

Friedrich-Schiller-Universität Jena
Philosophische Fakultät
Institut für Germanistische Sprachwissenschaft

Modellierung von Leseverhalten auf Online-Zeitungsportalen anhand von Deep Learning

Masterarbeit zur Erlangung des akademischen Grades
Master of Arts (M.A.)

vorgelegt von Susanna Rücker
Matrikelnummer: 182648
geboren am 02.05.1995 in Freiburg im Breisgau

Erstgutachter: Prof. Dr. Udo Hahn
Zweitgutachter: Dr. Steffen Wagner (INWT Statistics)

Jena, den 29. Juni 2021

Inhaltsverzeichnis

Abstract	1
1 Einleitung	2
2 Forschungsüberblick: Modellierung von User Engagement	6
2.1 Explizites vs. implizites Feedback und User Engagement	6
2.2 Aufenthaltsdauer als Proxy für Zufriedenheit und Relevanz	9
2.3 Popularität von Online-Zeitungsartikeln	12
2.4 (Personalisierte) News Recommendation	15
2.5 Abgrenzung: Aufenthaltsdauer vs. Lesedauer	18
3 Datengrundlage	21
3.1 Korpuserstellung: Beschaffung der Artikeltexte	21
3.2 Der vollständige Datensatz, Entscheidung für ████	23
3.3 Der ████ -Datensatz	27
3.4 Wahl der absoluten Aufenthaltsdauer als Zielvariable	31
4 Methodische Grundkonzepte	34
4.1 Textrepräsentation mit <i>Bag of Words</i> (BOW)	34
4.2 Wort- und Dokumentenvektoren (<i>Word Embeddings</i>)	37
4.3 Baseline-Klassifikatoren: Ridge Regression, XGBoost	39
4.4 Deep Learning: Grundlegende Architekturen	40
4.4.1 Feed-Forward Netze	41
4.4.2 Convolutional Neural Networks	43
4.4.3 Rekurrente Neuronale Netze	44
4.4.4 Training von Neuronalen Netzen	45
4.5 Transformer-Modelle: <i>Self-Attention</i>	46
4.6 BERT: <i>Pretraining</i> und <i>Transfer Learning</i>	49
5 Modelle zur Vorhersage der Aufenthaltsdauer	53
5.1 Baseline 1 und 2: Mittelwert und Textlänge	53
5.2 BOW-Modelle	53
5.3 Baseline 3: Dokumentenembeddings	54
5.4 CNN-Baseline	55
5.5 Einfache BERT-Modelle	56
5.6 BERT erweitert mit Textlänge	57
5.7 Hierarchische BERT-Modelle	58

6	Ergebnisse	62
6.1	Metriken zur Modellevaluation	62
6.2	Evaluationsergebnisse	63
7	Einblick in Modellentscheidungen mit SHAP	70
7.1	Grundidee von SHAP	70
7.2	Einblick in die Modellvorhersagen von BertFFN	71
8	Diskussion und Ausblick	76
9	Fazit	79
	Literaturverzeichnis	81
	Abbildungsverzeichnis	89
	Tabellenverzeichnis	90
A	Appendix: Einblick in BOWXGBoost mit SHAP	91
	Eigenständigkeitserklärung	95

Abstract

The present study deals with user engagement on German online news articles. Several implicit feedback measures or Key Performance Indicators (KPIs) – such as pageviews or dwell time – are commonly used as a proxy for measuring user engagement on websites. After a thorough discussion concerning the use of implicit user feedback and related fields of research, the main focus of this work is building different kinds of models for predicting average user dwell time given only the text of each article. A large corpus (consisting of news articles and their respective KPI measures) was created specifically for this study, part of which (36383 articles from the German daily newspaper [REDACTED], [REDACTED]) was then chosen for the prediction task. The line of models includes several baselines, two of them using Bag-of-Words features, but relies mostly on including the well-known pretrained transformer model BERT in several Deep Learning architectures. All models are evaluated and compared on unseen test data, using various evaluation metrics. This work deals with the problem of applying BERT to longer documents, given its limitation on input sequence length. Most of the models simply truncate the article and just use the first part – which turns out to be a valid approach resulting in good predictions of dwell time. However, two of the more complex models take a hierarchical approach, splitting the article in several smaller sections and combining the output of each section. A further analysis gives insights on the dwell time predictions of two models (one BOW-baseline and one model including BERT), using the tool SHAP for interpreting model predictions.

1 Einleitung

Was in den letzten Jahrzehnten der morgendliche Blick in die gedruckte Tageszeitung war, ist für viele Menschen heute der tägliche, wenn nicht stündliche Blick auf ein Display unterschiedlicher Größe. Viele Menschen nutzen zusätzlich oder ausschließlich Online-Nachrichtenportale, um auf dem Laufenden zu bleiben, oder um unterhalten zu werden. Angebot und Nutzungsmöglichkeiten sind dabei sehr vielfältig und teilweise unübersichtlich, wenn nicht gar überfordernd. Es gibt eine sehr große Anzahl an Anbietern und Artikeln, aus denen es auszuwählen gilt. Wie findet eine Nutzerin Artikel, die sie für interessant, relevant oder allgemein lesenswert hält? Sie kann auf verschiedene Weise vorgehen. Sie kann auf die ihr bekannten und von ihr geschätzten Quellen zurückgreifen – etwa die Webseite ihrer Lieblingszeitung oder ihrer regionalen Tageszeitung öffnen – und dort über Schlagzeilen, Fotos oder Rubriken Artikel auswählen. Neben den einzelnen Anbietern kann sie aber auch anbieterübergreifende Portale, wie etwa Google News, nutzen. Dort werden aktuelle Artikel präsentiert und vorgeschlagen, meist nach Thematik oder lokaler Relevanz sortiert. Auch hier entscheidet der Nutzer anhand von Titel, Thema einordnung, Erscheinungsdatum, Foto und vielleicht auch Seriosität oder sonstiger Attraktivität der Nachrichtenquelle, welche Artikel ihn interessieren könnten. Zusätzlich zu solchen übergreifenden Nachrichtensuchmaschinen, die Artikel gesammelt präsentieren, nutzen viele Menschen außerdem Soziale Medien wie Twitter oder Facebook, um für sie relevante Artikel zu finden. Dort gelangen sie entweder über die Auftritte der Zeitungsportale selbst zu den einzelnen Artikeln, oder sie folgen den Links, die andere Nutzer posten. Welche Nachrichten dem Nutzer bei dieser Art von Suche vorgeschlagen werden und welche nicht, ist ein sehr undurchsichtiger Prozess und abhängig von den inneren Arbeitsweisen der Portale. Wie wird entschieden, welche Artikel relevant sind und einem Nutzer präsentiert werden sollten? Wie zeigt sich Popularität eines Artikels? Viele Portale bedienen sich der Such- bzw. Klickhistorie eines einzelnen Benutzers, um dessen Interessen ausfindig zu machen und ihm Artikel zu präsentieren, die bei ihm mit hoher Wahrscheinlichkeit auf Interesse stoßen könnten. Hierbei handelt es sich um personalisierte Vorschläge.

Neben dieser benutzerspezifischen Relevanz können Artikel aber auch allgemein – also unabhängig von einem spezifischen Benutzer – in Bezug auf ihre Popularität oder Relevanz bewertet werden. Also allgemein gesprochen: Welcher Artikel ist interessant, relevant oder allgemein lesenswert? Welche Artikel haben gute Chancen, von vielen Lesern und Leserinnen gelesen zu werden? Welche Artikel hinterlassen danach einen möglichst zufriedenen Benutzer, der viel mit dem Portal interagiert, also Kommentare hinterlässt, den Artikel auf seinen eigenen Accounts verlinkt, das

Portal positiv bewertet, womöglich ein Abonnement eingeht, oder es zumindest in guter Erinnerung behält, weitere Artikel liest und somit auch Werbeanzeigen sieht und vielleicht anklickt?

Hier zeigt sich schnell, dass nicht nur der Leser selbst daran interessiert ist, (für ihn) besonders relevante Artikel und insgesamt ein angenehmes Leseerlebnis vorzufinden. Vor allem auch das Nachrichtenportal, die Tageszeitung, die Redaktion bzw. die Journalistin interessieren sich dafür, welche Artikel gut ankommen und viel gelesen werden. Daran können sich die inhaltliche und zeitliche Planung, sowie die Positionierung der Artikel, das Einblenden von Werbeanzeigen oder auch Entscheidungen über Zugang und Preisgestaltung der Artikel (Welche Artikel werden öffentlich zugänglich gemacht, welche hingegen nur bezahlenden Kunden?) orientieren. Auch in Hinblick auf die Attraktivität für Werbeschaltende hat ein Portal guten Grund, besonders hochwertige und relevante Artikel liefern zu wollen, die das Interesse einzelner, aber besser möglichst vieler Leser und Leserinnen wecken.

Nachrichtenportale haben also ein Interesse daran, ihren Nutzern besonders interessante und involvierende Inhalte zu liefern. Sie bezwecken damit, dass die Nutzer zufrieden mit dem Angebot sind und sich ausführlich mit der Webseite beschäftigen. Explizite Rückmeldungen (in Form von *Ratings* oder *Likes*) der Nutzer sind eine mögliche, aber vergleichsweise aufwändige, kostspielige und ineffiziente Art, Informationen über die Nutzerzufriedenheit zu erheben. Es müssen sinnvolle Fragen formuliert und der Nutzerin an geeigneter Stelle gestellt werden. Mutmaßlich hinterlässt nur ein sehr kleiner Teil der Nutzer tatsächlich Antworten. Gleichzeitig besteht das Risiko, dass sich Leser durch solche expliziten Nachfragen im Lesefluss gestört fühlen und entweder schlechte Bewertungen hinterlassen oder verärgert die Seite verlassen – was natürlich nicht im Sinne des Nachrichtenportals ist. Auch entsprechen die Werte, die Nutzer über sich und ihre Zufriedenheit selbst liefern, nicht unbedingt ihrem tatsächlichen (unbewussten) Verhalten.

Statt *explizitem* wird daher häufig auf *implizites Feedback* zurückgegriffen, gemeint sind Messwerte, die das echte ungesteuerte Verhalten der Nutzer beschreiben. Im Bereich von Webseitenoptimierung wird dieses Nutzerverhalten als *User Engagement* bezeichnet. Gemeint sind verschiedene Arten der Interaktion, mit denen der Nutzer in Kontakt mit der Webseite tritt. Um das Engagement der Nutzer zu quantifizieren, haben sich verschiedene Maßzahlen etabliert, die sogenannten *Key Performance Indicators* (KPIs). Mithilfe von Tracking-Software (wie etwa Google Analytics) können Anbieter von Webinhalten KPIs ihrer Nutzer messen, um Informationen über den Erfolg ihrer Inhalte und das Verhalten ihrer Nutzer analysieren und optimieren zu können. Der bekannteste und einer der meistgenutzten KPI-Werte für User Engagement ist die Zahl an Seitenaufrufen (*pageviews*, *clicks*, Klickzahlen, Besuche), die

eine Webseite generiert. Während Klickzahlen vor allem die generelle Sichtbarkeit und Reichweite von Inhalten widerspiegeln, sind sie aber nur eingeschränkt als Maß für *Zufriedenheit* mit den Inhalten interpretierbar. Gleichzeitig kann ein Artikel durchaus gut und lesenswert sein, aber aufgrund von ungünstiger Platzierung oder unbekannterer Nachrichtenquelle nur wenigen Nutzern vorgeschlagen werden und somit nur wenige Klicks erzielen. In Ergänzung zu den Seitenaufrufen wird daher vor allem die Aufenthaltsdauer (*Dwell Time*, *Time on Page*) bei Inhalten gemessen. Allgemein wird dabei eine längere Aufenthaltsdauer als positives Indiz für Nutzerzufriedenheit gewertet. Neben Klickzahlen und Aufenthaltsdauer finden sich seltener weitere Maßzahlen, wie etwa die Anzahl an Downloads oder gesetzten Lesezeichen, Anzahl der Kommentare, die Nutzer hinterlassen, Informationen über ihr Scrolling-Verhalten und ihre Mausbewegung oder – in Studien mit Laborexperimenten – auch ihr Blickverhalten.

Auch die vorliegende Arbeit beschäftigt sich – aus computerlinguistischer Sicht – mit implizitem Feedback, genauer mit dem Leseverhalten bei Zeitungsartikeln auf Online-Zeitungsportalen. Im Fokus der Arbeit steht die Modellierung der durchschnittlichen Aufenthaltsdauer, die Leser und Leserinnen bei einem Artikel verweilen, anhand des Artikeltextes. Als Grundlage für die Arbeit wurde ein großer Datensatz (mehr als 100000 Artikel) erstellt, bestehend aus Artikeln von Online-Portalen von vier Tageszeitungen. Neben den vollständigen Artikeltexten liegen für jeden Artikel einige KPI-Werte vor. Für die Modellierung dieser Arbeit wurde ein Teil davon (36383 Artikel und ihre KPI-Werte) ausgewählt, es handelt sich um Artikel der [REDACTED] ([REDACTED]).

Die computerlinguistischen Modelle, die zur Vorhersage der Aufenthaltsdauer verwendet werden, erhalten den Artikeltext selbst als Eingabe. Dabei werden in der Vorverarbeitung und Erstellung der Merkmale (*Features*) und bei den Modellarchitekturen sowohl herkömmliche als auch neuere Methoden aus dem Deep Learning (DL) verwendet. Fokus der Arbeit liegt in der Verwendung von BERT, einem vortrainierten Transformer-Modell. Die Anwendung von BERT auf lange Dokumente (wie bei Zeitungsartikeln der Fall) ist nicht unproblematisch, da BERT grundsätzlich nur Eingaben mit einer bestimmten Maximallänge (also Anzahl an Tokens) verarbeiten kann. Insbesondere zwei der Modelle (die hierarchischen BERT-Modelle) gehen dieses Problem an, indem sie den Artikel abschnittsweise verarbeiten.

In Abschnitt 2 wird zunächst ein Überblick über verschiedene Forschungsfelder gegeben, die sich mit der Modellierung von User Engagement bzw. impliziten Nutzerwerten wie Klickzahlen und Aufenthaltsdauer befassen. Neben Arbeiten aus dem Bereich der Modellierung von Popularität von Zeitungsartikeln kommt auch der Bereich Websuche zur Sprache, da dieser grundlegende Pionierarbeit in Bezug auf

den Zusammenhang zwischen explizitem und implizitem Nutzerfeedback geleistet hat. Als abzugrenzendes aber verwandtes Feld wird außerdem auf die Verwendung von User Engagement bei (personalisierter) *News Recommendation* eingegangen. Zwar unterscheidet sich hier die Anwendung deutlich von der vorliegenden Arbeit, da neben Artikeln und ihrer Popularität auch ein spezifischer Nutzer und sein Interessenprofil im Mittelpunkt stehen. Methodisch finden sich in diesem Bereich allerdings deutlich modernere Methoden zur Textverarbeitung, als in den Arbeiten zur *allgemeinen* Popularität von Artikeln. Schließlich wird die hier verfolgte Fragestellung – Modellierung der Aufenthaltsdauer – mit Arbeiten kontrastiert, die sich mit Lesezeit beschäftigen, zwei Konzepte, die nicht verwechselt werden sollten.

Abschnitt 3 beschreibt den gesamten, sowie den spezifischen █████-Datensatz und seine Erstellung näher. Dabei wird auf die Beschaffung der Artikeltexte eingegangen und eine Erklärung der einzelnen KPI-Werte geliefert. Außerdem wird die Entscheidung für den █████-Datensatz begründet.

Im Abschnitt 4 werden die für die Modellierung der Aufenthaltsdauer notwendigen computerlinguistischen Grundkonzepte und Werkzeuge vorgestellt. Neben der Textrepräsentation anhand von *Bag of Words* (BOW) und Wort- bzw. Dokumentenvektoren (*Word Embeddings*) wird ein Überblick in die Modellierung anhand von *Deep Learning* (DL) gegeben, bevor die Transformer-Modelle und insbesondere BERT als das zentrale Modellbauteil dieser Arbeit vorgestellt werden.

Der anschließende Abschnitt 5 verwendet die genannten Konzepte und stellt alle in der Arbeit verwendeten Modelle vor. Verwendet werden neben einer *Bag-of-Words*-Baseline in zwei Varianten weitere einfache sowie auf Word Embeddings basierende Baselines. Das Herzstück der Modellierung sind verschiedene DL-Modelle, die das Transformer-Modell BERT in unterschiedlicher Art verwenden. Hier wird zwischen den *einfachen* und den *hierarchischen* BERT-Modellen unterschieden. Erstere betrachten nur den ersten Textabschnitt, letztere unterteilen den Artikel in Abschnitte, verarbeiten sie getrennt und kombinieren die Ergebnisse.

Abschnitt 6 beschreibt zunächst die Evaluationsmetriken, widmet sich dann den Ergebnissen und vergleicht die einzelnen Modelle und ihre Performanz zur Vorhersage der Aufenthaltsdauer. In Abschnitt 7 werden anhand eines Tools zur Interpretierbarkeit von Modellen bzw. ihren Vorhersagen (SHAP) die Modellvorhersagen eines der BERT-Modelle näher betrachtet, also ein Einblick in den Bereich der *Interpretierbarkeit* von DL-Modellen gegeben.

In der Diskussion in Abschnitt 8 werden einige kritische Punkte der Arbeit und der Modellierung angesprochen, aus denen sich viele Ideen für Folgearbeiten ableiten lassen, bevor Abschnitt 9 die Erkenntnisse der vorliegenden Arbeit abschließend zusammenfasst.

2 Forschungsüberblick: Modellierung von User Engagement

Die vorliegende Arbeit beschäftigt sich mit der Modellierung von User Engagement – genauer der mittleren Aufenthaltsdauer – in Bezug auf Online-Zeitungsartikel. Im Folgenden werden verwandte Forschungsfelder vorgestellt, deren Arbeiten in unterschiedlichem Maße inhaltliche oder methodische Überschneidungen zur hiesigen Fragestellung aufweisen.

Der Aufbau dieses Überblicks ist dabei wie folgt. Zunächst werden in 2.1 einige (frühe) Arbeiten vorgestellt, die sich allgemein mit dem Zusammenhang von explizitem und implizitem Nutzerfeedback beschäftigen. Dabei wird speziell auf die Relevanz der Aufenthaltsdauer (*Dwell Time*) als Maß für Nutzerzufriedenheit eingegangen. Der Mehrwert, den dabei spezifisch die Aufenthaltsdauer gegenüber der reinen Betrachtung von Klickzahlen bietet, wurde vor allem in Arbeiten behandelt, die sich mit Dokumentenrelevanz bei Websuche und Webbrowsering beschäftigen. Daher werden in 2.2 einige solche Arbeiten behandelt, obwohl sich die dortigen Fragestellungen von der hier behandelten offensichtlich unterscheiden.

Nach dieser allgemeinen Betrachtung von Aufenthaltsdauer als Maß für Zufriedenheit und User Engagement, wird dann in Abschnitt 2.3 das Forschungsfeld der vorliegenden Arbeit angesprochen: Arbeiten, die sich der Vorhersage der Popularität von Online-Zeitungsartikeln (*Online News Popularity Prediction*) widmen.

Der anschließende Abschnitt 2.4 behandelt wiederum ein thematisch abzugrenzendes Feld: Während sich die vorliegende Arbeit mit der *generellen* Popularität von Zeitungsartikeln beschäftigt, werden hier Arbeiten aus dem Bereich der (personalisierten) News Recommendation vorgestellt. Trotz deutlicher Unterschiede in Fragestellung und Datengrundlage sind diese Arbeiten vor allem aus methodischer Sicht (Textverarbeitung), aber auch in ihrer Verwendung der Aufenthaltsdauer als Maß für Zufriedenheit relevant für die vorliegende Arbeit.

Im letzten Abschnitt 2.5 schließlich wird kurz der Bereich Lesedauer und Lesbarkeit angerissen. Dies dient primär der Abgrenzung: Die Aufenthaltsdauer sollte nicht mit der *Lesedauer* des Artikels verwechselt werden.

2.1 Explizites vs. implizites Feedback und User Engagement

Ob bei der Einschätzung der Qualität oder Relevanz von Suchergebnissen oder Artikelvorschlägen, oder bei der Bestimmung der Qualität oder Popularität von Zeitungsartikeln oder Webseiten, Wissen über die *Zufriedenheit* eines oder mehrerer Nutzer ist in vielen Bereichen von Bedeutung. Dieses *User Feedback* kann sowohl in

Form von *expliziten* als auch *impliziten* Rückmeldungen erhoben werden (Kim et al., 2014; Kelly & Belkin, 2004; Claypool et al., 2001; Fox et al., 2005; Homma et al., 2020; Kellar et al., 2004; Yi et al., 2014; C. Liu et al., 2010)

Explizite Erhebungen über die Zufriedenheit sind etwa Fragen, die den Nutzern gestellt werden. Der Nutzer gibt dabei selbst Auskunft darüber, wie sehr ihm ein Inhalt gefallen hat bzw. wie relevant oder passend er ihn fand. Dabei kann es sich um Entscheidungsfragen (*Fanden Sie diesen Treffer hilfreich?*) oder Ratings anhand von Likert-Skalen (*Auf einer Skala von 1 bis 5, wie hilfreich fanden Sie diesen Treffer?*) handeln (Kellar et al., 2004). Auch Freitexte im Sinne von Bewertungen oder Kommentaren, sind Methoden von explizitem Feedback. Andere – weniger aufwändige – Methoden sind das direkte Vergeben von *Likes* oder *Dislikes* oder die Vergabe von (meist 1 bis 5) Sternen. Explizites Erfragen von Nutzerzufriedenheit kommt häufig aber nicht ausschließlich in Laborstudien zur Anwendung, bei denen Studienteilnehmer in überwachten Settings beobachtet werden. Solche expliziten Befragungen sind technisch aufwändig (Laborexperimente) oder erfordern ein hohes Maß an Mitarbeit der Nutzer. Gleichzeitig unterbrechen sie das natürliche Browsingverhalten der Nutzer (Kellar et al., 2004). In realen Settings besteht das Risiko, dass nur ein sehr kleiner Teil der Nutzer überhaupt die erwünschten Bewertungen (*Likes* oder Sternbewertungen) hinterlässt. Gleichzeitig ist möglich, dass explizite Bewertungen gar nicht unbedingt die tatsächliche Zufriedenheit oder das Engagement der Nutzer widerspiegeln: Vielleicht hat eine Nutzerin einen Zeitungsartikel sehr interessiert gelesen, fand ihn aber emotional aufwühlend oder lehnt ihn inhaltlich ab. Sie hinterlässt anschließend eine negative Bewertung, völlig unabhängig davon, ob nach Zufriedenheit, Relevanz oder Interessantheit gefragt wurde.

Unter **implizitem** Nutzerfeedback hingegen versteht man die Verwendung von Daten, die Aufschluss über das natürliche Verhalten bzw. die Interaktion der Nutzer mit Online-Inhalten geben, ohne diese explizit zu erfragen. Die Erhebung von implizitem Feedback ist – im Gegensatz zu explizitem Feedback – mit geringerem Aufwand verbunden, viele der Daten liegen als Nebenprodukt vor oder sind mit einfachen Tools messbar. Gemeint sind Daten wie Klickzahlen (*pageviews*, *clickcount*), Klickrate (*Click-Through-Rate*, CTR)¹, Aufenthaltsdauern, Scrolling-, Maus- und Blickverhalten oder – vor allem in der mobilen Nutzung am Smartphone – Viewport-Daten, also Informationen über den dem Nutzer zu jedem Zeitpunkt sichtbaren Bereich einer Webseite (Lagun & Lalmas, 2016; Kellar et al., 2004). Weitere Maße zur Interaktion der Nutzer mit Webinhalten sind etwa die Nutzung von Lesezeichensetzung oder Download-, Druck-, Such- oder Teilfunktionen auf Webseiten, sowie *Copy-Paste*-

¹Gemeint ist ein Anteil: Wie oft wird ein Inhalt aufgerufen, ins Verhältnis gesetzt zu seiner Sichtbarkeit? Bezogen auf Online-Zeitungsartikel: Wie oft wird der Artikel tatsächlich angeklickt, wenn er einem Leser angezeigt wird?

Vorgänge oder das Öffnen von Tabs (Claypool et al., 2001). Für diese Interaktion zwischen Nutzer und Webinhalt finden sich verschiedene Bezeichnungen, die alle ähnliche Konzepte meinen, darunter neben **User Engagement** auch Nutzerverhalten (*User Behavior*), Nutzerzufriedenheit (*User Satisfaction*), oder Nutzerinteresse (*User Interest*). Die verschiedenen Maße von User Engagement werden unter dem Stichwort *Key Performance Indicator* (KPI) zusammengefasst, da sie als Maße für die Performanz oder Reichweite von Webseiten herangezogen werden.

Unter Arbeiten, bei denen User Engagement untersucht wird, lassen sich grundsätzlich zwei verschiedene Arten der Datenerhebung finden (C. Liu et al., 2010): Einerseits gibt es kontrollierte Laborstudien, deren Probanden in einem überwachten Setting in ihrem Verhalten bezüglich einer Aufgabe (Websuche, Webbrowsing, Lektüre von Nachrichten) beobachtet werden (implizites Feedback) und häufig gleichzeitig bzw. im Anschluss nach expliziten Bewertungen gefragt werden. Auf der anderen Seite gibt es Studien, deren Daten – wie in dieser Arbeit – aus natürlichen (realen) Webinteraktionen stammen, hier liegt meist ausschließlich implizites Feedback in Form von Nutzer-Logdateien vor. Außerdem unterscheiden sich Arbeiten darin, ob die Daten *einzelner* Nutzer vorliegen und analysiert werden (vor allem in der personalisierten News Recommendation), oder ob Aggregationswerte (Mittelwert, Median) mehrerer Nutzer betrachtet werden.

Das einfachste und etablierteste KPI-Maß sind Daten über die Anzahl der Seitenaufrufe (*pageviews*, auch als Besuche oder Klicks bezeichnet). Zwar bieten Seitenaufrufe ein gutes Maß für Sichtbarkeit oder Reichweite von Webseiten bzw. Inhalten, sie sagen aber nicht unbedingt etwas über das Verhalten oder die Zufriedenheit der Nutzer *nach* dem Klick aus (*post-click*-Verhalten). Dieser Missstand wird vor allem in neueren Arbeiten häufig kritisiert (Lu et al., 2018; Yi et al., 2014; C. Wu et al., 2020, 2021). Hier spielt das unter der Bezeichnung *Clickbaiting* bekannte Phänomen eine Rolle. Gemeint ist die mögliche Diskrepanz zwischen der Erwartungshaltung des Lesers durch den Titel und dem tatsächlichen Inhalt des Artikels, der sich hinter dem Titel verbirgt. Ein Artikel kann eine reißerische Überschrift haben, oft angeklickt werden, inhaltlich aber nicht den Erwartungen der Klickenden entsprechen und schlussendlich Unzufriedenheit seiner Leser hervorrufen (Omidvar et al., 2018; C. Wu et al., 2020). Außerdem sind Klickzahlen anfällig für Verzerrungen, die nichts mit der Nutzerzufriedenheit zu tun haben, wie etwa die Positionierung des zu klickenden Inhalts (Überschrift, Link) auf Suchmaschinen, Startseiten oder auf Social Media (Lu et al., 2018). Neben oder zusätzlich zu Klickzahlen wird hauptsächlich die Aufenthaltsdauer (*Dwell Time*, manchmal auch *Display Time* oder *Screen Time*) als Proxy für User Engagement verwendet (Homma et al., 2018; C. Liu et al., 2011, 2010; Lu et al., 2018; Zhou et al., 2018; C. Wu et al., 2021; Yi et al., 2014; J. Chen et al., 2021).

Zusammenfassend gilt: Die Erhebung von explizitem Feedback ist in vielen Bereichen ressourcenaufwändig und nur in geringem Umfang vorhanden. Gleichzeitig spiegelt explizites Feedback möglicherweise nicht das natürliche unbeobachtete Webverhalten der Nutzer wider, da es entweder in kontrollierten Laborsettings erhoben wurde, oder der Nutzer in seinem natürlichen Browsingverhalten unterbrochen wurde, um etwa Ratings abzugeben. Implizites Feedback hingegen kann mithilfe von Tracking-Software ohne großen Aufwand und in großen Datenmengen erhoben werden oder liegt bereits als Nebenprodukt vor. Die Nutzer werden dabei in ihrem natürlichen Verhalten nicht beeinflusst.

2.2 Aufenthaltsdauer als Proxy für Zufriedenheit und Relevanz

Die Verwendung von impliziten Feedback-Maßen beruht auf der Annahme, dass sie als Proxy für Konzepte wie *Zufriedenheit* oder *Relevanz* oder allgemeiner als *Engagement* der Nutzer herangezogen werden können. Studien, die diese Annahme überprüfen, entstammen vor allem dem Forschungsbereich von Websuchen oder allgemein Webbrowsing aus dem Bereich des Information Retrieval (IR). Sie überprüfen die Aussagekraft impliziter Maße, indem sie deren Zusammenhang mit explizit erhobenen Bewertungen betrachten. Diese (frühen) Arbeiten legen damit den Grundstein für die *generelle* Verwendung von impliziten Maßen für Nutzerzufriedenheit auch in anderen Bereichen und Fragestellungen – und sind daher auch für die vorliegende Arbeit relevant.

Der Großteil der Arbeiten findet einen positiven Zusammenhang zwischen der Aufenthaltsdauer oder anderen impliziten Maßen und gleichzeitig explizit erfragten Bewertungen von Nutzern zu Interesse oder Dokumentenrelevanz (Lagun & Lalmas, 2016; Claypool et al., 2001; Fox et al., 2005; Lu et al., 2018; C. Wu et al., 2020; Morita & Shinoda, 1994; Konstan et al., 1997; White & Kelly, 2006; Homma et al., 2018)².

Zu einem anderen Schluss kommen hingegen Kelly und Belkin (2004), die keine Korrelation zwischen der Aufenthaltsdauer bei Suchtreffern und der expliziten Bewertung zur Nützlichkeit der Treffer finden. Sie kritisieren ausdrücklich die Verwendung der mittleren Aufenthaltsdauer der einzelnen Nutzer und bewerten sie als nicht aussagekräftig genug. Die Autoren unterscheiden in ihrer Studie verschiedene Nutzung von Websuche (zum Beispiel Jobsuche, Lesen von Nachrichten, Online-Shopping) und beobachten signifikante Unterschiede in der (mittleren) Anzeigedauer (*Display*

²Homma et al. (2018) beobachten einen Zusammenhang zwischen Aufenthaltsdauer und Klickaktivität (gemeint ist hier das Klicken von Links auf der bereits betrachteten Seite), beides interpretieren sie als Ausdruck von *Interesse* des Nutzers.

Time) bei den Ergebnissen. Auch White und Kelly (2006) untersuchen Anzeigedauer als Maß für Relevanz und schließen, dass für unterschiedliche Tasks unterschiedliche Grenzwerte gewählt werden sollten. Auch J. Liu und Belkin (2015) und Kellar et al. (2004), die Dokumentenrelevanz betrachten, konstatieren die Aufenthaltsdauer zwar als guten generellen Indikator für Relevanz, weisen aber darauf hin, dass bestimmte kontextuelle Information (wie Task oder Thematik) die Interpretation der Dauer als Nützlichkeit der Dokumente erleichtert bzw. verfeinert.

Der Großteil der Studien unterstützt aber die Annahme, dass relevante Ergebnisse mehr Aufmerksamkeit des Nutzers erhalten, während mit irrelevanten Suchergebnissen wenig interagiert wird. Basierend auf dieser Erkenntnis verwenden viele weitere Arbeiten im Bereich Websuche implizites Feedback als Maß für die *Relevanz* der Suchergebnisse bzw. Webseiten und damit für die Zufriedenheit der Nutzer (Agichtein et al., 2006; C. Liu et al., 2011; Homma et al., 2018; Kim et al., 2014; C. Liu et al., 2010; Zhou et al., 2018; Guo & Agichtein, 2012)³. Nicht nur die Auswahl der bereitgestellten Suchergebnisse, auch die Anordnung (*Ranking*, etwa bei Xu et al. (2011)) kann mithilfe von implizitem Feedback optimiert werden.

Aufgrund der Beobachtung, dass ein Klick allein nicht unbedingt Zufriedenheit aussagt, wird häufig eine Mindestaufenthaltsdauer nach dem Klick als Hinweis für einen „zufriedenstellenden“ Klick gewertet, häufig wird ein Grenzwert von 30 Sekunden gewählt (Kim et al., 2014; Fox et al., 2005; Lu et al., 2018), Morita und Shinoda (1994) wählen 20 Sekunden. Homma et al. (2020) hingegen betrachten explizit kurze Aufenthaltsdauern (bis zu 5 Sekunden) auf Nachrichtenseiten und kommen zu dem Schluss, dass sehr kurzen Aufenthalte nicht unbedingt Desinteresse signalisieren, sondern nutzerspezifisch und je nach Artikeltopic auch mit Interesse assoziiert sind.

Kim et al. (2014) kritisieren die Verwendung eines fixen Grenzwertes als zu pauschal. Sie modellieren die Aufenthaltsdauer auf Webseiten, die als Ergebnis einer Suchanfrage vorgeschlagen wurden. Ihre Kritik beruht auf der Vermutung, dass Nutzer je nach Thematik (bzw. Suchanfrage) unterschiedlich viel Zeit brauchen, um zu identifizieren, ob die Webseite die gesuchte Information enthält⁴. Die Autoren lassen manuell Klickergebnisse auf Suchanfragen als (nicht) zufriedenstellend labeln und analysieren die unterschiedlichen Verteilungen von Aufenthaltsdauer je nach Label, beziehen aber auch andere Charakteristika wie Textlänge, Thema und Lesbarkeit

³Aufenthaltsdauer wird auch für die Optimierung von (personalisierter) Werbung eingesetzt: Etwa Barbieri et al. (2016) modellieren die Aufenthaltsdauer, die Nutzer auf den Zielseiten von Werbeanzeigen verbringen, die sie angeklickt haben.

⁴Intuitiv: Bei einer technischen oder medizinischen Fragestellung benötigt man mehr Zeit, um zu merken, ob die Seite die gewünschte Information enthält, als bei einer kurzen Wissensfrage oder beim Blick auf die Wettervorhersage.

der Webseite mit ein. So ermitteln sie schließlich flexiblere Grenzwerte und bauen ein Modell, das allgemein *Zufriedenheit* vorhersagt.

Auch C. Liu et al. (2011) modellieren die Aufenthaltsdauer auf Webseiten als Maß für die Nützlichkeit (*Usefulness*), dabei unterscheiden sie vier verschiedene Ziele (Tasks), die Nutzer in einer Laborstudie verfolgen sollten (zum Beispiel *Vorbereitung auf ein Interview* oder *Suche nach Hintergrundwissen*). Zusätzlich zu der Beobachtung der Aufenthaltsdauer (implizites Feedback), wurden die Probanden gebeten, nützliche Seiten zu speichern. Die Autoren finden einen Zusammenhang zwischen der Aufenthaltsdauer und dem Speichern einer Seite und werten dies als Hinweis darauf, dass Aufenthaltsdauer ein guter Indikator für Dokumentenrelevanz ist. Auch sie beobachten, dass die aufgabenabhängige Setzung von Grenzwerten sinnvoll sind.

C. Liu et al. (2010) betrachten das Nutzerverhalten bei Web browsing allgemein, also ohne eine konkrete Suchanfrage zu betrachten. Den Autoren liegen dabei die einzelnen Nutzerdaten pro Seite vor, sie modellieren die Verteilung der Aufenthaltsdauern der Nutzer auf Webseiten. Als Features verwenden sie neben vielen nicht-textbasierten Daten (etwa Dauer, die die Seite zum Laden braucht oder ihrer Größe) die Textlänge, die Textkategorie, sowie einfache Worthäufigkeiten.

