FOM Hochschule für Oekonomie & Management

Studienzentrum Berlin

Seminararbeit

über das Thema

Automatisierte Generierung politischer Reden auf Basis eines deutschsprachigen Korpus

von

Michael Schwabe, Louis Matheoschat, Florian Lorisch, Paul Abisch

Studiengang:	Big Data & Business Analytics (M.Sc.)
Fachsemester:	3
Lehrveranstaltung:	Big-Data-Analyseprojekt
Dozent:	Philipp Koch

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

AFD ..... *Alternative für Deutschland*
AWD-LSTM ..... *ASGD Weight-Dropped LSTM*
BERT ..... *Bidirectional Encoder Representations from Transformers*
BLEU ..... *Bilingual Evaluation Understudy*
CDU ..... *Christlich Demokratische Union Deutschlands*
CSU ..... *Christlich-Soziale Union in Bayern*
FDP ..... *Freie Demokratische Partei*
GloVe ..... *Global Vectors für Word Representation*
GPT-2 ..... *Generative Pretrained Transformer 2*
GPT-3 ..... *Generative Pretrained Transformer 3*
GPU ..... *Graphics Processing Unit*
LSTM ..... *Long-Short-Term-Memory*
NLG ..... *Natural Language Generation*
NLP ..... *Natural Language Processing*
NLU ..... *Natural Language Understanding*
NT-ASGD ..... *Non-monotonically Triggered Average-Stochastic Gradient Descent*
POS-Tagging ..... *Part-of-speech-Tagging*
RNN ..... *Rekurrentes Neuronales Netz*
ROUGE ..... *Recall-Oriented Understudy for Gisting Evaluation*
SPD ..... *Sozialdemoraktische Partei Deutschlands*
VRAM ..... *Video Random Access Memory*

Abstract

Die automatisierte Generierung von Texten mithilfe von Natural Language (NLG) Technologien, wird bereits in Bereichen wie Marketing und im Kundenservice eingesetzt, wie beispielsweise bei der Umsetzung interaktiver Dialogsysteme. Die Fähigkeit Text zu synthetisieren bietet darüber hinaus Potential für die Erschließung weiterer Anwendungsfelder. Ausgehend von vergleichbaren Arbeiten im englischsprachigen Raum werden mit dieser Arbeit unterschiedliche Sprachmodelle zur automatisierten Generierung politischer Reden in deutscher Sprache erprobt und verglichen. Der Fokus liegt auf einem Markov-Modell, unterschiedlichen LSTM-Architekturen sowie den Transformerarichtekturen BERT und GPT-2. Als Datensatz (Korpus) für das Training und Finetuning der Modelle werden, die durch das Open Discourse Projekt aufbereiteten Plenarprotokolle des deutschen Bundestags verwendet. Im Ergebnis wird deutlich, dass sich die Performanz, die durch englische Sprachmodelle erreicht werden, nicht ohne weiteres auf Basis eines deutschen Korpus replizieren lassen. Die Arbeit dient als Ausgangspunkt für weitere Forschungsprojekte zur Generierung politischer Reden in deutscher Sprache. Dabei sollte insbesondere die Verwendung von Transformerarchitekturen näher betrachtet werden.

1 Einleitung

1.1 Inhaltliche Einführung

Automatische Textgenerierung oder auch Natural Language Generation (NLG) ermöglicht die Generierung von Texten in menschlicher Sprache mithilfe algorithmischer und statistischer Verfahren. Die automatische Generierung von Text kann entweder auf der Basis von Daten (data-to-text) oder auf Grundlage eines textuellen Korpus (text-to-text) erfolgen (Dale & Reiter, 1997).

NLG-Verfahren finden bereits heute in vielen Bereichen des alltäglichen Lebens Verwendung. NLG-Verfahren bilden die grundlegende Technologie für die automatische Generierung von Wetterberichten auf Basis von Wetterdaten, die Erstellung journalistischer Texte (Robo-Journalismus) sowie für digitale Assistenten wie Amazons Alexa oder Chatbots wie Zendesk, die häufig im Bereich des Kundenservices Anwendung finden (Shum et al., 2018). Die Anwendungsbereiche für NLG sind bereits heute vielfältig und werden kontinuierlich erweitert (Gatt & Krahmer, 2017).

1.2 Problemidentifikation und Zielstellung

Im englischsprachigen Raum gibt es erste Ansätze für die Verwendung von NLG-Verfahren im politischen Kontext (Bullock & Luengo-Oroz, 2019; Kassarnig, 2016). Dabei werden NLG-Verfahren zur automatischen Generierung politischer Reden, im Stil von Reden, die im US-Kongress respektive der UN-Generalversammlung gehalten werden, eingesetzt. Aufbauend darauf, ist es Gegenstand der vorliegenden Arbeit die Möglichkeiten zur automatischen Generierung politischer Reden in deutscher Sprache mithilfe von NLG-Verfahren zu erkunden.

Die Datengrundlage dieses Vorhabens ist ein Datensatz aller digitalisierten Plenarprotokolle seit der ersten Sitzung des Bundestages im Jahr 1949 bis Februar 2020. Wichtiger Bestandteil der Plenarprotokolle sind sämtliche Bundestagsreden, die im Rahmen der Sitzungen gehalten werden. Bundestagsreden sind wie politische Reden im Allgemeinen ein geeigneter

Ausgangspunkt für die Verwendung von NLG-Verfahren, da sie unabhängig vom thematischen Schwerpunkt ähnliche Strukturen und Merkmale aufweisen.

Die Reden bestehen in der Regel aus einer Einleitung mit Danksagung an den Bundestagspräsidenten, einem argumentativen Hauptteil und einem kontextualisierenden und zusammenfassenden Schlusswort. Einige Phrasen und Argumente werden von unterschiedlichen Politikern wiederholt vorgetragen und geben Rückschlüsse auf die politische Zugehörigkeit oder Meinung. Auf diese Weise lassen sich beispielsweise Reden mit eindeutigen Themenschwerpunkten und einer konsistenten politischen Verortung erstellen. Das hohe Maß an Struktur sowie die inhaltlichen Abgrenzungen zwischen den einzelnen Bundestagsparteien und Rednern macht Bundestagsreden demnach sehr zugänglich für die Verarbeitung durch moderne NLG-Verfahren.

Die vorliegende Seminararbeit setzt auf den bestehenden Arbeiten aus dem englischsprachigen Raum auf (Bullock & Luengo-Oroz, 2019; Kassarnig, 2016). Ziel der Arbeit ist es, die darin gewonnen Erkenntnisse auf Basis eines deutschsprachigen Korpus zu replizieren und zu ergänzen. Neben den durch Bullock und Luengo-Oroz (2019) sowie Kassarnig (2016) verwendeten Modellen werden auch sogenannte Transformerarchitekturen verprobt, die aktuell zunehmend an Popularität gewinnen.

1.3 Kapitelübersicht

Ausgehend von einer Einbettung der Natural Language Generation in das Feld des Natural Language Processings bietet das folgende Kapitel eine Übersicht über relevante technologische Entwicklungen und aktuelle Forschungsansätze. In Kapitel 3 werden die Projektziele sowie der verwendete Datensatz näher beschrieben. In Kapitel 4 wird die Methodik und das Vorgehen zur Realisierung der Projektziele aufgezeigt. Der Fokus liegt dabei auf der Beschreibung der Datenaufbereitung und den verwendeten NLG-Verfahren. Die Ergebnisse werden anhand von Auszügen generierter Reden in Kapitel 5 diskutiert. Kapitel 6 umfasst eine kritische Zusammenfassung der Ergebnisse und bietet Ansatzpunkte für weitere Arbeiten in dem Feld.

2 Einführung in das Feld des Natural Language Processings und der Natural Language Generation

2.1 Zusammenhang zwischen NLP, NLU und NLG

Unser Alltag wird zunehmend von Anwendungen geprägt, die mithilfe von Technologien aus dem Bereich des Natural Language Processing (NLP) die Mensch-Maschinen-Kommunikation in natürlicher Sprache ermöglichen. NLP ist eng verwandt mit dem Forschungsfeld der Computational Linguistics, welches sich mit der Analyse menschlicher Sprache mithilfe von Computer gestützten Verfahren befasst, um das Verständnis menschlicher Sprache zu vertiefen. Das Ziel von NLP ist es entsprechende computergestützte Verfahren zur Verarbeitung menschlicher Sprache zu entwickeln, zu verbessern und zur Verfügung zu stellen (Reiter & Dale, 1997, S. 1).

NLP-Technologien bilden beispielsweise die Grundlage für automatisierte maschinelle Übersetzungen und die Umsetzung von Spamfiltern per Textklassifizierung. NLP verbessertet aber auch Suchalgorithmen und ermöglicht die Verbesserung der Mensch-Maschinen-Interkation mithilfe moderner Dialogsysteme wie Chatbots und sprachgesteuerten, digitalen Assistenten wie Siri und Alexa (Eisenstein, 2019, S. 2).

Diese Vielzahl an Anwendungen basiert auf der gezielten Verknüpfung von Algorithmen, Linguistik, Logik und Statistik. NLP umfasst die Gesamtheit der aktuell verfügbaren Methoden, um menschliche Sprache für Maschinen zugänglich zu machen (Eisenstein, 2019, S. 1). Das durch NLP-Technologien realisierte, maschinelle Verständnis menschlicher Sprache, basiert auf der Transformation von Texten in maschinell lesbare Repräsentationen wie Tensoren, Vektoren und Graphen (Rao & McMahan, 2019, S. 1)

Das Forschungsfeld Natural Language Processing umfasst die zwei Unterbereiche des Natural Language Understandings (NLU) und der Natural Language Generation (NLG). NLU bezieht sich auf das Verständnis von (Text-) Daten auf der Grundlage der Grammatik und des Kontextes (Allen, 1995). NLG

umfasst Algorithmen, die zur Generierung menschlicher Sprache eingesetzt werden und bildet die Grundlage der vorliegenden Seminararbeit.

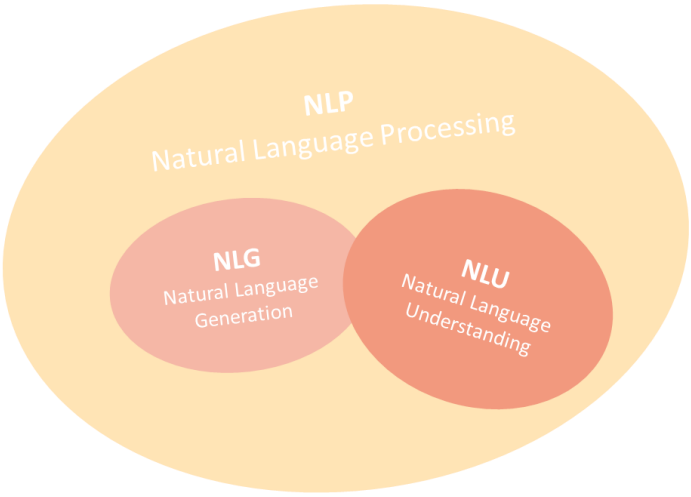


Abbildung 1: Zusammenhang zwischen NLP, NLU und NLG

2.2 Einführung in Natural Language Generation

Reiter und Dale (1997) definieren NLG als "*the construction of computer systems that can produce understandable texts in English or other human languages* " (1997, S. 1).

Die Generierung von Text durch NLG erfolgt entweder auf Basis von Daten (Data-to-Text) oder auf Basis bestehender Texte (Text-to-Text). So ermöglicht NLG die automatische Generierung von Wetterberichten auf Basis von Wetterdaten, ebenso wie die Zusammenfassung, Übersetzung und Neugestaltung bestehender Texte (Gatt & Krahmer, 2017, S. 67–68).

Bei der Generierung von Texten mithilfe von NLG wird grundsätzlich zwischen sechs Teilschritten unterschieden: 1. *Content determination*. 2. *Discourse planning*, 3. *Sentence aggregation*, 4. *Lexicalization*, 5. *Referring expression generation* und 6. *Linguistic realization* (Reiter & Dale, 1997, S. 9–12).

