Freie Universität Berlin

Institut für Informatik

Arnimallee 14, 14195 Berlin

Masterarbeit

Bereitstellung von Kuratierungsvorschlägen zur Reichweitenerhöhung durch den Einsatz von Maschinellem Lernen

Torsten Hain

Matrikelnummer: 4507440

04.12.2017

Betreut durch

Prof. Dr. Raúl Rojas

FU Berlin

Unterstützende Betreuung

Thomas Pietsch

Stoyan Stoyanov

WeltN24

Selbstständigkeitserklärung

Ich erkläre gegenüber der Freien Universität Berlin, dass ich die vorliegende Masterarbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe.

Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch bei keiner anderen Universität als Prüfungsleistung eingereicht und ist auch noch nicht veröffentlicht.

Berlin, 04.12.2017

_____

(Unterschrift Torsten Hain)

Zusammenfassung

In der vorliegenden Masterarbeit wurde die Problematik der Reichweitenvorhersage von Artikeln der Internetseite „der WELT“ analysiert und eine Lösung entwickelt, die Redakteuren eine reichweitenoptimierte Kuratierung ermöglicht. Hierzu wurde die Hypothese eines linearen Zusammenhangs zwischen gleichzeitigen Nutzern in den ersten 30 Minuten und der Reichweite nach 24 Stunden aufgestellt. Für die Überprüfung und Optimierung wurden vier Algorithmen zur linearen Regressionen ausgewählt, umgesetzt und die Modelle verglichen. Aus Artikeldaten und Webtracking-informationen wurden Merkmale erarbeitet, welche Einflüsse auf die Reichweite abbilden. Dabei konnte durch Webtrekking-Merkmale eine lineare Abhängigkeit hergestellt und als Basis für die Vorhersage verwendet werden. Da hohe Reichweiten besonders relevant sind, wurden die besten 10% getrennt betrachtet. Dabei wurden soziale Trends als größter Verbesserungsfaktor festgestellt. Unter anderem wurde außerdem die Ähnlichkeit von Artikeln mit Doc2Vec einbezogen, wobei nur ein geringer Zusammenhang zwischen der Reichweite eines Einzelnen und der Reichweite ähnlicher Artikel identifiziert werden konnte.

In der Umsetzung ist ein Dienst mit austauschbaren Algorithmen und Modellen entstanden. Für den Redakteur wurde die Vorhersage als Reichweitenpotential in das Kuratierungstool integriert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

API. Application Programming Interface

AWS. Amazon Web Services

CMS. Content-Management-System

CORS. Cross-Origin Ressource Sharing

CSV. Comma-Separated Values

HTTP. Hypertext Transfer Protocol

IDE. integrierte Entwicklungsumgebung

JSON. JavaScript Object Notation

KNN. Künstliches Neurales Netz

NER. Named Entity Recognition

NLP. Natural Language Processing

ML. Maschinelles Lernen

SaaS. Software-as-a-Service

1 Einführung

1.1 Motivation

Während Abonnenten- und Leserzahlen von Tageszeitungen sinken, wächst die Bedeutung von Online-Medien ständig. Der Markt von Newsportalen ist hart umkämpft und Betreiber stehen vor vielen Herausforderungen um ihre Plattform für Kunden so attraktiv wie möglich zu gestallten. Eine der wichtigsten Punkte ist hierbei die Startseite: Hier entscheidet sich, ob Inhalte für den Nutzer Interessant sind und dieser somit auf der Seite bleibt. Dabei spielt die Aktualität sowie Positionierung von Artikeln eine entscheidende Rolle. Um diese Aktualität und Positionierung für das Publikum optimal zu gestalten wird jede große Nachrichtenstartseite immer wieder angepasst. Diese Kuratierung wird in vielen Fällen durch Redakteure rund um die Uhr durchgeführt um eine möglichst hohe Reichweite zu erzielen. Die Entscheidung für die Positionierung von Artikeln sind dabei sehr vielschichtig. Sie hängen von Trends, Aktualität, Erfahrung, aktuellen Lage und weiteren Faktoren ab, um möglichst viele Nutzer anzusprechen. Diese Entscheidungen werden hierbei fortlaufend gefällt. Sie stellen ein auf Erfahrung basierendes Optimum dar, welches sich nicht einfach durch ein Regelwerk abbilden lässt. Ebenso ist ein Platzieren zur Verbesserung der Reichweite für den Redakteur nur schwer einschätzbar. In dieser Masterarbeit soll gezeigt werden, dass es möglich ist die Reichweite von Inhalten durch maschinelles Lernen vorherzusagen. Auch soll Untersucht werden, in wie weit sich hierdurch die Reichweite bei der Kuratierung optimieren lässt. Hierzu soll eine Implementierung produktiv integriert werden. Die Vorhersagen sollen dem Redakteur zunächst als Vorschläge präsentiert werden, um eine kritische Komponente wie die Startseite nicht zu gefährden. Eine Automatisierung kann auch helfen Unter- und Themenseiten zu kuratieren, welche aus Kostengründen meist nicht händisch gepflegt werden können.

1.2 Aufgabenstellung und Zielsetzung

Diese Masterarbeit beschäftigt sich einerseits mit der maschinellen Lernbarkeit der Reichweite von Inhalten und andererseits mit der Erstellung eines Dienstes, welcher die Ergebnisse nutzt um die gesamte Reichweite zu verbessern. Dies soll durch die Anzeige von Kuratierungsvorschlägen erreicht werden. Dabei ist das Ziel die möglichst genaue Vorhersagbarkeit der Reichweite, sowie die Zusammenhänge zwischen dieser und den Inhalten besser zu verstehen. Hierzu müssen die zur

Verfügung stehenden Daten (Artikeldaten, Webtracking, bisherige Kuratierung) geprüft, analysiert und Merkmale extrahiert werden. Um ein geeignetes Modell zu finden, sollen verschiedene Algorithmen auf ihre Eignung geprüft, implementiert und verglichen werden. Ein weiterer Aspekt der Umsetzung ist die einfache Anpassung und produktive Einsetzbarkeit. Die Ergebnisse sollen ausgewertet und der beste Algorithmus im Feldeinsatz getestet werden.

Das Ergebnis der Arbeit soll ein produktiv einsetzbarer Dienst sein, welcher die Reichweitenvorhersage durch ein geeignetes Modell implementiert. Dieser soll in das bestehende Kuratierungstool „der WELT“ eingebunden sein und dem Redakteur eine Hilfe zur Positionierung von Artikeln geben.

1.3 Abgrenzung

Die Arbeit begrenzt sich aus Kosten/Nutzen-Abwägungen für „die WELT“ auf die Analyse und Verwendung bereits bestehender Algorithmen zum maschinellen Lernen. Außerdem existiert bereits ein breites Spektrum an Algorithmen und Implementierungen, dessen Ergebnisse jedoch im Einzelfall entscheidend von den Eingabedaten abhängen. Bei der Reichweitevorhersage ist eines der Hauptprobleme die vielen Einflussfaktoren. Der Schwerpunkt der Arbeit wird somit auf die Auswahl und die Erarbeitung von Merkmalen gelegt. Hierzu werden die Daten aus bereits in „die WELT“ integrierten Lösungen verwendet. Der Aufbau von Webtracking und neuer Integrationen in die Seite „der WELT“ ist hingegen nicht Teil der Arbeit. Ebenso werden die Implementierungen in die von „der WELT“ genutzten Amazon Web Services Umgebung integriert, ohne tiefer auf die Plattform oder verwendeten Dienste einzugehen.

1.4 Aufbau

Abbildung 1-1: Aufbau der Masterarbeit

Die Masterarbeit wurde in 4 verschiedene Bereiche eingeteilt, wie in Abbildung 1-1 zu sehen ist:

1. Die **Einführung** beschreibt die Motivation und Struktur dieser Arbeit. Es werden die Ziele der Arbeit definiert und diese genauer abgesteckt.
2. In den ersten beiden Kapiteln werden der **Ausgangspunkt und die Voraussetzungen** für die Entwicklung eines Modells und des Dienstes aufgezeigt. Hierzu werden die Grundlagen und Stand der Technik zu genutzten Algorithmen, der Reichweite und den Datenquellen erläutert, analysiert und daraus Anforderungen an einen Dienst definiert.
3. **Design und Entwicklung** beinhaltet das Entwerfen, Umsetzen und Auswerten der Merkmale für die Reichweite, des Modells fürs maschinelle Lernen und der Implementierung in einem Dienst. Dabei kann der iterative Prozess leider nur bedingt abgebildet werden.
4. Die **Zusammenfassung** zeigt alle wichtigen Ergebnisse der Arbeit auf. Außerdem wird kurz auf den Einsatz des entstandenen Dienstes bei „der WELT“, die aufgetretenen Probleme und zukünftige Möglichkeiten eingegangen.

In den Kapiteln der Bereiche 2 und 3 wurde jeweils in die folgenden Unterbereiche gegliedert:

- Die Reichweite, Daten und Merkmale
- Das maschinelle Lernen
- Die Diensterstellung

Die Reihenfolge der Unterbereiche bleibt dabei gleich und spiegelt den logischen Ablauf wieder.

Eine genauere Beschreibung der einzelnen Kapitel wird jeweils am Anfang des jeweiligen Kapitels gegeben.

2 Grundlagen und Stand der Technik

In diesem Kapitel werden die Grundlagen und der Stand der Technik zusammengefasst, welche zum Verständnis der behandelten Themengebiete dieser Masterarbeit gebraucht werden. Dabei werden die Gebiete Nachrichtenwebseiten, Webtracking und maschinelles Lernen behandelt.

2.1 Nachrichtenwebseiten

Die heutige Form von Nachrichtenwebseiten ist geprägt durch die Verbreitung, Geschwindigkeit und Nutzung des Internets. Dabei ist durch die mobilen Geräte und die ständige Verfügbarkeit ein riesiger Markt entstanden, welcher durch viele Anbieter umkämpft ist. Die Angebote unterscheiden sich in vielen Punkten wie z.B. Aufmachung, Inhalten oder dem Geschäftsmodell. Jedoch gibt es auch Gemeinsamkeiten. Hierzu zählen der Aufbau, welcher bei allen großen deutschen Nachrichtenseiten dem gleichen Muster entspricht. Digitale Nachrichtenportale verdrängen immer mehr die klassischen Medien und die damit verbundenen Einnahmenquellen, wodurch die digitalen Werbeeinnahmen und Abonnements der Nachrichtenseiten immer mehr an Bedeutung gewinnen. Sowohl die Werbeeinnahmen als auch Abonnements stehen dabei in direktem Zusammenhang mit der Reichweite. [1]

2.1.1 Aufbau von Nachrichtenwebseiten

Durch die Nähe zur Zeitung wurden Nachrichtenseiten in ähnlicher Weise aufgebaut. So ist der erste Abschnitt der Startseite immer eine Art Titelblatt mit großen Aufmachern, welche die Nutzer ansprechen und zum Tiefer-Eintauchen einladen sollen. Wenn der Nutzer auf der Seite scrollt, sieht er dabei erst weitere aktuelle Artikel (im sogenannten „Tower“) und danach nach Themengruppen sortierte Inhalte. Durch die Menge an Themengruppen und Artikeln wird die Startseite sehr lang. Dies führt dazu, dass Nutzer an den oberen Artikeln entscheiden, ob sich darunter Interessante befinden oder nicht. Ein Standard sind auch Übersichtsseiten je Themengruppe, um fokussierte Interessen von Nutzern adressieren zu können. Dies führt zu einer großen Anzahl an Übersichtsseiten und Artikeln, welche kuratiert werden müssen. [1]

Dabei werden Artikel auf den Übersichtsseiten durch sogenannte Teaser dargestellt (Beispiel in Anhang I). Diese beinhalten ein Bild, einen Titel, einen kurzen Text und meist Informationen wie beispielsweise die Zeit oder in welchem Bereich der Artikel

angesiedelt ist. Sie sollen dem Nutzer einen Einblick bieten und diesen zum Weiterlesen bzw. Klicken verleiten.

2.1.2 Reichweite

Die Reichweite oder Medienreichweite bezeichnet die relative oder absolute Anzahl der Nutzer, welche durch einen Inhalt in einem bestimmten Zeitraum erreicht werden. Der Begriff wurde durch die Werbebranche geprägt um ein Maß für die Vergütung von Werbeschaltungen zu schaffen. Dabei wird eine Unterscheidung zwischen Brutto und Netto-Reichweite getroffen.

Brutto-Reichweite:

Dieser Wert beinhaltet die Summe jeglicher Kontakte von Nutzern mit dem Inhalt. Dabei wird die Anzahl an Nutzern nicht berücksichtigt.

Netto-Reichweite:

Die Netto-Reichweite ist die um Mehrfachkontakte bereinigte Brutto-Reichweite. Sie spiegelt die Anzahl der Nutzer wieder.

Die Brutto-Reichweite ist somit die Anzahl der Nutzungen, wogegen der Nettowert die Anzahl der Nutzer bezeichnet. In dieser Arbeit wird, wenn nicht anders geschrieben, immer mit der Brutto-Reichweite gearbeitet.

Zusätzlich wird in dieser Arbeit auch die **interne Reichweite** unterschieden. Dabei bezeichnet die interne Reichweite den Anteil, welcher über die Startseite auf Inhalte gelangt sind. Dieser Wert ist für die Kuratierung von besonderem Interesse.

2.1.3 Kuratierung

Die Kuratierung ist die Anordnung und Platzierung von Inhalten in einer Übersicht. Diese wird heute bei den meisten Webseiten händisch von Redakteuren oder mittels des Veröffentlichungszeitpunkts gehandhabt. Sie ist entscheidend für die interne Reichweite von Artikeln und die Absprungrate auf der Startseite. Eine Kuratierung allein durch die Zeit ist hier deutlich im Nachteil, da z.B. populäre Themen verdrängt werden. Eine manuelle Kuratierung ist jedoch durch die große Menge an Übersichtsseiten und Artikeln aufwändig und kostspielig.

WeltN24 hat 2016 für die Kuratierung der Startseite und einzelner Unterseiten ein eigenes Tool geschaffen, welches die Platzierung innerhalb der Webseitenstruktur erlaubt. Dabei orientiert sich der Redakteur an Live-Analysen z.B. Chartbeat (siehe 2.2.2), der aktuellen Lage, aber auch an seiner Erfahrung.

2.2 Webtracking

Allgemein wird beim Webtracking das Verhalten von Nutzern auf einer Webseite gespeichert. Diese Informationen können dann statistisch ausgewertet werden um die Webseite in vielen Bereichen zu optimieren z. B. Häufigkeit/Dauer von Besuchen, Bestellungen und Newsletter-Abonnements. Ein Hauptbestandteil ist das Wiedererkennen von Nutzern, da das „Hypertext Transfer Protocol“ (HTTP) ein zustandsloses Protokoll ist. Dies ist die Voraussetzung um das Verhalten und die Anzahl unterschiedlicher Nutzer bestimmen zu können.

Technisch gibt es viele Möglichkeiten die Wiedererkennung umzusetzen. Die allgemein übliche Methode sind Cookies, welche im Browser des Nutzers gespeichert werden und diesen eindeutig identifizieren. Sie können bei späteren HTTP-Verbindungen einfach wieder abgefragt werden. Andere Möglichkeiten sind die Verwendung von JavaScript um z.B. eindeutige Browserkennungen, Software oder Hardwaremerkmale zu erkennen oder moderne HTML5 Speichermethoden anstatt Cookies zu verwenden. Ebenso können die Ip-Adresse, Flash (Flash Cookies) oder andere Plugins genutzt werden.

Um die gesammelten Daten auswerten zu können, bieten Webtracking-Werkzeuge meist eine ganze Auswahl an Filtern, Analysen und grafischer Aufbereitung an.

Die in dieser Arbeit genutzten Quellen für Webtracking-Daten sind „Webtrekk“ und „Chartbeat“, welche beide als Software-as-a-Service genutzt werden. Sie werden im Folgenden näher behandelt.

2.2.1 Webtrekk

Webtrekk GmbH ist eine 2004 gegründete Firma, dessen Kerngeschäft die Erstellung und Bereitstellung von Webtracking Tools ist. Das gleichnamige Produkt wird in die Webauftritte per Script eingebunden und „[…] ist eine Customer Intelligence-Plattform, mit der Sie Ihre Web-, App- und Marketingdaten zusammenbringen – für ein personalisiertes Kundenerlebnis“ [3]. Die gesammelten Daten werden von Webtrekk auf Servern in Deutschland gespeichert und ihr Tool Webtrekk Q3 ist TÜV-Zertifiziert („Geprüfter Datenschutz“) [4].

Die genutzten Werkzeuge Q3 und Analytics arbeiten auf den gleichen gesammelten Daten. Diese werden jedoch bei der Verarbeitung um etwa 2 Stunden verzögert. Es ist möglich, rückwirkend über einen Zeitraum von 6 Monaten die Daten zu analysieren.

Aufgrund der anfallenden Datenmengen wird dabei auf Stichproben mit dem Verhältnis 1:5 gearbeitet.

Webtrekk unterscheidet in Reports und Analysen, wobei Analysen einzelne Graphen und Abfragen darstellen und diese in Reports zusammengefasst werden können. Es werden über 200 Metriken erstellt, die zusammen mit den vielen Möglichkeiten zur Erstellung von Analysen Webtrekk sehr flexibel machen. Für Q3 bietet Webtrekk eine JSON/RPC -Schnittstelle an um Reports zu exportieren. [2]

2.2.2 Chartbeat

Chartbeat ist 2009 gegründet und hat sich auf die Live-Analyse von Nutzern spezialisiert. Historische Daten können dafür nur stundengenau und über maximal einen Monat abgerufen werden. Dabei bietet Chartbeat zurzeit keine Möglichkeiten zur eigenen Erstellung von Analysen. Jedoch nimmt dies dem Nutzer auch die Notwendigkeit die Erstellung zu erlernen. Es bietet fertige Übersichten mit dutzenden verschiedenen Graphen, welche jedoch interaktiv und übersichtlich gestaltet sind. Chartbeat bietet mehrere JSON-Schnittstellen zum Abrufen von Live-Daten an. [3]

2.3 Maschinelles Lernen

Maschinelles Lernen ist ein interdisziplinäres Feld, welches sich aus Statistik, Wahrscheinlichkeitstheorie, Komplexitätstheorie, Approximationstheorie und vielen anderen Bereichen zusammensetzt. Viele der Mathematischen Grundlagen des maschinellen Lernens sind bereits im letzten Jahrhundert entwickelt worden. Jedoch werden durch die rasante Entwicklung des technischen Fortschritts nun viele komplexere Berechnungen und Modelle möglich. Dies liegt zum einen an den riesigen Datenmengen und zum anderen an der immer höheren Geschwindigkeit von Rechnersystemen. Die weite Verbreitung von mit dem Menschen interagierenden Maschinen und damit einhergehenden Nutzerzahlen und leistungsfähigen Sensoren sind durch immer besserer Vernetzung in der Lage immer größere Datenmengen zu generieren. Mittels maschinellem Lernen können diese Daten nutzbar gemacht werden um z.B. Vorschläge zu geben, Entscheidungen zu Treffen oder Prozesse zu optimieren. [4]

Dabei steht am Anfang fast immer eine Aufbereitung der gesammelten Daten (Merkmalsextraktion in 2.3.1) um die Daten in eine nutzbare Form zu bringen. Je nach Art des zu lösenden Problems können verschiedene Arten von Lernalgorithmen eingesetzt werden, um Daten zu kategorisieren, vorherzusagen oder in neue

Strukturen zu überführen. Dabei schaffen die Lernalgorithmen ein abstrahiertes Wissen in Form von Funktionen, welche als Modelle bezeichnet werden. Um die Leistung von gelernten Modellen bewerten zu können gibt es unterschiedliche Ansätze, welche in Kapitel 2.3.3 behandelt werden.

2.3.1 Merkmalsextraktion

Um die Voraussetzungen für das Lernen mit einem Algorithmus zu schaffen oder zu verbessern ist es oft nötig die Daten aufzubereiten und Merkmale für eine Vorhersage zu extrahieren. Die Ergebnisse eines Lernprozesses sind hierbei stark abhängig von der Qualität und Quantität der vorhandenen Daten. Dabei können „[u]ngeeignete Kodierungen aus einfachen Lernproblemen schwierige machen“ [5], weshalb die Repräsentation der Daten ebenso ein entscheidender Faktor ist. So kann z.B. das Alter oder die Zeit, als natürliche Zahlen, reelle Zahlen oder auch als binäre Klassen (z.B. „<18“, „18-30“, „>30“) wiederum mit oder ohne Überschneidungen abgebildet werden.

Je nach Lernalgorithmus gibt es verschiedene Anforderungen an die Merkmale. So müssen zum Beispiel die meisten linearen Lerner unkorrelierte unabhängige Variablen aufweisen, was in der Realität zu Problemen führen kann.

Dabei können aus den Daten auch qualitativ hochwertige Merkmale durch Vorwissen über Zusammenhänge gebildet werden, welche aufgrund des Algorithmus oder der geringen Menge an Trainingsdaten sonst nicht gelernt werden könnten.

2.3.1.1 One-Hot-Kodierung

Um Merkmalsvektoren aus nicht numerischen Daten zu generieren, kann bei der Textverarbeitung One-Hot-Kodierung verwendet werden. Bei der Kodierung wird jeweils nur das für die Information stehende Bit auf 1 gesetzt und die restlichen n-1 Bits auf 0. Die Dimension des Merkmalsvektors nimmt linear mit der Wörterbuchgröße zu.

Text/Kategorie	One-Hot-Kodierung	Dezimal-Kodierung	Binär-Kodierung
A	0 0 0 0 0 0 0 1	0	000
B	0 0 0 0 0 0 1 0	1	001
C	0 0 0 0 0 1 0 0	2	010
D	0 0 0 0 1 0 0 0	3	011

Tabelle 2-1: One-Hot-Kodierung Beispiel

2.3.1.2 Resampling

Im Kontext dieser Arbeit wird unter Resampling eine Normalisierung zu einer Bezugsgröße verstanden. Dieser Spezialfall wird als Abtastratenkonvertierung (engl. „Sample-rate conversion“) bezeichnet. Die Datenrate von einem diskreten Signal wird hierbei angepasst, um ein neues diskretes Signal zu erzeugen, welches die darunterliegende kontinuierliche Funktion weiterhin abbildet. Dabei wird zwischen Up- bzw. Downsampling unterschieden. Upsampling erhöht die Frequenz der Bezugsgrößen und es werden somit Zwischenräume geschaffen, welche mit Werten gefüllt werden müssen. Dagegen wird beim Downsampling die Frequenz verringert und Bereiche müssen zusammengefasst werden.

Das Auffüllen kann z.B. mittels linearer oder höhergradiger Interpolation oder Splines erfolgen. Bei der Zusammenfassung kann dagegen Beispielsweise der Durchschnitt, das Minimum oder das Maximum genommen werden.

Dabei ist die Abtastratenkonvertierung wichtig um eine zeitliche Normalisierung zu erreichen, weil andernfalls das Modell die Abhängigkeit von der Bezugsgröße erst lernen müsste. Ebenso ist es möglich, dass eine zu kleine Frequenz große Schwankungen der Werte enthält, welche sich jedoch im größeren Kontext z.B. durch Bildung des Mittelwerts relativieren. Andersherum können längere Signalpausen durch Upsampling überbrückt werden und so fehlende Werte aufgefüllt werden.

2.3.1.3 Pivot-Tabellen

Die Erstellung von Pivot-Tabellen ist eine Umformung von Daten. Dabei können die Ausgangsinformationen entweder gleichbleiben oder verdichtet werden. Eine Verdichtung wäre zum Beispiel durch den Mittelwert möglich.

Das folgende Beispiel zeigt die Erstellung einer verlustfreien Pivot-Tabelle:

	X	**Y**	**Z**
A 1	ax1	ay1	az1
A 2	ax2	ay2	az2
A 3	ax3	ay3	az3

Pivot-Tabelle

	X1	**X2**	**X3**	**Y1**	**Y2**	**Y3**	**Z1**	**Z2**	**Z3**
A	ax1	ax2	ax3	ay1	ay2	ay3	az1	az2	az3

Abbildung 2-1: Beispiel einer verlustfreien Pivot-Tabellen-Umformung

2.3.2 Unterteilung der Lernarten

Je nach Feedback können die meisten Algorithmen beim maschinellen Lernen in die folgenden Kategorien eingeteilt werden:

Überwachtes Lernen (engl. Supervisor learning):
Für das Trainieren und Testen des Modells stehen Daten mit korrekten Labeln bzw. Ergebnissen zur Verfügung. Beim Training werden die Gewichte möglichst so verändert, dass der Fehler zum korrekten Ergebnis minimiert wird.

Unüberwachtes Lernen (engl. unsupervised learning):
Beim unüberwachten Lernen ist dagegen zu den Daten nicht bekannt, welches die Ausgabe für eine Eingabe ist. Hierbei analysiert der Algorithmus ohne vorherige Kenntnis von Ergebnissen die Daten und versucht anhand von Abhängigkeiten zu Segmentieren bzw. eine Ordnung in die Daten zu bringen.
Da jedoch die Ausgaben vorher nicht definiert sind, müssen die Ausgaben meist interpretiert werden um ihre Bedeutung und Sinnhaftigkeit festzustellen.

Bestärkendes Lernen (engl. reinforcement learning):
Bei dieser Methode wird das Model anhand von positiven oder negativen Feedback optimiert. Dabei wird durch den Algorithmus automatisch die optimale Anpassung zur Verbesserung der Vorhersage bestimmt, ohne das die genaue Abweichung vom wahren Wert bekannt ist.

In dieser Arbeit wird hauptsächlich überwachtes Lernen für die Vorhersage der Reichweite genutzt. Bei der inhaltlichen Analyse werden jedoch Word2Vec und Doc2Vec genutzt, welche auf unüberwachtem Lernen basieren.

Weitere Unterscheidungen sind bei der Trainingsanzahl vor der Gewichtsanpassung möglich. Dabei werden dann Mittwerte oder Summen der Gewichtsdeltas genutzt.
Online: Nach jedem Training.
Batch: Nach der kompletten Trainingsmenge.
Micro-Batch: Nach einer bestimmten Anzahl von Trainings.

2.3.3 Modell Bewertung

Sobald das Problem definiert und die Daten in der richtigen Form sind, können durch verschiedene Algorithmen Modelle erstellt werden, um das Problem zu lösen. Dabei kann viel Zeit in die Auswahl, die Ausführung und Parameteroptimierung gesteckt werden. Es ist somit wichtig sicherzustellen, dass bei diesem Prozess das Modell der Problemlösung näherkommt. Um dies zu messen wird eine Fehlerfunktion verwendet. Eine einfache Funktion, welche allgemein genutzt wird, ist die Summe der quadratischen Fehler. Dabei ist durch Überanpassung (engl. overfitting) oder Unteranpassung (engl. underfitting) nicht immer gewährleistet, dass bei Verbesserung der Testdatenfehler auch allgemeines Wissen im Modell gelernt wird. Hierzu gibt es verschiedene Möglichkeiten Modelle zu vergleichen und zu bewerten. Im Folgenden werden hierbei die Fehler R^2 und angepasstes R^2 (engl. adjusted R^2) aufgezeigt. Einzelne Messwerte sind aber meist immer mit Vorsicht zu genießen. Durch eine zusätzliche Analyse von Graphen kann mehr Aufschluss über eine gute Verteilung und die Gründe für schlechte Anpassung gewonnen werden. Dabei sind Graphen wie z.B. Vorhersage gegen richtige Werte oder Residuen gut geeignet. [6]

2.3.3.1 R^2-Fehler

Der R^2-Fehler ist ein statisches Maß für die Güte einer Regression, welches über den quadratischen Fehler vergleichbare Werte liefert. Es ist das Verhältnis zwischen der Variation der abhängigen Variable von der Vorhersage durch die unabhängigen Variablen.

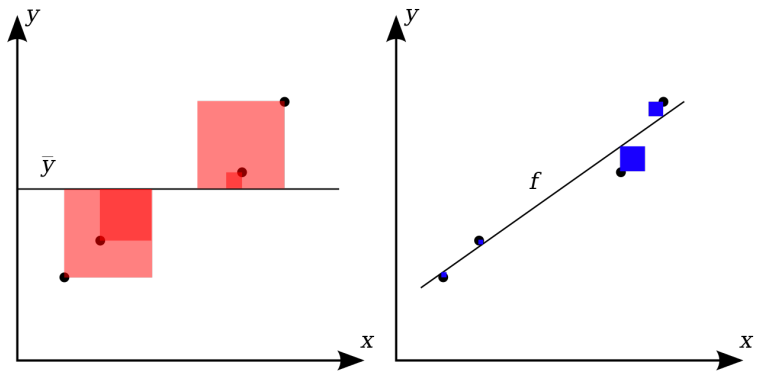

Abbildung 2-2: Veranschaulichung von SS_{Total} (links) und $SS_{Residuen}$ (rechts) (übernommen von Orzetto [7])

Sein y_i die Werte der abhängigen Variablen, f_i die Vorhersagewerte und $\bar{y}$ der Durchschnitt von y, dann ist die generelle Formel:

$$R^2 = 1 - \frac{SS_{Residuen}}{SS_{Gesamt}} = 1 - \frac{\text{Variation der Residuen}}{\text{Variation von y}} = 1 - \frac{\sum_i (y_i - f_i)^2}{\sum_i (y_i - \bar{y})^2}$$

Wobei $SS_{Bezeichner}$ die in der Literatur gängige Bezeichnung für die Summe der Quadrate (engl. „sum of squares“) ist. Anhand der Formel mit Abbildung 2-2 kann leicht abgelesen werden, dass bei einem maximalen Wert von 100% die Fehler 0 sein müssen und damit eine perfekte Anpassung vorliegt. Bei einem Wert von 0% wird dagegen durch die Regression nicht mehr erklärt, als durch den Mittelwert von y.
Im Fall von linearer Regression kann durch die Relation

$$SS_{Residuen} + SS_{Regression} = SS_{Gesamt}$$

die folgende Gleichung geschlossen werden [8]:

$$R^2 = \frac{SS_{Regression}}{SS_{Gesamt}} = \frac{\text{Variation der Regression}}{\text{Variation von y}} = \frac{\sum_i (f_i - \bar{y})^2}{\sum_i (y_i - \bar{y})^2}$$

Hierdurch kann direkt geschlossen werden, welcher Anteil der Variation von der Regression schon erklärt werden kann. Bei 0% wird somit der Mittelwert abgebildet und keinerlei Variation um diesen wird durch die Regression erklärt. Dagegen werden bei 100% durch die Regression die vollständige Variation um den Mittelwert in den Daten erklärt.
Bei genauerer Betrachtung ergibt sich jedoch ein Problem:
Der Messwert für R^2 kann beim Hinzufügen von neuen unabhängigen Variablen zum Modell nur größer und somit besser werden. Hierdurch kann bei mehr Variablen ein besseres Ergebnis entstehen, obwohl nicht mehr als Rauschen gelernt wurde.