Neben der Betrachtung von Klickverhalten (vor allem die *Click-Through-Rate*, CTR) hat sich also die Aufenthaltsdauer als Maß für Relevanz bzw. Zufriedenheit bei Websuche etabliert⁵.

Wie dieser Abschnitt gezeigt hat, ist es vor allem der Forschungsbereich von Websuche und Web browsing, der den Grundstein für die Verwendung von impliziten Maßen wie Klickzahlen und – für diese Arbeit relevanter – Aufenthaltsdauer als Proxy für Nutzerzufriedenheit und Dokumentenrelevanz gelegt hat. Natürlich ist die in der vorliegenden Arbeit verfolgte Fragestellung eine andere. Auch methodisch unterscheiden sich die beiden Themengebiete: Bei Websuchen spielt neben dem Dokument selbst die Suchanfrage (*Query*) eine entscheidende Rolle für die Bewertung von Relevanz. Neben der Textverarbeitung der Webseite muss also die Anfrage repräsentiert werden.

Nach diesem thematischen Umweg über den Bereich der Websuche als Grundlage für die Nutzung von Aufenthaltsdauer als Proxy für Zufriedenheit wird im Folgenden das Forschungsfeld betrachtet, dem sich die vorliegende Arbeit anschließt: Die Vorhersage der allgemeinen Popularität von Online-Nachrichten.

⁵Guo und Agichtein (2012) nehmen noch mehr implizite Maße (Mausbewegung und Scrollingverhalten) mit in ihre Betrachtung auf. Auch sie finden einen Zusammenhang zwischen Aufenthaltsdauer und expliziten Ratings der Nutzer bzgl. Interesse.

2.3 Popularität von Online-Zeitungsartikeln

Das Forschungsfeld der Analyse und Modellierung von Popularität von Online-Nachrichten hat im Zuge ihrer wachsenden Nutzung stark an Bedeutung und Interesse gewonnen (Arapakis et al., 2014, 2016; Davoudi et al., 2019; Hardt & Rambow, 2017; Hardt et al., 2018; Ambroselli et al., 2018; Tsagkias et al., 2009; Bandari et al., 2012; Omidvar et al., 2020; Fernandes et al., 2015; Reis et al., 2015; Rathord et al., 2019; Namous et al., 2018; Stokowiec et al., 2017). Manche Arbeiten untersuchen allgemein das Leseerlebnis in seinen unterschiedlichen Facetten (Analyse von Blickverhalten, Konzentration und Fokus in Laborstudien, oft im Rahmen von sehr ausführlichen Befragungen der Probanden). Andere widmen sich hingegen der Vorhersage von Popularität im engeren Sinne (implizite Engagement-Metriken) und sind damit näher an der vorliegenden Fragestellung. Aufgrund der schieren Menge an Arbeiten können hier nur einige davon betrachtet werden. Fokussiert wurden insbesondere solche, bei denen methodisch interessante Ansätze bezüglich des Artikeltextes selbst vorhanden sind⁶. Während meistens die Anzahl an Seitenaufrufen im Vordergrund steht, verwenden einige Arbeiten (zusätzlich) auch die Aufenthaltsdauer. Häufig wird nicht der gesamte Artikeltext in der Modellierung verwendet, sondern nur die Überschriften (oder der Teasertext). Bei der reinen Betrachtung von Klickzahlen ist dies nachvollziehbar, schließlich kann der Artikeltext erst *nach* dem Klick gelesen werden.

Als besonders relevant ist hier die Arbeit von Davoudi et al. (2019) zu nennen, die die Aufenthaltsdauer auf Zeitungsartikeln vorhersagen. Dabei fokussieren sie den Inhalt des Artikels und verfolgen somit eine sehr vergleichbare Fragestellung zur vorliegenden Arbeit. Bei ihrer Textrepräsentation extrahieren die Autoren Emotionen, Events und Entitäten aus den Artikeltexten und überführen diese in abstrakte Repräsentationen. Außerdem erstellen sie eine Tf-Idf-basierte Dokumentenrepräsentation, die als Eingabe in FFN-Layer verwendet wird. Beide Komponenten werden anschließend konkateniert und in die Ausgabeschicht zur Vorhersage eingegeben. Als Baseline verwenden Davoudi et al. (2019) außerdem Regressionsmodelle, die anhand der Topicverteilungen der Texte bzw. Doc2Vec-Dokumentenvektoren Vorhersagen treffen. Auch verschiedene DL-Modellarchitekturen (CNN, LSTM, FFN), die Wortvektoren (Word Embeddings) als Eingabe erhalten, werden als Baselines verwendet. Weder die Modelle, noch der verwendete Datensatz sind öffentlich zugänglich.

⁶Einige Arbeiten verwenden eine Vielzahl an (Meta-)Features, der Artikeltext selbst wird aber eher marginal behandelt. Verwendet werden unter anderem die Anzahl an Links, Fotos, der Anteil von Stopwörtern, mittlere Wortlänge, Häufigkeit bestimmter Schlagwörter, Polaritäts- oder Subjektivitätswerte.

Hardt und Rambow (2017) modellieren die Anzahl der Seitenaufrufe von Online-Nachrichtenartikeln anhand textbasierter Features. Das Problem wird in eine Klassifikation überführt, indem die Klickzahlen in prozentuale Bereiche eingeteilt werden. Dabei vergleichen die Autoren die Performanz ihrer Modelle, je nach verwendeter Textbasis (Titel und Teaser, oder Haupttext). Sie machen die Beobachtung, dass die Verwendung des gesamten Artikeltextes (statt nur Titel und/oder Teaser) die Modellvorhersagen verbessert, obwohl dieser dem menschlichen Leser zum Zeitpunkt des Klickens noch gar nicht bekannt ist. Als Erklärung folgern die Autoren: Die Thematik eines Artikels kann besser modelliert werden, je mehr Text zur Verfügung steht. Während ein erfahrener Zeitungsleser bereits aus einer kurzen Schlagzeile die Thematik antizipieren kann, tut sich ein Modell damit schwer.

Auch Hardt et al. (2018) widmen sich der Popularität (Seitenaufrufe) von Schlagzeilen als binärer Klassifikationsaufgabe. Dabei erweitern sie ihr DL-Modell (GRU) basierend auf vortrainierten Wortvektoren in einem Multi-Task-Aufbau mit zwei Hilfstasks bezüglich syntaktischer und semantischer Klassifikation, genauer der Erkennung von Wortarten (*Part-of-Speech-Tagging*), sowie der Einordnung in die Rubrik, aus der der Artikel stammt. Daneben wählen sie als Baseline eine logistische Regression mit Charakter-N-Grammen, sowie Uni- und Bigrammhäufigkeiten.

Ambroselli et al. (2018) wählen als Popularitätsmaß weder Klicks noch Aufenthaltsdauern, sondern die Anzahl von Kommentaren, die ein Artikel erhält. Auch sie überführen die Fragestellung in eine binäre Klassifikation (Fällt der Artikel in die 10% der Artikel mit höchster Kommentanzahl oder nicht?). Als Features kombinieren sie textuelle Daten (*Bag-of-Words*-Repräsentation, Keyword-Erkennung, Topic-Modeling, Doc2Vec) mit verschiedenen numerischen Metadaten (Autor, Genre, Wetter zum Erscheinungszeitpunkt).

Ebenso modellieren Tsagkias et al. (2009) die Anzahl an Kommentaren von Online-Zeitungsartikeln als Klassifikationsproblem. Neben der Textlänge verwenden die Autoren als Textfeatures die Häufigkeiten der 100 bedeutungstragendsten Tokens. Außerdem nutzen sie *Named-Entity-Recognition* (NER), extrahieren Orte, Personen und Organisationen und verwenden die jeweils 50 häufigsten Entitäten als Features. Auch Stokowiec et al. (2017) nehmen die Zahl an Kommentaren als Maß für Popularität und verwenden ein bidirektionales LSTM-Modell mit GloVe-Embeddings. Als Textbasis verwenden sie nur den Titel des Artikels.

Auch Bandari et al. (2012) sagen die Popularität von Artikeln vorher. Statt Klick- oder Kommentanzahl verwenden sie die Häufigkeit, mit der ein Artikel auf Twitter gepostet oder geteilt wird, als Kennzahl. Sie repräsentieren einen Artikel anhand mehrerer Features: die Nachrichtenquelle, die Artikelkategorie, ein ermittelter Subjektivitätswert, sowie *Named Entities*. Eine ähnliche Modellierung von der Zahl

an Weiterleitungen (*Shares*) führen auch Ren und Yang (2015) und Namous et al. (2018) durch, beide verwenden Textfeatures wie Sentiment- oder Subjektivitätswerte, Schlagworte und Topics und verwenden ansonsten vor allem textunabhängige Metadaten⁷.

Die bisher erwähnten Arbeiten zur Modellierung von Popularität von Zeitungsartikeln nutzen *ein* KPI-Maß (Aufrufe, Aufenthaltsdauer, Zahl an Kommentaren, Twitter-Weiterleitungen) als Maß für User Engagement. Vor allem bei Klickzahlen besteht die bereits mehrfach angesprochene Problematik, dass Klicks nur bedingt als Aussage über Zufriedenheit gewertet werden sollten. Einen interessanten Ansatz verfolgen stattdessen Omidvar et al. (2020), die sowohl Seitenaufrufe als auch Aufenthaltsdauer in Kombination betrachten und die Artikel anhand der beiden Maße in vier Qualitätskategorien einteilen (genauer: für jede Kategorie einen Wahrscheinlichkeitsscore berechnen)⁸. Ziel der Autoren ist also nicht generell die *Popularität* eines Artikels bzw. seiner Schlagzeile zu modellieren, sondern die *Qualität* einer Schlagzeile bezogen auf ihre Fähigkeit, den Artikel zu repräsentieren⁹. Methodisch basieren ihre Modelle daher vor allem auf der Ähnlichkeit zwischen Schlagzeile und Artikelinhalt. Beide Textteile werden mit fortgeschrittenen DL-Methoden einzeln in abstrakte Repräsentationen überführt. Sie nutzen dabei – wie die vorliegende Arbeit auch – das Transformermodell BERT. Parallel führen sie Topic Modeling (mit *Latent Dirichlet Allocation*, LDA) anhand von Tf-Idf-Features für beide Texte durch. Außerdem erstellen sie anhand von vortrainierten GloVe-Embeddings und Kosinusähnlichkeit eine Ähnlichkeitsmatrix zwischen Titel und Artikeltext, die sie in einer anschließenden CNN-Architektur weiterverarbeiten. Diese drei Bausteine (BERT-Repräsentationen, Topics, CNN-Ausgabe) werden in einem komplexen DL-Modell zusammengeführt, das eine finale Klassifikation in eine der vier Kategorien erstellt. Zum Vergleich ihres Modells verwenden die Autoren mehrere komplexe DL-Baselines, darunter FFNs,

⁷Die Wahl der Anzahl an Twitter-Teilungen als Zielvariable ist interessant. Im Gegensatz zu Klickzahlen spielt hier Clickbait eine kleinere Rolle: Nutzer entscheiden sich bewusst dazu, einen Artikel zu teilen. Gleichzeitig ist es aber möglich, dass Artikel geteilt, aber dabei kritisiert werden. Ob positiv oder negativ, als Maß für Engagement sind sie sicherlich geeignet.

⁸So labeln sie zum Beispiel *Clickbait*-Artikel (Artikel mit großer Zahl an Aufrufen, aber geringer mittlerer Aufenthaltsdauer) als *misleading*. Artikel mit geringer Zahl an Aufrufen, aber langen Aufenthaltsdauern sind zwar qualitativ gut, die Überschriften konnten aber nicht genug Interesse wecken. Artikel mit niedrigen Klickzahlen und geringer Aufenthaltsdauer sind wenig *engaging*. Der erwünschte Fall sind Artikel mit sowohl großer Zahl an Aufrufen, als auch langen Aufenthaltsdauern. Diese Artikel verfügen über eine interessante Schlagzeile, hinter der sich aber auch ein interessanter Artikel verbirgt. Neben der *harten* Klassifikation (eines der vier Labels/Gruppen) verwenden die Autoren *weiche* Vorhersagen, nämlich die einzelnen Wahrscheinlichkeitsscores der vier Labels (Regression). Letzteren Ansatz schlagen sie als ihr finales Modell vor.

⁹Auch Omidvar et al. (2018) beschäftigen sich mit der binären Klassifikation von Schlagzeilen und Artikeltexten als Clickbait. Hier liegen aber menschliche Ratings als Golddaten vor, das automatisierte Labeln der Artikel entfällt also.

CNNs, sowie RNNs in Kombination mit GloVe- oder Doc2Vec-Vektoren, sowie ein FFN mit Tf-Idf-Features.

Omidvar et al. (2020) verwenden dabei nur den *ersten* Abschnitt des Artikeltext und begründen dies damit, dass gerade dieser erste Teil eines Artikels dafür geeignet sei, den Gesamtinhalt zu repräsentieren. Zu diesem Schluss kommt auch Lopyrev (2015), der den ersten Absatz verwendet, um automatisiert eine Artikelüberschrift zu generieren. Interessant sind hier auch die Beobachtungen von Lagun und Lalmas (2016), die Lesemuster bei der Rezeption von Zeitungsartikeln untersuchen und dabei beobachten, dass häufig nur der erste Abschnitt eines Artikels gelesen wird. Diese Beobachtungen sind auch für die Modelle der vorliegenden Arbeit relevant und sollten im Hinterkopf behalten werden.

Neben dem hier fokussierten Bereich, der sich mit der *Popularität* von Artikeln bzw. User Engagement beschäftigt, gibt es auch Arbeiten, die stattdessen die *Qualität* eines Artikels betrachten und modellieren – zwei Konzepte, die nicht unbedingt gleichzusetzen sind. Einen guten Überblick bzw. Startpunkt bieten hier Arapakis et al. (2016), die ein Benchmark-Korpus zur Qualität von Online-Zeitungsartikeln bereitstellen.

In diesem Abschnitt wurden Arbeiten betrachtet, die sich der Modellierung der *allgemeinen* Popularität eines Zeitungsartikels widmen, hier diene also User Engagement mehrerer Nutzer bzw. Leser (Seitenaufrufe, Aufenthaltsdauern) als allgemeines Maß der Zufriedenheit mit einem Artikel. In dieses Feld ordnet sich auch die vorliegende Arbeit sowohl in Datengrundlage als auch Methodik ein.

Das im Folgenden behandelte Forschungsfeld (Personalisierte News Recommendation) verfolgt zwar eine andere Zielsetzung, aber auch hier spielt Aufenthaltsdauer auf Online-Artikeln als Maß für User Engagement eine wichtige Rolle. Außerdem bietet es vor allem aus methodischer Sicht als hochaktuelles Anwendungsgebiet einige interessante und moderne Ansätze der Textverarbeitung.

2.4 (Personalisierte) News Recommendation

Einen anderen Fokus setzen Arbeiten aus dem Bereich der (personalisierten) *News Recommendation*¹⁰. Gemeint sind Systeme wie etwa Google News oder Microsoft News, die gezielt bestimmte Artikel verschiedener Nachrichtenquellen sammeln und den Nutzern vorschlagen, meist beruhend auf aktuellen Geschehnissen und/oder spezifischer Nutzerpräferenz. Aus der großen Anzahl an grundsätzlich vorhandenen Artikeln müssen also besonders *relevante* Artikel erkannt werden.

¹⁰Der etablierte englische Begriff wird im Folgenden umständlichen Übersetzungen wie *Vorschlagssysteme* oder *Empfehlungssysteme* vorgezogen.

Bei dieser *personalisierten* News Recommendation geht es also darum, einem spezifischen Leser für ihn relevante und voraussichtlich lesenswerte Artikel vorzuschlagen bzw. die vorgeschlagenen Artikel nach der Leserpräferenz zu sortieren (*Ranking*). Zusätzlich zum Artikel (und seiner allgemeinen Popularität) müssen also der Nutzer und seine Interessen oder Präferenz modelliert werden (F. Wu et al., 2020; Yi et al., 2014; An et al., 2019; J. Chen et al., 2021; C. Wu, Wu, Ge et al., 2019). Konzeptionell beruht diese Modellierung meist auf der Betrachtung der bereits vom Nutzer gelesenen Artikel (*User/Click History*) und den zugehörigen KPI-Werten, und der Ähnlichkeit zu den Kandidaten-Artikeln¹¹. Das Vorliegen einzelner Nutzerdaten ist hier also unabdingbar, sowohl bei der Historie eines Nutzers (als Eingabe in die Modellierung), als auch – für das Training von Modellen und deren Evaluierung – bei der *Zufriedenheit* mit bereits vorgeschlagenen Artikeln¹². Diese Informationen über das (historische) Verhalten der einzelnen Nutzer sind in sogenannten Logdateien (Web-Logs) enthalten. Zur Erinnerung: Solche Individualwerte liegen in dem Datensatz der vorliegenden Arbeit nicht vor.

Die hier erwähnten Arbeiten unterscheiden sich in Zielstellung, Datengrundlage und Modellierung also stark von der vorliegenden Untersuchung, insbesondere jene, die die Nutzer-Modellierung in ihren Fokus stellen. Trotzdem bieten einige von ihnen interessante und sehr relevante Ansätze: Vor allem im Bereich der Textrepräsentation sind die Arbeiten aus diesem Bereich oft sehr viel aktueller und umfassender, der Einsatz von DL-Modellen und Transformern ist deutlich häufiger als in den Arbeiten zur allgemeinen Popularität von Artikeln, so etwa bei C. Wu et al. (2020); C. Wu, Wu, An, Huang et al. (2019); Wang et al. (2018); An et al. (2019); Okura et al. (2017); F. Wu et al. (2020); C. Wu et al. (2021).

Während auch im Bereich News Recommendation lange vor allem Klickdaten als Messung von Interesse verwendet wurden (J. Liu et al., 2010), hat sich in neueren Arbeiten die Verwendung von Aufenthaltsdauer (*Dwell Time*) statt oder meistens in Kombination mit Klickzahlen als Messung von Nutzerzufriedenheit etabliert. Auch diese Tatsache macht sie also für die vorliegende Arbeit attraktiv (C. Wu et al., 2020; Yi et al., 2014; C. Wu, Wu, Ge et al., 2019; C. Wu, Wu, An, Qi et al., 2019; C. Wu et al., 2021; J. Chen et al., 2021; C. Wu et al., 2021)¹³.

¹¹C. Wu, Wu, An, Qi et al. (2019) beziehen für die Modellierung des Interessenprofils eines Nutzers zusätzlich zu Klickhistorie und Inhalten der besuchten Seiten auch seine Suchanfragen ein.

¹²Verwendung findet vor allem die *Click-Through-Rate* (CTR) vergangener Vorschläge, also der Anteil der vorgeschlagenen Artikel, der tatsächlich erfolgreich war, also geklickt wurde.

¹³Auch Zhou et al. (2018) modellieren Klickzahlen in Kombination mit Aufenthaltsdauer, um Nutzerinteressen zu messen. Sie betrachten aber nicht spezifisch das Vorschlagen von Zeitungsartikeln, sondern allgemein an der Historie des Nutzers orientiertes Vorschlagen von Inhalten (etwa auch Werbung).

Yi et al. (2014) thematisieren die Problematik von Klickzahlen als Metrik für Zufriedenheit und verwenden als eine der ersten Arbeiten im Bereich News Recommendation zusätzlich zur Klickhistorie eines Nutzers auch die Aufenthaltsdauer, die er auf einer Seite verbringt, um seine Zufriedenheit oder Relevanz des Artikels zu erkennen. Den Artikeltext behandeln die Autoren bei der Modellierung der Aufenthaltsdauer allerdings nur spärlich, es wird lediglich das Topic (die Rubrik) des Artikels verwendet, zusätzlich zum Gerät (*device*, also Smartphone oder Computer), auf dem der Artikel gelesen wird.

Auch C. Wu et al. (2020) kombinieren Klickzahlen und Aufenthaltsdauer in ihrer Modellierung. Daher beziehen sie in ihre Nutzer-Modellierung sowohl das vergangene Klickverhalten des Nutzers ein, als auch die Aufenthaltsdauern auf den geklickten Webseiten¹⁴. Daraus ermitteln sie zwei Varianten von Nutzer-Embedding-repräsentationen, von denen eine die Präferenz für Schlagzeilen, die andere die Zufriedenheit mit gelesenen Inhalten modelliert. Diese Nutzerrepräsentation wird anschließend in ein Modell gegeben, das parallel zwei Trainingsaufgaben verfolgt und optimiert: Einerseits die Vorhersage der Klickwahrscheinlichkeit bei einer bestimmten Schlagzeile, andererseits die Vorhersage der Zufriedenheit mit dem Artikel selbst. Im Gegensatz zu bei Yi et al. (2014) ist die Textverarbeitung hier sehr ausführlich. Die Textrepräsentation von den geklickten Überschriften, sowie den gelesenen Artikeln erfolgt bei C. Wu et al. (2020) durch einen neuronalen Text-Encoder, der Wort- und Positionsempbeddings mit Multi-Head-Attention kombiniert. Die Autoren verwenden also nicht das in der vorliegenden Arbeit verwendete BERT-Modell, aber eine auf sehr ähnlichen Funktionalitäten basierende Encoder-Struktur¹⁵.

Auch An et al. (2019) verwenden einen Text-Encoder und einen Nutzer-Encoder. Der Text-Encoder besteht aus drei Teilen: Der Titel wird anhand einer CNN-Architektur mit anschließendem Attention-Layer repräsentiert. Außerdem werden Repräsentationen für Topic und Subtopic des Artikels gelernt (der verwendete Datensatz enthält entsprechende Labels). Die Artikelrepräsentation ist schließlich die Konkatenation dieser drei Vektoren. Der Haupttext des Artikels wird allerdings nicht beachtet, der Fokus der Arbeit liegt eher auf der Nutzer-Modellierung.

Eine hingegen sehr ausführliche Beschäftigung mit dem Artikeltext liegt bei F. Wu et al. (2020) vor, die ein großes (englisches) Korpus bereitstellen, das Online-Zeitungsartikel und Nutzer-Logdateien enthält. Zwar liegen hier ausschließlich Klick-

¹⁴Um der möglichen Abhängigkeit der Aufenthaltsdauer von der Textlänge gerecht zu werden, verwenden die Autoren dabei eine nutzerspezifische Lesegeschwindigkeit (*Reading Speed*), die sich aus Aufenthaltsdauer und Textlänge ergibt. So können sie gleichzeitig unterschiedliches Leseverhalten (Lesegeschwindigkeiten) der Nutzer abbilden. Das Problem von unvollständig gelesenen Artikeln wird allerdings nicht behandelt.

¹⁵Auch C. Wu, Wu, Ge et al. (2019) verfolgen ähnliche neuronale Attention-basierte Verfahren der Textrepräsentation.

daten (Klickverläufe der Nutzer) und keine Aufenthaltsdauern vor¹⁶, die vorgestellten Methoden zu Repräsentation der Artikeltexte sind aber sehr relevant für die vorliegende Arbeit. Die Autoren vergleichen zunächst anhand ihres Datensatzes verschiedene bestehende News-Recommendation-Systemarchitekturen. Anschließend widmen sie sich selbst der Artikelrepräsentation (CNN, LSTM, sowie Self-Attention-Mechanismen mit GloVe-Embeddings und schließlich auch BERT), setzen diese an geeigneter Stelle in die bestehenden Systeme ein und beobachten die Änderungen in der Vorhersageperformanz. Sie beobachten, dass das Einsetzen von (fixen) BERT-Embeddings als Textrepräsentation die Modellqualität bereits verbessert und schließlich *Fine-Tuning* von BERT die besten Vorhersagen liefert.

Wie dieser und der vorangehende Abschnitt gezeigt haben, ist die Textrepräsentation (also die Verarbeitung des Artikeltext) im Bereich der News Recommendation in der Regel deutlich fortgeschrittener und bedient sich neuerer Methoden (Attention, Transformer-Modelle) als in den Arbeiten, die sich mit der *generellen* Popularität von Zeitungsartikeln beschäftigen. Möglicherweise ist dies schlicht dadurch zu erklären, dass News Recommendation das neuere (und auf dem Bereich allgemeiner Popularität von Artikeln aufbauende) und aktuell sehr beliebte Feld ist. Jedenfalls wird hier eine deutliche methodische Lücke sichtbar. In dieser Lücke siedelt sich die vorliegende Arbeit an, indem sie den Artikeltext in den Fokus für die Modellierung von generellem User Engagement setzt und moderne Methoden zur Textrepräsentation nutzt.

2.5 Abgrenzung: Aufenthaltsdauer vs. Lesedauer

Es ist naheliegend, die Zeit, die eine Leserin auf einem Zeitungsartikel verbringt, als die Zeit zu interpretieren, die sie benötigt, um den Artikel zu lesen, also als die Lesedauer (*Reading Time*). Dieser Gleichsetzung wird hier – jedenfalls im Rahmen der vorliegenden Arbeit und ihrer Datengrundlage – ausdrücklich widersprochen.

Der hier zugrundeliegende Datensatz (genauer in Abschnitt 3) besteht aus Artikeltexten eines Online-Nachrichtenportals und zugehörigen KPI-Werten (wie Seitenaufrufe und Aufenthaltsdauer). Dabei sind zwei Punkte wichtig, die man im Auge behalten sollte:

Erstens handelt es sich pro Artikel um *aggregierte* Werte, die Aufenthaltsdauer ist also die gemittelte Aufenthaltsdauer über *alle* Besuche. Daten über einzelne Leser und ihr Verhalten liegen nicht vor.

¹⁶In einer anschließenden Arbeit (C. Wu et al., 2021) verwenden die Autoren zusätzlich zu Klickzahlen eine Vielzahl an weiteren Maßen, neben impliziten (Aufenthaltsdauer) auch explizite (*Likes* und *Shares*). Auch hier ist die Textrepräsentation sehr fortgeschritten.

Zweitens lassen die Daten keinerlei Wissen darüber zu, in welchem Umfang die Artikel von den Seitenbesuchern auch tatsächlich gelesen wurden. Es kann also nicht davon ausgegangen werden, dass die Artikel *vollständig* gelesen und nicht vorzeitig abgebrochen wurden. Gleichzeitig ist es möglich, dass Nutzer den Artikel zwar geöffnet hatten, sich aber nicht durchgehend mit ihm beschäftigten. Hinzu kommt die Problematik, dass Online-Zeitungsartikel deutlich mehr bieten, als nur den reinen Artikeltext: Fotos (teilweise auch Videos), Werbeanzeigen, Bildunterschriften und Kommentarspalten – alle diese Inhalte werden potenziell vom Nutzer während seines Aufenthalts betrachtet. Bereits die Aufenthaltsdauer *eines* Nutzers ist also keineswegs seine *Lesezeit*, gleiches gilt für die vorliegende gemittelte Aufenthaltsdauer.

Dies ist aber auch gar nicht der Anspruch der Arbeit. Die Aufenthaltsdauer wird hier als Maß für User Engagement bzw. Artikelpopularität verstanden, wie es auch andere bereits beschriebene Arbeiten vormachen. Die Tatsache, dass Leser einen Artikel nach kurzer Zeit abbrechen, oder aber ihn interessiert und vollständig lesen, vielleicht Fotos und Kommentare betrachten und sich insgesamt stark mit ihm beschäftigen, ist dabei genau das, was im Sinne von *User Engagement*, *Interesse* oder *Popularität* abgebildet werden soll.

Das Ziel dieser Arbeit ist also nicht zu verwechseln mit der Arbeit von Weller et al. (2020), die sich tatsächlich mit der *Lesezeit* von Zeitungsartikeln beschäftigen. Diese Arbeit der Autoren stellt dabei eine Ausnahme dar, die Lesezeit-Forschung modelliert meist Lesezeit auf Wort- und nicht auf Textebene (Weller et al., 2020; van Schijndel & Schuler, 2015; van Schijndel & Linzen, 2018). Bei der Untersuchung von Weller et al. (2020) handelt es sich um eine Studie, bei der die Probanden extra instruiert wurden, den Artikel vollständig zu lesen und auch nur der reine Artikeltext (und keine Fotos oder andere Elemente) selbst vorlag. Hier kann also tatsächlich von der Lesedauer ausgegangen werden. Die Autoren verwenden neben einfachen Baselines (Regression anhand der Textlänge; Faustregel einer durchschnittlichen Lesegeschwindigkeit von 240 Wörtern pro Minute) Dokumentenrepräsentationen mit XLNet, roBERTa und ELMO, die sie mit manuell erstellten Lesbarkeitsmetriken (unter anderem Schwierigkeitsgrad, Schriftgröße, Bekanntheitsgrad einzelner Tokens, Anteil an Verben) kombinieren. Sie beobachteten aber, dass tatsächlich schlicht die Textlänge das indikativste Merkmal zur Vorhersage der Lesezeit darstellt.

In Bezug auf das Problem, dass Aufenthaltsdauer nicht unbedingt mit *Lesedauer* gleichzusetzen ist, bieten Lagun und Lalmas (2016) eine interessante Betrachtung. Sie beschäftigen sich mit dem Leseverhalten von Nutzern auf Online-Zeitungsportalen und verwenden zusätzlich zur Messung der allgemeinen Aufenthaltsdauer Viewport-Daten, also die Registrierung des zu jedem Zeitpunkt sichtbaren Ausschnitts des Artikels. So können sie besser nachvollziehen, welche Teile des Artikels und in

welcher Reihenfolge tatsächlich gelesen werden (sie wählen dafür die Bezeichnung *User Attention*). Die Autoren identifizieren dabei zunächst verschiedene Muster im Leseverhalten¹⁷ und konstatieren anschließend anhand des Anteils am gelesenen Text vier unterschiedliche Engagement-Levels: *Bounce*, *Shallow*, *Deep* und *Complete Engagement*. In sekundären Analysen zur Aufenthaltsdauer beobachten die Autoren zwar einen positiven Zusammenhang zwischen der mittleren Aufenthaltsdauer und dem (steigendem) Engagement-Level, trotzdem sprechen sie sich dagegen aus, die reine Aufenthaltsdauer bei Artikeln als Maß für Engagement zu verwenden¹⁸. Mithilfe von Topic Modeling modellieren sie schließlich die Engagement-Levels.

Natürlich kann trotz dem Verständnis als Aufenthalts- statt Lesedauer die unterschiedliche Artikellänge ein Problem darstellen, da sie die Aufenthaltsdauer – wenn der Artikel in großen Teilen gelesen wird – beeinflusst. In Abschnitt 3.3 wird der Zusammenhang zwischen Textlänge und Aufenthaltsdauer im Datensatz genauer betrachtet. Eine gewisse positive Korrelation ist zwar vorhanden, sie ist aber weniger stark, als man vermuten könnte. Diese „fehlende“ oder geringe Korrelation zwischen Textlänge und Aufenthaltsdauer beobachten auch andere Arbeiten (Lagun & Lalmas, 2016; Homma et al., 2018, 2020; Omidvar et al., 2020; Morita & Shinoda, 1994; Claypool et al., 2001). Yi et al. (2014) hingegen beobachten einen Zusammenhang zwischen Textlänge und Aufenthaltsdauer, der allerdings bei langen Artikeln schwächer ausfällt. Interessant ist in diesem Zusammenhang auch die Analyse von Aufenthaltsdauern bei Nachrichtenartikeln von Seki und Yoshida (2018). Sie beobachten in verschiedenen Rubriken bzw. Textkategorien einen unterschiedlich stark ausgeprägten Zusammenhang zwischen Textlänge und Aufenthaltsdauer (bei Artikeln über Politik ist der Zusammenhang stärker als bei solchen über Technik), auch C. Liu et al. (2010) machen ähnliche Beobachtungen. Diese Unterschiede in der Korrelationsstärke könnten ein Hinweis darauf sein, dass sich erstens bereits sehr früh (etwa nach den ersten Sätzen) entscheidet, ob ein Nutzer den Artikel weiterliest oder abbricht und zweitens, Artikel unterschiedlicher Kategorien unterschiedliche Lesemuster (Abbrechen, schnelles Überfliegen oder vollständiges Lesen) anziehen.

¹⁷Ein Muster ist etwa das Lesen des ersten Abschnittes mit anschließendem Verlassen der Seite, ein anderes besteht darin, den Artikel konsekutiv zu lesen und am Ende wieder zum Anfang zu scrollen.

¹⁸Die Autoren versuchen in einer sekundären Analyse, anhand der Aufenthaltsdauer und entsprechenden Grenzwerten, das Engagement-Level vorherzusagen. Die Tatsache, dass dies nicht gut funktioniert, werten sie als Zeichen dafür, dass die Aufenthaltsdauer keine große Aussagekraft über User Engagement hat.

3 Datengrundlage

Die Datengrundlage zur Betrachtung und Modellierung der Popularität von Online-Zeitungsartikeln – ein Korpus bestehend aus Artikeltexten und einigen zugehörigen KPI-Werten (*Key Performance Indicator*) – wurde eigens im Rahmen dieser Arbeit angefertigt. Es handelt sich bei den KPI-Werten ausschließlich um *aggregierte* bzw. gemittelte Werte pro Artikel, die Daten der einzelnen Nutzer bzw. Besuche liegen nicht vor.

Die Bereitstellung der KPI-Tabellen und der originalen HTML-Dateien mit den Artikelinhalten verdankt diese Arbeit der Data Science Firma *INWT Statistics* (<https://www.inwt-statistics.de>) und speziell Dr. Steffen Wagner, der auch sämtliche Kommunikation mit den Verantwortlichen bei den Zeitungen übernommen hat. Ohne diese Datengrundlage wäre die vorliegende Arbeit nicht möglich gewesen.

Dieser Abschnitt beschreibt zunächst (3.1) die Schritte zur Beschaffung und Zusammenstellung des vollständigen Datensatzes (rund 100000 Dokumente). Eingeflossen sind Artikel aus vier Online-Nachrichtenportalen. Während die KPI-Werte (generiert mithilfe einer Tracking-Software) bereits aufbereitet vorlagen, mussten für die Beschaffung der Artikeltexte selbst einige Schritte durchlaufen werden.

Im Anschluss (3.2) wird der so entstandene Datensatz anhand einiger deskriptiver Statistiken beschrieben. Für die tatsächliche Modellierung der Aufenthaltsdauer wurde anschließend ein Teil dieses Datensatzes (eines der Nachrichtenportale, 36383 Artikel) ausgewählt, dessen Eigenschaften in Abschnitt 3.3 beschrieben werden.

3.1 Korpuserstellung: Beschaffung der Artikeltexte

Der Datensatz umfasst Zeitungsartikel von den Online-Auftritten vier deutscher Tageszeitungen. Es handelt sich dabei um die [REDACTED], den [REDACTED], die [REDACTED], und den [REDACTED].

Für alle vier Zeitungen lagen die KPI-Werte bereits in tabellarischer Form vor (bereitgestellt von *INWT Statistics*), die mit der Tracking-Software Google Analytics von den Seiten der Nachrichtenportale pro Artikelseite erhoben wurden. Eine Beschreibung der einzelnen KPI-Werte (etwa Seitenaufrufe, Aufenthaltsdauer) erfolgt im Rahmen der genaueren Beschreibung des ausgewählten Datensatzes (in Abschnitt 3.3).

Die Artikelinhalte (Artikeltexte) selbst lagen hingegen zunächst nicht vor und wurden erst im Rahmen dieser Arbeit mithilfe von HTML-Parsing beschafft. In einer ersten und kleineren Version des Datensatzes lagen zunächst zwar die ursprünglichen dpa-Meldungen als Texte vor, auf denen die Artikel beruhen. Die tatsächlich auf den

Portalen erschienenen Artikel unterscheiden sich aber teilweise sehr stark von diesen ursprünglichen Meldungen, sowohl in Umfang als auch Stil. Bei der Modellierung der Aufenthaltsdauer führte dies zu einigen Problemen, da KPI-Werte von nahezu unbekannten Texten (und damit auch unbekannter Länge!) vorhergesagt werden sollten.