Im Zuge der Content Determination werden Informationen identifiziert, die durch den zu generierenden Text wiedergegeben werden sollen. Discourse Planning legt fest, welchen Aufbau ein Text haben soll (bspw. Einleitung, Mittelteil, Schluss), wie die Argumentation verläuft und welche Informationen an welcher Stelle im Text verarbeitet werden sollen. Sentence Agreggation kann optional

durchgeführt werden, um inhaltlich zusammenhängende Sätze zu aggregieren. Die Teilschritte Lexicalization und Referring Expression Generation bauen aufeinander auf und ermöglichen es Begriffe, die verwendet werden sollen, um einen Sachverhalt darzustellen, zu identifizieren und diese Begriffe von anderen abzugrenzen. Linguistic Realisation bezieht sich auf die eigentliche Erstellung von Text unter der Verwendung grammatikalischer Regeln (ebd.).

Die hier beschriebenen Aufgaben im Bereich des NLG werden in der Praxis auf vielfältige Weise abgebildet. Die einzelnen Aufgaben können mithilfe eines oder mehrerer „NLG-Module“ durchgeführt werden. Die NLG-Technologien die zur Erledigung entsprechender Aufgaben genutzt werden haben sich seit Mitte des 20. Jahrhundert stetig weiterentwickelt.

2.3 Entwicklungen im Bereich der Natural Language Generation seit den 1940er Jahren

Der zentrale Bestandteil des Natural Language Generation sind sogenannte Language Models, beziehungsweise Sprachmodelle. Sprachmodelle sind Modelle zur Ermittlung von Wahrscheinlichkeiten für die Reihenfolge von Wörtern, was wiederum die Grundlage zur Generierung neuer Sätze in menschlicher Sprache darstellt (Eisenstein, 2019, S. 125).

Forschung im Bereich von Sprachmodellen gibt es bereits seit den späten 1940er Jahren, wobei sich die Geschwindigkeit bei der Entwicklung neuer Sprachmodelle, in den letzten zwei Jahrzenten vervielfacht hat. Die zunehmende Verfügbarkeit von Daten sowie Fortschritte in den Bereichen der Prozessor- und Speichertechnologie spielen bei dieser Entwicklung eine entscheidende Rolle. Moderne Sprachmodelle, sind wie andere Verfahren im Bereich des maschinellen Lernens datenhungrig und ressourcenintensiv. (Bullock & Luengo-Oroz, 2019, S. 1)

N-Gramme und Markov-Ketten

Die ersten Sprachmodell basieren auf sogenannten N-Grammen und werden in den späten 1940er Jahren entwickelt. Ein N-Gramm beschreibt eine Sequenz mit *N* Wörtern. Ein 2-Gramm (auch Bigramm) ist demnach eine Sequenz, die

aus zwei Worten besteht, wie beispielsweise „Hallo Peter“, „Ich bin“ oder „dein Vater“. Ein 3-Gramm (Trigramm) ist dementsprechend eine Wortsequenz mit der Länge drei, wie beispielsweise „Du heißt Peter“ oder „Ich lese Zeitung“ (Shannon, 1948).

Der Begriff N-Gramme wird zur Beschreibung dieser Wortsequenzen sowie zur Bezeichnung gleichnamigen Modellarchitekturen verwendet. N-gramm-Modelle sind wenig komplexe probabilistische Sprachmodelle, die jeweils das letzte Wort eines N-Gramms vorhersagen können, wenn die *N* vorangegangene Wörter gegeben sind. Der Vorteil eines N-Gramm-Modells besteht darin, dass nicht die gesamte Sequenz (ein Text) ausgewertet werden muss, um die Wahrscheinlichkeiten für das nächste Wort zu berechnen.

Wenn alle vorangegangen Worte gegeben sind, berechnet ein Bigramm-Modell zum Beispiel die Wahrscheinlichkeit des folgenden Wortes, indem es die bedingte Wahrscheinlichkeit des vorhergehenden Wortes verwendet.

Die grundlegende Annahme bei der Verwendung eines Bigramm-Modells ist demnach, dass die Wahrscheinlichkeit eines Wortes nur von dem vorhergehenden Wort abhängt. Diese Annahme wird auch als Markov-Annahme bezeichnet. N-gram basierte Sprachmodelle werden demnach auch Markov-Modelle genannt (Jurafsky & Martin, 2000).

In der Praxis sind N-Gramme nicht für die Verarbeitung komplexerer und längerer Texte geeignet. Als Antwort auf diese Herausforderungen werden Anfang der 1980er Jahre die ersten Sprachmodelle auf Basis rekurrenter neuronaler Netze (RNN) entwickelt (Eisenstein, 2019, S. 133).

RNN

RNN sind künstliche neuronale Netze mit rekurrenten Schichten. Rekurrente Schichten ermöglichen es, dass ein Neuron nicht nur den Input der aktuellen Iteration verarbeitet, sondern auch den Input aus einer vergangenen Iteration berücksichtigt. RNN besitzen also einen sogenannten Internal State, also eine Art Gedächtnis. Dies macht RNN insbesondere für die Verarbeitung sequenzieller Daten interessant. RNN sind demnach grundsätzlich sehr gut für

die Verarbeitung menschlicher Sprache geeignet (Eisenstein, 2019, S. 133–137).

Der Vorteil von RNN Sprachmodellen liegt in der Theorie darin, dass Informationen aus vorangegangenen Iterationen in die Berechnung des jeweils folgenden Worts einfließen. Ein weiterer Vorteil liegt darin, dass die Größe von RNN Sprachmodellen kontrollierbar ist. Im Gegensatz zu N-Gramm Modellen wächst die Modelgröße bei RNN Sprachmodellen nicht analog zur Länge der als Input verwendeten Sequenz (ebd.).

In der Praxis weisen RNN-Sprachmodelle zwei zentrale Nachteile auf. Zum einen ist die rekurrente Berechnung zeitaufwendig. Zum anderen, sind RNN in der Praxis nur bedingt in der Lage Informationen aus weiter zurückliegenden Iterationen zu verarbeiten. Mit steigender Zahl neuer Eingabewerten wird der Internal State und die rekurrente Wirkung der ersten Eingabewerte weniger berücksichtigt. Daraus geht hervor, dass RNN keine langfristigen Abhängigkeiten abbilden können. Dies wird auch als Long-term Dependencies Problem bezeichnet und führt dazu, dass die Länge der Textsequenzen, die durch RNN generiert werden können, stark limitiert ist (Bengio et al., 1994).

LSTM

Im Jahr 1997 veröffentlichen Hochreiter und Schmidhuber eine neue RNN Architektur. Long-Short-Term-Memory (LSTM) Netze sind explizit darauf ausgerichtet, die Herausforderungen klassischer RNN-Architekturen zu adressieren. LSTM-Netze unterscheiden sich von der klassischen RNN-Architektur darin, wie die rekurrente Struktur aufgebaut ist. Anstelle einer einzigen neuronalen Schicht weisen LSTM-Netze insgesamt vier Schichten auf, die untereinander interagieren (Eisenstein, 2019, S. 137–139).

Das entscheidende Merkmal der LSTM-Architektur ist der sogenannte Zellzustand (Cell State). Der Zellzustand verläuft geradlinig entlang der gesamten Kette, mit nur einigen geringfügigen linearen Interaktionen und ermöglicht es so Informationen unverändert weiterzugeben. Dem Zellzustand werden Informationen entnommen oder hinzugefügt. Dies ist möglich, indem sogenannte Gatter (Gates) Informationen selektiv durchlassen.

Zu dem Zeitpunkt als Hochreiter und Schmidhuber das erste LSTM veröffentlichen, jedoch fehlt es an der notwendigen Rechenleistung, um das Potential dieser Architektur für neuronale Sprachmodelle auszuschöpfen.

Vortrainierte Word Embeddings

Im Jahr 2013 veröffentlicht Google Word2Vec. Mit Word2Vec entwickelten Mikolov et al., 2013 eine Lösung um qualitativ hochwertige Word Embeddings aus umfassenden Korpora lernen zu können, die über NLP-Anwendungen hinweg genutzt werden können. Die Ergebnisse dieser vortrainierten Word Embeddings ebnen den Weg für eine Vielzahl von NLP-Anwendungen. Ausgehend von Word2Vec können beispielsweise RNN, LSTM sowie viele andere NLP-Anwendungen schneller trainiert und weiterentwickelt werden.

Aufmerksamkeitsmodelle

Basierend auf dem Anwendungsfall der automatisierten Übersetzung mithilfe neuronaler Netze (Neural Machine Translation) entwickeln Bahdanau et al., 2014 einen neuen Ansatz, um das bereits im Rahmen von RNNs beschriebene Long-term-Dependencies-Problem zu adressieren. Der daraus resultierende Aufmerksamkeitsmechanismus eröffnet neue Wege der automatisierten Übersetzung. Der Aufmerksamkeitsmechanismus ermöglicht es, dass sich Modelle auf Eingabewerte (Worte) konzentrieren, die für eine gute Übersetzung ausschlaggebend sind, anstatt ganze Eingabesequenzen auf einmal verarbeiten zu müssen.

Nach dem großen Erfolg und der schnellen Adoption von Aufmerksamkeit im Bereich des NLP entwickelt Google eine neue und vereinfachte Netzwerkarchitektur, die heute als Transformer bekannt ist (Vaswani et al., 2017)). Die Entwicklung der Transformerarchitektur markiert einen Paradigmenwechsel im Bereich der Sprachmodelle. Im Gegensatz zu anderen Modellen basiert die Transformerarchitektur ausschließlich auf dem Aufmerksamkeitsmechanismus und verzichtet auf die RNN Architektur (ebd.).

Die Arbeiten von Vaswani et al. (2017) führen bereits im Jahr 2018 zur Entwicklung weiterer Modelle, die heute den Stand der Technik im Bereich der Sprachmodelle darstellen. Dazu gehört die ebenfalls von Google entwickelte

Bidirectional Encoder Representations from Transformers (BERT) Architektur (Devlin et al., 2018) sowie OpenAi GPT-2 (Radford et al., 2019). Vortrainierte Sprachmodelle wie BERT und GPT-2 markieren den zwischenzeitlichen Höhepunkt der rasanten Weiterentwicklung von Sprachenmodellen.

Die Funktionsweise der im Zuge der vorliegenden Seminararbeit implementierten Architekturen (Markov-Kette, LSTM, BERT und GPT-2) sind in 4 im Detail beschrieben.

2.4 Anwendung von Natural Language Generation im politischen Kontext

Wie bereits Eingangs beschrieben, eröffnet die rasante Entwicklung im Bereich des Natural Language Processings im Allgemeinen und dem Bereich der Natural Language Generation im speziellen, eine Vielzahl neuer Anwendungsfelder (siehe Kapitel 1.1). Auch in Bezug auf das Thema der vorliegenden Seminararbeit existieren erste Ansätze zur automatischen Generierung politischer Reden in englischer Sprache.

Kassarnig (2016) entwickelt ein Verfahren zur automatischen Generierung politischer Reden auf Basis von Reden aus dem US-Kongress. Der verwendete Datensatz umfasst insgesamt 3587 Redebeiträge aus 53 Debatten des US-Kongresses im Jahr 2005. Im Rahmen der Datenaufbereitung werden kurze Redebeiträge (Single-Sentence-Speeches) eliminiert und Start- und Stop-Tokens eingefügt, um eine automatisierte Trennung von Reden zu ermöglichen. Zudem wird der Datensatz entsprechend der politischen Parteien (Republikaner und Demokraten) sowie des Sentiments der Reden (Zustimmung oder Ablehnung) in vier separate Datensätze unterteilt.

