durchgeführt wird. Dann sind die Normalisierungen ohne Momentummethode,

aber mit Lernrate α folgendermaßen

$$\overline{V} \Leftarrow -\alpha \frac{\partial L}{\partial \overline{W}}$$
$$\overline{W} \Leftarrow \overline{W} + \overline{V},$$

wobei $\Leftarrow$ das jeweilige Update und $\overline{V}$ den Updatevektor bezeichnen soll.
Wird die Momentummethode angewendet, wird der Updatevektor $\overline{V}$ mithilfe einer exponentiellen Glättung optimiert:

$$\overline{V} \Leftarrow \beta \overline{V} - \alpha \frac{\partial L}{\partial \overline{W}}$$
$$\overline{W} \Leftarrow \overline{W} + \overline{V}$$

Wobei $\beta \in (0, 1)$ der Glättungsparameter ist. Das Verwenden der Momentummethode kann auch dazu führen, dass die Lösung in die initiale Richtung leicht überschießt. Allerdings ist das Ergebnis auch dann noch besser, als wenn die Momentummethode nicht verwendet wird. Große β helfen lokale Optima zu vermeiden, aber sie führen auch am Ende zu mehr Oszillation um das korrekte Ergebnis. [10]

3.1.6 Parameterspezifische Lernraten

Die Wahl der richtigen Lernraten ist von wichtiger Bedeutung, denn diese tragen maßgeblich zur Perfomence des Modelles bei. Mit der Einführung der Momentummethode und der Lernraten wurde hierfür bereits einen wichtigen Schritt zu einem schnelleren Modell gemacht. Die Hauptidee zu der Momentummethode vom vorigen Unterpunkt ist, dass der Algorithmus den Zickzackkurs für alle Parameter bei den Updates erkennt und verringert. Dieser Vorgang lässt sich noch weiter beschleunigen, indem man für die Parameter spezifische Lernraten verwendet und diese anpasst. Der Hintergedanke hierbei ist, dass Gradienten mit großen partiellen Ableitungen oft oszillieren, wohingegen Gradienten mit kleinen partiellen Ableitungen nicht so häufig oszillieren. [10]

Adagrad und RMSProp

Zwei Methoden für die parameterspezifische Anpassung der Lernraten sind Adagrad und RMSProp. Der Adagrad Algorithmus untersucht den aggregierten quadratischen Wert der partiellen Ableitung für jeden Parameter, denn die Wurzel dieses Wertes ist proportional zur quadratischen Steigung dieses Parameters. Im Gegensatz dazu verwendet der RMSProp Algorithmus exponentielle Mittelwertbildung. Beide Methoden vereint, dass die Lernrate für Parameter mit großen partiellen Ableitungen der Fehlerfunktion schneller sinkt, als für

jene mit kleinen partiellen Ableitungen. [10]

Um das Update des i-ten Parameters w_i zu berechnen, benötigt man bei beiden Algorithmen die Variable A_i. Die Bedeutung dieser Variable unterscheidet sich jedoch. Bei dem Adagrad Algorithmus handelt es sich bei A_i um den aggregierten Wert des i-ten Parameters und beim RMSProp Algorithmus handelt es sich um den exponentiellen Mittelwert vom Parameter w_i. Der genauen Berechnung von A_i widmen wir uns nach dem Update des i-ten Parameters mit der Lernrate α, der für beide Algorithmen gegeben ist durch:

$$w_i \Leftarrow w_i - \frac{\alpha}{\sqrt{A_i}} \left(\frac{\partial L}{\partial w_i} \right) \quad \forall i$$

Um eine Division durch 0 zu vermeiden, kann man statt $\sqrt{A_i}$ auch $\sqrt{A_i + \epsilon}$ wählen, wobei ϵ eine sehr kleine positive Zahl sein soll. Der einzige Unterschied zwischen den zwei Methoden, liegt, wie schon erwähnt, in der Berechnung von A_i. Für den Adagrad Algorithmus ist A_i der aggregierte Wert des i-ten Parameters. Dieser wird durch folgende Formel berechnet:

$$A_i \Leftarrow A_i + \left(\frac{\partial L}{\partial w_i} \right)^2 \quad \forall i$$

Im Gegensatz dazu ist A_i bei dem RMSProp der exponentielle Mittelwert vom Parameter w_i. Hierfür wird eine Abfallrate $\rho \in (0, 1)$ eingeführt.

$$A_i \Leftarrow \rho A_i + (1 - \rho) \left(\frac{\partial L}{\partial w_i} \right)^2 \quad \forall i$$

Ein Vorteil von RMSProp zu Adagrad ist, dass die Bedeutung von alten Gradienten mit der Zeit abnimmt. Ein Nachteil ist jedoch, dass durch die laufende Schätzung von A_i die Momentums zweiter Ordnung in frühen Iterationen verzerrt sind, weil die A_i mit 0 initialisiert sind. [10]

Adam Optimizer

Standardmäßig wird für die Anpassung der Lernrate die Adam Methode verwendet. Sie hat im Gegensatz zu den vorherigen Algorithmen einige Vorteile. Denn die Anzahl der Schritte ist bei Adam begrenzt und das Ausmaß der Anpassung verändert nicht die Dimension der Gradienten. Des Weiteren funktioniert der Adam Optimizer auch für schwach besetzte Gradienten. [23]

Wie beim RMSProp Algorithmus bezeichnet A_i auch beim Adam Algorithmus den expo-

nentiellen Mittelwert vom Parameter w_i mit der Abfallrate $\rho \in (0, 1)$:

$$A_i \Leftarrow \rho A_i + (1 - \rho) \left(\frac{\partial L}{\partial w_i} \right)^2 \quad \forall i$$

Zusätzlich zu A_i wird mit F_i die i-te Komponente des exponentiell geglätteten Wertes des Gradienten berechnet. Für diesen geglätteten Wert wird eine zweite Abfallrate $\rho_f \in (0, 1)$ eingeführt:

$$F_i \Leftarrow \rho_f F_i + (1 - \rho_f) \left(\frac{\partial L}{\partial w_i} \right) \quad \forall i$$

Sowohl A_i als auch F_i werden mit 0 initialisiert. Außerdem konvergieren die beiden Abnehmraten für wachsende t gegen 0. Das Update des i-ten Parameter unterscheidet sich nun vom RMSProp Algorithmus, weil der Gradient durch den exponentiell geglätteten Wert ersetzt wird:

$$w_i \Leftarrow w_i - \frac{\alpha_t}{\sqrt{A_i}} F_i \quad \forall i$$

Außerdem hängt die Lernrate α_t nun von einem Iterationsindex t ab und ist definiert durch

$$\alpha_t = \alpha \left(\frac{\sqrt{1 - \rho^t}}{1 - \rho_f^t} \right).$$

Da die Abhnehmraten für steigendes t gegen 0 konvergieren, konvergiert die Lernrate gegen 1. [10] In [23] wird vorgeschlagen, dass für ρ ein Wert von 0,999 und für ρ_f ein Wert von 0,9 gewählt wird. Außerdem wird für die Schrittgröße α ein Wert von 0,001 empfohlen.

3.2 Rekurrent Neuronal Networks (RNN)

Im Gegensatz zu Feed-Forward Neuronal Networks ist das rekurrente neuronale Netzwerk für multidimensionale Daten geeignet, die nicht unabhängig voneinander sind. Die Daten werden nicht zuerst vom Input-Layer, dann zum Hidden-Layer und danach zum Output Layer übergeben, sondern sie sind auch miteinander verknüpft. Dadurch können bestimmte Strukturen in den Input-Daten besser erkannt werden. Dies ist besonders bei Zeitreihen, wie beispielsweise bei Aktienkurse, oder bei Textanalysen von Bedeutung. [36]

Die Eingabe eines rekurrenten neuronalen Netzwerk ist $\overline{x_1}, ..., \overline{x_t}$, wobei $\overline{x_i}$ für alle i von $1, ..., t$ ein d-dimensionaler Vektor ist. In dieser Art von Netzwerk ist es möglich, dass die Eingabe mit den Werten des Hidden-Layers, die von den früheren Eingaben erzeugt wurden, interagiert. Deswegen bezeichnen wir jeden Schritt des Durchlaufes als Zeitpunkt. Mit dem p-dimensionen Vektor $\overline{h_i}$ bezeichnen wir den Wert des Hidden-Layers zum Zeitpunkt $i = 1, ..., t$. Dieser verändert sich zu jedem Zeitpunkt, indem neue Daten durchgespielt werden. Jeder Zeitpunkt hat auch einen Output $\overline{y_1}, ..., \overline{y_t}$, wobei $\overline{y_i}$ ebenfalls ein d-dimensionaler

Vektor ist. Haben wir ein endliches rekurrentes Netzwerk, kann dieses entfalten werden, sodass ein feedforward neuronales Netz entsteht. Die Berechnung eines Wertes im Hidden-Layer erfolgt folgendermaßen

$$\overline{h_t} = f(\overline{h_{t-1}}, \overline{x_t}) \tag{3.2}$$

Die Funktion f wird dabei gebildet aus den Gewichtsmatrizen und den Aktivierungsfunktionen, wobei die Matrizen der Gewichte von den einzelnen Zeitpunkten gemeinsam verwendet werden. Daraus folgt, dass die Funktion $f(\cdot,\cdot)$ für alle Zeitpunkte gleich ist, nachdem das Training abgeschlossen ist.

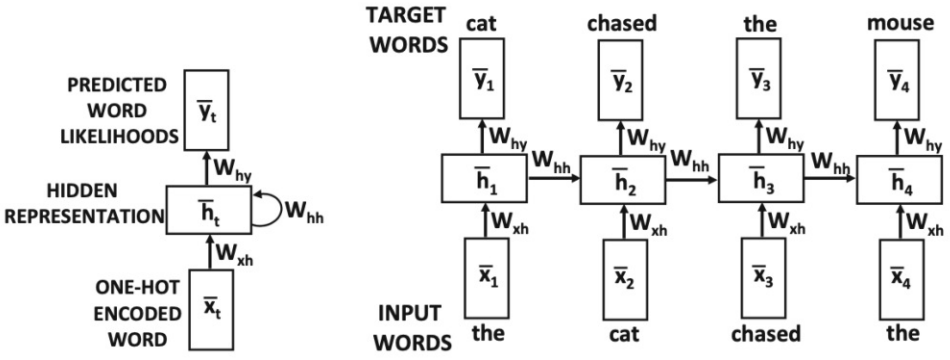
Konkret wird als Funktion f folgendes verwendet:

$$\overline{h_t} = tanh(W_{xh}\overline{x}_t + W_{hh}\overline{h}_{t-1})$$

Dabei bezeichnet W_{xh} eine $p \times d$ Matrix mit den Gewichten zwischen dem Input-Layer und dem Hidden-Layer. W_{hh} bezeichnet die $p \times p$ Matrix mit den Werten zwischen den Hidden-Layern. Außerdem wird der tanh komponentenweise angewendet.
Auch für die Berechnung des Outputs aus den Werten des Hidden-Layers, wird eine Funktion g verwendet, die mithilfe der $d \times p$ Matrix der Gewichtswerte zwischen Hidden- und Output-Layer geschrieben werden kann:

$$\overline{y_t} = g(\overline{h_t}) = W_{hy}\overline{h_t}$$

Durch die rekursive Definition von $\overline{h_t}$ in 3.2, kann der Input in ein rekurrentes neuronales Netzwerk von verschiedener Länge sein.



(a) RNN (b) Time-layered representation of (a)

Abbildung 3.8: Architektur eines RNN, am Beispiel des Satzes "The cat chased the mouse". [10]

Beim feedforward neuronalen Netzen werden die einzelnen Worte einzeln in das Netzwerk eingespielt, dadurch wird die Reihenfolge der Worte ignoriert. Allerdings ist die Reihenfolge von Wörtern oft von Bedeutung. Betrachtet man die zwei Sätze „Die Katze jagt die Maus.“ und „Die Maus jagt die Katze.“ so ist die Bedeutung der zwei Sätze unterschiedlich. In feedforward neuronalen Netz würden beide allerdings als identisch gelten, da die Reihenfolge keine Rolle spielt. Im Gegensatz dazu erkennt ein rekurrentes neuronales Netz den Unterschied der beiden Sätze. Eine Anwendung des Netzwerks wäre zum Beispiel bei der Vorhersage der nächsten Wörter beim Schreiben am Handy aufgrund der letzten geschriebenen Wörter. [10]

3.2.1 Backpropagation through Time (BPTT)

Für die Backpropagation bei einem Rekurrenten neuronalen Netzwerk verwenden wir die Softmax Funktion aus Kapitel 3.1.1. Dazu muss zuerst der Output Vektor $\overline{y_t} = [\hat{y}_t^1, ..., \hat{y}_t^d] \in \mathbb{R}^d$ in d Wahrscheinlichkeiten umgerechnet werden. Für dies wird die Softmax-Funktion verwendet:

$$[\hat{p}_t^1, ..., \hat{p}_t^d] := \text{Softmax}([\hat{y}_t^1, ..., \hat{y}_t^d])$$

Die Verlustfunktion berechnet sich nach Kapitel 3.1.2 für jeden Zeitpunkt gesondert, folgendermaßen

$$L = -\sum_{t=1}^{T} log(\hat{p}_t^{j_t}),$$

wobei j_t der Index des j-ten Inputs zum Zeitpunkt t ist. Mittels der partiellen Differentiation

$$\frac{\partial L}{\partial \hat{y}_t^k} = \hat{p}_t^k - I(k, j_t)$$

kann man die Backpropagation durchführen, die unter Kapitel 3.1.3 erklärt wurde. Dabei bezeichnet $I(k, j_t)$ die Indikatorfunktion, die eins ist, wenn $k = j_t$ ist und null sonst. Mit der Annahme, dass die Gewichte der verschiedenen Layer voneinander getrennt sind, ist die Durchführung der Backpropagation kein Problem.