2.3.3.2 Angepasster R^2-Fehler

Um das zuvor aufgezeigte Problem anzugehen, wurde ein angepasster R^2-Fehler (engl. adjusted R^2) entwickelt. Dieser bezieht die Anzahl an genutzten Daten und unabhängigen Variablen mit ein. So verringert sich der Wert, bei gleicher Anzahl Trainingsdaten und größerer Anzahl unabhängiger Variablen, wenn sich nicht mehr Varianz durch diese erklären lässt. Wie in Abbildung 2-3 zu sehen ist, werden dabei niedrige R^2-Fehler mehr bestraft.
Die Formel ist wie folgt:

$$R^2_{adj} = 1 - \left[\frac{(1 - R^2)(n - 1)}{n - k - 1}\right]$$

Wobei n die Anzahl an genutzten Daten ist und k die Anzahl an unabhängigen Variablen ist.

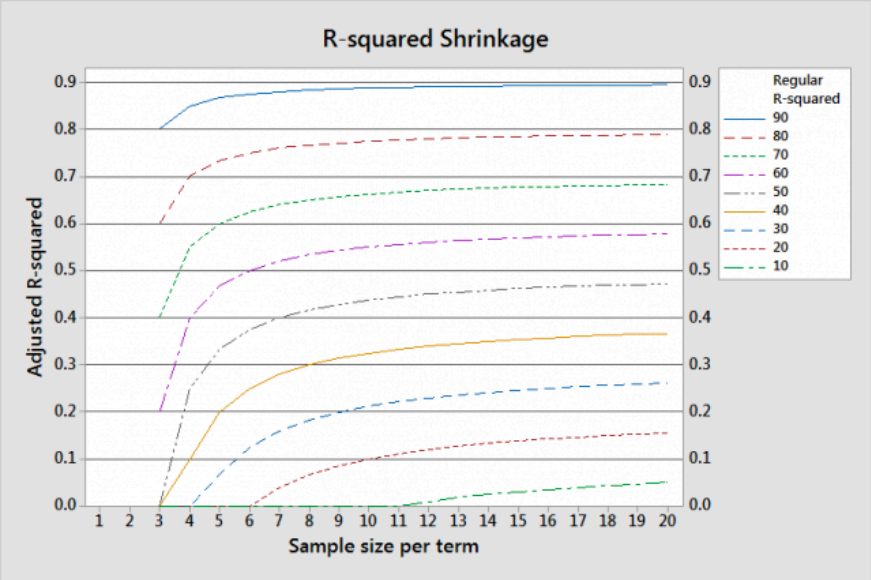

Abbildung 2-3: Angepasstes R^2 in Abhängigkeit von Daten per Variable (übernommen von [9])

2.3.3.3 Kreuzvalidierung

Die Kreuzvalidierung (engl. k-fold cross-validation) ist eine Technik um einer eventuell schlechten Wahl bei der Aufteilung von Trainings- und Testdaten entgegen zu wirken. Durch eine einfache Aufteilung und Durchführung ist die Verlässlichkeit von Maßen wie R^2 nicht gegeben. Hier setzt Kreuzvalidierung ein und teilt die zur Verfügung stehenden n Daten in k gleichgroße disjunkte Mengen. Mit diesen k Mengen wird k mal trainiert, wobei jeweils alle k Mengen einmal Testmenge sind und der Rest jeweils Trainingsmenge ist (siehe Abbildung 2-4). [4]

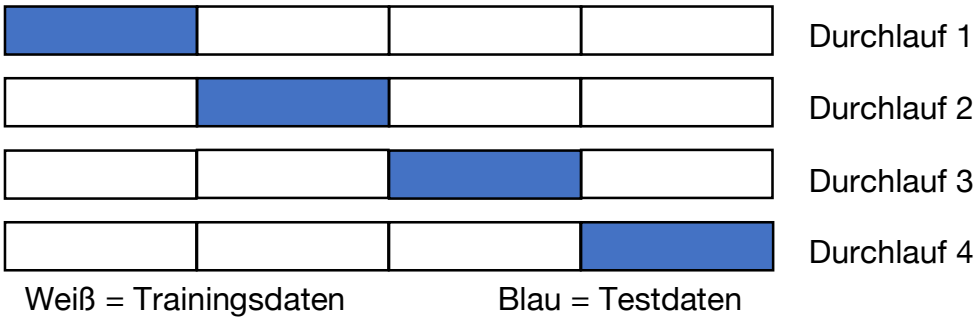

Abbildung 2-4: Kreuzvalidierung mit k=4

2.3.4 Regression

Die Regression bezeichnet in der Statistik die Abhängigkeit zwischen Variablen. Dabei wird bei der Regressionsanalyse der Zusammenhang einer abhängige Variable von einer oder mehreren unabhängigen Variablen bestimmt. Als Regressionsfunktion wird hierbei die zu findende Funktion bezeichnet, welche als Parameter die unabhängigen Variablen bekommt und die abhängige Variable möglichst gut annähert.

Diese Regressionsfunktionen können in lineare und nicht lineare Funktionen unterschieden werden, wonach sich auch daraus resultierenden Modelle einteilen lassen. In dieser Arbeit wurden zur Regression aufgrund der Analyse nur die folgenden linearen Modelle betrachtet. Diese haben die generelle Form

$$y = XB + U$$

wobei y den Vektor mit den Ergebnissen bezeichnet, X die Matrix mit den um eins erweiterten unabhängigen Variablen für jedes Ergebnis, B die Gewichte für jedes Merkmal und U die Fehler für jedes Ergebnis.

Um ein gewünschtes Verhalten der Regression zu gewährleisten, gibt es die Möglichkeit, den Regressionsterm mit Regulatoren zu versehen, die als Nebenbedingungen auftreten. Hierdurch kann zum Beispiel eine Überanpassung an den Trainingsdatensatz vermieden werden. Die Robustheit gegenüber Ausreißern, welche in der Grundform kaum gegeben ist, nimmt somit zu. [4]

2.3.4.1 Multiple Lineare Regression

Bei der Multiplen Linearen Regression wird die abhängige Variable durch eine Linearkombination der Regressionskoeffizienten dargestellt. Dabei wird angenommen, dass der Fehlervektor und damit auch die unabhängigen Variablen unkorreliert sind. Die Formel ist durch die generelle Form schon gegeben:

$$y = X\Phi + e$$

Die zu minimierende Fehlerfunktion ist dabei die Summe aller quadratischen Fehler:

$$E = e^T e = (y - X\Phi)^T (y - X\Phi)$$

$$= \sum^{N} (y_1 - \phi_0 - \phi_1 x_1 - \cdots - \phi_p x_{ip})^2$$

Dabei gibt es viele mögliche Lösungsansätze für das Problem der kleinsten Quadrate wie z.B. partiale Ableitungen, Vektor Ableitungen, die Pseudoinverse, statistische Ansätze, normalen Projektion, vervollständigen der Quadrate und die Fisher Diskriminante. [10]

Als Beispiel wird kurz die statistische Lösung aus [10] erläutert. Dabei wird angenommen, dass alle Daten zentriert sind, also $E(y) = 0$ und $E(x_j) = 0$ für alle Variablen $x_1, x_2, \ldots, x_p$ gilt.

Durch die Grundform mit X^T multipliziert ergibt sich

$$X^T y = X^T X\Phi + X^T e$$

$X^T e$ bildet nun die Summe aus E(y) und E(X). $X^T e$ muss ein Nullvektor sein, weil ansonsten mindestens eine Komponente k nicht Null sein müsste. Aber die k-te Komponente $\sum e_i x_{ik}$ ist die Covarianz von x_{ik} und e_i. Wenn diese Kovarianz nicht Null ist, wäre ein Teil der Fehler von p_{ik} abhängig und somit ist der Vektor e im linearen Modell proportional abhängig zu k-ten Reihe von X. Womit die Gewichte nicht optimal wären. Daraus folgt, dass $X^T e = 0$ sein muss und somit

$$X^T y = (X^T X)\Phi \Rightarrow \Phi = (X^T X)^{-1} X^T y$$

Multiple Lineare Regression ist für mehrere Ergebnisvektoren y anwendbar, sodass für jeden Vektor eigene Gewichtsvektoren und Fehler entstehen.

2.3.4.2 Ridge

Ridge Regression ist eine Erweiterung der linearen Regression um eine Regulation (L_2). Diese verbessert den Fehler der Vorhersage indem große Koeffizienten bestraft werden um Überanpassung zu reduzieren. Hierfür wird vor der Bildung der Inversen noch ein konstantes kleines λ auf die Diagonale addiert. Daraus folgt:

$$\Phi_{Ridge} = (X^T X + \lambda I_P)^{-1} X^T y$$

Durch den Regulationsterm wird das folgende Minimum gebildet:

$$\min_{\beta}\{\|y - X\beta\|_2^2 + \lambda\|\beta\|_2^2\}$$

Wobei λ eine Konstante ist und $\|\beta\|_2^2$ das Quadrat der ℓ_2-norm vom Gewichtsvektor ist.

2.3.4.3 Lasso

Lasso (least absolute shrinkage and selection operator) ist ebenfalls eine Erweiterung um Regulation (L_1). Jedoch wird im Vergleich zu Ridge eine Untermenge der Variablen ausgewählt. Dabei ist ein Ziel die Interpretierbarkeit von Regressionsmodellen zu verbessern. Hierzu wird ein Limit t eingeführt, welches die Gewichte nicht überschreiten dürfen. Im Vergleich zu Ridge können hier ganze Koeffizienten Null werden.

Dabei wird bei Lasso das folgende Minimum gelöst [11]:

$$\min_{\beta}\{\|y - X\beta\|_2^2 + \lambda\|\beta\|_1\}$$

Wobei λ eine Konstante ist und $\|\beta\|_1$ die ℓ_1-norm vom Gewichtsvektor ist.

2.3.4.4 Elastisches Netz

Beim elastischen Netz werden die Regulatoren von Ridge und Lasso kombiniert. Es können so ganze Koeffizienten Null und gleichzeitig große Koeffizienten bestraft werden. In der Formel werden sowohl L_1 als auch L_2 vereint:

$$\min_{\beta}\{\|y - X\beta\|_2^2 + \lambda_2\|\beta\|_2^2 + \lambda_1\|\beta\|_1\}$$

2.3.5 Neurale Netze

Neurale Netze oder Künstliche Neurale Netze (KNN) sind Modelle welche durch den Biologischen Aufbau des Menschlichen Gehirns inspiriert wurden. [12] Dabei werden Neuronen verknüpft und geben anhand ihres Inputs und einer Aktivierungsfunktion das Ergebnis an folgende Neuronen weiter. Dies geschieht bis zur Ausgabe. Dabei hat ein einzelnes Neuron immer eine Aggregationsfunktion g für alle eingehenden Kanten und eine Aktivierungsfunktion f für die Ausgabe. Ein Spezialfall eines Neurons bildet das Perzeptron, welches die Summe der gewichteten Eingaben bildet und durch eine Schwellenwertentscheidung bei 0, 1 oder 0 ausgibt (siehe Abbildung 2-5).

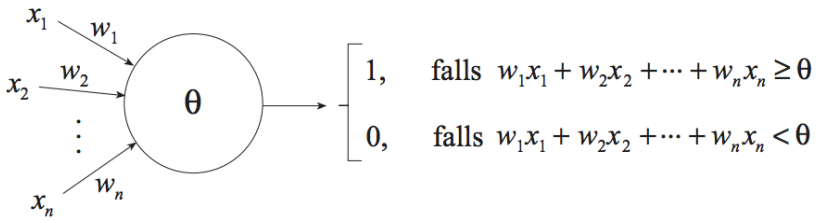

Abbildung 2-5: Darstellung eines Perzeptrons (übernommen von R. Rojas [13])

In KNNs wird das Netz in verbundene Schichten unterteilt, welche jeweils eine bestimmte Anzahl Neuronen besitzen und zwischen angrenzenden Schichten einen vollständigen Graphen bilden. Die erste Schicht wird die Eingangs- und die letzte Ausgangsschicht genannt, da sie die Ein- bzw. Ausgabe übernehmen und der Anzahl der Merkmale bzw. Ausgaben (z.B. Klassen) entsprechen müssen (siehe Abbildung 2-6). Die Schichten dazwischen sind die verborgenen Schichten (engl. hidden layers). Im Folgenden wird beispielhaft das Lernen in vorwärtsgerichteten Netzen behandelt. Dabei wird Backpropagation zur Gewichtsanpassung verwendet. Wie in der Abbildung 2-6 zu sehen, kann bei Perzeptron-Lernen bzw. dem Schwellenwert Null jede außer der Ausgabeschicht um 1 erweitert werden, um eine Verschiebung durch das hinzugewonnene Gewicht zu ermöglichen. Als Aktivierungsfunktion wird die Sigmoidfunktion gewählt, da sie stetig und überall differenzierbar ist. Außerdem bildet die Funktion auf das Intervall (0,1) ab.

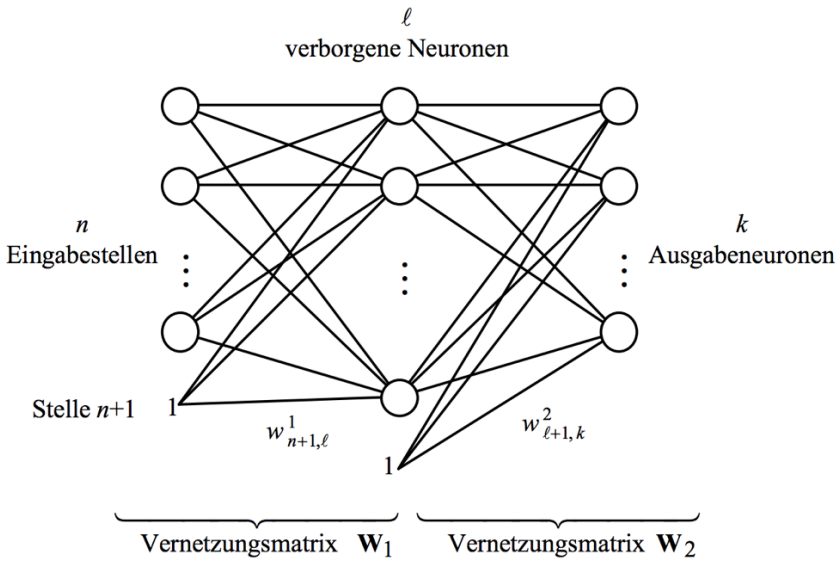

Abbildung 2-6: Struktur eines neuronalen Netzes mit einer verborgenen Schicht (übernommen von R. Rojas [13])

Beim Durchlaufen des Netzwerkes ergibt sich für jedes Ausgabeneuron so ein Wert. Um den Fehler zu bestimmen kann das Netz erweitert werden mit jeweils einem Neuron für den quadratischen Fehler und am Ende eines, welche die Summe dieser bildet.

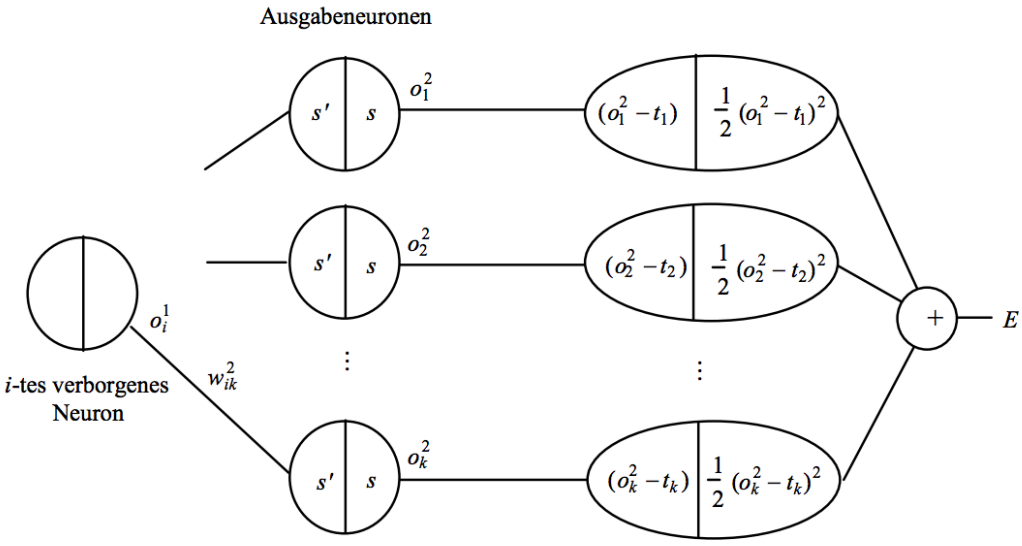

Abbildung 2-7: Erweitertes Netz zur Berechnung der Fehlerfunktion (übernommen von R. Rojas [13])

Durch Backpropagation kann nun über die Ableitungen der rückwärtsverteilte Fehler für die einzelnen Neuronen bzw. Gewichte bestimmt werden.

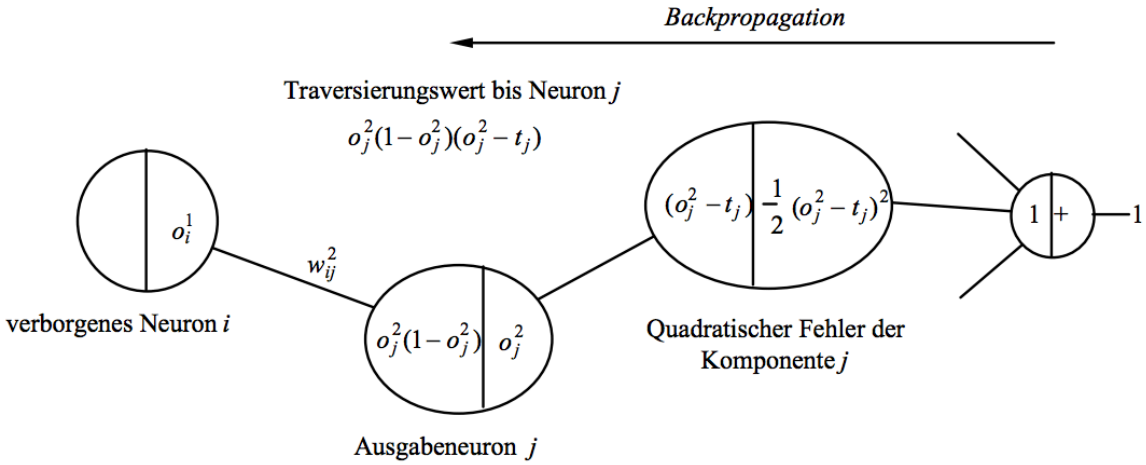

Abbildung 2-8: Backpropergation Schritt (übernommen von R. Rojas [13])

Im letzten Schritt können die Gewichtsanpassungen in negativer Gradientenrichtung erfolgen. Dabei wird durch eine Lernkonstante γ die Konvergenzgeschwindigkeit geregelt.

2.3.5.1 Word2Vec

Neurale Einbettungen (engl. neural embeddings) wurden erstmals von Bengio und anderen in 2003, in Form eines vorwärtsgerichteten-Netzwerkes vorgeschlagen. [14] Word2Vec [15] verwendet einfachere und effizientere Neurale Architekturen um Wort-Vektoren zu lernen, jedoch ist das Prinzip des Einbettungsproblems das Gleiche.

Hierbei wird der Kontext des zu repräsentierenden Wortes als Grundlage für das Modell genutzt. Für jedes Wort in einem Text wird ein Rahmen gebildet, der je nach Algorithmus als Eingabe bzw. Ausgabe dient und in der Größe anpassbar ist.

Dabei werden die Kontextwörter bei Skip-grams (SG) unabhängig von der genauen Position vorhergesagt, wohingegen bei „Cotonous Bag of Words“ (CBOW) die Kontextwörter genutzt werden um das eingebettete Wort vorherzusagen.

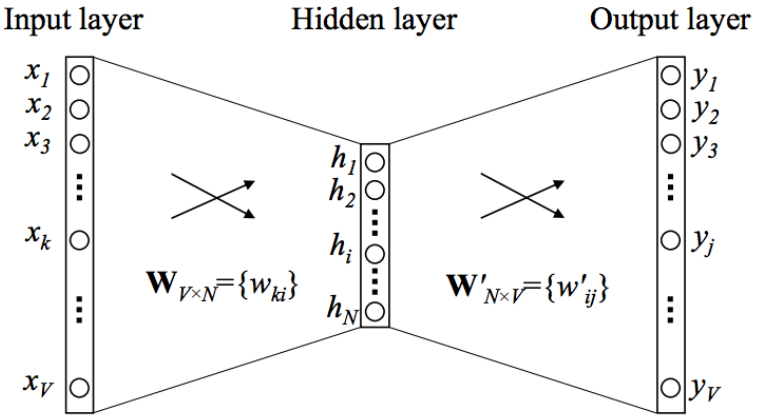

Abbildung 2-9: Ein einfaches CBOW Model mit nur einem Wort als Kontext (Erstellt von X. Rong [16])

In der Abbildung 2-9 ist der Aufbau des vollständig verbundenen Neuralen Netzes zu sehen, welches beim einfachsten Fall von einem ein Wort als Kontext entsteht. Dabei werden Wörter in x und y One-Hot-Kodiert und durch eine einzelne verdeckte Schicht (engl. hidden layer) verbunden. Durch das Kodieren wird die k-te Reihe der Matrix W beim Vorwärtsschritt in h kopiert und ist gleichzeitig eine Vektorrepräsentation des Wortes x_k. Formal unter der Annahme $x_k = 1$ und $x_{k'} = 0$ für $k' \neq k$:

$$h = W^T x = W^T_{(k,\cdot)} \coloneqq v^T_{w_I}$$

Dieser Vektor h wird beim Vorwärtsschritt weiter mit den Gewichten W' multipliziert um einen Ergebnisvektor u_j zu erhalten.

$$u_j = {v'_{w_j}}^T h$$

wobei v'_{w_j} die j-te Spalte von W' ist und die zweite Repräsentation darstellt. Nun wird Softmax als Verteilungsfunktion verwendet, um eine Polynominalverteilung zu erreichen.

$$p(w_j|w_I) = y_j = \frac{\exp(u_j)}{\sum_{j'=1}^{V} \exp(u_{j'})}$$

Durch Einsetzten ergibt sich

$$p(w_j|w_I) = y_j = \frac{\exp\left({v'_{w_j}}^T v_{w_I}\right)}{\sum_{j'=1}^{V} \exp\left({v'_{w_{j'}}}^T v_{w_I}\right)}$$

Beim Rückwärtsschritt wird die folgende Verlustfunktion minimiert:

$$E = -\log p(w_O|w_I)$$

Dies geschieht durch die Ableitung, sodass für die Gewichte W' bzw. v'_w folgender Updateschritt entsteht [16]:

$${v'_{w_j}}^{(new)} = {v'_{w_j}}^{(old)} - \eta \cdot e_j \cdot h \qquad \text{for } j = 1,2,\dots,V$$

Für W bzw. v_w ergibt sich:

$$v^{(new)}_{w_I} = v^{(old)}_{w_I} - \eta E H^T$$

Mit EH, als N-Dimensionaler Vektor, welcher die Summe aller Ausgabevektoren aller Wörter in V darstellt, gewichtet nach ihrem Fehler $e_j = y_j - t_j$ (t_j ist die Wahrheit: 0 oder 1).

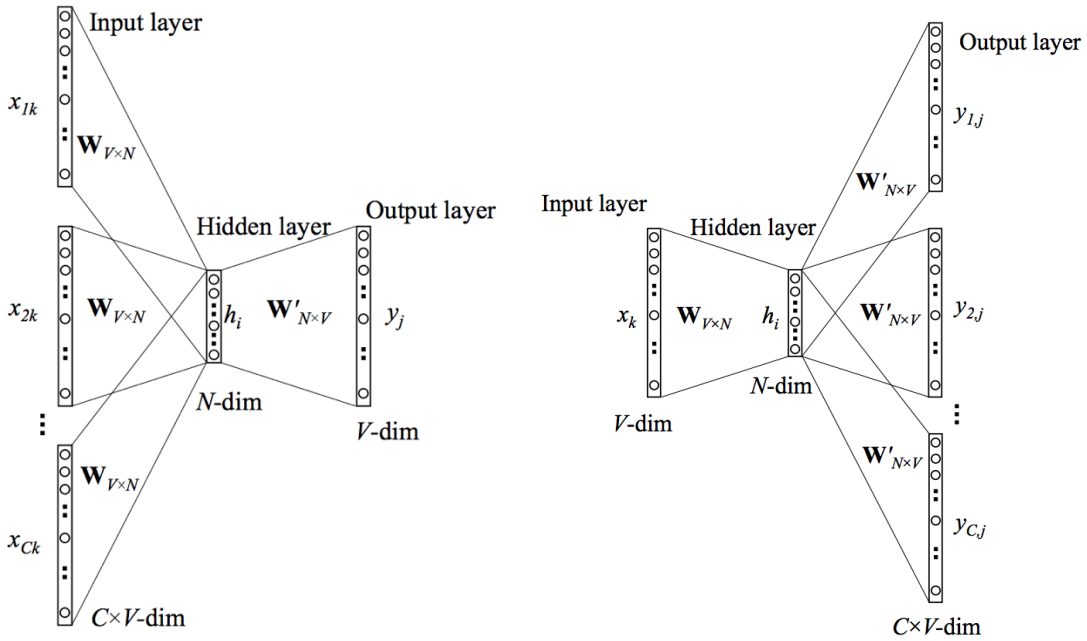

Abbildung 2-10: Das CBOW Modell (links) und Skip-gram Modell (rechts) (Erstellt von X. Rong [16])

Durch mehrere Kontextwörter wird das Modell entsprechend der Abbildung 2-10 erweitert. Dabei werden anstatt die Gewichte für w direkt in h zu kopieren, die Vektoren für alle Kontextwörter gemittelt.

$$h = \frac{1}{C}\left(v_{w_1} + v_{w_2} + \cdots + v_{w_C}\right)^T$$

Die Verlustfunktion ändert sich damit zu:

$$E = -\log p\left(w_O | w_{I,1}, \ldots, w_{I,C}\right)$$

Die Updates bleiben unverändert, außer dass die Eingangsgewichte für alle Wörter im Kontext angepasst werden müssen.

Wie in Abbildung 2-10 zu sehen, ist das Skip-gram-Modell das genaue Gegenteil zum CBOW-Modell. Auf die Formeln wird an dieser Stelle verzichtet.

Die Implementierung wird durch „Hierarchical Softmax“ und „Negativ Sampling“ optimiert, wobei „Hierachical Softmax“ durch binäre Suchbäume die Komplexität von O(V) auf O(log(V)) senkt und damit große Geschwindigkeitsvorteile bringt. „Negative Sampling“ geht hingegen auf die Anzahl der anzupassenden Gewichte ein. Da die Gewichtsanzahl multiplikativ von der Anzahl verschiedener Wörter und der Anzahl Knoten in der verdeckten Schicht abhängt, müssen bei jedem Training sehr viele Gewichte angepasst werden. Beim „Negative Sampling“ werden nach dem Vorwärtsschritt eine bestimmte Anzahl von falschen Beispielen gewählt und nur diese

zusätzlich angepasst. Die Auswahl findet in Abhängigkeit zur Wordhäufigkeit statt. [16]

2.3.5.2 Doc2Vec

Doc2Vec ist 2014 von Quoc und Mikolov als Erweiterung von word2vec für Paragraphen entwickelt worden. [17] Hierbei bekommen Paragraphen eine eindeutige Kennung zugewiesen. Jeder word2vec Rahmen erhält zusätzlich diese eindeutige Kennung. Die Kennungen werden dann der Eingangsschicht hinzugefügt, wie dies bei mehreren Wörtern im Kontext bei word2vec geschehen ist (siehe Abbildung 2-11). Beim Lernen wird ebenso die Kennung vorhergesagt und angepasst wie zuvor. Dadurch wird beim Lernen des Modells ebenso für jeden Paragraph ein Vektor erstellt.