Auf Basis dieser Datensätze wird ein Topic Modelling mithilfe von POS-Tagging (Part-of-Speech Tagging) (Justeson & Katz, 1995) durchgeführt. Das Topic Modelling ermöglicht die Identifizierung wiederkehrender Themen innerhalb der oben beschriebenen Kategorien. Um sicherzugehen, dass generische Themen nicht übergewichtet werden, wird zudem eine Signifikanzberechnung durchgeführt. Auf diese Weise werden mithilfe des Topic Modellings Themen

identifiziert, die kategoriespezifisch sind, also in einer Kategorie mit einer höheren Wahrscheinlichkeit auftreten als in einer anderen Kategorie.

Das verwendete Sprachmodell ist ein auf N-Grammen (6-Gramm) basiertes Markov-Modell. Die automatisierte Textgenerierung wird durch einen randomisierten Eingabewert (ein Wort aus dem Korpus) initialisiert. Die folgenden Worte werden mithilfe der Wahrscheinlichkeit des Sprachmodels sowie der Wahrscheinlichkeit des Topic Modells berechnet. Ausgehend von den 6-Grammen werden mithilfe der Wahrscheinlichkeit des Sprachmodells die Kandidaten für das folgende Wort unter Berücksichtigung der jeweils fünf vorrangegangenen Wörter priorisiert. Die Wahrscheinlichkeit des Topic Modells ermöglicht es darüber hinaus zu beschreiben, ob ein Wort thematisch zu den bereits generierten Wörtern passt.

Zur Evaluierung der generierten politischen Reden nutzt Kassarnig einen zweistufigen Evaluierungsprozess bestehend aus manueller und automatisierter Evaluierung. Im Zuge der automatisierten Evaluierung werden die POS-Tags, der generierten Reden identifiziert und mit den POS-Tags der im Korpus vorhandenen Sätze abgeglichen. Werden dabei Sätze mit identischer POS-Tag-Struktur im Korpus gefunden, schließt Kassarnig daraus, dass die grammatikalische Struktur des generierten Satzes tendenziell gut ist.

Darüber hinaus ermöglicht die automatisierte Evaluierung den inhaltlichen Abgleich generierter Reden mit den Inhalten des Korpus. Ziel dieses inhaltlichen Abgleichs ist die Überprüfung, ob die innerhalb einer generierten Rede aufgegriffenen Themen sich kohärent zu der Verteilung der Themen im Korpus verhalten. Auf diese Weise könnte beispielsweise eine aus Sicht eines Republikaners generierte Rede, die sich restriktiv zum Waffenrecht äußert als inhaltlich inkonsistent identifiziert werden.

Die manuelle Evaluierung ermöglicht eine tiefergehende inhaltliche und grammatikalische Prüfung der generierten Reden, unter Berücksichtigung menschlichen Domänen-Wissens. Die manuelle Evaluierung erfolgt auf Basis eines Kriterienkatalogs. Folgende Kriterien werden berücksichtigt:

- Grammatikalische Richtigkeit,
- Satzübergänge,
- Aufbau der Rede,
- Inhalt der Rede

Innerhalb der Kriterien werden die generierten Reden auf einer Punkte Skala (1-5) bewertet. Auf diese Weise können Reden überprüft werden, die im Zuge der automatisierten Evaluierung nicht eindeutig bewertet werden konnten

Unter Berücksichtigung der in Kapitel 2 beschriebenen technologischen Entwicklung im Bereich des NLG wird deutlich, dass das von Kassarnig (2016) als Sprachmodell verwendete Markov-Modell nicht dem aktuellen Stand der Technik entspricht. Die durch den Autor zur Verfügung gestellten generierten Reden sind dennoch in weiten Teilen inhaltlich schlüssig und grammatikalisch korrekt. Dabei ist jedoch zu berücksichtigen, dass es sich lediglich um einen vom Autor selektierten Auszug der generierten Reden handelt.

Bullock und Luengo-Oroz (2019) nutzen im Gegensatz zu Kassarnig ein neuronales Sprachmodell zur Generierung politischer Reden. Die Datenbasis bilden alle Reden, die durch politische Vertreter einzelner Länder im Rahmen von Generalversammlungen der Vereinten Nationen zwischen 1970 und 2015 gehalten wurden. Der verwendete Datensatz umfasst 7507 Reden mit unterschiedlichen thematischen Schwerpunkten.

Im Rahmen der Datenaufbereitung werden die Reden in 283.593 Absätze unterteilt, bereinigt und mithilfe der SpaCy Bibliothek tokenisiert. Als Sprachmodell nutzen Bullock und Luengo-Oroz (2019) ein AWD-LSTM. Zum Zeitpunkt der Veröffentlichung waren AWD-LSTM nahe am Stand der Technik im Bereich der Sprachmodellierung. Die Abkürzung AWD-LSTM steht für ASGD Weight-Dropped LSTM. Es verwendet DropConnect und eine Variante von Average-SGD (NT-ASGD) sowie weitere etablierte Regularisierungsmethoden. Durch die Verwendung von DropConnect sind AWD-LSTM Netzwerke im Vergleich zu anderen RNN Architekturen sehr robust gegen Überanpassung (Overfitting) (Merity et al., 2017).

Das AWD-LSTM wird auf Basis des Wikitext-103 Datensatzes vortrainiert und mit dem eingangs beschriebenen Datensatz feinjustiert. Für die automatisierte Textgenerierung wird das AWD-LSTM Sprachmodell mit dem Anfang eines Satzes oder Absatzes aus dem Korpus initialisiert. Die Textgenerierung wird durch die Autoren auf zwei bis fünf Sätze (50-100 Wörter) je Thema beschränkt. Die Autoren testen das Modell in unterschiedlichen Kontexten: (1) minimaler Input - nur ein einfaches Thema, (2) automatische Vervollständigung der Anmerkungen des UNO-Generalsekretärs und (3) Behandlung sensibler Themen wie Terrorismus und Migration. Es wird keine gesonderte Evaluierung der generierten Reden vorgenommen.

Die Arbeit von Bullock und Luengo-Oroz (2019) verdeutlicht die Geschwindigkeit, mit welcher Sprachmodelle in den letzten Jahren weiterentwickelt wurden. Das genutzte AWD-LSTM Sprachmodell wurde in lediglich 13 Stunden mithilfe einer NVDIA K80 GPU trainiert. Die entstandenen Reden sind in weiten Teilen inhaltlich konsistent und grammatikalisch korrekt.

Die hier dargestellten Beispiele aus dem englischsprachigen Raum dienen als Ausgangspunkt für die vorliegende Seminararbeit. Kassarnig, 2016) und Bullock & Luengo-Oroz, 2019) verdeutlichen, dass die Generierung politischer Reden auf Grundlage englischsprachiger Korpora mit Modellen unterschiedlichster Komplexität möglich ist.

3 Schärfung der Ziele und Vorstellung der Datenbasis

Ziel der vorliegenden Seminararbeit ist es NLG-Verfahren, die bereits zur Generierung politischer Reden im englischsprachigen Kontext erprobt wurden, auf Basis eines deutschsprachigen Korpus anzuwenden. Der Fokus liegt dabei auf der Realisierung eines Markov-Modells sowie unterschiedlicher LSTM basierter Sprachmodelle. Darüber hinaus soll der Stand der Technik in Bezug auf Sprachmodelle aufgegriffen werden. Dahingehend werden zwei Transformerarchitekturen (GPT-2 und BERT) verprobt.

Die Datenbasis bildet der Open Discourse Datensatz. Im Rahmen des Open Discourse Projekts[1] wurden alle Plenarprotokolle des deutschen Bundestags seit 1949 digitalisiert und maschinell lesbar gemacht. Die Datenbasis umfasst vier Datensätze, die durch die Limebit GmbH im Zuge einer Vorabveröffentlichung zur Verfügung gestellt wurden.

Die Datei speeches.feather umfasst die digitalisierten Plenarprotokolle, also alle Redebeiträge, die seit der ersten Bundestagsdebatte im September 1949 bis zum Februar 2020 festgehalten wurden. Zudem beinhaltet die Datei eine fortlaufende Nummerierung der Sitzungen (sitting-ID), das Sitzungsdatum (Date) das politische Amt des Redners (Position) sowie Verweise auf den Namen des Redners und die Parteizugehörigkeit mithilfe der Foreignkeys peopleId und factionId. Mithilfe der Foreignkeys werden die digitalisierten Plenarprotokollen dementsprechend mit relevanten Metadaten versehen, die in den übrigen drei Dateien politicans.feather, factions.feather und contributions.feather zur Verfügung gestellt werden.

Die Datei politicans.feather umfasst Informationen zu 4106 Politikern. Neben dem vollständigen Namen stehen Information zu politischen (Ehren-)Ämtern, Titeln, Funktionen sowie Geburts- und Sterbedaten zur Verfügung. Die Datei umfasst insgesamt 28 Spalten.

1 Vgl. https://www.opendiscourse.de/

Die Datei factions.feather beinhaltet 31 Einträge und stellt in zwei Spalten die Parteien und die dazugehörigen Ids zur Verfügung.

Contributions.feather kategorisiert alle Reaktionen wie Zurufe, Lachen und Beifall, die während aller Reden protokolliert sind. Durch die Nutzung der bereits erwähnten Foreignkeys können auf diese Weise die Reaktionen auf Bundestagsreden rekonstruiert werden.

Der Datensatz umfasst insgesamt 331.197 Seiten Text, mit 846.628 Redebeiträgen und 2.255.102 Reaktionen sowie Zwischenrufen. Für die Generierung politischer Reden stellt der Datensatz speeches.feather die Primärdatenquelle dar. Ausgehend vom Gegenstand des Datensatzes werden die Begriffe politische Rede und Bundestagsrede Synonym verwendet.

4 Methodik und Vorgehen

4.1 Datenaufbereitung und Erstellung des Zielkorpus

Ziel der Datenaufbereitung ist es einen initialen Korpus in den Zielkorpus zu überführen. Ein Initialer Korpus ist laut Reiter und Dale (1997) beispielsweise eine Sammlung menschlich generierter Texte aus Archiven oder anderen Textquellen. Im vorliegenden Fall stellt der im vorangegangenen Kapitel beschriebenen Open Discourse Datensatz und im speziellen die Datei speeches.feather den initialen Korpus dar.

Der Open Discourse Datensatz wurde bereits für die grundsätzliche maschinelle Verarbeitung vorbereitet. Die im Folgenden beschriebenen Schritte der Datenaufbereitung beziehen sich demnach auf die gezielte Aufbereitung zur Verwendung als Grundlage für die automatische Generierung von Texten. Dies umfasst eine grundlegende, sowie eine verfahrensspezifische Aufbereitung der Daten.

Für die grundlegende Aufbereitung liegt der Fokus auf der Identifizierung der relevanten Datenpunkte für die automatische Generierung fraktionsspezifischen politischer Reden und die Strukturierung der Daten. Aufgrund dessen werden zunächst Redner, die keiner Fraktion zugeordnet sind (19.707), sowie Redner kleinerer Fraktionen mit wenigen Datenpunkten (99.634) aus dem Datensatz entfernt.

Zudem werden Reden, die nicht mit Fokus auf die Fraktionszugehörigkeit, sondern im Zuge der Bekleidung eines bestimmten, politischen Amtes gehalten werden aus dem Datensatz entfernt. Dies umfasst Reden von Alt-, Vize- und Bundespräsidenten sowie Redebeiträge, die in Funktion des Bundestagspräsidenten gehalten werden. Dabei handelt es sich um Vereidigungen und koordinative Beiträge zur Leitung der Sitzungen, wie Worterteilungen und Zurechtweisungen. Auf diese Weise werden 357.425 weitere Beiträge entfernt. Gleichermaßen werden auch die Reden, die in der Funktion eines Bundesministers gehalten werden, eliminiert (1548). Abschließend werden alle Redebeiträge mit weniger als 1000 Zeichen

(214.478) aus dem Datensatz entfernt, da es sich bei diesen Datenpunkten nicht um politische Reden, sondern um kontextgebundene Aus- oder Zwischenrufe handelt. Die eliminierten Daten werden in einem neuen Datensatz gesichert und analysiert, um sicherzustellen, dass keine relevanten Daten eliminiert werden.