Es besteht auch die Möglichkeit, gemeinsame Gewichte zu verwenden. Mathematisch lässt sich das wie folgt darstellen: Sei ω ein Gewicht, das in T verschieden Knoten hergenommen wird. Wir kopieren nun dieses Gewicht in $\omega_1, ..., \omega_T$. Somit hat jeder Knoten ein eigenes Gewicht der gleichen Höhe. Wir nehmen also an, dass diese Gewichte von einander unabhängig

sind. Mithilfe der Kettenregel ergibt dies

$$\frac{\partial L}{\partial \omega} = \sum_{i=1}^{T} \frac{\partial L}{\partial \omega_i} \cdot \underbrace{\frac{\partial \omega_i}{\partial \omega}}_{=1} = \sum_{i=1}^{T} \frac{\partial L}{\partial \omega_i}$$

Somit können wir auch weiterhin den Backpropagation Algorithmus anwenden. Um die gemeinsamen Gewichte nun anzuwenden, werden die Gewichte $W_{xh}^{(t)}$, $W_{hh}^{(t)}$,$W_{hy}^{(t)}$ für jeden Zeitpunkt t definiert. Diese sind von den Gewichten anderer Zeitpunkten unabhängig. Wir erhalten die partiellen Ableitungen

$$\begin{aligned}
\frac{\partial L}{\partial W_{xh}} &= \sum_{t=1}^{T} \frac{\partial L}{\partial W_{xh}^{(t)}} \\
\frac{\partial L}{\partial W_{hh}} &= \sum_{t=1}^{T} \frac{\partial L}{\partial W_{hh}^{(t)}} \\
\frac{\partial L}{\partial W_{hy}} &= \sum_{t=1}^{T} \frac{\partial L}{\partial W_{hy}^{(t)}}
\end{aligned}$$

Diese Darstellung entspricht der Anwendung des Backpropagation Algorithmus eines feedforward neuronalen Netzes. Die Backpropagation-in-time ist also nicht von der normalen Backpropagation zu unterscheiden. [10]

3.2.2 Probleme beim Trainieren von rekurrenten neuronalen Netzwerken

Ein rekurrentes neuronales Netzwerk ist schwer zu trainieren, da es meist sehr tief ist. Die Tiefe des Netzwerkes ist von der Eingabe abhängig. Wie schon beim feedforward neuronalen Netz ist auch hier die Verlustfunktion unterschiedlich stark empfindlich. Die Empfindlichkeit hängt von der Tiefe des Netzwerkes ab. Da des Weiteren die Gewichtsmatrizen von verschiedenen Layern verwendet werden, ist das Netzwerk oft sehr instabil. Wie auch schon beim feedforward neuronalen Netz sind die zwei Hauptprobleme hierbei das Problem mit den verschwindenden Gradienten und das Problem der explodierenden Gradienten. Dass das öftere Multiplizieren mit der selben Matrix zu Problemen führen kann, zeigt auch folgendes Lemma:

Lemma 2. *Es sei A eine quadratische Matrix und λ der betragsmäßig größte Eigenwert von A. Nun gilt, falls $\lambda < 1$ tendieren die Werte von A^t gegen 0 für steigendes t. Falls $\lambda > 1$ divergieren die Werte von A^t gegen große Werte.*

Beweis. Sei oBdA die Matrix diagonalisierbar. Ist die Matrix nicht diagonalisierbar, folgt

der Beweis mit der Jordanschen Normalform.
Sei nun P die Basiswechselmatrix und Δ die Diagonalmatrix, die die Eigenwerte enthält.

$$A = P\Delta P^{-1}$$

Nun ist wegen $P^{-1} \cdot P = I$, wobei I die Einheitsmatrix ist

$$\begin{aligned} A^t &= (P\Delta P^{-1})^t \\ &= \underbrace{P\Delta P^{-1} \cdot P\Delta P^{-1} \cdot ... \cdot P\Delta P^{-1}}_{\text{t mal}} \\ &= P\Delta^t P^{-1} \end{aligned}$$

Der Betrag des größten Diagonaleintrags von Δ^t verschwindet entweder für größere t oder wächst immer weiter, je nachdem ob der Eigenwert größer oder kleiner 1 ist. □

Wendet man nun das obrige Lemma an, dann tendiert die Matrix A^t im ersten Fall gegen 0 und der Gradient verschwindet. Im zweiten Fall explodieren die Gradienten. [10]

3.3 Long Short-Term Memory (LSTM) Networks

Wie in Kapitel 3.2.2 besprochen, haben rekurrente Netzwerke beim Trainieren das Problem des verschwindenden bzw. explodierenden Gradienten. Dies resultiert aus der immer wieder kehrenden Multiplikation mit den Gewichten zu jedem Zeitpunkt t. Man kann also sagen, dass diese Art von Netzwerk sehr gut ist, um ein Kurzzeitgedächtnis zu simulieren, aber schlecht für die Simulation eines Langzeitgedächtnis. Wir wollen nun also den Langzeitspeicher für die Rekursion des Vektors im Hidden-Layers einführen.

Wie zuvor bezeichnet $\overline{h}_t^{(k)}$ den p-dimensionalen Zustand des k-ten Hidden-Layers. Der Einfachheit halber bezeichnen wir den d-dimensionalen Input-Wert $\overline{x}_t$ als $\overline{h}_t^{(0)}$. Wir fügen nun einen neuen p-dimensionalen Vektor $\overline{c}_t^{(k)}$ hinzu, der den Zellstatus repräsentieren soll. Dieser Zellstatus soll nun unser Langzeitgedächtnis simulieren, indem ein Teil der Informationen der früheren Zustände erhalten bleibt und ein anderer Teil mit der Zeit vergessen wird. Wie beim rekurrenten Netzwerk bezeichnet $W^{(k)}$ die Gewichtsmatrix. Diese wird für die Multiplikation der Zeilenvektoren $[\overline{h}_t^{(k-1)}, \overline{h}_{t-1}^{(k)}]^T$ benützt. Durch diese Mulitplikation erhalten wir 4p-dimensionale [1] Vektoren, die wir mit $\overline{i}, \overline{f}, \overline{o}$ und $\overline{c}$ bezeichnen. $\overline{i}$ stellt dabei im Update-Prozess die Input-Variable dar, $\overline{f}$ die Vergessensvarible und $\overline{o}$ die Output-Variable.

Der Update-Prozess besteht nun aus folgenden Schritten:

[1] $W^{(k)}$ hat die Dimension 4p × 2p, diese multipliziert mit $[\overline{h}_t^{(k-1)}, \overline{h}_{t-1}^{(k)}]^T$ ergibt eine Dimension von 4p

- **Einrichten von Zwischenstufen:** Wir wollen die Variablen $\bar{i}, \bar{f}, \bar{o}$ und $\bar{c}$ berechnen. Die ersten drei sind dabei Werte zwischen (0,1). Wobei eigentlich die Idee hinter dem Modell ist, dass diese drei Werte Binärwerte sein sollten, da die Multiplikation von Binärwerten wie die UND-Operation agiert. Die Variable $\bar{i}$ soll zeigen, ob ein neuer Status der Zelle hinzugefügt werden soll, die Variable $\bar{f}$, ob ein Zustand einer Zelle vergessen werden soll und $\bar{o}$, ob Information in einen Hidden-State zugelassen werden soll. $\bar{c}$ soll den neu vorgeschlagenen Zustand der Zelle symbolisieren, wobei $\bar{i}$ und $\bar{f}$ regeln wie stark der Zustand verändert werden darf.

$$\begin{matrix}\text{Eingabewert:}\\ \text{Vergessenswert:}\\ \text{Ausgabewert:}\\ \text{Neuer Status:}\end{matrix} \begin{bmatrix}\bar{i}\\ \bar{f}\\ \bar{o}\\ \bar{c}\end{bmatrix} = \begin{pmatrix}\text{sigm}\\ \text{sigm}\\ \text{sigm}\\ \tanh\end{pmatrix} W^{(k)} \begin{bmatrix}\bar{h}_t^{(k-1)}\\ \bar{h}_{t-1}^{(k)}\end{bmatrix}$$

  Sigm bezeichnet die Sigmoid Aktivierungsfunktion.

- **Selektives Vergessen und Hinzufügen zum Langzeitgedächtnis:** Dieser Prozess besteht aus zwei Teilen. Der erste entscheidet dabei, welche der Zellzustände des vorherigen Zeitschrittes auf 0 zurückgesetzt werden soll. Der zweite Teil entscheidet, ob eine Komponente vom $\bar{c}$ zu unserem Langzeitgedächtnis hinzugefügt werden soll. Dieses Hinzufügen geschieht additiv, weshalb wir kein Problem mit explodierenden Gradienten haben.

$$\bar{c}_t^{(k)} = \bar{f} \odot \bar{c}_{t-1}^{(k)} + \bar{i} \odot \bar{c}$$

  Hier bezeichnet $\odot$ die elementweise Multiplikation.

- **Selektives Durchsickern des Langzeitgedächtnisses zum Zustand des Hidden-Layers:** Nun muss nur mehr der Wert des Hidden-Layers aufgrund der Information in $\bar{o}$ verändert werden. Es kann allerdings auch sein, dass nicht immer die tanh Funktion verwendet wird.

$$\bar{h}_t^{(k)} = \bar{o} \odot \tanh(\bar{c}_t^{(k)})$$

Wie bei den anderen neuronalen Netzwerken, wird auch für das LSTM-Netzwerk die Backpropagation zum Trainieren benützt. Wie bereits im zweiten Punkt erwähnt, hat ein LSTM Netzwerk im Vergleich zu einem feedforward Netzwerk den großen Vorteil, dass die Aktualisierungen nicht mittels Multiplikation, sondern mittels Addition hinzugefügt werden. [10]

3.4 Text Mining

Text Mining ist ein Teilbereich des Data Minings. Unter Data Mining wird der Prozess des Extrahieren von Information aus Daten verstanden. Die Abgrenzung zwischen Machine Learning und Data Mining ist, dass man beim Machine Learning den Algorithmus

betrachtet, der beim Data Mining gebraucht wird, um eine Struktur aus den Rohdaten herauszulesen. [31]
Genauer versteht man unter Text Mining den Prozess von Auslesen von textbasierten Daten. Wir gehen davon aus, dass der Text in natürlicher Sprache geschrieben ist und beschäftigen uns nicht mit künstlicher Sprache, wie Source Code oder mathematischen Gleichungen. Wir gehen hierbei von Wörtern und nicht Buchstaben als kleinsten Bestandteil aus, da Wörter im Allgemeinen im Vergleich zu Buchstaben eine Bedeutung haben.