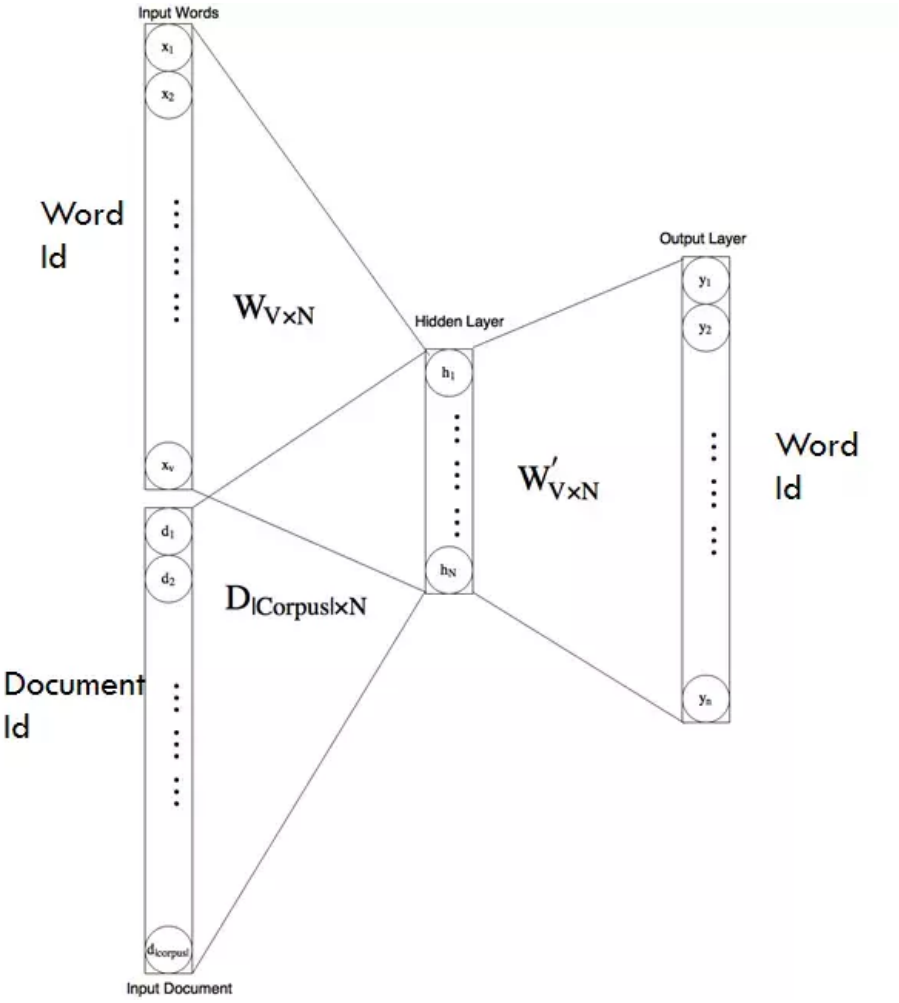

Abbildung 2-11: Schematische Darstellung von Doc2Vec (übernommen von Piyush B. [16])

2.4 Zusammenfassung

In diesem Kapitel wurden die Grundlagen und der Stand der Technik für diese Arbeit aufgezeigt. Es wurde erläutert, dass die Kuratierung als ein wichtiger Faktor für die Nachrichtenseiten ist, um Reichweite und damit Werbeeinnahmen und Abonnementverkäufe zu erzielen. Die Reichweite ist dabei ein definiertes Maß für die Anzahl von Nutzern. Beim Webtracking wurden Webtrekk als Reichweitenlieferant und Chartbeat für Live-Daten aufgrund ihrer Möglichkeiten festgestellt und kurz betrachtet. Zum Schluss wurde das maschinelle Lernen erarbeitet. Dabei wurde ein Überblick über Merkmalsextraktion und Regression gegeben und insbesondere auf die genutzten Methoden und linearen Modelle eingegangen. Dabei ist jedoch, aufgrund der Menge, der Fokus auf die in der Arbeit genutzten Methoden gelegt worden. So wurde auch kurz auf das Grundprinzip von Neuralen Netzen am Beispiel von Vorwärtsgerichteten Netzen eingegangen um dann genauer auf word2vec und doc2vec einzugehen.

3 Analyse und Anforderungen

Um die Reichweite durch Vorschläge bei der Kuratierung erhöhen zu können, wird als erstes die Reichweite als solche analysiert. In dieser Arbeit wird die Hypothese aufgestellt, dass sich die Reichweite von Inhalten durch gut gewählte Merkmale vorhersagen lässt. Hierdurch können Inhalte mit vielen potentiellen Nutzern besser platziert werden und dadurch insgesamt sogar noch mehr Nutzer erreichen. Um dies Umzusetzen und zu überprüfen, wird im Anschluss dessen Vorhersagbarkeit durch maschinelles Lernen erörtert. Im Anschluss werden die Anforderungen an den umsetzenden Dienst definiert, dies beinhaltet sowohl den Aufbau, die Auswahl von Frameworks für die Algorithmen, als auch das Anforderungen für das User Interface.

3.1 Reichweite

Die Reichweite ist eine Kennzahl für die Popularität und den Erfolg des veröffentlichten Inhalts. Durch sie lassen sich Artikel, ebenso wie die Kuratierung seit Beginn des Webtrackings vergleichen und optimieren. Dabei lassen sich jedoch die im nachherein gemessenen Werte nur noch zur Auswertung nutzen und nur teilweise auf neue Inhalte transferieren. So kann die Reichweite bei regelmäßigen Artikeln, wie z.B. Sportergebnissen recht gut verglichen werden. Jedoch sind viele Artikel in ihrem Inhalt und ihrer Situation so verschieden, dass es nicht klar ist, welche Reichweite wird ein Artikel bekommen, bzw. hätte ein geänderter oder anders positionierter Artikel erreichen können.

Dabei wird in dieser Arbeit vorausgesetzt, dass die Nutzergruppen „der Welt“ in der Regel auch ein bestimmtes Muster haben, nachdem sie Artikel anklicken und lesen. Durch diese Eigenschaft steht die Reichweite von neuen Artikeln im Zusammenhang mit den historischen Werten anderer Artikel. Daher ist es möglich, dass Redakteure anhand ihrer Erfahrung ein Gefühl für ihre Nutzer bekommen und so grob die Reichweite einschätzen können. Allerdings ist dieses Gefühl sehr subjektiv und kaum an konkreten Merkmalen oder Regeln festzumachen.

Dabei ist die Reichweite für Inhalte abhängig von vielen Faktoren. Um diese so genau wie möglich prognostizieren zu können, müssen diese aufgeschlüsselt und mögliche Merkmale identifiziert werden. Die Tabelle auf der nächsten Seite gibt einen Überblick über die Einflüsse unterteilt in Kategorien.

Weltgeschehen	- Aktuelle Themen - Trends
Zeitlich	- Erscheinungszeitpunkt - Aktualitätsdauer
Inhalt	- Verfasser - Qualität - Zielgruppe - Neuheit - Quantität - Medien - (Aussehen)
Sichtbarkeit	- Platzierung (Seite) - Aufmachung - Vermarktung - Soziale Verbreitung

Tabelle 3-1: Übersicht der Reichweiteneinflussfaktoren

Während sich der Wert für die Reichweite fest definieren und messen lässt, sind die meisten in der Übersicht aufgezeigten Kategorien nicht trivial durch Messwerte abbildbar.

Bei der Vorhersage von Nutzerverhalten (z.B. Anzahl von Retweets, Youtube-Likes, etc.) gibt es bereits Forschung, welche sich in vor und nach der Veröffentlichung des Inhalts einteilen lässt. So wird in [18] die Anzahl an Tweets zu einem unveröffentlichten Artikel vorhergesagt, je nach Anzahl an Posts auf Twitter und inhaltlicher Merkmale. Es ist anzunehmen, dass eine Prognose zusätzlich basierend auf Daten kurz nach der Veröffentlichung wesentlich genauer wird, da Inhaltliche ausnahmen besser eingefangen werden. Dies ist auch stark von der Qualität der Inhaltlichen Merkmale und der Größe der Datenbasis abhängig. Durch die zusätzlichen ersten Eindrücke, wird das benötigte Wissen über das Weltgeschehen stark reduziert. Da dieses für Nachrichten komplex, schnell verändernd und somit nur schwer in ein Modell zu fassen ist, wurden in dieser Arbeit auch Daten „kurz“ nach der Veröffentlichung einbezogen.

Dabei wurden in den genutzten Quellen Zeiten von 10 Minuten bis 1 Stunde nach der Veröffentlichung verwendet. Um eine geeignete Zeit zu finden, wurden die Anzahl der

Inhalte mit Messwerte der ersten 10 Tage herangezogen (siehe Abbildung 3-1). In dieser Zeit wurden nach 30 Minuten für über 96% der Inhalte schon Messwerte generiert und für ca. 92% genau in der 30. Minute.

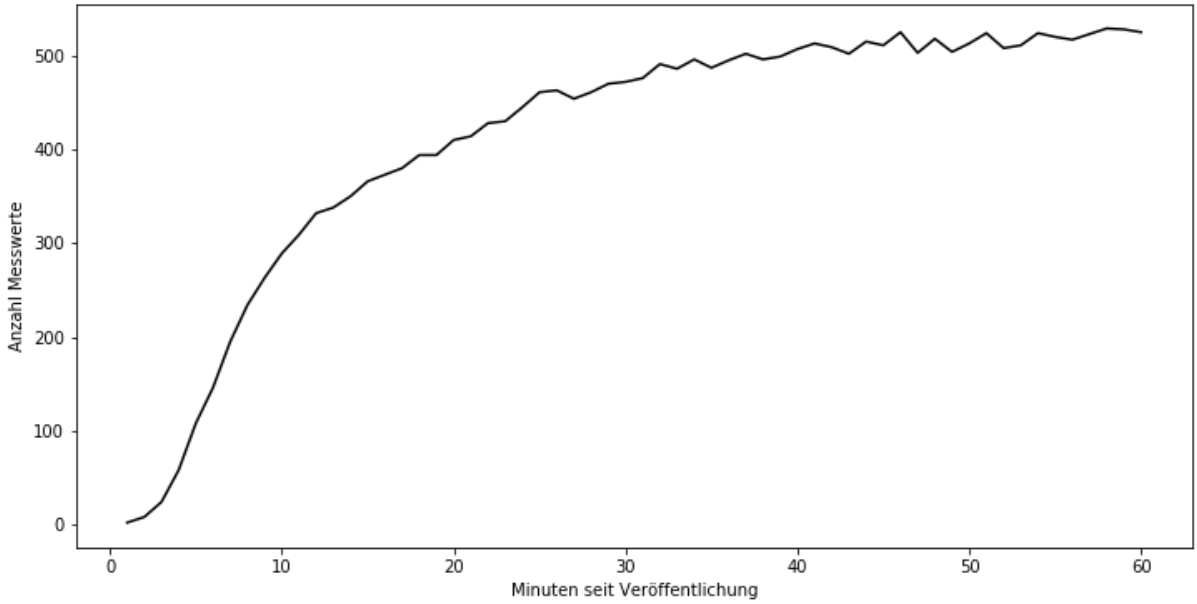

Abbildung 3-1: Anzahl Inhalte mit Messwerten nach vergangenen Minuten (vom 01.08.17-10.08.17)

Um die Reichweite vorhersagen zu können muss diese vereinheitlich werden, sodass sie für alle Inhalte den gleichen relativen zeitlichen Bereich abdeckt. Dieser wird dabei relativ zur Veröffentlichung gelegt, sodass der Zeitpunkt Null jeweils auf die Veröffentlichung des Artikels fällt. Die obere Schranke wurde aufgrund der Daten einmal 24 Stunden und 30 Tage ausgewählt. Dabei wird die Hypothese aufgestellt, dass nach 24 Stunden der Hauptwert eines Inhalts (einer Nachricht auf einer Nachrichtenseite) und damit seiner Reichweite erreicht ist. Um dies validieren zu können und einen Vergleichswert zu bestimmten wird als zweites 30 Tage festgelegt. Um die externen Faktoren besser einschätzen zu können sollten jeweils Daten für die innere (siehe 2.1.2) und gesamte Reichweite ermittelt werden, sodass sich folgender Zielvektor für das maschinelle Lernen ergibt:

$$\vec{Y} = \begin{pmatrix} gesamt\ Reichweite\ nach\ 24h \\ gesamt\ Reichweite\ nach\ 30d \\ interne\ Reichweite\ nach\ 24h \\ interne\ Reichweite\ nach\ 30d \end{pmatrix}$$

Formel 3-1: Reichweiten Zielvektor für einen Inhalt

Allgemein ist die Reichweite von Inhalten sehr ungleich Verteilt. Aus dem Verteilungsgraph für Mai 2017 (siehe unten) wird ersichtlich, dass die meisten Inhalte im Bereich von 0 bis 25000 liegen. Dabei generieren diese 58% (1.029/1.783) der

Inhalte nur etwa 17% (12.258.020/70.859.384) der Reichweite. Von Inhalten über 77400 Reichweite werden etwa 50% (35.422.599/70.859.384) der gesamten Reichweite erzeugt, was jedoch nur 13% (224/1783) der Inhalte entspricht.

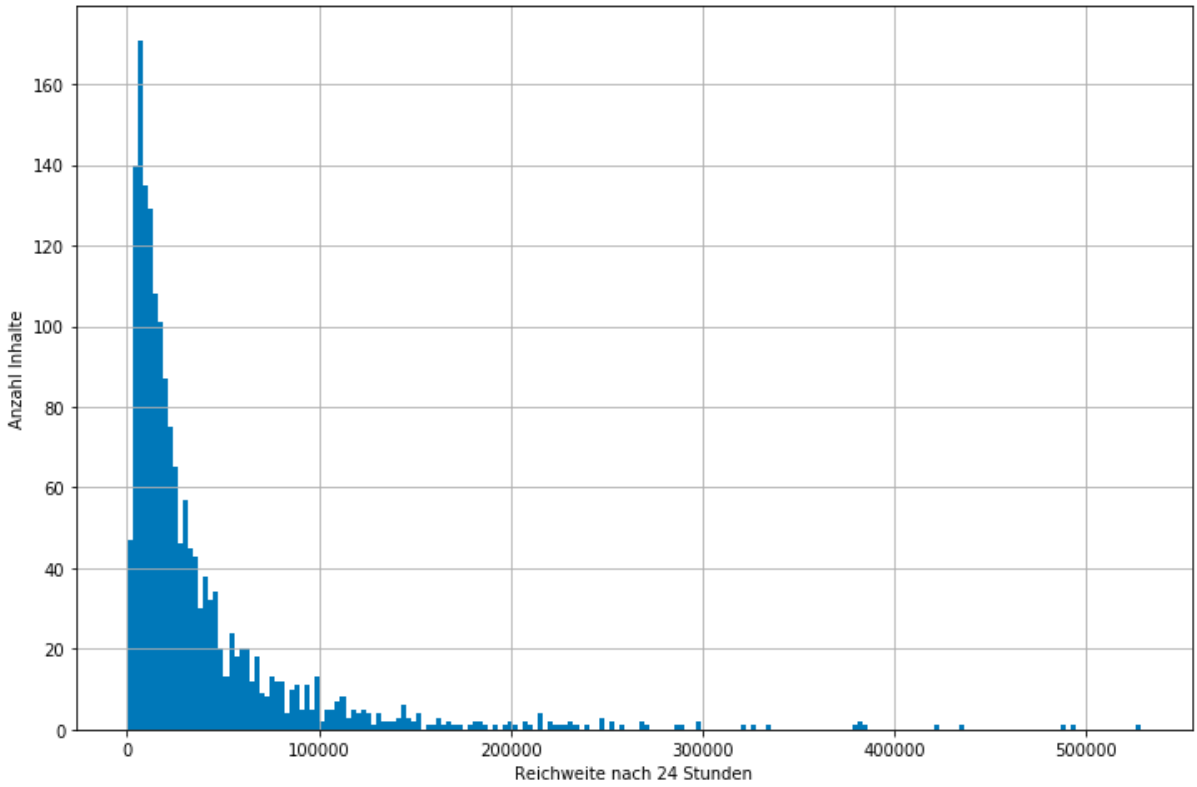

Abbildung 3-2: Reichweitenverteilung von Inhalten (05.2017)

Hierdurch ergibt sich auch, dass es zwar ca. 1800 Datenpunkte im Monat für die Reichweite gibt, jedoch wesentlich weniger für die Kuratierung interessant sind. Dies sollte in der Auswertung berücksichtigt werden.

Durch die durchschnittlich geringe Scrolltiefe der Nutzer und den Aufbau der Seite sind die oberen Plätze für den Großteil der Reichweite verantwortlich.

Um eine Erhöhung der gesamten Reichweite anhand der Kuratierung zu erreichen, ist es somit sinnvoll die wenigen Artikel mit hohem Potential schnell sichtbar auf die wenigen oberen Plätze zu legen. Hierdurch können auch andere Inhalte durch weiterführende Artikel profitieren.

3.2 Maschinelles Lernen

Die Idee beim Einsatz des maschinellen Lernens ist ein möglichst gutes Modell für die erreichte Reichweite von Inhalten zu finden, welches das Wissen abstrahiert, sodass dieses die Reichweite für neue Inhalte prognostizieren kann.

Durch die Abhängigkeit der festgesetzten Reichweite von anderen Merkmalen ist dies ein Regressionsproblem. Hierbei bietet sich überwachtes Lernen an, da durch die Vorgabe der Reichweite dessen Abhängigkeiten gelernt werden sollen. Es wäre jedoch auch möglich, grob ein Klassenproblem mit Intervallen zu definieren oder durch bestärkendes Lernen nur die Richtung vorzugeben. Da dies jedoch die Genauigkeit der Daten nicht berücksichtigen würde, wurde sich im Rahmen der Arbeit für überwachtes Lernen und Regression entschieden. Unüberwachtes Lernen wurde hingegen als nicht sinnvoll eingestuft für die Reichweitenvorhersage eingestuft.

Um geeignete Ansätze zur Vorhersage der Reichweite zu bestimmen, werden im Folgenden die Ausgangsbedingungen der Daten zusammengefasst:

- Die Reichweite liegt für die letzten 6 Monate Stundengenau vor
  Insgesamt für 75.669 Artikel
- Live-Daten sind nicht für alle Artikel verfügbar und wurden erst während der Masterarbeit gespeichert und sind dadurch nur für 3 Monate verfügbar
  Insgesamt für 5.434 Artikel
  - Von diesen nur 392 interessante (> 100.000 Reichweite) Artikel
- Viele Merkmale (102) in den Live-Daten
- Text muss gesondert betrachtet werden

Bei der Auswahl von Algorithmen muss somit auf die relativ geringe Anzahl von Trainingsdaten und dafür hohe Anzahl an Merkmalen geachtet werden.
Dabei wird in mehreren Quellen eine hohe lineare Korrelation zwischen log-transformierten Anzahlen von Klicks in einem kurzen Zeitintervall und der log-transformierten langfristigen Klickanzahl festgestellt. So wird in [19], die Anzahl der Retweets/Klicks in der ersten Stunde genutzt und ein einfacher linearer Faktor α zur Lösung von n_t durch n_0 vorgeschlagen z.B. $\log n_t = \alpha \cdot \log n_0$. Ebenso ging in [20] für das Ranking von Online Nachrichten das „Linear-log“-Modell als bestes Modell hervor. Aufbauend auf diesen Ergebnissen wäre für die Reichweite ein lineares Modell sinnvoll. Zusätzlich spricht für lineare Modelle, dass diese auch bei hoher Anzahl an Merkmalen schnell und effizient trainiert werden können. Jedoch muss in dieser Arbeit evaluiert werden, ob auch aus den gleichzeitigen Nutzern ein linearer Zusammenhang zu der Gesamtreichweite besteht bzw. die wenigen Daten ausreichen.

Der Logarithmus wirkt sich dabei auch positiv auf die Reichweitenverteilung aus (vorher Abbildung 3-2, nachher Abbildung 3-3). Durch die Transformation entsteht eine Normalverteilung.

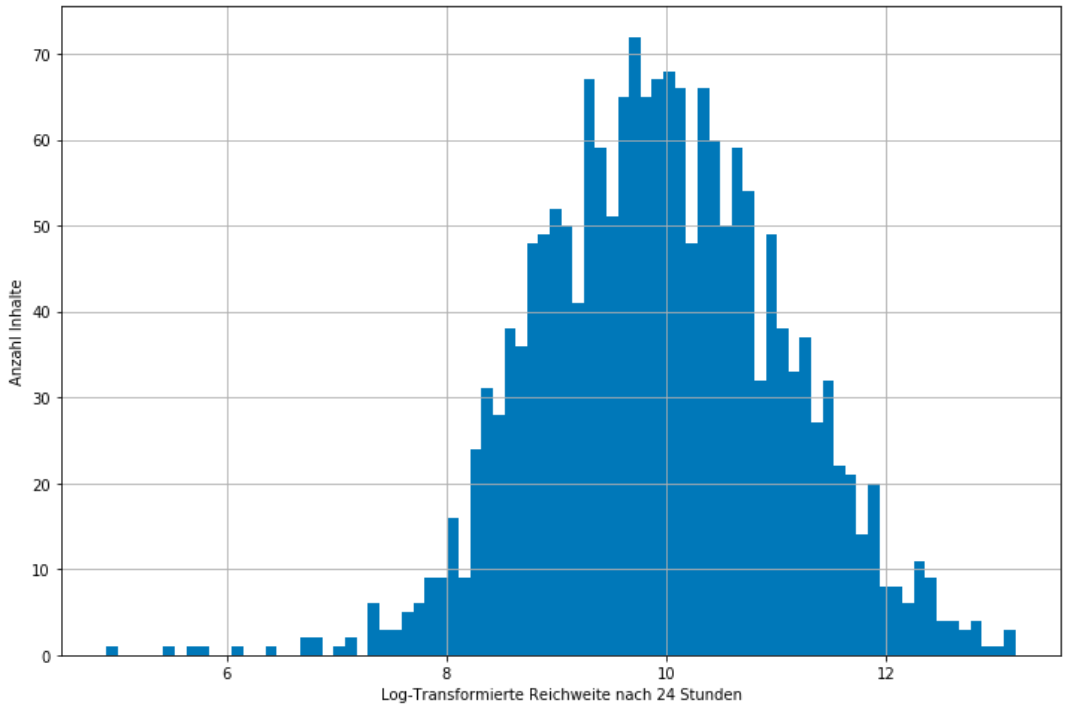

Abbildung 3-3: Reichweitenverteilung von Inhalten nach Log-Transformation (selbe Daten wie Abbildung 3-2)

Als Algorithmen wurden zusätzlich zur Multiplen Linearen Regression, Ridge Regression, Lasso Regression und Elastisches Netz ausgewählt, da sie durch Regulationen eine Lösung für die Auswahl der quantitativen Merkmale und deren starke Korrelation bereitstellen. Die Regulierungen und Anzahl an Parametern wurden in der nachfolgenden Tabelle zusammengefasst:

Algorithmus	**Regulierung**	**Anzahl Parameter**
Multiple Lineare Regression	Keine	Keine
Ridge Regression	L1	1
Lasso Regression	L2	1
Elastisches Netz	L1 und L2	2

Tabelle 3-2: Übersicht der genutzten Algorithmen für lineare Regression

Ein weiterer Vorteil von linearen Modellen ist die Möglichkeit Rückschlüsse aus den Koeffizienten zu ziehen. Dadurch können Einflüsse auf die Reichweite ausgewertet werden und so auch ggf. Fehler beim Modell erkannt werden.

Um zusätzlich einen Vergleich mit einem nicht linearen Modell zu bekommen, wurde für die Auswertung eine Kombination aus Hauptkomponentenanalyse und Entscheidungsbäumen gewählt. Die Hauptkomponentenanalyse reduziert dabei die Dimension, damit der Entscheidungsbaum bessere Unterscheidungen in diesen

machen kann. Da dieser durch die Reduktion der Dimension aber keine direkten Rückschlüsse mehr erlaubt und nicht in die lineare Betrachtung passt, wurde zugunsten der Fokussierung auf lineare Lerner und der Ähnlichkeit mit Doc2Vec keine genauere Betrachtung dieser Methoden in dieser Arbeit vorgenommen.
Eine weitere Annahme ist, dass die Reichweite inhaltlich ähnlicher Artikel miteinander korreliert. Jedoch lässt sich die Ähnlichkeit nicht eindeutig definieren und die Inhalte haben keine Label für die Ähnlichkeit, wodurch eine Optimierung und Auswertung erschwert wird.
Um dennoch die Reichweite auch von dem Inhalt des Artikels abhängig prognostizieren zu können, müssen neue Merkmale generiert werden. Da der Inhalt am besten durch den Text abgebildet wird und dieser in linearen Modellen nicht verarbeitet werden kann, muss für diesen auf die maschinelle Verarbeitung natürlicher Sprache (Engl. „Natural Language Processing“) zurückgegriffen werden.
Für die Analyse der Ähnlichkeit von Artikeln gibt es verschiedene statistische Möglichkeiten, welche in dieser Arbeit nicht näher beleuchtet werden. Aufgrund des Fokus auf maschinelles Lernen wurde Doc2Vec [21] wegen der guten Geschwindigkeit und Performanz ausgewählt. [22]
Aus dem Text können jedoch auch direkte Merkmale abgeleitet werden, wie z.B. Personen, Organisationen und Orte. Dies ist durch den Einsatz von verschiedenen „Named Entity Recognition“ (NER) möglich. Dabei zeigt sich im Vergleich [23], dass bereits gute Lösungen bestehen und die meisten Tools „generische Ansätze [nutzen] die es möglich machen jede Art von Text“ [23] zu verarbeiten. Dabei sind die Ergebnisse von Stanford NER [24], Illinois NET [25], OpenCalais WS [26], LingPipe [27] sehr nahe beieinander, sodass auf eine eigene Analyse verzichtet wurde. Als Tool wurde Stanford NER ausgewählt, da die Ergebnisse überzeugten und in das Tool auch für die deutsche Sprache Anpassungen eingeflossen sind.

3.3 Maschinelles Lernen als Dienst

Um die Ergebnisse im produktiven Einsatz nutzen zu können ist ein weiterer Teil dieser Arbeit eine Implementierung und Bereitstellung als Dienst. Dem Redakteur soll es möglich sein, intuitiv und gut integriert in die bestehende Kuratierung, Artikel mit hoher Reichweite zu erkennen. Dadurch wird dieser in der Lage versetzt, bessere Positionierungen vornehmen zu können. Ebenso werden Reaktionen auf unerwartet gut performende Inhalte möglich.

3.3.1 Anforderungen

Die folgenden Anforderungen wurden zu Beginn dieser Arbeit identifiziert und dabei in funktionale und nichtfunktionale unterschieden.

Nr.	Funktionale Anforderung	Priorität
1	Vorhersage der Reichweite für Inhalte	Höchste
2	Unterstützung verschiedener Algorithmen	Hoch
3	Automatische Datensammlung	Höchste
4	REST Schnittstelle	Hoch
5	„Health“ Schnittstelle	Mittel
6	NER Bereitstellung	Hoch
7	Integration in das bestehende Kuratierungstool	Höchste

Tabelle 3-3: Funktionale Anforderungen

Vorhersage der Reichweite für Inhalte

Die Hauptfunktionalität des Dienstes ist die Vorhersage der Reichweite für neue Inhalte. Dabei soll diese für 24 Stunden und 30 Tage nach der Veröffentlichung und jeweils für die interne und die gesamte Reichweite prognostiziert werden (siehe Formel 3-1).

Unterstützung verschiedener Algorithmen

Es sollten die folgenden Algorithmen aus Kapitel 3.2 unterstützt werden, um diese Vergleichen zu können:

- Multiple Lineare Regression
- Lasso Regression
- Ridge Regression
- Elastische Netze

Automatische Datensammlung

Um die Vorhersage live nutzen zu können, müssen die Daten für neue Inhalte immer auf dem neusten Stand zur Verfügung stehen.

REST Schnittstelle

Um eine Abstraktionsschicht zwischen dem maschinellen Lernen und der Integration zu erreichen, wurde eine REST Schnittstelle gewählt. Es wären auch andere Schnittstellen möglich, jedoch ist REST ein einfacher und robuster Standard, welcher sich schnell integrieren lässt.
Durch die extra Schicht wird eine lose Kopplung zwischen der Vorhersage und der Integration in das Kuratierungstool ermöglicht, sowie die Bereitstellung für andere Klienten.

„Health“ Schnittstelle

Der Dienst stellt eine „Health“ Schnittstelle per REST zur Verfügung, der den Status zurückgibt. Dies ist ein Standard um Dienste automatisiert Monitoren und ggf. neu starten zu können.

NER Bereitstellung

Für die Extraktion von Eigennamen gibt es bereits gute allgemeine Lösungen und gelernte Modelle. [23] Dabei sind diese jedoch eigenständige Problemlösungen und können gekapselt als eigener Dienst bereitgestellt werden. Wodurch diese Komponente beliebig skaliert werden kann und nicht jede Instanz des Dienstes den zusätzlichen Speicher für NER braucht. Ebenso ist eine Nutzung durch andere Dienste möglich.

Integration in das bestehende Kuratierungstool

Damit der Dienst durch Redakteure genutzt werden kann, ist eine Integration in das Kuratierungstool notwendig. Hierbei sollte für den Redakteur im Kuratierungstool ersichtlich werden, welches die Inhalte mit den besten Prognosen sind.

Nr.	Nichtfunktionale Anforderung	Priorität
1	Benutzerfreundlichkeit	Hoch
2	Vorhersagegüte	Höchste
3	Skalierbarkeit	Mittel
4	Wiederverwendbarkeit	Hoch
5	Kontinuierliche Integration	Hoch
6	Performanz	Hoch
7	Sicherheit	Mittel
8	Verfügbarkeit	Hoch

Tabelle 3-4: Nichtfunktionale Anforderungen

Benutzerfreundlichkeit

Die Nutzung soll für die Zielgruppe (Redakteure) einfach und übersichtlich gestaltet werden. Diese sollen auf den ersten Blick verstehen, welche Inhalte besonders viel Reichweite generieren werden.
Die Benutzerfreundlichkeit bzw. Nutzbarkeit ist so wichtig, da für den Erfolg des Dienstes die positive Interaktion mit dem Redakteur zwingend erforderlich ist.

Vorhersagegüte

Als Hauptziel dieser Arbeit ist die Qualität der Prognose entscheidend, um darauf Aufbauend die gesamte Reichweite verbessern zu können. Die Definition eines Schwellenwertes für die Vorhersagengüte ist aufgrund der nicht direkt messbaren Auswirkungen nicht möglich. Dennoch wird die Anforderung aufgenommen, um die Priorisierung der Optimierung der Vorhersagegüte abzubilden.