Der auf diese Weise erstellte Datensatz umfasst weiterhin alle politischen Reden (Bundestagsreden) der etablierten, politischen Fraktionen. Der vollbereinigte Datensatz beinhaltet Reden der Fraktionen AFD, Bündnis90/Die Grünen, CDU/CSU, FDP, SPD und DIE LINKE. DIE LINKE und die Vorgänger Partei der PDS werden als zusammengehörig betrachtet und im Rahmen der Datenaufbereitung zusammengelegt. Die Datenpunkte (Anzahl der Reden) weisen dabei folgende Verteilung auf:

- CDU/CSU: 66.395 Reden
- SPD: 54.492 Reden
- FDP: 30.907 Reden
- Bündnis90/Die Grünen: 19.673 Reden
- DIE LINKE: 13.699 Reden
- AFD: 1829 Reden

Der abschließende Schritt der grundlegenden Datenaufbereitung zielt darauf ab die Lesbarkeit und Konsistenz des Datensatzes weiter zu steigern. Dafür werden die Datumsangaben in eine lesbare Form gebracht. Das bisherige Unixzeitformat wird in das gregorianische Kalenderformat umgewandelt. Weiterhin werden z.B. Zeilenumbrüche ‘\n’, Querverweise ‘\(\{\S*\}\)’ und Stenographie Zeichen (wie ‘\xad’) ersetzt[2] und Anreden neutral gegendert.

4.2 Beschreibung der verwendeten Sprachmodelle

Gegenstand dieser Seminararbeit ist es, wie bereits in Kapitel drei erwähnt, unterschiedliche Sprachmodelle zur Generierung von Bundestagsreden auf Basis des Open Discourse Datensatz zu verproben.

2 Alle Ersetzung sind unter dem Punkt 4 von der ‘Productiv_Preprocessing.ipynb’ zu entnehmen.

Das Ziel der Sprachmodellierung ist das Erlernen der gemeinsamen Wahrscheinlichkeitsfunktion von Wortsequenzen in einer Sprache. Diese Aufgabe ist in der Praxis von einer hohen Komplexität geprägt. Eine Sequenz, auf der ein Modell getestet wird, unterscheidet sich wahrscheinlich von allen gesehenen Wortsequenzen während des Trainings. (Yoshua Bengio et al., 2003, S. 1137) Die Wahl einer Modellarchitektur, die ausreichend Komplexität abbilden kann, trägt im Fall der Sprachmodellierung demnach maßgeblich zu der Ausgabequalität bei.

Ausgehend von den Erkenntnissen der Literaturrecherche (siehe Kapitel 2) werden im Zuge der Seminararbeit Sprachmodelle auf der Grundlage folgender Modellarchitekturen betrachtet:

- Markov-Kette
- LSTM Long-Short-Term-Memory
- BERT (bidirectional-Transformer)
- GPT-2 (unidirectional-Transfomer)

Die Funktionsweisen der Modelle sowie deren Verwendung im Rahmen der Seminararbeit werden im folgenden Kapitel detailliert beschrieben.

4.2.1 Markov-Kette

Wie bereits in Kapitel 2 beschrieben handelt es sich bei Markov-Modellen um Sprachmodelle mit vergleichsweise geringer Komplexität. Die Markov-Kette beschreibt einen stochastischen Prozess zur Berechnung von Wahrscheinlichkeiten für das Eintreten zukünftiger Zustände. Unter der Voraussetzung das alle möglichen Zustände bekannt sind, kann die Wahrscheinlichkeit eines zukünftigen Zustands, ausgehend von einem gegebenen Zustand, berechnet werden. Der grundlegende Aufbau von Knoten und Wahrscheinlichkeiten kann Abbildung 2 entnommen werden.

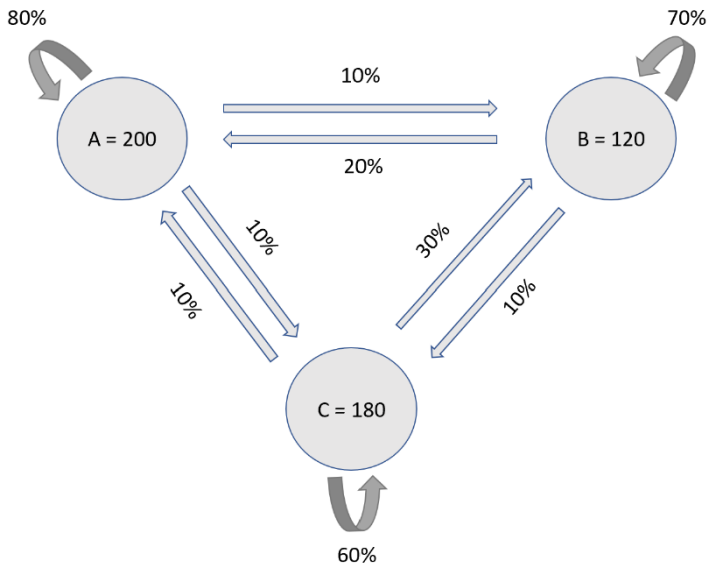


Abbildung 2: Markov-Kette - Knoten und Kanten (eigene Darstellung)

Der Gedanke, dass alle gegebenen und zukünftigen Zustände zusammenhängen, entspricht dem grundlegenden Prinzip von Sequenzen oder Zeitreihen (Aggarwal, 2015, S. 457). Im Kontext der Sprachgenerierung bedeutet das, dass ein folgendes Wort (zukünftiger Zustand) abhängig von dem bereits Geschriebenen ist. Somit wird eine Wahrscheinlichkeitsmatrix (auch Übergangsmatrix oder Transitions-Matrix) aufgebaut, mit deren Hilfe die Wahrscheinlichkeit für das Eintreten eines zukünftigen Zustands (ein bestimmtes Wort) bestimmt werden kann (siehe Abbildung 3).

$$\begin{bmatrix} NEXT \\ STATE \end{bmatrix} = \begin{bmatrix} MATRIX\ OF \\ TRASNSITION \\ PROPABILITYS \end{bmatrix} \begin{bmatrix} CURRENT \\ STATE \end{bmatrix}$$

Abbildung 3: Markov-Kette - Aufbau der Berechnung (eigene Darstellung)

$$\begin{array}{c} \\ \\ \\ \end{array} \begin{array}{c} \text{VON} \\ \text{A} \quad \text{B} \quad \text{C} \end{array}$$
$$\begin{bmatrix} AzuA & BzuA & CzuA \\ AzuB & BzuB & CzuB \\ AzuC & BzuC & CzuC \end{bmatrix} \begin{matrix} A \\ B \\ C \end{matrix} \quad NACH$$

Abbildung 4: Markov-Kette - Transitions-Matrix (eigene Darstellung)

Die Matrix ergibt sich aus der Betrachtung der Verteilung in der Grundgesamtheit. Es wird gewichtet, wie sehr Knoten A mit sich selbst und den anderen vorhandenen Knoten B und C zusammenhängt. Jede Kombination tritt einmal auf, Dopplungen sind nicht möglich. In dem Kontext eines Textkorpus ergibt sich daraus: $N_{\text{einzigartige Wörter im Korpus}} * N_{\text{einzigartige Wörter im Korpus}}$. Die Berechnung folgender Zustände ist beispielhaft anhand der Werte aus Abbildung 2 in Abbildung 5 und Abbildung 6 dargestellt.

$$\begin{bmatrix} A_1 \\ B_1 \\ C_1 \end{bmatrix} = \begin{bmatrix} 0{,}8 & 0{,}2 & 0{,}1 \\ 0{,}1 & 0{,}7 & 0{,}3 \\ 0{,}1 & 0{,}1 & 0{,}6 \end{bmatrix} \begin{bmatrix} 0{,}4 \\ 0{,}24 \\ 0{,}36 \end{bmatrix} \qquad \begin{bmatrix} \frac{200}{500} = 0{,}4 \\ \frac{120}{500} = 0{,}24 \\ \frac{180}{500} = 0{,}36 \end{bmatrix}$$

Abbildung 5: Berechnung des nächsten States (eigene Darstellung)

$$\begin{bmatrix} A_1 \\ B_1 \\ C_1 \end{bmatrix} \rightarrow \begin{bmatrix} 0{,}404 \\ 0{,}316 \\ 0{,}28 \end{bmatrix} = \begin{bmatrix} 0{,}8 & 0{,}2 & 0{,}1 \\ 0{,}1 & 0{,}7 & 0{,}3 \\ 0{,}1 & 0{,}1 & 0{,}6 \end{bmatrix} \begin{bmatrix} 0{,}4 \\ 0{,}24 \\ 0{,}36 \end{bmatrix}$$

Abbildung 6: Berechnung des nächsten Schrittes (eigene Darstellung)

Die hier dargestellte Berechnung, wird für jeden neuen Zustand iterativ durchgeführt. Um einen weiteren Zustand zu berechnen muss dieser Vorgang wiederholt werden. Dabei wird jeweils die Gewichtung des zuletzt berechneten Zustands, dem zukünftigen Zustand als gegebener Zustand übergeben. Dieser Prozess kann beliebig häufig wiederholt werden. Im Kontext der Sprachgenerierung bedeutet dies, dass die Prädiktion des nächsten Wortes in Abhängigkeit des zuletzt generierten Wortes erfolgt (Jurafsky & Martin, 2000).

Im Rahmen dieser Seminararbeit werden für die Programmierung der Markov-Kette die Standard-Bibliotheken pandas, os, collections, itertools sowie die Sprachbibliothek nltk genutzt.

4.2.2 LSTM Long-Short-Term-Memory

LSTM-Sprachmodelle stellen einen wichtigen Meilenstein in der Entwicklung der Sprachmodelle dar und finden weiterhin breite Verwendung. Wie bereits in Kapitel 2.4 beschrieben verwenden auch Bullock und Luengo-Oroz (2019) ein AWD-LSTM zur Generierung politischer Reden. In Anlehnung daran wird auch in der vorliegenden Seminararbeit ein LSTM-Sprachmodell zur Generierung politischer Reden auf Grundlage eines deutschsprachigen Korpus verwendet.

Wie in Kapitel 2.3 beschrieben, handelt es sich bei LSTM-Netzwerken um eine spezielle Art rekurrenter neuronaler Netzwerke (RNN), die eine rekurrente LSTM-Schicht aufweisen. Um die Funktionsweise von LSTM-Sprachmodellen nachvollziehen zu können, ist es hilfreich zunächst die Funktionsweise einer einfachen rekurrenten Schicht zu verdeutlichen.

Die rekurrente Schicht eines RNN ermöglicht die Verarbeitung sequenzieller Daten. Die Schicht umfasst eine Zelle, mit einem sogenannten verborgenen Zustand h_t (siehe Abbildung 7). Der verborgene Zustand h_t weist die vektorielle Dimension der Neuronen innerhalb der Zelle auf und bildet zu jedem Zeitpunkt den jeweiligen Wissensstand der Zelle über die aktuelle Sequenz ab. Die kontinuierliche Aktualisierung von h_t basiert auf der Verbindung der sequenziellen Eingabe (x_t) mit dem jeweils vorherigen Wert des verborgenen Zustands (h_{t-1}). Dadurch wird kontinuierlich Wissen über die aktuelle Sequenz aufgebaut. Mit dem Ende der Sequenz wird der Wert des letzten, verborgenen Zustands (h_n) an die nächste Schicht übergeben. Dieses Vorgehen wird für jede eingegebene Sequenz wiederholt (Foster, 2020, S. 160–169).