Aufgrund dieser Erfahrungen wurde beschlossen, dass die *echten* Artikeltexte der Portale für die Modellierung unabdingbar sind. Daher wurden mittels HTML-Crawling die Seiteninhalte der Artikel eingeholt (dies übernahm *INWT Statistics*) und anschließend die HTML-Dateien ausgeparst, um die Artikeltexte zu erhalten (dies erfolgte durch die Autorin der Arbeit). Im Zuge dieser Änderung konnte der Datensatz – ursprünglich ausschließlich aus dpa-Artikeln bestehend – auf eine größere Zahl von Artikeln ausgeweitet werden.

Die Artikelinhalte lagen anschließend in Form von HTML-Dateien vor, die Zuordnung zwischen Artikel und KPI-Werten ermöglicht eine innerhalb einer Zeitung eindeutige Artikel-ID. Um aus den HTML-Dateien den Artikeltext als Textbasis zu erhalten, wurde der Seiteninhalt ausgeparst¹⁹. Anschließend wurde die Struktur nach textenthaltenden Abschnitten durchsucht. Da sich Tag-Struktur und -Verwendung bei den vier Nachrichtenportalen teilweise unterscheiden, mussten hier pro Zeitung einige Anpassungen vorgenommen werden. Der Vorgang wird hier zusammenfassend beschrieben. Dabei wurden also sowohl Überschriften, als auch Textabsätze, aber auch etwa Listen oder sonstige Textfelder aufgenommen und deren Inhalte konkateniert²⁰.

Bei einer stichprobenartigen Durchsicht der Artikel fielen einige problematische Artikel auf. Gemeint sind Artikel, die ein Video, Tweet-Sammlungen, Fotostrecken oder auch Audiobeiträge (Podcastfolgen) enthalten. Diese Elemente wirken sich mutmaßlich sehr stark auf das Nutzerverhalten (und spezifisch die Aufenthaltsdauer) aus, eine Modellierung anhand des reinen Textes erscheint hier wenig sinnvoll. Daher wurden während des Textparsings einige Heuristiken eingebaut, um betroffene Artikel entsprechend kennzeichnen und anschließend aus dem Datensatz entfernen zu können. Geachtet wurde dabei etwa auf Verlinkungen auf die Video-Plattform YouTube,

¹⁹Verwendet wurde dafür die Python-Bibliothek BeautifulSoup: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

²⁰Eine Besonderheit stellen Links dar, etwa zu anderen Artikeln oder weiterführenden externen Webseiten. Hier gibt es zwei Unterscheidungen. Manchmal sind normale Teile des Textes (etwa Namen oder Firmen) mit einem Link hinterlegt, hier sollte der Text normal beibehalten werden. Andererseits gibt es eigenständige Links (*Lesen Sie hier weiter zum Thema.* oder *Kennen Sie schon unseren Wochenrückblick?*), deren Text nicht mit aufgenommen werden sollte. Hier wurde heuristisch anhand des Anteils des Linktextes am Abschnitt vorgegangen. Reine Links, die nicht Teil des Textes waren, wurden somit gelöscht. Auch entfernt wurden Absätze, die fehlerhafte Artefakte oder Werbeanzeigen vermuten lassen, wie zum Beispiel durch Schlagwörter wie *Debug Info* oder Absätze, die mit *Anzeige* beginnen.





das Vorkommen anderer bekannter Videoplayer, oder andere Schlagwörter, die auf eingebundene externe Inhalte hinweisen. Es handelt sich hier um ein pragmatisches Aussieben von problematischen Artikeln, Vollständigkeit kann nicht gewährleistet werden. Vereinzelt wurden weiterhin Artikel anderer Sprachen (Englisch, Spanisch) entdeckt. Mithilfe einer automatischen Sprachprüfung²¹ wurden diese aus dem Datensatz entfernt.

Der Artikeltext besteht in der Regel aus drei Teilen: Titel, Teaser (Vorspann, *Lead*) und Haupttext, die prinzipiell getrennt verarbeitet werden könnten. In den Experimenten wurde sich aber für die simple Konkatenation dieser drei Teile entschieden, die hier als *Artikeltext* bezeichnet wird. Die Vorverarbeitung war nur minimal, es wurden lediglich Zeilenumbrüche beseitigt und eventuelle mehrfache Leerzeichen in einzelne überführt. Entfernt wurden Artikel, bei denen nach dem HTML-Parsing kein Artikeltext vorlag, meist aufgrund von korumpierter oder unsystematischer Verwendung der HTML-Tags in den Originaldateien

Anhand der eindeutigen ID-Nummern der Artikel konnte anschließend der Artikeltext mit den KPI-Werten zusammengeführt und damit der Datensatz zusammengestellt werden.

3.2 Der vollständige Datensatz, Entscheidung für

Der erstellte komplette Datensatz umfasst 107159 Artikel aus den vier Nachrichtenportalen der Zeitungen. Generell wurden nur Artikel in den Datensatz aufgenommen, die eine Mindestanzahl von 50 verwertbaren Seitenaufrufen verbuchen konnten. Dies dient dazu, dass die über alle Aufrufe aggregierten (gemittelten) KPI-Werte weniger anfällig für einzelne Ausreißer sind. Anhand der Artikeltexte und der KPI-Werte konnten einige sekundäre Maße berechnet werden, wie die Artikellänge (in Tokens) und die Korrelation dieser Textlänge mit der Aufenthaltsdauer.

Tabelle 1 liefert einige Eckdaten über den Datensatz, aufgeteilt in die vier Subkorpora (mit den Kürzeln , , , ), sowie in der letzten Zeile (gesamt) zusammenfassend über alle vier. In den Spalten dargestellt sind die enthaltene Anzahl an Dokumenten, also Artikeln, (N), Mittelwert sowie Median der Seitenaufrufe (Klicks), Mittelwert und Standardabweichung der Artikellänge in Tokens²², sowie Mittelwert, Standardabweichung und Median der durchschnittlichen Aufenthaltsdauer in Sekunden, die Nutzer auf einem Artikel verbringen. Außerdem sind zwei Korrelationsmaße aufgeführt, die den Zusammenhang zwischen der Artikellänge und der mittleren Aufenthaltsdauer beschreiben: Der Pearson-Korrelationskoeffizient r be-

²¹Verwendet wurde der *LanguageDetector* aus der Spacy-Boibliothek.

²²Interpunktionszeichen wurden nicht mitgezählt.

schreibt lineare Zusammenhänge, der Spearman-Korrelationskoeffizient ρ beschreibt allgemein monotone Rangzusammenhänge.





Publisher	N	Klicks		Tokens		Aufenthaltsdauer			Korrelation	
		Mean	Median	Mean	SD	Mean	SD	Median	r	ρ
	36383	1298.7	364	461.9	481.9	185.85	131.64	164.76	0.26	0.52
	17023	2329.5	444	551.2	323.3	148.99	51.89	144.84	0.31	0.34
	32272	1147.3	268	468.1	260.1	57.2	40.98	42.38	0.06	0.08
	21481	1353.6	232	490.9	295.1	51.35	36.76	41.57	0.09	0.09
gesamt	107159	1427.9	312	483.8	366.4	114.29	103.89	98.97	0.18	0.15

Tabelle 1: Überblick über den vollständigen Datensatz.

Wie in Tabelle 1 deutlich wird, unterscheiden sich die Datensätze der verschiedenen Zeitungen in manchen Werten stark. Für die genauere Betrachtung und Modellierung wurde entschieden, allein den **FAZ**-Datensatz zu verwenden (daher der Fettdruck in Tabelle 1), um Inkonsistenzen zwischen den Zeitungsportalen zu vermeiden.

Die **FAZ** (**Frankfurter Allgemeine Zeitung**) bildet mit 36383 Artikeln den größten Anteil (ein gutes Drittel) an der Gesamtmenge der vorliegenden Daten, gefolgt von der **WZL** (**Westfälische Zeitung**). Die durchschnittliche Zahl an Artikelaufrufen variiert, was aufgrund der unterschiedlich großen Verbreitungsgebiete und Lesergemeinschaften der Zeitungen nicht überraschend ist. Hier erreichen Artikel aus dem **FAZ** (**Frankfurter Allgemeine Zeitung**) deutlich mehr Leser oder Leserinnen, die **FAZ** steht – je nach Betrachtung von Durchschnitt oder Median – an zweiter oder dritter Stelle in Bezug auf diese Reichweite. Dies wäre zwar zunächst kein Grund, sich gegen eine gesammelte Modellierung der Aufenthaltsdauer zu entscheiden, ist aber bereits ein Hinweis darauf, dass sich das Leserverhalten bei den verschiedenen Zeitungen unterscheiden könnte. Doch auch die mittlere Textlänge der Artikel unterscheidet sich zwischen den Zeitungen. Bei der **FAZ** beträgt sie durchschnittlich 462 Tokens pro Artikel, ähnlich wie bei **WZL** und **WZ** (**Westfälische Zeitung**), die **FAZ**-Artikel fallen im Durchschnitt länger aus. Diese Tatsache kann für die Modellierung von Leseverhalten – spezifisch die Zeit, die Leser und Leserinnen auf Artikeln verbringen – durchaus problematisch sein, da sich die Artikellänge mutmaßlich auf die Aufenthaltsdauer auswirkt. In der Tat ist dieser Zusammenhang in den vorliegenden Daten aber geringer, als man vermuten könnte. Obwohl die durchschnittliche Artikellänge der Zeitungen nur geringfügig unterschiedlich ist, fällt die durchschnittliche Aufenthaltsdauer (ebenso wie der Median) bei den Artikeln der **FAZ** deutlich größer aus. Während Nutzer bei der **WZL** und **WZ** im Durchschnitt eine knappe Minute auf Artikeln verbringen, liegt der Mittelwert bei der **FAZ** bei durchschnittlichen 3 Minuten (185.85 Sekunden). Das **FAZ**-Korpus (das die durchschnittlich längsten

Artikel enthält) liegt dazwischen bei ca. 2 bis 3 Minuten. Diese Inkonsistenzen sind ein starker Hinweis dafür, dass neben der Textlänge wohl andere Faktoren großen Einfluss auf die Aufenthaltsdauer haben und – und das ist für die Entscheidung für die ■■■ als Modellierungsgrundlage zentral – sich diese unbekannten Faktoren zwischen den Zeitungen möglicherweise stark unterscheiden bzw. unterschiedlich stark Auswirkungen haben.

Die letzten beiden Spalten – die Korrelation von Artikellänge und Aufenthaltsdauer – verstärken diesen Eindruck. Auffällig ist, dass bei ■■ und ■■ tatsächlich kaum ein Zusammenhang festgestellt werden konnte, bei der ■■■ und bei ■■■ immerhin ein schwacher bis mäßiger, hier tendieren längere Texte zu höheren Aufenthaltsdauern. Die Spearman-Korrelation fällt dabei größer aus als die Pearson-Korrelation. Dies zeigt, dass die Aufenthaltsdauer nicht unbedingt linear mit der Textlänge steigt (also ein doppelt so langer Text nicht unbedingt eine doppelt so lange Aufenthaltsdauer aufweist), aber dennoch ein positiver Zusammenhang der beiden Größen besteht. Eine Betrachtung der Standardabweichungen der Verteilungen der Aufenthaltsdauer pro Artikel unterstützt diese Beobachtung, bei ■■ und ■■ streuen die Werte weniger als bei ■■■ und ■■■. Bei ■■ und ■■ sind also die Aufenthaltsdauern der Artikel im Vergleich mit ■■■ und ■■■ konstanter und hängen nicht mit der Textlänge zusammen.

Gründe für diese Unterschiede können hier nur als Vermutungen aufgezeigt werden, die anhand einer stichprobenartigen manuellen Durchsicht einiger Artikel aufgestellt wurden²³. Ein generelles Problem ist, dass im Datensatz allein der Text des Artikels vorliegt. Informationen über die Anzahl an Bildern, Videos, Links, Werbeanzeigen oder andere Inhalte der Seite liegen nicht vor. Gerade das Auftreten von Fotos und Videos kann sich mutmaßlich sehr stark auf die Aufenthaltsdauer auswirken. Unter anderem aus diesem Grund wird in dieser Arbeit die Bezeichnung *Aufenthaltsdauer* der *Lesedauer* oder *Lesezeit* vorgezogen, vgl. Abschnitt 2.5. Tatsächlich enthalten gerade die ■■- und ■■■-Artikel sehr häufig Videobeiträge. Während einige dieser Videos den Artikel thematisch ergänzen oder zusammenfassen, enthalten andere Artikel Videos, bei denen kein erkennbarer thematischer Zusammenhang zum Artikel vorliegt²⁴. Viele der Videos haben eine ähnliche Länge, dies könnte eine mögliche Erklärung für die fehlende Korrelation zwischen Textlänge und Aufenthaltsdauer sein.

²³Es konnten nur diejenigen Artikel genauer inspiziert werden, die zu dem Zeitpunkt noch online verfügbar waren, da die vorliegende HTML-Basis nur bedingt Rückschlüsse auf Elemente wie Fotos und Videos zulässt.

²⁴Ob dies bereits zum Erscheinungszeitpunkt des Artikels der Fall war, oder erst später andere/aktuellere Videos geschaltet werden, und wie dies zustande kommt, ist nicht bekannt.

Diese Problematik scheint bei den Artikeln der ■■■■ und ■■■■ weniger ausgeprägt zu sein. Es fand sich keine Möglichkeit, verlässlich die Artikel mit Videoinhalt herauszufiltern, ihr Anteil scheint aber deutlich geringer, wie die manuelle Durchsicht ergab. Gleichzeitig unterstützt auch die bei diesen beiden Zeitungen vorhandene Korrelation zwischen Textlänge und Aufenthaltsdauer die Annahme, dass hier tatsächlich der Artikeltext eine wichtigere Rolle spielt. Trotzdem gibt es natürlich weitere Faktoren, die sich auf die Aufenthaltsdauer auswirken können, wie etwa Fotos oder Werbeanzeigen, oder auch Schriftgröße und -stil. Die automatische Unterscheidung von Artikeln mit und ohne Videoinhalte hat sich im Rahmen dieser Arbeit als nicht zuverlässig durchführbar herausgestellt. Daher wurde sich generell gegen die Verwendung von Artikeln aus ■■■■ und ■■■■ entschieden, schließlich steht der Artikeltext im Mittelpunkt der Modellierung.

Ein weiterer Grund für die Fokussierung auf *einen* Korpusteil (eine Zeitung) ist, dass sich die Mittelwerte und Standardabweichungen der Zielvariablen (Aufenthaltsdauer) in den Zeitungen stark unterscheiden (siehe Tabelle 1), was eine gemeinsame Modellierung erschwert. In Entwicklungsexperimenten auf Daten aller Zeitungen wurde beobachtet, dass die Modelle sich darauf spezialisierten, anhand von spezifischen Floskeln oder häufigen Autoren- oder Ortsnamen die Zeitung zu erkennen und dementsprechend ihre Vorhersagen zu treffen, statt auf den Artikeltext als solchen zu achten²⁵. Dies führt zwar über alle Datensätze betrachtet nicht unbedingt zu schlechten Vorhersagen, bildet aber die Unterschiede innerhalb eines Datensatzes nur schlecht ab und geht am eigentlichen Sinn der Fragestellung vorbei. Hier würde gegebenenfalls eine Normierung der Aufenthaltsdauer innerhalb der Zeitungen Abhilfe schaffen, aber aufgrund der anderen genannten Gründe wurde die Fokussierung auf einen Publisher vorgezogen.

Die Entscheidung zwischen ■■■■ und ■■■■ fiel aufgrund des insgesamt größeren Umfangs des ■■■■-Teils. Eine Ausweitung und weitere Beschäftigung mit den übrigen drei Teilen des Datensatzes ist auf jeden Fall ein Anknüpfungspunkt für zukünftige Arbeiten.

Das endgültig verwendete Korpus umfasst somit die 36383 Artikel der ■■■■, bestehend aus den Artikeltexten und verschiedenen zugehörigen KPI-Werten, die im Folgenden genauer beschrieben werden.

²⁵Gemeint sind Formulierungen, die sich in vielen oder allen Artikeln einer Zeitung, jedoch weniger in denen einer anderen finden, wie etwa *Hat Ihnen dieser Artikel gefallen? Dann lesen Sie auch ...*, oder *Klicken Sie hier für ein ■■■■-Abonnent*, oder auch die Verlinkung anderer Artikel, deren Domain die Zeitung offenbart. Häufig steht auch der Autorenname und ein bestimmter Ortsname zwischen Titel und Teasertext.

3.3 Der [REDACTED]-Datensatz

Die 36383 Artikel, die den verwendeten [REDACTED]-Datensatz bilden, stammen vom Nachrichtenportal [REDACTED], dem Online-Auftritt der Tageszeitung [REDACTED] ([REDACTED]) und wurden 2019 bzw. 2020 veröffentlicht. Die gedruckte Version der [REDACTED] hatte im Jahr 2020 eine Auflage von ca. 60000 verkauften Exemplaren, das Online-Portal erzielte 2020 etwa 9 Millionen Besuche pro Monat.

Tabelle 2 gibt einen Überblick über den Datensatz, einige beschreibende Werte und die drei für die Modellierung erstellten Datensplits (train, dev, test).

Bevor allerdings näher auf die Daten (Klickzahlen, Aufenthaltsdauer, sowie die Datensplits) in Tabelle 2 eingegangen wird, muss zunächst ein Überblick über die Datenstruktur und die Bedeutungen der einzelnen KPI-Werte gegeben werden. Tabelle 3 auf Seite 33 gibt einen Überblick über die Struktur des Datensatzes. Aufgelistet sind die datensatzinternen Bezeichnungen einiger KPIs (die den Bezeichnungen von Google Analytics entsprechen) und struktureller Textdaten, deren im Text synonym verwendete Bezeichnungen, sowie jeweils eine kurze Beschreibung. In der letzten Spalte ist für illustrative Zwecke und zum besseren Verständnis der jeweilige Eintrag eines Beispielartikels (mit der Artikel-ID [REDACTED]) angegeben.

Split	N	Seitenaufrufe		Tokens		Dauer		r
		Mean	Median	Mean	SD	Mean	SD	
train	29106	1322.89	364	462.07	518.27	185.81	132.74	0.25
dev	3638	1220.62	366.5	468.2	301.52	189.77	139.71	0.39
test	3639	1183.33	363	454.21	287.77	182.29	113.06	0.4
gesamt	36383	1298.7	364	461.9	481.94	185.85	131.64	0.26

Tabelle 2: Überblick über Trainings-, Entwicklungs- und Testdatensatz der [REDACTED].

Wichtige KPI-Werte zur Beschreibung der Performanz einer Webseite (hier spezifisch eines Artikels) sind Angaben über das Klickverhalten der Nutzer. Neben der allgemeinen Zählung *aller* Seitenaufrufe (*pageviews*) liegen Angaben darüber vor, wie oft die Seite als Einstieg in die Sitzung eines Nutzers verwendet wurde (*entrances*). Unter einem Einstieg ist gemeint: Eine Nutzerin gelangt beispielsweise über eine Suchanfrage bei Google, über den Vorschlag eines Nachrichtenfeeds, oder über einen Link zu dem Artikel, befand sich also vorher noch nicht auf anderen Artikeln des Portals. Der umgekehrte Fall ist die Zählung der Seitenausstiege über den Artikel (*exits*): Der Nutzer betrachtet den Artikel als letztes und beendet danach die Session. Beides (*entrances* und *exits*) sind in der Zählung der *pageviews* enthalten. Ein weiterer Spezialfall wird extra erfasst: Unter *bounces* versteht man *single-page*-Besuche. Der Nutzer öffnet also – ohne vorher bereits auf anderen Artikeln des

Portals gewesen zu sein – einen Artikel und schließt ihn, ohne einen weiteren Besuch bei anderen Artikeln des Portals vorzunehmen.

Neben diesen Angaben, die sich dem Klickverhalten der Nutzer bezüglich eines Artikels widmen, wird die Aufenthaltsdauer auf der spezifischen Seite gemessen. Erhoben wird diese Zeit durch die Differenz zweier Zeitstempel, die zeitgleich zu den Klicks des Nutzers erfasst werden. Beim ersten Klick (Laden des Artikels) wird der Zeitpunkt mit einem Zeitstempel erfasst. Klickt der Nutzer auf einen weiteren Artikel, wird dieser Zeitpunkt des Ladens des nächsten Artikels erneut als Zeitstempel erfasst. Als Aufenthaltsdauer wird dann die Differenz beider Zeitstempel verwendet. Klickt er noch einen dritten Artikel an, so wird wieder ein Zeitstempel gesetzt. Zu beachten ist, dass bei dieser Vorgehensweise von Google Analytics jeweils für den letzten Artikel einer Session (also ein Aufruf, der unter *exits* fällt) keine Aufenthaltsdauer gemessen werden kann, da das Verlassen des Nutzers durch eine „fremde“ Seite (etwa durch Schließen des Browsers, aber auch Sprung zu einem anderen Seitenanbieter) nicht mit einem Zeitstempel registriert wird und folglich keine Differenz gebildet werden kann. Der beispielhafte Sessionverlauf in Abb. 1 verdeutlicht dies: Eine Nutzerin gelangt über einen Newsfeed auf den ersten Artikel im Portal (*entrance*). Danach navigiert sie per Klick noch auf einen zweiten und dritten Artikel, bevor sie dann das Zeitungsportal verlässt und per URL zu einer Videoplattform wechselt. Für Artikel 1 und 2 können anhand der registrierten Zeitstempel ihre Aufenthaltsdauern ermittelt werden, für Artikel 3 (*exit*) hingegen nicht, da kein zweiter Zeitstempel vorliegt. Alle drei Artikel registrieren ihren Besuch als Seitenaufruf (*pageview*), aber nur für Artikel 1 und 2 fließen die Zeiten der Nutzerin mit ein.

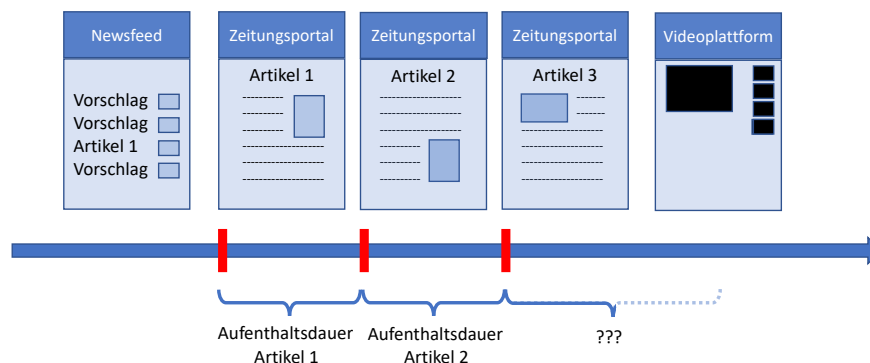


Abbildung 1: Beispielsession eines Nutzers, Ermittlung der Aufenthaltsdauern.


Der Wert ***timeOnPage*** ist die aufsummierte Aufenthaltsdauer, die Nutzer auf einer bestimmten Seite (einem Artikel) verbracht haben (mit der genannten Einschränkung, dass bei *exits* die Aufenthaltsdauer nicht gezählt werden kann). Die durchschnittliche Aufenthaltsdauer (***avgTimeOnPage***) wird schließlich berechnet,

indem die summierte Aufenthaltsdauer durch die Seitenaufrufe geteilt wird, bei denen die Zeit gemessen werden konnte, also *pageviews* – *exits*.

Es liegt also pro Artikel ausschließlich diese durchschnittliche Aufenthaltsdauer der Besuche vor. Weder Aufenthaltsdauern einzelner Nutzer, noch Angaben über die Verteilung (etwa Streuungsmaße) der Aufenthaltsdauern liegen vor. Ebenso gibt es keine weiteren Angaben über das Klick- oder Leseverhalten, etwa wie häufig der Artikel von den gleichen Nutzern geöffnet wird, oder wie viele Tabs ein Nutzer gleichzeitig geöffnet hat. Ob der Artikel also tatsächlich gelesen wird, wenn er von einem Nutzer geöffnet ist, ist nicht klar. Dies gilt für die einzelnen (unbekannten) Nutzerdaten und schlägt sich natürlich auch in den vorliegenden Mittelwerten nieder. Die Hoffnung ist aber, dass durch die Mittelwertbildung größerer Mengen an Einzeldaten einzelne solcher „fehlerhaften“ Ausreißer in beide Richtungen ausgeglichen werden und somit weniger stark ins Gewicht fallen, sodass die Mittelwerte tatsächlich Aufschluss auf das generelle Nutzerverhalten bzw. ihr Interesse oder Präferenz für die verschiedenen Artikel geben können²⁶.

Wie in Tabelle 2 bereits angerissen, wurde der Datensatz mit Blick auf die Modellierung zufällig in drei Teile eingeteilt. Dabei bilden 80% der Artikel die Trainingsdaten für die Modelle (*train*), 10% werden zur finalen Evaluation der Modelle als Testdatensatz zurückgehalten (*test*). Weitere 10% bilden den Entwicklungsdatensatz (*dev*), der vor allem bei den DL-Modellen, aber auch bei den BOW-Modellen in der Modellierungsphase genutzt werden kann, um die Modellgüte abzuschätzen, verschiedene Varianten eines Modells zu vergleichen und Hyperparameter festzulegen.

Die zufällig eingeteilten Splits unterscheiden sich geringfügig in ihren statistischen Eigenschaften: Die Texte im Testset sind im Durchschnitt etwas kürzer als die im Trainings- und Entwicklungsset und streuen dabei etwas weniger. Gleiches gilt für die Aufenthaltsdauer, auch hier liegt der Durchschnitt im Testdatensatz (182.29 Sekunden) etwas unter dem der Trainingsdaten (185.81 Sekunden) und die Standardabweichung ist geringer. Auch die Korrelation zwischen Textlänge und Aufenthaltsdauer ist in den drei Sets unterschiedlich stark ausgeprägt. Diese Beobachtung wird bei der Modellevaluation (siehe Abschnitt 6) wieder aufgegriffen.

Abb. 2 zeigt in Ergänzung zu den Beschreibungsmaßen in Tabelle 2 die Verteilungen von der Anzahl an Seitenaufrufen, Textlänge und Aufenthaltsdauer im -Datensatz.

Wie die Verteilung in der ersten Grafik in Abb. 2 und die Diskrepanz zwischen Mittelwert und Median zeigt, schwankt die Zahl an Seitenaufrufen pro Artikel sehr

²⁶Einige Arbeiten (Seki & Yoshida, 2018; Homma et al., 2018; Claypool et al., 2001) wählen statt des Mittelwertes den *Median* der Aufenthaltsdauer. Dies ist in der vorliegenden Arbeit aufgrund der nicht vorhandenen Einzeldaten nicht möglich. Es werden aber nur Artikel verwendet, deren durchschnittliche Aufenthaltsdauer sich aus mindestens 50 Besuchen zusammensetzt.

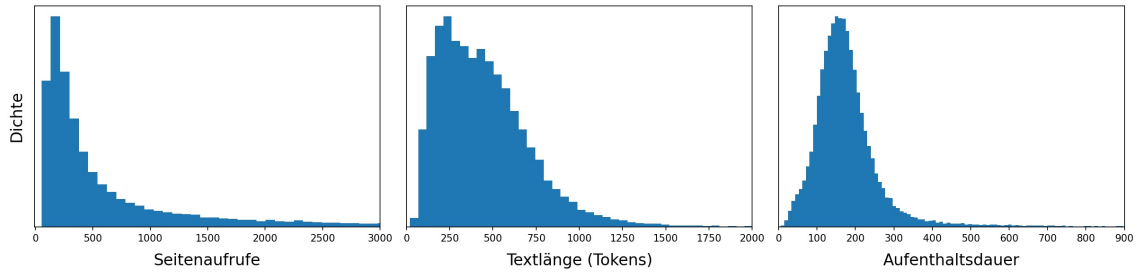


Abbildung 2: Verteilungen von Seitenaufrufen, Textlänge und Aufenthaltsdauer in den [REDACTED]-Artikeln.

stark und hat dabei eine stark rechtsschiefe Verteilung: Die meisten der Artikel haben weitaus weniger als 1000 Klicks (gut 75% liegen unter 1000 Klicks, der Median liegt bei 364 Aufrufen), wenige Artikel haben hingegen sehr große Klickzahlen²⁷.

Die Verteilung der Textlänge (gemessen in Zahl der Tokens ohne Satzzeichen, die mittlere Grafik in 2) zeigt, dass nur wenige Texte mehr als 1000 Tokens umfassen (dies betrifft 4%, nur 0.4% sind länger als 2000 Tokens und fallen damit aus dem in Abb. 2 dargestellten Bereich). Die mittlere Textlänge beträgt knapp 462 Tokens, die Verteilung ist leicht rechtsschief, der Median liegt bei 409 Tokens²⁸.

Die durchschnittliche Aufenthaltsdauer pro Artikel (rechte Grafik in 2) hat eine ziemlich symmetrische Verteilung. Der Mittelwert liegt bei 185.85 Sekunden, der Median bei 164.76 Sekunden, dies liegt an einigen Ausreißern mit besonders langen Aufenthaltsdauern einzelner Artikel. 1.6% der Artikel haben eine durchschnittliche Aufenthaltsdauer von mehr als 10 Minuten (600 Sekunden). Dies wäre für einen einzelnen Nutzer vielleicht nicht ungewöhnlich, mit Blick auf das Mittelungsverfahren und im Vergleich zum Großteil der Artikel, der unter der 5-Minuten-Grenze liegt, könnte es sich um Ausreißer handeln. Allerdings liegt die mittlere Textlänge in dieser Gruppe auch bei 937 Tokens, es handelt sich also auch um überdurchschnittlich lange Texte, was die hohe Aufenthaltsdauer realistisch erscheinen lässt.

Der bereits angesprochene Zusammenhang zwischen Textlänge und Aufenthaltsdauer ist in Abb. 3 abgetragen. Für eine bessere Sichtbarkeit wurden nur Texte mit einer maximalen Textlänge von 2000 Tokens, sowie einer maximalen Aufenthaltsdauer von 600 Sekunden abgebildet (dies schließt weniger als 1% der Artikel aus). Es lässt sich die Tendenz beobachten, dass längeren Texten auch längere Aufenthaltsdauern entsprechen, der Zusammenhang ist mit einem Korrelationskoeffizienten von 0.26

²⁷Der meistgeklickte Artikel hat 976357 Aufrufe. In Abb. 2 ist zur besseren Sichtbarkeit die X-Achse auf 3000 beschränkt, knapp 9% der Artikel liegen über dieser Grenze.

²⁸Für die Einordnung der BERT-Modelle ist die Anzahl an BERT-Tokens von Bedeutung. Bei dieser Zählung liegt der Mittelwert bei knapp 675 Tokens pro Text, der Median bei 597 Tokens. Viele der Artikel sind also länger als die feste BERT-Begrenzung auf Eingaben mit bis zu 512 BERT-Tokens.

(Pearson) bzw. 0.52 (Spearman) schwach bis moderat ausgeprägt. Die rechte Grafik zeigt denselben Sachverhalt, hier wurden die Artikel nach der Textlänge in Gruppen (*Bins*) mit Intervallbreite 50 zusammengefasst (also beispielsweise alle Artikel der Längen 200 bis 250 zusammengefasst) und deren Aufenthaltsdauern gemittelt. Die Steigung der mittleren Aufenthaltsdauer mit der Textlänge ist so deutlich sichtbar und erkennbar linear.

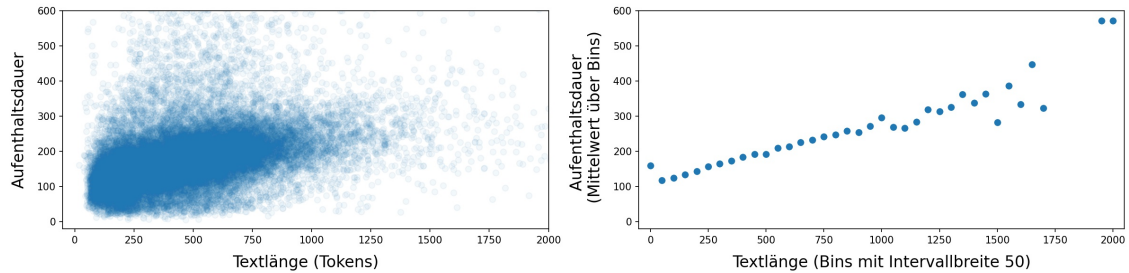


Abbildung 3: Zusammenhang von Textlänge und Aufenthaltsdauer, links jeder Artikel einzeln, rechts mit Gruppierung der Artikel nach Textlänge.

3.4 Wahl der absoluten Aufenthaltsdauer als Zielvariable

Für die Modellierung von Popularität bzw. User Engagement wurde die durchschnittliche absolute Aufenthaltsdauer der Artikel (in Sekunden) als Zielvariable ausgewählt. Die Gründe, die gegen eine Modellierung der Seitenaufrufe sprechen, wurden bereits in Einleitung und Forschungsstand dargestellt (Unklarheit über tatsächliche Zufriedenheit, Verzerrung durch Positionierungseffekte).

Möglich wäre auch eine Betrachtung der *Stickiness*, also der Anteil der Aufrufe, bei denen im Anschluss weitere Artikel des Portals aufgerufen wurden, die also nicht unter *Bounce* oder *Exit* fallen. Hier könnte vermutet werden, dass dies für Nutzerzufriedenheit spricht. Doch auch hier besteht Unsicherheit. Der Nutzer könnte auch gerade *weil* er den aktuellen Artikel uninteressant findet einen anderen anklicken. Da keine spezifischen Aufenthaltsdauern vorliegen, kann darüber nichts ausgesagt werden.

Die Aufenthaltsdauer wurde daher als bestmögliche Annäherung für die Zufriedenheit mit dem Artikel ausgewählt und als Zielvariable der Modellierung verwendet. Wie bereits angesprochen bleibt dabei die Unsicherheit bestehen, wie viel Zeit die Nutzer dabei tatsächlich mit dem Artikeltext beschäftigt sind, welchen Textanteil sie lesen und wie viel Zeit sie stattdessen auf Fotos, Werbung oder Kommentaren verbringen. Aus diesem Grund wurde auch die zunächst naheliegende Idee verworfen, die Aufenthaltsdauer an der Artikellänge zu normieren. Diese Normierung wäre vor allem dann sinnvoll, wenn die Dauer als tatsächliche Lesedauer des gesamten Artikels

interpretiert werden könnte. Da dies aber nicht der Fall ist, wurde die absolute Aufenthaltsdauer beibehalten (eine ähnliche Begründung findet sich auch bei Omidvar et al. (2020)).

Weiterhin wurde die Überführung in eine binäre Entscheidung (Klassifikation) bezüglich Popularität bzw. User Engagement eines Artikels in Betracht gezogen. Hier hätte ein Grenzwert als Trennung zwischen den beiden Klassen herangezogen werden müssen, dessen Wahl sich aber nur schwer festlegen ließe. Der häufig verwendete Grenzwert von 30 Sekunden wird einerseits generell stark kritisiert und andererseits ist er vor allem für *einzelne* Besuche sinnvoll, nicht unbedingt für – wie hier vorliegend – gemittelte Aufenthaltsdauern aller Besuche. Bei der Verteilung der Aufenthaltsdauern der Artikel (siehe das rechte Histogramm in Abb. 2) sind keine offensichtlichen „Knicks“ vorhanden, die als natürliche Grenze hätten dienen können. Auch hier spielt natürlich das Problem der unterschiedlichen Artikellängen eine große Rolle, ebenso wie die unterschiedliche Zahl an Fotos oder anderen Elementen im Artikel, sowie die unterschiedlichen thematischen Bereiche, die sich alle auf die Aufenthaltsdauer auswirken.