Unter Stoppwörtern versteht man Wörter, die man nur für die richtige grammatikalische Aneinanderreihung benötigt und denen keine bestimmte Bedeutung zuwächst. Beispiele dazu sind „ein“ oder „die“. Diese Stoppwörter werder beim Text Mining oft entfernt, damit die Wörter, die eine Bedeutung haben, einen größeren Stellenwert bekommen. [12]

3.5 Sentiment Analyse

Der Hintergedanke der Sentiment Analyse ist, dass man aufgrund von geschriebenen Textes in natürlicher Sprache die Meinung über ein Thema von Personen automatisch herausliest. Unter Meinung versteht man hier subjektive Information zu einem Thema. Bereits Anfang der 2000er Jahre war dies eines der Themen, im Bereich der natürlichen Sprachverarbeitung, an dem am meisten geforscht wurde. Doch dieser Teilbereich hat durch den Aufstieg der Sozialen Netzwerke noch einmal deutlich an Brisanz gewonnen. Das Gewinnen von meinungsbezogenen Daten aus den Sozialen Netzwerken öffnet ein großes Tor in die Gedanken und Meinungen anderer. Diese Daten richtig zu verarbeiten und zu analysieren, stellt hier das große Hauptproblem dar. Doch setzt man dies um, kann man komplexe gesellschaftliche Phänomene nicht nur leichter verstehen sondern auch vorhersagen. [32]

Mithilfe von der Sentiment Analyse können nicht nur Unternehmen sondern auch die Endverbraucher profitieren. Für Menschen ist es schwer die große Anzahl an Seiten, an den Meinungen veröffentlicht werden, zu überwachen. Außerdem ist es auch schwer verschiedene Meinungen, die in Artikeln oder Blogs veröffentlicht werden herauszulesen und diese dann zu einer Meinung zusammenzufassen. Deswegen benötigt man eine automatische Analyse. [9]

Doch für Menschen ist das Verstehen und Sprechen von Sprache eines der ersten Dinge, die man im Leben lernt. Den Lauten, die aus den Mündern unseren Mitmenschen kommen, einen Hintergrund und eine Emotion zu geben, fühlt sich für uns natürlich an. Doch das Verstehen von Sprache ist für ein Computer ein schweres Unterfangen. Dabei spielen nicht nur Emotionen, wie Ironie und Sarkasmus, sondern auch die Doppeldeutigkeit von Wörtern eine Rolle. [27]

3.5.1 Sentiment Analyse von Zeitungsartikeln

Zeitungsartikel sind eine stabile Quelle von Information über börsennotierten Gesellschaften. Wissenschaftler sehen einen starken Zusammenhang zwischen den Veränderungen eines Aktienpreises und von veröffentlichten Artikeln. [12]

Online Nachrichtenartikel berichten über Ereignisse, die erst vor kurzem stattgefunden haben. Dadurch ist eine Auswertung über die positiven, neutralen oder negativen Eindrücke, die in dem Artikel niedergeschrieben sind, in Echtzeit möglich. [35]

3.5.2 Sentiment Analyse von Tweets

Das Wissen, das wir aus den Sozialen Netzwerken bekommen können, ist von großer Bedeutung. Doch Texte, die auf den sozialen Netzwerken gepostet werden, unterscheiden sich meist von Zeitungsartikeln. Die folgenden Punkte sind besondere Herausforderungen bei der Analyse von Daten von den Sozialen Netzwerken:

1. **Kurze Texte:** Die geposteten Texte sind meist sehr kurz, aber transportieren in diesen wenigen Zeichen meist eine große Emotion.
2. **Schlechte Grammatik:** Posts sind häufig nicht in richtigen grammatikalischen Sätzen formuliert. Sie verwenden umgangssprachliche Ausdrücke, Abkürzungen, Emoticons, Wortverlängerungen, unregelmäßige Großschreibung und emphatische Ausdrücke.
3. **Verschiedene Sprachen:** Gerade in großen Sozialen Netzwerken, wie Twitter, sind Personen von verschiedenen Ländern und Kontinenten vertreten. Das heißt, diese veröffentlichen auch Texte in verschiedenen Sprachen. So sind weniger als 50 % der Tweets auf Englisch. Der Algorithmus für die Sentiment Analyse muss also auch verschiedenen Sprachen händeln können.
4. **Beziehungen:** Neben einer großen Quelle an Inhalten beinhalten soziale Netzwerke auch viel Informationen über Beziehungen. Diese Beziehungen können einen großen Mehrwert für die Sentiment Analyse bringen. So kann man die Stimmung zu einem Thema, aus den verfügbaren Beziehungsinformationen ableiten.
5. **Dynamik:** In den sozialen Netzwerken kommen Trends sehr schnell auf, Nutzer diskutieren über sie und sind dann oftmals auch schnell wieder vergessen. Eine Herausforderung der Sentiment Analyse ist, die sich ändernden Nutzerinteressen herauszufiltern.

Die Besonderheiten von Sozialen Netzwerken führen also dazu, dass die Algorithmen für Tweets im Vergleich zu den Algorithmen für Zeitungsartikel angepasst werden sollen, um ein optimales Ergebnis zu erzielen. [32]

3.5.3 Ebenen der Sentiment Analyse

Man unterscheidet verschiedene Ebenen, an denen die Sentiment Analyse auf einem Text angewendet wird.

Auf Dokumentenebene
Diese Granularität betrachtet entweder einen Paragraph oder ein gesamtes Dokument. Hier wird davon ausgegangen, dass der gesamte Paragraph oder das gesamte Dokument nur ein Thema beleuchtet. Nach dieser Annahme können im gesamten Text keine Vergleiche zu anderen Themen gezogen werden. Dies führt dazu, dass das Ergebnis der Sentiment Analyse binär oder eine Bewertungskennzahl ist.

Auf Ebene eines Satzes
Wie der Name vermuten lässt, wird hier das Sentiment auf Ebene eines Satzes herausgelesen. Auch hier wird nur von einem Thema ausgegangen, das betrachtet wird. Man kann sich einen Satz als ganz kurzes Dokument vorstellen, das also weniger Information enthält. Es wird sich die Subjektivität und die Objektivität eines Satzes angesehen und dann klassifiziert. Ist der Inhalt subjektiv, wird dieser mittels der Polaritätsklassifizierung als positiv oder negativ eingeteilt. Viele bereits implementierte Machine Learning Algorithmen ergänzen diese Klassen noch durch eine neutrale Klasse.

Auf Ebene eines Gesichtspunktes
Hier wird versucht, den Gesichtspunkt des Objekts (wie zum Beispiel ein Buch oder ein Film, die bewertet werden) von dem Menschen, der die Meinung äußert, zu trennen. Um dies durchzuführen, benötigt man vier Aufgaben:

1. Das Herauslesen der Gesichtspunkte: Hier ist die Hauptaufgabe, eine Liste mit allen Gesichtspunkten zu erstellen. Man nehme an, es handelt sich um eine Rezession eines Laptops und die Rezession lautet 'Ich bin sehr zufrieden mit dem Design, der langen Batterielaufzeit und der schnellen Ladegeschwindigkeit.' Dann sind die drei Gesichtspunkte Design, Batterielaufzeit und die Ladegeschwindigkeit.

2. Die Polarität der Gesichtspunkte erkennen: Hier wollen wir für die Gesichtspunkte, die im ersten Beispiel herausgelesen wurden, herausfinden, ob der Text über sie positiv oder negativ ist. Für das Beispiel aus dem ersten Punkt ist das Ergebnis {Design: positiv}, {Batterielaufzeit: positiv} und {Ladegeschwindigkeit: positiv}

3. Das Erkennen von Kategorien der Gesichtspunkte: Hier wird die behandelte Kategorie aus einer zuvor definierten Menge von Kategorien herausgelesen. Sind zum Beispiel die vordefinierten Kategorien Preis, Design und Performance und der Satz 'Er ist schön aber nicht leistbar', sind die zurückgegebenen Kategorien Design und Preis.

4. Die Polarität der Kategorien der Gesichtspunkte: Im letzten Schritt werden vom Algorithmus die in 3 ermittelten Kategorien in positives, neutrales oder negatives Sentiment eingeteilt. Für das oben angeführte Beispiel ist die Rückgabe {Design: positiv} und {Preis: negativ}

Bei der Sentiment Analyse auf Ebene eines Gesichtspunktes werden gezielt zu verschiedenen Ziel-Gesichtspunkten die dazugehörige Stimmung herausgelesen. [9]

4 Versuchsaufbau und Umsetzung

Die Vorhersage von Aktien ist schwierig. Doch mit der Anwendung von Machine-Learning Modellen, kann das Risiko der Vorhersage minimiert werden. [25] Wir wollen nun die in Kapitel 3 besprochenen Vorgehensweisen anwenden, um die Aktienpreise vorherzusagen. Daraus ergibt sich der folgende Workflow:

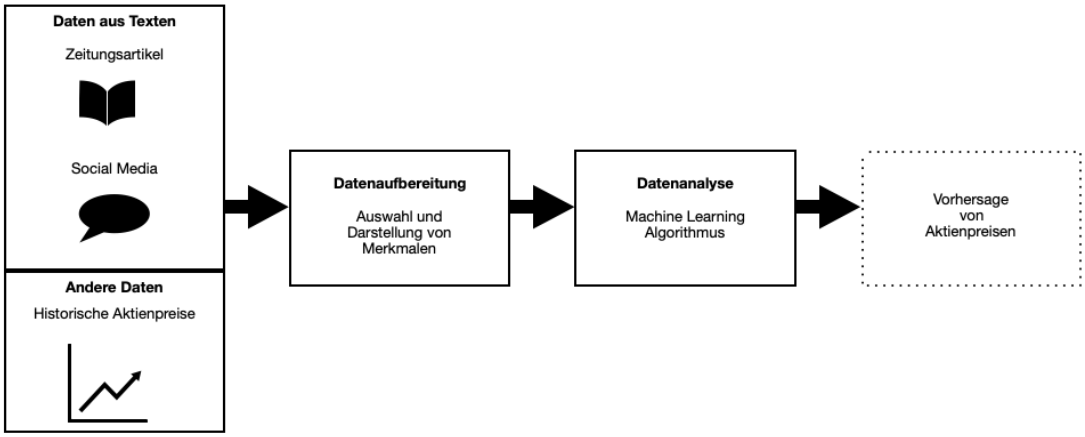


Abbildung 4.1: Workflow für die Vorhersage der Aktienpreise

Das Ziel ist die Aktienpreise für die drei Unternehmen Apple Inc., Tesla Inc. und Biontech SE vorherzusagen. Für diese Vorhersage verwenden wir die Programmiersprache Python, da diese sowohl die Werkzeuge für die Sentiment Analyse als auch für die neuronalen Netze mittels Packages zur Verfügung stellt. Die einzelnen Schritte, die dafür nötig sind, werden in den folgenden Unterpunkten erklärt.

4.1 Data Mining

In diesem Artikel wollen wir uns damit beschäftigen, welche Daten wir für die Vorhersage der Aktienpreise benötigen und woher wir diese bekommen. Wie in Abbildung 4.1 ersichtlich, benötigen wir einerseits die Aktienpreise der Unternehmen und andererseits die Daten aus Texten. Bei den Daten aus Texten werden wir Posts von Twitter und Nachrichtenartikel aus der New York Times verwenden.

Bei Twitter handelt es sich um eine Micro-Blogging-Plattform, bei der jede angemeldete Person öffentlich eine Kurznachricht, einen sogenannten Tweet, mit maximal 140 Zeichen veröffentlichen kann. Die öffentliche Timeline von Twitter ist ein Echtzeit-Informationsstrom mit mehr als einer Million Nachrichten pro Stunde [13], den wir uns für die Vorhersage zu Nutze machen möchten. Auf Twitter wird ein Post Tweet genannt.

4.1.1 Tesla und Apple Twitter Daten

In unserem Versuch benötigen wir für die Sentiment Analyse also die Tweets über Tesla und Apple. Als Quelle wurde die folgende Sammlung von allen Tweets über die Top NASDAQ Unternehmen zwischen 2015 bis 2020 hergenommen:

https://www.kaggle.com/omermetinn/
tweets-about-the-top-companies-from-2015-to-2020

Diese Datei enthält Tweets der Unternehmen Apple, Google, Amazon, Tesla und Microsoft. Insgesamt sind darin mehr als drei Millarden Tweets enthalten. Für Tesla enthält die Datei insgesamt mehr als eine Millarde Tweets und für Apple mehr als 1.4 Millarden Tweets in diesem Zeitraum. Die Daten sind in drei CSV-Dateien gespeichert, die man von Kaggle herunterladen kann. Eine Datei beinhaltet die tatsächlichen Inhalte der Tweets, die zweite die Unternehmen und die dritte Datei enthält das Mapping zwischen den Unternehmen und den Tweets. Im Folgenden werde ich die einzelnen Dateien und den Inhalt ihrer Spalten genauer beschreiben. [2]

Tweet.csv

Wie bereits erwähnt enthält diese Datei die Informationen zu den einzelnen Tweets und hat dabei die folgenden Spalten:

- **tweet_id:** Diese Spalte enthält eine Identifikationsnummer der einzelnen Tweets. Diese Identifikationsnummer wird von Twitter vergeben und ist eindeutig.
- **writer:** In dieser Spalte ist die Identifikation des Nutzers, der den Tweet veröffentlicht hat. Jeder Nutzer hat eine eindeutige Identifikation, kann aber natürlich mehrere Tweets veröffentlichen und ist somit in unserer Tabelle nicht gezwungenermaßen eindeutig.
- **post_date:** Beim post_date handelt es sich um die Anzahl der Sekunden seit der Unixzeit, dem 1. Jänner 1970, 00:00 UTC.
- **body:** Der Inhalt des jeweiligen Tweets ist in dieser Spalte gespeichert. Diese Spalte ist die wichtigste für unsere Aufgabenstellung.
- **comment_num:** Die Anzahl der Kommentare unterhalb des Tweets ist in der Spalte comment_num gesichert.