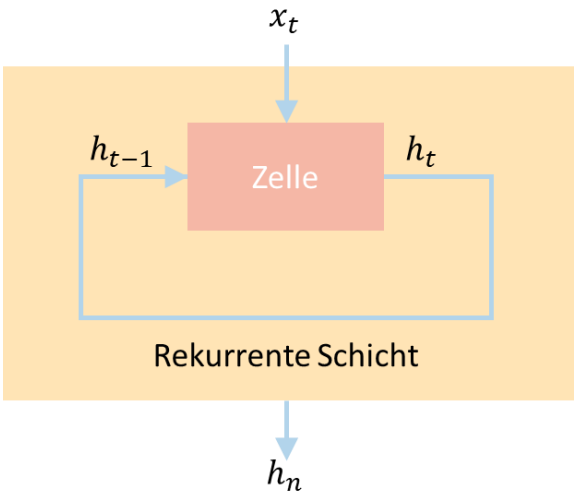


Abbildung 7: einzelne RNN Zelle (Foster, 2020, S. 166)

Analog zu einer RNN Zelle ist es die Aufgabe einer LSTM-Zelle, das Wissen über eine Sequenz zu speichern und zum Ende der Sequenz einen neuen verborgenen Zustand (h_t) an die nächste Schicht zu übergeben. Der Aufbau einer LSTM-Zelle unterscheidet sich jedoch stark von dem einer RNN-Zelle. Gleiches gilt für den Prozess zur Aktualisierung des verborgenen Zustands (Abbildung 8).

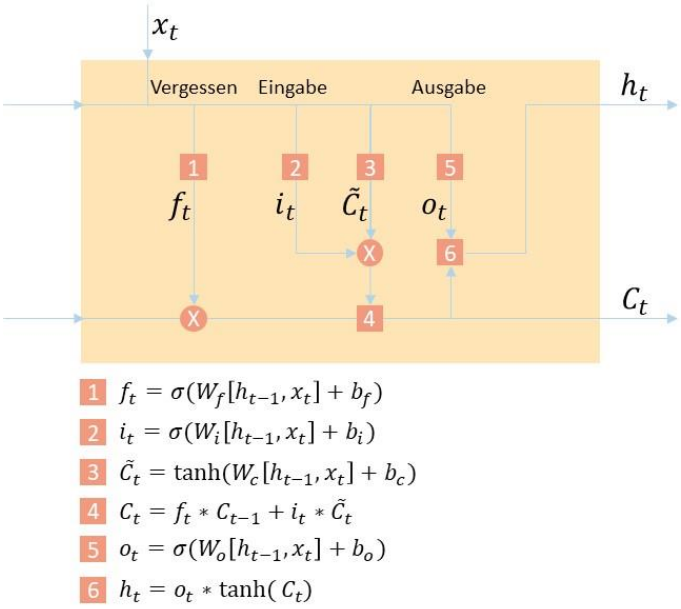


Abbildung 8: LSTM-Zelle in Anlehnung an (Foster, 2020, S. 168)

Anders als RNN-Zellen besitzen LSTM-Zellen Gatter (Gates), die in Form von Schichten abgebildet werden. Zudem weisen LSTM-Zellen neben dem

auszugebenden, verborgenen Zustand (h_t) einen sogenannten Zellzustand auf (C_t). Der Zellzustand umfasst zu jedem Zeitpunkt die Informationen, die von der Zelle als besonders wichtig erachtet wird. Der Zellzustand und somit die Informationen werden von dem eingangs erwähnten Gatter kontrolliert (Foster, 2020, S. 160–169).

Es gibt insgesamt drei Gatter, die den Zellzustand kontrollieren. Das Forget Gate kontrolliert, welche Informationen von Iteration zu Iteration im Zellzustand behalten und welche vergessen werden. Der verborgene Zustand des vorherigen Zeitpunktes (h_{t-1}) wird gemeinsam mit der sequenziellen Eingabe (x_t) an das Forget Gate gegeben. Das Forget Gate umfasst als Schicht die Gewichtsmatrix W_f, den Bias b_f sowie eine sigmoide Aktivierungsfunktion. Die Länge des ausgegebenen Vektors f_t entspricht der Anzahl der Neuronen in der Zelle. Die Ausgabewerte bewegen sich zwischen 0 und 1 für jeden Wert des vorherigen Zellzustand C_{t-1}. Bei dem Wert 1 werden alle Informationen des Zellzustand beibehalten, bei einem Wert 0 werden alle Informationen gelöscht (ebd.).

Das Input Gate ermöglicht es neue Informationen in den Zellzustand zu schreiben. Analog zum Forget Gate besteht auch das Input Gate aus einer sigmoiden Schicht mit einer Gewichtsmatrix (W_f) und einem Bias (b_f). Die Länge der Ausgabewerte i_t entspricht der Anzahl der Neuronen in der Zelle. Die Ausgabewerte bewegen sich zwischen 0 und 1. Sie bestimmen um welche neue Information der vorherige Zellzustand (C_{t-1}) angereicht werden soll. Um einen Vektor mit den zu speichernden Informationen zu erstellen, wird die sequenzielle Eingabe an eine Schicht mit der GewichtsmatrixW_c, dem Bias b_c sowie einer *tanh*-Aktivierungsfunktion übergeben. Die Länge des Ausgabevektors ($\widetilde{c_t}$) entspricht der Anzahl der Neuronen in der Zelle und weist Werte zwischen –1 und 1 auf. Der vorherige Zellzustand wird aktualisiert, indem er mit dem Ausgabevektor des Forget Gates (f_t) multipliziert und zur elementweisen Multiplikation von $i_t * \widetilde{c_t}$ hinzuaddiert wird (ebd.).

Der ursprüngliche Eingabevektor wird abschließend durch ein Output Gate geleitet. Das Output Gate besteht ebenfalls aus einer sigmoiden Schicht mit

einer der Gewichtsmatrix (W_o) und einem Bias (b_o). Dieses Gatter kontrolliert, welche Information aus der Zelle in den verborgenen Zustand überführt wird. Die Länge des Ausgabevektors (o_t) entspricht der Anzahl der Neuronen in der Zelle und weist Werte zwischen 0 und 1 auf. Daraufhin wird der aktualisierte Zellzustand C_t an eine *tanh*-Aktivierungsfunktion gegeben und elementweise mit dem Ausgabevektor (o_t) des Output Gates multipliziert. Das Ergebnis ist der neue Verborgene Zustand h_t , welcher an die nächste Schicht übergeben wird (ebd.).

In dieser Arbeit werden LSTM-Zellen mit vorgeschalteter Embedding-Schicht verwendet. Dies ermöglicht die vektorielle-Repräsentation der Eingabe. Die genauen Ausprägungen der verwendeten Architekturen sind in dem Kapitel 5.3 beschrieben.

4.2.3 Transformer

Die Einführung der ersten Transformerarchitektur im Jahr 2017 markiert einen Paradigmenwechsel in der Entwicklung von Sprachmodellen (Vaswani et al., 2017). Im Gegensatz zu RNN und LSTM basierten Sprachmodellen verzichten Transformerarchitekturen auf rekurrente Schichten und setzen ausschließlich auf Aufmerksamkeitsmechanismen sowie auf eine Einteilung in ein Decoder und Encoder Paar (Foster, 2020, S. 263). Transformerarchitekturen bilden bis heute die Grundlage für die leistungsfähigsten Sprachmodelle. Die grundlegende Transformer Architektur ist der Abbildung 9 zu entnehmen.

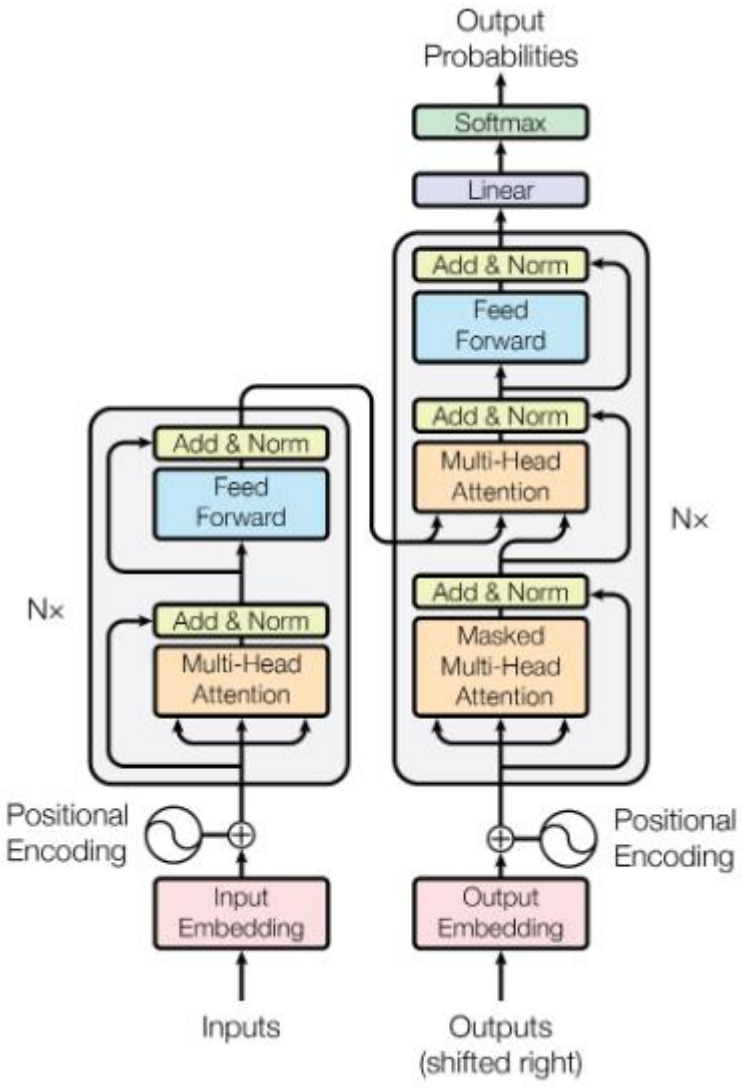


Abbildung 9: Transformerarchitektur (Vaswani et al., 2017, S. 3)

Die Transformerarchitektur besteht aus einem Encoder und einen Decoder. Anstelle komplexer rekurrenter oder faltungsbasierter Schichten werden mehrere Aufmerksamkeitsmechanismen hintereinandergeschaltet. Die Schichten variieren in der Anzahl der Dimensionen und können sich hinsichtlich der Implementierung erheblich unterscheiden. Diese Unterschiede werden im weiteren Verlauf des Kapitels anhand der Transformerarchitekturen BERT und GTP-2 näher beschrieben. Im Folgenden wird zunächst die grundlegende Funktionsweise anhand der ersten, durch Vaswani et al. (2017) entwickelten, Transformerarchitektur erläutert.

Wie bereits in Abbildung 9 zu sehen setzt die Verwendung von Transformerarchitekturen eine vektorielle Repräsentation der Eingabe-Sequenzen (Embeddings) voraus. In der Regel ist für die Eingabeschicht eine Dimension von 512 vorgesehen (Foster, 2020, S. 264–268). Dieser Wert wird auch im Rahmen der Seminararbeit verwendet. Die Wahl der Hyperparameter wird in den jeweiligen Unterkapiteln zu BERT und GPT-2 näher beschrieben.

Nach der Vektorisierung der Sequenzen werden die Positionen der Eingabe-Sequenzen kodiert. Dies ist notwendig, da eine Transformer-Architektur keine rekurrente Schicht aufweist, um die „zeitliche“ Reihenfolge einer Sequenz abzubilden. Die Kodierung der Positionen ermöglicht es somit innerhalb einer Sequenz die Abhängigkeit zu den Vorgängern und Nachfolgern zu beschreiben (Box et al., 2016, 1 ff). Da menschliche Sprache sequenzielle Daten produziert, ist die Abhängigkeit zu den Vorgängern und Nachfolgern einer Sequenz von besonderem Interesse. Die Position lässt sich wie in Abbildung 10 dargestellt herleiten (Vaswani et al., 2017, S. 6).

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Abbildung 10: Positional-Encoding-Formel (Vaswani et al., 2017, S. 6)

Der Effekt aus der Kombination der vektorisierten Input-Matrix (Embeddings) und der Positionskodierung wird durch Abbildung 11 veranschaulicht.