Skalierbarkeit

Die Skalierbarkeit des Dienstes kann in vertikale und horizontale Skalierbarkeit unterschieden werden. Wobei vertikale die Ressourcen eines Knotens meint und horizontale die Skalierbarkeit durch das Hinzufügen weiterer Knoten.

Beim Trade-off zwischen Trainings und Vorhersagedauer wird das Training meist komplex und die Vorhersage dafür schnell zu berechnen. Für das Training ist die Skalierbarkeit in beiden Richtungen somit von Vorteil. Für die Vorhersage dagegen eher die horizontale Skalierbarkeit zum Zwecke der Verfügbarkeit, da zusätzlich in erster Linie Redakteure Zugriff haben sollen und die Anfragen somit sehr begrenzt sind.

Wiederverwendbarkeit

Die Wiederverwendbarkeit ist einer der wichtigsten Faktoren um nicht nur einmaligen Nutzen aus der Implementierung dieser Arbeit zu ziehen. Es sollte eine Anpassbarkeit der Algorithmen und Frameworks geben, sodass die Grundstruktur des Dienstes für maschinelles Lernen sowohl für verbesserte Ansätze gerüstet ist, als auch für zukünftige Projekte genutzt werden kann.

Kontinuierliche Integration

Der Bauprozess sollte Kontinuierliche Integration (CI) unterstützen, wodurch mögliche lokale Fehlerquellen ausgeschlossen werden. Ebenso ist ein standardisiertes automatisches Bauen und Bereitstellung für die Weiterentwicklung von Vorteil.

Performanz

Die Performanz des Dienstes muss schnell genug sein um dem Redakteur Zeitnah (innerhalb von Sekunden) eine Vorhersage für neue Inhalte liefern zu können. Das Training von Modellen hat dagegen von dieser Seite keine Restriktionen, da es nur bei Änderungen neu ausgeführt werden muss. Jedoch sind kleinere Trainingszeiten für das explorative Arbeiten und die Optimierung von Parametern von Vorteil.

Sicherheit

Bei der Sicherheit ist zu beachten, dass ein Zugriff nicht öffentlich, sondern nur von Redakteuren möglich sein soll. Dabei sind die Daten weniger ein Sicherheitsrisiko, als der indirekte Zugriff auf das CMS um Artikeldaten

auszuwerten. Entsprechende Zugänge sollten durch entsprechende Maßnahmen geschützt werden.

Verfügbarkeit

Die Verfügbarkeit ist auch für die Annahme beim Redakteur wichtig. Solange jedoch keine vollständige Automatisierung vorgenommen wird, können Ausfälle vom Redakteur kompensiert werden und sind somit nicht Produktionsentscheidend.

3.3.2 Frameworks

Im Rahmen der Arbeit wurden mehrere Frameworks für maschinelles Lernen verglichen um ein für die Anforderung passendes Framework auszuwählen. Dafür wurden die unterstützten Algorithmen für Regression, die aktive Community, die Dokumentation, unterstützte Programmiersprachen, die Skalierbarkeit und die Einfachheit einer Implementierung herangezogen und in der folgenden Tabelle dargestellt.

Bei der Einfachheit einer Implementierung wurde eine einfache multiple lineare Regression mit Kreuzvalidierung umgesetzt und der Aufwand beschrieben. Die Auswertung werden hierbei durch eine Ordinalskala von Negativ (-), Neutral (o) und Positiv in 3 Abstufungen (+, ++, +++) durch eigene Einschätzungen abgebildet, um den Rahmen der Arbeit nicht zu sprengen.

Framework		TensorFlow	Scikit-Learn	MLlib	Gensim
Version		1.3.0	0.19.0	2.2.0	2.3.0
Unterstützte Programmiersprachen		Python, C++, Java, Go, C#[1], Haskell[1], Julia[1], Ruby[1], Rust[1], Scala[1]	Python	Java, Scala, Python	Python
Algorithmen für Regression (R.) Anzahl verwandter Algorithmen in Klammern		- Linear R. - Neural Network R.	- Linear R. - Generalized Linear R. (16) - Decision tree R. (3) - AdaBoost - Kernel ridge R. - Support Vector Maschines - Nearest Neighbors R. - Gaussian R. - Isotonic R. - Neural Network R.	- Linear R. - Generalized linear R. - Decision tree R. (3) - Survival R. - Isotonic R. - Neural Network R.	-[3]
Community	**Active Pull Requests**	315	106	82[2]	45
	Active Issues	553	134	179[2]	52
Doku-mentation	**Ausführlichkeit**	++	+++	++	o
	Genauigkeit	+++	++	+	++
Einfachheit einer Implementierung		- low-level	+++ schnell und einfach	+	++[3]
Skalierbarkeit		+++ (inkl. GPU)	++ (Algorithmen abhängig)	++	+++ (inkl. GPU)
Besonderheiten		Guter Guide, langsame Lernkurve	Benutzerfreundlich, sehr vielfältig, viele Tools	Nutzung von Spark vorgegeben	Speziell für NLP, Interfaces für Scikit-Learn

Tabelle 3-5: Vergleich der Frameworks TensorFlow, Scikit-Learn, MLlib und (Gensim)

[1] Unterstützung durch die Community
[2] Werte nicht vergleichbar, da für ganz Spark (https://issues.apache.org/jira)
[3] Gensim ist speziell für NLP, daher gibt es keine Regressionsalgorithmen (als Test wurde doc2Vec verwendet)

Aus der Analyse und Tabelle geht hervor, dass die verglichenen Frameworks verschiedene Einsatzzwecke dienen. So ist TensorFlow mehr auf die spezialisierten hoch performanten Implementierungen spezialisiert. Hierbei werden viele Werkzeuge und Interfaces in die Hand gegeben um eigene Lösungen im kleinsten Detail optimieren zu können. Dabei wird ein besonderer Fokus auf tiefe Neurale Netze gelegt. Dagegen ist das zu Spark gehörende MLlib gut in dessen Echtzeitanalyse und Streamingprozesse integriert. Es bietet standardisierte Lösungen für große Knotenanzahlen und die damit verbundenen Datenmengen. Durch die Verteilung lassen sich z.B. die Speicherbegrenzung einzelner Knoten lösen, jedoch kommen dadurch auch „Overhead“ und Spark Eigenheiten und Abhängigkeiten hinzu. Als eher wissenschaftliche Alternative wurde Scikit-Learn ausgewählt, welches ein sehr breites Spektrum an Algorithmen und Werkzeugen zur Verfügung stellt. Welche durch gute Schnittstellen und Konzepte leicht kombiniert und ausgewertet werden können. Dies steigert die Flexibilität und erleichtert somit die Findung und Implementierung. Dabei war vor allem die schnelle Lernkurve, die gute Dokumentation und die Werkzeuge zur Merkmalsextraktion entscheidend für die Festlegung auf Scikit-Learn. Da alle drei Frameworks zum Zeitpunkt der Arbeit aktiv weiterentwickelt werden und viel Unterstützung durch die Community erhalten, ist dies nicht weiter in die Entscheidung eingeflossen. Durch die Entscheidung für Scikit-Learn wurde Python als Programmiersprache für den Dienst ausgewählt, um die Implementierung so einfach wie möglich zu halten.

Da in Scikit-Learn jedoch nur beschränkte Textanalyse integriert ist, wurde zusätzlich Gensim als weiteres Python Framework ausgewählt um auch NLP durchführen zu können. Dieses enthält optimierte Implementierungen für Doc2Vec, Word2Vec und andere NLP Algorithmen. Des Weiteren werden für Doc2Vec und Word2Vec entsprechende Scikit-Learn kompatible Schnittstellen bereitgestellt.

Zu Beginn der Arbeit wurde auch eine Implementierung durch eine Cloud basierte SaaS Lösung in Betracht gezogen. Hierzu wurden Amazon Machine Learning [28], Google Cloud Machine Learning Engine [29] und Azure Machine Learning Studio [30] verglichen und Implementationen mit Testdaten durchgeführt. Danach wurde jedoch der Upload und die Verarbeitung der Artikel- und Trafficdaten in den entsprechenden Clouds als problematisch eingestuft, da es sich um kritische Daten handelt. Weil eine

Lösung im Rahmen von der WeltN24 Zeitaufwendig erschien, wurde auf eine Ausführung und weitere Betrachtung von SaaS verzichtet.

3.4 Zusammenfassung

Um die Reichweite insgesamt durch die Kuratierung zu verbessern, wird angenommen, dass sich diese für einzelne Artikel durch maschinelles Lernen prognostizieren lässt. Dabei wird die Reichweite durch viele Faktoren beeinflusst, für welche geeignete Merkmale gefunden werden müssen. Es ist zudem sinnvoll Daten von den ersten 30 Minuten nach der Veröffentlichung einzubeziehen, um das Modell genauer zu machen. Die folgenden Regressionsalgorithmen wurden für eine genauere Prüfung ausgewählt: Multiple Lineare Regression, Lasso Regression, Ridge Regression. Durch die Linearität ist es möglich den Einfluss von Merkmalen durch ihre Koeffizienten leicht abzulesen, wodurch sich auch weitere Aussagen über die Einflüsse auf die Reichweite erzielen lassen. Der Dienst besteht einerseits aus der Integration in das Kuratierungstool, welches vom Redakteur angenommen werden muss, um Erfolgreich zu sein und andererseits aus der Vorhersage, welche möglichst gute Ergebnisse und Flexibilität bieten muss. Als Framework wurden Scikit-Learn und Gensim aufgrund ihres breiten Spektrums an Algorithmen und Werkzeugen, ihres leichten Einstiegs und guten Dokumentation festgelegt. Damit einhergehend wurde für den Dienst Python als Programmiersprache gewählt.

4 Konzept

In diesem Kapitel werden Lösungen für die in der Analyse identifizierten Probleme und Anforderungen erarbeitet. Zuerst wird auf die Reichweite und die extrahierten Merkmale eingegangen, um mit diesen im maschinellen Lernen eine Modellierung der Reichweitenvorhersage vornehmen zu können. Im Anschluss wird die Architektur für den Dienst konzipiert und erläutert.

4.1 Reichweiten

In der Analyse der Reichweite wurden die Einflüsse grob in Kategorien eingeteilt. Um diese abzudecken und für die Vorhersage zu nutzen, wurden im folgenden Merkmale für die Vorhersage ausgearbeitet. Dabei wurden diese ähnlich der Kategorien eingeteilt in Metadaten, Inhaltliche, Zeitliche und Social-Trend Merkmale.

4.1.1 Metadaten

Die Metadaten sind durch das CMS von „der WELT“ gegeben und enthalten den Titel, den Teasertext, den vollständigen Inhalt, den Typ, die Sektion, den Premiumstatus, die Anzahl der Bilder, die Anzahl der Videos und das Veröffentlichungsdatum. Das Veröffentlichungsdatum wird jedoch zusätzlich in Wochentag und die Uhrzeit unterteilt.

4.1.2 Inhaltsbasierte Merkmale

Aufbauend auf dem Inhalt sind die folgenden Merkmale ausgearbeitet worden

- **Reichweite ähnlicher Inhalte**. Um die Interessen der Nutzer abzubilden ist die Reichweite von ähnlichen Inhalten ein vielversprechendes Maß. Dabei wird die Reichweite der 10 ähnlichsten Artikel verwendet, um die Reichweite des neuen Artikels vorherzusagen. Dabei lassen sich anhand der Sortierung verschiedene Merkmale erstellen. Einerseits lassen sich aus den Werten Merkmale für die größte bis kleinste Reichweite der Top10 ähnlichsten Artikel bilden, welche somit das Maximum, Minimum und die Verteilung abbilden und dadurch jeweils in Proportion zur neue Reichweite stehen können. Andererseits lassen sich Merkmale für den ähnlichsten, zweitähnlichsten Artikel usw. durch dessen Reichweite bilden, durch welche die Ähnlichkeitsbeziehung stärker Beachtung findet.
- **Named Entities**. Eine weitere Möglichkeit zur Inhaltsanalyse besteht durch die Extraktion von im Artikel verwendeten Personen, Orten, Organisationen. So

haben Artikel über bekannte Personen oder Organisationen meist schon aus dem Grund der Bekanntheit eine größere Reichweite. Bei der Nutzung der Personen wurde sich jedoch auf die einfache Anzahl der Personen, Orte und Organisationen beschränkt.

- **Neuheit des Artikels**. Ein wichtiger Faktor für die virale Verbreitung ist die Neuheit des Inhalts. [31] Dabei sind neue Artikel durch ihre meist unerwarteten Themen nicht durch die bestehenden Daten abgedeckt. Um für diese ein extra Maß zu haben, wurde für die Neuheit des Artikels zwei Merkmale gengeriert. Einerseits wurden die Ähnlichkeitswerte der Top 10 einbezogen:

$$Neuheit = 1 - \frac{1}{N}\sum p_i$$

Wobei p_i den Ähnlichkeitswert und N die Anzahl an genutzten Werten (in diesem Fall 10) beschreibt. Durch die Restbildung zu 1 ergibt sich ein Wert, der nahe 0 ist, wenn die Top 10 sehr Ähnlich sind, und nahe 1, wenn die Top 10 kaum Übereinstimmungen haben.

Andererseits wurde die Anzahl an Artikeln über einem bestimmten Schwellenwert des Ähnlichkeitswertes benutzt. Hierfür wurden 3 Merkmale mit den Schwellenwerten 0,6; 0,8 und 0,9 erstellt.

$$Neuheit_2 = \frac{\beta}{\beta + |M|} \; mit \; M = \{a \mid \ddot{a}hnlichkeit(a, A) > \lambda\}$$

Wobei der Schwellenwert λ ist und A der neue Artikel. Der Kehrwert erzeugt ebenfalls einen Wert nahe 0, wenn viele Artikel ähnlichkeiten über λ haben und nahe 1, wenn wenige Artikel Ähnlich sind. Zu Beginn wurde die Formel mit $\beta = 1$ genutzt und später auf $\beta = 10$ angepasst, um eine bessere Verteilung nahe 1 und folglich mehr Robustheit gegen einzelne Artikel mit hoher Ähnlichkeit zu erreichen.

- **Lesbarkeit**. Über die virale Verbreitung und der daraus resultierenden Reichweite schreiben Berger und Milkman [32], dass eine positive Abhängigkeit zwischen langen Artikeln und der vermehrten Teilung in sozialen-Medien besteht. Die Autoren vermuten, dass Redakteure bei brisanten Themen längere Artikel schreiben. Dies konnte jedoch nicht in den Daten „der WELT“ festgestellt werden. Zur Bestimmung der Lesbarkeit wird der Lesbarkeitsindex Flesch-Reading-Ease [33] genutzt, welcher durch Toni Amstad [34] auf die deutsche Sprache übertragen wurde.

  $$FRE_{deutsch} = 180 - ASL - (58{,}5 \cdot ASW)$$

  Wobei ASL die durchschnittliche Satzlänge und ASW die durchschnittliche Silbenanzahl pro Wort angibt. Es wurden außerdem die Titellänge, die Teaserlänge, die Textlänge, die Wortanzahl, die durchschnittlichen Wörter per Satz und die Anzahl der Sätze als Merkmale hinzugefügt.

4.1.3 Zeitliche Merkmale

Wie in der Analyse beschrieben können die Daten der ersten 30 Minuten Aufschluss über das Weltgeschehen und Trends geben, die in den Trainingsdaten nicht abgebildet werden können. Diesbezüglich wurden die folgenden zeitlichen Merkmale erzeugt:

- **Daten der ersten 30 Minuten.** Als Grundlage für diesen Bereich werden dazu die Rohdaten in Merkmale überführt. Die aufgezeichneten Chartbeatdaten sind minutengenau und haben pro Artikel jeweils 102 Werte wovon 38 unterschiedliche Maße abbilden (Auflistung in Anhang II). Da die Menge an Merkmalen mit 30 mal 102 zu groß wäre und die minütlichen Werte größeren Schwankungen unterliegen, wurde durch Resampling mit verschiedenen Aggregationsfunktionen und Pivot-Tabellen in 5 bzw. 15 Minuten Abschnitten Merkmale erstellt.
- **Beschleunigung der Nutzerzahlen in 30 Minuten**. Dieses Merkmal wurde von [35] übernommen und wird wie folgt berechnet:

  $$Beschleunigung = \frac{\sum_{t=2}^{N} n_t^x - n_{t-1}^x}{N}$$

  Wobei n_t^x die Anzahl der Seitenbesuche von Artikel x im Zeitintervall t und N die Gesamtanzahl von Zeitintervallen ist. Da in dieser Arbeit anstatt Reichweite für die ersten 30 Minuten nur die jeweils gleichzeitigen Nutzer gemessen werden

und diese nicht stetig steigend sind, muss dieser Wert im Vergleich zu [35] nicht positiv sein. Dies sollte jedoch nicht den Nutzen der Metrik beeinträchtigen.

- **Erstes Auftreten**. Die Zeitdauer, bis ein Artikel in den Inhalten mit den gleichzeitig meisten Nutzern und damit in Chartbeat auftaucht, gibt einen Einblick in die Situation in der der Artikel publiziert wird.
- **Relative Besuche**. Um ein mehr vergleichbares Merkmal zu schaffen, welches im Vergleich besonders starke Artikel hervorhebt und mehr Unabhängigkeit von der aktuellen Situation schafft, wurden relative Merkmale entwickelt. Hierfür werden die Chartbeatwerte der ersten 30 Minuten für Artikel zu denen der Startseite in Relation gesetzt. Dabei soll dies jedoch als ein getrennter Vergleich zu den herkömmlichen Chartbeatdaten umgesetzt werden um die Merkmalsanzahl nicht zu verdoppeln.

4.1.4 Social-Trend Merkmale

Um die sozialen Trends besser abbilden zu können wurde bei der Konzeption eine Nutzung von Twitter bzw. Facebook Daten angedacht um entsprechende Tweets oder Kommentare auszuwerten. Dabei wurden zwei Probleme festgestellt:

1. Veröffentlicht „die WELT“ nur bestimmte Artikel in sozialen Medien, wodurch zu diesen Artikeln gesammelte Daten zu einer Verzerrung führen würden.
2. Zusätzlich wurde die Findung von Tweets/Posts zu einem Artikellink betrachtet, bei denen aufgrund des direkten Seitenbezugs weniger Verzerrung erwartet wurde. Jedoch konnten entsprechende Werkzeuge zur Analyse nicht frei zugänglich gefunden werden.

Die Probleme führten dazu, dass aufbauend auf den Daten der ersten 30 Minuten folgende Merkmale zur sozialen Verbreitung erarbeitet wurden:

- **Kommentare**. Die Anzahl schreibender Nutzer und die Anzahl der Kommentare in den ersten 30 Minuten wurden als Merkmale für die virale Wirkung mit einbezogen.
- **Soziale Verbreitung**. Die soziale Verbreitung wurde anhand der Besucher von anderen Seiten gemessen. Dabei werden zuerst die Seiten bestimmt, von denen insgesamt am meisten Nutzer kommen und für jede Seite ein eigenes Merkmal erstellt. Ein weiteres Merkmal gibt zusätzlich die Summe aller externen Besucher an.

Metadaten Merkmale	
Typ	News Artikel, Eilmeldung, Bildergalerie…
Sektion	Channel, in dem der Artikel veröffentlicht wurde. Diese werden von unterschiedlichen Nutzergruppen konsumiert.
Veröffentlichungszeitpunkt	Das Veröffentlichungsdatum wurde mit der Tageszeit und dem Wochentag berücksichtigt.
Autorname	Der Name des Autors. Bestimmte Autoren ziehen besondere Aufmerksamkeit auf sich.
Premium	Wahrheitswert, ob der Inhalt als Premium deklariert ist. Premiuminhalte sind auf der Startseite gekennzeichnet und können nur mit Abo vollständig gelesen werden.
Bilder	Anzahl der Bilder
Videos	Anzahl der Videos
Inhaltliche Merkmale	
Reichweite ähnlicher Inhalte	- Reichweite Top 10 ähnlichste Inhalte - Ähnlichkeits-Score der Top 10 ähnlichsten Inhalte
Named Entities	Anzahl der extrahierten Personen, Organisationen und Orte
Neuheit des Inhalts	Zwei auf der Ähnlichkeit aufbauende Metriken zur Neuheit
Lesbarkeit des Textes	Flesch-Reading-Ease Index und verschiedene Messwerte zum Text (u.a. Textlänge, Wortlänge, Wörter pro Satz)
Zeitliche Merkmale	
Erste 30 Minuten	Daten der ersten 30 Minuten durch Chartbeat
Beschleunigung in 30 Min.	Die Beschleunigung der gleichzeitigen Besucher
Erstes Auftreten	Zeit bis zum ersten Datenpunkt
Relative Besuche in 30 Min.	Das Verhältnis von Artikel zu Startseitenbesuchen
Social-Trend Merkmale	
Kommentare	Anzahl schreibender Nutzer Anzahl Kommentare
Soziale Verbreitung	Nutzer von anderen Plattformen

Tabelle 4-1: Übersicht über die evaluierten Merkmale

Die Nutzung von Autoren wäre für die Vorhersage ein weiterer guter Indikator und wurde zu Testzwecken verwendet, dennoch wurde sich gegen eine Nutzung ausgesprochen. Da bei der Verwendung die gelernten Koeffizienten eine Voreingenommenheit bei der Auswahl von Redakteuren darstellt und somit keine Fairness zwischen diesen garantiert wird.

4.2 Maschinelles Lernen

In diesem Abschnitt wird der Aufbau für die Vorhersage erarbeitet. Da die Reichweitendaten der Ausgangspunkt für die Modellierung sind, wird zuerst die Berechnung des Reichweitenvektors als Zielvektor erläutert. Danach werden die verschiedenen Konzepte für die Merkmalsgenerierung (Kodierung der Zeit, Relative-Chartbeatwerte und Resampling) behandelt. Im Anschluss wird ein getrenntes ML-Konzept zur Lösung für die Ähnlichkeitsbeziehung durch Doc2Vec aufgestellt. Zum Schluss wird kurz das entstehende Modell und das Konzept zur Optimierung von Parametern beleuchtet.

4.2.1 Der Reichweitenvektor

Die Quelle für die Reichweite ist durch die eingebundenen Webtracking-Tools auf Webtrekk beschränkt. Webtrekk bietet keine Möglichkeit als Dimension die Zeit seit Veröffentlichung zu nutzen. Jedoch ist die Reichweite von Inhalten für Zeitbereiche verfügbar, da meist auf Tages oder Monatsbasis Reichweiten analysiert werden.
Durch die alleinige Nutzung von Tagen als Dimension (jeweils von 0 Uhr bis 23:59 Uhr) wäre jedoch ein großer Fehler entstanden, sodass auf die kleinstmögliche Dimension Stunden zurückgegriffen wird.
Um die Reichweite zu erhalten wird bei jedem Datenpunkt die Differenz zum Veröffentlichungszeitpunkt berechnet und dann die Summe über allen Reichweiten bis zum 24 Stunden bzw. 30 Tage Limit gebildet.

$$Y(Inhalt)_{24h} = \sum_{t_i < V(Inhalt)+24h} Reichweite(Inhalt, t_i)$$

Wobei V(Inhalt) den Veröffentlichungszeitpunkt und t_i die Messzeiten beschreibt.
Hierbei ist der Fehler für 30 Tage selbst bei einer naiven Annahme von gleicher Verteilung mit $e_{30d} < \frac{1}{24 \cdot 30} = \frac{1}{720} \approx 0.15\%$ verschwindend gering. Bei gleicher Annahme wäre bei 24 Stunden die Abweichung mit $e_{24h} = \frac{1}{24} \approx 4.2\%$ recht hoch, jedoch ist laut Hypothese nach 24 Stunden der Inhalt kaum noch aktuell, sodass der vermutete niedrige anstieg auch mit einem geringen statistischen Fehler einhergehen müsste. Dies wird durch die Daten bestätigt, wie in Abbildung 4-1 zu erkennen ist.

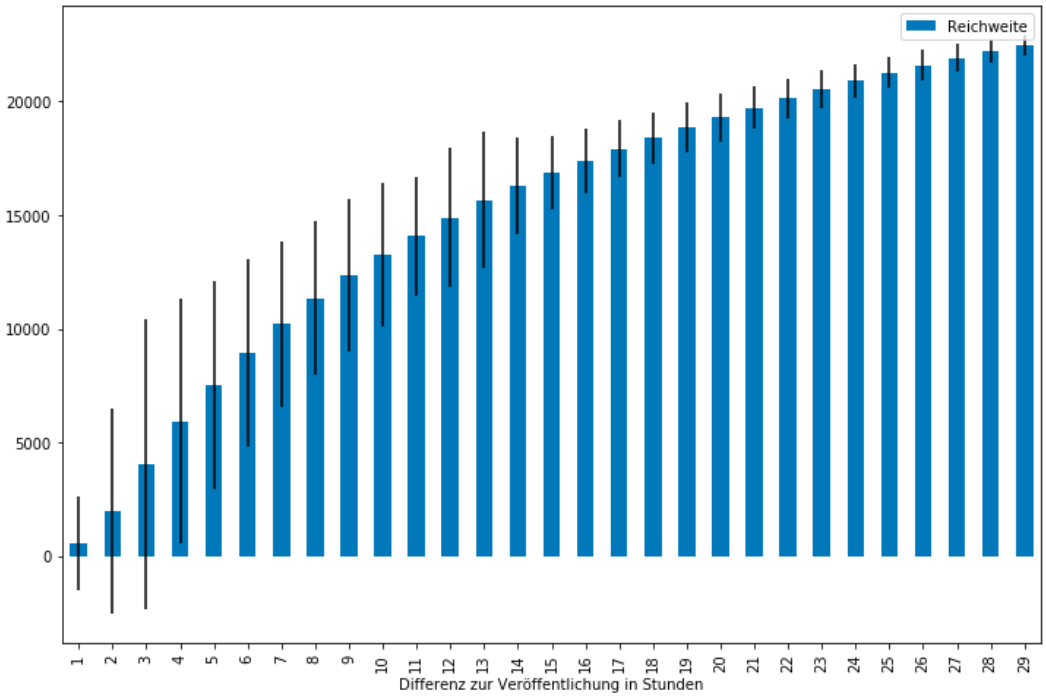

Abbildung 4-1: Durchschnittliche Reichweitenentwicklung mit Fehler der Zunahme in der jeweiligen Stunde (05.2017-07.2017)

4.2.2 Kodierung der Zeit

Um den wichtigen Faktor der Uhrzeit der Veröffentlichung in das lineare Modell einfließen zu lassen, ist es nicht möglich nur den numerischen Stunden Wert als ein Merkmal zu nutzen. Dies erklärt sich daraus, dass eine lineare Abhängigkeit zwischen dem 0-24 Stunden Intervall und der Reichweite nicht gegeben ist.

Um die Uhrzeit dennoch zu nutzen, können Intervalle und somit Kategorien definiert werden z.B. Stundenintervalle: [0-1), [1-2), …, [23-24). Bei linearen Modellen wird somit für jedes Intervall ein zusätzliches Gewicht hinzugefügt, welches die Abweichung beschreibt. Um zusätzlich zu den harten Intervallen fließende Übergange zwischen den Zeiten zu testen, wurde die in Abbildung 4-2 gezeigte alternative Entwickelt. Dabei wird für eine bestimmte Anzahl Merkmale eine verschobene Cosinusfunktion im Intervall $\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$ verwendet. Die Idee ist, dass eine bestimmte Anzahl Gewichte, je nach Abstand zur Veröffentlichung unterschiedlich stark beeinflusst werden. Um die Anzahl betroffener Gewichte zu steuern wurde zusätzlich eine Spreizungskonstante eingefügt. Dabei wurde die folgende Formel aufgestellt:

$$M_i = \begin{cases} \cos\left(\frac{t_V - 24\frac{i}{N}}{\alpha} \cdot \frac{\pi}{2}\right), & \left|t_V - 24\frac{i}{N}\right| < \alpha \\ 0, & sonst \end{cases}$$

Wobei M_i das i-te Merkmal, N die Anzahl der Merkmale, t_V die Veröffentlichungszeit in gebrochenen Stunden und α der Spreizungsfaktor ist.

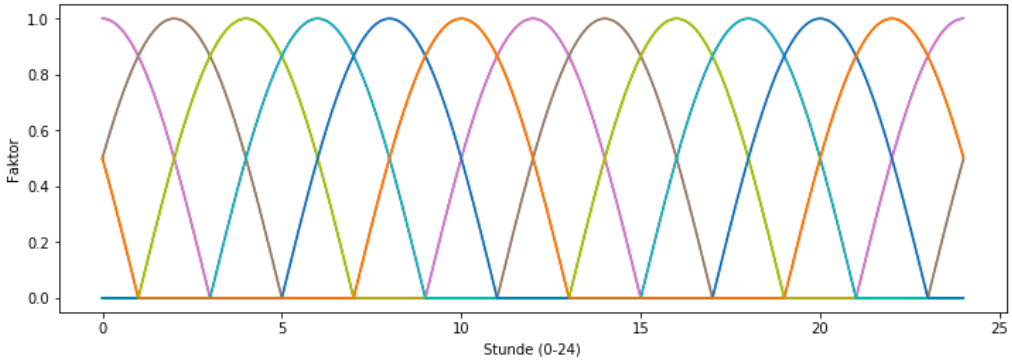

Abbildung 4-2: Zeitlich fließende Übergänge durch Überlagerung (Beispiel 2 Stunden)

4.2.3 Resampling der Chartbeatwerte

Da die großen Schwankungen (siehe Abbildung 4-3) in den minütlichen Daten von einem linearen Modell nicht gut ausgeglichen werden können, wurde Resampling mit verschieden Aggregationsfunktionen durchgeführt um die Daten zu glätten und so abstrahiertes Wissen vorzubereiten. Die Minutengenaue Übernahme der Chartbeatdaten in Merkmalen hätte außerdem zu einer unüberschaubar großen Menge an Merkmalen geführt.