- **retweet_num:** In dieser Spalte ist gespeichert, wie oft der Tweet von anderen Twitter-Benutzer weitergeleitet worden ist.
- **like_num:** Die Anzahl der Daumen hoch ist in der Spalte like_num gesichert.

Company.csv

In der Datei Company.csv wird zu den Namen der Unternehmen das dazugehörige Kürzel gespeichert.

- **ticker_symbol:** Die Spalte ticker_symbol enthält das eindeutige Kürzel des jeweiligen börsennotierten Unternehmens.
- **company_name:** Der Name des Unternehmens ist in dieser Spalte gesichert.

Company_Tweet.csv

Diese Datei führt die Tweet Daten der Tweet.csv Datei mit den Unternehmensdaten zusammen.

- **ticker_symbol:** Diese Spalte enthält das Unternehmenskürzel aus der Company.csv Datei.
- **tweet_id:** Die eindeutige Identifikationsnummer der Tweet.csv Datei ist in dieser Spalte gespeichert.

4.1.2 Biontech Twitter Daten

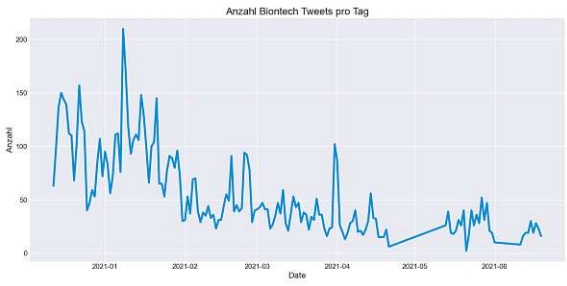
Bei Biontech stößt man auf das Problem, dass vor der Entwicklung des Impfstoffes gegen COVID-19 gemeinsam mit Pfizer sehr wenig über das Unternehmen getwittert wurde. Deshalb werden hier nur Tweets ab der Erstzulassung der Impfung von Biontech und Pfizer für die Sentiment Analyse gewählt. Auch diese Daten werden auf der Webseite Kaggle unter

`https://www.kaggle.com/gpreda/pfizer-vaccine-tweets`

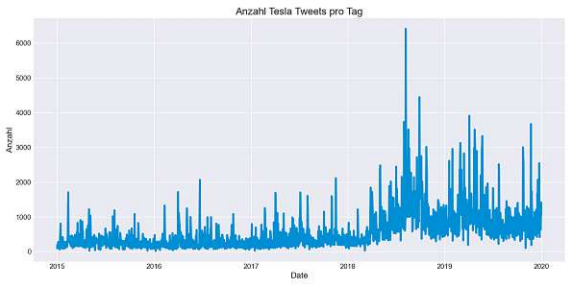
zur Verfügung gestellt. Bei den Daten handelt es sich um eine csv-Datei mit über 8800 Tweets, die zwischen dem 12. Dezember 2020 und dem 19. Juni 2021 gepostet wurden. [3] Die Datei enthält die folgenden Spalten:

- **id:** Diese Spalte enthält, wie die tweet_id Spalte bei der Datei für die Tesla und Apple Daten, die eindeutige Identifikationsnummer der einzelenen Tweets.
- **user_name:** Diese Spalte beinhaltet den Namen der Person, die den Tweet gepostet hat.

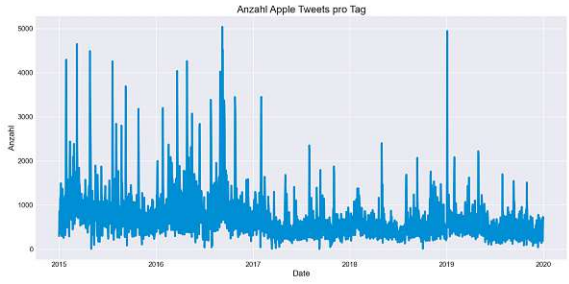
- **user_location:** Jeder Nutzer kann bei seinem Benutzerprofil auf Twitter eine Location festlegen, diese ist in dieser Spalte enthalten. Das Datum hat folgendes Format 'yyyy-mm-dd hh:mm:ss'.
- **user_description:** Ähnlich wie bei der user_location kann jeder Twitter Benutzer eine Beschreibung zu seinem Profil hinzufügen. Diese ist in dieser Spalte zu sehen.
- **user_created:** In dieser Spalte sieht man das Datum, an dem sich der jeweilige Benutzer auf Twitter registriert hat
- **user_followers:** Diese Spalte enthält die Anzahl der Benutzer, dem denjenigen folgen, der den jeweiligen Tweet veröffentlicht hat.
- **user_friends:** Wie vielen Personen der jeweilige Benutzer auf Twitter folgt, ist in dieser Zeile gespeichert.
- **user_favourites:** In dieser Spalte sind die Anzahl der Likes des Benutzers dargestellt.
- **user_verified:** In dieser Spalte ist angegeben, ob der jeweilige Benutzer danach geprüft wurde, ob es sich bei dem jeweiligen Account um die Person handelt, die sie vorgibt zu sein. Man erkennt diese Personen auf Twitter an einem blauen Haken, der neben dem Benutzernamen aufscheint.
- **date:** Bei der Spalte date handelt es sich um das Datum, an dem der Tweet gepostet wurde. Das Format, in dem die Zeiten gespeichert sind, ist wieder folgendes 'yyyy-mm-dd hh:mm:ss'.
- **text:** In dieser Spalte ist der Text des Tweets aufgelistet.
- **hashtags:** Die Hashtags (#), die in dem Tweet gepostet wurden, sind in dieser Spalte gespeichert.
- **source:** In dieser Spalte kann man sehen auf welchem Gerät der Tweet gepostet wurde.
- **retweets:** Die Anzahl, wie oft der jeweilige Tweet von Personen noch einmal gepostet wurde, ist in dieser Spalte zu sehen.
- **favorites:** Die Anzahl der Likes sind in der Spalte favorites zu sehen.
- **is_retweet:** Bei diesem TRUE oder FALSE Wert handelt es sich darum, ob es sich bei dem jeweiligen Tweet selbst, um einen weitergeleiteten Tweet handelt



(a) Biontech



(b) Tesla



(c) Apple

Abbildung 4.2: Anzahl der Tweets pro Tag der einzelnen Unternehmen

4.1.3 Nachrichtenartikel

Um die Nachrichtenartikel zu bekommen, wird die NY Times Search API verwendet. Dabei handelt es sich um eine von den New York Times bereitgestellte API, mit der man zu einem bestimmten Suchbegriff Fakten über Zeitungsartikel bekommt. Um die API zu verwenden, muss man sich bei den New York Times registrieren, um einen persönlichen Schlüssel zu erhalten. Dieser Schlüssel dient als Authentifizierung und muss bei jeder Suchanfrage übergeben werden. Die verwendete Suchanfrage, um Nachrichten zu Tesla zu extrahieren, hat folgende Gestalt

```
url = 'https://api.nytimes.com/svc/search/v2/articlesearch.json?
      q=tesla&facet=true&begin_date='+date_string +'&end_date='+
      date_string +'&api-key=' + key
r = requests.get(url)
```

In der Url unter *q* muss man den Suchbegriff eingeben. Im Falle der Tesla Suchanfrage ist dieser natürlich „tesla“. Unter facets versteht man Felder wie Nachrichtenredaktion oder den Namen des Abschnitts des Artikels, mithilfe dessen man die Abfrage besser filtern kann. Standardmäßig ignoriert die Suchanfrage alle Filter, um diese zu verwenden, muss die Variable facets mit TRUE übergeben werden. Um alle Artikel in einem bestimmten Zeitraum zu durchsuchen, wird mit Hilfe einer for-Schleife jedes Datum durchlaufen. Die-

ses Datum muss das Format “yyyymmdd“ haben. Für den ersten März 2020 ist das zum Beispiel “20200301“. Für die Suchanfrage war das Beginndatum begin_date und das Enddatum end_date jeweils dasselbe Datum. Es wird also immer nur ein Tag durchsucht. Am Schluss muss der personalisiert Schlüssel übergeben werden. Wird eine richtige Abfrage abgesendet, bekommt man eine JSON Datei zurück. Stimmt etwas bei der Abfrage nicht, gibt es die Fehler “401: Unauthorized request.“ und “429: Too many requests“. Bei ersterem gibt es ein Problem mit dem API-Schlüssel und es muss überprüft werden, ob dieser korrekt eingegeben und übergeben wurde. Bei letzterem Fehler wurden zu viele Abfragen in zu kurzer Zeit gestellt. Die API erlaubt nur 6 Abfragen pro Minute und 4000 Abfragen pro Tag. Um dieses Limit nicht zu überschreiten, wird im Code ein Timer eingebaut, der nach jeder Abfrage sechs Sekunden wartet. [4]

4.1.4 Aktienkurse

Für die Vorhersage der Aktienpreise werden natürlich auch die historischen Aktiekurse benötigt. Einerseits als Input-Daten für das neuronale Netzwerk, andererseits um die Akkuratheit der vorhergesagten Ergebnisse zu messen. Ein einfacher Weg, um an die Aktienkurse von Unternehmen in Python zu kommen, ist die yfinance Bibliothek von Yahoo Finance. Diese beinhaltet aktuelle und historische Aktienpreise zu verschiedenen Zeitpunkten. [26] Um die historischen Aktienpreise eines Unternehmens herunterladen zu können, wird folgender Befehl verwendet

```
apple = yf.download('AAPL', start='2015-01-01', end='2019-12-31')
```

Dabei werden von der yfinance Bibliothek, die hier mir dem Kürzel yf versehene ist, die Aktienpreise von Apple vom 1. Jänner 2015 bis zum 31. Dezember 2019 heruntergeladen. Bei diesem Befehl erhalten wir ein DataFrame - das ist ein Datenobjekt in Python - mit 1258 Zeilen, das folgende Gestalt hat:

Date	Open	High	Low	Close	Adj Close	Volume
2014-12-31	28.205000	28.282499	27.552500	27.594999	25.057606	165613600
2015-01-02	27.847500	27.860001	26.837500	27.332500	24.819241	212818400
2015-01-05	27.072500	27.162500	26.352501	26.562500	24.120045	257142000
2015-01-06	26.635000	26.857500	26.157499	26.565001	24.122320	263188400
2015-01-07	26.799999	27.049999	26.674999	26.937500	24.460564	160423600
...	...	...	...	...	...	...

Tabelle 4.1: Apple Aktienpreise

Für die Vorhersage der Aktienpreise werden wir die Close-Preise verwenden.

4.2 Datenaufbereitung und Speicherung

Wir benötigen nun ein geeignetes System, um die gesammelten Daten zu verwalten. Dafür benötigen wir ein System, das uns die Daten effektiv speichert und bei dem der Zugriff auf die Daten sich leicht lösen lässt.

Dafür wird eine SQLite Datenbank verwendet. Bei SQLite handelt es sich um ein Management System von Relationalen Datenbanken (RDBMS). Diese RDBMS helfen, große Menge an Daten in tabellarischer Form zu speichern. Das Lite im Namen SQLite bezieht sich jedoch in keiner Weise auf die Fähigkeiten des Systems, sondern auf den geringen systematischen Anforderungen und den geringen Verbrauch an Resourcen. Für die Verwendung einer SQLite Datenbank hat die Tatsache gesprochen, dass kein Server von Nöten ist und eine Datei die gesamte Datenbank enthalten kann. Diese Datei enthält also sowohl die Architektur der Datenbank, als auch die Daten selbst. [24]

Der Aufbau der Datenbank sieht mit den in 4.1 gesammelten Daten wie folgt aus:

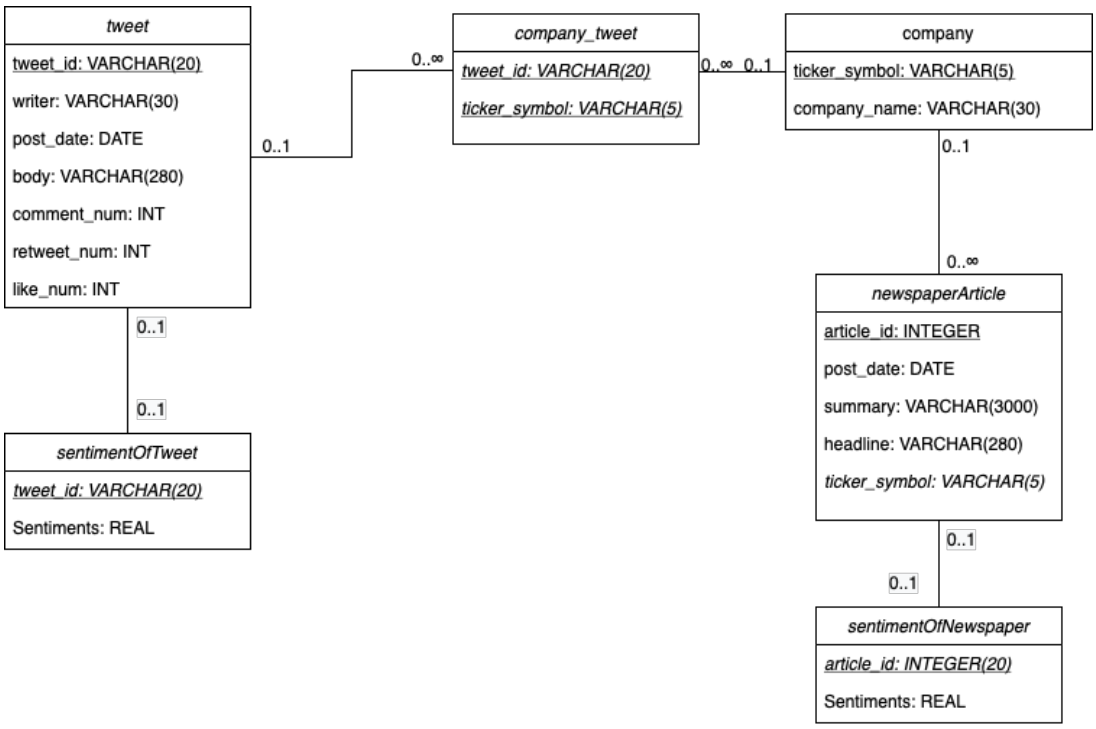


Abbildung 4.3: Diagramm des Aufbaus der SQLite Datenbank

4.3 Sentiment Analyse

Nachdem wir nun alle unsere Daten in einer Datenbank gespeichert haben, können wir uns um die Sentiment Analyse der Nachrichtenartikel und Tweets kümmern. Die Vorgehensweise unterscheidet sich für diese zwei Punkte, aufgrund der Herausforderung bei der Analyse der Daten aus Sozialen Netzwerke.