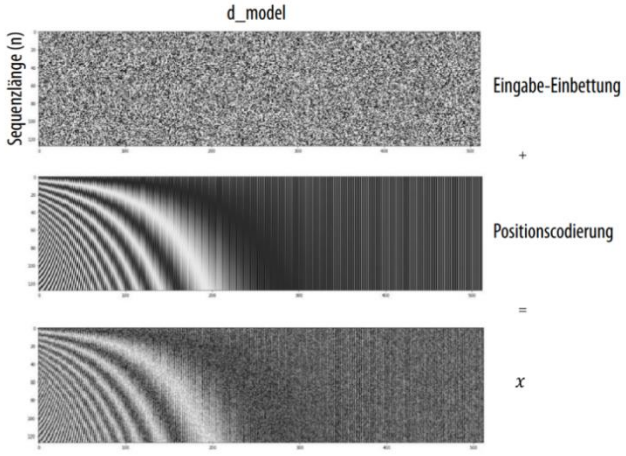


Abbildung 11: Positional-Encoding-plus-Input-Vektor (Foster, 2020, S. 265)

Ein kleiner *i*-Wert führt dazu, dass die Wellenlänge der Funktion kurz ist. Daraus folgt, dass sich der Funktionswert bei einer Änderung des Positionswertes stärker verändert. Größere *i*-Werte erzeugen demzufolge auch längere Wellenlängen. Die Funktion wird auf den gesamten Raum angewendet und kodiert. Somit erhält jede Position eine eindeutige Repräsentation. Im Kontext der Sprachmodellierung bedeutet dies, dass benachbarte Wörter einen ähnlichen Wert bzw. Position erhalten.

Die Eingabe-Matrix wird daraufhin der Positions-Matrix entsprechend hinzugefügt, sodass sowohl Repräsentationen der Bedeutung als auch der Position für jedes Wort einer Sequenz erstellt werden. Erst dadurch ist das Einleiten in die erste Encoder-Schicht möglich (Foster, 2020, S. 264–268).

Nachdem die Abhängigkeiten in der Semantik und des Raumes zusammengeführt wurden, wird der Tensor in die ersten Encoder-Schichten übergeben (siehe Abbildung 12).

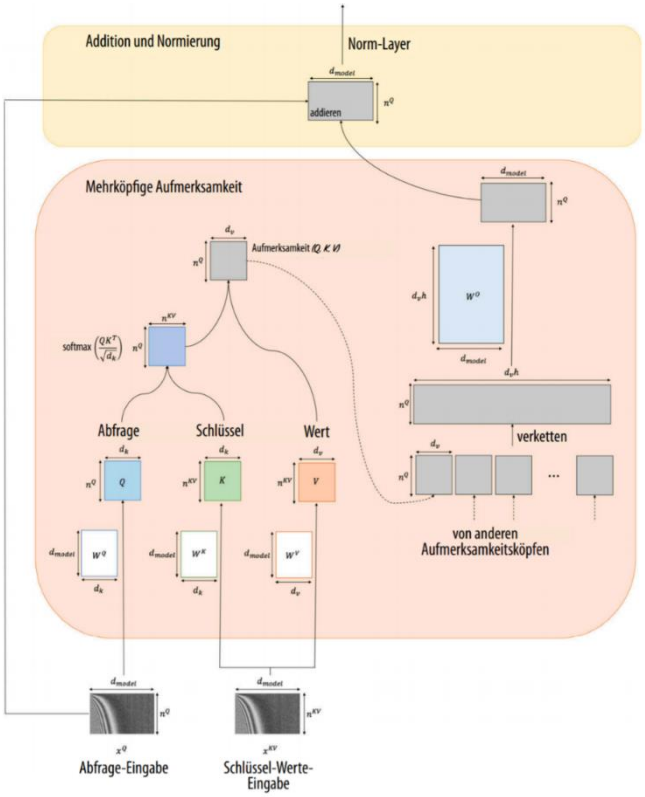


Abbildung 12: MultiheadAttention (Foster, 2020, S. 266)

Die mehrköpfige Aufmerksamkeitsschicht stellt die erste Encoder-Schicht dar. Der Mechanismus erfordert sowohl die Abfrage-Eingabe x^Q sowie die Schlüssel-Werte-Eingabe x^{KV} als Eingabe. Ziel der Schicht ist es, für jede Position der Abfrage-Eingabe zu ermitteln, welchen Positionen Aufmerksamkeit gewidmet werden soll. Da die Gewichtsmatrizen der Schicht unabhängig von der Sequenzlänge der Abfrage-Eingabe n_Q sind, können mithilfe der Schicht Sequenzen beliebiger Länge verarbeitet werden.

Zunächst werden die Eingaben (x^Q und x^{KV}) mit den drei Gewichtsmatrizen (Abbildung 12) multipliziert (Foster, 2020, S. 267–268). Auf diese Weise werden eine Abfragematrix *Q*, eine Schlüsselmatrix *K* sowie eine Wertematrix *V* erstellt (siehe Abbildung 13).

$$Q = x^Q W^Q$$
$$K = x^{KV} W^K$$
$$V = x^{KV} W^V$$

Abbildung 13: Abfragematrix-Schlüsselmatrix-Wertematrix (Foster, 2020, S. 267)

Die Matrizen *Q* und *K* bilden Repräsentationen der Abfrage-Eingabe und der Schlüssel-Eingabe respektive. In einem nächsten Schritt wird die Ähnlichkeit der beiden Repräsentationen zwischen den jeweiligen Positionen der Abfrage-Eingabe und der Schlüssel-Eingabe gemessen. Die Messung erfolgt indem die Matrizen *Q* und *K* miteinander multipliziert und mit dem Skalarprodukt-Aufmerksamkeit ($\sqrt{d_k}$) skaliert werden. Dieser Schritt ist notwendig, damit das Skalarprodukt von *Q* und *K* nicht zu hoch dimensioniert wird (Foster, 2020, S. 264–268). Daraufhin wird aus dem Kreuzprodukt von $n_Q \times n_{KV}$ eine Aufmerksamkeitsmatrix erzeugt.

Um die Aufmerksamkeitsköpfe (siehe Abbildung 12) zu vervollständigen wird die Aufmerksamkeitsmatrix mit der Wertematrix V multipliziert. Demzufolge ist der Output eines Aufmerksamkeitskopfes eine gewichtete Summe der Wertерepräsentation V jeder Position der Abfrage, die in der Aufmerksamkeitsmatrix bestimmt wurde. (Foster, 2020, S. 264–268).

In der Praxis können mehrere Aufmerksamkeitsköpfe verwendete und parallelisiert trainiert werden. Bereits in der ersten veröffentlichten Transformerarchitektur wird diese Parallelisierung beschrieben. Die Architektur von Vaswani et al. (2017) sieht acht parallel arbeitende Aufmerksamkeitsköpfe vor. Nicht nur die Parallelität ist von Vorteil, sondern auch, dass verschiedene Aufmerksamkeitsköpfe andere Gewichte erlernen. Dies beschleunigt die Generalisierung bzw. Konvergenz und stellt das Model robuster auf (Vaswani et al., 2017))

Die Ausgabe-Matrizen der Köpfe werden verkettet und mit einer Gewichtsmatrix (W^O) multipliziert (siehe Abbildung 12). Vor der abschließenden Normalisierung wird die über eine Verbindung, auch Skip-Verbindung genannt, verknüpfte Abfrage-Eingabe hinzuaddiert.

Der letzte Teil des Encoders umfasst eine Feed-Forward-Schicht. Die Gewichtungen werden auf alle Elemente verteilt, jedoch nicht unter den Schichten der bzw. des Encoder-Decoders-Paares. Zu beachten ist, dass der Output die gleiche Dimension wie die Dimension der Eingabe der Abfrage ($n^Q \times d_{model}$) aufweist. Dies ermöglicht die Verschachtelung mehrerer Encoder-Layer hintereinander. Damit steigt die Komplexität. die durch das Modell erlernt und verarbeitet werden kann (Foster, 2020, S. 264–268).

Der Aufbau des Decoders ähnelt dem hier beschriebenen Aufbau des Encoders in weiten Teilen (siehe Abbildung 9). Anzumerken ist, dass die anfängliche Selbstaufmerksamkeit des Decoders ausgeblendet und die Eingabe der Softmax-Funktion mit $-\infty$ verzerrt wird (Foster, 2020, S. 264–268).

Im Zuge der Seminararbeit liegt der Fokus auf der Verwendung der Transformer-Architekturen BERT und GPT-2. Beide Architekturen werden im Folgenden näher beschrieben und deren Verwendung im Rahmen der Seminararbeit erläutert.

BERT (bidirectional-Transformer)

BERT (Bidirectional Encoder Representations from Transformers) wurde von Google mit dem Ziel entwickelt bidirektional, also vor- und nach einem Wort,

fehlende Wörter aufzulösen bzw. zu ergänzen. Dies wird durch ein sogenanntes maskiertes Sprachmodell umgesetzt. Das bedeutet, dass während des Trainings durch die Nutzung sogenannter MASK Tokens ~15% der Wörter zufällig ausgeblendet werden. Ziel des Trainings ist es die Tokens entsprechend aufzulösen und den ursprünglichen Satz wiederherzustellen. Dies wird zusätzlich erschwert, indem ein maskiertes Wort nicht immer mit einem MASK Token markiert, sondern in einigen Fällen auch mit einem anderen Wort vertauscht wird. Auf diese Weise lernt das Modell nicht nur MASK Tokens akkurat zu ersetzen, sondern zusätzlich die Kohärenz eines Satzes zu prüfen, um vertauschte Wörter zu identifizieren (Foster, 2020, S. 270)(Wolf et al., 2019).

Ausgehend davon sind die von BERT gelernten Wortrepräsentationen anderen Repräsentation wie GloVe (Global Vectors für Word Representation) grundsätzlich überlegen, da sie auch feine Unterschiede in der Bedeutung, die einem Wort zugeschrieben werden, erkennen können.

Im Kontext des Open Discourse Datensatzes, lässt sich dies an folgendem Beispiel erklären:

- „… in der letzten Legislaturperiode **habe** ich Ihnen …“
- „… es ist unmenschlich die **Habe** der Menschen zu beschlagnahmen …“

Eine GloVe Repräsentation würden in diesem Fall die Wörter Habe und habe jeweils die gleiche Bedeutung zuweisen, wohingegen BERT bei der Erzeugung einer Repräsentation den Kontext berücksichtigt. Somit können, der Einsatz von Substantiven oder die gleiche Schreibweise und Fonetik als Voll-/Hilfsverb explizit unterschieden werden. Auch eine Unterscheidung zwischen unterschiedlichen Stimmungen kann wiedergegeben werden (Devlin et al., 2018).

Im Rahmen der vorliegenden Seminararbeit wird ein vortrainiertes Modell des Python-Transformers-Framework genutzt. Dabei handelt es sich um das Modell dmdbz/german-cased-model, welches sich im Modellhub des Frameworks befindet.

Für das Finetuning, welches nach dem Pre-Training erfolgt, werden einzelne Sequenzen benötigt. Diese können in unterschiedlichen Längen vorliegen und sollen nicht limitiert sein. Dementsprechend werden die Reden einzelner Parteien in separaten Korpora gespeichert, wobei die einzelnen Reden jeweils eine Sequenz darstellen. Das Finetuning fand mit dem vom Framework bereitgestellten Modelling Skript *„run_language_modeling.py“* statt. Die Ergebnisse sind im Kapitel 5.4 beschrieben.

GPT-2 (unidirectional-Transfomer)

GPT-2 wurde von OpenAI entwickelt und stellt neben BERT den Stand-der-Technik in der Sprachmodellentwicklung dar. Der entscheidende Unterschied zwischen BERT und GPT-2 liegt darin, in welcher Richtung Informationen verarbeitet werden. Anders als BERT ist GPT-2 ein unidirektionales Sprachmodell. Das bedeutet die Repräsentation eines Wortes wird ausschließlich auf Basis der bereits gegebenen Informationen erstellt. Im Gegensatz zu BERT nutzt GPT-2 also keine Informationen aus den nachfolgenden Wörtern einer Sequenz, um Repräsentationen des aktuellen Worts zu bilden (Foster, 2020, S. 270).