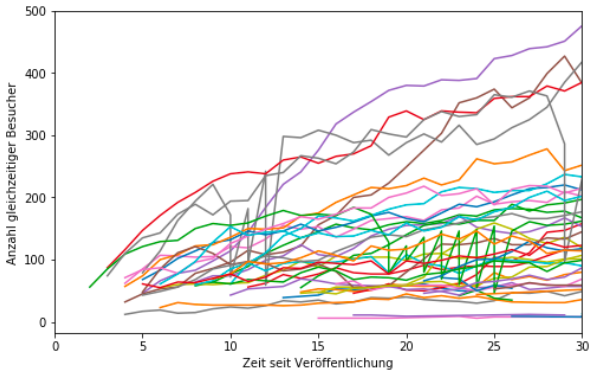

Abbildung 4-3: Gleichzeitige Besucher auf Artikeln

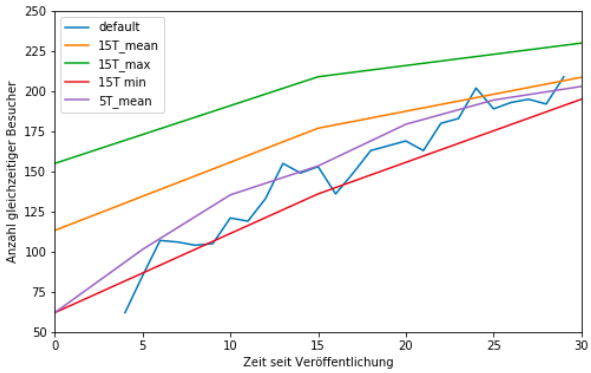

Abbildung 4-4: Resamplingfunktionen am Beispiel

Dabei wurden 5 Minuten und 15 Minuten Intervalle gewählt um darunterliegende Funktion in 3 bzw. 7 neuen Messwerten abzubilden. Als Aggregationsfunktionen wurden Min, Max, Mittelwert und die kumulative Summe (cusum) ausgewählt, wobei cusum aufgrund seiner Ähnlichkeit zur Reichweitenverteilung gewählt wurde. Die Ergebnisse der Funktionen sind an einem Beispiel in Abbildung 4-4 dargestellt, jedoch aufgrund der Größenordnung ohne cusum.

4.2.4 Relative Chartbeatwerte

Um eine bessere lineare Abhängigkeit zu erreichen wurde die Idee von der Verwendung relativer Chartbeatdaten entwickelt. Dabei sind die Werte der Startseite (siehe Abbildung 4-5) der Bezugspunkt, um die Daten der ersten 30 Minuten zu relativieren. Hierdurch sollen die unterschiedlichen Ausgangsbedingungen (siehe Abbildung 4-6) stärker als nur durch die Zeit berücksichtigt werden. Die Abbildung 4-7 zeigt, dass der relative Durchschnitt wesentlich gleichmäßiger Verteilt ist und so die Stellen mit hohen Abweichungen herausstechen.

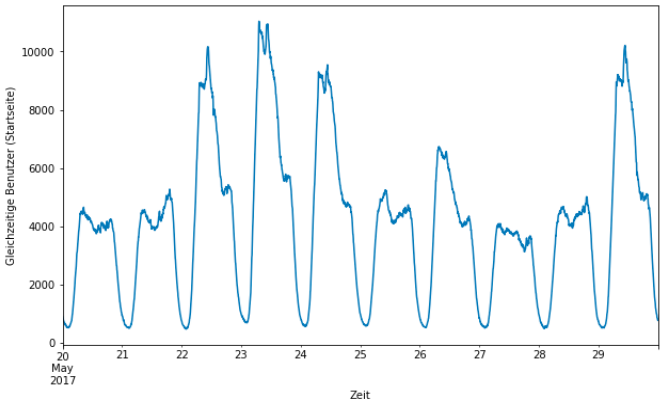

Abbildung 4-5: Gleichzeitige Besucher auf der Startseite (20.05.17-30.05.17)

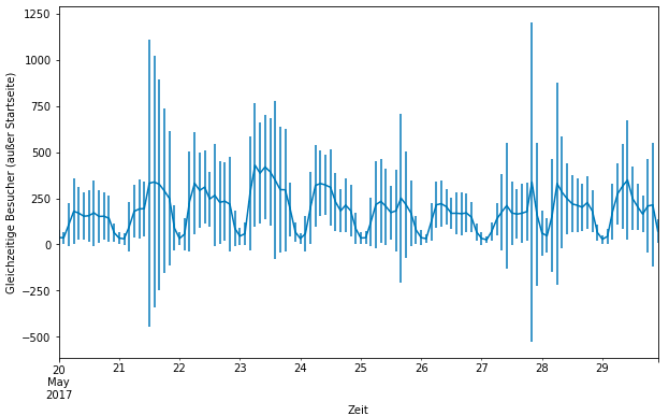

Abbildung 4-6: Mittlere gleichzeitige Besucher mit Fehlern (ohne Startseite)

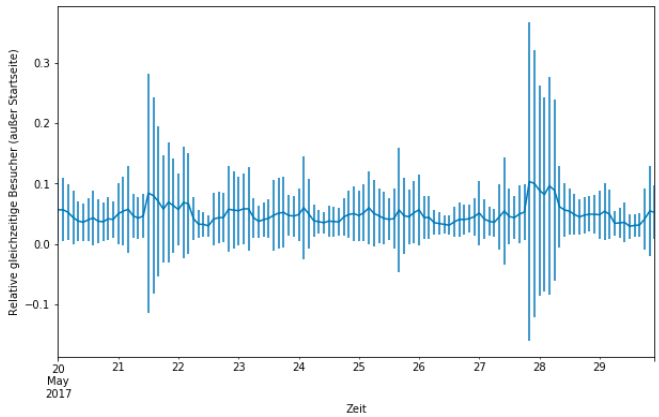

Abbildung 4-7: Mittlere relative gleichzeitige Besucher mit Fehlern (ohne Startseite)

4.2.5 Ähnliche Inhalte

Die Bestimmung ähnlicher Inhalte wurde in der Analyse als schwieriges Problem identifiziert. Das Fehlen einer genauen Berechnung der realen Ähnlichkeit macht eine Aussage über das Modell schwierig. Das Doc2Vec Modell kann einerseits durch den Trainingserror optimiert werden, was jedoch keine Aussagekraft über die Sinnhaftigkeit der Ähnlichkeit des Modells hat. Andererseits ist es möglich das Ähnlichkeitsmodell anhand der Hypothese des Reichweitenzusammenhangs zu optimieren. Dafür wird anhand einer linearen Regression und des R^2 Maß, die Erklärbarkeit der Reichweitenvarianz durch die Reichweiten der ähnlichen Artikel errechnet. Dabei können als Korpus alle Artikel mit vorhandener Reichweite und Bodytext benutzt werden. Die Verwendung von 68.163 Artikeln sollte eine ausreichende Basis darstellen. Eine Optimierung der Hyperparameter im Sinne der „richtigen“ Ähnlichkeit lässt sich nicht durch eine Testmenge validieren. Es kann jedoch der Einfluss der Merkmale auf die Reichweitenvorhersage durch eine lineare Regression gemessen werden. Die Hyperparameter werden somit auf die Vorhersagegüte der Reichweite optimiert.

4.2.6 Das Log-Transformierte Modell

Die lineare Abhängigkeit zwischen Kurz- und Langzeitmesswerten für Besucher/Klicks sind in der Analyse für deren Log-Transformationen von verschieden Quellen festgestellt worden. Jedoch wurde nicht beleuchtet in welchem Zusammenhang die gleichzeitigen Nutzer stehen. Ein linearer Zusammenhang ist bei den Log-Transformierten Daten in dieser Arbeit nicht direkt erkennbar. Jedoch konnten im Vergleich zu negativen Werten ohne Logarithmus immerhin 0,340 +/- 0,042 der Varianz allein durch die gleichzeitigen Benutzer erklärt werden. Da dies jedoch viel zu gering ist, wurde er Fokus vor allem auf die Findung allgemeinerer Merkmale gelegt.

Für den Aufbesserungseffekt wird die gleichmäßigere Verteilung der Logarithmischen Reichweite angenommen und somit für alle Daten verwendet.
Dadurch entsteht für das lineare Modell die folgende Gleichung:

$$\log(Y) = \log(X)\beta + e$$

Wobei Y der Zielvektor mit jeweils einer Reichweite, X die Matrix der unabhängigen Variablen, β der Gewichtsvektor und e der Fehler ist. Die Gewichte lassen sich durch den Logarithmus nicht mehr linear interpretieren.
Eine Optimierung des Modells erfolgt anhand des angepassten R^2-Wertes, welcher die Erklärung der Varianz durch die unabhängigen Variablen entspricht.

4.2.7 Hyperparameteroptimierung

Bei den ausgewählten Lernern können bei Ridge und Lasso jeweils ein Parameter und beim elastischen Netz zwei Parameter zur Regulierung von den Gewichten optimiert werden. Jedoch sind die Lernzeiten und Parameteranzahlen so gering, dass eine große Menge an Parametern probiert werden kann, wodurch eine komplexere Strategie überflüssig wird. Um die optimalen Werte zu finden wurde Rastersuche (engl. grid search) ausgewählt, da es eine genaue, einfach zu verwendende Methodik ist.

4.3 Architektur

Ein Überblick über die erstellte Architektur wird in Abbildung 4-8 gegeben. Im Folgenden werden die Designentscheidungen für diesen Aufbau genauer erläutert.
Die Implementierung des maschinellen Lernens und der Integration werden voneinander getrennt um eine Entkopplung der Logik und des Benutzerinterfaces zu erreichen. Dies hat sowohl den Vorteil, dass der Dienst fürs maschinelle Lernen leicht ausgetauscht werden kann, als auch, dass die Schnittstelle für andere Dienste gut definiert ist. Ebenso wird das verwendete Stanford NER in einen eigenen Dienst ausgelagert, um diesen einzeln Skalieren zu können (sowohl horizontal, als auch vertikal) und ihn auch durch andere Dienste erreichbar zu machen. Die durch Stanford NER gebrauchte Java-Umgebung wird so auch sauber getrennt.
Die erforderlichen Daten werden hierbei aus Amazons Speicherlösung S3 und indirekt aus dem CMS abgeholt. Wobei die Chartbeatdaten für die Vorhersage gebraucht werden und somit automatisch abgeholt und gespeichert werden müssen. Hierfür wird jede Minute eine Funktion in einem AWS-Lambda ausgeführt, welche die Chartbeat-API abruft und auf S3 speichert.

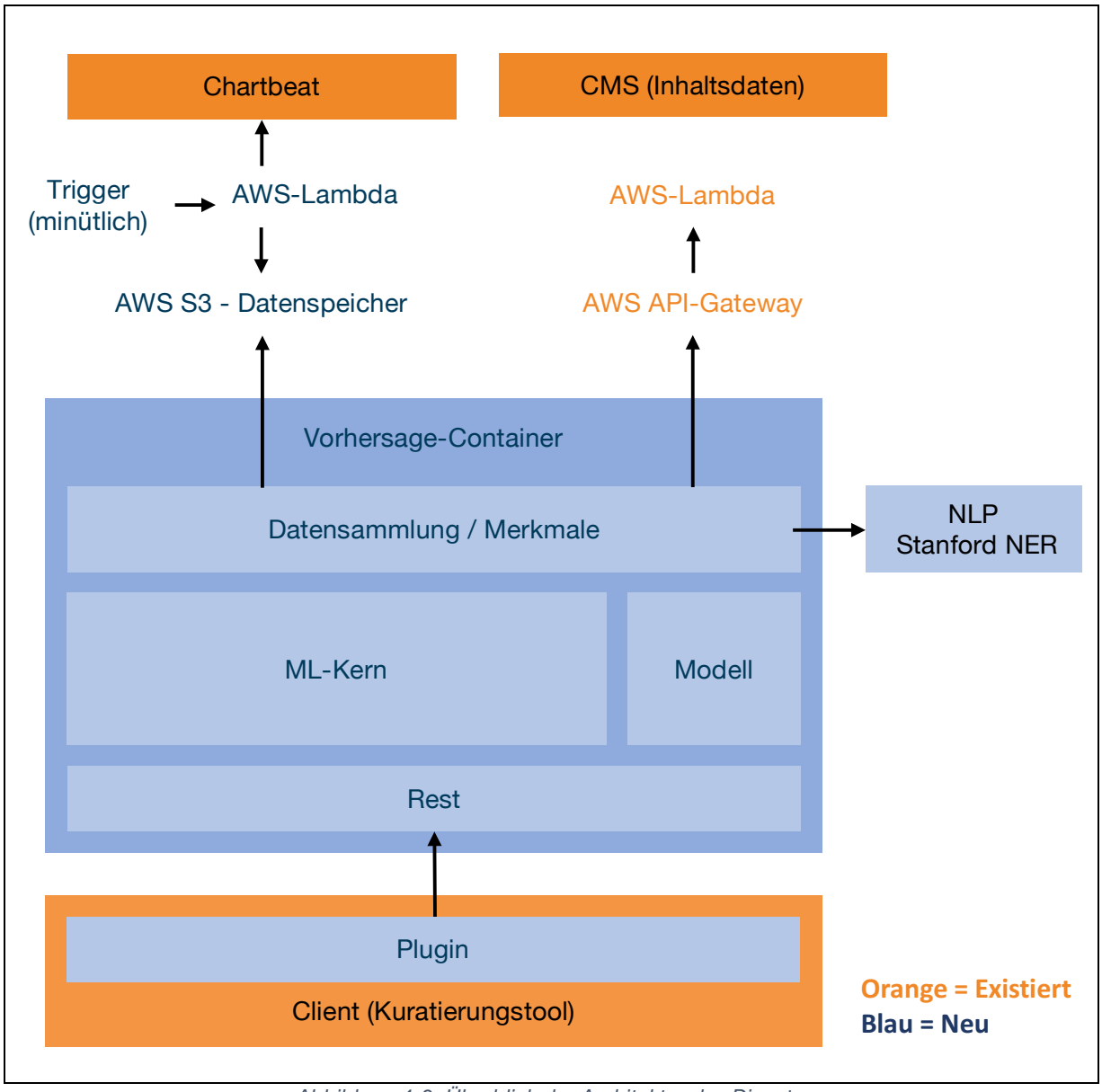

Abbildung 4-8: Überblick der Architektur des Dienstes

In einem ersten Entwurf sollte der Hauptteils des Dienstes als AWS Lambda umgesetzt werden, um Komplexität zu sparen. Jedoch stellte sich die Umsetzung später als schwierig heraus. Viele Python Bibliotheken enthalten nativen C-Code, der somit für Amazons Images kompiliert werden müssen. Außerdem wurden bei der Verwendung der nötigen Bibliotheken das Paket so groß, dass es nur mit Tricks knapp unter der Größengrenze für Lambdas blieb. Da die Architektur weniger sauber und schlecht erweiterbar war, wurde sie durch diese überarbeitet.

4.3.1 Container mit modularem Aufbau

Bei der Architektur wurde sich für die Verwendung von einem Container basierten Dienst entschieden. Durch diese Architektur können Lerner mit bereits gelernten Modellen abgelegt und austauschbar verwendet werden. Ein weiterer Vorteil ist, dass bei größerem Rechenaufwand des Modells, der Lernprozess auch aus dem Kern auf ein Cluster ausgelagert werden könnte und nur das resultierende Modell in dem Container verbleibt. Dabei wird durch den Einsatz bei „der WELT“ und der breiten Unterstützung und Verbreitung, Docker als Container-Lösung eingesetzt. Durch Betrieb von zwei oder mehr Instanzen des Vorhersage-Containers hinter einem Lastenverteiler (engl. Load-Balancer) kann somit auch die Verfügbarkeit sichergestellt werden, falls diese in Abwägungen mit den zusätzlichen Kosten gebraucht wird.
Dabei wird innerhalb des Vorhersage-Containers jeweils noch in verschiedene Module mit unterschiedlichen Zuständigkeiten unterschieden (siehe Abbildung 4-9). Die Sammlung, Verarbeitung und Merkmalsbildung erfolgt in der Sammlungs-Komponente. Die Umsetzung der Algorithmen erfolgt in der ML-Kern-Komponente, ein Modell für den späteren Dienst erzeugt. Der Zugriff über auf das Modell wird durch die Rest-Komponente gewährleistet.

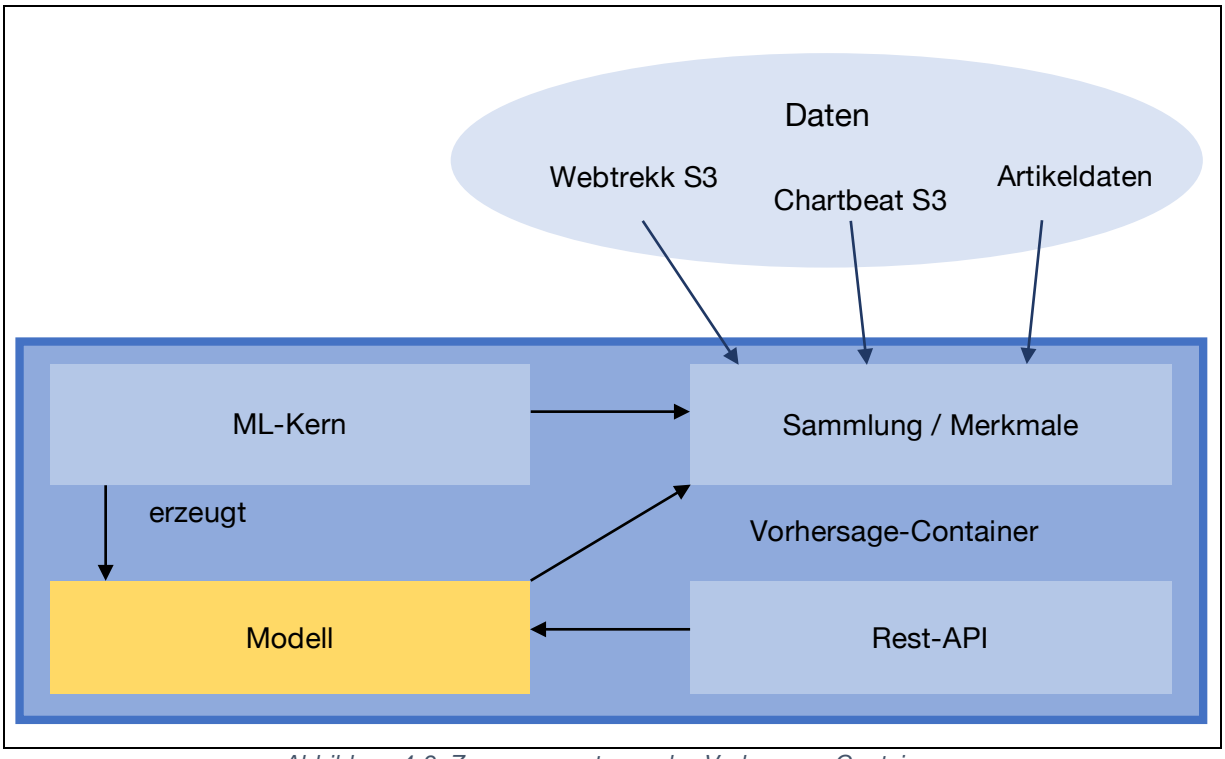

Abbildung 4-9: Zusammensetzung des Vorhersage-Containers

4.3.2 Schnittstellendefinition

Eine Definition der Schnittstellen ist wichtig, um die Dienste testen und einzeln entwickeln zu können. Ebenso ist eine klare Schnittstelle die Voraussetzung für die spätere Nutzung durch andere Dienste. In Tabelle 4-2 wurde die REST-Schnittstelle zwischen dem Vorhersage-Dienst und dem Plugins für das Kuratierungstool definiert. Die gelichzeitige Abfrage von Artikeln wurde aufgrund der ansonsten vielen Anfragen für alle Artikel auf der Startseite genutzt. Um zu verhindern, dass einzelne nicht vorhersagbare Werte zu Fehlern führen, wurde die Rückgabe fehlertolerant gestaltet, sodass Erfolgreiche und Fehler gemeinsam zurückgegeben werden können. Für den NER-Dienst wurde ebenfalls eine Definition in Tabelle 4-3 erstellt. Die Daten auf Amazons S3 werden im CSV-Dateien mit einem Zeitstempel im Namen abgelegt. Die anderen Schnittstellen einschließlich der Nutzung von S3 sind bereits durch die jeweiligen Dienste festgelegt und müssen nur verwendet werden.

URL	**/predict**
Methode	**POST**
Anfrage	**Beispiel:** **Content-Type:** `application/json` **Content:** `{ "articleIds": ["123456"] }`
Erfolgs Antwort	**Beispiel:** **Code:** `200 OK` **Content-Type:** `application/json` **Content:** `{ "results": [ {"articleId": "123456", "24H": 4000, "24H_Home": 1000, "30D": 5000, "30D_Home": 1300 } ] }`
Fehler Antwort	**Beispiel:** **Code:** `200 OK` **Content:** `{ "results": [ {"articleId": "123456", "error": "<Fehlernachricht>" } ] }`

Tabelle 4-2: Schnittstellendefinition der Vorhersage (REST)

URL	**/ner**
Methode	**POST**
Anfrage	**Beispiel:** **Content-Type:** `application/json` **Content:** `{ "text": "Peter arbeitet bei Siemens in Berlin" }`
Erfolgs Antwort	**Beispiel:** **Code:** 200 OK **Content-Type:** `application/json` **Content:** `{ "persons": ["Peter"], "organisations": ["Siemens"], "locations": ["Berlin"] }`
Fehler Antwort	**Beispiel:** **Code:** 500 Interner Fehler **Content:** `{ "error": "<Fehlernachricht>" }`

Tabelle 4-3: Schnittstellendefinition NER-Dienst (REST)

4.3.3 Kuratierungstool-Plugin

Da im bestehenden Kuratierungstool bisher kein direktes Plugin-Konzept angedacht ist, muss die Integration möglichst gut gekapselt sein, um ein leichten Ein/Ausbau zu gewährleisten. Dies wird durch die Erstellung einer eigenen AngularJS 2-Komponente erreicht. Dies Impliziert die Verwendung von TypeScript, wie dies auch im Kuratierungstool der Fall ist.

Das wichtigste Ziel des Plugins ist jedoch die gute Nutzbarkeit der gewonnenen Informationen. Dabei gibt es viele Möglichkeiten die vorhergesagten Artikelreichweiten für den Redakteur aufzubereiten. Es wurden mehrere Ideen entwickelt, welche u.a. Zahlen, Popups, Einfärbungen, Pfeile und Striche bei Artikeln beinhalteten. Eine erste Erkenntnis war, dass die meisten Darstellungen zu viele Informationen präsentierten und dadurch eher verwirrend wirkten. Weiter wurde festgestellt, dass die Redakteure meist unter Zeitdruck arbeiten und somit eine Behinderung durch z.B. Popups nicht in Frage kommt. Ebenso wäre eine gezielte Auswahl von Vorschlägen anhand der Vorhersage möglich, würde jedoch dem Redakteur bei absichtlichen Platzierungen von Artikeln oft nicht weiterhelfen. Weshalb sich für eine Lösung mit den ganzen vereinfacht dargestellten Informationen entschieden wurde, welche wenig Interpretation durch den Redakteur bedarf.

Dabei entstand die Idee durch Striche verschiedene Gruppen von Vorhersagen zu bilden und so die Reichweitenvorhersage vereinfacht darzustellen. Jedoch bedürfen Striche einer Erklärung durch eine Legende und sind nicht Intuitiv. Bei der Suche nach einer Alternative wurde das Batteriesymbol als beste Lösung ausgemacht. Dabei stellt die Batterie eine Metapher für das Potential eines Artikels dar. Ebenso ist das Batteriesymbol jedem Redakteur durch das Handy vertraut. Die Nutzung von Metaphern ist ein Konzept bei der Erstellung von Nutzerschnittstellen und erlaubt es Nutzern intuitiv mit neuen Objekten umzugehen.
Das Batteriesymbol wird dabei in 5 Stufen unterteilt, von leer (die Vorhersage ist unter 10.000) bis voll (die Reichweitenvorhersage ist über 100.000), um eine Unterscheidung möglich zu machen, jedoch nicht zu viel Informationen anzuzeigen. Die Grenzen wurden durch die Verteilung des Testdatensatz auf 10.000, 20.000, 50.000 und 100.000 festgelegt. Um dennoch die Daten anzubieten, können beim Überfahren der Symbole durch einen Tooltip die auf Tausend gerundeten Werte abgelesen werden.

4.4 Zusammenfassung

In der Konzeption wurden die Merkmale zur Bestimmung der Reichweite erarbeitet und in Metadaten, Inhaltliche, Zeitliche und Social-Trend Merkmale unterschieden. So wurden für die Inhaltlichen Merkmale z.B. die Ähnlichkeit von Artikeln, Neuheit von Artikeln und die Lesbarkeit durch neue Metriken messbar gemacht. Im maschinellen Lernen wurde auf verschiedene entwickelte und benutzte Konzepte zur besseren Lernbarkeit der Daten eingegangen. Ebenso wurde kurz auf das Modell mit Log-Transformation eingegangen und die Optimierung der Hyperparameter durch Rastersuche. In Abbildung 4-8 wurde eine Übersicht über die Architektur und aller Komponenten erstellt. Dabei wurde sich für ein Modular aufgebautes Containermodell entschieden, welches trainierte Modelle abspeichern kann. Beim Start des Dienstes wird über eine REST-Komponente der Zugriff auf das geladene Vorhersagemodell ermöglicht. Durch die Wahl von Docker können die gebauten Container auf beliebiger Infrastruktur verteilen und ausgetauscht werden. Die Integration in das Kuratierungstool erfolgt mittels einer AngularJS-Komponente. Für die Nutzerschnittstelle wurde die Einbindung eines Batteriesymbols mit 5 Stufen für jeden Artikel aus mehreren Ideen ausgewählt. Dabei stellt das Batteriesymbol eine Metapher für das Potential eines Artikels und damit die Reichweitenvorhersage dar.

5 Umsetzung

In diesem Kapitel werden wichtige und interessante Aspekte der Umsetzung aufgezeigt. Dabei werden zu Beginn der Projektaufbau und die verwendete Umgebung und verwendeten Werkzeuge erläutert. Im Anschluss werden die Implementierung der Datensammlung und die Extraktion der im Konzept erarbeiten Merkmale für die Reichweite beleuchtet. Um aus den Merkmalen ein Modell lernen zu können wird die Kern-Komponente und die darin enthaltene Umsetzung bzw. Nutzung der Algorithmen des maschinellen Lernens genauer erklärt. Danach wird jeweils kurz auf die REST-Komponente, die Integration in das Kuratierungstool und auf die Bereitstellung des Dienstes eingegangen.

5.1 Projektaufbau

Das Projekt wurde als Git-Repository angelegt, um das Projekt automatisch nach Änderungen durch Jenkins bauen zu können und eine Versionskontrolle zu erhalten. Dies ist eine Grundlage für Kontinuierlichen Integration.

In Tabelle 5-1 wird Beispielhaft auf den Projektaufbau des Vorhersagedienstes eingegangen. Die Dienste für die Chartbeatspeicherung auf S3, NER sind ähnlich im Aufbau, jedoch kleiner im Umfang.

Dateien/Ordner	**Zugehörigkeit**	**Beschreibung**
.gitignore	Git	Verhindert ungewollte Versionierung von Dateien
chartbeat/	Daten	Ordner zur Speicherung der Chartbeatdaten
Dockerfile	Docker	Definiert den Container in dem der Dienst läuft
environment.env	Umgebungs-variablen	Enthält Parameter wie Passwörter/Schlüssel und wird nicht in Git gespeichert; Wird beim Bauen durch Jenkins wieder eingefügt
features.py	Quelltext	Merkmals-Komponente
model.pkl	Modell	Exportiertes Modell (durch Joblib)
model_features.pkl	Modell	Exportierte Liste an Merkmalen (durch Joblib)
model.py	Quelltext	ML-Kern-Komponente

Tabelle 5-1: Projektaufbau des Vorhersagedienstes (Teil 1)

Dateien/Ordner	Zugehörigkeit	Beschreibung
requirements.txt	Pip	Definiert alle Python-Abhängigkeiten, zur Einrichtung der Umgebung (insbesondere für den Docker-Container)
service.py	Quelltext	REST-Komponente
service_test.py	Quelltext	Testet den Dienst (Integrationstest)
similarity.model	Gensim	Enthält das gespeicherte Ähnlichkeitsmodell
tf/	Terraform	Enthält mehrere Dateien für Terraform zur Dienstbereitstellung; Definiert Speicher, Cpu, Skalierung, Sicherheitsregeln etc.; Wird durch Jenkins ausgeführt und automatisiert Anlegen/Veränderung der Infrastruktur

Tabelle 5-2: Projektaufbau des Vorhersagedienstes (Teil 2)

5.2 Umgebung und Werkzeuge

Im Folgenden werden kurz die verwendete Umgebung und Werkzeuge vorgestellt. welche zur Analyse und Entwicklung eingesetzt wurden:

Jupyter Notebooks

Die Open-Source Software Jupyter stellt für verschiedene Programmiersprachen fernsteuerbare Laufzeitumgebungen genannt „Kernel“ bereit. Dabei startet Jupyter Notebooks einen Webserver, welcher eine komplette integrierte Entwicklungsumgebung (IDE) anbietet. Beim Benutzen von Dokumenten in der IDE wird automatisch für jedes ein Kernel gestartet und ermöglicht so die entfernte Ausführung von z.B. Python durch den Browser. Dabei werden bis zum Beenden eines Kernels alle Daten im Speicher gehalten. In einem Dokument können beliebig viele Zellen erstellt werden, welche jeweils z.B. Quelltext oder Markdown enthalten können. Bei der Ausführung einer Zelle wird dabei das Ergebnis direkt im Dokument aufbereitet und dargestellt. Dies kann dabei beliebig oft wiederholt werden und so schnell und interaktiv mit Daten experimentiert werden. Die Ergebnisse bleiben dabei im Dokument gespeichert und können auch ohne erneute Ausführung betrachtet werden.