4.3.1 Sentiment Analyse der Tweets:

Um die Sentiment Analyse durchzuführen, werden die Tweets aus der Datenbank ausgelesen. Danach werden die Daten bereinigt und nach [5] aufbereitet. Zuletzt werden die Sentiment Werte wieder in der Datenbank gespeichert. Um die Twitterdaten bestmöglich zu klassifizieren, werden die Daten zunächst von Verunreinigungen, die bei der Sentiment Analyse stören, bereinigt. Hierfür wird besonders das Python Package Re verwendet, mit dem man leicht Strings verändern kann. In diesem Paket ist vor allem die Funktion sub für uns von großer Bedeutung. Diese Funktion, die auf einen Text angewendet wird, übernimmt zwei Argumente, wobei die Zeichen, die dem Muster des ersten Arguments entsprechen durch das zweite Argument ersetzt werden.

Die Codeteile zur Bereinigung der Daten sind im Folgenden abgebildet und erklärt:

```
tweets.body = tweets.body.apply(lambda x:re.sub('@[^\s]+','',x))
tweets.body = tweets.body.apply(lambda x:re.sub(r'\B#\S+','',x))
tweets.body = tweets.body.apply(lambda x:re.sub(r"http\S+", "", x))
```

Bei der Bereinigung der Daten werden zuerst Twitter Handler oder Usernamen entfernt. Bei Twitter beginnen Usernamen mit dem Zeichen @ und deswegen werden dieses Zeichen mit den nachfolgenden Buchstaben bis zum nächsten Leerzeichen entfernt. Das gleiche Verfahren wird auch bei den Hashtags der Tweets, die mit dem Zeichen # beginnen, angewendet. Auch in Tweets geteilte Links sind bei der Analyse der Tweets ein Störfaktor und gehören deswegen entfernt. Links starten meist mit der Zeichenfolge http und enthalten kein Leerzeichen. Deswegen wird eine Zeichenfolge beginnend mit http bis zum nächsten Leerzeichen gelöscht.

```
tweets.body = tweets.body.apply(lambda x:
        re.sub(r'\s+[a-zA-Z]\s+', '', x))
tweets.body = tweets.body.apply(lambda x:
        re.sub(r'\s+', ' ', x,flags=re.I))
```

Danach werden noch Buchstaben, die einzeln stehen - also die in der Mitte von zwei Leerzeichen stehen - entfernt. Außerdem werden mehrere Leerzeichen hintereinander durch nur ein Leerzeichen ausgetauscht.

Nun sind die Daten bereinigt und die Sentiment Analyse wird mittels „Valence Aware Dictionary and Sentiment Reasoner“ (VADER) durchgeführt. Bei VADER handelt es sich um ein regelbasiertes Werkzeug, um Sentiment Analyse durchzuführen. VADER ist dabei besonders auf Soziale Netzwerke ausgerichtet und berücksichtigt auch die Reihenfolge der Wörter. [20] Zu den Vorteilen von VADER gehört, dass die Bedeutung von Emojis und Smileys interpretiert werden können. Außerdem wird zwischen verschiedenen Schreibstilen von Wörtern beziehungsweise Wortgruppen unterschieden. Ein Beispiel hierfür ist, wenn einzelne Wörter oder Wortgruppen in GROSSBUCHSTABEN geschrieben werden, um ihnen mehr Gewicht zu verleihen. Des Weiteren werden öfter aufeinanderfolgende Interpunktionen beachtet und interpretiert. Ein weiterer Vorteil von VADER für die Sentiment Analyse von Tweets ist, dass Slang-Wörter und Akronyme (wie zum Beispiel lol) verarbeitet werden können. [21]

Das Ergebnis einer Analyse mittels VADER besteht aus vier Werten, einem positiven, negativen und neutralen Wert und dann zusätzlich noch aus einem zusammengesetzten, dem compound Wert. Die ersten drei Werte addieren sich auf 1. Der zusammengesetzte Wert wird hingegen durch Summieren der Wertigkeiten der einzelnen Bestandteile des Textes berechnet. Danach wird er auf das Intervall [-1,1] skaliert. Wobei ein Wert von -1 für extrem negativ und ein Wert von +1 für extrem positives Sentiment steht. Um Texte in positiv, negativ oder neutral einzuteilen, werden meist die folgenden Schwellen hergenommen

- **positiv:** compound-Wertung $\geq$ 0.05
- **neutral:** -0.05 $\leq$ compound-Wertung $\leq$ 0.05
- **negativ:** compound-Wertung $\leq$ -0.05

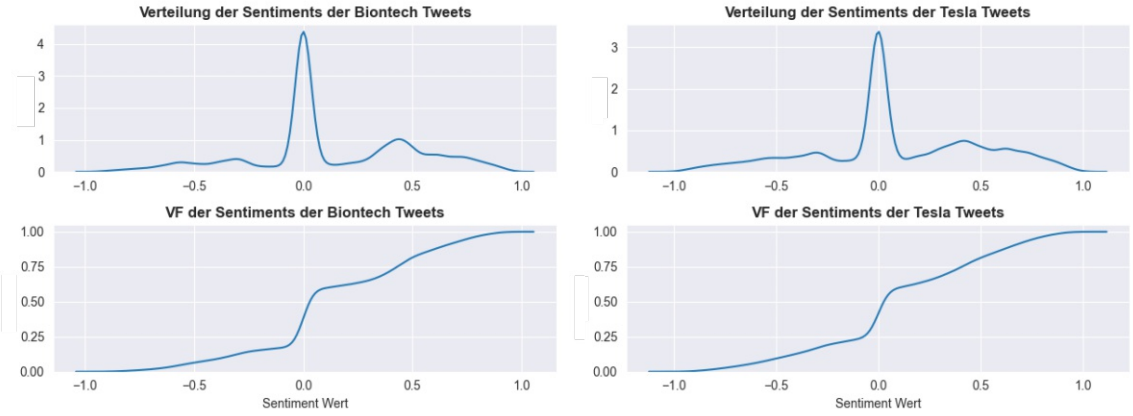
Bei der positiven, negativen oder neutralen Einteilung wird das Verhältnis des Satzes, das in eine bestimmte Kategorie passt, berechnet. [21]
Zur Veranschaulichung wurden verschiedene Sätze mittels dem VADER Sentiment Analyzer bewertet:

	Satz	Ergebnis
1	My new MacBook is amazing	'neg': 0.0, 'neu': 0.513, 'pos': 0.487, 'compound': 0.5859
2	I hate my new IPhone	'neg': 0.552, 'neu': 0.448, 'pos': 0.0, 'compound': -0.5719
3	The battery life of my new MacBook is not amazing	'neg': 0.254, 'neu': 0.746, 'pos': 0.0, 'compound': -0.4717
4	The battery life of my awesome MacBook is not amazing	'neg': 0.202, 'neu': 0.527, 'pos': 0.27, 'compound': 0.2565
5	I bought an iPad	'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0

Tabelle 4.2: Beispiele zur Sentiment Analyse

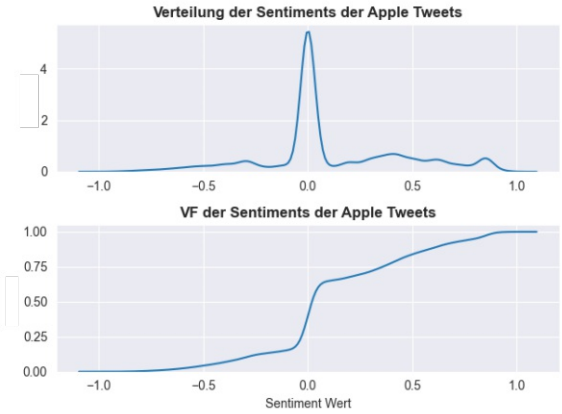
In diesen Beispielen kann man gut sehen, dass die compound Wertung von (1) und (2) entgegengesetzt ist, was auch dem menschlichen Empfinden der Sätze entspricht. Schreibt man bei (1) amazing in Großbuchstaben erhöht sich der compound-Wert auf 0.6739. Des Weiteren erkennt VADER, dass der Satz weniger negativ ist, wenn nur die Batterielaufzeit des Laptops schlecht bezeichnet wird und nicht das ganze Gerät (3). An (4) sieht man, dass der großartige Laptop, die nicht so gute Batterielaufzeit übertrumpft, da der compound Wert positiv ist. Allerdings ist die Gesamtwertung nicht so gut wie in (1). Dies entspricht dem allgemeinen Sprachverständnis. Wie man in (5) sieht, erkennt VADER auch einen neutral formulierten Satz.

Wir wollen nun die durch die Sentiment Analyse generierten Werte für die einzelnen Unternehmen vergleichen. Dazu betrachten wir zuerst die Dichte und die Verteilungsfunktion der einzelnen Unternehmen.



(a) Biontech

(b) Tesla



(c) Apple

Abbildung 4.4: Dichte und VF der Sentiments der Tweets über die einzelnen Unternehmen

In Abbildung 4.4 kann man erkennen, dass bei allen drei Unternehmen die meisten Tweets als neutral eingestuft werden. Außerdem erkennt man, dass bei allen drei Unternehmen mehr positive als negative Tweets abgesetzt wurden.

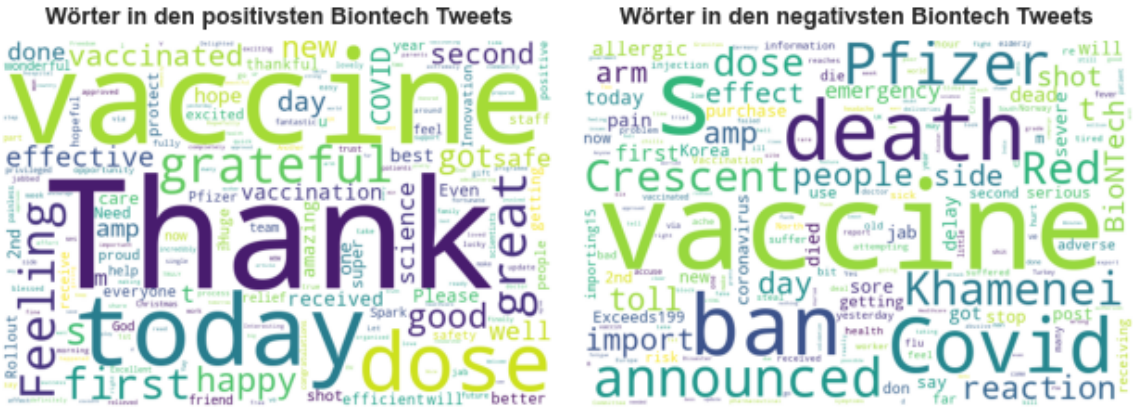
Eine weitere gute Möglichkeit, unsere Ergebnisse zu illustrieren, liefern uns sogenannte WordClouds. Diese WordClouds helfen uns die Themen für die einzelnen Unternehmen herauszufinden, über die positiv oder negativ geschrieben wurde. Bei einer WordCloud handelt es sich um ein Python Paket, mit dem man die häufigsten Wörter analysiert und diese dargestellt werden. Je häufiger ein Wort vorkommt, desto größer ist es geschrieben. [6]

Um dieses WordCloud für die positiven Tweets zu erstellen, werden jene Tweets herausgelesen, bei denen der positive Sentiment Wert zwischen 0,4 und 1 liegt. Für die negative WordCloud werden jene ausgewählt, deren negativer Sentiment Wert zwischen 0,25 und

1 ist. Danach werden alle ausgewählten Tweets zu einem langen Text zusammengefügt. Dieser Text muss danach noch dem Paket übergeben werden.