Bezeichnung im Datensatz	synonym im Text verwendet	Kurzbeschreibung	Beispielartikel
<i>pageviews</i>	Aufrufe/Klicks/ Besuche	Anzahl Seitenaufrufe	985
<i>entrances</i>		Anzahl an Seiteneinstiegen über den Artikel: Nutzer beginnt bei dem Artikel seine Session (Teilmenge von <i>pageviews</i>)	299
<i>exits</i>		Anzahl an Seitenausstiegen über den Artikel: Nutzer verlässt über den Artikel die Session (Teilmenge von <i>pageviews</i>)	441
<i>bounces</i>		Anzahl an <i>single-page</i> -Besuchen: Nutzer landet auf dem Artikel und geht ohne weitere Klicks wieder (Schnittmenge von <i>exits</i> und <i>entrances</i>)	63
<i>timeOnPage</i>	summierte Aufenthalts- dauer, Gesamtzeit	über alle Aufrufe summierte Aufenthaltsdauer in Sekunden (ohne <i>exits</i> und <i>bounces</i> , da hier kein zweiter Zeitstempel möglich)	104132
<i>avgTimeOnPage</i>	Aufenthalts- dauer, Dauer, Zeit	durchschnittliche Aufenthaltsdauer auf dem Artikel in Sekunden, $avgTimeOnPage = \frac{timeOnPage}{pageviews - exits}$	191.419
<i>title</i>	Titel, Überschrift	Artikelüberschrift (Haupt- und Untertitel zusammengefasst)	
<i>teaser</i>	Teaser, Vorspann, Lead	Text des Teasers/Vorspanns	
<i>article_body</i>	Haupttext, Lauftext	Haupttext des Artikels	
<i>article_text</i>	Artikeltext	vollständiger Artikeltext: Konkatenation von Überschrift, Teaser und Haupttext	
<i>nr_tokens_text</i>	Tokenzahl, Textlänge	Textlänge in Tokens (ohne Satzzeichen)	488
<i>nr_tokens_punct</i>		Textlänge in Tokens (mit Satzzeichen)	544
<i>nr_tokens_BERT</i>	BERT-Tokens (Subtokens)	Textlänge in BERT-Tokens	681
<i>publisher</i>	Publisher, Zeitung	Angabe der Zeitung	

Tabelle 3: Einige KPIs und strukturelle Angaben im Datensatz mit Kurzbeschreibung und beispielhaften Werten anhand eines Artikels.

4 Methodische Grundkonzepte

In diesem Abschnitt werden die methodischen Grundkonzepte dargestellt, auf denen die unterschiedlichen Modelle beruhen, die zur Vorhersage der Aufenthaltsdauer verwendet werden. Die spezifischen Modellarchitekturen befinden sich im anschließenden Abschnitt 5.

Für die Repräsentation von Texten als numerische Features spielen beispielsweise *Bag-of-Words* (BOW), sowie andere Arten von Wort- und Dokumentenvektoren (*Word Embeddings*) eine wichtige Rolle. Auf Ebene der Modelle liegt neben den Baselines (Regressionsmodelle und ein XGBoost-Modell) der Fokus dieser Arbeit vor allem auf Modellen aus dem Bereich *Deep Learning* (DL). Ein *Convolutional Neural Network* (CNN) wird einmalig verwendet, die anderen DL-Modelle enthalten alle eine Transformer-Komponente (BERT), sowie FFN-Architekturen (*Feed Forward Network*). Eines der Modelle enthält weiterhin eine RNN-Komponente (*Recurrent Neural Network*).

Zunächst liegen unvorverarbeitete Zeitungsartikeltexte (Dokumente) vor. Sei

$$D = \{d_1, d_2, \dots, d_N\} \quad (1)$$

ein Korpus, also eine Menge von N Dokumenten. Sei weiterhin

$$Y = \{y_1, y_2, \dots, y_N\} \quad (2)$$

eine Menge von Labeln, die jedem Dokument d_i ein Label y_i (hier: eine durchschnittliche Aufenthaltsdauer) zuweist. Der im letzten Abschnitt beschriebene Datensatz lässt sich also formal als Paar (D, Y) darstellen. Gesucht wird also allgemein ein Modell f , das Y vorhersagt, gegeben D .

4.1 Textrepräsentation mit *Bag of Words* (BOW)

Um Textdaten (Wörter, Sätze, Texte) als Eingabe in computerlinguistischen Modellen verwenden zu können, ist es notwendig, sie in eine maschinenlesbare Repräsentation zu überführen, genauer in einen numerischen Vektor. Einfache Ansätze einer solchen Vektorisierung lassen sich unter dem Stichwort *Bag-of-Words* (BOW) zusammenfassen. Dabei werden die einzelnen Bestandteile eines Textes – die Tokens – und ihr Vorkommen bzw. ihre Häufigkeit betrachtet. Die Reihenfolge der Tokens wird dabei nicht berücksichtigt, ein Dokument wird also als Menge seiner Wörter betrachtet.

Hierfür müssen die Dokumente zunächst einigen Vorverarbeitungsschritten unterzogen werden. Zunächst werden durch Aufspaltung (Tokenisierung) die einzelnen

Wörter der Dokumente identifiziert. Aus diesen Tokens ergibt sich das Vokabular des Korpus: Jedes *unterschiedliche* Token (ein Tokentype) wird in das Vokabular aufgenommen. Bei den im folgenden aufgestellten Dokumentenvektoren entspricht jede Position des Vektors einem distinkten Tokentype.

Bezüglich der Vorverarbeitung stellen sich einige Entscheidungsoptionen: Neben der Wahl eines geeigneten *Tokenizers* muss beispielsweise über die Verwendung von Satzzeichen und *Stopwörtern* entschieden werden. Unter Stopwörtern versteht man sehr häufig auftretende Wörter einer Sprache (etwa *alle*, *und*, *ist*), die aufgrund ihrer Häufigkeit und ihres eher geringen semantischen Gehalts nur wenig inhaltliche Aussagekraft besitzen, da sie bei den meisten Dokumenten mit ähnlich hoher Frequenz zu vermuten sind. Aus diesem Grund werden Stopwörter häufig aus dem Vokabular entfernt, also nicht weiter beachtet. Außerdem kann man sich für oder gegen eine generelle Überführung der Tokens in Kleinschreibung entscheiden²⁹. Ein beliebter Vorverarbeitungsschritt ist die *Lemmatisierung*³⁰ der Tokens vor der Berechnung der Häufigkeiten. Tokens werden dafür in ihre Lemmata (also ihre Grundform) überführt, sodass etwa Flexionsformen (*sind*, *war*, *bist*) vereinheitlicht und damit gemeinsam gezählt werden (als Lemma *sein*). Gleiches gilt für Kasus- oder Numerusendungen, so werden die Tokens *verlag*, *verlage*, *verlages* alle zum Lemma *verlag*. Alle diese Schritte werden im Folgenden unter *Normalisierung* zusammengefasst.

Nach der Vorverarbeitung des Korpus durch Tokenisierung und Normalisierung lässt sich jedes Dokument d_i als eine Abfolge von Wörtern (Types)

$$d_i = (w_1^i, w_2^i, \dots, w_{n_i}^i) \text{ für } 1 \leq i \leq N \quad (3)$$

darstellen, n_i ist dabei die Anzahl an Wörtern im Dokument d_i . Die Menge aller Wörter, die in mindestens einem der Dokumente in D auftauchen, ergibt das Vokabular V mit

$$V = \{\omega_1, \omega_2, \dots, \omega_M\} \quad (4)$$

Jedes Wort w_j^i entspricht also einem Vokabularelement $\omega \in V$ für $1 \leq i \leq N$ und $1 \leq j \leq n_i$.

Nun lässt sich jedes Dokument $d \in D$ (auf Dokumenten-Indizes wird im Folgenden ggf. zur besseren Lesbarkeit verzichtet) in einen *Bag-of-Words*-Vektor

$$x = (x_1, x_2, \dots, x_k, \dots, x_M) \quad (5)$$

²⁹Gerade im Deutschen führt das zwar gelegentlich zum Verlust von einigen semantisch relevanten Unterscheidungsmöglichkeiten (*Folgen* vs. *folgen*), allerdings umgeht man so das Problem der Großschreibung am Satzanfang (*Gestern* vs. *gestern*), was andernfalls zu distinkten Types führen würde.

³⁰Daneben ist auch *Stemming* eine Möglichkeit, also das Zurückführen auf einen Wortstamm durch simples Entfernen von Endungen.

überführen, wobei die Komponente x_k die Häufigkeit des Vokabularelements ω_k im Dokument d beschreibt. Hierfür können verschiedene Arten der Zählung verwendet werden:

Bei der einfachsten Variante, der *binären* Kodierung, wird ausschließlich betrachtet, welche der Types im Dokument d_i vorliegen und welche nicht:

$$x_k^{bin} = \begin{cases} 1, & \text{wenn } \omega_k \text{ in } d_i \text{ enthalten ist} \\ 0, & \text{sonst.} \end{cases} \quad (6)$$

In der verbreiteteren und meist unter BOW verstandenen Variante werden statt binärer Betrachtung die *absoluten Häufigkeiten* der Tokens im Dokument ermittelt. Die Komponente x_k ist also

$$x_k^{abs} = count(\omega_k, d_i) \quad (7)$$

wobei die *count*-Funktion die Häufigkeit eines Wortes in einem Dokument angibt.

Statt absoluten können auch *relative* Häufigkeiten verwendet werden, dabei werden die absoluten Häufigkeiten anhand der Anzahl der Wörter im Dokument (n_i) normiert:

$$x_k^{rel} = \frac{count(\omega_k, d_i)}{n_i} \quad (8)$$

Eine weitere BOW-Variante ist der Tf-Idf-Ansatz (*Term Frequency, Inverse Document Frequency*), der die Stopwortproblematik auf andere Weise löst. Hier werden pro Token zwei Maße betrachtet. Das erste ist die absolute Häufigkeit des Tokens im Dokument (*Term Frequency*):

$$tf_{\omega_k} = count(\omega_k, d_i) \quad (9)$$

Das zweite Maß ist die inverse Dokumenthäufigkeit (*Inverse Document Frequency*), sie wird anhand der Anzahl an Dokumenten im Korpus berechnet, die das Token enthalten:

$$idf_{\omega_k} = \frac{N}{count(d, \omega_k)} \quad (10)$$

wobei die *count*-Funktion die Anzahl der Dokumente zurückgibt, die das Token ω_k enthalten, N ist die Anzahl aller Dokumente im Korpus. Das Produkt dieser beiden Maße ergibt schließlich das Tf-Idf-Maß:

$$tf-idf_{\omega_k} = tf_{\omega_k} \cdot idf_{\omega_k} \quad (11)$$

Tokenhäufigkeiten werden dabei also anhand ihrer „Seltenheit“ im Korpus gewichtet, also gewissermaßen anhand ihrer „Aussagekraft“.

Zusätzlich zu einzelnen Tokens bzw. Lemmata werden häufig auch *N-Gramme* betrachtet. Gemeint sind kleine Gruppen von aufeinanderfolgenden Tokens. Dabei muss über die Länge der zu betrachtenden N-Gramme entschieden werden. Bigramme (*morgen früh, New York*) und Trigramme (*am vergangenen Mittwoch*) werden häufig betrachtet, N-Gramme länger als 5 nur selten. Da bei einer großen Anzahl an Dokumenten und je nach Vorverarbeitung und Wahl der N-Gramm-Längen sehr schnell eine sehr große Menge an Features zusammenkommt, viele davon aber sehr selten auftreten, wird oft eine Maximalgröße (bspw. 10000) bestimmt. Es werden dann nur die häufigsten 10000 im Korpus auftretenden Tokens und N-Gramme berücksichtigt.

Ein großer Vorteil der beschriebenen BOW-Ansätze zur Textrepräsentation ist ihre einfache Prozessierung und Interpretierbarkeit. BOW-Modelle erzielen oft bereits gute Ergebnisse bei vielen Anwendungen (etwa Textklassifikation) und bilden damit eine exzellente Baseline. Andererseits wird bei dieser Betrachtung eines Textes als Menge seiner Tokens einige Information nicht beachtet, die sequentielle Abfolge des Textes bzw. das Auftreten der Tokens in syntaktischen und semantischen Beziehungen zueinander kann hier nicht abgebildet werden. Auch Wissen über die tatsächliche *Bedeutung* der Wörter liegt in der BOW-Modellierung nicht vor.

Dem Problem der fehlenden Semantik in der Repräsentation widmet sich der nächste Abschnitt zu Wort- und Dokumentenvektoren (*Word Embeddings*) als modernere Repräsentation von Text.

4.2 Wort- und Dokumentenvektoren (*Word Embeddings*)

Im vorigen Abschnitt wurde *Bag-of-Words* (BOW) als Art der Repräsentation von Dokumenten anhand von Worthäufigkeiten vorgestellt, eine Methode, die Wortbedeutung sowie Bedeutungszusammenhänge zwischen (benachbarten) Wörtern außer Acht lässt. Eine neuerer Ansatz, der sich mit der numerischen Repräsentation von *Wörtern* und deren Bedeutung befasst, sind Wortvektoren (*Word Embeddings*). Ein Type w wird dabei als ein Vektor $vec(w) \in \mathbb{R}^c$ repräsentiert, vec bezeichnet den Zugriff auf die Wortvektoren, c bezeichnet die Dimensionalität des Vektorraums (in der Regel zwischen 50 und 1000, meist 300).

Die einzelnen latenten Dimensionen tragen dabei keine offenkundige Bedeutung, trotzdem ist die Bedeutung des Tokens darin kodiert. Die Tokens werden in einen multidimensionalen Vektorraum abgebildet, Wörter mit ähnlicher Bedeutung (bzw. ihre Vektoren) liegen in räumlicher Nähe zueinander. Das ermöglicht, semantische Ähnlichkeit oder Zusammenhänge zwischen Tokens mathematisch zu erfassen.

Bedeutung wird dabei im Sinne der distributionellen Semantik verstanden die Bedeutung eines Wortes ergibt sich aus seinem Auftreten und Gebrauch im Zusammenhang mit anderen Wörtern (vgl. etwa Firth (1957)). Wörter mit semantischer Ähnlichkeit treten in ähnlichen Kontexten, also tendenziell in Nachbarschaft zu den gleichen Wörtern auf – sofern eine ausreichend große Menge an Texten betrachtet wird.

Trainiert werden solche Wortvektoren daher anhand von sehr großen Mengen an Textdaten. Verwendet werden unüberwachte Lernverfahren, die Daten sind also Rohtexte, es sind keine Annotationen notwendig. Im Wesentlichen gibt es zwei Ansätze zur Berechnung von Wortvektoren der Tokens:

Der erste Ansatz besteht in der Zählung aller Kookkurrenzen eines Tokens, also der Wörter, die in einem Fenster bestimmter Größe um das Token auftreten. Auf diese Weise kann eine Kookkurrenz-Matrix erstellt werden. Mithilfe von statistischen Verfahren zur Matrizendekomposition werden diese hochdimensionalen und dünnbesetzten (*sparse*) Matrizen dann in dichtbesetzte (*dense*) Matrizen kleinerer Dimensionen überführt. Jede Zeile entspricht dann dem Wortvektor eines Tokens.

Der zweite, modernere Ansatz verwendet maschinelle neuronale Lernverfahren anhand von Trainingsaufgaben, die darauf abzielen, in den Gewichten allgemeines Wissen über semantische Bedeutung abzubilden. Ein bekannter Algorithmus ist dabei word2vec (Mikolov et al., 2013) in seinen zwei Varianten. In der einen lernt ein Modell, aus bekannten Kontextwörtern ein maskiertes Zielwort vorherzusagen (*continuous bag-of-words*, CBOW). Die zweite Variante verfolgt den umgekehrten Weg: Anhand eines fokussierten Wortes lernt das Modell, die umliegenden Wörter (also deren Wahrscheinlichkeiten) zu erraten (*skip-gram*). Im Laufe des Trainingsprozesses lernt das Modell auf diese Weise, die zu Beginn zufällig initialisierten Gewichte der Wortvektoren mit Semantik zu „füllen“. Diese so trainierten Gewichte entsprechen dann den Wortvektoren und stehen für weitere Anwendungen zur Verfügung. Weitere Algorithmen bzw. vortrainierte Wortvektoren, die verwendet werden können, sind etwa GloVe (Pennington et al., 2014) oder fastText (Grave et al., 2018). GloVe verbindet die beiden Ansätze (Matrixfaktorisierung und neuronale Lernverfahren), während fastText eine Erweiterung des word2vec-Algorithmus darstellt, bei dem Subtoken-Information miteinfließt.

In der Tat sind solche vortrainierten Wortvektoren in der Lage, semantischen Gehalt zu repräsentieren. Vektoren ähnlicher Wörter (zum Beispiel *Hund* und *Welpen*) erhalten ähnliche Vektoren, wie mathematische Ähnlichkeitsmaße (etwa Kosinusähnlichkeit) zeigen. Auch semantische Relationen zwischen Wörtern lassen sich anhand von Rechenoperationen mit Wortvektoren darstellen. So verhält sich die Differenz der Vektoren *Hund* und *Welpen* ähnlich wie die Differenz von *Pferd* zu *Fohlen*.

Ein Problem solcher vortrainierten *off-the-shelf*-Wortvektoren ist Polysemie: Für jedes Token liegt in der Regel genau *ein* Vektor vor, Polyseme (wie die verschiedenen Bedeutungen von *Bank*) fallen zusammen und können nicht unterschieden werden.

Wortvektoren (Word Embeddings) repräsentieren also primär einzelne Wörter. Um damit Sequenzen von Wörtern (also Sätze oder längere Dokumente) zu repräsentieren, gibt es verschiedene Möglichkeiten. Die einzelnen Wortvektoren können untereinander in einer Matrix angeordnet werden und somit von sequentiellen Modellen verarbeitet werden (wie CNNs und RNNs, siehe Abschnitt 4.4). Ein anderer Ansatz ist das simple Verwenden des Mittelwerts aller Wortvektoren der Sequenz als Dokumentenembedding (manchmal *Bag-of-Vectors* oder auch *Zentroid* genannt). Das Dokument $d_i = (w_1^i, w_2^i, \dots, w_{n_i}^i)$ wird dabei als Vektor

$$x = \frac{1}{n_i} [vec(w_1^i) + vec(w_2^i) + \dots + vec(w_{n_i}^i)] \quad (12)$$

dargestellt. Die strukturelle Anordnung der Tokens im Text geht dabei verloren. Die Wahl hierzu erfolgt je nach Modellarchitektur und Fragestellung.

Die Verwendung von Wortvektoren hat gegenüber der BOW-Repräsentation einige Vorteile. Es können semantische Bedeutungen und Ähnlichkeitsbeziehungen abgebildet werden³¹. Der verwendete Merkmalsraum ist homogen, dicht und vergleichsweise klein, was Overfitting der Modelle vorbeugt.

Zwar werden im Training der Wortvektoren sehr große Mengen an Daten und Rechenkapazität gebraucht, es sind aber keine menschlichen Annotationen nötig, die Modelle lernen unüberwacht. Andere Vorhersagemodelle, die ein spezifisches computerlinguistisches Problem (bspw. Textklassifikation, Übersetzung oder das Erkennen von Wortarten) behandeln, können auf die bereits vortrainierten Wortvektoren zurückgreifen und benötigen somit eine sehr viel geringere Menge an taskspezifisch annotierten Trainingsdaten, als es ohne vortrainierte Wortvektoren der Fall wäre (Buechel et al., 2020).

4.3 Baseline-Klassifikatoren: Ridge Regression, XGBoost

Mehrere der Baseline-Modelle verwenden **Ridge Regression** als Modell zur Vorhersage der Aufenthaltsdauer pro Artikel. Dabei handelt es sich um eine Erweiterung der normalen linearen Regression.

Die reine lineare Regression verwendet als zu minimierende Kostenfunktion die Summe der quadratischen Abweichungen der Vorhersagen (Residuen, Fehler) zu

³¹Dies ist bei BOW zwar auf der Dokumentenebene, nicht aber auf der Wortebene möglich.

den wahren Werten. Bezeichnet wird dieses Verfahren oft als Methode der kleinsten Quadrate oder auch *Ordinary-Least-Squares*-Methode.

Bei der Ridge Regression wird der *Least-Squares*-Kostenfunktion ein Term hinzugefügt, und zwar ein Penalisierungsfaktor λ , der große Werte in den Gewichten bestraft. So soll *Overfitting* verhindert werden. Von *Overfitting* spricht man, wenn ein Modell zu sehr an die Trainingsdaten angepasst ist und darauf sehr gute Werte erzielt, aber nicht gut auf ungesehene (Test-)Daten generalisieren kann und dort keine guten Vorhersagen liefert. Diese Art der Regularisierung bezeichnet man als *L2*-Regularisierung, da sich der Regularisierungsfaktor auf die *quadrierten* Gewichte bezieht, bei *L1*-Regularisierung wird hingegen der absolute Wert verwendet.

In der Kostenfunktion (13) ist y_i der wahre Wert an Stelle i , \hat{y}_i die Vorhersage an Stelle i . N bezeichnet die Anzahl an Beobachtungen, m die Anzahl an Gewichten (Koeffizienten, Parametern, Merkmalen) im Modell. λ ist der Penalisierungsfaktor, θ_j ist der gelernte Koeffizient (die *Steigung*) von Merkmal j .

$$loss_{Ridge} = \sum_{i=1}^N (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^m \theta_j^2 \quad (13)$$

Das Modell wird also angehalten, zusätzlich zur Minimierung der Residuen darauf zu achten, die Gewichte klein zu halten. Dies verhindert eine zu genaue Anpassung an die Trainingsdaten durch große Gewichte. Die Wahl von λ ist entscheidend, zu große Werte führen zu *Underfitting*, das Modell lernt keine gute Anpassung, zu kleine Werte zu *Overfitting*, das Modell passt sich zu sehr an die Trainingsdaten an. ($\lambda = 0$ entspricht der normalen linearen Regression ohne Regularisierung.)

Eines der BOW-Modelle (siehe 5.2) nutzt zusätzlich zum linearen Ridge-Regression-Modell einen **XGBoost**-Klassifikator, einen Algorithmus, auf den hier nur sehr knapp eingegangen wird. Bei XGBoost (*Extreme Gradient Boosting*) handelt es sich um ein Modell, das auf Entscheidungsbäumen basiert. Als Ensemble-Modell werden mehrere Modelle (Bäume) parallel trainiert und kombiniert. Boosting meint hier eine Technik, die die Fehler von früheren Verzweigungen nutzt, um neue Verzweigungen zu bestimmen. Die Stärken von XGBoost liegen in der hohen Variabilität und Flexibilität in der Anwendung, aber vor allem in der optimierten Prozessierung dank der Parallelisierbarkeit der Bäume und anderer effizienter und pragmatischer Entscheidungen im Algorithmus.

4.4 Deep Learning: Grundlegende Architekturen

Unter Deep Learning (DL) versteht man die Anwendung von (tiefen) künstlichen neuronalen Netzwerken (*Deep Artificial Neural Networks*). Als Eingabe solcher

Netzwerke bedarf es numerischer Vektoren. Bei der Anwendung auf Textdaten müssen diese also zunächst in Vektorrepräsentationen überführt werden, vor allem die in Abschnitt 4.2 beschriebenen Wort- und Dokumentenvektoren (*Word Embeddings*) kommen hier zum Tragen.

Neuronale Netzwerke bestehen aus einer Vielzahl an Neuronen (*Units*, latente Variablen), die in Schichten (*Layer*) bestimmter Größe angeordnet sind. Die Information der Eingabeschicht (*Input Layer*) fließt durch diese Schichten und unterläuft vielen Transformationen, bevor sie durch die Ausgabeschicht (*Output Layer*) in eine Vorhersage, bzw. einen Ausgabewert überführt wird.

Abb. 4 zeigt schematisch vereinfacht den Aufbau eines neuronalen Netzes (genauer: ein Feed-Forward Netz, siehe unten) mit der Eingabeschicht, zwei Zwischenschichten und einer Ausgabeschicht. Deutlich gemacht ist die Verbindung zwischen den Neuronen zweier Schichten, in blau die Berechnung eines einzelnen Neurons, nicht abgebildet sind Bias und Aktivierungsfunktion.

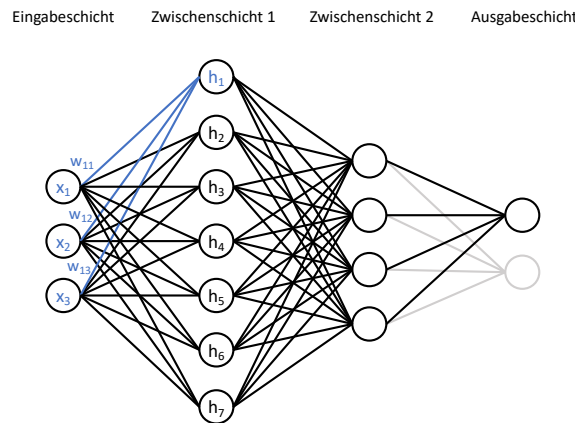


Abbildung 4: Schematische Darstellung eines künstlichen neuronalen Netzwerks.

Innerhalb der DL-Modelle gibt es verschiedene Architekturen, über die ein kurzer Überblick gegeben wird. Den Anfang machen die Feed-Forward Netze (FFN), anhand derer auch einige generelle Grundlagen von neuronalen Netzen erläutert werden. Anschließend werden CNNs (*Convolutional Neural Networks*) und RNNs (*Recurrent Neural Networks*) vorgestellt.

4.4.1 Feed-Forward Netze

Im einem Feed-Forward Netz (**FFN**) ist jedes Neuron einer Schicht mit jedem Neuron der vorherigen Schicht verknüpft, daher tragen solche Schichten auch die Bezeichnung *Fully Connected Layer* oder *Dense Layer*. Ein Neuron fasst dabei die Information, die es aus den Eingabeneuronen erhält, als eine gewichtete Summe zusammen. Hinzu

kommt ein neuronspezifischer Bias-Wert. Anschließend erfolgt eine nicht-lineare Aktivierungsfunktion σ , die über die endgültige Ausgabe des Neurons entscheidet. Üblich ist etwa die Sigmoid-Funktion, oder (inzwischen häufiger) *Leaky Rectified Linear Unit* (ReLU). In Formel 14 ist dies für ein einzelnes Neuron j dargestellt. Es erhält als Input einen Vektor $x \in \mathbb{R}^d$. d bezeichnet die Länge der Eingabe, also die Größe der vorherigen Schicht bzw. bei der ersten Schicht die Dimension der Embeddings. w_{ji} bezeichnet das Gewicht des Neurons j für die Eingabe an i -ter Stelle, b_j ist der Bias-Wert des Neurons j . σ bezeichnet eine Aktivierungsfunktion, die schließlich die Ausgabe h_j bestimmt.

$$h_j = \sigma \left(\sum_{i=1}^d w_{ji}x_i + b_j \right) \quad (14)$$

Da dieser Prozess für jedes Neuron einer Schicht gilt, kann dafür eine Matrixnotation verwendet werden:

$$h = \sigma(Wx + b) \quad (15)$$

Jede Schicht verfügt dabei über eine Gewichtsmatrix W , die für jedes ihrer Neuronen jedem Neuron der Eingabeschicht (bzw. bei tiefen Netzwerken der vorherigen Schicht) ein Gewicht zuordnet. Außerdem ist jeder Schicht ein Bias-Vektor b zugeordnet.

Die Ausgabe jeder Schicht entspricht also einer nicht-linearen Transformation der Eingabe und damit einer neuen abstrakten Repräsentation der Eingabe als Vektor einer bestimmten vordefinierte Größe (Anzahl der Neuronen/Units in der Schicht). Tiefe Netzwerke enthalten mehrerer dieser Zwischenschichten (*Hidden Layer*).

Die letzte Schicht (Ausgabeschicht, *Output Layer*) macht schließlich die Vorhersage. Bei einem Regressionsproblem ist dies eine eindimensionale Ausgabe, also ein skalarer Wert, bei Klassifikationsproblemen ist die Ausgabe ein Vektor, dessen Länge der Anzahl der Klassen entspricht (in Abb. 4 ist dies Unterscheidung durch Graufärbung bei der Ausgabe angedeutet). Um bei der Klassifikation die reinen Logit-Ausgabewerte in Wahrscheinlichkeiten der einzelnen Klassen zu überführen, wird eine Softmax-Funktion verwendet, sie überführt die Werte in den Bereich $[0, 1]$, sodass sie in der Summe 1 ergeben.

In dieser Arbeit finden FFN-Architekturen vor allem dazu Anwendung, Repräsentationen aus anderen Modellkomponenten (vor allem BERT-Modelle) weiterzuverarbeiten und sie in die endgültige Modellvorhersage zu überführen.

4.4.2 Convolutional Neural Networks

Neben den prototypischen FFNs, die etwa Dokumentenembeddings als Eingabe erhalten, gibt es andere DL-Architekturen, CNNs und RNNs, die es ermöglichen, die Eingabe in ihrer *Abfolge* zu betrachten, also die Reihenfolge der Wörter beizubehalten. Bezogen auf die Anwendung auf Textdaten bedeutet dies, lokale Abfolgen (Wortgruppen) zu betrachten (CNNs) oder einen Satz Wort für Wort zu durchlaufen (RNNs). Als Eingabe erhalten diese Modelle daher nicht einen Vektor (einen Wortvektor), sondern eine Matrix.

Das Dokument $d = (w_1, w_2, \dots, w_n)$ (auf Dokumenten-Indizes wird im Folgenden zur besseren Lesbarkeit verzichtet) kann mithilfe der Wortvektoren aller enthaltenen Wörter folgendermaßen als Matrix $X \in \mathbb{R}^{(n \times c)}$ repräsentiert werden:

$$X = \begin{pmatrix} \text{---} & \text{vec}(w_1)^T & \text{---} \\ \text{---} & \text{vec}(w_2)^T & \text{---} \\ & \vdots & \\ \text{---} & \text{vec}(w_n)^T & \text{---} \end{pmatrix} \quad (16)$$

Dabei ist n die Anzahl der Wörter im Dokument und c die Dimension der Wortvektoren.

Das Grundprinzip eines *Convolutional Neural Networks* (**CNN**) ist es, einen *Filter* der Größe s Schritt für Schritt über eine solche Eingabe-Matrix X zu „schieben“. Die einzelnen Positionen (Wörter) in der Eingabe werden als *Zeitschritte* t bezeichnet, s kann als Größe der vom Filter betrachteten Wortgruppe interpretiert werden. Dafür werden jeweils die Vektorrepräsentationen von s Wörtern konkateniert und mit dem Gewichtsvektor des Filters (w) multipliziert (*1d convolution*), bevor eine nichtlineare Aktivierungsfunktion σ angewendet wird. Für ein Dokument der Länge n ergeben sich somit $n - s + 1$ Positionen, an denen der Filter angewendet werden kann. Die Ausgabe $z_t \in \mathbb{R}$ an der Position t ergibt sich als

$$z_t = \sigma(X_{[t:t+s-1]}w + b) \quad (17)$$

wobei $X_{[t:t+s-1]}$ den Vektor der Länge $s \cdot c$ bezeichnet, der sich aus der Konkatenation der s Zeilenvektoren ($X_t, X_{t+1}, \dots, X_{t+s-1}$) ergibt.

Kombiniert man k dieser Filter, ergibt sich für jede zulässige Position t eine neue latente Repräsentation h_t durch

$$h_t = (z_1, z_2, \dots, z_k) = \sigma(X_{[t:t+s-1]}W + b) \quad (18)$$

wobei die Matrix W die Gewichtsvektoren aller Filter zusammenfasst. Die k Komponenten dieser latenten Repräsentation werden *Kanäle* (*channels*) genannt. (h_t sind in der verwendeten Notation Zeilenvektoren und entsprechen damit der Orientierung der Wortrepräsentationen in X .)

Mit Hilfe solcher Filter unterschiedlicher Größe können lokale Muster (etwa Bi- oder Trigramme) erkannt werden und in eigene Repräsentationen überführt werden. Am Schluss werden die einzelnen Ausgaben der Filter kombiniert, etwa über Mittelungs- oder Maximierungsverfahren (*average* bzw. *max pooling*) und Konkatination, und mithilfe normaler *Dense Layer* weiterverarbeitet.

4.4.3 Rekurrente Neuronale Netze

Auch Rekurrente Neuronale Netze (*Recurrent Neural Networks*, **RNN**) erhalten eine Eingabematrix X . Sie lesen die Eingabe dann sequenziell ein, also Wortvektor für Wortvektor. Das Grundprinzip ist hier: Bei jedem Zeitschritt t (Token der Eingabe) wird die Vektorrepräsentation (*Hidden State*) aus dem vorherigen Zeitschritt h_{t-1} verwendet. Jede neu eingelesene Eingabe aktualisiert also die bestehende Repräsentation, sodass nach vollständiger Einlese der Eingabe eine Repräsentation besteht, die Informationen aus *allen* Schritten beinhaltet. Allgemein lautet die Berechnung der Zwischenrepräsentation (h_t) eines Tokens an Stelle t :

$$h_t = \sigma(h_{t-1}W_h + X_tW_X + b) \quad (19)$$

h_{t-1} ist die Repräsentation des vorherigen Tokens bzw. Zeitschrittes (h_0 wird zufällig initialisiert), X_t ist der Wortvektor des aktuellen Tokens (eine „Zeile“ der Eingabematrix X), W_h und W_X sind gelernte Gewichtsmatrizen, b ist der Bias-Term, σ ist eine nicht-lineare Aktivierungsfunktion.

Der Vorteil von RNNs ist ihre Fähigkeit, die Wörter in ihrer Abfolge abzuarbeiten, Verbindungen zwischen den Wörtern zu erkennen und so auch aus mehreren Wörtern zusammengesetzte Bedeutungen zu finden, also den Kontext zu beachten. Es gibt verschiedene Erweiterungen der RNN-Architektur, etwa bidirektionale Architekturen (die Eingabe wird in beide Richtungen durchlaufen). Andere Architekturen widmen sich dem Problem, dass die originalen RNNs dazu neigen, Information aus frühen Zeitschritten im Laufe der Eingabe zu „vergessen“ (bekannt als *Vanishing Gradient Problem*). Das Einbauen von sogenannten *Gates* ermöglicht es, dass relevante Informationsteile direkt zum nächsten Zeitschritt weitergereicht werden (ähnlich wie bei den *Residual Connections* in einem Transformer, siehe unten). Solche Erweiterungen sind *Long Short-Term Memory* Modelle (LSTM) und *Gated Recurrent Units* (GRU).

In dieser Arbeit verwendet eines der Modelle (siehe die zweite Variante des hierarchischen BERT-Modells in 5.7) eine GRU-Architektur. Hier besteht die Eingabe aber nicht aus einer Wortsequenz, sondern aus bereits abstrahierten Repräsentationen der einzelnen Textabschnitte eines Artikels.

4.4.4 Training von Neuronalen Netzen

Bei der Verwendung von neuronalen Netzen in der Modellierung dienen Wortvektoren bzw. Eingabe-Matrizen als Features. Das händische Erstellen von Merkmalen, sei es BOW oder auch Erkennen von bestimmten Schlüsselbegriffen (zum Beispiel *Named Entities*) oder anderen linguistischen Einheiten, entfällt damit. Das Modell übernimmt diese Aufgabe und erlernt idealerweise selbst, die für den Task relevanten Muster zu erkennen.

Wie trainieren und lernen also solche neuronalen Netzwerke? Die Werte der Gewichte und Biases werden zu Beginn zufällig initialisiert und im Laufe des Trainingsprozesses erlernt, also angepasst. Dies erfolgt durch die schrittweise Minimierung einer Fehler-Funktion (*Loss-Funktion*), die aus den aktuellen Vorhersagen und den wahren Werten der durchlaufenen Trainingsdaten berechnet wird. Diese Minimierung erfolgt anhand von statistischen Algorithmen zum Gradientenabstieg. Der Fehler wird berechnet, der Gradient der Fehlerfunktion ermittelt und in einem anschließenden Backpropagation-Schritt werden die Gewichte minimal verändert, sodass das Modell – hoffentlich – im nächsten Schritt bessere Vorhersagen macht. Dieser Vorgang wird iterativ fortgesetzt, bis die Performanz des Modells zufriedenstellend ist bzw. keine Verbesserung der Modellvorhersagen mehr beobachtet wird.