NumPy

NumPy ist in Python die Basisbibliothek für wissenschaftliche Berechnungen. Sie stellt performante N-Dimensionale Arrays und viele Operationen bereit z.B. für lineare Algebra oder Fourier Transformationen.

Pandas

Pandas ist eine Open-Source Bibliothek, welche performante, einfach nutzbare Daten Strukturen und Daten Analyse Werkzeug für Python liefert. [36] Durch Pandas können große Datenmengen schnell und einfach manipuliert und analysiert werden. So sind SQL ähnliche Gruppierungen und Joins möglich, ebenso wie das Erzeugen von Pivot-Tabellen (siehe 2.3.1.3) oder Resampling (siehe 2.3.1.2). Dabei ist Pandas zu NumPy und Scikit-Learn kompatibel.

Joblib

Joblib ist eine Erweiterung von dem in Python integrierten „pickle“, welches eine Sterilisierung von Objekten erlaubt. Joblib schließt auch Funktionen ein. Dabei können verschiedenste im Speicher liegende Typen in einer Datei abgelegt werden. Durch das Laden dieser Datei kann der Zustand wiederhergestellt werden. Hierdurch müssen Berechnungen nicht mehrfach durchgeführt werden. Joblib komprimiert dabei die Objekte in binärer Form. Bei der Verwendung muss jedoch besonders auf die Abhängigkeiten geachtet werden. So müssen alle benutzten Importe in den gleichen Versionen vorliegen, wie beim Speichern, da es sonst zu Fehlern kommen kann.

Flask

Die Flask-Bibliothek ist Open-Source und erlaubt die einfache Erstellung von REST-Diensten in Python. Dazu wird mit Flask ein REST-Server gestartet und per Annotation im Quelltext REST-Schnittstellen für Funktionen festgelegt. Diese Funktionen können dann über den REST-Server direkt aufgerufen werden. Dabei können durch spezielle Variablen auf Parameter zugegriffen werden. Ebenso können die Antworten und deren Header gesetzt werden.

5.3 Datensammlung und Merkmalserstellung

Die Umsetzung der Datensammlung erfolgt in den bei der Architektur festgelegten einzelnen Teilen. Als erstes wurde der Chartbeat Dienst umgesetzt, um eine größtmögliche Datenbasis für diese Arbeit zu sammeln. Hierbei werden jeweils die erarbeiten Konzepte für die Verarbeitung der Daten und Merkmalsextraktion eingesetzt.

5.3.1 Chartbeat Dienst

Da Chartbeats eigene API für die Historie nur Stundengenaue Daten liefern kann, wurde zu Beginn dieser Arbeit ein Dienst erstellt, welcher die Live-Daten von Chartbeat jede Minute abruft und ablegt.

Dazu wurde mit JavaScript in Amazon Web Services (AWS) ein Lambda erstellt, welches die Live-Daten über die API abruft. Dieses wird durch einen Trigger automatisch jede Minute aufgerufen. Die Daten werden von diesem aus einer JSON-Struktur in eine tabellarische CSV überführt und auf AWS S3 ablegt. Die Überführung von JSON in CSV erfolgt um den Overhead so gering wie möglich zu halten. Dabei wurden die Anzahl an Informationen über Referrer auf die Top 5 reduziert, da ein unbegrenztes Array sich in CSV nicht abbilden lässt. Im Monat Mai wurden so 45472 Dateien mit insgesamt 1.818.880 Zeilen und 102 Spalten auf S3 abgelegt.

5.3.2 Merkmals-Komponente

Um die Merkmale extrahieren zu können wurde zuerst eine Sammlung der Roh-Daten implementiert. Dabei werden von Amazon S3 die Chartbeat-Dateien der letzten 48 Stunden durch die Amazon-Python-Bibliothek in den Ordner „chartbeat“ synchronisiert. Dabei werden 48 Stunden verwendet um einerseits den Speicher und Berechnungsbedarf gering zu halten und andererseits auch Vorhersagen für Artikel vom Vortag machen zu können. Durch eine Umgebungsvariable kann dabei im Lernmodus ein Zeitraum für die Synchronisierung eingestellt werden, um die entsprechende Datenbasis zu erhalten. Die Dateien werden ansonsten jede Minute aktualisiert und in ein DataFrame (Tabellen bei Pandas) geladen. Dabei wird immer nur das neuste geladen und alle älteren Daten wieder aus dem Speicher entfernt. Bei einer Container-Betrachtung ist dabei aufgefallen, dass der Speicherverbrauch um mehr als 300Mb schwankt. Die Ursache konnte in der spät laufenden „Garbage Collection“ gefunden und mit gc.collect() nach jeder Aktualisierung gelöst werden. Um die Daten von Artikeln nicht bei jeder Vorhersage neu holen zu müssen, wie dies in einer ersten Implementierung der Fall war, wurde ebenfalls ein DataFrame mit

Artikeldaten genutzt, welches in Abhängigkeit von der Chartbeattabelle für neue die Daten vom CMS holt und nicht vorkommende Artikel wieder entfernt. Nur im Lernmodus werden auch die vorverarbeiteten Webtrekkdaten von S3 geladen.
In den folgenden Unterkapiteln wird auf die Verarbeitung von Chartbeat, Webtrekk und Artikeldaten eingegangen. Daraus wurden dann mithilfe der erarbeiten Merkmalskonzepte und Formeln die Merkmale in einem DataFrame generiert. Als Beispiel mit besonderer Komplexität wird die Ähnlichkeitsbeziehung und dessen Umsetzung mit Doc2Vec ausgewählt und erläutert.

5.3.2.1 Chartbeat-Daten

Bei der Implementierung wurde auf Pandas gesetzt, welches eine größtenteils C-optimierte Verarbeitung der Daten ermöglicht. Es wurde außerdem darauf geachtet möglichst viele parallel optimierbare Umformungen zu nutzen und wenige Funktionen dabei einzusetzen, da die für das Lernen notwendige Datenmengen eine effiziente Verarbeitung voraussetzen.
Als erster Schritt wird den Chartbeatdaten mit dem Veröffentlichungszeitpunkt aus den Artikeldaten verknüpft um daraus dann die Differenz zur Veröffentlichung berechnen zu können.

```
df = pd.merge(df, dfArticles[['id', 'Publication']], how='left',
      left_on='path', right_on='id')
df = df.dropna(how='any', subset=['Publication'])
df['deltaPub'] = df.apply(lambda x: x.time - x.Publication, axis=1)
```

Dabei werden alle Chartbeatdaten entfernt, für die keine Artikeldaten vorhanden waren und somit kein Veröffentlichungsdatum bestimmt werden kann. Dies betrifft zum Beispiel später gelöschte Artikel.
Mit dieser relativen Zeit kann die Anzahl im Speicher liegenden Daten auf für die Vorhersage wichtigen Teil von 30 Minuten reduziert werden.
Um das Resampling wie im Konzept definiert umzusetzen und mit unterschiedlichen Funktionen und Zeiten testen zu können, wurde folgendes Objekt definiert:

```
resampleFunctions = {
   'default': lambda df: df,
   '15T_mean': lambda df: df.resample('15T').mean(),
   '5T_max': lambda df: df.resample('5T').max(),
   '5T_min': lambda df: df.resample('5T').min(),
   '5T_mean': lambda df: df.resample('5T').mean(),
   '5T_cusum': lambda df: df.resample('5T').mean().cumsum(),
   '15T_cusum': lambda df: df.resample('15T').mean().cumsum()
}
```

Wovon bei Tests alle, jedoch im Dienst nur noch einzeln ausgewählte angewendet werden. Aus den zeilenweisen Daten wurde durch eine verlustfreie Pivot-Tabellen Umformung (siehe 2.3.1.3), die Basis der Zeitlichen Merkmale.

Die Umsetzung der relativen Werte erfolgt durch einen Eingriff ganz zu Beginn der Datenverarbeitung. Dazu wird jeweils beim einlesen der Chartbeat-Dateien die Startseite herausgefiltert und dann alle Werte durch die jeweiligen Werte der Startseite geteilt. Dabei mussten bestimmte Spalten wie die Top Referrer ausgenommen werden.

Bei der Analyse der Top Referrer aller Chartbeatdaten wurden insgesamt 20.121 Domains festgestellt. Dabei wird in Abbildung 5-1 gezeigt, dass die Besucher sehr ungleich Verteilt sind. Jedoch können auch kleinere Seiten ein Indiz sein, sodass die Referrer aller Domains mit einer Summe von über 50.000 extrahiert wurden und daraus eigene Merkmale erstellt wurden:

`google`	`facebook`	`achgut.com`
`amp-welt-de.cdn.ampproject.org`		`amp.welt.de`
`android-app`	`danisch.de`	`flipboard.com`
`hartgeld.com`	`msn.com`	`news64.net`
`newstral.com`	`pi-news.net`	`politikversagen.net`
`redir.xing.com`	`redirect.viglink.com`	`secure.mypass.de`
`t.co`		

Wobei alle Domains mit Google oder Facebook zusammengefasst wurden. (Anmerkung: „t.co“ ist die Twitter URL)

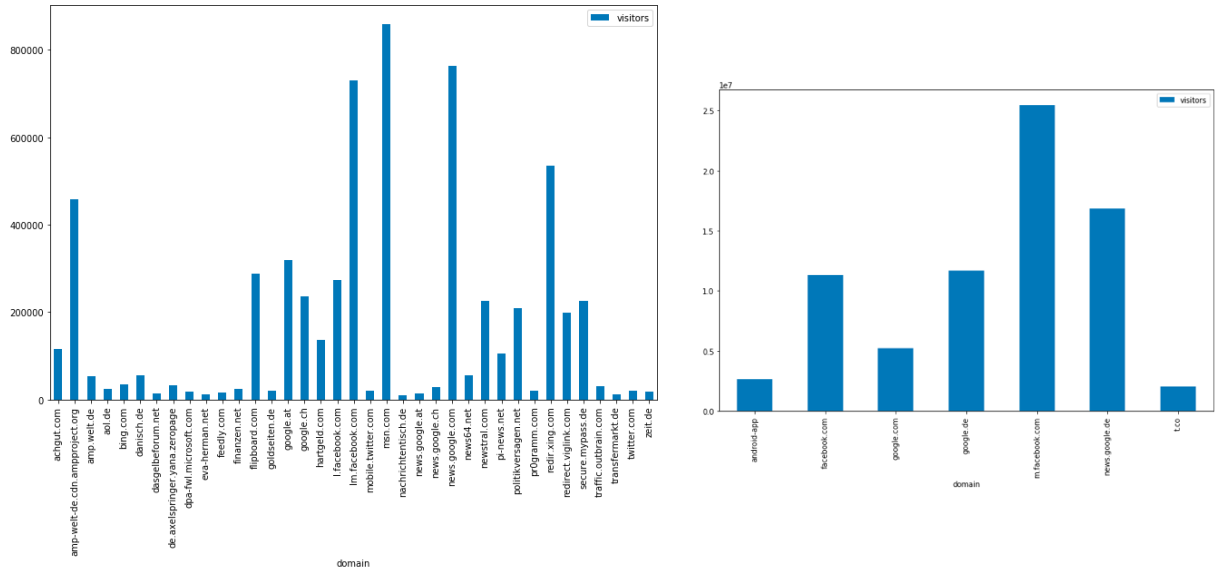

Abbildung 5-1: Chartbeat Summe gleichzeitiger Referrer per Domain (links: 10.000 - 1.000.000; rechts: >1.000.000)

5.3.2.2 Webtrekk-Daten

Die Extraktion aus der Daten aus Webtrekk hat durch die benötigten Kenntnisse über die komplexe Erstellung von Reports, die langsame Oberfläche und die Limitierung der Zeilenanzahl einige Probleme bereitet. Dabei wurde erst mit einem Zugang zu der neuen Oberfläche (Wechsel von Q3 zu Analytics) ein exploratives Testen möglich. Jedoch konnte die verwendete API wiederum nur Reports aus Q3 ausführen, sodass nach dem Konfigurieren und Finden richtiger Einstellungen, der Report für Q3 nachgebaut wurde. Der Report wurde so erstellt, dass er die folgenden Spalten für einen definierten Zeitraum enthielt:

Stunde, Seite, Veröffentlichungszeit, Reichweite, ReichweiteDurchStartseite.

Durch die im Konzept beschriebene Aggregation wurden daraus die folgenden Spalten für den Zielvektor berechnet:

Seite, Veröffentlichungszeit, Reichweite24h, Reichweite24hDurchStartseite, in24h, Reichweite30d, Reichweite30dDurchStartseite, in30d.

Durch die Invariante, jeweils für 24 Stunden nur maximal 24 Reichweiten aufzusummieren ($in24h \leq 24$), wurde bei der Umsetzung festgestellt, dass es auch Reichweiten-Werte für $\Delta t < 0$ gibt.

Nachforschungen bei der Redaktion ergaben, dass die Publikationsdaten teilweise bewusst neu gesetzt werden, wenn z.B. Inhalte weiterverwendet oder neu Veröffentlich werden. Der Anteil ist mit etwa 5% der Daten recht gering. Um dieses Problem zu umgehen wurde der Veröffentlichungszeitpunkt aus dem CMS, durch das in Webtrekk verwendete Publikationsdatum ersetzt, welches beim Zeitpunkt des ersten Aufrufes gesetzt und nicht mehr verändert wird.

5.3.2.3 Artikeldaten

Die abgerufenen Artikeldaten sind die Grundlage für die Metadaten und Inhaltlichen Merkmale. Bei der Vorverarbeitung wurde z.B. die Anzahl der Bilder und Videos genommen oder aus dem Bodytext Html und XML Elemente entfernt.

Durch eine Anfrage an den NER-Dienst wurde für jeden Bodytext die Anzahl der Personen, Organisationen und Orte als Merkmale hinzugefügt.

Mit der Python-Bibliothek textstat [37] wurden die Silben bestimmt und daraus die im Konzept beschriebene Lesbarkeit für deutsche Texte.

Die One-Hot-Kodierung von Sektionen, Wochentag und Stunden erfolgte über die Klasse DictVectorizer, welche wie folgt eingesetzt wurde:

```
vec = DictVectorizer()
sections = vec.fit_transform([{ 'rootSection': s[0],
                                'homeSection': s[1],
                                'hour': str(s[2].hour),
                                'day': s[2].strftime("%A")}
                      for s in df[['rootSection', 'homeSection',
                      'Publication']].values]).toarray()
for i, section in enumerate(vec.get_feature_names()):
    data[section] = sections[:,i]
    pass
```

Dagegen ist für die Cosinus-modellierten Zeitmerkmale folgender Quellcode entstanden:

```
features = 12
spread = 3
data[["zeit_{}".format(i) for i in range(features)]] =
  data.apply(lambda row: pd.Series(
      [np.cos((row.pubTimeH - (24 * i/features)) / spread * np.pi/2)
            if np.abs(row.pubTimeH - (24* i/features)) < spread else 0
                  for i in range(features)]
  ),axis=1)
```

Wobei erst 12 Merkmale, also jeweils 2 Stunden, mit 3 als Spreizung geprüft wurde. Bei Tests stellten sich jedoch bei verschiedenen Parametern kaum Änderungen der Vorhersagegüte ein, sodass dieser Ansatz nicht weiterverfolgt wurde.

5.3.2.4 Artikelähnlichkeitsmodell

Bei der Umsetzung des Ähnlichkeitsmodells mit Doc2Vec wurde wie in der Analyse festgelegt die gensim-Implementierung genutzt. Dabei wurde darauf geachtet, dass die Python-Umgebung Zugriff auf einen C-Compiler hat, da die Optimierung im Vergleich zu nicht kompiliertem Python etwa um den Faktor 70 schneller ist. [38]

Die Texte der 66.404 Artikel mit Reichweite wurden als erstes normalisiert. Dies beinhaltete die Umwandlung in Kleinbuchstaben und das Einfügen von Leerstellen vor und nach jedem Satzzeichen. In ersten Versuchen wurde zusätzlich mit dem Weglassen von Stoppwörtern, also häufiger Wörter ohne inhaltlicher Relevanz experimentiert, jedoch keine Verbesserung festgestellt.

Der erste Versuch erfolgte durch:

```
model = Doc2Vec(size=100, window=3, min_count=2, iter=10)
model.build_vocab(trainDocs)
model.train(trainDocs, total_examples=model.corpus_count,epochs=model.iter)
```

Dieser Durchlauf dauerte 9 Minuten 47 Sekunden, jedoch stieg die Ausführungsdauert mit Anpassung der Vektorgröße, dem Kontext und der Iterationen auf über eine

Stunde an. Wobei die Ergebnisse wie im Konzept beschrieben in entsprechende Merkmale gebracht wurden. Durch Multiple Lineare Regression mit 10Fach-Kreuzvalidierung und dessen R^2-Wert, wurde die Erklärbarkeit der Reichweitenvarianz durch die Ähnlichkeitswerte ermittelt. Anhand dieses Wertes wurden die in Tabelle 5-3 dargestellten Kombinationen von Parametern getestet. Dabei wurde die Entscheidung auf den Werten für die 24 Stunden und interne Reichweitenvorhersage getroffen, da diese die vielversprechendsten Ergebnisse lieferten.

Parameter				interne Reichweite R^2_{24}
Vektorgröße	Kontext	Min. Anzahl	Iterationen	
100	5	1	20	0,152 (+/- 0,128)
100	5	2	20	0,140 (+/- 0,132)
100	10	2	20	0,159 (+/- 0,108)
500	5	2	20	0,313 (+/- 0,122)

Tabelle 5-3: Doc2Vec Parameter mit R^2 für die Reichweitenvorhersage der Testmenge

Die Erhöhung des Kontextrahmens hatte dabei kaum einen Einfluss auf die Ergebnisse. Somit wurde dieser Parameter ansonsten auf den von gensim vorgegebenen Wert festgelegt. Das Modell mit der größten Erklärbarkeit für die Reichweite wurde anschließend ausgewählt und wie folgt gespeichert:

```
model.save('similarity.model')
```

jupyter Vorverarbeitung-meta_und_textdaten Last Checkpoint: 08/06/2017 (autosaved) Logout

File Edit View Insert Cell Kernel Help Not Trusted Python 3

Code Dashboard View:

In [411]:
```
articleSim['dif_0'] = articleSim['Imp24h'] - articleSim['Imp24h_0']
articleSim['err_0'] = np.abs(articleSim['Imp24h'] - articleSim['Imp24h_0'])
articleSimDescribe = articleSim.drop('articleId', axis=1).describe()
round(articleSimDescribe[list(articleSimDescribe)[-2:] + list(articleSimDescribe)[:-2]],2)
```

Out[411]:

	dif_0	err_0	Imp24h	similarity_0	Imp24h_0	similarity_1	Imp24h_1	similarity_2	Imp24h_2	similarity_3	...	similarity_5	Imp24h_5	sir
count	66404.00	66404.00	66404.00	66404.00	66404.00	66404.00	66404.00	66404.00	66404.00	66404.00	...	66404.00	66404.00	
mean	354.26	10078.14	6708.15	0.82	6353.89	0.79	6005.90	0.78	6074.66	0.78	...	0.77	5777.62	
std	35672.74	34221.33	27896.43	0.08	25717.77	0.07	24186.96	0.07	25563.50	0.07	...	0.07	24599.46	
min	-930190.00	0.00	0.00	0.26	0.00	0.24	0.00	0.24	0.00	0.24	...	0.22	0.00	
25%	-615.00	45.00	0.00	0.77	0.00	0.75	0.00	0.74	0.00	0.73	...	0.72	0.00	
50%	0.00	635.00	105.00	0.82	130.00	0.80	120.00	0.79	115.00	0.78	...	0.77	110.00	
75%	655.00	6190.00	2680.00	0.87	2480.00	0.84	2375.00	0.83	2295.00	0.82	...	0.82	2185.00	
max	2454075.00	2454075.00	2454510.00	1.00	1110950.00	0.99	930270.00	0.99	930270.00	0.99	...	0.99	1367235.00	

8 rows × 23 columns

Abbildung 5-2: Beispiel für die Ähnlichkeitsverteilung des Modells mit Differenz und absolutem Fehler des ganzen Korpus für die interne Reichweite (Bei Size=100,Min_count=2 und iter=10)

Wenn der Dienst nicht im Lernmodus gestartet wird, wird nur das Modell geladen und entsprechend die Ähnlichkeitsmerkmale berechnet. Um die Reichweite für alle Artikel nicht immer laden zu müssen, wurde der DokumentenId zusätzlich die Reichweite durch einen Punkt getrennt angehängt.

5.4 ML-Kern

Zur Umsetzung des maschinellen Lernens wurden verschiedenen Versionen der Kern-Komponente erstellt, welche jeweils durch einen Algorithmus und eine Menge von Merkmalen ein Modell erstellen. Dieses Modell kann entweder Lokal oder per Kontinuierlicher Integration erzeugt werden, in dem in den Umgebungsvariablen der Lernmodus aktiviert wird. Nur in diesem Fall wird die Kern-Komponente ausgeführt und ein Modell generiert. Dabei wird das ganze Modell inklusive einer Vorhersagemethode durch Joblib in einer Modelldatei gespeichert, welche beim Bauen des Docker-Containers eingefügt wird. Beim Starten des Docker-Container wird diese dann geladen und nur noch die Vorhersagemethode genutzt. Im Lernmodus wird mittels 10-Kreuzvalidierung angepasstes R^2 für alle Daten und R^2 für die Top 10% der Daten berechnet und ausgegeben. Dabei wurde in der Analyse immer auch ein Residuen-Graph erzeugt um die Verteilung der Abweichung von Vorhergesagten zu Gemessenen Daten beurteilen zu können.

5.4.1 Merkmalsauswahl

Da unterschiedliche Merkmale verglichen werden sollen, wurde der Kern so angepasst, dass darin beliebige einzelne oder ganze Bereiche von Merkmalen definiert werden können. Da dies jedoch zu vielen Versionen führte, welche sich nur in der Merkmalsmenge unterscheiden, wurde die Auswahl dieser in die Umgebungsvariablen ausgelagert. Im Lernmodus werden dann jeweils ein Modell und zusätzlich verwendete Merkmalsmenge abgelegt.

5.4.2 Umsetzungen

Da die Algorithmen durch Scikit-Learn implementiert wurden, musste nur auf die entsprechenden Klassen zurückgegriffen werden.

Algorithmus	Scikit-Learn Klasse	Parameter
Multiple Lineare Regression	LinearRegression	
Ridge Regression	Ridge	alpha = λ_1
Lasso Regression	Lasso	alpha = λ_2
Elastisches Netz	ElasticNet	alpha = $\lambda_1 + \lambda_2$, l1_ratio = $\lambda_1 / (\lambda_1 + \lambda_2)$

Tabelle 5-4: Verwendete Scikit-Learn Regressionsklassen

Bei Elastischen Netzen wird durch Scikit-Learn nur die Möglichkeit geboten ein Verhältnis der Regulierer anzugeben. Für Auswertung wurde somit darauf geachtet,

dass die beiden Regulierungsparameter λ_1 und λ_2 entsprechend gesetzt und daraus alpha und l1_ratio berechnet werden. Dies war auch wichtig um bei der Rastersuche die Parameter in unterschiedlichen Verteilungen testen zu können.
Für die Rastersuche wurde GridSearchCV gewählt, wobei diese Klasse gleichzeitig Kreuzvalidierung implementiert.
Für Lasso ergab sich somit Beispielsweise folgender Code:

```
alpha = np.logspace(-6, 1, 200)
lasso = linear_model.Lasso()
estimator = GridSearchCV(lasso, dict(alpha=alpha), scoring=adjr2, cv=10)
```

5.5 NER-Dienst

Der Dienst zur Named Entity Bestimmung wurde durch Stanford NER einzeln mittels Docker-Container umgesetzt. Dafür wurde ein Docker-Container mit Java-Umgebung, als Grundlage genutzt. Darauf aufbauend ist ein einfacher REST-Server entstanden, welcher das Aufrufen von Stanford NER mit dem offiziellen bereits vorgelernten Modell für Deutsch über eine „exec“-Ausführung übernimmt. Das Ergebnis des Aufrufs wird als JSON zurückgegeben. Die Schnittstelle wurde wie im Konzept (siehe 4.3.2) erarbeitet umgesetzt, wobei bei der Verarbeitung noch auf Unicode geachtet werden musste, da das genutztes Modell mit dieser Kodierung trainiert wurde. Der Dienst ist durch eine Sicherheitsregel in AWS nur aus dem Firmennetz erreichbar.

5.6 Rest-Komponente

Um die REST-Komponente so einfach wie möglich zu halten wurde für die Implementierung Flask genutzt. Dabei wurde der REST-Server mittels des folgenden Codes auf Port 8080 geöffnet:

```
app = Flask(__name__)
app.run(host='0.0.0.0', port=8080)
```

Damit die REST-Schnittstelle durch einen Browser mit dem Kuratierungstool aufgerufen werden kann, musste zusätzlich ein „Cross-Origin“ Header gesetzt werden. Dies wird als „Cross-Origin Ressource Sharing“ (CORS) bezeichnet und konnte durch ein Einbinden von „flask_cors“ gelöst werden.
Die Vorhersagefunktion wurde danach mit der folgenden Annotation versehen:

```
@app.route('/predict', methods=['POST'])
```

Wie im Konzept entwickelt, wird die Antwort als JSON mit der Vorhersage für jeden Artikel gegeben, welche durch die Rückgabe der Funktion definiert wird.

Die Health-Schnittstelle wurde wie folgt integriert:

```
@app.route('/health', methods=['GET'])
def health():
    ok = dataHealth()
    resp = jsonify({'ok': ok})
    resp.status_code = 200 if ok else 503
    return resp
```

Wobei dataHealth() ein Boolean für die Vollständigkeit und Aktualität der gesammelten Daten zurückgibt. Ist dies Wahr wird als Status 200-Ok zurückgegeben, ansonsten 503-Service nicht verfügbar. Die Antwort erfolgt als JSON mit dem Attribut „ok“ und dem gleichen Wahrheitswert.

5.7 Integration in das Kuratierungstool

Die Implementierung wurde ebenso wie das Kuratierungstool mit AngularJS 2 in TypeScript erstellt, um eine direkte Integration zu ermöglichen. Im ersten Schritt wurde ein AngularJS Service geschrieben, welcher für Artikelnummern die Vorhersagen von dem bereitgestellten Dienst abholt. Im Anschluss wurde in das Modell des Artikels um ein Attribut für die Reichweitenvorhersage erweitert und im Aktualisierungs-Service ein Aufruf des zuvor erstellten Service integriert. Dieser sorgt dafür, dass das Abholen aller bisher unbekannten Vorhersagen für die angezeigten Artikel erfolgt.

Als letztes wurde die Artikel-Komponente, welche für die Anzeige einer Artikelzeile verantwortlich ist, um eine Vorhersagen-Komponente erweitert. Für diese wurde dann mittels Stylesheets das in Abbildung 5-3 zu sehende Batteriesymbol umgesetzt. Durch ´das Eventsystem wird in der Vorhersage-Komponente bei einer Aktualisierung der Artikelvorhersage das Icon in der entsprechenden Stufe angezeigt.

Abbildung 5-3: Erstellte Batteriesymbole für das Reichweitenpotential

5.8 Bereitstellung

Die Bereitstellung des Dienstes erfolgt einerseits in Docker Containern über Amazon Elastic Container Service (Amazon ECS) und andererseits als Erweiterung im Kuratierungstool selbst. Um den Docker Container zu bauen wird Jenkins benutzt. Vor dem bauen werden in Jenkins die Integrationstests ausgeführt um die Funktionsfähigkeit zu gewährleisten. Danach wird durch einen Job das Dockerfile mit

den gespeicherten Modell- und Merkmalsnamen gebaut und in die Docker Registry von Amazon ECS hochgeladen. Ein Interessantes Detail ist die Reichenfolge in der die Komponenten hinzugefügt werden, so wird erst ganz zum Schluss das Modell hinzugefügt, um ein effizientes Neubauen bei geänderten Daten zu ermöglichen. Ein getrennter Job macht das gleiche für den NER-Dienst. Die Dienste können so automatisch gebaut und ohne Ausfallzeit ausgetauscht werden. Die Zugriffe auf die Dienste sind durch erstellte Sicherheitsregeln in Amazon auf das Firmennetzwerk beschränkt. Ein weiterer Jenkins Job aktualisiert das Lambda zur Chartbeat-Datenspeicherung auf S3 mittels Terraform, einem Werkzeug zum automatischen Aufbau von AWS Komponenten und Infrastruktur.

5.9 Zusammenfassung

In der Umsetzung wurde der Projektaufbau beispielhaft am Vorhersagedienst gezeigt. Es wurde auf benutzte Umgebung und Werkzeuge eingegangen, die zur Analyse der Daten und Entwicklung des Dienstes eingesetzt wurden. Es wurde kurz auf den Dienst zur Chartbeatspeicherung eingegangen. Um darauf aufbauend die Merkmalskomponente und Verarbeitung der Chartbeat bzw. Webtrekkdaten genauer zu zeigen. Dabei wurden z.B. die Top Referrer bestimmt und in Merkmale umgewandelt. Es konnten alle erarbeiten Merkmalskonzepte umgesetzt werden. Als Beispiel für ein Merkmal wurde die Umsetzung der Ähnlichkeitsbestimmung ausgewählt, welches durch Doc2Vec und dessen Optimierung eine besondere Komplexität aufwies. Jedoch konnten keine direkten Einflüsse auf die Reichweite gefunden werden. Die Umsetzung der linearen Modelle war durch die Nutzung von Scikit-Learn problemlos möglich. Im Vergleich dazu war ein sehr hoher Aufwand mit dem Lösen vieler kleiner Probleme und dem Optimieren bei der Gewinnung der Merkmale verbunden. Dazu beigetragen haben vor allem die großen Datenmengen, welche in der Analyse nicht nur in 30 Minuten, sondern vollständig verarbeitet werden mussten, sodass viele der Optimierungen unumgänglich waren. Bei der Integration in das Kuratierungstool wurden der Aktualisierungsdienst und die Anzeigekomponente mit dem verwendeten Batteriesymbol erläutert. Als letztes wurde auf die Kontinuierliche Integration durch das Bauen des Dockercontainers eingegangen. Welche im Anschluss automatisch in Amazon ECS in Betrieb genommen werden. Die Infrastruktur kann dabei durch Terraform automatisch aufgebaut und verändert werden.