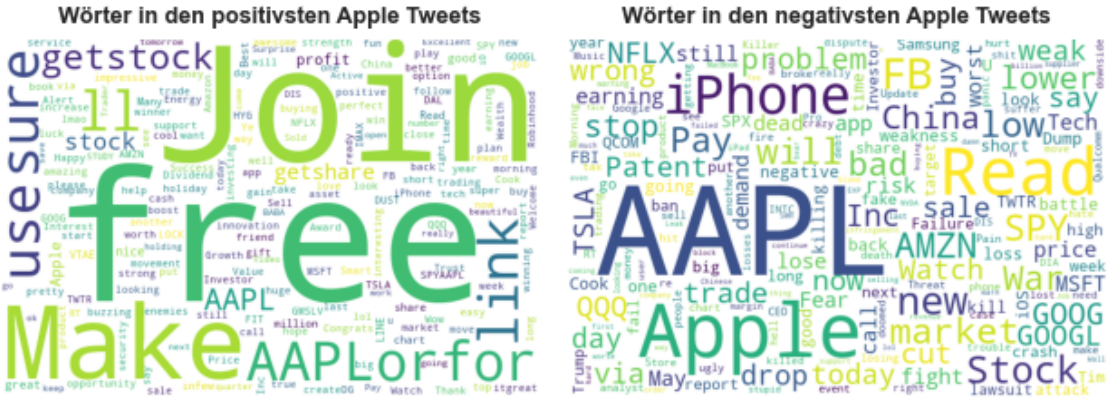
```
pwc = WordCloud(width=600, height=400, collocations=False,
        background_color='white').generate(Most_Positive_text)
```

Mittels width und height lässt sich die Größe des Bildes, das generiert wird, einstellen. Mittels collocation kann man einstellen, ob bestimmte Zwei-Wort-Kombinationen (Bigramme) dargestellt werden sollen, oder nicht. Ein Beispiel für ein solches Bigramm ist „super cool “. Mit dem letzten Argument wird der Hintergrund des Bildes weiß eingefärbt. Die Word-Clouds für die Tweets unserer drei Unternehmen sehen so aus:



(a) Biontech

(b) Tesla



(c) Apple

Abbildung 4.5: WordCloud der Tweets über die einzelnen Unternehmen

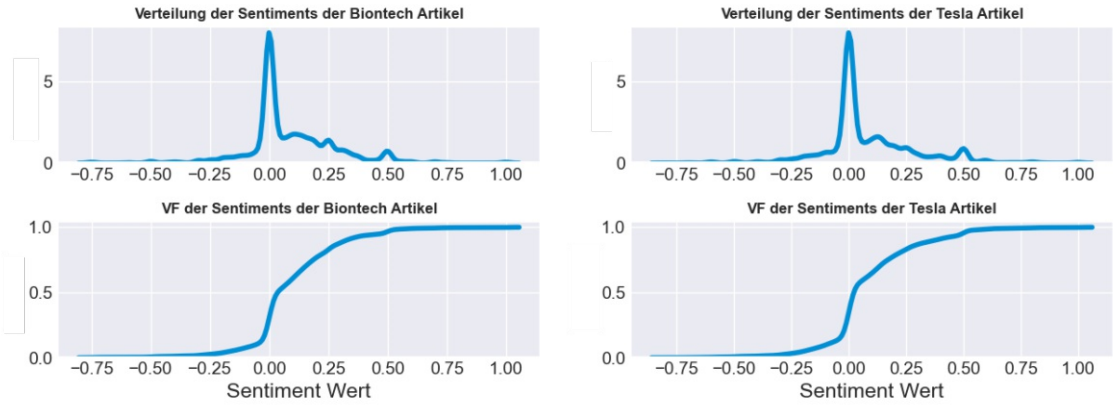
4.3.2 Sentiment Analyse der Nachrichtenartikel:

Da VADER besonders für Texte aus Sozialen Netzwerken geeignet ist, wird es für die Zeitungsartikel nicht verwendet. Stattdessen kommt die Python Bibliothek TextBlob zum Zug. Dabei handelt es sich, wie bei VADER, um eine Bibliothek, die auf NLTK aufbaut. Mithilfe von TextBlob kann man unter anderem Texte klassifizieren, das Sentiment analysieren und Phrasen extrahieren. Bei der Sentiment Analyse von TextBlob übergibt man einen Text und bekommt als Rückgabewert zwei Floatwerte, die die Polarität und die Subjektivität des Textes repräsentieren sollen. Der Polaritätswert liegt innerhalb von [-1;1] und die Subjektivität zwischen [0;1]. [1] Da wir die allgemeine Stimmungslage in den Nachrichtenartikeln verwenden wollen, wird nur der Polaritätswert hergenommen.

Das folgende Beispiel soll die Verwendung mit TextBlob illustrieren. Am 20. April 2021 verunglückte ein Tesla in der Nähe von Housten, Texas und zwei Personen starben bei diesem Unfall. Die New York Times berichtete darüber und das Abstrakt dieses Artikels war *'Two people died in the crash, and officials said no one had been behind the wheel. Elon Musk, Tesla's chief executive, said recovered data showed "Autopilot was not enabled" in the vehicle.'* [8] Wir wollen nun das Sentiment dieses Abstrakts, das in der Variable text gespeichert ist, bestimmen.

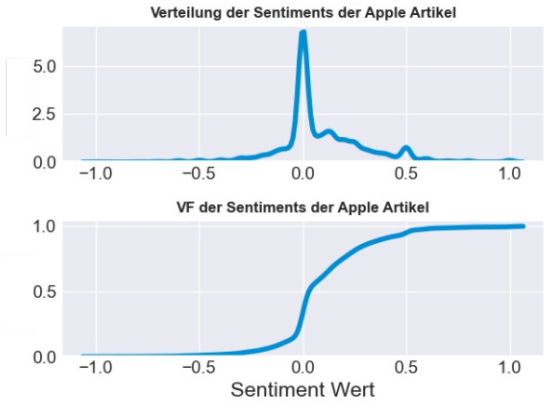
```
tb  = TextBlob(text)
tb.sentiment.polarity
```

Der zugehörige Polaritätswert ist -0,4. Um auch die Ergebnisse der Nachrichtenartikel besser vergleichen zu können, werden auch diese aufbereitet. Wie bei den Tweets wollen wir auch hier die erhaltenen Werte genauer ansehen. Zuerst betrachten wir wieder die Verteilungsfunktion:



(a) Biontech

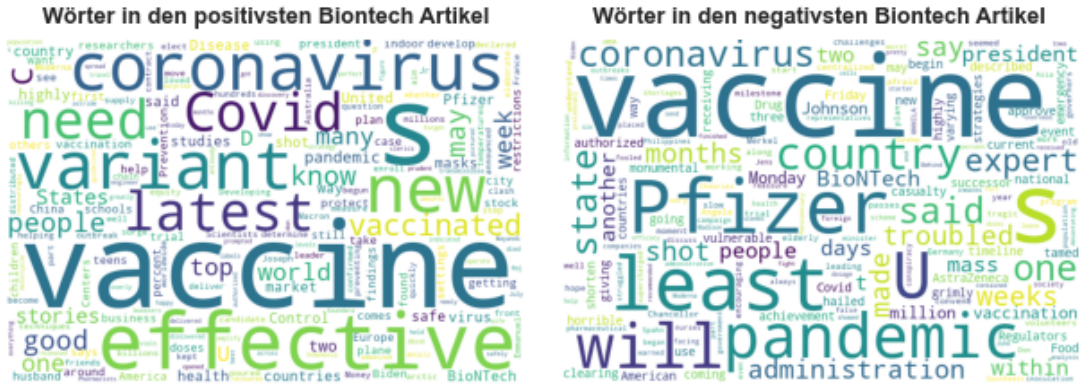
(b) Tesla



(c) Apple

Abbildung 4.6: Dichte und VF der Sentiments der Nachrichtenartikel über die einzelnen Unternehmen

Auch in Abbildung 4.6 kann man erkennen, dass die geschriebenen Artikel in der NY Times über die Unternehmen eher positiver waren als negativ. Auch für die Nachrichtenartikel wird wieder für die positivsten und für die negativsten Artikel eine WordCloud erstellt, um die Themen, die zu der positiven beziehungsweise negativen Bewertung beigetragen haben, besser analysieren zu können und wir werden sie für unsere Zwecke vernachlässigen.



(a) Biontech

(b) Tesla

(c) Apple

Abbildung 4.7: WordCloud der Artikel über die einzelnen Unternehmen

Wenn man die klein geschriebenen Wörter in Abbildung 4.7c genauer betrachtet, fällt auf, dass sich ein paar Rezepte unter die untersuchten Artikel eingeschrieben haben. Dies resultiert aus dem Suchbegriff Apple. Da diese aber nur sehr klein geschrieben sind, kommen

diese Wörter nicht recht häufig vor.

4.4 Klassische Statistische Verfahren

Wir haben nun alle Daten und die Sentiment Werte der Nachrichtenartikel und der Posts aus den sozialen Netzwerken. Im nächsten Schritt werden wir mit den Daten ein neuronales Netz aufbauen, um die Aktienpreise vorherzusagen.
Die klassischen statistischen Verfahren werden mittels Keras durchgeführt. Keras ist ein einfach zugängliches und einfach zu verwendendes Framework zur Anwendung von Deep Learning. Keras ist eine TensorFlows High-Level-API und basiert auf Tensorflow 2.0. [17] Bei Tensorflow handelt es sich um eine plattformübergreifende Schnittstelle zum Implementieren und Ausführen von Machine Learning Algorithmen. Diese wurden vom Google Brain Team zuerst für den internen Gebrauch entwickelt, wurden dann aber Open Source zur Verfügung gestellt. [33]

Um das neuronale Netzwerk zu konfigurieren, werden zuerst die historischen Aktienpreise, wie in Kapitel 4.1.4 beschrieben, heruntergeladen. Für die Vorhersage werden die Close-Preise verwendet. Diese werden dann zwischen -1 und 1 skaliert und in einen Teil Trainingsdaten und in einen Testdaten-Teil eingeteilt, wobei 75 % der Daten die Trainingsdaten darstellten und die restlichen 25 % die Testdaten. Bei dieser Einteilung wird allerdings darauf Rücksicht genommen, dass eine bestimmte Anzahl von Tagen an historischen Aktienpreisen als Eingabeparameter ins Modell übergeben wird. Das heißt, in den Testdaten werden um diese Anzahl von Tage mehr Daten übergeben.

```
lengthOfTraining = math.ceil(len(closearray) * 0.75)
scaler = MinMaxScaler(feature_range=(-1, 1))
scaledclose = scaler.fit_transform(closearray)
traindata = scaledclose[0:lengthOfTraining, :]
testdata = scaledclose[lengthOfTraining - days:, :]
```

Danach wird aus den SQLite Datenbanken die jeweiligen Sentiment Werte des Tages zu einem Mittelwert aggregiert, wobei die jeweils die Nachrichtenartikel und die Tweets zu einem eigenen Wert zusammengefasst werden. Zusätzlich werden noch die Anzahl der Tweets an einem Tag gezählt, um die Brisanz des jeweiligen Unternehmens in das Modell einfließen zu lassen. Der Vorteil der Sentiment-Werte ist, dass diese schon im Spektrum zwischen -1 und 1 liegen und deshalb für die Verwendung im neuronalen Netz nicht mehr skaliert werden müssen. Die Anzahl der Tweets muss jedoch wieder skaliert werden, damit die Werte im Intervall [-1;1] angesiedelt sind.

Der nächste Schritt ist, die Test- und Trainingsdaten der Aktienpreisen mit den jeweiligen Sentimentwerten und der Anzahl der Tweets zu den finalen Test- und Trainingsdaten, zu-

sammenzufügen. Dazu wird in einer Schleife jedes Datum bei den Close Tagen durchlaufen und mit den zugehörigen Werten zu Arrays zusammengefügt und in die richtige Form gebracht.