Das „Wissen“ eines neuronalen Netzwerkes liegt also – wie in den anderen Modellen auch – in den spezifischen Werten der Gewichte der Parameter. Neuronale Netzwerke verfügen dabei aber über eine sehr große Anzahl an Parametern, je nach Größe und Anzahl der Zwischenschichten, die gewählt werden. Damit sind sie in der Lage, höchst komplexe Muster zu erkennen und sehr gute Vorhersagen zu treffen. Gleichzeitig benötigen sie aber sehr große Mengen an Trainingsdaten und sind aufgrund ihrer Komplexität sehr rechenintensiv. Außerdem ist Overfitting ein Problem, das nicht unterschätzt werden sollte.

Hier spielt die Verwendung von Datensplits und insbesondere der Entwicklungssatzensatz (dev-Set) eine entscheidende Rolle: Während das Modell nur auf den Trainingsdaten trainiert und aufgrund der so berechneten Fehler und Ableitungen optimiert wird, wird es in regelmäßigen Abständen in seinem aktuellen Zustand (also anhand der aktuellen Gewichte) auf dem Entwicklungssatz evaluiert und beobachtet. Dies ist wichtig, um sicherzustellen, dass das Modell seine Vorhersagekraft auf ungesehenen Daten im Laufe des Trainings verbessert, also tatsächlich *dazu lernt*. Auf diese

Weise kann auch Overfitting bemerkt und gegebenenfalls durch Änderung bestimmter Hyperparameter oder Einführen von Regularisierungsoptionen verhindert werden. Die Beobachtung anhand des dev-Sets ermöglicht schließlich auch, einen Endpunkt des Trainings zu bestimmen – in der Regel dann, wenn stabil keine Verbesserungen mehr beobachtet werden.

Ein Nachteil der Modellierung mit neuronalen Netzen und der Verwendung von Wortvektoren als Features ist, dass die Modelle und ihre Entscheidungen nur sehr schwer für den menschlichen Anwender interpretierbar sind, was oft als *Black-Box*-Problem bezeichnet wird. Anders als bei linearen Modellen kann nicht einfach das Gewicht eines bestimmten Merkmals betrachtet werden, um dessen Aussagekraft für die Modellvorhersagen einzuholen. Einige Tools wie etwa SHAP (Lundberg & Lee, 2017) gehen dieses Problem an und stellen Werkzeuge bereit, mit denen die Modellvorhersagen von – unter anderem – neuronalen Netzen genauer betrachtet und interpretiert werden können. Siehe hierfür den Überblick und die Anwendung von SHAP auf zwei Modelle dieser Arbeit in Abschnitt 7 bzw. im Anhang der Arbeit.

4.5 Transformer-Modelle: *Self-Attention*

Eine neuere DL-Architektur, die eine Trendwende in der computerlinguistischen Modellierung eingeleitet hat, sind sogenannte **Transformer**-Modelle (Vaswani et al., 2017). Die Eingabe wird dabei nicht mehr sequenziell eingelesen und verarbeitet (wie in RNNs) oder auf lokale Muster untersucht (CNNs). Stattdessen nimmt ein neuer Prozess die zentrale Rolle für die Repräsentation von Verbindungen in der Eingabe ein, die sogenannte *Self-Attention*. Hierbei wird jedes Wort der Eingabe mit jedem anderen Wort in Beziehung gesetzt und eine Art „Verbindungsstärke“ zwischen den Wörtern ermittelt. So können Transformer-Modelle Zusammenhänge von Wörtern auch mit größeren Abständen erkennen und für ihre abstrakten Repräsentationen nutzen. Die Repräsentation eines Tokens enthält also Informationen aus allen Tokens der Eingabe.

Die ursprüngliche Anwendung von Transformern ist maschinelle Übersetzung. Dafür wird der Ursprungstext durch mehrere *Encoder*-Schichten in abstrakte Repräsentationen überführt, anhand derer anschließend mit *Decoder*-Schichten der Zieltext ausgegeben wird. Der Encoder-Teil der Transformerarchitektur wurde nachträglich auf monolinguale Textverarbeitung unterschiedlicher Anwendungen angepasst und verwendet (wie etwa bei BERT). Daher wird hier nur die Encoder-Komponente des Transformers angesprochen.

Ein Transformer enthält mehrere *Encoder*-Schichten, die alle dem gleichen Prinzip folgen und aus vier Blöcken bestehen (siehe die schematische Darstellung in Abb. 5):

Self-Attention, Summe & Normalisierung, FFN, erneute Summe & Normalisierung. Diese vier Schritte werden im Folgenden erklärt.

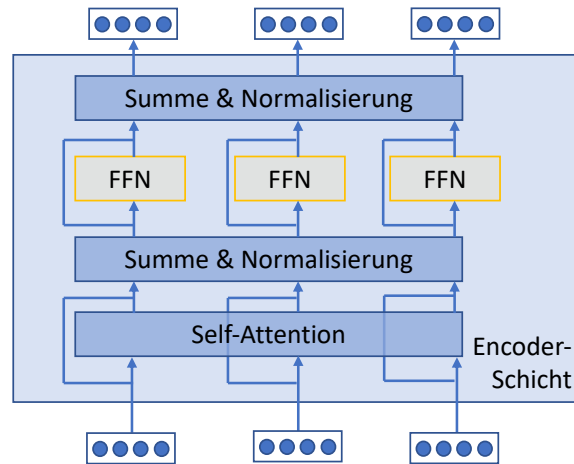


Abbildung 5: Schematische Darstellung einer der Encoder-Schichten im Transformer.

Self-Attention wird anhand von jeder Position der Eingabe berechnet. Zunächst wird jedes Token w_i in einen Wortvektor $vec(w_i)$ überführt (dies geschieht einmalig im ersten Block, die weiteren Blocks erhalten bereits die Repräsentationen des vorherigen Blocks). Anschließend werden für jedes Token drei unterschiedliche Vektoren erstellt, nämlich ein *Query*-Vektor (q_i), ein *Key*-Vektor (k_i) und ein *Value*-Vektor (v_i). Dies geschieht jeweils durch lineare Transformation des Wortvektors $vec(w_i)$ mit drei im Training erlernten Gewichtsmatrizen W_Q , W_K und W_V .

In Matrizennotation lässt sich das so formulieren (X ist die Eingabematrix, also die Aneinanderreihung der einzelnen Wortvektoren, vgl. Formel 16):

$$\begin{aligned} Q &= XW_Q \\ K &= XW_K \\ V &= XW_V \end{aligned} \tag{20}$$

Um für ein bestimmtes Token w_i seine Repräsentation z_i zu berechnen, wird sein Query-Vektor q_i mit den Key-Vektoren (k_1, \dots, k_n) jedes Tokens der Eingabe skalar-multipliziert. Auf diese Weise erhält jedes Token einen *Attention-Score*, der Auskunft darüber gibt, wie „relevant“ das jeweilige Token für Token w_i ist. Diese Scores werden skaliert (dividiert durch $\sqrt{d_k}$, d_k ist die Dimension der Key-Vektoren) und mithilfe einer Softmax-Funktion in den Bereich $[0,1]$ überführt, sodass sie in der Summe 1 ergeben. Anschließend werden diese Attention-Scores mit den Value-Vektoren (v_1, \dots, v_n) multipliziert und die so berechneten Vektoren summiert. Als Ergebnis erhält man z_i , eine abstrakte Repräsentation für Token w_i , genauer eine anhand der Attention-Scores gewichtete Summe der Vektoren aller Tokens. Dieser Vorgang wird

für alle Wortvektoren der Eingabe durchgeführt, in Matrizenschreibweise lautet dies unter Verwendung der drei Matrizen Q , K und V aus Formel 20:

$$Z = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V \quad (21)$$

In den Transformer-Modellen wird **Multi-Head-Attention** verwendet. Dies meint, dass nicht nur ein Attention-Score und eine Repräsentation z_i pro Token berechnet wird, sondern mehrere. Jeder sogenannte Attention-Head h erhält eigene Gewichtsmatrizen W_{hQ} , W_{hK} und W_{hV} . Für Token w_i erhält man also mehrere (je nach Anzahl m der *Heads*) Repräsentationen z_{i1}, \dots, z_{im} . Bevor diese multiplen Repräsentationen dem FFN-Layer (siehe unten) übergeben werden können, werden sie konkateniert und durch Multiplikation mit einer weiteren Gewichtsmatrix W^O wieder in einen einzigen Vektor der passenden Größe überführt. Auf diese Weise soll das Modell befähigt werden, verschiedene Arten der (semantischen oder grammatischen) Beziehung zwischen den Tokens abzubilden. (Die Notwendigkeit mehrerer solcher Attention-Heads ist aber umstritten, vgl. Michel et al. (2019).)

Im an die Self-Attention anschließenden Block (**Summe & Normalisierung**) wird zu der neuen Repräsentation z_i die ursprüngliche Repräsentation (im ersten Block der Wortvektor $\text{vec}(w_i)$) summiert und das Ergebnis normiert (*Layer Norm*), bevor es anschließend ein FFN-Layer durchläuft. Hier kann also Information aus den Eingabevektoren unverändert in die Repräsentation einfließen (wie die Pfeile in Abb. 5 andeuten). Solche Direktverbindung, die ein „Umgehen“ einer Schicht erlauben, bezeichnet man als Residual-Verbindungen (*Residual Connections*, eine ähnliche Funktionalität wie die *Gates* der RNN-Erweiterungen). Auf diese Weise ist nicht ausschließlich der Self-Attention-Prozess entscheidend für die Repräsentation, und wichtige Information aus dem vorherigen Schritt kann direkt einfließen. Die Repräsentation jedes Tokens durchläuft anschließend *einzeln* einen **FFN-Block** (*position-wise* FFN). Nach dem FFN erfolgt ein weiterer **Summe-&-Normalisierung-Block**, auch hier kann also mithilfe der Residual-Verbindung die vorherige Repräsentation über direkten Weg in die neue Repräsentation einfließen.

In den weiteren Encoder-Schichten erfolgen die gleichen Prozesse (Self-Attention, Summe & Normalisierung, FFN, Summe & Normalisierung) pro Position in der Eingabe. Hier dient – statt wie in der ersten Schicht die Wortvektoren – direkt die Ausgabe der vorherigen Schicht (z_1, \dots, z_n) als Eingabe.

Transformer-Modelle enthalten also mehrere solcher Encoder-Schichten mit Self-Attention als zentralem Element, um immer abstraktere Zusammenhänge abbilden zu können. Ein Vorteil gegenüber sequenziellen RNN-Architekturen ist, dass die Self-Attention der Kombinationen parallel berechnet werden kann. Das Modell ist nicht

mehr auf sequenzielles Einlesen und auf die Repräsentation des vorherigen Zeitschritts angewiesen. Im Gegensatz zu CNNs sind Transformer in der Lage, Abhängigkeiten über große Abstände in der Eingabe zu erkennen (*long distance dependency*). Andererseits sind die Self-Attention-Berechnungen sehr komplex und quadratisch mit der Länge der Eingabe (also der Textlänge), die Modelle erfordern also hohe Rechenkapazität und verfügen über eine sehr große Anzahl an Parametern. Aufgrund dieser quadratischen Komplexität der Berechnung ist bei Transformermodellen in der Regel die Eingabelänge beschränkt. BERT (siehe folgender Abschnitt) verarbeitet etwa nur Sequenzen mit höchstens 512 Subtokens.

4.6 BERT: *Pretraining* und *Transfer Learning*

Für diese Arbeit zentral ist dabei ein bestimmtes Modell aus der Transformer-Gruppe, nämlich **BERT**, das im Jahr 2018 von Google vorgestellt wurde (*Bidirectional Encoder Representations from Transformers*) und das Feld der Computerlinguistik stark geprägt hat (Devlin et al., 2019). BERT nutzt die Self-Attention-basierte Encoder-Komponente der Transformer-Architektur, um Eingaben *kontextualisiert* in semantisch bedeutungsvolle latente Repräsentationen zu überführen.

Einen weiteren Grundstein, auf dem BERT aufbaut – die Idee, große *vortrainierte* Modelle für die kontextualisierte Repräsentationen von Text bzw. Wörtern zu nutzen –, legen die *Embeddings from Language Models* (ELMo) von Peters et al. (2018). Statt einem Token einen festen vortrainierten Wortvektor zuzuordnen (wie die Embeddingmodelle aus Abschnitt 4.2), erhält bei ELMo jedes Token im Text seine Repräsentation abhängig von seiner Umgebung. ELMo verwendet dafür ein bidirektionales rekurrentes Netzwerk (LSTM), das mit der Trainingsaufgabe von Sprachmodellierung auf einem großen Textkorpus trainiert wurde. Sprachmodellierung (*language modeling*) bedeutet, anhand einer bisherigen Sequenz von Tokens die Wahrscheinlichkeiten des nächsten Tokens vorherzusagen. Als bidirektionalem Netzwerk steht ELMo dabei der Kontext sowohl vor als auch nach dem Token zur Verfügung. So wird allgemeines Sprachverständnis und Kontextwissen erlernt. Dieses Wissen kann dann auf *andere* Aufgaben angewendet werden (*Transfer Learning*).

Transfer Learning meint die Möglichkeit, vortrainierte allgemeine Modelle auf eine spezifische Fragestellung (etwa Textklassifikation oder Textübersetzung) anzuwenden. Man nutzt das allgemeine sprachliche Wissen eines Modells, das sehr große Mengen an Text verarbeitet hat, und nutzt dieses Sprachverständnis für das eigene spezifische Problem. Das beschleunigt und verbessert den Trainingsprozess des finalen Modells stark und benötigt viel kleinere Mengen an Trainingsdaten oder sonstigen Ressourcen.

BERT verfolgt das gleiche Ziel: Anhand von Trainingstasks in einer *Pretraining*-Phase allgemeines Sprachwissen zu erlangen und dieses durch *Transfer Learning* für

andere Aufgaben zur Verfügung zu stellen. Anders als die rekurrente Architektur von ELMo verwendet BERT aber eine Transformer-Architektur, basierend auf Self-Attention.

In Abb. 6 ist die Architektur von BERT schematisch und stark vereinfacht dargestellt. Unten im Bild ist die Eingabe, ein Artikeltext, zu sehen und die Tokenisierung in Einzeltokens. Anschließend erfolgt die Verarbeitung durch die BERT-Encoder-Schichten, die den beschriebenen Transformer-Aufbau haben (Self-Attention, Summe & Normalisierung, FFN, Summe & Normalisierung). An die BERT-Ausgabe angeschlossen ist ein Ausgabe-Modell (dessen spezifische Architektur je nach Zielsetzung angepasst ist), welches schließlich eine finale Ausgabe bzw. Vorhersage liefert.

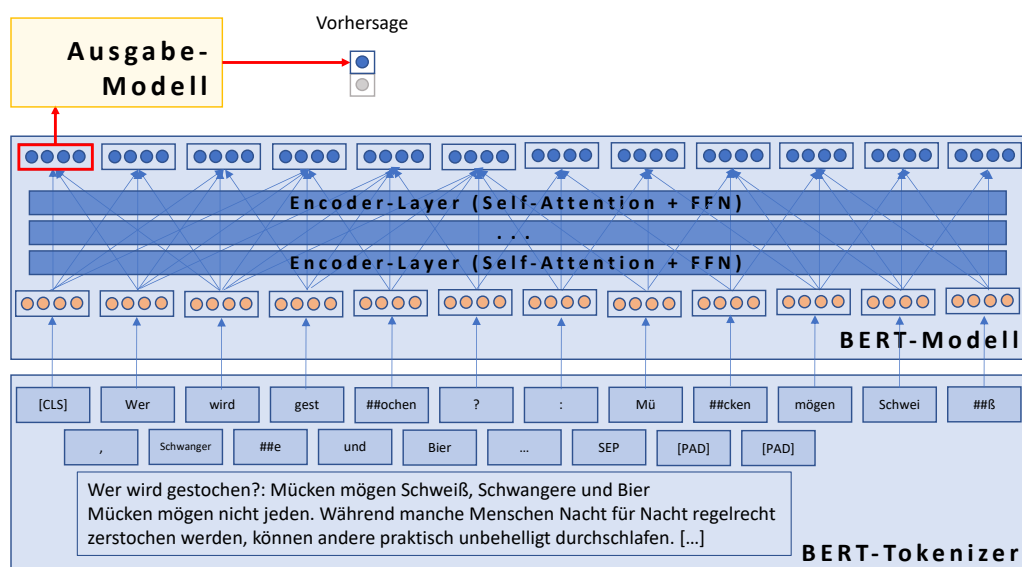


Abbildung 6: Schematische Darstellung von BERT und einem Ausgabe-Modell.

Zu Beginn der Verarbeitung steht der BERT-Tokenizer, der die Eingabesequenz in ihre Einzelteile zerlegt. Im Gegensatz zu der etwa bei BOW angesprochenen Tokenisierung verfolgt er dabei eine Subtokens-Strategie, längere morphologisch komplexere Wörter werden aufgespalten und als einzelne Tokens weiterbehandelt. So werden beispielsweise aus *Sonnenuntergang* die Tokens [*Sonnen*, *##unter*, *##gang*], die Verwendung von *##* signalisiert dabei den Subtokencharakter. Diese Subtokens entsprechen also den Wörtern/Types w_j in Gleichung 3, aus denen ein Dokument besteht und die schließlich das Vokabular V bilden. Diese Verwendung von Subtokens hat unter anderem den Vorteil, dass so die Anzahl an unbekannten Tokens deutlich verkleinert wird. Eine weitere Eigenheit des BERT-Tokenizers ist das Hinzufügen von besonderen Tokens (*Special Tokens*). So wird zu Beginn der Eingabe das CLS-Token (*Classification Token*) hinzugefügt, dessen Repräsentation relevant wird, wenn die

Problemstellung eine gesammelte Repräsentation der vollständigen Eingabe erfordert, wie etwa bei der Textklassifikation. Zusätzlich zum CLS-Token gibt es Tokens, die Satzgrenzen markieren (SEP), sowie ein Padding-Token (PAD). Die so entstandenen Tokens werden in einer Embeddingschicht in Wortvektoren überführt, die als Eingabe in die erste Encoder-Schicht dienen.

In Abb. 6 ist bei der Darstellung der Tokens und ihrer Embeddings ein Detail unterschlagen, nämlich die Positionsempfindungen. Im Gegensatz zu etwa RNN-Architekturen verfügen Transformer und ihre Self-Attention-Mechanismen über keine Möglichkeit, die Reihenfolge oder Nachbarschaften der Tokens in der Eingabe zu beachten und in die Repräsentation aufzunehmen. Vor der Eingabe der Wortvektoren in die erste Encoder-Schicht werden die Wortvektoren daher zu gelernten Positionsvektoren addiert, die Information über die Position des Tokens in der Eingabe enthalten. Diese summierten Vektoren dienen dann als Eingabe in die Encoder-Schichten, die auf den im vorherigen Abschnitt behandelten Transformer-Mechanismen beruhen³².

Das BERT_{BASE}-Modell, das in dieser Arbeit verwendet wurde, enthält 12 Encoder-Schichten, die Dimension der Repräsentationen (*Hidden States*) beträgt $H = 768$, die Anzahl an Attention Heads ist $A = 12$. Insgesamt enthält BERT_{BASE} damit 110 Mio. Parameter.

Nach diesem Überblick über die BERT-Architektur wird nun auf das **Pretraining** des Modells genauer eingegangen. Dies geschieht auf einem sehr großen Datensatz. Trainiert wurde das Modell anhand von zwei Trainingsaufgaben, die auf allgemeines Sprachwissen abzielen. Bei der ersten handelt es sich um ein maskiertes Sprachmodell (*Masked Language Model*): Das Modell erhält dabei Sätze aus dem Trainingskorpus, bei dem manche Tokens absichtlich (zufällig) maskiert sind, und soll nun aus dem Kontext das Wort „erraten“. Auf diese Weise lernt das Modell viel über semantische Beziehungen zwischen benachbarten Wörtern. Die zweite Aufgabe ist die *Next Sentence Prediction*: Das Modell erhält zwei Sätze aus dem Korpus und soll klassifizieren, ob es sich um aufeinanderfolgende Sätze handelt oder nicht (also um zwei unabhängige Sätze aus dem Korpus). Es handelt sich also um einen unüberwachten Lernprozess, für beide Aufgaben ist kein händisches Erstellen der Trainingsdaten nötig, da sie automatisiert aus dem Textkorpus erstellt werden können. Die Gewichte, die BERT anhand dieser Hilfsaufgaben trainiert und erlernt hat, enthalten allgemeines sprachliches Wissen über Wortbedeutungen und Zusammenhänge innerhalb von Sequenzen.

³²Neben den Wortembeddings und den Positionsempfindungen gibt es noch einen dritten Summanden, sogenannte Segmentembeddings. Diese sind vor allem für Anwendungen, die auf der Eingabe von zwei Sätzen beruhen (etwa aus dem Bereich der *Natural Language Inference*, NLI), von Bedeutung und geben Auskunft darüber, aus welchem der Sätze das Token entstammt. Die Eingabe in das BERT-Modell ist also die Summe dieser drei Embeddings. Vereinfachend wird hier trotzdem von Wortvektoren als Eingabe gesprochen.

Im anschließenden **Fine-Tuning**-Prozess kann dann eine spezifische Ausgabeschicht hinzugefügt werden, die die eigentliche Problemstellung mithilfe der abstrakten BERT-Repräsentationen zu lösen erlernt (*Transfer-Learning*) und Vorhersagen der gewünschten Form liefert. Bei diesen Downstream-Aufgaben kann es sich beispielsweise um Textklassifikation (etwa: Welches Thema behandelt der Text?) oder -regression (etwa die Vorhersage der Emotion(en), die im Text vermittelt sind), oder um Tagging (etwa Wortartenerkennung, *Part-of-Speech-Tagging*) handeln. Für Textklassifikation oder -regression wird dabei in der Regel – wie in Abb. 6 dargestellt – die Vektorrepräsentation des CLS-Tokens als Repräsentation des gesamten Eingabetextes verwendet. Für andere, wortbasierte Arten von Fragestellungen, etwa *Part-of-Speech-Tagging*, *Named Entity Recognition* oder auch Übersetzung, werden hingegen die einzelnen Tokenrepräsentationen verwendet. Gleiches gilt auch bereits im Pretraining anhand von Next-Sentence-Prediction (CLS-Token) und Masked-Language-Model (spezifische Tokens). Auf einer vergleichsweise geringen Menge an bezüglich der *Downstream*-Aufgabe annotierten Trainingsdaten kann das Modell dann weitertrainiert werden und sowohl die BERT-Gewichte, vor allem aber die Gewichte der Ausgabeschicht anpassen.

5 Modelle zur Vorhersage der Aufenthaltsdauer

Im vorigen Abschnitt wurden allgemeine Konzepte und Modelle beschrieben, die für die vorliegende Arbeit und ihre Modellierung der durchschnittlichen Aufenthaltsdauer eines Artikels relevant sind. Im Folgenden werden nun diese Konzepte aufgegriffen, um die spezifischen Modelle vorzustellen. Dabei wird auf die Architekturen, die nötigen Schritte in der Featureerstellung, die Implementierungsdetails und die Entscheidungen über Hyperparameter eingegangen. Die fettgedruckten Kürzel der Modelle erleichtern die Zuordnung in der Tabelle mit den Evaluationsergebnissen (Tabelle 4). Zunächst werden einige unterschiedliche Baselines vorgestellt, bevor mit den BERT-Modellen der eigentliche Fokus dieser Arbeit erreicht wird.

5.1 Baseline 1 und 2: Mittelwert und Textlänge

Für die Einordnung und Interpretation der Evaluationsergebnisse der fortgeschritteneren Modelle bietet sich der Vergleich mit zwei sehr einfachen Baselines an. Verwendet wurde hier eine einfache Mittelwert-Baseline (**BaselineMean**), die stets den Mittelwert der Aufenthaltsdauer im Trainingsdatensatz vorhersagt, den Artikeltext also in keiner Weise berücksichtigt.

Die zweite einfache Baseline bedient sich der bereits mehrfach (auch kontrovers) diskutierten Annahme über einen Zusammenhang der Aufenthaltsdauer mit der Textlänge des Artikels, dass längere Texte also zu einer längeren mittleren Aufenthaltsdauer führen. Als eine weitere Baseline wurde also eine univariate lineare (Ridge) Regression mit der Tokenanzahl im Artikeltext als Eingabe verwendet (**BaselineTextlength**). Auch hier spielt also der semantische Gehalt des Artikeltextes keine Rolle. Zwar ist, wie Abschnitt 3.3 bereits gezeigt hat – die Korrelation zwischen Textlänge und Aufenthaltsdauer nicht besonders stark, trotzdem bietet die Baseline einen guten Vergleichspunkt für komplexere Modelle.

5.2 BOW-Modelle

Für eine solide *Bag-of-Words*-Baseline wurden zwei verschiedene Modelle verwendet. Beide basieren auf einfachen BOW-Merkmalen, also absoluten Häufigkeiten der Tokens bzw. N-Gramme in den Dokumenten aus dem Trainingsdatensatz. Wie in Abschnitt 4.1 beschrieben, verfügt der BOW-Ansatz über keinerlei Information zur Abfolge der Tokens/N-Gramme im Text. Die Entscheidung fiel für absolute (statt

relative oder binäre) Häufigkeiten, da diese die Textlänge implizit enthalten, die für die Modellierung der Aufenthaltsdauer mutmaßlich relevant ist³³.

Mit diesen Features wurde eine Ridge Regression (**BOWRidge**) und ein XGBoost-Regressor (**BOWXGBoost**) trainiert. Die Präprozessierungsschritte bei der Feature-Erstellung unterscheiden sich teilweise, es wurden verschiedene Kombinationen ausprobiert und auf dem Entwicklungsdatensatz getestet und daran ausgewählt.

Dabei stellten sich die folgenden Spezifikationen heraus. Für beide Modelle wurden Interpunktionszeichen der Artikeltexte entfernt und die Tokens lemmatisiert und in Kleinschreibung überführt³⁴. Bei beiden Modellen wurden Stopwörter entfernt. Nicht nur einzelne Tokens (Unigramme) wurden verwendet, sondern auch N-Gramme der Länge 1 bis 4, die in mindestens fünf Dokumenten im Trainingskorpus vorkamen. Die Features der beiden Modelle unterscheiden sich aber in der Anzahl der maximal betrachteten Tokens bzw. N-Gramme: Für die Ridge Regression wurden nur die häufigsten 1000 Tokens bzw. N-Gramme betrachtet, für das XGBoost-Modell jedoch die häufigsten 10000. Da der Fokus der Arbeit auf den DL-Modellen liegt, wurde bei den BOW-Modellen nicht so viel Zeit in die Entscheidungen über die (insbesondere bei XGBoost) große Menge an Hyperparametern investiert, sondern mit wenigen Ausnahmen die Default-Parameter beibehalten.

5.3 Baseline 3: Dokumentenembeddings

Im Gegensatz zu den beiden Baselines, die den Inhalt des Artikels nicht beachten, verwenden die hier beschriebenen Baseline-Modelle den Artikeltext selbst in ihrer Modellierung. Der Artikel wird hier als Vektor fester Länge, also als Dokumentenembedding repräsentiert. Verwendet wurden zwei Arten von Dokumentenembeddings, sowie zwei Modellarten, woraus sich in der Kombination vier verschiedene Modelle ergeben.

Die ersten Dokumentenembeddings verwenden den in 4.2 beschriebenen Ansatz, der die Mittelung der einzelnen Wortvektoren aller Tokens eines Dokuments vorsieht (*Bag-of-Vectors*, *Zentroid*, siehe Gleichung 12). Verwendet wurden die deutschen fastText-Vektoren von Grave et al. (2018) als Eingabe, die auf Common Crawl und

³³In Entwicklungsexperimenten wurden neben absoluten, relativen und binären Häufigkeiten auch mit der Verwendung von Tf-Idf-Merkmalen experimentiert sowie die Obergrenze der Feature-Anzahl variiert oder Stopwörter beibehalten. Die besten Ergebnisse auf dem dev-Set wurden aber mit der Version mit den beschriebenen Spezifikationen erzielt, und diese wurde somit in der endgültigen Version verwendet.

³⁴Verwendet wurde für Präprozessierung, Featureerstellung und Modellierung die Python-Bibliothek scikit-learn (Pedregosa et al., 2011), sowie der scikit-learn-Wrapper der originalen xgboost-Bibliothek (T. Chen & Guestrin, 2016), außerdem NLTK und Spacy. Die Ridge Regression verwendet die default-Parameter ($\alpha = 0.1$), das XGBoost-Modell hat folgende Parameter: $n_estimators = 300$, $max_depth = 10$, $learning_rate = 0.1$.

der deutschen Wikipedia trainiert wurden³⁵. Es handelt sich um Vektoren der Dimension 300. Das einfache Mitteln der Wortvektoren lässt die Struktur bzw. die Ordnung der Tokens im Text außer Acht. Für das Ermitteln der einzelnen Wortvektoren wurden Interpunktionszeichen und Stopwörter entfernt (diese hätten das Erkennen inhaltlicher Unterschiede der Artikeltexte vermutlich behindert), auf Lemmatisierung wurde hier aber verzichtet, da die fastText-Vektoren auch Flexionsformen mit unterschiedlichen Vektoren enthalten.

Der zweite Ansatz zur Erstellung von Dokumentenembeddings besteht darin, BERT für die Feature-Extraktion, aber nicht für die Weiterverarbeitung bzw. Training und Fine-Tuning zu verwenden. BERT wird hier also *nur* als Feature-Extractor verwendet, es findet keine Feinjustierung (*Fine-Tuning*) auf dem ██████-Datensatz statt. Verwendet wird der *Last Hidden State* des CLS-Tokens als abstrakte Repräsentation des Textes, die Länge des Vektors ist 768.

Mit diesen beiden Versionen von Dokumentenembeddings wurde jeweils eine Ridge Regression trainiert: **EmbsAvgRidge** und **EmbsBertRidge**. Verwendet wurde – wie auch bei den anderen Baselines – die Python-Bibliothek scikit-learn mit Default-Parametern (Pedregosa et al., 2011).

Mit den selben Featurevektoren als Eingabe wurde auch jeweils ein einfaches Feed-Forward Netz trainiert (**EmbsAvgFFN** und **EmbsBertFFN**). Das FFN umfasst zwei Hidden Layer (mit 256 bzw. 64 Neuronen), Dropout von 0.3 sowie LeakyReLU als Aktivierungsfunktion. Die Batch Size betrug 32, Learning Rate war $l = 0.001$, verwendet wurde der AdamW-Optimizer. Beide Modelle trainierten 60 Epochen lang.

Diese beiden, sowie auch sämtliche andere neuronalen Modelle dieser Arbeit, wurden in Pytorch (Paszke et al., 2019)³⁶ implementiert. Alle DL-Modelle dieser Arbeit wurden so lange trainiert, bis auf dem dev-Set keine Verbesserungen mehr zu beobachten waren. Für die Evaluation (siehe Abschnitt 6 und die dortigen Evaluationsmaße) wurde daraus der *beste* Modellzustand (Checkpoint) ausgewählt.

5.4 CNN-Baseline

Die CNN-Baseline (**CNN800**) verwendet erneut die deutschen Wortvektoren von fastText (Grave et al., 2018). Statt der Mittelung der Wortvektoren wird der Text hierfür aber sequenziell in einer Eingabematrix X repräsentiert, bei der die einzelnen Wortvektoren die Zeilen darstellen (siehe Abschnitt 4.4.2 und die dortige Formel 16)³⁷.

³⁵<https://fasttext.cc/docs/en/crawl-vectors.html>

³⁶<https://pytorch.org/>

³⁷Aus Prozessierungsgründen wurde eine maximale Textlänge von 800 Tokens verwendet. Längere Texte wurden also trunziert, kürzere gegebenenfalls mit Nullvektoren gepaddet, abhängig von der maximalen Textlänge der Minibatch im Trainingsprozess.

Die Modellarchitektur beinhaltet jeweils 32 Filter der Größen 2 bis 5, Max-Pooling, Dropout von 0.2 und LeakyReLU als Aktivierungsfunktion. Die Ausgaben der Convolution-Maxpooling-Layer wurden konkateniert und durch eine FFN-Schicht (1 Hidden Layer der Größe 64) auf die skalare Vorhersage der Aufenthaltsdauer transformiert. Trainiert wurde das CNN-Modell 50 Epochen lang mit einer Batch Size von 32 und einer Learning Rate von $l = 0.0002$ (auch hier mit AdamW als Optimizer).

5.5 Einfache BERT-Modelle

Wenden wir uns nun den BERT-Modellen, dem Fokus dieser Arbeit, zu. Als einfache BERT-Modelle wurden drei verschiedene Varianten verwendet, die alle auf einem vortrainierten deutschen BERT_{BASE}-Modell basieren und sich lediglich in der Weiterverarbeitung des BERT-Outputs unterscheiden. Die drei Varianten sind schematisch in Abb. 7 dargestellt (in Anlehnung an Abb. 6 zur BERT-Grundarchitektur).

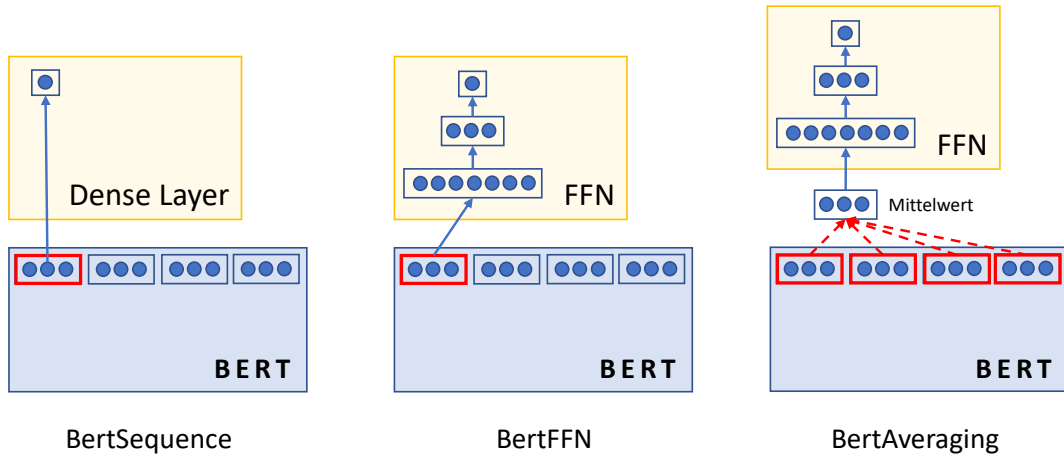


Abbildung 7: Architekturen von BertSequence, BertFFN und BertAveraging.

Die erste Variante (**BertSequence**) ist die Standardverwendung von BERT auf Regressionsprobleme. Dabei wird der letzte Hidden State des CLS-Tokens des BERT-Outputs durch ein Lineares Layer ohne Aktivierungsfunktion auf eine eindimensionale Ausgabe transformiert. Damit stellt diese Variante die direkteste und unveränderte Version des BERT-Outputs dar.

Die zweite Variante (**BertFFN**) setzt auf den BERT-Output – wieder wird der letzte Hidden State des CLS-Tokens verwendet – stattdessen ein Feed-Forward Netz an. Die Architektur umfasst hier (wie in den FFN-Komponenten aller Modelle, bereits bei EmbsBertFFN beschrieben) 2 Layer (der Größe 256 und 64), LeakyReLU und Dropout von 0.3.