6 Auswertung

In der Auswertung wird zuerst die Testumsetzung behandelt, um die Methodik und das Vorgehen während der Arbeit zu bewerten. Um die Reihenfolge der Arbeit einzuhalten wird danach auf die Merkmale und Einflüsse auf die Reichweite eingegangen. Da die Ähnlichkeit von Artikeln ebenso ein wichtiger Bestandteil ist, wird sie getrennt betrachtet. Dabei wird bei beiden der Sieger der Ergebnisse der einzelnen Kerne bzw. Algorithmen genutzt, welcher im darauffolgenden Unterkapitel ausgewertet wird. Um die Algorithmen zu bewerten wird dabei kurz einzeln auf jeden Algorithmus eingegangen und am Ende ein Vergleich aller Ergebnisse vorgenommen. Nach der Bestimmung des geeignetsten Kerns wurde mit diesem eine Validierung des Dienstes durchgeführt. Abschließend wird die Umsetzung des Dienstes, anhand der in 3.3.1 definierten Anforderungen bewertet.

6.1 Testumsetzung

Um während der Entwicklung und Optimierung der Implementierung direktes Feedback zu erhalten wird direkt nach jedem Lernen Fehler-Werte und Graphen anhand gleicher Methodik bestimmt. Dabei wurde 10fach Kreuzvalidierung (siehe 2.3.3.3) benutzt um eine zufällig schlechte Verteilung in Trainings- und Testmenge zu vermeiden. Ein Ergebnis der Reichweitenanalyse war, dass Inhalte mit großer Reichweite vergleichsweise wichtig für die Kuratierung sind. Um dies einzubeziehen, wurden immer auch die Top 10% der Inhalte mit der größten Reichweite getestet um dessen Güte getrennt betrachten zu können. Bei den Ergebnissen kam es häufig zu negativen Werten, diese wurden in Tabellen nur mit einem „-“ gekennzeichnet, da sie keine weitere Aussagekraft haben.

Als Fehlermaß zur Einschätzung der Güte wurden das angepasste R^2 für alle Daten und R^2 für die Top 10% benutzt. Für die oberen 10% ist das Verhältnis von Anzahl der Daten zu Merkmalen verändert, sodass der Wert automatisch geringer wird. Dies geschieht verstärkt bei niedrigen Werten (siehe Abbildung 2-3), wodurch der Sinn der Anpassung verloren geht und deshalb bei den Top 10% auf R^2 zurückgegriffen wurde. Die Maße liefern dabei Aufschluss über den Anteil an erklärter Variation durch die Merkmale. Beide Maße liefern vergleichbare Werte und wurden von vergleichbaren wissenschaftlichen Arbeiten genutzt. Dabei wurden anfangs mit einem Monat

gesammelter Daten gearbeitet und am Ende mit drei Monaten. In der folgenden Tabelle wird die aggregierte Datenanzahl dargestellt:

Zeitraum	Anzahl Artikel von Chartbeat[1]	Anzahl Artikel von Webtrekk[1]	Anzahl nutzbarer Artikel[2]	Top 10%
01.05.17-01.06.17	2.144	10.641	1.783	178
01.05.17-01.08.17	6.521	32.784	5.434	543

Tabelle 6-1: Anzahl gesammelter[1] und genutzter[2] Daten

Die Anzahl genutzter Artikel ist etwas kleiner als die Anzahl von Chartbeat gesammelter Daten, da ein kleiner Anteil erst nach 30 Minuten in Chartbeat auftaucht. Die Tabelle 6-2 zeigt eine Übersicht über die Einteilung, der für die zur Auswertung verwendeten Daten. Dafür wurden sie in die bei der Konzeption unterschiedenen Merkmalsbereiche aufgeteilt.

Bereich	Anzahl Merkmale t = Anzahl Zeiten nach dem Resampling
Metadaten Merkmale	160
Inhaltliche Merkmale	42
Zeitliche Merkmale	2 + 102·t (Relative Werte)
Sozial-Trend Merkmale	2 + 18·t

Tabelle 6-2: Übersicht über die verwendeten Merkmale (01.05.2017-01.08.2017)

6.2 Merkmale und Reichweite

Durch den Einsatz von linearen Modellen lassen sich Rückschlüsse auf die Merkmale und ihren Einfluss auf die Reichweite ziehen. Um die Struktur der Arbeit beizubehalten, wird an dieser Stelle den Ergebnissen vorgegriffen, welche aus dem Gütevergleich (siehe 0) hervorgehen. So erfolgt der Gütevergleich der Merkmalsbereiche in Tabelle 6-3 anhand der Multiplen Linearen Regression für 24 Stunden und mittels interner Reichweite, welche die besten Gesamtergebnisse erzielte. Dabei werden aus den Bereichen folgende Schlüsse abgeleitet:

- Die Metadaten haben allein keine Aussagekraft, da sie keine direkte lineare Abhängigkeit besitzen. Sie sind jedoch zusammen mit den Zeitlichen Merkmalen für einen signifikanten Anstieg der Erklärbarkeit verantwortlich.
- Die Inhaltlichen Merkmale geben keinen Aufschluss auf die Top 10%. Dadurch wird angenommen, dass zwar eine gewisse Abhängigkeit bei der Masse (kleinen)

Reichweiten besteht, jedoch große Reichweiten nicht linear Fortgesetzt werden können und so durch die Streuung nicht mehr Erklärt werden können.

- Die zeitlichen Merkmale bilden auf Grundlage der ersten 30 Minuten die größte Basis für die Vorhersage, können jedoch nur teilweise die Abweichungen der Top 10% erklären.
- Die Soziale Verbreitung kann allein nur teilweise die Reichweite erklären, ist aber einer der wichtigsten Faktoren für die Top 10%. Dies lässt sich durch den Anstieg bei zeitlichen und Social-Trend Merkmalen belegen und wird auf die Bedeutung von viralen Themen zurückgeführt.

Merkmalsbereich	Adj. R^2	Top 10% R^2
Metadaten Merkmale	-	-
Inhaltliche Merkmale	31,3	-
Zeitliche Merkmale	63,1	20,4
Zeitliche + Metadaten Merkmale	73,2	23,9
Social-Trend Merkmale	25,4	-
Zeitliche + Sozial-Trend Merkmale	68,1	35,4
Alle Merkmale	79,7	40,3

Tabelle 6-3: Gütevergleich der Merkmalsbereiche (interne Reichweite 24h)

Um die Koeffizienten auszuwerten wurde dagegen auf die zweitbesten Ergebnisse von Ridge Regression zurückgegriffen, da hierbei die Koeffizienten durch die Regulierung besser ablesbar waren. Die Einflüsse wurden jeweils in verschiedenen Konstellationen getestet und sind aufgrund der Interferenzen dabei unterschiedlich Verteilt. Um dies zu berücksichtigen wurden die Koeffizienten entsprechend ihrer Größe und Abweichung dargestellt. Daraus wurden die jeweils 10 betragsmäßig größten Koeffizienten und dadurch Einfluss auf die Reichweite in den folgenden beiden Tabellen festgehalten.

Merkmal	Beschreibung	Bereich
Internal (30 Min.)	Nutzer von Internen Quellen	Zeitliche M.
isPremium	Premium Status	Metadaten
Platform_a (30. Min.)	Mobile Plattform Nutzer	Zeitliche M.
hour=1	Artikel veröffentlicht um 1 Uhr	Metadaten
hour=0	Artikel veröffentlicht um 0 Uhr	Metadaten
ref_google (30. Min.)	Google Nutzer	Social-Trend
rootSection=kmpkt	Kompakt Kanal „der WELT“	Metadaten
visits_hist (30. Min.)	Gleichzeitige Nutzer in 2 Min.	Zeitliche M.
rootSection=Geld	Geld Kanal „der WELT“	Metadaten
Read (30. Min.)	Anzahl lesender Nutzer	Zeitliche M.

Tabelle 6-4: Die 10 größten positiven Koeffizienten

Die 10 größten negativen Einflüsse:

Merkmal	Beschreibung	Bereich
rootSection=Print	Print Kanal „der WELT“	Metadaten
rootSection=Newsticker	Newsticker Kanal „der WELT“	Metadaten
Platform_g (30. Min.)	Tablet Plattform Nutzer	Zeitliche M.
Idle (30. Min.)	Nicht aktive gleichzeitige Nutzer	Zeitliche M.
hour=11	Artikel veröffentlicht um 11 Uhr	Metadaten
hour=9	Artikel veröffentlicht um 9 Uhr	Metadaten
hour=12	Artikel veröffentlicht um 12 Uhr	Metadaten
platform_f (25. Min.)	Desktop Plattform Nutzer	Zeitliche M.
ref_t.com (30. Min.)	Twitter	Zeitliche M.
domload_median (30. Min.)	Median der Ladezeit	Zeitliche M.

Tabelle 6-5: Die 10 negativsten Koeffizienten

Bei den Einflüssen lassen sich einige Interessante Effekte ablesen. Durch die positiven Koeffizienten lassen sich bei den Merkmalen durch den Durchschnitt unterschätze Artikel ablesen z.B. bei Veröffentlichung in der Nacht (0 und 1 Uhr). Dies lässt dadurch erklären, dass Artikel, welche in der Nacht veröffentlich werden, meist wichtig sind und deswegen noch publiziert werden. Die ersten 30 Minuten haben jedoch geringe Nutzerzahlen, sodass er in Verhältnis durch seine Wichtigkeit am nächsten Tag mehr

Reichweite bekommt. Bei den Premiumartikeln wird der hohe Koeffizient dagegen durch die im Vergleich zur Reichweite niedrigen Chartbeatwerten erklärt. Der Einfluss von Google und der Mobilen Plattform war dagegen erwartet worden.
Bei den negativen Einflüssen werden dagegen überschätzte Artikel bestraft. Dies ist in den Mittagsstunden der Fall oder bei Print bzw. Newsticker Artikeln. Ebenso ein wichtiger Faktor ist die Ladezeit der Seite.
Auffällig sind bei den Ergebnissen, dass fast nur Zeitliche und Metadaten Merkmale vorkommen. Jedoch sind die Inhaltlichen Merkmale (insbesondere die Neuheitsmerkmale) für weitere 10% Verbesserung im Vergleich zum Rest verantwortlich.
Die verschiedenen Resampling-Funktionen wurden ebenfalls einzeln mit Multipler Linearer Regression verglichen. Dabei kamen jedoch nur die zeitlichen Merkmale zum Einsatz, um mögliche Verfälschungen zu vermeiden. Die Ergebnisse in Tabelle 6-6 zeigen, dass 5 Minuten Resampling mit Verwendung des Durchschnitts am besten abgeschnitten hat. Wobei die Fehler im 10fach Kreuzvalidierung alle unter +/- 1% lagen.

Resampling	Minimen	Maximum	Mean	Cusum
5 Minuten	60,2	57,6	63,1	28,5
15 Minuten	56,3	54,2	59,6	36,3
Keins (nur 30. Min.)	38,2	38,2	38,2	-

Tabelle 6-6: Vergleich der Resamplingfunktionen (angepasstes. R^2)

Die relativen Werte brachten dagegen keine Verbesserung für die Vorhersagegenauigkeit. Das Ergebnis von 63,1% wurde bei Aktivierung auf 36,1% zurückgeworfen, sodass die relativen Werte nicht weiter beachtet wurden. Durch ein paralleles Einbauen wären jedoch eventuell noch Verbesserungen möglich.

6.3 Artikelähnlichkeit

Während der Umsetzung wurde die Ähnlichkeitsbeziehung anhand der Verbesserung des Vorhersagefehlers für die Reichweite gemessen und optimiert, da eine entsprechende Messung oder ein Vergleichskorpus nicht existierte. Für das Training wurden insgesamt 64.404 der 75.669 Artikel mit Reichweitenwerte genutzt, da die restlichen knapp 10% keinen Body-Text hatten. Die Reichweiten wurden aus Zeitgründen nur für die 24 Stunden mit interner Reichweite verglichen, nachdem die

geringe Aussagekraft erkannt wurde. Die Merkmale sind somit für die anderen Reichweiten im Zielvektor Fehlerbehaftet.

Für die Auswertung der Ähnlichkeitsbeziehung wurden einerseits Stichprobenartig 15 zufällige neue Artikel ausgewählt und jeweils die Titel mit den der Top 10 ähnlichsten Artikel verglichen. Die 15 ausgewählten Artikel blieben über unterschiedliche Parameter hinweg die Gleichen. Die Ähnlichkeit wurde jeweils mit 0-3 bewertet, wobei 0 keine Gemeinsamkeiten und 3 eine Übereinstimmung in mehreren Punkten bedeutet. Die Ergebnisse sind in der nachfolgen Tabelle dargestellt und weisen auf eine gute Ähnlichkeitsbeziehung hin, sind jedoch für den Datensatz nicht repräsentativ.

Parameter			Ähnlichkeit Top 10 (Skala 0-3)				
Vektorgröße	Min. Anzahl	Iterationen	Mittelwert	#0	#1	#2	#3
100	1	20	0,993	63	41	30	16
100	2	20	1	61	44	29	16
500	2	20	1,433	33	46	44	27

Tabelle 6-7: Manuelle Top 10 Ähnlichkeits-Auswertung von Doc2Vec anhand des Titels (an 15 Beispielen)

Dabei wurde darauf geachtet, die gleiche Vorhersage beim gleichen neuen Artikel auch den gleichen Ähnlichkeitswert zuzuordnen. Die niedrigen Werte für die Ähnlichkeit der Titel sind jedoch relativ zufällig über die ganzen Top 10 verteilt und nicht nur im unteren Bereichen zu finden. Daraus wird angenommen, dass die Merkmale sortiert nach dem Wert einen großen Störfaktor haben. Die Tabelle ist jedoch auch ein Indiz dafür, dass einzelne Wörter kaum einen Einfluss auf die Ähnlichkeit haben.

Da die Artikel im Datensatz sehr unterschiedliche Bereiche abdecken, gibt es zu Artikeln nur wenige Artikel mit ähnlichen Inhalten. Dafür ist die Trefferrate mit Übereinstimmungen im Titel (>0) in den Top 10 Artikeln mit mehr als 2/3 ein gutes Ergebnis. Insgesamt wird somit das Ähnlichkeitsmodell anhand der Stichprobenauswertung als gut Bewertet.

Jedoch zeigt sich bei der Bestimmung der Reichweite durch die Reichweite der ähnlichsten Artikel ein anderes Bild (siehe Abbildung 6-1). Dabei wurde versucht ein lineares Modell anhand aller neuen Daten zu erzeugen. Der Fehler betrug dabei R^2 = 0,273 (+/- 0,098).

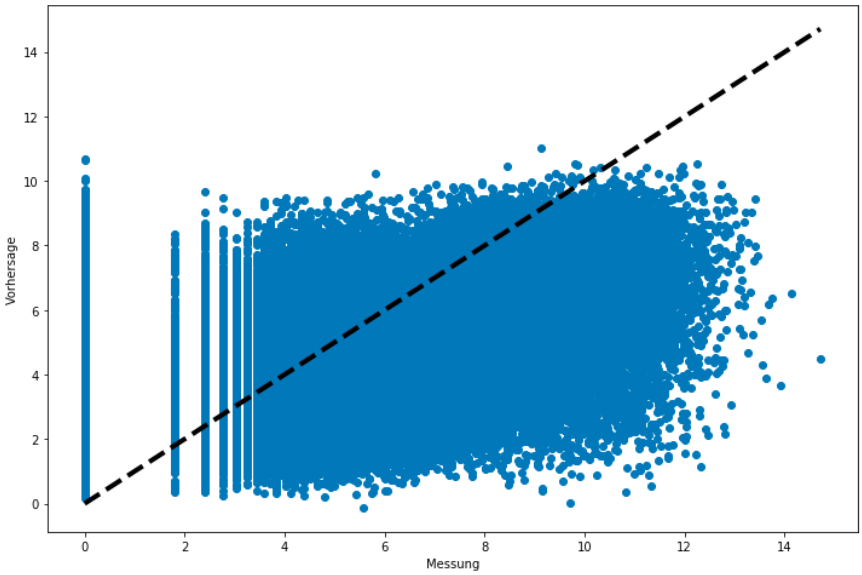

Abbildung 6-1: Reichweitenvorhersage durch die Reichweite ähnlicher Artikel

Die Verteilung der Koeffizienten der Reichweite von der höchsten Ähnlichkeit zur niedrigsten (0,105; 0,115; 0,103; 0,102; 0,091; 0,088; 0,084; 0,089; 0,079; 0,081) ist jedoch nahezu monoton absteigend und deutet darauf hin, dass weniger ähnliche Artikel auch weniger Einfluss auf die Reichweite haben.

6.4 ML-Kerne

Die Auswertung der einzelnen ML-Kerne erfolgte jeweils per Algorithmus mit den im Kapitel Testumgebung festgelegten Methoden und Daten über 3 Monate mit allen Merkmalen. Um die optimalen Güte-Werte vergleichen zu können, wurde zuerst eine Bestimmung der Hyperparameter für Ridge Regression, Lasso Regression und das Elastische Netz vorgenommen. Danach wurden die Ergebnisse aller Algorithmen verglichen und analysiert.

6.4.1 Hyperparameter

Die Optimierung der Hyperparameter durch den angepassten R^2-Wert wurde mittels Rastersuche durchgeführt. Dabei wurde die Werteverteilung zu Beginn logarithmisch gewählt, damit vor allem kleine Anpassungswerte in genauen Stufen betrachtet werden. In ersten Versuchen mit 200 Werten im Intervall $(10^{-6},1)$ wurden die Parameter immer zu den Maximalwerten 1 bei Ridge bzw. 0 (10^{-6}) bei Lasso gezogen und hatten dann gleiche Ergebnisse wie Multiple Lineare Regression. Dies ist zwar ein erwartetes Verhalten für R^2, da eine Optimierung der Koeffizienten zwar eine Generalisierung jedoch keine weitere Abweichungserklärung bringt. Jedoch sollte sich durch die verbesserte Auswahl von Koeffizienten der Wert für angepasstes R^2

verbessern, da das Verhältnis zu den Daten zu weniger Bestrafung führt. Bei einer Überprüfung der Berechnung von angepassten R^2 wurde dabei festgestellt, dass die Bestimmung der Koeffizientenanzahl k durch ein Gleitkomma-Problem immer alle Koeffizienten zählte. Dies wurde durch einen sehr geringen Schwellenwert wie folgt behoben:

```
def adjr2(estimator, X, y):
    r2 = metrics.r2_score(estimator.predict(X),y)
    k = np.sum((estimator.coef_)>0.00001)
    n = len(ground_truth)
    adj = 1 - ((1 - r2)*float(n-1)/(n-k-1))
    return adj
```

Nach der Korrektur ergaben sich die korrekt optimierten Hyperparameter, welche die erwarteten verbesserten angepassten R^2-Werte erreichten. Für Ridge Regression entstand nach der Korrektur ein Wert über 0,1. Da in diesem Bereich die logarithmische Verteilung nur noch vergleichsweise ungenau ist, wurde für Ridge diese verworfen und stattdessen eine mehrstufige Rastersuche genutzt. Diese wurde mit jeweils 100 Werten durchgeführt, wobei jeweils das neue Intervall um den alten Bestwert gelegt wurde. Die optimierten Hyperparameter sind in Tabelle 6-8 zusammengefasst, wobei die entstandenen Gütewerte im folgenden Unterkapitel aufgeführt sind.

Algorithmus	λ_1	λ_2
Ridge Regression	0.132	-
Lasso Regression	-	0.00455
Elastisches Netz	0.102	0.00634

Tabelle 6-8: Optimierte Parameter für Ridge Regression, Lasso Regression und das elastische Netz

6.4.2 Gütevergleich

Die Werte für die Güte der zu Vergleichenden Algorithmen wurden wie in der Testumsetzung (siehe 6.1) beschrieben durchgeführt. Dabei wurden die nicht relativen Daten, 5 Min Resampling mit Durchschnitt und die im vorigen Kapitel bestimmten Hyperparameter genutzt. Die Modelle wurden dabei mit den 5.434 Artikeln aus 3 Monaten trainiert.

Algorithmus	Adj. R^2_{24h}	Top 10% (R^2_{24h})	Adj. R^2_{30d}	Top 10% (R^2_{30d})
Multiple Lineare Regression	79,7	40,3	54,8	13,3
Ridge Regression	81,4	25,1	54,2	-
Lasso Regression	62,7	-	23,2	-
Elastisches Netz	76,5	12,2	48,6	-
Tree Regression	84,7	6,9	66,5	-

Tabelle 6-9: Gütevergleich der Algorithmen (interne Reichweite)

Algorithmus	Adj. R^2_{24h}	Top 10% (R^2_{24h})	Adj. R^2_{30d}	Top 10% (R^2_{30d})
Multiple Lineare Regression	56,5	-	32,4	-
Ridge Regression	57,3	-	34,7	-
Lasso Regression	43,2	-	-	-
Elastisches Netz	52,9	-	19,0	-
Tree Regression	74,3	-	43,1	-

Tabelle 6-10: Gütevergleich der Algorithmen (gesamet Reichweite)

Die Tabelle 6-9 zeigt dabei die Vorhersage der Internen Reichweite und Tabelle 6-10 die gesamte Reichweite, wobei ein Strich für negative Werte und damit eine schlechtere Abweichungserklärung als der jeweilige Mittelwert steht. Dabei steht Regression durch Entscheidungsbäume mit vorheriger Hauptkomponentenanalyse außer Konkurrenz und wurde nur zum Vergleich mit einem nicht linearen Modell eingefügt. Dabei konnte dieser Algorithmus zwar bessere Ergebnisse erreichen, jedoch im Vergleich zu den linearen Modellen keine Abstraktion auf die wenigen hohen Reichweiten fortsetzen. Dieser lineare Zusammenhang wurde durch einfache Multiple Lineare Regression am besten beibehalten. Im grafischen Vergleich mit Ridge zeigt sich dieser Zusammenhang in Abbildung 6-2 und Abbildung 6-3 recht deutlich, während Ridge bei geringeren Reichweiten ein dichteres Anliegen an der optimalen Diagonalen aufweist, wird die Streuung im oberen Reichweitenbereich wesentlich

höher, als bei der einfachen linearen Regression. Generell wurde diese Streuung bei den meisten Modellen in den Top 10% sehr groß, sodass diese oft keine Erklärung für die Reichweite bieten konnten.

Aus dem Vergleich der beiden Tabellen wird ebenso direkt ersichtlich, dass generell eine interne Reichweitenvorhersage wesentlich besser durch die erarbeiten Merkmale vorhergesagt werden konnte als die gesamte Reichweite. Dies lässt sich zum einen durch die geringere Aussagekraft von Metriken auf „der WELT“ für externe Aufrufe erklären und zum anderen dadurch, dass die Menge der Einflüsse zu einer hohen Verzerrung führt, welche durch die wenigen Daten nicht gelernt werden kann.

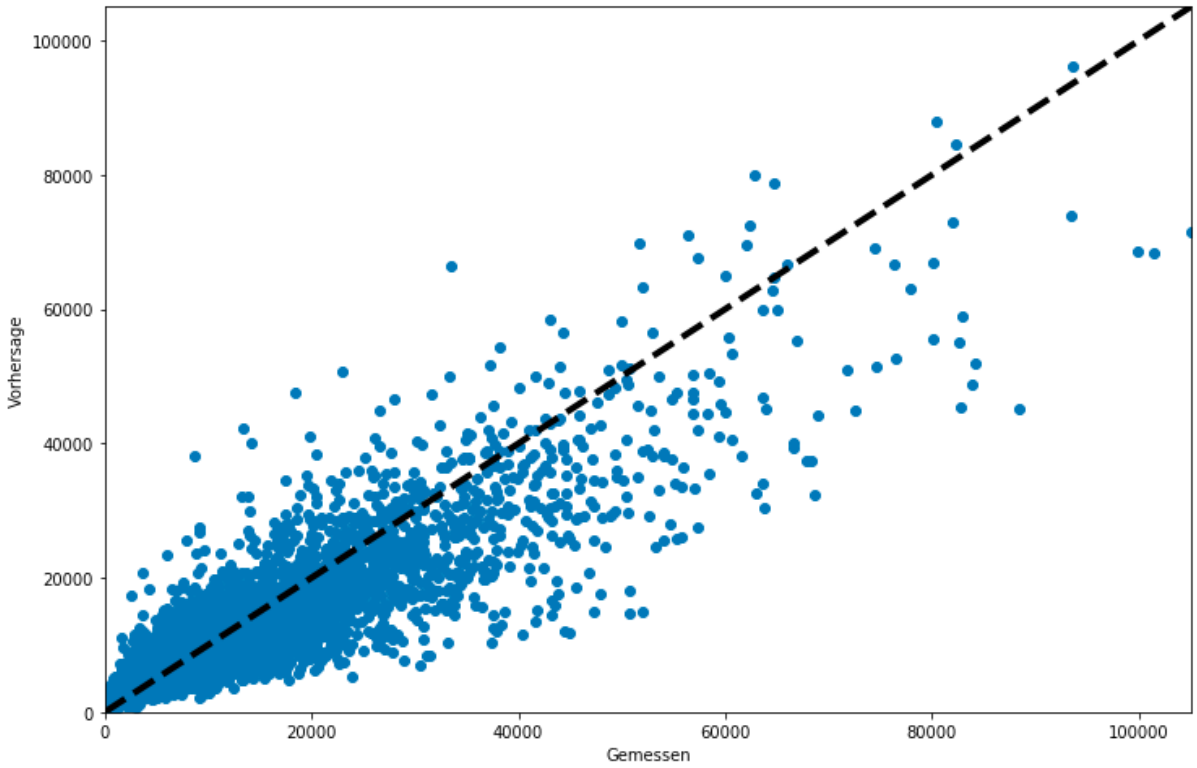

Abbildung 6-2: Ergebnis der Multiplen Linearen Regression

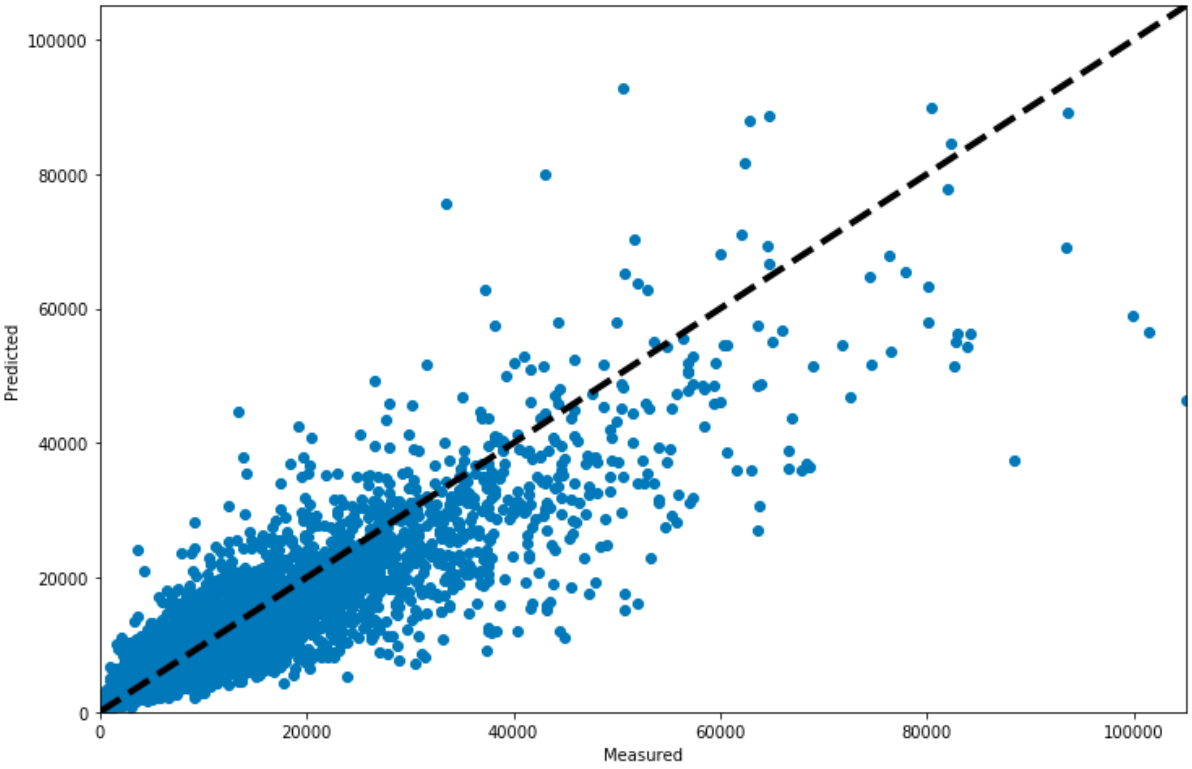

Abbildung 6-3: Ergebnis der Ridge Regression

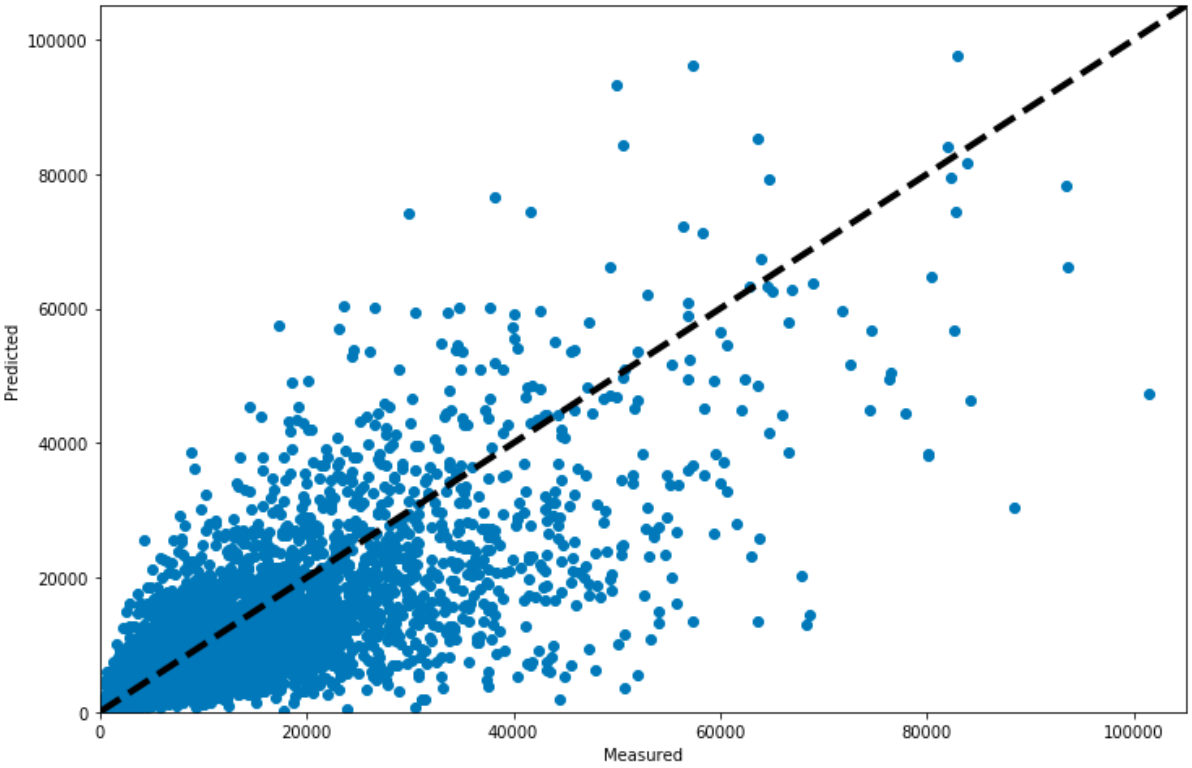

Abbildung 6-4: Ergebnis der Lasso Regression

6.5 Validierung

Um den Dienst und die Vorhersagegüte zu validieren, wurde dieser im realen Einsatz mit neuen Daten getestet. Nach der Auswahl von Multipler Linearer Regression und der Fertigstellung des Dienstes wurde dazu das Modell mit den gesamten Daten der 3 Monate trainiert. Bei der Validierung wurde der Zeitraum vom 10.10.2017 bis zum 10.11.2017 ausgewertet. Die Reichweitenanzahl aus Webtrekk für diesen Zeitraum betrug 9.262 wovon für 1.823 Artikel Daten von Chartbeat gespeichert wurden. Von diesen wurden wiederrum 1.803 Artikel von Anfragen der Kuratierung abgedeckt.
Die Tabelle 6-11 zeigt die Vorhersagegüte des Modells für die neuen Daten. Die Güte ist bei der internen Reichweite sehr nahe an vorherigen Testwerten, was die gute Abstraktion des Modells zeigt. Dabei konnten die Werte sogar leicht verbessert werden, was vermutlich auch auf die größere Trainingsbasis zurückzuführen ist. Die gesamte Reichweite konnte dagegen schlechter vorhergesagt werden und zeigt, dass das Modell externe Faktoren nicht ausreichend abbilden kann.