Nun können wir mit Hilfe dieser Array ein Modell aufbauen. Der folgende Code ist ein Beispiel, da wir im nächsten Kapitel verschiedene Konfigurationen für das Modell betrachten.

```
model = Sequential()
model.add(LSTM(50, return_sequences=True, activation='relu',
input_shape = (x_train.shape[1],1)))
model.add(Dropout(0.2))
model.add(LSTM(50, activation='relu', return_sequences=False))
model.add(Dense(25,  activation='relu'))
model.add(Dense(1))
model.compile(optimizer='adam',loss='mean_squared_error')
model.fit(x_train,y_train, batch_size=1, epochs=20)
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)
```

Wir errichten ein Modell mit linearer Anordnung mit dem Befehl Sequential(). Nun können die einzelnen Layer mit dem Befehl .add() hinzugefügt werden, wobei die einzelnen Layer für den eigenen Zweck konfiguriert werden können. Der erste und der zweite Layer sind in diesem Beispiel LSTM-Layer. Wollen wir keine LSTM-Layer, werden nur Layer mit dem Befehl Dense() hinzugefügt. Die Anzahl der Knoten für den Layer wird mit dem ersten Parameter mitgegeben. Da die Input-Daten mehrdimensional sind, muss im ersten Layer die Form der Trainingsdaten in input_shape übergeben werden. Mit dem Befehl activation kann die gewünschte Aktivierungsfunktion gewählt werden. Hier ist standardmäßig die lineare Aktivierungsfunktion eingestellt. Mit dem Befehl Dropout() kann man dem Befehl des overfitting im LSTM Modell entgegenwirken. Zusammenfassend gesagt ist dieses Beispielmodell ein Modell mit vier Layern. Die ersten zwei Layer sind LSTM Layer mit 50 Knoten je Layer. Danach kommen zwei feedforward Layer, wobei der erste 25 Knoten hat und der letzte nur einen Knoten. Bei den ersten drei Layern ist die Aktivierungsfunktion die ReLU.

Ist das Modell nach den eigenen Wünschen erstellt, kann es mit .compile() konfiguriert werden. Hier wird als Parameter für den Optimizer Adam übergeben. Als Verlustfunktion wird in diesem Beispielmodell der mittlere quadratische Fehler gewählt.

Nun kann das Modell iterativ mit dem Befehl .fit() durchlaufen werden. Die Anzahl der Durchläufe mit den Trainingsdaten kann mit dem Parameter epochs eingestellt werden. Die

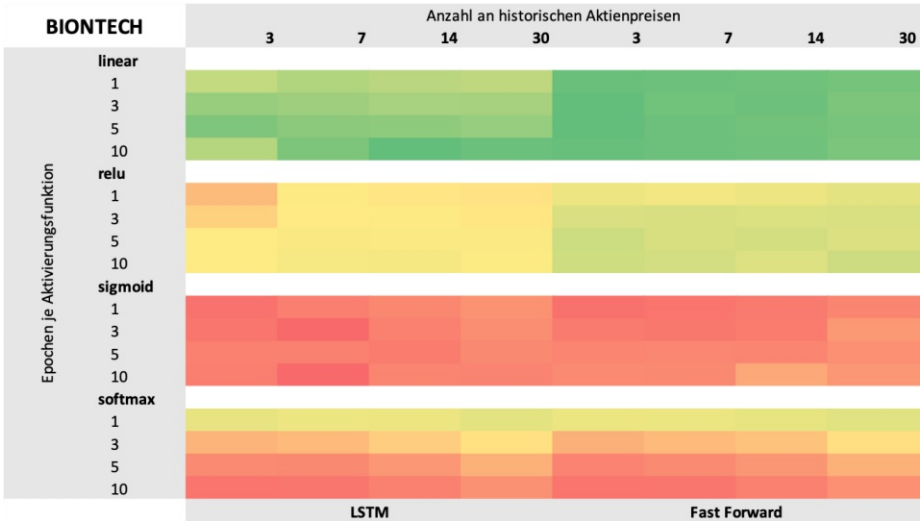
Input-Trainingsdaten und auch Vergleichswerte für die Trainingsdaten müssen zusätzlich noch übergeben werden.

Nun ist das Modell trainiert und kann auf andere Daten angewendet werden, dies geschieht mit dem Befehl .predict(). Nachdem die vorgesagten Werte aus dem Modell kommen, müssen diese wieder auf die ursprüngliche Größe skaliert werden.

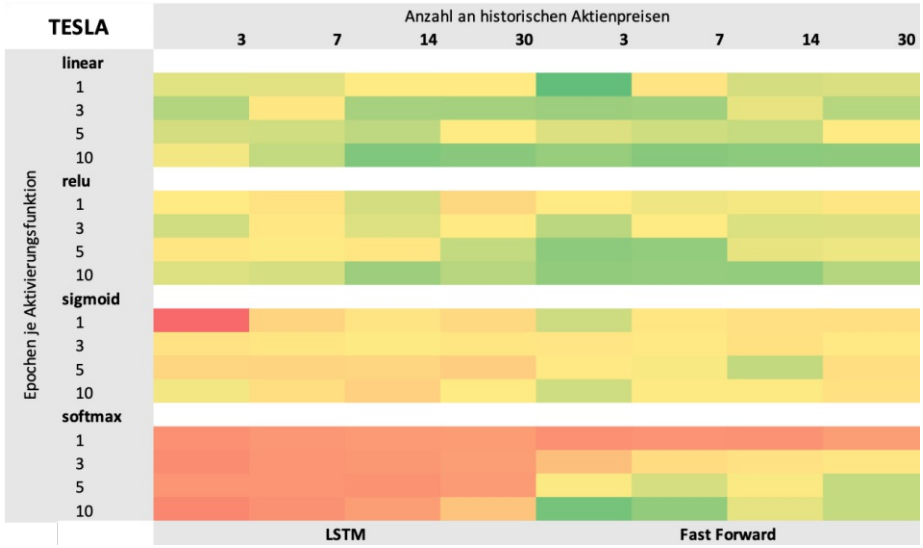
5 Ergebnisse

Wie in Kapitel 3 vorgestellt ist ein LSTM Modell eine gute Möglichkeit Zeitreihen zu modellieren. Da es sich bei den historischen Aktienkursen um Zeitreihen handelt, ist dieses Modell keine schlechte Wahl. Da wir allerdings nicht nur Zeitreihen verwenden, sondern auch die Sentiment Werte der Tage, ist es nicht mehr sicher, ob das LSTM Modell die richtige Wahl ist. Um dies herauszufinden, werden für die drei ausgewählten Unternehmen Apple Inc., Tesla Inc. und Biontech SE sowohl LSTM Modelle durchlaufen als auch reine feedforward neuronale Netzwerke und die Ergebnisse anschließend gegenübergestellt.

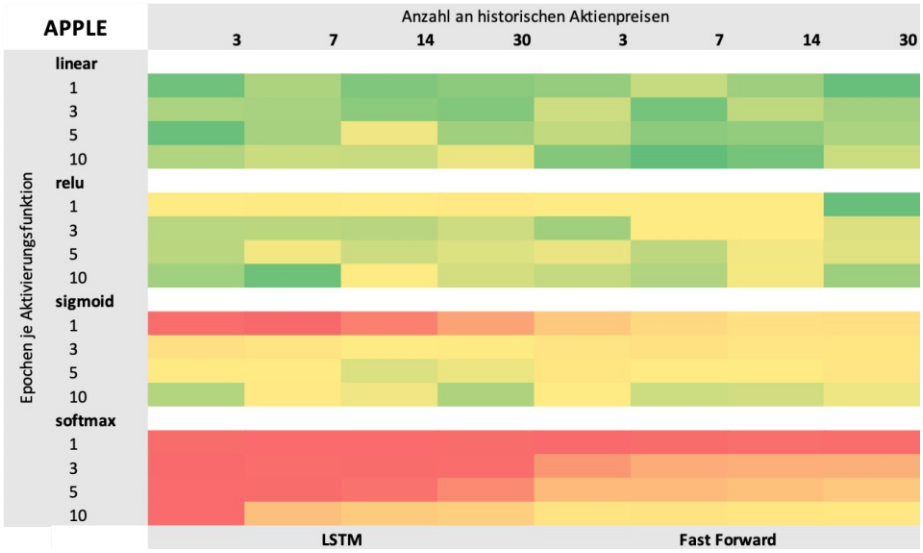
Um die Netzwerke bestmöglich zu konfigurieren, werden verschiedene Konfigurationen für jeweils das LSTM und das reine feedforward neuronale Netz getestet. Für alle drei Unternehmen werden die Anzahl der vergangenen Closepreise, die in das Netzwerk eingespielt werden, die Anzahl der durchlaufenen Epochen und die Aktivierungsfunktionen variiert. Die verschiedenen Anzahl der Closepreise, die getestet werden, sind 3, 7, 14 und 30. Es werden außerdem die lineare, ReLU, Softmax und Sigmoid Aktivierungsfunktionen ausprobiert. Die verschiedenen Epochen sind 1, 3, 5 und 10. Für jede Konfiguration wird das Modell 10 mal aufgestellt und durchlaufen, um daraus einen Mittelwert des mittlernen quadratischen Fehlers pro Modelllauf zu bilden. Allerdings wird die Anzahl der Layer und Knoten aufgrund vorheriger Experimente auf fünf Layer mit 50, 50, 40, 25 und 1 Knoten beim feedforward neuronalen Netz und vier Layer mit 50, 50 , 25 und 1 beim LSTM Modell festgelegt. Außerdem wurde bei diesen Experiment festgestellt, dass die Ergebnisse mit Dropout besser als die Ergebnisse ohne Dropout sind. Deswegen wird kein Dropout beim LSTM Modell verwendet. Als Optimizer wird der Adam Optimizer gewählt. Mithilfe einer Heatmap werden die Ergebnisse von den LSTM Läufen und den feedforward Läufen gegenübergestellt.



(a) Biontech



(b) Tesla



(c) Apple

Abbildung 5.1: Heatmap zur Gegenüberstellung der Ergebnisse von den LSTM und den feedforward neuronalen Netz für die einzelnen Unternehmen

In Abbildung 5.1 entspricht die grüne Farbe, die Ergebnisse mit den besten mittleren quadratischen Fehlern und die rote Farbe Ergebnisse mit den größten mittleren quadratischen Fehlern. Bei allen drei Unternehmen kann man in Abbildung 5.1 sehen, dass die Ergebnisse des reinen feedforward neuronalen Netzwerks besser sind als jene des LSTM Netzwerkes. Außerdem lässt sich erkennen, dass die Ergebnisse mit der ReLU und der linearen Aktivierungsfunktion die besten Ergebnisse liefern. Vergleicht man nun allerdings die Ergebnisse der feedforward neuronalen Netze der drei Unternehmen mit einer Heatmap erkennt man, dass die Güte des Netzwerkes maßgeblich mit der verfügbaren Historie von Daten zusammenhängt.

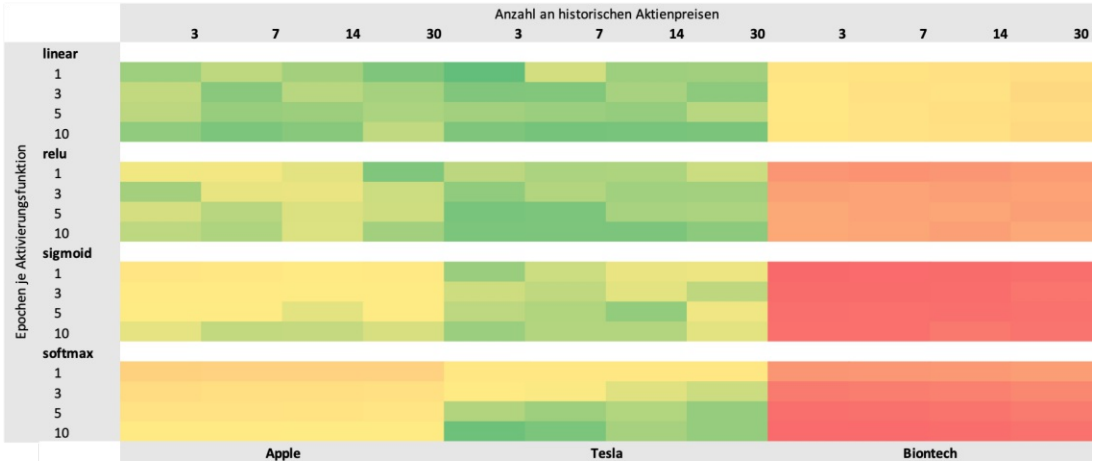


Abbildung 5.2: Heatmap für den Vergleich der drei Unternehmen

In Abbildung 5.2 sieht man, dass die Vorhersage für Tesla und Apple um einiges besser sind, als die Vorhersage von Biontech. Allerdings ist auch die Datenhistorie von fünf Jahren von Tesla und Apple um einiges länger als die Datenhistorie von etwa 6 Monaten von Biontech. Doch nicht nur die Datenhistorie ist bei den ersteren besser, in Abbildung 4.2 ist ersichtlich, dass auch die Anzahl der Tweets pro Tag für die Unternehmen sehr unterschiedlich ist. Des Weiteren handelt es sich bei der Gesamtsituation von Biontech um eine Besonderheit. Die Gegebenheit ist durch die COVID19 Pandemie eine besondere und auch sehr abhängig von den politischen Entscheidungen in vielen Ländern und das Unternehmen Biontech kam erst durch die Pandemie und die Impfstoffentwicklung in Zusammenarbeit mit Pfizer auf das Radar von vielen. Dadruch ist vor Dezember 2020, wo die Erstzulassung des Coronavirusimpfstoffes erfolgte [1], zu wenig über Biontech in den sozialen Netzwerken geschrieben worden, um eine adäquate Historie zu bekommen.