Die dritte Variante (**BertAveraging**) ist der zweiten (BertFFN) sehr ähnlich. Sie verwendet aber als BERT-Output nicht den letzten Hidden State des CLS-Tokens, sondern mittelt die letzten Hidden States *aller* Tokens der Eingabe. Der anschließende FFN-Modellteil ist identisch zu dem der zweiten Variante.

Da BERT-Modelle aufgrund der komplexen Attention-Berechnungen eine feste Längenbeschränkung von 512 (BERT-)Tokens haben, verwenden alle diese drei Modelle ausschließlich den Beginn des Artikeltextes (maximal 512 BERT-Tokens) und lassen den Rest des Artikels außer Acht. In Entwicklungsexperimenten wurde versuchsweise auch der letzte Abschnitt bzw. ein zufälliger Abschnitt (*random window*) gleicher Länge als Eingabe verwendet, ebenso wurde mit dem zufälligen Maskieren einzelner Tokens im Text experimentiert. Die Vorhersage auf dem dev-Set erzielte aber bei Verwendung des Textbeginns und unmaskierter Eingabe deutlich bessere Ergebnisse. Intuitiv ergibt die Fokussierung auf den ersten Abschnitt eines Artikels Sinn, auch andere Arbeiten machen diese Beobachtung (siehe Forschungsüberblick). Schließlich entscheidet sich der oder die Lesende in der Regel anhand des Textanfangs, ob der Artikel lesenswert ist oder nicht. Es scheint also durchaus angemessen, den Textanfang als Repräsentation für den gesamten Artikel und die Abschätzung seiner Popularität zu verwenden.

Verwendet wurde das vortrainierte deutsche BERT_{BASE}-Modell „bert-base-german-cased“ der transformer-Bibliothek von Huggingface (Wolf et al., 2020)³⁸, die eine Vielzahl an vortrainierten BERT-Modellen für verschiedene Sprachen zur Verfügung stellt³⁹.

BertSequence wurde mit einer Learning Rate von $l = 10^{-5}$ trainiert, bei BertFFN und BertAveraging wurde bei der Learning Rate zwischen den verschiedenen Komponenten unterschieden: Die BERT-Teile der Architektur (also das originale Huggingface-Modell) wurden mit $l = 10^{-5}$ trainiert, das anschließende FFN hingegen mit $l = 0.001$, da diese im Gegensatz zum BERT-Encoder nicht vortrainiert sondern zufällig initialisiert sind. Trainiert wurde für ca. 20 Epochen, die Modelle erreichten ihre Bestwerte allerdings schon nach ca. 5 Epochen.

5.6 BERT erweitert mit Textlänge

Das Modell **BertTextlength** ist im Wesentlichen eine Erweiterung von BertFFN. Der Unterschied ist, dass das an BERT anschließende FFN neben der BERT-Ausgabe (dem letzten Hidden State des CLS-Tokens) auch die Textlänge des Artikels als

³⁸<https://github.com/huggingface/transformers>

³⁹Neben BERT gibt es dort noch viele weitere Transformer-Modelle, etwa Erweiterungen wie RoBERTa, oder XLNet und GPT-2. Für viele dieser Modelle gibt es aber keine oder kaum deutsche vortrainierte Modelle, was für diese Arbeit die Voraussetzung war. Außerdem ist BERT nach wie vor eines der meistgenutzten Transformer-Modelle für Textklassifikation bzw. -regression.

numerische Eingabe erhält⁴⁰. Da es sich bei der Textlängeninformation um einen Skalarwert handelt, wurde eine einfache Konkatenation an den BERT-Ausgabevektor gewählt. Das Modell vereint also textuelle Features und Metadaten. Auch wenn in dieser Arbeit lediglich die Textlänge Verwendung gefunden hat, so illustriert der verfolgte Architekturansatz viele weitere Möglichkeiten für Folgearbeiten. Mithilfe von Embeddingschichten könnten etwa in anschließenden Arbeiten auch weitere Metadaten – sowohl numerischer als auch kategorialer Form – einfließen.

Die Architekturspezifika sind die gleichen wie bei BertFFN (2 Layer mit 256 bzw. 64 Hidden Units, Dropout von 0.3, LeakyReLU), auch hier wurde bei der Learning Rate zwischen den BERT-Teilen ($l = 10^{-5}$) und dem FFN ($l = 0.001$) unterschieden. Trainiert wurde für 20 Epochen, auch hier war nach ca. 5 Epochen keine Verbesserung mehr zu beobachten.

5.7 Hierarchische BERT-Modelle

Wie bereits beschrieben, ist die Anwendung von BERT oder anderen Transformer-Architekturen auf lange Dokumente problematisch, da die Rechenzeit des Self-Attention-Mechanismus mit quadratischer Komplexität bezüglich der Eingabelänge steigt. BERT ist daher auf Eingaben mit einer maximalen Länge von 512 Tokens beschränkt⁴¹.

Das Umgehen dieser Problematik ist ein aktuell sehr aktives Forschungsfeld (Pappagari et al., 2019; Grail et al., 2021; Dai et al., 2019; Zaheer et al., 2020; Beltagy et al., 2020; Qiu et al., 2020). Der einfachste Weg ist das schlichte Trunkieren der Dokumente und die Betrachtung des Textanfangs (wie es die bisher beschriebenen BERT-Modelle dieser Arbeit tun) oder eines anderen Teils des Textes. Ein anderer Ansatz besteht in der Segmentierung des Dokuments in Abschnitte kürzerer Länge, die hierarchisch verarbeitet und kombiniert werden⁴². Diesen Ansatz verfolgen auch die beiden hier beschriebenen Modelle, deren Architekturen ähnlich sind wie der bei Pappagari et al. (2019) verwendete Ansatz: BertHiMean und BertHiGRU (*Hi* steht für *hierarchisch*) wenden BERT abschnittsweise hierarchisch an und kombinieren

⁴⁰Als Maß für die Artikellänge wurde die Anzahl an Tokens (ohne Interpunktion) verwendet. Versuchsweise wurde auch die Anzahl an BERT-Tokens verwendet, diese Variante erzielte aber auf den dev-Daten schlechtere Werte.

⁴¹Es handelt sich um 512 BERT-Tokens, also bei einer konventionellen Tokenzählung um Texte mit weitaus weniger Tokens.

⁴²Andere komplexere Ansätze verarbeiten nicht Segmente zunächst getrennt voneinander (oder trennen sie mit Überlappung), sondern bedienen sich stattdessen einer *sliding-window*-Architektur wie etwa der Longformer von Beltagy et al. (2020), der auch in der Bibliothek von Huggingface enthalten ist, für den allerdings keine deutschen vortrainierten Modelle vorliegen. Andere (Grail et al., 2021) wenden stattdessen RNN-Module nicht erst im Anschluss an die Abschnittsrepräsentationen, sondern bereits innerhalb der Attention-Blocks im Transformer-Modell an, sodass Information bereits innerhalb des Encoders über die Abschnitte hinweg propagiert werden kann.

anschließend die Ausgaben. Der Text wird in Abschnitte fester Länge (≤ 512 BERT-Tokens) eingeteilt und für jeden Abschnitt wird eine Vorhersage bzw. Repräsentation erstellt⁴³. Die beiden Varianten unterscheiden sich in der Art, wie sie die einzelnen Abschnitte zu einer finalen Vorhersage kombinieren.

Das erste hierarchische Modell (**BertHiMean**, Abb. 8) sagt für jeden Abschnitt eine Aufenthaltsdauer vorher, also einen Vektor $\hat{y} = (\hat{y}_1, \hat{y}_2, \dots, \hat{y}_N)$, wobei N die Anzahl der Abschnitte ist. Aus diesen Ergebnisse wird dann mithilfe einer gewichteten Mittelung der endgültige Wert \hat{y}_w für den gesamten Text berechnet:

$$\hat{y}_w = \frac{\sum_{i=1}^N w_i \hat{y}_i}{\sum_{i=1}^N w_i} = \frac{w \cdot \hat{y}}{\sum_{i=1}^N w_i} \quad (22)$$

wobei $w = (w_1, w_2, \dots, w_N)$ der Gewichtungsvektor ist.

Der Gewichtungsvektor ist dabei ein Parameter des Modells, die spezifische Gewichtung der Abschnitte wird also im Trainingsprozess gelernt. Die Architektur bei den einzelnen Abschnittsvorhersagen ist dabei identisch zu dem bereits beschriebenen BertFFN-Modell. Statt der gewichteten Mittelung der einzelnen Abschnittsvorhersagen wurde auch mit einer gewichteten Summierung experimentiert, dieser Ansatz ist vielleicht etwas intuitiver, da sich die Aufenthaltsdauer eines Textes unmittelbar aus den Dauern der einzelnen Abschnitte zusammensetzt. Die Qualität der Vorhersagen dieser Variante war vergleichbar. Trotzdem wurde schlussendlich die Mittelung vorgezogen, da diese Variante das Laden von Parametern aus bereits trainierten Modellen sinnvoller ermöglicht (siehe unten).

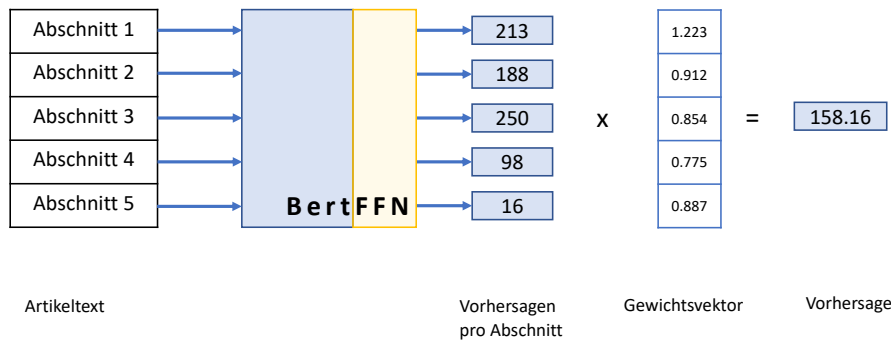


Abbildung 8: Architektur von BertHiMean: Vorhersage ist gewichteter Mittelwert der Abschnittsvorhersagen.

Die andere Version (**BertHiGRU**, Abb. 9) verfolgt einen leicht anderen Ansatz. Auch hier werden die einzelnen Abschnitte zunächst getrennt voneinander in eine BERT-Repräsentation überführt, dafür dient der letzte Hidden State des CLS-

⁴³Es handelt sich hier nicht um *Abschnitte* im typografischen Sinne, sondern schlicht um Tokensequenzen fester Länge.

Tokens jedes Abschnitts. Anschließend werden diese Abschnittsrepräsentationen durch eine RNN-Komponente in eine finale Dokumentenrepräsentation überführt. In Entwicklungsexperimenten stellte sich hierfür ein GRU-Modell als beste Variante heraus. Verwendet wurden zwei Schichten, die Größe der Hidden States wurde bei 768 (wie die BERT-Outputs) belassen. Als Dokumentenrepräsentation wurde sich für den letzten Hidden State vom letzten Zeitschritt der letzten (also zweiten) Schicht entschieden. Anschließend erfolgt – analog zu den anderen BERT-Modellen – ein 2-Layer-FFN (Hidden Units 256 bzw. 64, Dropout von 0.3, LeakyReLU), um den Vektor in eine skalare Vorhersage zu überführen.

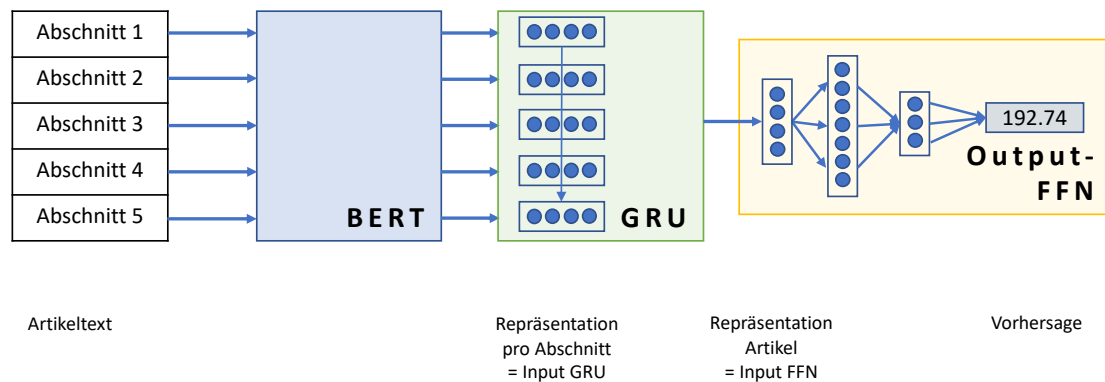


Abbildung 9: Architektur von BertHiGRU: Erstellen einer Gesamtrepräsentation mit anschließender Vorhersage.

Die Learning Rate im Training betrug erneut $l = 10^{-5}$ für die BERT-Parameter und $l = 0.001$ für die FFN-Parameter. Der Gewichtungsvektor (BertHiMean) wurde mit $l = 0.001$ gelernt, bzw. die GRU-Parameter (BertHiGRU) mit $l = 10^{-5}$.

Bezüglich der Wahl von Abschnittsgröße und -anzahl wurden verschiedene Kombinationen ausprobiert. In der Ergebnistabelle sichtbar sind Versionen mit a) einer Abschnittsgröße von 512 Tokens und einer maximalen Abschnittszahl von 5 (Modellkürzel mit ***512**) und b) einer Abschnittsgröße von 400 Tokens und einer maximalen Abschnittszahl von 6 (Modellkürzel mit ***400**).

Trainiert wurde für mindestens 20 Epochen, nach ca. 10 Epochen zeigte sich auf den dev-Daten keine Verbesserung mehr bzw. es konnten Tendenzen von Overfitting beobachtet werden.

Neben den „normalen“ Trainingsläufen mit zufällig initialisierten Parametern (abgesehen natürlich von den nach wie vor vortrainierten BERT-Gewichten der Huggingface-Modelle) wurden bei beiden hierarchischen Modellen auch Trainingsprozesse durchgeführt, bei denen zu Beginn Gewichte aus bereits trainierten BertFFN-Modellen geladen werden. Dies ist möglich, da die hierarchischen BERT-Modelle

und die BertFFN-Modelle bei der Repräsentation der Abschnitte die gleiche Architektur (BERT-Ausgabe + anschließendes FFN) aufweisen. Diese Entscheidung lässt sich intuitiv begründen: Die trainierten BertFFN-Modelle sind bereits darauf spezialisiert, die Texteingaben so zu repräsentieren, dass sie für die Vorhersage einer Aufenthaltsdauer geeignet sind. Dieses Vorgehen wird nun statt nur auf den ersten auf *alle* Abschnitte angewendet. Das Nutzen dieses „Grundwissens“ ermöglicht – so die Annahme – den hierarchischen Modellen ein schnelleres und vielleicht sogar erfolgreicheres, weil spezifischeres Training. Vollständig neu trainiert werden müssen also bei diesen hierarchischen BERT-Modellen mit Pretraining (**BertHiMean*_pre** und **BertHiGRU*_pre**, * meint die unterschiedliche Wahl der Abschnittsgröße) der Gewichtungsvektor bzw. die GRU-Parameter. Die BERT-Parameter sind aber auch hier nicht eingefroren, sondern werden weiter angepasst. Verwendet wurden jeweils Gewichte aus BertFFN-Modellen, die im Training Textlängen betrachtet hatten, die der Abschnittsgröße des hierarchischen Modells entsprechen (also 400 bzw. 512 Tokens).

Tatsächlich starteten die Modelle mit den BertFFN-Gewichten bereits mit besseren Evaluationswerten und erreichten im Training deutlich schneller ihre Bestwerte, sie wurden für mindestens 8 Epochen trainiert, erreichten aber bereits nach 1 bis 2 Epochen ihren Bestwert.

6 Ergebnisse

Dieser Abschnitt erläutert zunächst die verwendeten Evaluationsmetriken zur Modell-evaluation. Anschließend werden die Ergebnisse der einzelnen Modelle sowohl auf dem Entwicklungs- als auch dem Testdatensatz dargestellt und die Kernergebnisse beschrieben sowie ein kurzer Vergleich zu Ergebnissen anderer Arbeiten gegeben.

6.1 Metriken zur Modellevaluation

Da es sich bei der gewählten Zielvariablen – Aufenthaltsdauer in Sekunden – um eine kontinuierliche Größe handelt, liegt ein Regressionsproblem vor. Verglichen werden die Vorhersagen mit den echten Werten im zu testenden Datensatz (wahlweise Entwicklungs- oder Testdaten). Als Gütekriterium der Modellqualität wird primär der Pearson-Korrelationskoeffizient r zwischen Vorhersagen und echten Werten berechnet:

$$r = \frac{\sum_{i=1}^N (t_i - \bar{t})(p_i - \bar{p})}{\sqrt{\sum_{i=1}^N (t_i - \bar{t})^2 \sum_{i=1}^N (p_i - \bar{p})^2}} \quad (23)$$

N bezeichnet die Anzahl an betrachteten Vorhersagen (also die Größe des Entwicklungs- bzw. Testdatensatzes), p_i die Vorhersage bei Datenpunkt i (*Prediction*), t_i den echten Wert am Datenpunkt i (*True Value*), und \bar{p} bzw. \bar{t} den Mittelwert der Vorhersagen bzw. der wahren Werte in den Testdaten.

Weiter werden als Fehlermaße der mittlere absolute Fehler (*Mean Absolute Error*, MAE, Formel 24) und der mittlere quadratische Fehler (*Mean Squared Error*, MSE, Formel 25) betrachtet. Zusätzlich wird der Relative Absolute Fehler (*Relative Absolute Error*, RAE, Formel 26) angegeben. Dabei handelt es sich um den Anteil des summierten absoluten Fehlers der Vorhersagen an der Summe der absoluten Schwankungen zum Mittelwert der echten Werte⁴⁴.

$$MAE = \frac{1}{N} \sum_{i=1}^N |t_i - p_i| \quad (24)$$

$$MSE = \frac{1}{N} \sum_{i=1}^N (t_i - p_i)^2 \quad (25)$$

⁴⁴Intuitiv: Ein Modell wird daran bewertet, einen wie großen Anteil des Fehlers es zu reduzieren vermag im Vergleich zu einem Modell, das stets den Mittelwert der Zielvariablen vorhersagt.

$$RAE = \frac{\sum_{i=1}^N |t_i - p_i|}{\sum_{i=1}^N |t_i - \bar{t}|} \quad (26)$$

Alle in Abschnitt 5 vorgestellten Modelle wurden anhand des ██████-Datensatzes trainiert, die Aufteilung in Trainings-, Entwicklungs- und Testdaten wurde dabei konstant gehalten. Trainiert wurde also immer auf den gleichen Trainingsdaten.

Der Trainingsfortschritt der neuronalen Modelle wurde anhand der beschriebenen Metriken auf dem Entwicklungsdatensatz überwacht⁴⁵. Als Loss-Funktion der DL-Modelle wurde der mittlere quadratische Fehler (MSE) verwendet, zur Selektion der besten Modellzustände wurde aber die Pearson-Korrelation verwendet⁴⁶. Es wurde im Training also stets der aktuelle Modellzustand (*Checkpoint*) gespeichert, wenn ein größerer Korrelationskoeffizient zwischen Vorhersagen und wahren Werten (auf den dev-Daten) beobachtet wurde als zuvor.

6.2 Evaluationsergebnisse

In Tabelle 4 sind die Evaluationsergebnisse der Modelle sowohl für das Entwicklungsset (dev) als auch das Testset (test) angegeben. Beide Datensätze waren den Modellen im Training nicht zugänglich und bilden daher gut die Fähigkeiten der Modelle ab, auf ungesehenen Daten Vorhersagen zu machen. In der Tabelle sind für beide Datensätze zunächst die vier beschriebenen Metriken (Pearsons r , MSE, MAE, RAE) angegeben, die ein Modell (Kürzel in den Zeilennamen) in der Evaluation erzielt. Dabei gilt: Ein Modell ist desto besser, je größer der Pearson-Korrelationskoeffizient r , je kleiner hingegen die Fehlerwerte (MSE, MAE, RAE) sind.

Wie Tabelle 4 zeigt, stimmen die Metriken in ihrer Sortierung der Modelle nicht unbedingt überein, was einen Schluss auf die Gesamtqualität der Modelle erschwert. Für einen besseren Überblick wurde daher zunächst innerhalb jeder Metrik ein Ranking der Modelle erstellt. Diese Einzelränge wurden anschließend jeweils innerhalb der beiden Datensätze pro Modell gemittelt und daraus ein Gesamtranking erstellt, das in den beiden Spalten *rank* vorliegt. Je kleiner die Rangplatzierung, desto besser das Modell. Sortiert wurde in Tabelle 4 schlussendlich nach dem Ranking auf dem Testdatensatz (also anhand der Spalte ganz rechts).

⁴⁵Verwendet wurde dafür das Tool Tensorboard zur grafischen Visualisierung von Modellexperimenten. Tensorboard ist eigentlich Teil des Tensorflow-Frameworks, doch auch Pytorch stellt eine Einbindung bereit: <https://pytorch.org/docs/stable/tensorboard.html>

⁴⁶Die Entscheidung dazu erklärt sich dadurch, dass Korrelation unabhängiger von datensatzspezifischen Größen (im Vergleich etwa zum absoluten Fehler) ist, was den Vergleich über *unterschiedliche* Datensätze ermöglichen würde, wenn dies auch durch den Fokus auf den ██████-Datensatz weniger relevant wurde.

Modell	DEV-SET					TEST-SET				
	r	MSE	MAE	RAE	rank	r	MSE	MAE	RAE	rank
BaselineMean	-	19533	70.2	0.985	21	-	12795	65.9	1.013	21
BertSequence	-	19758	68.3	0.958	20	-	12847	63.7	0.979	20
EmbsAvgRidge	0.365	16954	68.4	0.960	19	0.374	11002	62.6	0.961	19
CNN800	0.461	16497	68.6	0.963	18	0.476	10833	62.8	0.965	18
BOWRidge	0.475	15197	66.8	0.937	17	0.427	10955	62.5	0.960	17
BaselineTextlength	0.389	17832	63.9	0.897	16	0.401	11478	59.5	0.914	16
EmbsBertRidge	0.514	14379	66.5	0.933	15	0.512	9593	60.7	0.933	15
EmbsAvgFFN	0.532	14004	64.1	0.900	14	0.531	9217	58.2	0.895	14
BOWXGBoost	0.500	14657	60.8	0.853	13	0.500	9754	54.7	0.840	13
BertHiMean400	0.708	10884	60.8	0.854	10	0.627	9110	57.1	0.877	12
EmbsBertFFN	0.616	12173	58.7	0.823	12	0.572	8692	53.6	0.823	11
BertAveraging	0.678	10738	54.4	0.763	9	0.604	8529	50.3	0.773	10
BertHiMean512	0.698	10176	54.6	0.765	7	0.643	7994	50.4	0.775	9
BertFFN	0.695	10397	54.7	0.767	8	0.624	7959	49.9	0.766	8
BertHiGRU400pre	0.672	10901	57.1	0.801	11	0.659	7611	51.3	0.789	7
BertHiGRU512pre	0.705	9857	55.7	0.781	6	0.657	7485	51.3	0.788	6
BertTextlength	0.703	9961	54.1	0.760	4	0.657	7673	50.0	0.768	5
BertHiGRU512	0.696	10372	53.5	0.751	3	0.645	7796	49.1	0.754	4
BertHiMean512pre	0.719	9674	54.2	0.761	2	0.655	7486	49.7	0.764	3
BertHiGRU400	0.713	9647	53.9	0.756	1	0.655	7593	49.3	0.758	2
BertHiMean400pre	0.686	10419	53.6	0.752	5	0.649	7409	48.7	0.748	1

Tabelle 4: Evaluationsergebnisse auf dem Entwicklungs- (DEV) und Testdatensatz (TEST), sortiert auf dem **Testdatensatz** nach dem gemittelten Rang über alle vier Metriken (letzte Spalte **rank**).

Die beiden Datensätze (dev und test) weisen einige Unterschiede auf, sowohl bezüglich der Evaluationswerte als auch in der Sortierung der Modelle.

Die absoluten (MAE) und quadrierten (MSE) Fehlerwerte fallen konsistent auf dem Testdatensatz kleiner aus als auf dem Entwicklungsdatensatz. Dies ist durch Unterschiede in der Verteilung der Aufenthaltsdauer innerhalb der beiden Sets zu erklären, die bei der zufälligen Einteilung nicht aktiv verhindert wurden, wie bereits in Abschnitt in 3.3 und der dortigen Tabelle 2 beobachtet wurde. Auf den relativen absoluten Fehler (RAE) sowie auf die Korrelationswerte r wirkt sich dieser Unterschied weniger aus.

Hier zeigt sich hingegen ein anderes erwartbares Phänomen: Die Korrelationswerte der DL-Modelle sind (im Gegensatz zu den Baseline-Modellen und den beiden BOW-Varianten) auf dem Testdatensatz generell etwas niedriger als auf dem Entwicklungsdatensatz. Das ist aus zwei Gründen nicht überraschend: Erstens kam der

Entwicklungsdatensatz bei der Wahl einiger Hyperparameter bereits zum Einsatz, ist also salopp ausgedrückt nicht ganz so „ungesehen“ wie die Testdaten. Der zweite und wohl wichtigere Punkt ist aber: Es wurden im Trainingsprozess ja anhand der Pearson-Korrelation (und nicht anhand einer der Fehler-Metriken) entschieden, ob ein neuer Modellzustand (*Checkpoint*) gespeichert wird. Da die Maße aber nicht immer übereinstimmen, ist damit nicht gewährleistet, dass dies unbedingt dem besten Zustand auch laut einer anderen Metrik entspricht bzw. auf den Testdaten genauso hohe Korrelationswerte erzielt werden.

Insgesamt lassen sich unter Berücksichtigung aller Metriken in Tabelle 4 folgende Ergebnisse zur Modellgüte festhalten:

- Wie zu erwarten, übertreffen alle Modelle die schlichte Mittelwert-Baseline (BaselineMean). Die Regression anhand der Textlänge ohne den Text zu betrachten (BaselineTextlength) ist überraschend gut (Rang 16, $r = 0.39$ bzw. 0.40). Dies überrascht, da (siehe Tabelle 1) im ██████-Datensatz eigentlich nur eine mittelstarke Korrelation von $r = 0.26$ zwischen Textlänge und Aufenthaltsdauer beobachtet wurde. Dies spiegelt aber lediglich die bereits in Abschnitt 3.3 und Tabelle 2 beobachtete Diskrepanz der drei Datensets wider, die Korrelationen sowohl im dev- als auch im test-Set fallen (zufällig) stärker aus.
- Die beiden auf *Bag-of-Words*-Features beruhenden Modelle (BOWRidge und BOWXGBoost) sind beide besser als die einfachen Baselines, werden jedoch deutlich von den BERT-Modellen übertroffen. Die Modellierung mit XGBoost ist deutlich besser als die Ridge Regression (Rang 13 vs. Rang 17). Dies bestätigt auch ein gepaarter zweiseitiger Wilcoxon-Test über die absoluten Fehler der Vorhersagen⁴⁷. BOWRidge ist hingegen kaum besser als die Textlängen-Regression.
- Das CNN mit fastText-Embeddings (CNN800) schneidet ziemlich schlecht ab. Gleiches gilt auch für die Varianten mit der Mittelung der fastText-Wortvektoren und anschließender Ridge Regression (EmbsAvgRidge). Die Variante mit anschließendem FFN (EmbsAvgFFN) hingegen ist vergleichbar mit der XGBoost-Modellierung. Bei beiden Modellklassen führt die Verwendung von (eingefrorenen) BERT-Embeddings zu einer deutlichen Verbesserung (EmbsBertRidge und EmbsBertFFN). Dies illustriert deutlich den Vorteil von kontextualisierten BERT-Embeddings, der sich hier bereits bei der Verwendung als Dokumentenembeddings zeigt.
- Eine zentrale Erkenntnis ist: Die Modelle, die BERT als Komponente enthalten, übertreffen sehr konsistent die einfachen sowie auch die BOW-Baseline-Modelle,

⁴⁷Sowohl auf dem dev- als auch dem test-Set ist $p < 0.001$.

dies zeigt auch ein vergleichender Signifikanztest anhand der absoluten Fehler der Vorhersagen⁴⁸. Die Variante BertFFN ist dabei leicht besser als BertAveraging, die Repräsentation des CLS-Tokens ist also durchaus sinnvoll.

Auffällig ist aber, dass die kanonische Verwendung von BERT auf Regressionsprobleme, bei der an die BERT-Ausgabe lediglich eine lineare Transformation auf eine skalare Ausgabe ansetzt, keine Erfolge verbuchen kann (BertSequence): Der Überwachungsprozess im Training zeigt, dass das Modell lediglich lernt, den Mittelwert vorherzusagen. (Daher steht es in Tabelle 4 auch direkt neben BaselineMean, bei der Vorhersage eines konstanten Wertes kann kein Korrelationskoeffizient berechnet werden.) Der stochastische Lernprozess findet hier (anders als bei Verwendung eines FFNs) keine geeignete lineare Transformation, um die CLS-Repräsentation auf eine Aufenthaltsdauer abzubilden. Dies legt den Verdacht nahe, dass die CLS-Repräsentation zwar für die Vorhersage der Aufenthaltsdauer brauchbare Information enthält, diese aber nur durch komplexe, nichtlineare Transformationen (FFN) extrahiert werden kann.

- Bezüglich des Einbeziehens der Textlänge kann anhand der Evaluationsergebnisse kein eindeutiges Fazit gezogen werden. BertTextlength ist kaum besser als die Architektur ohne Einbezug der Textlänge, BertFFN. Dies deutet darauf hin, dass der erste Abschnitt tatsächlich bereits so indikativ für den Erfolg (Entscheidung des Lesers bzgl. Weiterlesen oder Abbruch) eines Artikels ist, dass die Textlänge kaum noch zusätzliche Hinweise geben kann. Gleichzeitig bietet sich für zukünftige Arbeiten der Einbezug weiterer Metadaten in die Modellierung an, möglicherweise sind andere Daten relevanter.
- Ein weiterer Schluss aus Tabelle 4 ist: Wenn auch nicht völlig konsistent über alle konkreten Modellvarianten, so übertreffen die hierarchischen BERT-Modelle (BertHiMean und BertHiGRU) doch grundsätzlich die einfachen (BertFFN).

Sowohl das erfolgreichste Modell auf dem test-Set (BertHiMean400pre), als auch das auf dem dev-Set (BertHiGRU400) sind dabei signifikant besser als BertFFN, auch hier überprüft anhand zweiseitiger Wilcoxon-Tests auf den absoluten Fehlern⁴⁹.

- Die Modelle, die Wissen über die Textlänge besitzen (BertHiMean, BertHiGRU und BertTextlength), scheinen Ausreißer besser vorherzusagen als das sonst auch

⁴⁸Ein gepaarter zweiseitiger Wilcoxon-Test zeigt einen signifikanten Unterschied zwischen BOWXGBoost und BertFFN, auf beiden Sets ist $p < 0.001$.

⁴⁹BertHiMean400pre: $p < 0.01$ auf dev-Set, $p < 0.01$ auf test-Set; BertHiGRU400: $p < 0.05$ auf dev-Set, $p < 0.01$ auf test-Set.

sehr gute BertFFN. Sie unterscheiden sich in MAE und RAE nur unwesentlich, bei Pearsons r und MSE hingegen stark.

- Eine interessante Beobachtung ist, dass bei den hierarchischen BERT-Modellen nicht unbedingt die besseren Ergebnisse erzielt werden, wenn die volle BERT-Länge von 512 Tokens pro Abschnitt „ausgeschöpft“ wird (vgl. die Modellkürzel mit *400 und *512). Zur Erinnerung: Die hierarchischen Modelle verarbeiten – im Gegensatz zu etwa BertFFN – den *gesamten* Artikeltext⁵⁰, aufgeteilt in kleinere einzeln repräsentierte Abschnitte. Möglicherweise überwiegt dabei die Flexibilität, die Unterschiede der Abschnitte modellieren zu können, gegenüber dem Ausnutzen möglichst langer Textabschnitte. Hier bietet sich eindeutig ein Anknüpfungspunkt für weitere Experimente.
- Zwischen den beiden Varianten der hierarchischen Modelle – Mittelwert der einzelnen Aufenthaltsvorhersagen der Abschnitte (BertHiMean) und Vorhersage anhand einer kombinierten Dokumentenrepräsentation (BertHiGRU) – lässt sich nicht eindeutig ein Gewinner ausmachen, beide Varianten sind vergleichbar gut, die Abschätzung fällt je nach Set und Metrik inkonsistent aus.
- Die vortrainierten Varianten der hierarchischen Modelle (Kürzel mit *pre) sind etwas besser als ihre nicht vortrainierten Äquivalente, allerdings nur in geringem Ausmaß. Eine Verbesserung der Modellqualität war hier aber auch nicht primär angestrebt, ihr Vorteil liegt vor allem im *schnelleren* Erreichen der Bestwerte im Trainingsprozess.
- Die Annahme, dass der erste Abschnitt eines Artikels eine besondere Rolle für die Vorhersage der Aufenthaltsdauer spielt, wird durch eine weitere interessante Beobachtung unterstützt. Wie in Abschnitt 5.7 beschrieben, bildet BertHiMean einen *gewichteten* Mittelwert der Vorhersagen der einzelnen Abschnitte. Dieser Gewichtungsvektor wird neutral als Vektor $(1, 1, \dots, 1)$ initialisiert, sodass zunächst alle Abschnitte das gleiche Gewicht erhalten und die Gewichtung im Training wie alle anderen Parameter angepasst wird. Tatsächlich tendiert das Modell dazu, die Abschnitte unterschiedlich stark in die Gesamtaufenthaltsdauer einzubeziehen, wie die Visualisierung der Gewichte in Abb. 10 aus dem besten Modellzustand von BertHiMean400 zeigt⁵¹.

⁵⁰Nicht ganz, Texte, die über die maximal gewählte Abschnittszahl hinausgehen, werden trunziert. Die Wahl der Maximallänge fiel aber großzügig.

⁵¹Das Modell verwendet sechs Abschnitte mit je 400 BERT-Tokens, die anderen BertHiMean-Modelle sind damit konsistent, die Ausprägung der Gewichtungsunterschiede ist teilweise aber weniger deutlich.

Deutlich wird: Die Gewichtung nimmt zunächst mit jedem Abschnitt ab, bevor schließlich der letzte Abschnitt wieder etwas stärker gewichtet wird. Dies ist intuitiv sinnvoll: Der erste Abschnitt eines Artikels – der im Übrigen die Artikelüberschrift und den Teasertext enthält – muss den Leser so „fesseln“, dass er den Artikel weiterliest. Zu den anderen Abschnitten kommen Leser also gegebenenfalls gar nicht mehr. Zur Beobachtung, dass der *letzte* Abschnitt eine Sonderstellung einnimmt, gibt es zwei Vermutungen: Einerseits gibt es ein bekanntes (etwa bei Lagun und Lalmas (2016) erwähntes) Lesemuster, bei dem Nutzer schnell zum Ende der Seite scrollen, vermutlich weil sie dort direkt das Fazit, besonders relevante Information oder interessante Kommentare vermuten, und um abzuschätzen, ob sich eine Lektüre des Artikels für sie lohnen könnte. In dieser Hinsicht ähnelt der letzte also dem ersten Abschnitt – beide sind gute Stellvertreter für den Artikel als Ganzes. Andererseits erkennt das Modell anhand des letzten Abschnitts möglicherweise, ob der Artikel tatsächlich *noch* länger als die maximale Abschnittszahl ist – und somit eine höhere Aufenthaltsdauer vermuten lässt.