Daten	Anzahl	Adj. R^2_{intern}	Top 10% (R^2_{intern})	Adj. R^2_{gesamt}	Top 10% (R^2_{gesamt})
Alle Artikel	1823	80,6	41,8	52,5	-
Vorhergesagte Artikel	1803	80,4	41,2	51,4	-

Tabelle 6-11: Validierung des Modells für die Vorhersage der Reichweite nach 24 Stunden (10.10.17-10.11.17)

In der Validierungsphase lief der Dienst ohne Komplikationen mit einer Verfügbarkeit von 100%, bei minütlichem Monitoring. Durch den Dienst konnten für 84% aller Anfragen Reichweitenvorhersagen gegeben werden, dessen Veröffentlichung mindestens 30 Minuten her ist. Von den 16% nicht Vorhersagbaren Anfragen war der Grund für 98%, dass keine Chartbeat-Daten verfügbar waren. Die restlichen Fehler konnten auf ungültige Daten zurückgeführt werden, welche beim Abruf vom CMS (Artikel nicht mehr vorhanden z.B. wieder unveröffentlicht) oder Verarbeiten durch NER nicht möglich (kein Inhalt) verursacht wurden. Um für die ca. 16% der bisher nicht Vorhersagbaren eine Prognose erstellen zu können, müsste die Chartbeatsammlung ausgeweitet werden. Jedoch muss dann zwischen den großen entstehenden Datenmengen und dem Nutzen für die Kuratierung abgewogen werden. In dem gemessenen Zeitraum hat sich die gesamte interne Reichweite im Vergleich zu den vorigen Monaten wie folgt entwickelt.

Zeitraum	Juli	August	September	10.10. - 10.11.
Interne Reichweite in Mil.	53,4	54,4	58,6	56,7
Totale Reichweite in Mil.	181,1	178	196,7	185

Tabelle 6-12: Interne Reichweitenentwicklung (jeweils auf 30 Tage normiert)

Aufgrund der natürlichen Schwankung und des kurzen Zeitraums lässt sich aus den Daten statistisch keine Aussage über die Auswirkung des Dienstes auf die gesamte interne Reichweite ableiten. Zusätzlich wäre eine Analysierung der optionalen Nutzung erforderlich, um einen Zusammenhang herstellen zu können.

6.6 Dienstumsetzung

Um die Dienstumsetzung auszuwerten, werden die in der Analyse (siehe 3.3.1) definierten Funktionalen und Nichtfunktionalen Anforderungen in den Tabelle 6-13 und Tabelle 6-14 überprüft. Dabei konnten bis auf die vertikale Skalierbarkeit des Lernens alle Anforderungen erfüllt werden und ein produktiv einsetzbarer Dienst zur Reichweitenvorhersage entwickelt werden. In Abbildung 6-5 wird die fertige Oberfläche gezeigt, welche das für den Redakteur nutzbare Ergebnis dieser Arbeit darstellt.

Nr.	Funktionale Anforderung	Priorität	Erfüllt	Erläuterung
1	Vorhersage der Reichweite für Inhalte	Höchste	Ja	Die Varianz vom Mittel wurde zu 76% erklärt
2	Unterstützung verschiedener Algorithmen	Hoch	Ja	Es wurden mehrere nutzbare Kerne entwickelt
3	Automatische Datensammlung	Höchste	Ja	Umgesetzt durch Sammlungs-Komponente
4	REST Schnittstelle	Hoch	Ja	Umgesetzt in REST-Komponente (Def. 4.3.2)
5	„Health“ Schnittstelle	Mittel	Ja	Umgesetzt in der REST-Komponente (Def. 4.3.2)
6	NER Bereitstellung	Hoch	Ja	NER-Dienstes
7	Integration in das bestehende Kuratierungstool	Höchste	Ja	Umgesetzt (siehe Abbildung 6-5)

Tabelle 6-13: Umsetzungsstatus der funktionalen Anforderungen

Nr.	Nichtfunktionale Anforderung	Priorität	Erfüllt	Erläuterung
1	Benutzerfreundlichkeit	Hoch	Ja	Konzept erarbeitet und Umgesetzt (Einfach und Verständlich)
2	Vorhersagegüte	Höchste	Ja	Priorisierung erfolgt; Adj. R^2 von 74,4% in der Validierung
3	Skalierbarkeit	Mittel	Teilweise	Horizontal ja, vertikal der Dienst, jedoch nicht das Lernen mit Scikit-Learn
4	Wiederverwendbarkeit	Hoch	Ja	Wurde im Konzept und der Umsetzung berücksichtigt
5	Kontinuierliche Integration	Hoch	Ja	Umgesetzt (siehe 5.8)
6	Performanz	Hoch	Ja	Beachtet in der Umsetzung und linearen Modellen
7	Sicherheit	Mittel	Ja	Zugriffsrechte in AWS und Auslagerung der Passwörter
8	Verfügbarkeit	Hoch	Ja	Betrieb hinter Load-Balancer mit 2 Instanzen möglich

Tabelle 6-14: Umsetzungsstatus der nichtfunktionalen Anforderungen

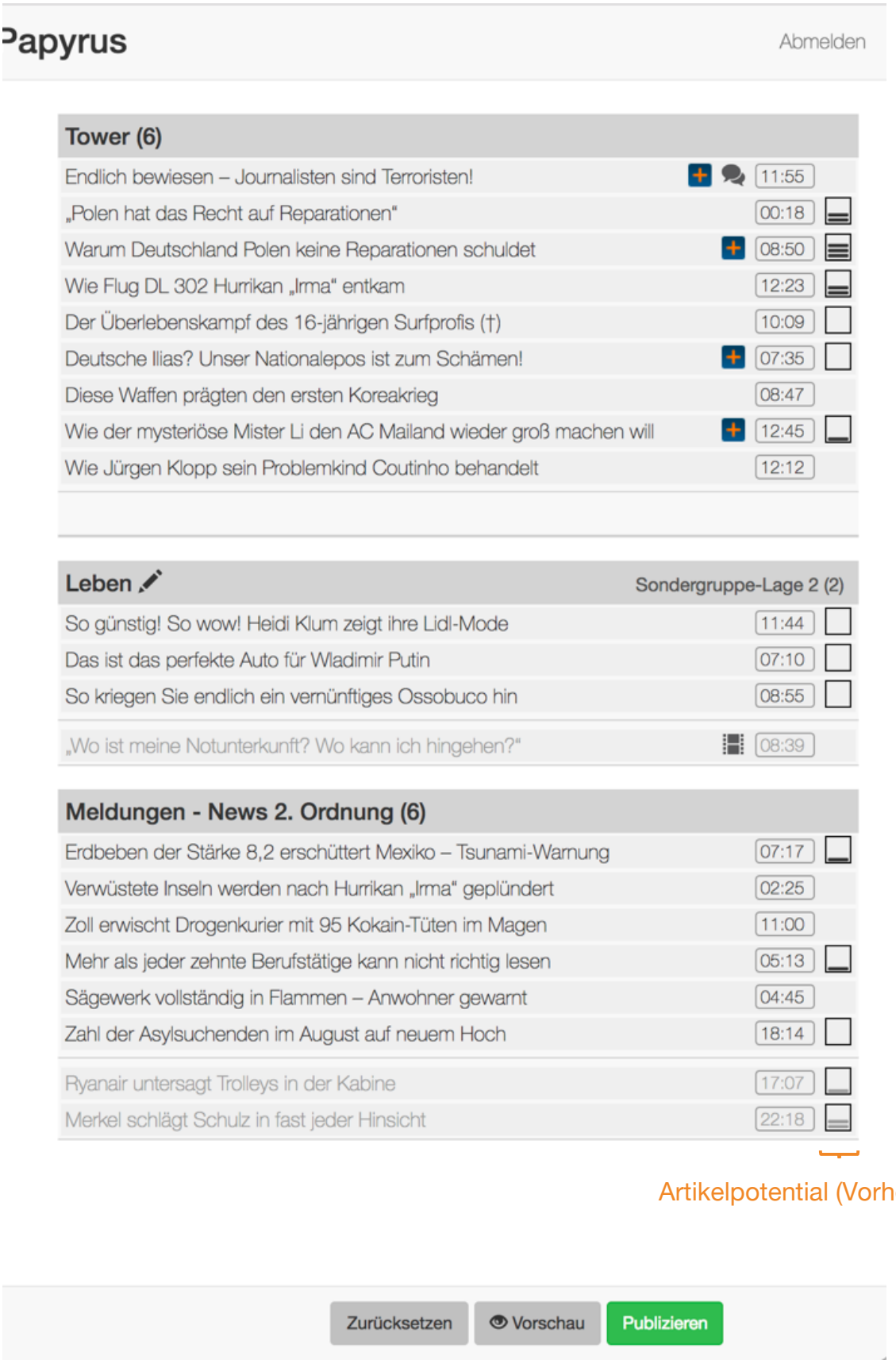

Abbildung 6-5: Screenshot des Kuratierungstools mit Vorhersagenintegration

6.7 Zusammenfassung

In der Auswertung der betrachteten Algorithmen konnte Multiple Lineare Regression als bestes Modell für die Reichweitenvorhersage evaluiert werden. Dafür wurden optimale Hyperparameter für alle Algorithmen gefunden und die Vorhersagegüte beurteilt. Das so bestimmte Modell wurde einer Validierung im realen Einsatz unterzogen und dabei gleichzeitig der Dienst im Einsatz getestet.

Im Kapitel wurde weiterhin auf die Einflüsse der Merkmale auf die Reichweite eingegangen. So stellen die Zeitlichen Merkmale die Basis für die Vorhersage dar. Durch die zusätzlichen Metadaten und Social-Trend Merkmale kann die Vorhersage jeweils signifikant verbessert werden. Vor allem die Social-Trend Merkmale geben Aufschluss über die Artikel mit besonders großen Reichweiten. Da die Inhaltlichen Merkmale und vor allem die Ähnlichkeitsbeziehung kaum einen Einfluss haben, wurden diese getrennt Betrachtet. Die Auswertung kommt zu dem Schluss, dass zwar die Ähnlichkeitsbeziehung durchaus funktioniert, aber die Abhängigkeit zur Reichweite in der erarbeiteten Form nicht gegeben ist. Die Dienstumsetzung wurde anhand der definierten Anforderungen überprüft und erfüllt diese bis auf einen Teilpunkt vollständig. Aus der Auswertung ergibt sich somit die erfolgreiche Umsetzung eines Dienstes zur Reichweitenvorhersage, welcher in das Kuratierungstool integriert ist und durch Redakteure zur Reichweitenoptimierenden Kuratierung genutzt werden kann.

7 Schlussbetrachtung

7.1 Fazit

In der vorliegenden Arbeit wurde die Problematik der Reichweitenvorhersage analysiert und eine Lösung entwickelt, die Redakteuren „der WELT“ eine reichweitenoptimierte Kuratierung ermöglicht. Dafür wurden einerseits verschiedene Algorithmen für lineare Regression erarbeitet, verwendet und deren Ergebnisse ausgewertet und verglichen. Zum anderen wurde NLP eingesetzt um auf Basis der Textinhalte Merkmale zu erstellen. Hierfür wurde Doc2Vec zur Bestimmung von ähnlichen Inhalten mit der Hypothese eingesetzt, dass die Reichweite von ähnlichen Inhalten mit der Reichweite korreliert. Dieser Zusammenhang konnte jedoch in der Auswertung nicht bestätigt werden. Ebenso wurde Stanford NER genutzt um Personen, Organisationen und Orte zu extrahieren. Dabei wurde in der Arbeit aus den zur Verfügung stehenden Daten ein möglichst breites Spektrum an Merkmalen extrahiert, welche die Einflüsse auf die Reichweite abbilden.

Durch die Ergebnisse wurde gezeigt, dass ein linearer Zusammenhang zwischen den unterschiedlichen Messgrößen der gleichzeitigen Nutzer der ersten 30 Minuten und der Reichweite nach 24 Stunden hergestellt werden kann. Eine weitere Verbesserung des Modells lässt sich durch Merkmale für Metadaten und Inhalte erreichen. Die wichtigen, reichweitenstarken Artikel ließen sich am besten durch die Merkmale für soziale Trends verbessern. Dies lässt sich durch die Viralität solcher Artikel erklären.

Die Vorhersagemodelle wurden für 24 Stunden, sowie für 30 Tage und jeweils interne und gesamte Reichweite ermittelt. Das Modell für interne Reichweite nach 24 Stunden war dabei am präzisesten. Dies ist für die Kuratierung von Vorteil, da es die Nutzer „der WELT“ abbildet.

Die Umsetzung eines Dienstes für die Nutzung durch Redakteure war ebenfalls ein wichtiger Bestandteil dieser Arbeit. Hierzu wurden Anforderungen definiert durch welche die Konzeption, Umsetzung und Auswertung erfolgt ist. Für den Dienst wurden drei große Frameworks für das maschinelle Lernen verglichen. Als Ergebnis wurde Scikit-Learn ausgewählt, welches sich durch die wenigen Probleme und Einfachheit der Umsetzung als gute Wahl herausstellte. Dabei ist als Resultat sowohl ein Docker basierter modularer Dienst mit austauschbarem Algorithmus und Modell entstanden, als auch eine benutzerfreundliche Integration in das Kuratierungstool „der WELT“.

7.2 Einsatz

Der entstandene Dienst wurde erfolgreich in die produktive Umgebung von „der WELT“ eingebunden. Im Validierungszeitraum ist der Dienst ohne Zwischenfall betrieben worden und zeigt, durch die Integration in das Kuratierungstool, dem Redakteur das Reichweitenpotential neuer Artikel an. Als nächster Schritt ist eine Nutzung der Vorhersage für eine automatische Kuratierung von Unterseiten aus dem Dienst angedacht. Durch die Nutzung ist es auch möglich nicht nur die Kuratierung zu steuern, sondern bspw. besondere Inhalte mit mehr Informationen, Bildern oder Videos anzureichern, noch bevor die meisten Nutzer den Inhalt sehen. Hierdurch lassen sich wiederum optimierte Nutzerzahlen durch eine Verbesserung erreichen um den qualitativen Eindruck vieler mit wenig Aufwand zu verbessern. Es wäre auch möglich Artikel mit großen Potential besonders zu vermarkten.

7.3 Probleme

Eines der größten Probleme waren die verfügbaren Daten. So wurde erst im Verlauf der Arbeit die Datenbasis mit Chartbeatdaten aufgebaut. Somit waren genauere Analysen zu den Merkmalen nicht zu Beginn, sondern erst zum Ende der Arbeit möglich. Hierdurch wurde ein Teil der Konzepte für Merkmale allein auf Hypothesen gestützt entwickelt und erst im Anschluss durch die Daten überprüft. In dieser Zeit wurde durch die fehlende Datenbasis auch verstärkt das Dienstkonzept entwickelt.

Ein weiteres Problem welches nicht weiter beleuchtet wurde, weil es in der jetzigen Form keine Daten hierzu gibt, ist der Einfluss der Vorhersage bzw. des Vorschlags auf das Modell selbst. Eine hohe Vorhersage führt bei besserer Platzierung so auch zu höherer Reichweite. Bei einem erneuten Lernen mit den neuen Daten wird so die Reichweite der hohen weiter steigen und der Effekt sich weiter verstärken. Ein möglicher Ansatzpunkt wäre die Gegensteuerung mit einer Positionierungsbestrafung gegenzusteuern.

7.4 Ausblick

Obwohl in dieser Arbeit erheblicher Aufwand betrieben wurde um die Reichweitenvorhersage so weit wie möglich zu optimieren, gibt es immer noch viele Punkte und Ansätze zur Verbesserung.

Ein nächster Schritt wäre durch die Entwicklung eines oder mehrerer Modelle für die noch zukünftige Reichweite möglich. Dadurch könnten auch längere Effekte einbezogen werden und die Ergebnisse direkt verglichen werden. Ein Ansatz dafür

wären Neuronale Netze, welche mit verschobenen Rahmen um die generierten Reichweiten trainiert werden, ähnlich den Bag-of-Words bei Word2Vec [15].
Ebenso wäre eine Anpassung der Startseite an Personen oder Gruppen von Facebook oder Twitter mit jeweils eigenen Modellen möglich. Dafür wären jedoch größere Datenmengen notwendig.
Ein anderer unbeleuchteter Aspekt einer automatisierten Kuratierung ist deren Manipulationssicherheit. Mit Blick auf die mutmaßlichen Manipulationen der US-Wahlen durch Russland, wäre die Erkennung und das Ausnutzen von Mustern solcher Lösungen bei einer oder mehrerer Nachrichtenportale ebenfalls ein gutes Instrument um die allgemeine Stimmung zu beeinflussen.

Literaturverzeichnis

[1] U. Dolata und J.-F. Schrape, Internet, mobile devices und die Transformation der Medien: radikaler Wandel als schrittweise Rekonfiguration, edition sigma, 2012.

[2] Webtrekk, „JSON/RPC API," 24 03 2017. [Online]. Available: https://support.webtrekk.com/hc/de/articles/115001497529-JSON-RPC-API. [Zugriff am 11 06 2017].

[3] Chartbeat, „Chartbeat Api," [Online]. Available: https://chartbeat.com/docs/api/. [Zugriff am 14 06 2017].

[4] C. M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Secaucus, NJ: Springer-Verlag New York, Inc., 2006.

[5] S. Rudolph, *Maschinelles Lernen,* Heidelberg, 2010.

[6] Statwing, „Interpreting residual plots to improve your regression," [Online]. Available: http://docs.statwing.com/interpreting-residual-plots-to-improve-your-regression/. [Zugriff am 20 08 2017].

[7] Orzetto, „Coefficient of determination," 06 12 2010. [Online]. Available: https://en.wikipedia.org/wiki/File:Coefficient_of_Determination.svg. [Zugriff am 10 10 2017].

[8] J. Frost, „Regression Analysis: How Do I Interpret R-squared and Assess the Goodness-of-Fit?," 30 05 2013. [Online]. Available: http://blog.minitab.com/blog/adventures-in-statistics-2/regression-analysis-how-do-i-interpret-r-squared-and-assess-the-goodness-of-fit. [Zugriff am 02 07 2017].

[9] Business Intelligence Info, „R-squared Shrinkage and Power and Sample Size Guidelines for Regression Analysis," 11 11 2014. [Online]. Available: http://www.businessintelligenceinfo.com/category/predictive-analytics/minitab. [Zugriff am 12 10 2017].

[10] R. Rojas, „The Noble Eightfold Path to Linear Regression," [Online]. Available: https://www.inf.fu-berlin.de/inst/ag-ki/rojas_home/documents/tutorials/LinearRegression.pdf. [Zugriff am 17 07 2017].

[11] R. Tibshirani, „Regression Shrinkage and Selection via the Lasso," *Journal of the Royal Statistical Society. Series B (Methodological),* Bd. 58, Nr. 1, pp. 267-288, 1996.

[12] R. Rojas, „Was können neuronale Netze?,“ in *Mathematische Aspekte der Angewandten Informatik*, Mannheim, BI-Verlag, 1994, pp. 55-88.

[13] R. Rojas, „Künstliche Neuronale Netze als Neues Paradigma der Informationsverarbeitung,“ 2011. [Online]. Available: http://page.mi.fu-berlin.de/rojas/2001/nn-paradigma.pdf. [Zugriff am 20 10 2017].

[14] Y. Bengio, R. Ducharme, P. Vincent und C. Janvin, „A neural probabilistic language model,“ *The Journal of Machine Learning Research,* Nr. 3, pp. 1137-1155, 3 1 2003.

[15] T. Mikolov, K. Chen, G. Corrado und J. Dean, „Efficient Estimation of Word Representations in Vector Space,“ in *Proceedings of Workshop at the International Conference on Learning Representations*, Scottsdale, 2013.

[16] X. Rong, „word2vec Parameter Learning Explained,“ CoRR, 2014.

[17] Q. Le und T. Mikolov, „Distributed Representations of Sentences and Documents,“ in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, Beijing, China, 2014.

[18] R. Bandari, S. Asur und B. A. Huberman, „The pulse of news in social media: Forecasting popularity,“ CoRR, 2012.

[19] G. Szabo und B. A. Huberman, „Predicting the Popularity of Online Content,“ *Communications of the ACM,* Nr. 53, pp. 80-88, 08 2010.

[20] A. Tatar, P. Antoniadis, M. Dias De Amorim und S. Fdida, „From Popularity Prediction to Ranking Online News,“ *Social Network Analysis and Mining,* p. 4:174, 12 02 2014.

[21] A. M. Dai, C. Olah und Q. V. Le, „Document Embedding with Paragraph Vectors,“ CoRR, 2015.

[22] E. J. Alvarez und H. Bast, „A review of word embedding and document similarity algorithms applied to academic text.,“ Freiburg, 2017.

[23] S. Atdag und V. Labatut, „A Comparison of Named Entity Recognition Tools Applied to Biographical Texts,“ CoRR, 2013.

[24] J. R. Finkel, T. Grenager und C. Manning, „Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling,“ in *43rd Annual Meeting on ACL*, 2005.

[25] L. Ratinov und D. Roth, „Design challenges and misconceptions in named entity recognition,“ in *13th Conference on Computational Natural Language Learning*, 2009.

[26] T. Reuters, „Calais Web Service,“ 2008. [Online]. Available: http://www.opencalais.com/. [Zugriff am 12 07 2017].

[27] Alias-i, „LingPipe 4.1.0,“ 2008. [Online]. Available: http://alias-i.com/lingpipe. [Zugriff am 12 07 2017].

[28] Amazon, „Amazon Machine Learning,“ [Online]. Available: https://aws.amazon.com/de/aml/. [Zugriff am 21 06 2017].

[29] Google, „Google Cloud Machine Learning Engine,“ [Online]. Available: https://cloud.google.com/ml-engine/. [Zugriff am 21 06 2017].

[30] Microsoft, „Azure Machine Learning Studio,“ [Online]. Available: https://azure.microsoft.com/de-de/services/machine-learning-studio/. [Zugriff am 17 06 2017].

[31] Y. Borghol, S. Ardon, N. Carlsson, D. L. Eager und A. Mahanti, „The Untold Story of the Clones: Content-agnostic Factors that Impact YouTube Video Popularity,“ CoRR, 2013.

[32] J. Berger und K. L. Milkman, „What Makes Online Content Viral?,“ *Journal of Marketing Research,* Bd. 49, Nr. 2, pp. 192-205, 2012.

[33] F. R., „A new readability yardstick.,“ *Journal of Applied Psychology,* Bd. 32, Nr. 3, pp. 221-233, 1948.

[34] T. Amstad, „Wie verständlich sind unsere Zeitungen?,“ Studenten-Schreib-Service, Zürich, 1978.

[35] S. Kong, F. Ye und L. Feng, „Predicting future retweet counts in a microblog,“ *Journal of Computational Information Systems,* Nr. 10, pp. 1393-1404, 2014.

[36] „Pandas,“ [Online]. Available: https://pandas.pydata.org/index.html. [Zugriff am 04 08 2017].

[37] S. Bansal, „Textstat,“ 04 08 2017. [Online]. Available: https://github.com/shivam5992/textstat. [Zugriff am 14 09 2017].

[38] R. Řehůřek, „Word2vec in Python, Part Two: Optimizing,“ 21 09 2013. [Online]. Available: https://rare-technologies.com/word2vec-in-python-part-two-optimizing/. [Zugriff am 10 06 2017].

[39] C. Bishop, Pattern recognition and machine learning, Bd. Vol. 4., New York: Springer, 2006.

[40] R. O. Duda, P. E. Hart und D. G. Stork, Pattern Classification, Bd. 2nd Edition, John Wiley & Sons, 1999.

[41] Webtrekk GmbH, „Webtrekk: Customer Intelligence & Marketing Analytics Platform,“ [Online]. Available: https://www.webtrekk.com/de/startseite/. [Zugriff am 19 07 2017].

[42] C. Sauer, „Webtrekks Webanalyse-Software Q3 erhält TÜV Zertifikat,“ 18 01 2010. [Online]. Available: https://www.adzine.de/2010/01/webtrekks-webanalyse-software-q3-erhaelt-tuev-zertifikat-web-analytics/. [Zugriff am 16 09 2017].

[43] P. Bhardwaj, „How does doc2vec represent feature vector of a document?,“ 29 10 2016. [Online]. Available: https://www.quora.com/How-does-doc2vec-represent-feature-vector-of-a-document-Can-anyone-explain-mathematically-how-the-process-is-done/answer/Piyush-Bhardwaj-7. [Zugriff am 22 10 2017].

[44] MongoDB Inc., October 2013. [Online]. Available: http://docs.mongodb.org/manual/reference/operator/update/.

Anhang

AUSLAND SCHWARZE LISTE

Trump erklärt Nordkorea zum Förderer von Terrorismus

US-Präsident Donald Trump hat Nordkorea zum staatlichen Förderer von Terrorismus erklärt. Er habe die nordkoreanische Regierung wieder auf eine entsprechende schwarze Liste gesetzt, sagte Trump.

20.11.2017 | 3 Kommentare

Anhang I: Beispiel für einen Teaser „der WELT“ (vom 20.11.2017)

Merkmalsname	Zusätzliche Werte
sections	
stats_engaged_visit	10 historische Werte, Mittel und Median
stats_engaged_time	10 historische Werte, Mittel und Median
stats_links	
stats_people	
stats_read	
stats_direct	
stats_visits	10 historische Werte, Mittel und Median
stats_recirc	
stats_platform_a	
stats_platform_d	
stats_platform_g	
stats_platform_f	
stats_platform_m	
stats_platform_t	
stats_platform_engaged_a	
stats_platform_engaged_d	
stats_platform_engaged_g	
stats_platform_engaged_f	
stats_platform_engaged_m	
stats_platform_engaged_t	
stats_toprefs_0	Domain/Wert
stats_toprefs_1	Domain/Wert
stats_toprefs_2	Domain/Wert
stats_toprefs_3	Domain/Wert
stats_toprefs_4	Domain/Wert
stats_toprefs_5	Domain/Wert
stats_search	
stats_domload	10 historische Werte, Mittel und Median
stats_write	
stats_num_refs	
stats_idle	
stats_internal	
stats_social	
stats_new	
stats_type	
stats_scroll	10 historische Werte, Mittel und Median

Anhang II: Chartbeat Metriken