Um die Ergebnisse besser einordnen zu können, werden für Apple und Tesla die Ergebnisse mit den vorhergesagten Aktienpreisen mit den tatsächlichen Aktienpreisen geplottet. Für die Analyse wird die Konfiguration genommen, die die besten Ergebnisse geliefert haben.

Die besten Ergebnisse für Tesla hat das feedforward neuronale Netzwerk geliefert. Dabei werden jeweils die drei letzten Aktienkurse in das Netzwerk mit linearen Aktivierungsfunktionen eingespielt und nur eine Epoche durchlaufen. In Abbildung 5.3 sieht man, wie akkurat die Vorhersage der Aktienpreise war. In diesem Durchlauf wurde ein mittlerer quadratischer Fehler von 0,1647043410380175 erzielt. Für den Plot wurde dieses Ergebnis genommen, da dieser mittlerer quadratische Fehler sehr nah an dem besten durchschnittli-

[1] Erstzulassung des mRNA-Impfstoff BNT162b2 von Biontech und Pfizer in Großbritannien [7]

chen mittleren quadratischen Fehler von 0,18294 liegt. Somit zeigt dieser Plot einen guten Durchschnitt der Läufe mit den besten Einstellungen für Tesla.

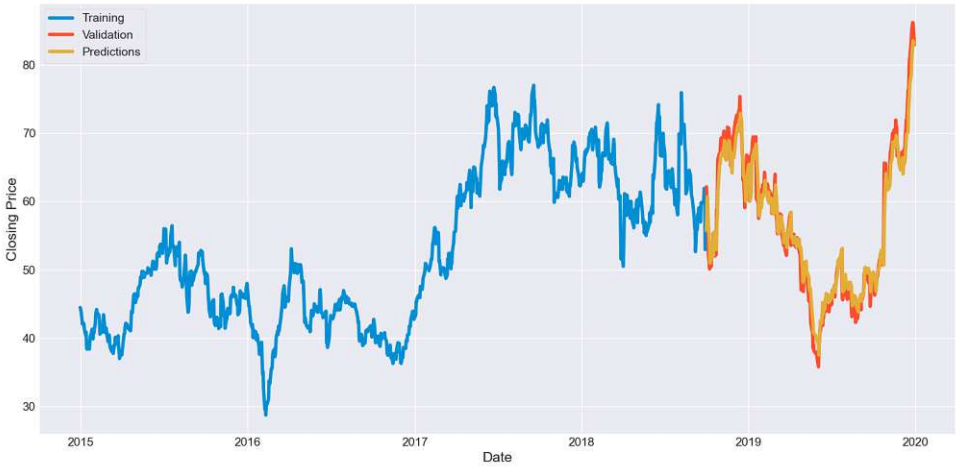


Abbildung 5.3: Vorhersage des Tesla Aktienkurses mithilfe eines feedforward Netzwerks

Für Apple werden die durchschnittlichen besten Ergebnisse auch mit einem feedforward neuronalen Netz mit linearen Aktivierungsfunktionen erzielt. Allerdings wird das Modell mit 10 Epochen gefittet und die letzten 7 Aktienpreise werden in das System eingespielt. Der mittlere quadratische Fehler des Ergebnisses im Plot, der in Abbildung 5.4 ersichtlich ist, ist 0.17476969919386942 und etwas geringer als der durchschnittliche mittlere quadratische Fehler für diese Einstellungen.

Abbildung 5.4: Vorhersage des Apple Aktienkurses mithilfe eines feedforward Netzwerks

6 Zusammenfassung

Das Ziel dieser Arbeit war, die Aktienkurse für die Unternehmen Tesla Inc., Apple Inc. und Biontech SE mittels neuronaler Netze und Sentiment Analyse vorherzusagen. Wie wir gesehen haben, hängen die Ergebnisse sehr stark von der Wahl des Modells und dessen Einstellungen ab. Will man die Aktienpreise für ein anderes Unternehmen vorhersagen, sollte man auch hier wieder durch verschiedene Durchläufe die besten Einstellungen herausfinden.

Ein weiterer wichtiger Faktor für die Güte der Ergebnisse ist die ausreichende Menge und Qualität an historischen Daten. Wie sich zeigte, sind die Ergebnisse für Biontech, bei denen weitaus weniger Daten zur Verfügung stehen, deutlich schlechter.

Bei der Anwendung der in dieser Arbeit vorgestellten Methoden sollte man sich bewusst sein, dass sich die Prognose jeweils nur auf den nächsten Tag bezieht. Eine vorstellbare praktische Anwendung könnte die Implementierung in einem Echtzeitsystem sein. Somit könnte man mittels High Frequency Trading sofort auf Kurs beeinflussende Meldungen reagieren. Das Modell lässt sich durch Hinzufügen von mehrerer Zeitungs- beziehungsweise Social Media Quellen beliebig erweitern.

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

[1] März 2020. URL: https://textblob.readthedocs.io/en/dev/quickstart.html#sentiment-analysis.

[2] Mai 2021. URL: https://www.kaggle.com/omermetinn/tweets-about-the-top-companies-from-2015-to-2020.

[3] Aug. 2021. URL: https://www.kaggle.com/gpreda/pfizer-vaccine-tweets.

[4] Mai 2021. URL: https://developer.nytimes.com/docs/articlesearch-product/1/overview.

[5] Juli 2021. URL: https://www.kaggle.com/amithasanshuvo/pfizer-vaccine-sentiment-and-time-series-analysis.

[6] Aug. 2021. URL: http://amueller.github.io/word_cloud/index.html.

[7] Nov. 2021. URL: https://www.pharmazeutische-zeitung.de/weltweit-erste-zulassung-fuer-biontech-impfstoff-122257/.

[8] URL: https://www.nytimes.com/2021/04/20/business/tesla-crash-texas.html.

[9] Basant Agarwal u. a. *Deep Learning-Based Approaches for Sentiment Analysis*. eng. Algorithms for Intelligent Systems. Singapore: Springer Singapore Pte. Limited, 2020. ISBN: 9789811512155.

[10] Charu C Aggarwal. *Neural Networks and Deep Learning: A Textbook*. eng. Cham: Springer International Publishing. ISBN: 9783319944623.

[11] Martin Aigner. *Graphentheorie: Eine Einführung aus dem 4-Farben Problem*. ger. 2., überarb. Aufl. 2016. Springer Studium Mathematik - Bachelor. Wiesbaden: Springer Fachmedien Wiesbaden. ISBN: 9783658103224.

[12] Faten Alzazah und Xiaochun Cheng. "Recent Advances in Stock Market Prediction Using Text Mining: A Survey". In: Juni 2020. DOI: 10.5772/intechopen.92253.

[13] Malhar Anjaria und Ram Mohana Reddy Guddeti. "Influence factor based opinion mining of Twitter data using supervised learning". In: *2014 Sixth International Conference on Communication Systems and Networks (COMSNETS)*. 2014, S. 1–8. DOI: 10.1109/COMSNETS.2014.6734907.

[14] Yang Bing, Jian Kun Hao und Si Chang Zhang. "Stock Market Prediction Using Artificial Neural Networks". In: *Information Technology for Manufacturing Systems III*. Bd. 6. Advanced Engineering Forum. Trans Tech Publications Ltd, Dez. 2012, S. 1055–1060. DOI: 10.4028/www.scientific.net/AEF.6-7.1055.

[15] H Buehler u. a. "Deep hedging". eng. In: *Quantitative finance* 19.8 (2019), S. 1271–1291. ISSN: 1469-7688.

[16] O Bustos und A Pomares-Quimbaya. "Stock market movement forecast: A Systematic review". eng. In: *Expert systems with applications* 156 (2020), S. 113464. ISSN: 0957-4174.

[17] Francois Chollet. *Deep Learning mit Python und Keras - Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek*. ger. mitp Verlag, 2018. ISBN: 3958458408.

[18] Peggy Daume. *Finanz- und Wirtschaftsmathematik im Unterricht Band 1: Zinsen, Steuern und Aktien*. ger. 1. Aufl. 2016. Wiesbaden: Springer Fachmedien Wiesbaden. ISBN: 9783658106140.

[19] Jay Dawani. *Hands-On Mathematics for Deep Learning: Build a Solid Mathematical Foundation for Training Efficient Deep Neural Networks*. eng. Birmingham: Packt Publishing, Limited, 2020. ISBN: 1838647295.

[20] Shihab Elbagir und Jing Yang. "Twitter sentiment analysis using natural language toolkit and VADER sentiment". In: *Proceedings of the International MultiConference of Engineers and Computer Scientists*. Bd. 122. 2019, S. 16.

[21] CJ Hutto Eric Gilbert. "Vader: A parsimonious rule-based model for sentiment analysis of social media text". In: *Eighth International Conference on Weblogs and Social Media (ICWSM-14). Available at (20/04/16) http://comp. social. gatech. edu/papers/icwsm14. vader. hutto. pdf*. 2014. URL: http://eegilbert.org/papers/icwsm14.vader.hutto.pdf.

[22] Roger D. Huang und Hans R. Stoll. "Dealer versus auction markets: A paired comparison of execution costs on NASDAQ and the NYSE". In: *Journal of Financial Economics* 41.3 (1996), S. 313–357. ISSN: 0304-405X. DOI: https://doi.org/10.1016/0304-405X(95)00867-E. URL: https://www.sciencedirect.com/science/article/pii/0304405X9500867E.

[23] Diederik Kingma und Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations* (Dez. 2014).

[24] Jay A. Kreibich. *Using SQLite*. eng. 1st ed.. Sebastopol: O'Reilly, 2010. ISBN: 1449399649.

[25] Gourav Kumar, Sanjeev Jain und Uday Pratap Singh. "Stock Market Forecasting Using Computational Intelligence: A Survey". In: *Archives of Computational Methods in Engineering* (2020). DOI: 10.1007/s11831-020-09413-5. URL: https://doi.org/10.1007/s11831-020-09413-5.

[26] Eryk Lewinson. *Python for finance cookbook : over 50 recipes for applying modern Python libraries to financial data analysis*. eng. 1st edition. Birmingham ; Mumbai: Packt Publishing, 2020. ISBN: 1789617324.

[27] John Paul Mueller und Luca Massaron. *Machine learning for dummies*. eng. 1. Aufl. For dummies. Hoboken: Wiley, 2016. ISBN: 9781119245513.

[28] Thien Hai Nguyen und Kiyoaki Shirai. "Topic modeling based sentiment analysis on social media for stock market prediction". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2015, S. 1354–1364.

[29] Thien Hai Nguyen, Kiyoaki Shirai und Julien Velcin. "Sentiment analysis on social media for stock movement prediction". In: *Expert Systems with Applications* 42.24 (2015), S. 9603–9611. ISSN: 0957-4174.

[30] Venkata Sasank Pagolu u. a. "Sentiment analysis of Twitter data for predicting stock market movements". In: *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*. 2016, S. 1345–1350. DOI: 10.1109/SCOPES.2016.7955659.

[31] Josh Patterson und Adam Gibson. *Deep Learning*. eng. 1. Aufl. O'Reilly Media, Inc, 2017. ISBN: 9781491914250.

[32] Federico Alberto Pozzi u. a. *Sentiment Analysis in Social Networks*. eng. 1. Aufl. San Francisco: Elsevier Science und Technology, 2016. ISBN: 0128044128.

[33] Sebastian Raschka und Vahid Mirajalili. *Python machine learning : machine learning and deep learning with Python, scikit-learn, and TensorFlow*. eng. Second edition, fully revised and updated.. Expert Insight. Birmingham, England ; Mumbai, [India]: Packt, 2017.

[34] Stefan Richter. *Statistisches und maschinelles Lernen : Gängige Verfahren im Überblick*. ger. 1st ed. 2019. Berlin Heidelberg: Springer Berlin Heidelberg Imprint: Springer Spektrum, 2019. ISBN: 3662593548. URL: 10.1007/978-3-662-59354-7.

[35] Soonh Taj, Baby Bakhtawer Shaikh und Areej Fatemah Meghji. "Sentiment Analysis of News Articles: A Lexicon based Approach". In: *2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*. 2019, S. 1–5. DOI: 10.1109/ICOMET.2019.8673428.

[36] Xian-Da Zhang. *A Matrix Algebra Approach to Artificial Intelligence*. eng. 1st ed. 2020. Singapore: Springer Singapore Imprint: Springer, 2020. ISBN: 9811527709. URL: 10.1007/978-981-15-2770-8.