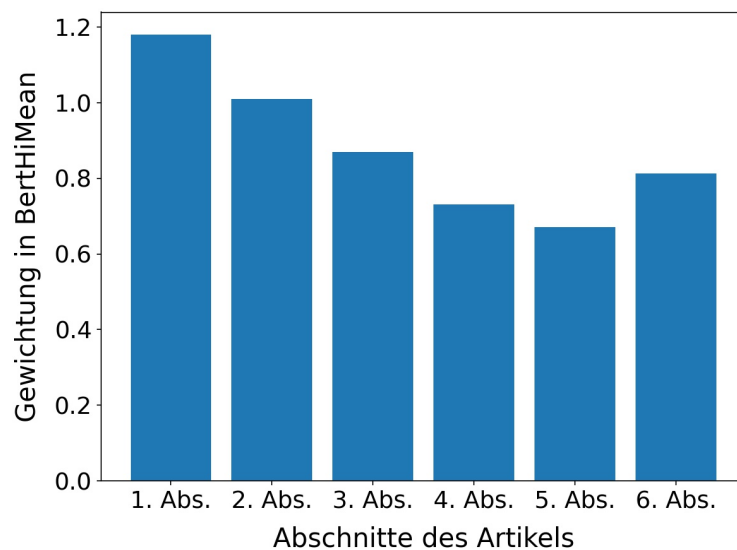


Abbildung 10: Gelernte Gewichtung der Abschnitte in BertHiMean.

Es erfolgt abschließend ein kurzer Blick in die Ergebnisse von drei in Abschnitt 2 angesprochenen Arbeiten. Davoudi et al. (2019), die die Aufenthaltsdauer auf Online-Nachrichtenartikeln vorhersagen und damit der vorliegenden Fragestellung am ähnlichsten sind, geben in der Evaluation ihrer Modelle MSE und RAE an. Für ihr bestes Modell (ein DL-Modell, das eine neuronale Repräsentation des Artikeltext und Features wie Emotionen, Events und Entitäten kombiniert) geben sie $RAE = 0.785$ auf dem verwendeten Datensatz an (MSE ist nicht vergleichbar, da die Autoren keine Angaben über Verteilung oder absoluten Wertebereiche der Aufenthaltsdauern

geben). Damit sind die einfachen BERT-Modelle der vorliegenden Arbeit kompetitiv, die hierarchischen Modelle übertreffen das Modell von Davoudi et al. (2019) relativ konsistent.

Omidvar et al. (2020) geben in ihrer Evaluation MAE und RAE an. Ihr bestes Modell erzielt dabei $RAE = 0.806$, was die BERT-Modelle dieser Arbeit deutlich übertreffen. Allerdings ist die dortige Problemstellung nicht besonders gut mit der vorliegenden vergleichbar: Die Autoren modellieren nicht die Aufenthaltsdauer, sondern Wahrscheinlichkeitsscores der von ihnen gewählten vier Qualitätskategorien, in die sie die Artikel anhand von Klickzahlen und Aufenthaltsdauer einteilen.

Lagun und Lalmas (2016) analysieren Lesemuster anhand von Viewport-Daten und modellieren primär die Zugehörigkeit von Zeitungsartikeln in eine ihrer Engagement-Klassen. Zusätzlich (sozusagen „nebenbei“) machen sie eine Vorhersage der Aufenthaltsdauer, dort erzielt ihr Modell (basierend vor allem auf Topic Modeling) eine Korrelation von $r = 0.572$, wird also von den hiesigen BERT-Modellen übertroffen.

Obwohl sich die Ergebnisse der drei genannten Arbeiten aufgrund von unterschiedlichen Problemstellungen und Datensätzen (nicht zuletzt die Sprache der Artikel) nicht vollständig mit den Ergebnissen der vorliegenden Arbeit vergleichen lassen, so zeigt sich dennoch, dass die Ansätze der vorliegenden Arbeit (Fokussierung auf den Artikeltext, Modellierung mit BERT) sehr kompetitive Ergebnisse erzielen konnten.

7 Einblick in Modellentscheidungen mit SHAP

7.1 Grundidee von SHAP

Modelle aus dem Deep Learning sind dafür berüchtigt, als *Black Box* nur wenig Interpretation über ihre Entscheidungen und Vorhersagen zuzulassen. Im Gegensatz zu linearen Modellen, bei denen die Koeffizienten der einzelnen Features einen gewissen Einblick erlauben, liegt das Wissen von DL-Modellen in Parametern, die sich durch die hochdimensionalen Transformationen nur schwer den ursprünglichen Merkmalen zuordnen lassen. Dies gilt sowohl für jede einzelne Vorhersage, als auch generell für die Entscheidungen, die das Modell trifft. Bezogen auf die Fragestellung, die in dieser Arbeit behandelt wird – die Vorhersage von Aufenthaltsdauern bei Online-Zeitungsartikeln – lässt sich dieses Problem folgendermaßen umschreiben:

Welche Merkmale (Wörter) im Text sind allgemein relevant für die Modellierung der Aufenthaltsdauer? Welche dieser Merkmale haben einen positiven, welche einen negativen Einfluss auf die Aufenthaltsdauer?

Einer explorativen Annäherung an diese Fragen widmet sich dieser Abschnitt beispielhaft für eines der einfachen BERT-Modelle, genauer für BertFFN. Im Anhang der Arbeit wird zusätzlich das baumbasierte BOWXGBoost-Modell angesprochen.

Verwendet wird hierfür das Tool **SHAP** (*Shapley Additive Explanations*), das Lundberg und Lee (2017) vorstellen. Ziel von SHAP ist es, die Auswirkung bestimmter Features auf die Modellvorhersagen zu berechnen und damit die Ausgabe eines Modells zu erklären. Die Python-Bibliothek von SHAP⁵² bietet dafür verschiedene Werkzeuge bzw. Erklärungsmodelle (*Explainer Models*), die für verschiedene Arten von zu erklärenden Modellarchitekturen geeignet sind, darunter sowohl lineare, als auch baumbasierte und schließlich DL-Modelle.

SHAP basiert auf der Berechnung von Shapley-Werten, die ursprünglich der Spieltheorie entstammen. Sie bilden ab, wie viel ein bestimmtes Merkmal (bezogen auf ein Spiel: ein bestimmter Spieler) zum Gesamtergebnis beigetragen hat. Die Funktionsweise von SHAP ist grob dargestellt wie folgt. SHAP erhält als Eingabe das zu interpretierende Modell (im bereits trainiertem Zustand), sowie eine Menge an Daten (hier: Dokumente, also Artikel). SHAP gibt die Daten in das trainierte Modell und beobachtet dabei systematisch, wie sehr sich die Modellvorhersage ändert, wenn ein bestimmtes Feature (hier: ein (Sub-)Token) aus den Daten entfernt oder durch ein beliebiges anderes ersetzt wird. Dies wird für alle Features und für alle Dokumente durchgeführt. Die verschiedenen Explainer-Modelle, die in der SHAP-Bibliothek für unterschiedliche Architekturen der Ausgangsmodelle bereitgestellt

⁵²<https://github.com/slundberg/shap>, bzw. <https://shap.readthedocs.io/en/latest/>

werden, unterscheiden sich in ihrer Arbeitsweise, grundsätzlich ist dieser Vorgang aber modellunabhängig. Auf diese Weise erlangt SHAP Wissen über die Relevanz eines Features (*Feature Importance*) für jede Vorhersage. SHAP arbeitet also mit *lokaler* Featurerelevanz, da es die Veränderung in der Vorhersage zunächst jeweils für *einen* Datenpunkt betrachtet. Da dieser Vorgang aber für alle eingegebenen Daten durchgeführt wird, kann durch aggregierte Betrachtung (Mittelung) der einzelnen Shapley-Werte anschließend auch Einblick in die *globale* Relevanz der Merkmale gegeben werden.

7.2 Einblick in die Modellvorhersagen von BertFFN

BertFFN wurde für die Anwendung mit SHAP ausgewählt, da es in der Modellevaluation gut abschneidet, aber in der Architektur weniger komplex als die hierarchischen BERT-Modelle ist. Während die Anwendung von SHAP auf baumbasierte oder lineare scikit-learn-Modelle zu SHAPs Standard gehört, spezifische Explainer-Modelle dafür vorliegen und somit die Implementierung einfach ist, sind bei BertFFN einige „Umwege“ nötig. Verwendet wurde SHAPs modellunabhängiges *Explainer*-Modell. Der *Explainer* benötigt als Eingabe eine vollständige Vorhersagefunktion (die also direkt von einer Texteingabe eine Modellvorhersage liefert). Das BertFFN-Modell musste dafür also, gemeinsam mit allen Schritten der Vorverarbeitung und Batch-Formatierung, in eine zusammenfassende Vorhersagefunktion (*wrapper*-Funktion) eingebettet werden. Die zweite Eingabe in den *Explainer* ist der BERT-Tokenizer. Dieser wird benötigt, um die einzelnen Features des Transformer-Modells (also die Tokens) systematisch maskieren zu können und somit ihre Shapley-Werte zu berechnen. Der *Explainer* kann dann eine Menge an Daten – also Artikeltexte – erhalten und die einzelnen Shapley-Werte für die Features in jedem Artikel lokal berechnen. Wie bei BertFFN selbst werden auch hier nur die ersten 512 BERT-Tokens eines Artikels betrachtet. Verwendet wurde der Entwicklungsdatensatz (dev-Set), also 3638 Artikel, grundsätzlich könnte aber jeder Teil des Datensatzes zum Einsatz kommen, da das Modell nicht neu trainiert wird.

In der Ausgabe des Explainer-Modells von SHAP wird jedem der Features (hier: BERT-Tokens) in jedem der eingegebenen Artikel ein Shapley-Wert zugeordnet. Die Anzahl an Shapley-Werten pro Dokument ist also je nach Textlänge unterschiedlich, auch gleiche Tokens in einem Dokument erhalten jeweils eigene Werte.

Shapley-Werte können sowohl positiv als auch negativ sein. Ein positiver Wert bedeutet, dass das Feature (in der spezifische Vorhersage) dazu beigetragen hat, die Zielvariable zu vergrößern (also zu einer *längeren* Aufenthaltsdauer führt), ein negativer Shapley-Wert bedeutet, dass sich das Feature negativ auf die Zielvariable ausgewirkt hat (also eine *kürzere Aufenthaltsdauer* bewirkt).

Abb. 11 (verwendet wird der *text plot* von SHAP) veranschaulicht die *lokale* Arbeitsweise von SHAP und visualisiert einen Text und die vergebenen Shapley-Werte. Tokens, die positive Shapley-Werte tragen, sind rot hinterlegt, solche, die negative Shapley-Werte tragen, sind blau hinterlegt. Die Farbintensität spiegelt den Betrag der Shapley-Werte wider (je tiefer gefärbt, desto größer der Betrag).

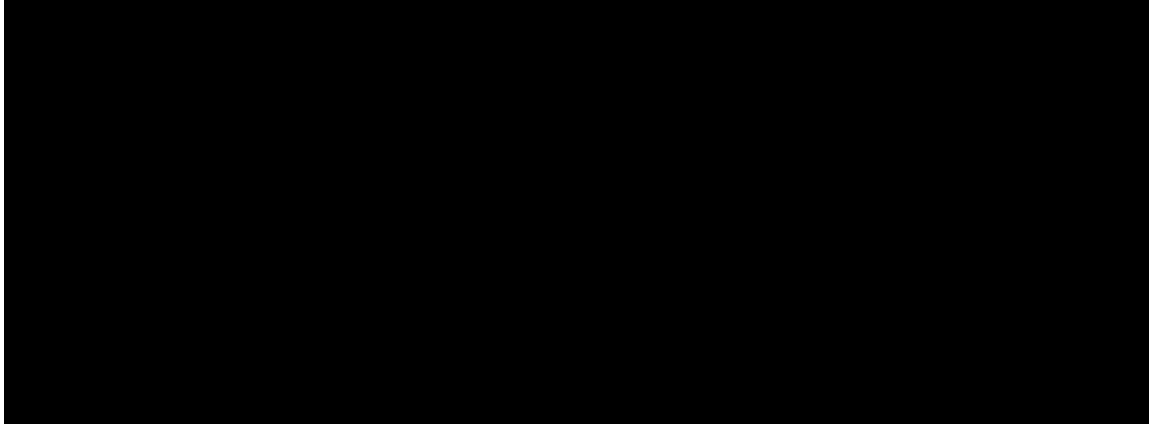


Abbildung 11: Visualisierung der Shapley-Werte innerhalb eines Artikels bei BertFFN.

Ein Token erhält also für jedes seiner Vorkommen lokal einen spezifischen Shapley-Wert zugeordnet, dabei können sowohl positive als auch negative Werte vorkommen. Um daraus trotzdem einen *globalen* Einblick in die Auswirkung bzw. Relevanz eines Features über *alle* Vorhersagen zu erhalten, können alle Shapley-Werte gemeinsam betrachtet werden.

Abb. 12 zeigt zu Illustrationszwecken die Verteilung der Shapley-Werte von drei spezifischen ausgewählten Features (*ich*, *Berlin* und XXXXXXXXXX). Deutlich wird – ohne genauer auf die einzelnen Verteilungen einzugehen –, dass insgesamt eine sehr starke Varianz innerhalb eines Features beobachtet werden kann. Die globale Einordnung als Feature mit positiver bzw. negativer Auswirkung ist also nicht eindeutig.

Um trotzdem einen Einblick zu bekommen, welche Features sich global betrachtet tendenziell *positiv* bzw. *negativ* auf die Vorhersage (also die Aufenthaltsdauer) aus-

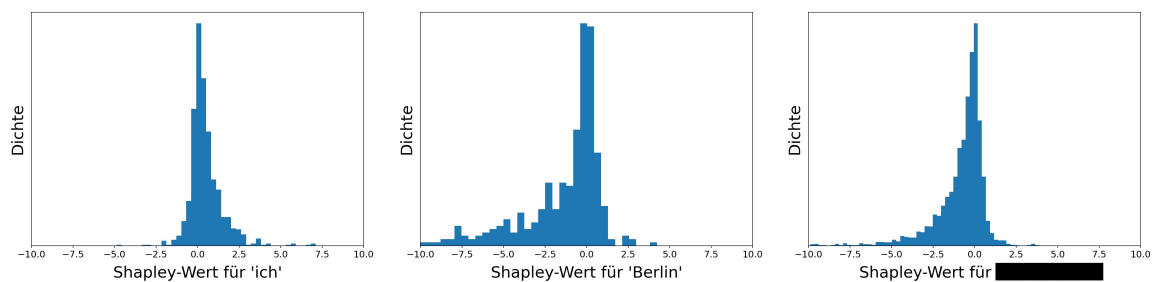


Abbildung 12: Histogramme der Shapley-Werte für *ich*, *Berlin* und XXXXXXXXXX.

wirken, wurde der Mittelwert aller Shapley-Werte (*mean*) eines Features betrachtet. In Tabelle 5 sind links die 20 Tokens mit größter positiver und rechts die 20 Tokens mit größter negativer mittlerer Auswirkung dargestellt (also sortiert nach *mean*)⁵³.

Feature	count	mean	mean_abs	Feature	count	mean	mean_abs
ich	960	0.468	0.712	Berlin	397	-4.177	4.495
mir	236	0.392	0.634	2019	568	-1.831	2.064
Ich	749	0.341	0.653	2020	714	-1.491	1.680
mich	328	0.337	0.600	Cor	2570	-1.279	1.392
bin	207	0.276	0.624	verletzt	346	-1.263	1.323
damals	200	0.274	0.591	██████████	255	-1.260	1.384
[CLS]	3638	0.268	0.427	##ona	2625	-1.231	1.348
ja	260	0.216	0.516	Foto	908	-1.215	1.498
[SEP]	3638	0.210	0.308	██████████	222	-1.211	1.346
man	1148	0.176	0.499	Del	969	-1.152	1.291
mal	354	0.146	0.585	##pel	259	-1.149	1.326
Mutter	230	0.143	0.898	██████████	2989	-1.147	1.372
habe	1746	0.142	0.479	2018	271	-1.138	1.367
hätte	258	0.112	0.448	Ib	200	-1.129	1.212
sagt	1176	0.110	0.458	##urg	210	-1.125	1.235
eigentlich	203	0.106	0.579	██████████	832	-1.112	1.274
würde	309	0.102	0.436	██████████	1150	-1.102	1.237
Dr	229	0.101	0.628	██████████	704	-1.045	1.235
Aber	609	0.090	0.504	##men	1071	-1.042	1.212
da	673	0.088	0.453	Me	943	-0.992	1.163

Tabelle 5: Die 20 positivsten (links) und negativsten (rechts) Tokens bei BertFFN (laut *mean*-Wert, nur Tokens mit Frequenz ≥ 200 betrachtet).

Auffällig in Tabelle 5 ist, dass unter den „positiven“ Tokens an oberster Stelle vier Personalpronomen (*ich*, *mir*, *Ich*, *mich*), sowie die Verbform *bin* – also alles der 1. Person Singular zuzuordnende Wörter – auftreten. Dies spricht dafür, dass Artikel mit wörtlicher (persönlicher) Rede, wie möglicherweise Erfahrungsberichte oder auch Interviews, eine hohe Aufenthaltsdauer der Nutzer erhalten. Auch die Tokens *damals*, *sagt* und *Mutter* unterstützen diese Vermutung, dass persönliche Erfahrungen gern gelesen werden.

Die Tokens *ja*, *man*, *mal*, *eigentlich* und *würde* entstammen (etwa als Diskursmarker) eher der mündlichen Sphäre, auch hier zeigt sich also die Präferenz von Artikeln, bei denen sich der Leser möglicherweise direkter eingebunden oder angesprochen

⁵³ *count* ist das absolute Auftreten des Tokens in allen Dokumenten, *mean_abs* ist der Mittelwert der absoluten Shapley-Werte, gibt also einen Hinweis auf die Stärke der Relevanz bzw. Auswirkung des Tokens allgemein, ungeachtet der positiv-negativ-Ausrichtung.

fühlt. Konnektoren wie *Aber* und *da* könnten darauf hindeuten, dass sich ein flüssiger und verständlicher Schreibstil positiv auf die Aufenthaltsdauer auswirkt.

Das Auftreten von [CLS] und [SEP] – also die beiden Spezialtokens, die der BERT-Tokenizer jeweils zu Beginn und Ende des Artikeltextes setzt – überrascht zunächst, schließlich sind sie in *allen* Texten enthalten. Eine mögliche Erklärung könnte hier sein: BERT repräsentiert die Tokens dank der Attention-Mechanismen im Zusammenhang. SHAP erkennt in seinem Maskierungsvorgang Features, die stark zusammenhängen und gruppiert diese (bezeichnet als *Hierarchical Values*, vergleiche in dem Zusammenhang auch die gleichförmig gefärbten Textblöcke in Abb. 11). Gerade die allerersten Tokens im Text – die Überschrift und eventuell der Teasertext – stellen mutmaßlich sehr relevante Features dar, aufgrund der Gruppierung von SHAP färbt das auch auf das [CLS]-Token ab. Diese Beobachtung deckt sich mit BertHiMean und seiner gelernten Gewichtung der Abschnitte. Für [SEP] könnte ähnliches gelten, oder aber: Hier spielt möglicherweise inhaltliche Information weniger eine Rolle als vielmehr die Tatsache, dass BERT anhand der letzten Tokens (Vorhandensein von Padding-Tokens oder „echten“ Tokens) erkennt, ob der trunkierte Artikel eigentlich noch länger ist, oder aber ein sehr kurzer Artikel vorliegt.

Die Liste der Tokens mit im Mittel größter negativer Auswirkung auf die Aufenthaltsdauer (rechte Tabelle in 5) lässt weniger klare Schlüsse zu. Sie enthält viele Ortsnamen aus dem Verbreitungsgebiet der [REDACTED], bzw. Subtokens, die Ortsnamen vermuten lassen ([REDACTED], gemeint sein könnten [REDACTED]). Ortsnamen treten üblicherweise direkt zu Beginn eines Artikels auf, als sogenannte *Spitzmarke*. Dieser negative Einfluss auf die Aufenthaltsdauer könnte sich entweder dadurch erklären, dass Artikel mit regionaler Thematik wenig Interesse auf sich ziehen – oder aber, dass diese im Mittel als Meldungen eher kurz gehalten sind. Allerdings fallen auch [REDACTED] und vor allem *Berlin* in diese Liste von „negativen“ Tokens. Letzteres könnte mit der thematischen Ausrichtung der Artikel zusammenhängen, die mit der Bezeichnung *Berlin* auftreten – vermutlich vor allem politische Meldungen. Eine weitere mögliche Begründung ist schlicht die Positionierung solcher *Spitzmarken* zu Beginn des Teasers/Vorspann: Dieser Teil wird mutmaßlich häufig schlicht überlesen, da bereits der Titel die Hauptinformationen liefert und die Ortsmeldung meist keine wirklich relevante Information für den Leser enthält. Er liefert also einen negativen Beitrag für die Gesamtdauer.

Eine weitere interessante Beobachtung ist auch die negative Auswirkung von *Corona* auf die Aufenthaltsdauer (bzw. der beiden Subtokens *Cor* und *##ona*, hier zeigt sich, dass BERTs Vokabular noch nicht auf dem neuesten Stand ist). Dies könnte wiederum verschiedene Gründe haben: Entweder sind Corona-Artikel schlicht

relativ kurz (etwa eine tägliche Meldung zu Inzidenzen und anderen Zahlen), oder aber Nutzer schließen die Artikel schnell, weil sie sie nur kurz nach einer spezifischen Information (etwa die Öffnungssituation von Schulen/Geschäften) durchsuchen⁵⁴. Auch das Token *verletzt* kommt mutmaßlich vor allem in sehr kurzen Unfallmeldungen vor.

Die durchschnittlich negative Auswirkung des Tokens *Foto* ist interessant. Fotos in Artikeln – wenn davon ausgegangen wird, dass sich vom Vorkommen des Tokens *Foto* tatsächlich auf die Präsenz von Fotos schließen lässt – führen wohl zu kürzeren Aufenthaltsdauern. Die negative Auswirkung von Jahreszahlen (*2019*, *2020*, *2018*) ist auffällig. Möglicherweise sind sie ein Anzeichen von eher „trockenen“ Artikeln, bspw. Wirtschaftsberichten.

Eine generelle Beobachtung beim Vergleich der Tokens mit positiver und denen mit negativer Auswirkung ist, dass erstere deutlich kleinere absolute mittlere Shapley-Werte aufweisen als letztere. Dies könnte sich dadurch erklären, dass sich auf der positiven Liste vor allem Wörter befinden, die persönlichen/verbalen Stil kennzeichnen. Es handelt sich dabei um Wörter, die häufig mehrmals im Text auftreten. Jedes einzelne Auftreten erhält daher geringere Werte. In der rechten Tabelle (negative Auswirkung) finden sich hingegen vor allem Tokens, die vermutlich nur einmal auftreten (etwa die Ortsnamen) und daher *einen* (absolut) großen Wert erhalten.

Diese Betrachtung der lokalen und – anhand von schlichter Mittelung aller Shapley-Werte – globalen Relevanz der Features konnte interessante Einblicke in die Modellentscheidungen von BertFFN geben. Es konnten einige Tendenzen beobachtet werden, welche Faktoren oder Textmerkmale einen (positiven oder negativen) Einfluss auf die Aufenthaltsdauer der Leser von Online-Zeitungsartikeln haben können. Es zeigt sich, wie relevant und interessant der Einblick in die *Black Box* von DL-Modellen anhand von SHAP oder anderer Tools ist.

Im Anhang findet sich in paralleler Herangehensweise die Anwendung von SHAP auf eines der BOW-basierten Baseline-Modelle (BOWXGBoost). Hier stellte sich allerdings heraus, dass die Betrachtung mit SHAP nicht besonders sinnvoll ist, da die Interpretierbarkeit der Shapley-Werte (und ihrer Mittelung) bei den absoluten BOW-Features einige Probleme aufwirft.

⁵⁴Hier könnte auch ein Zusammenhang mit der hohen Stellung von *Berlin* vermutet werden. Eine manuelle Inspektion bestätigt, dass Corona-Meldungen meist mit *Berlin* versehen sind.

8 Diskussion und Ausblick

In der vorliegenden Arbeit wurde anhand des Artikeltextes die durchschnittliche Aufenthaltsdauer bei einem Online-Zeitungsartikel modelliert. Eine längere Aufenthaltsdauer – gemittelt über alle Aufrufe des Artikels – wurde dabei als mehr Nutzerinteraktion, höheres User Engagement, Interesse und schließlich als Indikator für Popularität des Artikels verstanden. Diese Interpretation wird im digitalen Marketing bzw. bei der Evaluierung von Webseitenperformanz häufig angewendet und wird durch viele Untersuchungen gestützt, die explizite Nutzerbefragungen mit impliziten Messungen (Aufenthaltsdauer, Klickzahlen, Scrollingverhalten) kombinieren. In diesem Abschnitt werden einige kritische Punkte angesprochen und Anknüpfungspunkte für Folgearbeiten bzw. interessante weitere Ansätze angerissen.

Der verwendete Datensatz enthält aggregierte Nutzerdaten. Es liegen keine einzelnen Logdateien (oder spezifische Aufenthaltsdauern) der Nutzer vor, sondern lediglich die Gesamtzahl an Artikelaufrufen, sowie die aufsummierte Aufenthaltsdauer auf dem Artikel. Aus diesen beiden Werten kann die hier als Zielvariable verwendete mittlere Aufenthaltsdauer berechnet werden. Weder Informationen über die Streuung innerhalb der Nutzer, noch über Ausreißer (besonders lange Aufenthaltsdauern bei gleichzeitiger Inaktivität, etwa eines Nutzers, der mehrere Tabs geöffnet hat) sind bekannt, Google Analytics stellt solche Werte nicht zur Verfügung. Auch wie sehr die Nutzer tatsächlich während ihrer Aufenthaltsdauer den Artikel gelesen (oder generell mit ihm interagiert) haben, ist nicht bekannt. Aus diesem Grund wurden nur Artikel mit mindestens 50 Besuchen verwendet, trotzdem muss dieses große Maß an Unsicherheit bedacht werden.

Damit zusammen hängt die mehrfach angesprochene Problematik der Interpretation der Aufenthaltsdauer als *Lesedauer*. Es liegen Artikel sehr unterschiedlicher Länge vor, gleichzeitig lassen die Daten keinen Schluss darüber zu, welcher Anteil (und welche konkreten Abschnitte) des Artikels tatsächlich gelesen wurde – auch hier wären natürlich einzelne Nutzerdaten sinnvoll. Gleichzeitig unterscheiden sich Nutzer stark in ihrem Leseverhalten (Interessenprofil, Lesegeschwindigkeit, Interaktion mit Fotos oder Kommentarspalten), die Mittelung ist also eine eher grobe Annäherung an *den generellen Leser*, hier wären Scrolling- oder *Viewport*-Daten wünschenswert. Auch die Behandlung aller Artikel unabhängig von ihrer Thematik oder Rubrik ist potenziell problematisch, so werden etwa *Lifestyle*-Artikel sicherlich anders als Artikel aus den Bereichen Technik oder Politik rezipiert. Auch Länge der Artikel, sowie die Wahrscheinlichkeit für rege Kommentarspalten ist hier vermutlich unterschiedlich. Hier wäre eine detailliertere Betrachtung interessant, die Thematik der

Artikel müsste – etwa mit Hilfe von Topic Modeling – ermittelt werden⁵⁵ oder andere Daten verwendet werden, bei denen mehr Metainformation vorliegt. Neben Thematik könnten auch weitere Textebenen betrachtet werden, etwa der journalistische Stil, die Lesbarkeit, der Neuigkeitsgehalt oder der emotionale Gehalt des Artikels.

In der Modellierung wurde nur der Artikeltext (und die Textlänge) verwendet. Natürlich ist dies auch der Fokus der vorliegenden computerlinguistischen Arbeit, trotzdem wäre die Verwendung von Informationen über weitere Elemente (Fotos, Videos, Werbung, Kommentare) oder Metadaten (Autor, Erscheinungszeitpunkt) interessant und potenziell hilfreich. Die Beschränkung auf den reinen Text bildet natürlich die Nutzererfahrung auf einem Online-Zeitungsportal sehr stark vereinfacht ab. Technisch konnte hierfür das BertTextlength-Modell einen ersten Ansatz liefern, wie textuelle Features mit Metadaten in DL-Architekturen kombiniert werden können. Hier wäre eine Erweiterung sicherlich möglich und sinnvoll.

Methodisch wäre ein weiterer interessanter Ansatz die getrennte Verwendung von Titel, Teasertext und Haupttext, in der vorliegenden Modellierung wurden die drei Teile schlicht konkateniert und als *Artikeltext* verwendet. Gerade der Teasertext dient dazu, das Interesse des Lesers zu wecken. Die Beobachtung dieser Arbeit, dass gerade der Textanfang besonders relevant für die Vorhersage der Aufenthaltsdauer ist, unterstützt dies.


Auch der Umgang mit der BERT-Textlängenbeschränkung und dem daraus resultierenden Problem der Anwendung von BERT auf lange Dokumente – also die beiden hierarchischen BERT-Modelle – könnte verfeinert werden. Denkbar wären beispielsweise statt der harten Teilung in Abschnitte fester Größe und (zunächst) getrennter Weiterverarbeitung, ein überlappender Ansatz oder die Zuhilfenahme der natürlichen Abschnittsunterteilung der Artikel. Auch die Beobachtung, dass nicht unbedingt möglichst lange Abschnitte die besten Ergebnisse erzielen, bietet Raum für Folgearbeiten, auch eine feingliedrigere (etwa satzbasierte) Herangehensweise könnte sinnvoll sein⁵⁶.

In der vorliegenden Arbeit wurde die Aufenthaltsdauer als Zielvariable und Proxy für User Engagement gewählt, statt etwa Seitenaufrufen zu betrachten. Die Gründe dafür wurden deutlich gemacht (starke Abhängigkeit der Seitenaufrufe von anderen Faktoren wie Platzierung, Problematik von Klickzahlen und Zufriedenheit *nach* dem

⁵⁵Zwar gibt es im [REDACTED]-Datensatz die Spalten *category* und *rubric*, beide sind aber für eine thematische Klassifikation nur wenig hilfreich. So tragen etwa 80% der Artikel unter *category* einen der Werte *lokales*, *deutschland-und-welt* oder *sport*, also eine sehr grobe Einteilung. Unter *rubric* findet sich eine Mischung aus regionalen und thematischen Einordnungen, die häufigsten Einträge sind: [REDACTED], [REDACTED], *vermishtes*, *politik*, [REDACTED], [REDACTED], [REDACTED].

⁵⁶Generell wäre auch eine systematische Betrachtung von Auswirkungen unterschiedlich gesetzter Hyperparameter in den DL-Modellen sinnvoll, dies konnte im Rahmen dieser Arbeit nicht geleistet werden, es wurde daher in der Regel auf *übliche* Werte zurückgegriffen.

Klicken, der Artikeltext liegt dem Leser erst nach dem Seitenaufruf vor), trotzdem wäre eine Auseinandersetzung mit den Klickzahlen eine interessante Fortsetzung⁵⁷. Auch hier müsste gut ausgewählt werden, welche der Artikelteile für die Modellierung zu verwenden sind. Auch eine kombinierte Betrachtung von Aufenthaltsdauer und Klicks sowie eine Untersuchung speziell der Überschriften bietet sich in diesem Zusammenhang an, angelehnt an Arbeiten wie Omidvar et al. (2020) aus dem Bereich der *Clickbait*-Forschung. Damit zusammen hängt auch das Verhältnis zwischen der – hier fokussierten – Popularität und der *Qualität* eines Artikels, das hier nicht weiter behandelt werden konnte.

⁵⁷Andere Maße, wie etwa Zahl an Kommentaren, Teilungen/Verlinkungen, liegen im -Datensatz nicht vor, wären aber bei einer Betrachtung anderer Daten ebenfalls interessant.

9 Fazit

Die vorliegende Arbeit behandelt User Engagement bei Online-Zeitungsportalen. Der Fokus liegt dabei auf der Modellierung der durchschnittlichen Aufenthaltsdauer, die Leser auf einem Artikel verbringen.

Zunächst wurde ein Überblick über verschiedene Forschungsfelder gegeben, die implizites Feedback bzw. KPI-Werte (*Key Performance Indicators*) als Maß für Popularität bzw. Nutzerzufriedenheit verwenden. Die vorliegende Arbeit ordnet sich in das Feld der Vorhersage von Popularität von Zeitungsartikeln (*Online News Popularity Prediction*) ein und verwendet Aufenthaltsdauer (*Dwell Time*) als Indikator für Popularität. Soweit bekannt, ist dies die erste Arbeit, die sich dieser Aufgabe anhand von *deutschen* Zeitungsartikeln widmet.

Anschließend wurde der verwendete Datensatz beschrieben, der eigens für die Arbeit erstellt wurde. Ursprünglich lag eine Sammlung von Artikeln von vier Online-Nachrichtenportalen vor, bestehend aus den Artikeln in Form von HTML-Dokumenten, sowie den mit einer Tracking-Software (Google Analytics) erhobenen KPI-Werten jedes Artikels (unter anderem Zahl der Seitenaufrufe und Angaben zur Aufenthaltsdauer pro Artikel). Hierbei handelt es sich um aggregierte Werte über alle Besucher des Artikels. Um an die Artikeltexte selbst zu gelangen, wurde ein HTML-Parsing aufgesetzt. Für die Modellierung der Aufenthaltsdauer – die zentrale Leistung dieser Arbeit – wurden schließlich 36383 Artikel der [REDACTED] ([REDACTED]) ausgewählt.

Nach einer Beschreibung der computerlinguistischen Grundlagen und verwendeten Modellierungstechniken – besonderer Fokus liegt hier auf der Verwendung von BERT, einem modernen Transformer-Modell –, wurden verschiedene Modelle zur Vorhersage der Aufenthaltsdauer anhand des Artikeltextes beschrieben. Dabei wurden sowohl herkömmliche Methoden (*Bag-of-Words*-Features, Regressionsmodelle) verwendet, als auch moderne Architekturen aus dem Bereich Deep Learning verwendet. Aus methodischer Sicht besonders hervorzuheben sind dabei die beiden hierarchischen BERT-Modelle, die das Problem angehen, dass BERT grundsätzlich eine Längenbeschränkung seiner Eingabe vorgibt und sich daher für *lange* Dokumente Schwierigkeiten ergeben. Während die anderen Modelle daher nur den Artikelanfang verwenden, wird bei den hierarchischen Modellen ein komplexerer Ansatz gewählt: Die Artikel werden in Abschnitte aufgeteilt, einzeln weiterverarbeitet und anschließend in eine kombinierte Modellvorhersage überführt. Eines der Modelle verwendet dabei einen gewichteten Mittelwert der Abschnittsvorhersagen, das andere erstellt aus den einzelnen Abschnittsrepräsentationen eine gemeinsame Artikelrepräsentation und trifft dann seine Vorhersage mittels einer nachfolgenden rekurrenten Schicht.

Für das Training der Modelle wurde der [REDACTED]-Datensatz in drei Teile aufgeteilt: der Trainingsdatensatz zur Schätzung der Modellparameter, der Entwicklungsdatensatz zur Überwachung der Trainingsprozesse der DL-Modelle, sowie ein Testdatensatz. Letzterer wurde für eine umfassende Evaluation der Modelle verwendet, die einige interessante Erkenntnisse liefern konnte: Die BERT-Modelle übertreffen die herkömmliche *Bag-of-Words*-Modellierung eindeutig. Trotz der Beschränkung auf den ersten Teil eines Artikels erzielen bereits die einfachen BERT-Modelle gute Ergebnisse bei der Vorhersage der durchschnittlichen Aufenthaltsdauer. Zur besseren Einordnung der Evaluationswerte: Die Problemstellung der Arbeit ist durchaus sehr komplex. Den Modellen liegt *ausschließlich* der Artikeltext vor, Informationen über andere Elemente wie Fotos, Werbung, Kommentarspalten und natürlich auch die Tatsache, dass Nutzer sich in Leseverhalten und Interesse stark unterscheiden, sind den Modellen nicht zugänglich. Gleichzeitig wirken diese sich mutmaßlich stark auf die Aufenthaltsdauer der Nutzer aus. Wird dies bedacht, so sind die Evaluationswerte sehr zufriedenstellend. Das über alle Metriken betrachtet bestabschneidende Modell – eines der hierarchischen Modelle – erzielt eine Korrelation von $r = 0.65$ und einen mittleren absoluten Fehler von 48.7 Sekunden beim Vergleich der vorhergesagten und der wahren Werte im Testdatensatz. Das gute Abschneiden der einfachen BERT-Modelle und die Abschnittsgewichtung bei BertHiMean zeigt, dass gerade der Artikelanfang eine entscheidende Rolle für die Aufenthaltsdauer spielt. Allgemein übertreffen die abschnittsweise hierarchischen BERT-Modelle fast immer die einfachen BERT-Modelle.

Eines der BERT-Modelle wurde in einer sekundären Analyse mit dem Tool SHAP näher betrachtet. Dabei konnte unter anderem beobachtet werden, dass die Verwendung der ersten Person und vieler verbaler Elemente, vermutlich als Kennzeichen von persönlichem oder „erzählendem“ Stil, positive Auswirkungen auf die Aufenthaltsdauer von Nutzern hat.

Die vorliegende Arbeit bietet interessante Einblicke in das Leseverhalten von Nutzern von Online-Nachrichtenportalen. Ausschließlich anhand des Artikeltextes konnte mit zufriedenstellender Qualität die Aufenthaltsdauer eines Artikels vorhergesagt werden. Damit können die hier gewonnenen Ergebnisse unmittelbar für die Betreiber von Online-Portalen hilfreich sein und ihnen die Auswahl oder das Editieren und Positionieren von Artikeln *vor* ihrem Erscheinen erleichtern. Auch zukünftige Arbeiten könnten auf die Ergebnisse aufbauen, beispielsweise um Tools zu erstellen, die Journalisten beim Schreiben von besonders lesenswerten Artikeln unterstützen.

Literaturverzeichnis

- Agichtein, E., Brill, E. & Dumais, S. (2006). Improving web search ranking by incorporating user behavior information. In *SIGIR 2006 — Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (S. 19–26). Seattle, Washington, USA, August 6 – 11, 2006.
- Ambroselli, C., Risch, J., Krestel, R. & Loos, A. (2018). Prediction for the newsroom: Which articles will get the most comments? In *NAACL-HLT 2018 — Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (Bd. 3: Industry Papers, S. 193–199). New Orleans, Louisiana, USA, June 1 – 6, 2018.
- An, M., Wu, F., Wu, C., Zhang, K., Liu, Z. & Xie, X. (2019). Neural news recommendation with long- and short-term user representations. In *ACL 2019 — Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (S. 336–345). Florence, Italy, July 28 – August 2, 2019.
- Arapakis, I., Lalmas, M., Cambazoglu, B. B., Marcos, M.-C. & Jose, J. M. (2014). User engagement in online news: Under the scope of sentiment, interest, affect, and gaze. *Journal of the Association for Information Science and Technology*, 65 (10), 1988–2005.
- Arapakis, I., Peleja, F., Berkant, B. & Magalhaes, J. (2016). Linguistic benchmarks of online news article quality. In *ACL 2016 — Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (Bd. 1: Long Papers, S. 1893–1902). Berlin, Germany, August 7 – 12, 2016.
- Bandari, R., Asur, S. & Huberman, B. (2012). The pulse of news in social media: Forecasting popularity. In *ICWSM 2012 — Proceedings of the 6th International AAAI Conference on Weblogs and Social Media* (S. 26–33). Dublin, Ireland, June 4 – 7, 2012.
- Barbieri, N., Silvestri, F. & Lalmas, M. (2016). Improving post-click user engagement on native ads via survival analysis. In *WWW 2016 — Proceedings of the 25th International Conference on World Wide Web* (S. 761–770). Montréal, Québec, Canada, April 11 – 15, 2016.
- Beltagy, I., Peters, M. E. & Cohan, A. (2020). Longformer: The long-document transformer. *arXiv:2004.05150 [cs.CL]*.
- Buechel, S., Sedoc, J., Schwartz, H. A. & Ungar, L. (2020). Learning emotion from 100 observations: Unexpected robustness of deep learning under strong data limitations. In *PEOPLES 2020 — Proceedings of the Third Workshop on Computational Modeling of People’s Opinions, Personality, and Emotions*

- in Social Media @ COLING 2020* (S. 129–139). Online (Barcelona, Spain), December 13, 2020.
- Chen, J., Zhuang, F., Wang, T., Lin, L., Xia, F., Du, L. & He, Q. (2021). Follow the title then read the article: Click-guide network for dwell time prediction. *IEEE Transactions on Knowledge & Data Engineering*, 33 (7), 2903–2913.
- Chen, T. & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *KDD 2016 — Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (S. 785–794). San Francisco, California, USA, August 13 – 17, 2016.
- Claypool, M., Le, P., Wased, M. & Brown, D. (2001). Implicit interest indicators. In *IUI 2001 — Proceedings of the 6th International Conference on Intelligent User Interfaces* (S. 33–40). Santa Fe, New Mexico, USA, January 14 – 17, 2001.
- Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. & Salakhutdinov, R. (2019). Transformer-XL: Attentive language models beyond a fixed-length context. In *ACL 2019 — Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (S. 2978–2988). Florence, Italy, July 28 – August 2, 2019.
- Davoudi, H., An, A. & Edall, G. (2019). Content-based dwell time prediction model for news articles. In *NAACL-HLT 2019 — Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (Bd. 2: Industry Papers, S. 226–233). Minneapolis, Minnesota, USA, June 2 – 7, 2019.
- Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT 2019 — Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (Bd. 1: Long and Short Papers, S. 4171–4186). Minneapolis, Minnesota, USA, June 2 – 7, 2019.
- Fernandes, K., Vinagre, P. & Cortez, P. (2015). A proactive intelligent decision support system for predicting the popularity of online news. In *EPIA 2015 — Progress in Artificial Intelligence – 17th Portuguese Conference on Artificial Intelligence* (S. 535–546). Coimbra, Portugal, September 8 – 11, 2015.
- Firth, J. R. (1957). A synopsis of linguistic theory 1930-55. *Studies in Linguistic Analysis*. (Reprinted in Palmer, F. (ed.) 1968. Selected Papers of J. R. Firth. Longman, Harlow.)
- Fox, S., Karnawat, K., Mydland, M., Dumais, S. & White, T. (2005). Evaluating implicit measures to improve web search. *ACM Transactions on Information*

Systems, 23 (2), 147–168.

- Grail, Q., Perez, J. & Gaussier, E. (2021). Globalizing BERT-based transformer architectures for long document summarization. In *EACL 2021 — Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics* (Bd. Main Volume, S. 1792–1810). Online, April 19 – 23, 2021.
- Grave, E., Bojanowski, P., Gupta, P., Joulin, A. & Mikolov, T. (2018). Learning word vectors for 157 languages. In *LREC 2018 — Proceedings of the 11th International Conference on Language Resources and Evaluation* (S. 3483–3487). Miyazaki, Japan, May 7 – 12, 2018.
- Guo, Q. & Agichtein, E. (2012). Beyond dwell time: Estimating document relevance from cursor movements and other post-click searcher behavior. In *WWW 2012 — Proceedings of the 21st International Conference on World Wide Web* (S. 569–578). Lyon, France, April 16 – 20, 2012.
- Hardt, D., Hovy, D. & Lamprinidis, S. (2018). Predicting news headline popularity with syntactic and semantic knowledge using multi-task learning. In *EMNLP 2018 — Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (S. 659–664). Brussels, Belgium, October 31 – November 4, 2018.
- Hardt, D. & Rambow, O. (2017). Predicting user views in online news. In *Proceedings of the 2017 EMNLP Workshop: Natural Language Processing meets Journalism @ EMNLP 2017* (S. 7–12). Copenhagen, Denmark, September 7, 2017.
- Homma, R., Seki, Y., Yoshida, M. & Umemura, K. (2020). Analysis of short dwell time in relation to user interest in a news application. *arXiv:2012.13992 [cs.IR]*.
- Homma, R., Soejima, K., Yoshida, M. & Umemura, K. (2018). Analysis of user dwell time on non-news pages. In *Big Data 2018 — 2018 IEEE International Conference on Big Data* (S. 4333–4338). Seattle, Washington, USA, December 10 – 13, 2018.
- Kellar, M., Watters, C., Duffy, J. & Shepherd, M. (2004). Effect of task on time spent reading as an implicit measure of interest. *Proceedings of the American Society for Information Science and Technology*, 41 (1), 168–175.
- Kelly, D. & Belkin, N. J. (2004). Display time as implicit feedback: Understanding task effects. In *SIGIR 2004 — Proceedings of the 27th Annual ACM International Conference on Research and Development in Information Retrieval* (S. 377–384). Sheffield, United Kingdom, July 25 – 29, 2004.
- Kim, Y., Hassan, A., White, R. W. & Zitouni, I. (2014). Modeling dwell time to predict click-level satisfaction. In *WSDM 2014 — The 7th Annual International*

- ACM Conference on Web Search and Data Mining* (S. 193–202). New York, New York, USA, February 24 – 28, 2014.
- Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R. & Riedl, J. (1997). Grouplens: Applying collaborative filtering to usenet news. *Communications of the ACM*, 40 (3), 77–87.
- Lagun, D. & Lalmas, M. (2016). Understanding user attention and engagement in online news reading. In *WSDM 2016 — Proceedings of the Ninth ACM International Conference on Web Search and Data Mining* (S. 113–122). San Francisco, California, USA, February 22 – 25, 2016.
- Liu, C., Liu, J., Belkin, N., Cole, M. & Gwizdka, J. (2011). Using dwell time as an implicit measure of usefulness in different task types. *Proceedings of the American Society for Information Science and Technology*, 48 (1), 1–4.
- Liu, C., White, R. W. & Dumais, S. (2010). Understanding web browsing behaviors through Weibull analysis of dwell time. In *SIGIR 2010 — Proceeding of the 33rd international ACM SIGIR Conference on Research and Development in Information Retrieval* (S. 379–386). Geneva, Switzerland, July 19 – 23, 2010.
- Liu, J. & Belkin, N. J. (2015). Personalizing information retrieval for multi-session tasks: Examining the roles of task stage, task type, and topic knowledge on the interpretation of dwell time as an indicator of document usefulness. *Journal of the Association for Information Science and Technology*, 66 (1), 58–81.
- Liu, J., Dolan, P. & Pedersen, E. R. (2010). Personalized news recommendation based on click behavior. In *IUI 2010 — Proceedings of the 15th International Conference on Intelligent User Interfaces* (S. 31–40). Hong Kong, China, February 7 – 10, 2010.
- Lopyrev, K. (2015). Generating news headlines with recurrent neural networks. *arXiv:1512.01712 [cs.CL]*.
- Lu, H., Zhang, M. & Ma, S. (2018). Between clicks and satisfaction: Study on multi-phase user preferences and satisfaction for online news reading. In *SIGIR 2018 — The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval* (S. 435–444). Ann Arbor, Michigan, USA, July 8 – 12, 2018.
- Lundberg, S. M. & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *NIPS 2017 — Proceedings of the 31st International Conference on Neural Information Processing Systems* (S. 4768–4777). Long Beach, California, USA, December 4 – 9, 2017.
- Michel, P., Levy, O. & Neubig, G. (2019). Are sixteen heads really better than one? In *NeurIPS 2019 — Advances in Neural Information Processing Systems 32*. Vancouver, Canada, December 8 – 14, 2019.

- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *NIPS 2013 — Proceedings of the 26th International Conference on Neural Information Processing Systems* (Bd. 2, S. 3111–3119). Lake Tahoe, Nevada, USA, December 5 – 8, 2013.
- Morita, M. & Shinoda, Y. (1994). Information filtering based on user behavior analysis and best match text retrieval. In *SIGIR 1994 — Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (S. 272–281). Dublin, Ireland, July 3 – 6, 1994.
- Namous, F., Rodan, A. & Javed, Y. (2018). Online news popularity prediction. In *ITT 2018 — Fifth HCT Information Technology Trends* (S. 180–184). Dubai, UAE, November 28 – 29, 2018.
- Okura, S., Tagami, Y., Ono, S. & Tajima, A. (2017). Embedding-based news recommendation for millions of users. In *KDD 2017 — Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (S. 1933–1942). Halifax, Canada, August 13 – 17, 2017.
- Omidvar, A., Jiang, H. & An, A. (2018). Using neural network for identifying clickbaits in online news media. *arXiv:1806.07713 [cs.CL]*.
- Omidvar, A., Pourmodheji, H., An, A. & Edall, G. (2020). Learning to determine the quality of news headlines. In *ICAART 2020 — Proceedings of the 12th International Conference on Agents and Artificial Intelligence* (Bd. 1: NLPinAI, S. 401–409). Valletta, Malta, February 22 – 24, 2020.
- Pappagari, R., Żelasko, P., Villalba, J., Carmiel, Y. & Dehak, N. (2019). Hierarchical transformers for long document classification. *arXiv:1910.10781 [cs.CL]*.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., . . . Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS 2019 — Advances in Neural Information Processing Systems 32* (S. 8024–8035). Vancouver, Canada, December 8 – 14, 2019.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Édouard Duchesnay (2011). SCIKIT-LEARN: Machine learning in PYTHON. *Journal of Machine Learning Research*, 12 (85), 2825–2830.
- Pennington, J., Socher, R. & Manning, C. (2014). GloVe: Global vectors for word representation. In *EMNLP 2014 — Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (S. 1532–1543). Doha, Qatar, October 25 – 29, 2014.
- Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. & Zettlemoyer, L. (2018). Deep contextualized word representations. In *NAACL-HLT 2018 — Proceedings of the 2018 Conference of the North American Chapter of the*

- Association for Computational Linguistics: Human Language Technologies* (Bd. 1: Long Papers, S. 2227–2237). New Orleans, Louisiana, USA, June 1 – 6, 2018.
- Qiu, J., Ma, H., Levy, O., Yih, W., Wang, S. & Tang, J. (2020). Blockwise self-attention for long document understanding. In *EMNLP 2020 — Findings of the Association for Computational Linguistics* (S. 2555–2565). Online, November 16 – 20, 2020.
- Rathord, P., Jain, D. A. & Agrawal, C. (2019). A comprehensive review on online news popularity prediction using machine learning approach. *SMART MOVES JOURNAL IJOSCIENCE*, 5 (1), 25–29.
- Reis, J., Benevenuto, F., Olmo, P., Prates, R., Kwak, H. & An, J. (2015). Breaking the news: First impressions matter on online news. In *ICWSM 2015 — Proceedings of the Ninth International AAAI Conference on Web and Social Media* (S. 357–366). Oxford, England, May 26 – 29, 2015.
- Ren, H. & Yang, Q. (2015). Predicting and evaluating the popularity of online news. *Stanford University Machine Learning Report*.
- Seki, Y. & Yoshida, M. (2018). Analysis of user dwell time by category in news application. In *WI 2018 — 2018 IEEE/WIC/ACM International Conference on Web Intelligence* (S. 732–735). Santiago, Chile, December 3 – 6, 2018.
- Stokowiec, W., Trzciński, T., Wołk, K., Marasek, K. & Rokita, P. (2017). Shallow reading with deep learning: Predicting popularity of online content using only its title. *arXiv:1707.06806 [cs.CL]*.
- Tsagkias, M., Weerkamp, W. & de Rijke, M. (2009). Predicting the volume of comments on online news stories. In *CIKM 2009 — Proceedings of the 18th ACM Conference on Information and Knowledge Management* (S. 1765–1768). Hong Kong, China, November 2 – 6, 2009.
- van Schijndel, M. & Linzen, T. (2018). A neural model of adaptation in reading. In *EMNLP 2018 — Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (S. 4704–4710). Brussels, Belgium, October 31 – November 4, 2018.
- van Schijndel, M. & Schuler, W. (2015). Hierarchic syntax improves reading time prediction. In *NAACL-HLT 2015 — Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (S. 1597–1605). Denver, Colorado, USA, May 31 – June 5, 2015.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention is all you need. In *NIPS 2017 — Proceedings of the 31st International Conference on Neural Information Processing Systems*

- (S. 6000–6010). Long Beach, California, USA, December 4 – 9, 2017.
- Wang, H., Zhang, F., Xie, X. & Guo, M. (2018). DKN: Deep knowledge-aware network for news recommendation. In *WWW 2018 — Proceedings of the 2018 World Wide Web Conference* (S. 1835–1844). Lyon, France, April 23 – 27, 2018.
- Weller, O., Hildebrandt, J., Reznik, I., Challis, C., Tass, E. S., Snell, Q. & Seppi, K. (2020). You don’t have time to read this: An exploration of document reading time prediction. In *ACL 2020 — Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (S. 1789–1794). Online, July 5 – 10, 2020.
- White, R. W. & Kelly, D. (2006). A study on the effects of personalization and task information on implicit feedback performance. In *CIKM 2006 — Proceedings of the 15th ACM International Conference on Information and Knowledge Management* (S. 297–306). Arlington, Virginia, USA, November 6 – 11, 2006.
- Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., . . . Rush, A. M. (2020). Transformers: State-of-the-art natural language processing. In *EMNLP 2020 — Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (S. 38–45). Online, November 17 – 20, 2020.
- Wu, C., Wu, F., An, M., Huang, J., Huang, Y. & Xie, X. (2019). Neural news recommendation with attentive multi-view learning. In *IJCAI 2019 — Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence* (S. 3863–3869). Macao, China, August 10 – 16, 2019.
- Wu, C., Wu, F., An, M., Qi, T., Huang, J., Huang, Y. & Xie, X. (2019). Neural news recommendation with heterogeneous user behavior. In *EMNLP-IJCNLP 2019 — Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing* (S. 4874–4883). Hong Kong, China, November 3 – 7, 2019.
- Wu, C., Wu, F., Ge, S., Qi, T., Huang, Y. & Xie, X. (2019). Neural news recommendation with multi-head self-attention. In *EMNLP-IJCNLP 2019 — Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing* (S. 6389–6394). Hong Kong, China, November 3 – 7, 2019.
- Wu, C., Wu, F., Qi, T. & Huang, Y. (2020). User modeling with click preference and reading satisfaction for news recommendation. In *IJCAI 2020 — Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence* (S. 3023–3029). Online, January 7 – 15, 2021.

- Wu, C., Wu, F., Qi, T. & Huang, Y. (2021). FeedRec: News feed recommendation with various user feedbacks. *arXiv:2102.04903 [cs.IR]*.
- Wu, F., Qiao, Y., Chen, J.-H., Wu, C., Qi, T., Lian, J., ... Zhou, M. (2020). MIND: A large-scale dataset for news recommendation. In *ACL 2020 — Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (S. 3597–3606). Online, July 5 – 10, 2020.
- Xu, S., Jiang, H. & Lau, F. C. M. (2011). Mining user dwell time for personalized web search re-ranking. In *IJCAI 2011 — Proceedings of the 22nd International Joint Conference on Artificial Intelligence* (Bd. 3, S. 2367–2372). Barcelona, Catalonia, Spain, July 16 – 22, 2011.
- Yi, X., Hong, L., Zhong, E., Liu, N. N. & Rajan, S. (2014). Beyond clicks: Dwell time for personalization. In *RecSys 2014 — Proceedings of the 8th ACM Conference on Recommender Systems* (S. 113–120). Foster City, Silicon Valley, California, USA, October 6 – 10, 2014.
- Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., ... Ahmed, A. (2020). Big Bird: Transformers for longer sequences. In *NeurIPS 2020 — Advances in Neural Information Processing Systems* (S. 17283–17297). Vancouver, Canada, December 8 – 14, 2019.
- Zhou, T., Qian, H., Shen, Z., Zhang, C., Wang, C., Liu, S. & Ou, W. (2018). JUMP: A jointly predictor for user click and dwell time. In *IJCAI 2018 — Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence* (S. 3704–3710). Stockholm, Sweden, July 13 – 19, 2018.

Abbildungsverzeichnis

1	Beispielsession eines Nutzers, Ermittlung der Aufenthaltsdauern . . .	28
2	■■■■: Verteilung Seitenaufrufe, Textlänge, Aufenthaltsdauer	30
3	■■■■: Zusammenhang von Textlänge und Aufenthaltsdauer	31
4	Schematische Darstellung eines künstlichen neuronalen Netzwerks . .	41
5	Schematische Darstellung einer der Encoder-Schichten im Transformer	47
6	Schematische Darstellung von BERT und einem Ausgabe-Modell . . .	50
7	Architekturen von BertSequence, BertFFN und BertAveraging	56
8	Architektur von BertHiMean	59
9	Architektur von BertHiGRU	60
10	Gelernte Gewichtung der Abschnitte in BertHiMean	68
11	Visualisierung der Shapley-Werte innerhalb eines Artikels bei BertFFN	72
12	Übersicht über die Shapley-Werte ausgewählter Features in BertFFN	72
13	Visualisierung der Shapley-Werte innerhalb eines Artikels bei BOWXG- Boost	91
14	Übersicht über die Shapley-Werte ausgewählter Features in BOWXG- Boost	92

Tabellenverzeichnis

1	Überblick über den vollständigen Datensatz	24
2	■-Datensatz: Überblick über die Datensplits	27
3	KPIs und ihre Bedeutung	33
4	Evaluationsergebnisse	64
5	Die 20 positivsten und negativsten Tokens bei BertFFN	73
6	Die 20 positivsten und negativsten Tokens bei BOWXGBoost	93

A Appendix: Einblick in BOWXGBoost mit SHAP

In Abschnitt 7 wurde mithilfe von SHAP ein Einblick in die Modellvorhersagen von BertFFN gegeben. Parallel zu BertFFN wurde auch das *Bag-of-Words*-basierte Modell BOWXGBoost mit SHAP analysiert. Verwendet wurde der auf baumbasierte Modelle spezialisierte *TreeExplainer* von SHAP, der das trainierte Modell sowie die Feature-Matrix (BOW-Matrix) der zu erklärenden Datenmenge erhält. Auch hier wurden die Artikeltexte aus dem dev-Set verwendet. Es stellten sich einige Probleme heraus, die die Interpretation der Ergebnisse bei BOWXGBoost erschweren, diese werden weiter unten erläutert.

Als Äquivalent zum Textplot bei BertFFN (Abb. 11) visualisiert Abb. 13 eine *einzelne* Modellvorhersage (ein *Force Plot* von SHAP). Gezeigt werden die Features (Tokens bzw. N-Gramme und ihre Ausprägung, also die absolute Häufigkeit im betrachteten Text), die für die Vorhersage eine besonders große (absolute) Auswirkung hatten. Rote Färbung entspricht positiver Auswirkung auf die Zielvariable (Aufenthaltsdauer), blaue Färbung entspricht negativer Auswirkung.

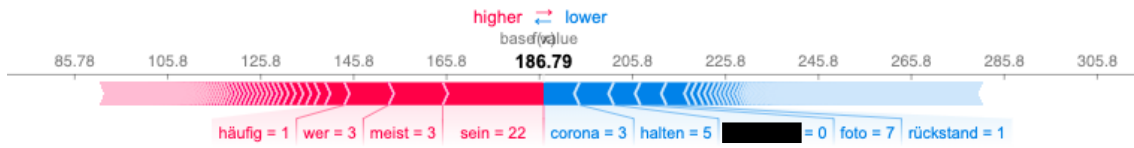


Abbildung 13: Visualisierung der Shapley-Werte innerhalb eines Artikels bei BOWXGBoost.

Auch hier werden einzelne Features *global*, also über alle Vorhersagen, betrachtet. Äquivalent zu den Histogrammen bei BertFFN (siehe Abb. 12), können hier die einzelnen Featureausprägungen (absolute Häufigkeit) und die erhaltenen Shapley-Werte abgetragen werden, wie in Abb. 14 beispielhaft für die Features *foto*, [REDACTED] und *sein*⁵⁸ getan. Jeder Punkt entspricht hier einem Dokument (Artikel).

Auch für BOWXGBoost wurden die 20 Tokens mit (durchschnittlich) größter positiver und die 20 Tokens mit größter negativer Auswirkung auf die Vorhersagen betrachtet, siehe dafür (parallel zu Tabelle 5 für BertFFN) die Ergebnisse in Tabelle 6.

⁵⁸Die Präsenz von *sein* als Feature ist eigentlich ein unerwünschtes Artefakt, schließlich wurden Stopwörter in der Präprozessierung entfernt. Zu erklären ist dieser Fehler durch ein ungünstiges Zusammentreffen im Präprozessierungsvorgang: Der Artikeltext wurde zunächst lemmatisiert, anschließend eine (ebenso lemmatisierte) Stopwortliste angewendet. Während Flexionsformen vom Verb *sein* (etwa *ist*, *war*) in das Lemma *sein* überführt werden, wird das Auftreten der Form *sein* zu *mein* (aufgrund der Ambiguität von *sein* als Verbinfinitiv und Possessivpronomen). Daher wurden Flexionsformen von *sein* im Text nicht als Stopwörter erkannt. Es scheint sich aber um einen Einzelfall zu handeln, andere Stopwörter wurden erfolgreich in der Präprozessierung entfernt.

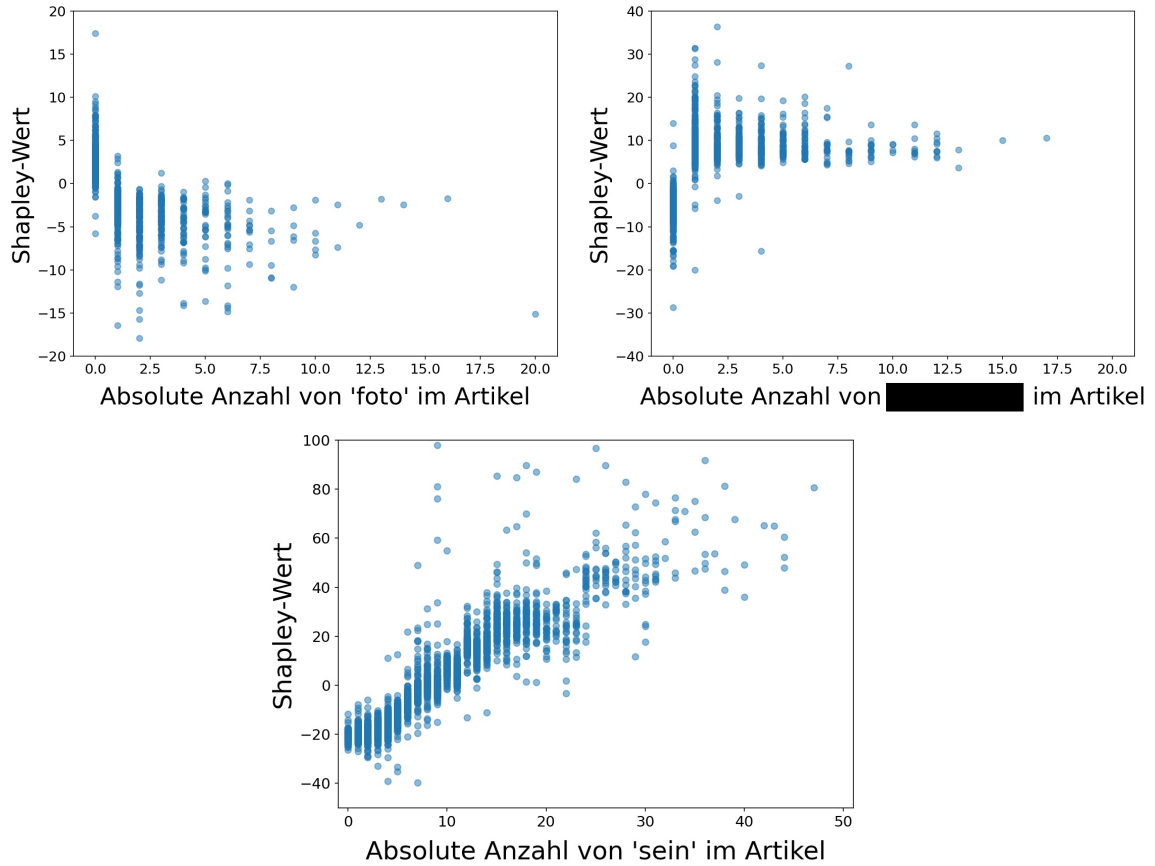


Abbildung 14: Shapley-Werte für *foto*, [redacted] und *sein* bei BOWXGBoost.

Die Interpretierbarkeit der Ergebnisse bezüglich der (positiven und negativen) Relevanz einzelner Features ist bei BOWXGBoost deutlich schwerer, es lassen sich kaum konsistente Beobachtungen aufstellen. Bei genauerer Überlegung konnten verschiedene Probleme erkannt werden, die die Aussagekraft und Interpretierbarkeit der SHAP-Ergebnisse bei BOWXGBoost im Vergleich mit BertFFN erschweren bzw. unmöglich machen⁵⁹. Diese werden hier kurz angerissen:

Die beiden Modellklassen – BERT und XGBoost – funktionieren sehr unterschiedlich und beruhen auf unterschiedlichen Feature-Strukturen. BertFFN sieht immer nur einen *Teil* eines Artikels (die ersten 512 BERT-Tokens). Aus diesem muss das Modell ableiten, ob es sich insgesamt (also inklusive des unbekannten Teils) um einen Artikel mit niedriger oder hoher Aufenthaltsdauer handelt. Eine hohe positive Relevanz bei BertFFN bedeutet also, dass das Feature (Sub-Token) vor allem in Texten mit hoher Aufenthaltsdauer zu finden ist. Das ermöglicht einige sinnvolle Schlüsse (siehe Abschnitt 7).

⁵⁹Bei Betrachtung der Ridge-Variante BOWRidge lassen sich ähnliche Ergebnisse und Probleme beobachten.

Feature	count	mean	mean_abs	Feature	count	mean	mean_abs
sein	31583	0.866	16.455	kind	1752	-0.181	0.466
sehen	1589	0.147	0.595	frau	1117	-0.134	0.262
geben	4217	0.126	2.138	corona	2520	-0.094	1.651
foto	1848	0.124	1.87	mutter	362	-0.091	0.277
erklären	1426	0.11	0.753	meist	468	-0.06	1.783
100	460	0.102	0.17	schreiben	459	-0.052	0.866
deutschland	1141	0.09	0.997	mögen	658	-0.05	0.117
wer	900	0.088	1.401	mal	1160	-0.042	0.112
januar	408	0.061	0.423	müssen	1734	-0.033	0.248
unternehmen	1010	0.049	0.401	wohl	426	-0.028	0.197
neu	1889	0.048	0.983	spät	756	-0.025	0.322
liegen	1033	0.047	0.928	immer	1897	-0.024	0.177
000	1097	0.041	0.147	gleichen	308	-0.022	0.061
tatsächlich	267	0.041	0.756	land	1100	-0.022	0.126
sagen	4551	0.04	0.883	bremen	223	-0.022	0.075
■	570	0.039	0.295	schon	2466	-0.022	0.087
ab	1696	0.039	1.16	straße	1358	-0.02	0.457
dabei	1495	0.037	0.139	sein wichtig	258	-0.02	0.15
lang	1098	0.036	0.546	weiß	478	-0.019	0.663
oft	408	0.035	0.231	warum	437	-0.018	0.282

Tabelle 6: Die 20 positivsten (links) und negativsten (rechts) Tokens bei BOWXGBoost (laut *mean*-Wert, nur Tokens mit Frequenz ≥ 200 betrachtet).

Im Gegensatz dazu erhält das BOWXGBoost-Modell Information über den *vollständigen* Text (als BOW-Matrix). Wichtiger ist aber: Es hat aufgrund der *absoluten* Häufigkeiten als Feature-Ausprägung unmittelbaren Zugriff auf die tatsächliche Textlänge (wenn von der Entfernung der Stopwörter und der maximalen Anzahl an betrachteten Features abgesehen wird). Ein positiver Shapley-Wert eines Features (Tokens bzw. N-Gramm) kann daher unterschiedliche Gründe haben, und dies wird in Tabelle 6 deutlich:

Erstens: Das Vorkommen von häufigen „Füllwörtern“ spricht allgemein einfach für einen längeren Text (wie etwa *sehen*, *geben*, *liegen* und auch *sein*). Daher spricht eine hohe absolute Frequenz dieser Wörter völlig unabhängig von ihrer Bedeutung für eine höhere Aufenthaltsdauer.

Oder zweitens: Das Auftreten von Wörtern spricht tatsächlich „inhaltlich“ für einen Text mit längerer Aufenthaltsdauer, da es etwa einen spannenden oder allgemein lesenswerten Artikel vermuten lässt (dies könnte bei *erklären* oder ■ (der Fußballverein in ■ ist der ■) der Fall sein).

Diese beiden Phänomene können aber unmöglich getrennt werden, daher ist die Anwendung von SHAP hier nicht besonders ertragreich.

Generell ist die Einordnung eines Feature als *positiv* oder *negativ* anhand des Mittelwerts der Shapley-Werte problematisch. Dies wird anhand der Scatterplots in Abb. 14 deutlich: Ein Token erhält – im Gegensatz zu BertFFN – auch dann einen Shapley-Wert, wenn es eben *nicht* im Text vorkommt (also mit der absoluten Frequenz 0). Dieser Shapley-Wert kann sowohl positiv als auch negativ sein. Hier entsteht eine gewissermaßen paradoxe Interpretation: Ein positiver Shapley-Wert spricht hier für die positive Auswirkung, die die *Abwesenheit* des Tokens auf die Vorhersage der Aufenthaltsdauer hat. Dies ist konträr zur Bedeutung von positiven Shapley-Werten bei einer absoluten Häufigkeit > 0 . Bei der Mittelung werden aber alle Shapley-Werte gleich behandelt, dies führt zu einer (möglichen) Fehlinterpretation.

Konkret kann die Problematik am Beispiel des Tokens *foto* deutlich gemacht werden: Laut Tabelle 6 hat das Token *foto* im Durchschnitt über alle Vorhersagen betrachtet eine positive Auswirkung auf die Aufenthaltsdauer. Bei einer einzelnen Betrachtung des Features (siehe erster Plot in Abb. 14) zeigt sich aber: Die positiven Shapley-Werte gehören fast ausschließlich zu den Dokumenten, in denen das Token *foto* gerade *nicht* vorliegt, also mit absoluter Frequenz 0. Dokumente mit häufiger Nennung von *foto* (die X-Achse zeigt die Anzahl in den Artikeln) tragen sogar tendenziell negative Shapley-Werte: Je mehr ein Artikel also über Fotos spricht, desto *geringer* ist seine Aufenthaltsdauer. Dies steht konträr zu dem positiven Shapley-Wert aus Tabelle 6. Die gleiche Problematik, nur in umgekehrter Richtung, zeigt sich am Token ██████████, siehe zweiter Plot in Abb. 14. Dies zeigt, dass die mittleren Shapley-Werte nicht besonders gut geeignet sind, um von ihnen auf die Auswirkung eines Features auf die Aufenthaltsdauer zu schließen.

Dies bedeutet nicht, dass grundsätzlich von der Modellierung anhand von absoluten Häufigkeiten Abstand zu nehmen ist. Der mittelbare Einbezug der Textlänge ist für die Vorhersage der Aufenthaltsdauer durchaus sinnvoll und BOWXGBoost stellt eine sehr gute Baseline dar. Deutlich wird aber: SHAP ist kein geeignetes Tool zur Interpretation dieses Modells bzw. seiner Vorhersagen.

Eigenständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Hilfsmittel und Quellen angefertigt habe. Die eingereichte Arbeit ist nicht anderweitig als Prüfungsleistung verwendet worden oder in deutscher oder einer anderen Sprache als Veröffentlichung erschienen.

Seitens der Verfasserin bestehen keine Einwände, diese Arbeit für die öffentliche Benutzung zur Verfügung zu stellen.

Jena, den 14. Juni 2021

Susanna Rücker