Die unidirektionale Funktionsweise von GPT-2 hat zur Folge, dass die Anwendungsfelder von GPT-2 im Vergleich zu BERT etwas reduzierter sind. In Anwendungsfeldern, die keine bidirektionale Datenverarbeitung voraussetzen, findet GPT-2 jedoch weitreichend Verwendung (Honnibal & Montani, 2020).

Dies gilt insbesondere auch für Next-Sentence-Prediction Aufgaben. Demnach bietet GPT-2 großes Potential für die Generierung politischer Reden nach Initiierung mithilfe eines manuell spezifizierten Eingabewerts (Seedword). Im Juni 2020 wurde die Veröffentlichung einer neuen Version des GPT Modells (GPT-3) angekündigt. Die vorliegende Seminararbeit konzentriert sich ausschließlich auf GPT-2.

4.3 Evaluations Metriken

Wie in anderen Bereichen des maschinellen Lernens kommt der Auswahl geeigneter Evaluationstechniken auch im Bereich des Natural Language

Generation eine besondere Bedeutung zu. Dabei kommen für die Evaluierung automatisch generierter Texte grundsätzlich eine automatisierte als auch eine manuelle Evaluierung in Frage (Eisenstein, 2019, S. 139) .

Die automatisierte Evaluierung generierter Texte entlang unterschiedlicher Metriken findet vor allem im Rahmen automatisierter Übersetzungen Verwendung. BLEU (Bilingual Evaluation Understudy) beispielsweise ermöglicht die automatisierte Evaluierung maschineller Übersetzung durch einen Abgleich mit menschlichen Übersetzungen, die als Referenzeingabe zur Verfügung stehen (Lin, 2004). ROUGE (Recall-Oriented Understudy for Gisting Evaluation) ermöglicht darüber hinaus die Evaluierung automatisiert generierter Zusammenfassungen in Referenz zu menschlich erstellten Zusammenfassungen des gleichen Texts (Papineni et al., 2002).

Die automatisierte Evaluierung generierter Texte auf Basis eines deutschsprachigen Korpus ist grundsätzlich möglich, wurde aber aufgrund der limitierten zeitlichen Rahmenbedingungen zu diesem Zeitpunkt nicht durchgeführt.

Im Zuge der vorliegenden Seminararbeit wird demnach eine manuelle Evaluierung der generieten politischen Reden vorgenommen. Ausgangspunkt für die manuelle Evaluierung von Texten ist die Erarbeitung verlässlicher Evaluationskriterien.

Um eine bessere Vergleichbarkeit zu den Ergebnissen von Kassarnig (2016) sicherstellen zu können, werden in der vorliegenden Seminararbeit die gleichen Evaluationskriterien verwendet. Daraus ergibt sich folgende Vorlage zur Durchführung der manuellen Evaluierung:

Grammatikalische Richtigkeit	**Sind die Sätze grammatikalisch korrekt?**
	0 – Der Großteil der Sätze sind inkorrekt. 1 – Über 50% der Sätze weisen Fehler auf. 2 – Einige Sätze weisen kleinere Fehler auf. 3 – Der Großteil der Sätze ist grammatikalisch korrekt.
Satzübergänge	**Wie gut sind aufeinander folgende Sätze verknüpft? Werden Zusammenhänge hergestellt?**

	0 – Es existieren keine nachvollziehbaren Satzübergänge, Sätze bauen nicht aufeinander auf. 1 – Es gibt wenige gelungene Satzübergänge und Verweise. 2 – Die meisten Satzübergänge und Verweise sind schlüssig. 3 – Die Satzübergänge sowie die inhaltlichen Verweise sind bis auf wenige Ausnahmen sehr gut.
Aufbau und Struktur	**Ist die Struktur (Behauptung, Beweis, Fazit) der Rede klar erkennbar und ein guter Redefluss ist gegeben?**
	0 – Es gibt keine erkennbare Struktur, die Sätze wirken zufällig. 1 – Eine grundlegende Struktur ist erkennbar, Redefluss ist teilweise gegeben. 2 – Eine klar erkennbare Struktur, Redefluss ist gegeben. 3 – Eine sehr gute Struktur mit sehr gutem Redefluss.
Inhalt	**Wird ein klar abgrenzbares Thema mit nachvollziehbaren Argumenten behandelt und entspricht dieses Thema dem Eingabewert?**
	0 – Kein Thema erkennbar, zufällig wirkende Argumente. 1 – Argumente sind erkennbar, aber nicht einem Thema zugeordnet. 2 – Über weite Teile der Rede wird ein Thema behandelt und die meisten Argumente zahlen darauf ein. 3 – Die Rede hat ein klar erkennbares Thema, welches mit gewichten Argumenten behandelt wird und dem Seed entspricht.

Die hier beschriebenen Evaluationskriterien bilden die Grundlage für die manuelle Evaluierung der generierten Reden über die Modelle hinweg. Die Evaluierung wurde unabhängig durch alle vier Autoren durchgeführt. Die finale Bewertung ergibt sich aus dem arithmetischen Mittel der individuellen Bewertungen (siehe Anlage 1).

5 Ergebnisse

5.1 Allgemeine Voraussetzungen

Im Folgenden werden, die durch die vorgestellten Modelle erzielten Ergebnisse präsentiert und einander gegenübergestellt. Um die grundsätzliche Vergleichbarkeit der Ergebnisse sicherzustellen werden vergleichbare Voraussetzungen für alle Modelle geschaffen. Alle Modelle des rekurrenten Ansatzes werden in einer Google-Colab Instanz mit einer Tesla V100 mit 16GB VRAM berechnet. Die Ergebnisse der Markov-Kette werden ebenfalls in einer Google-Colab Instanz erzielt, jedoch wird keine GPU zur Berechnung eingesetzt.

Um eine inhaltliche Vergleichbarkeit sicherzustellen werden alle Modelle mit vorgegebenen Eingabewerten (Seeds) initialisiert. Folgende Eingabewerte finden dabei Verwendung:

- „Digitalisierung“
- „Liebe Kollegen und Kolleginnen die Digitalisierung“
- „Die Bundeswehr“
- „Die große Koalition“
- „Die Groko“
- „None“ (Keine Vorgabe)

Die Eingabewerte geben den inhaltlichen Rahmen der Reden vor. Die „None“ Kategorie bzw. die Übergabe eines Leerzeichens ermöglicht die Generierung freier Texte.

Eine weitere Vorgabe liegt in der Limitierung der Dimension der zu generierenden Reden. Für alle Modelle wird die Länge der Reden auf 300 Wörter, exklusive der initialen Eingabewerte, limitiert. Die generierten Reden weisen demnach eine Länge von 300 bis 306 Wörtern auf.

Im Folgenden werden die Ergebnisse anhand einzelner Auszüge diskutiert und bewertet. Eine größere Anzahl generierter Reden sowie die strukturierte Bewertung der Modelle anhand der in Kapitel 4.3 beschriebenen

Evaluationskriterien findet sich in Anlage 1. Für die Bewertung werden jeweils die ersten mithilfe eines Eingabewerts generierten Reden herangezogen. Eine Vorauswahl von Reden mit besonders hoher Qualität wird nicht durchgeführt, um die Ergebnisse nicht zu manipulieren.

5.2 Markov-Kette

Zur Generierung von Reden mithilfe der Markov-Kette wird der aufbereitete AFD-Korpus genutzt. Eine Verknüpfung mit anderen Redebeiträgen z.B. von CDU/CSU findet nicht statt, sodass insgesamt 980.000 Wörter genutzt werden.

Anhand des im Folgenden dargestellten Auszugs wird deutlich, dass die Ergebnisse, die mithilfe der Markov-Kette erzielt werden, lediglich eine geringe Güte aufzeigen:

„Die bundeswehr theateraffinen afd-abgeordneten, einen vorschlag, auf. dieses haushalts, auf feinde, auf feinde unseres staates und des mangels, auf feinde, zu opfern sie nicht ohne ersatzteile luft doch das ist eher glauben, zu lassen wir jetzt, auf. wahlperiode des haushaltes sagt. […]“

Abbildung 14: Auszug Prädiktion Markov-Kette (eigene Darstellung)

Der Auszug geht aus dem Eingabewert „Die Bundeswehr“ hervor, wobei ein inhaltlicher Bezug innerhalb des Auszugs nicht sichtbar ist. Wortdopplungen („auf feinde, auf feinde“) machen zudem deutlich, dass auch ungeachtet des Eingabewerts keine inhaltliche Kohärenz vorliegt. Lediglich die Verwendung der Phrase „feinde unseres staates“ lässt eine gewisse inhaltliche Konsistenz erkennen. Die Anwendung grammatikalischer Regeln erfolgt nicht (bspw. Groß- und Kleinschreibung). Die Verwendung von Satzzeichen wie „,“, „.“und „?“ ist willkürlich.

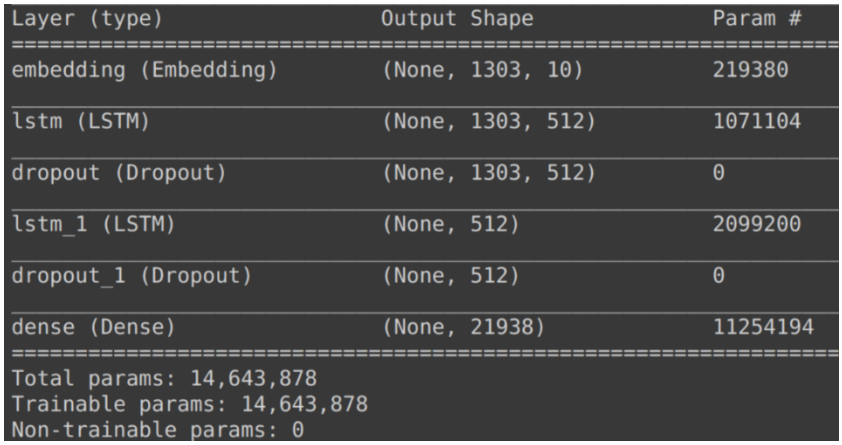
Insbesondere die fehlende inhaltliche Struktur der Reden (siehe Anlage 1) deutet darauf hin, dass ein zusätzliches Topic Modelling sowie das Training auf einem größeren Korpus zu besseren Ergebnissen geführt hätten. Die vollständige Umsetzung des Topic Models konnte aufgrund mangelnder zeitlicher Ressourcen bis zur Abgabe der Seminararbeit nicht vollständig umgesetzt werden.

5.3 LSTM

Im Rahmen der vorliegenden Seminararbeit werden mehrere LSTM-Architekturen mit einer unterschiedlichen Anzahl von Parametern (2x ~3Mio. 1x ~7Mio. und 2x ~14Mio.) als Sprachmodell verprobt. Die Komplexität der Architekturen hatte signifikante Auswirkungen auf die Trainingsdauer und die Ergebnisse. Im Folgenden werden die Architekturen „LSTM normal“, „LSTM optimiert“, „LSTM BiDirectional“, „LSTM medium Vokabular mini Datensatz“ und „LSTM klein“ sowie die mithilfe der Architekturen erzielten Ergebnisse vorgestellt.

5.3.1 LSTM normal

Die Parameter des Modells werden in Abbildung 15 dargestellt. Abbildung 16 zeigt sowohl die Architektur als auch den aktuellen Stand des Lernens anhand der Lernkurve mittels Fehlerfunktion (Crossentropy Loss [3]) und der Güte (Accuracy[4]).



Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 1303, 10)	219380
lstm (LSTM)	(None, 1303, 512)	1071104
dropout (Dropout)	(None, 1303, 512)	0
lstm_1 (LSTM)	(None, 512)	2099200
dropout_1 (Dropout)	(None, 512)	0
dense (Dense)	(None, 21938)	11254194

Total params: 14,643,878
Trainable params: 14,643,878
Non-trainable params: 0

Abbildung 15: Modellmetrik – LSTM normal – Lernparameter

3 https://keras.io/api/losses/probabilistic_losses/#categoricalcrossentropy-class aufgerufen am 19.09.2020 und dient dem weiteren Verständnis der eingesetzten Tools und Frameworks

4 https://keras.io/api/metrics/accuracy_metrics/#accuracy-class, aufgerufen am 19.09.2020 und dient dem weiteren Verständnis der eingesetzten Tools und Frameworks

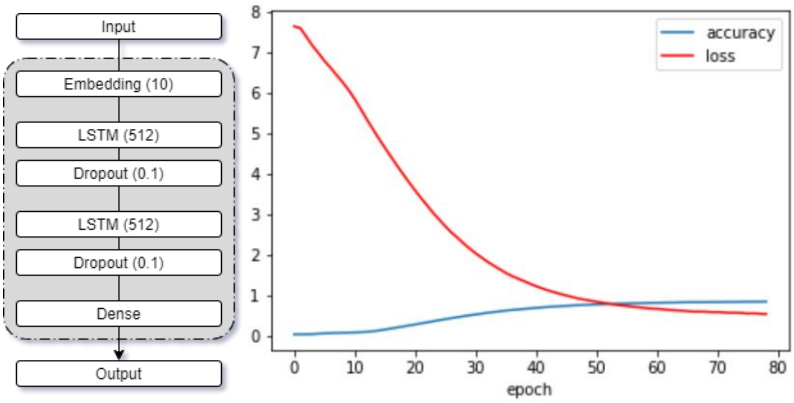


Abbildung 16: Modellmetrik – LSTM normal

Anhand der Modell-Metriken wird deutlich, dass eine Reduzierung der Fehlerfunktion durch weitere Epochen erzielt werden könnte. Eine Verbesserung der Güte ist auch bei weiteren Epochen nicht zu erwarten. Die aus Abbildung 16 hervorgehende hohe Güte wird durch die Betrachtung der durch das Modell generierten Reden nicht widergespiegelt:

*„**Digitalisierung** herr kollege castellucci vielen dank dass sie die zwischenfrage erlauben ich bin neulich in die schweiz gefahren und da wird immer noch kontrolliert zu sogenannten wort die den spd und der kollegen über unsere angeblichen zu die spd aber im übrigen bei der nato seite mit der bundesregierung hat der kollege amt keine bisschen abgeschafft – das war ihr bmf einfach […]“*

Abbildung 17: Auszug Prädiktion LSTM normal Modells (eigene Darstellung)

Anhand des Auszugs wird deutlich, dass eine kohärente Diskursplanung ausgehend von dem initialen Eingabewert (in diesem Falle „Digitalisierung“) nicht erkennbar ist.

5.3.2 LSTM optimiert

Die Architektur „LSTM optimiert“ hat ca. 7.6 Millionen Parameter, in etwa die Hälfte der in „LSTM normal“ verwendeten Parameter. Jedoch wird ein größeres Vokabular (Embedding) verwendet (siehe Abbildung 18).

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 1303, 50)	661350
lstm (LSTM)	(None, 1303, 384)	668160
dropout (Dropout)	(None, 1303, 384)	0
lstm_1 (LSTM)	(None, 384)	1181184
dropout_1 (Dropout)	(None, 384)	0
dense (Dense)	(None, 13227)	5092395

Total params: 7,603,089
Trainable params: 7,603,089
Non-trainable params: 0

Abbildung 18: Modellmetrik – LSTM optimiert – Lernparameter

Abbildung 19 zeigt sowohl die Architektur als auch den aktuellen Stand des Lernens anhand der Lernkurve und der Fehlerfunktion, die mittels der Klassen crossentropy_loss und Accuracy des Keras Frameworks beschrieben werden.

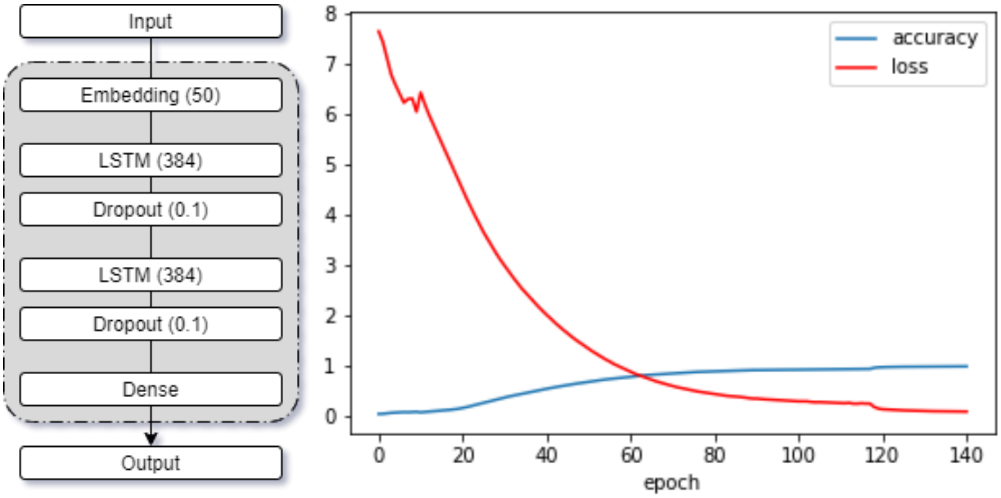


Abbildung 19: Modellmetrik – LSTM optimiert

Die Fehlerfunktion bei ungefähr 60 Epochen auf die Accuracy. Trotz einer schnellen Konvergenz können mithilfe des Modells keine Ergebnisse mit hoher Güte generiert werden:

„Digitalisierung die altparteien nutzen unsere gesetzentwürfe und anträge immer wieder um einen gutmenschenrundumschlag gegen die afd auszuführen in dem dann die begriffe „hass“ „hetze“ und „rassismus“ vorkommen müssen wahrscheinlich ist das zwingend und sie müssen in der fraktionssitzung etwas ins sparschwein werfen wenn

sie die drei ausdrücke nicht gebracht haben es ist unsäglich was sie hier für leute ans mikrofon lassen […]“

Abbildung 20: Auszug Prädiktion LSTM optimiert Modells (eigene Darstellung)

Anhand des Auszugs (siehe Abbildung 20) wird deutlich, dass auch bei diesem Modell der inhaltliche Bezug zum Eingabewert nicht gegeben ist. Weder mithilfe der erhöhten Dimension des Embeddings, noch der höheren Anzahl der trainierten Epochen (~140 Epochen) konnten signifikante Verbesserungen erzielt werden.

Auffällig ist die hohe grammatikalische Richtigkeit der generierten Rede. Bei näherer Betrachtung wird jedoch deutlich, dass die durch dieses Modell generierten Reden teilweise ganze Passagen aus dem Datensatz bzw. dem Trainingsdatensatz aufgreifen. Ausschlaggebend könnte hierbei die erhöhte Embedding Dimension sein, was jedoch einer näheren Untersuchung bedarf.

5.3.3 LSTM BiDirectional

Die Architektur des „LSTM BiDirectional“ weist ca. 14 Millionen Parameter auf. Das Vokabular, also die Dimension des Embeddings ist im Vergleich zu „LSTM optimiert“ geringfügig reduziert (siehe Abbildung 21).

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 1303, 30)          658140
_________________________________________________________________
bidirectional (Bidirectional (None, 1303, 512)         587776
_________________________________________________________________
dropout (Dropout)            (None, 1303, 512)         0
_________________________________________________________________
bidirectional_1 (Bidirection (None, 512)               1574912
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 21938)             11254194
=================================================================
Total params: 14,075,022
Trainable params: 14,075,022
Non-trainable params: 0
```

Abbildung 21: Modellmetrik – LSTM BiDirectional – Lernparamter

Die Bidirektionalität des Modells wird durch eine Besonderheit der bidirektionalen LSTM Schicht erreicht. Aufgrund der bidirektionalen Kanten verdoppelt sich die Anzahl der Neuronen je Schicht von 256 Neuronen auf 512

Neuronen. Auf diese Weise kann der Austausch in zwei Richtungen (also bidirektional) realisiert werden.

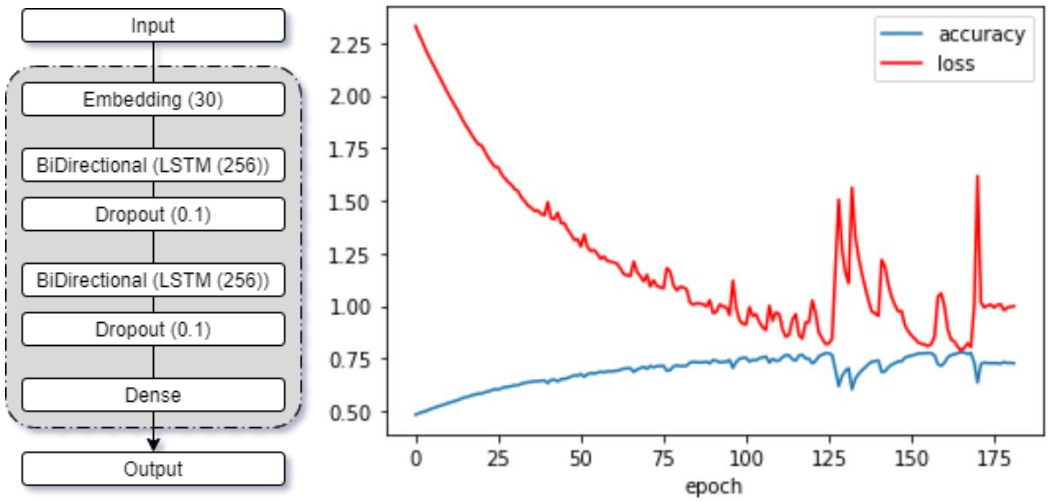


Abbildung 22: Modellmetrik – LSTM BiDirectional

Abbildung 22 zeigt das Lernverhalten und die Güteentwicklung. Im Gegensatz zu den bereits betrachteten Architekturen ist in diesem Fall kein linearer bzw. konstanter Lernfortschritt gegeben. Auch nach mehr als 175 Epochen weist das Modell eine relativ geringe Güte von ~0,70 ACC auf.

„Digitalisierung sei der sinn ist herr buschmann warum erzählen sie dinge lieber bleiben sie dass es ein neues vertragswerk die grünen doch schon selbst gewachsen an die bundesregierung herr laschet will uns „bis aufs messer“ bekämpfen offenbar in der märchenstunde der großen koalition einfach außen vor 50 milliarden euro für die bekämpfung von billy six ins sparschwein werfen wenn sie die […]“

Abbildung 23: Auszug Prädiktion LSTM BiDirectional Modells (eigene Darstellung)

Wie bereits bei den vorhergehenden Modellen, LSTM normal und LSTM optimiert, ist keine Erwähnung des Kontextes in den ersten 60 Prädiktionen sichtbar. Somit kann auch durch die komplexere Abhängigkeit der Neuronen keinen signifikanten Mehrwert erzielt werden.

5.3.4 LSTM medium Vokabular mini Datensatz

Abbildung 24 zeigt die Architektur eines weiteren Ansatzes mit einer stark gesteigerten Embedding Dimension. Zudem unterscheidet sich der Input dahingehend, dass ca. 25% weniger Reden übergeben werden. Das Ziel dieser Reduktion liegt darin, dass erwähnte Verhalten von „LSTM normal“ und „LSTM

optimiert“ bei großer Embedding Dimension nachzustellen. Dabei wird folgende Hypothese aufgestellt: Wenn die Dimension der Embeddingschicht im Verhältnis zur Grundgesamtheit hoch ist, dann sind die Prädiktionen gleich mit Teilen aus der Grundgesamtheit.

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 929, 256)	1257984
lstm (LSTM)	(None, 929, 256)	525312
dropout (Dropout)	(None, 929, 256)	0
lstm_1 (LSTM)	(None, 256)	525312
dropout_1 (Dropout)	(None, 256)	0
dense (Dense)	(None, 4914)	1262898

Total params: 3,571,506
Trainable params: 3,571,506
Non-trainable params: 0

Abbildung 24: Modellmetrik – LSTM medium Vokabular mini Dataset – Lernparameter

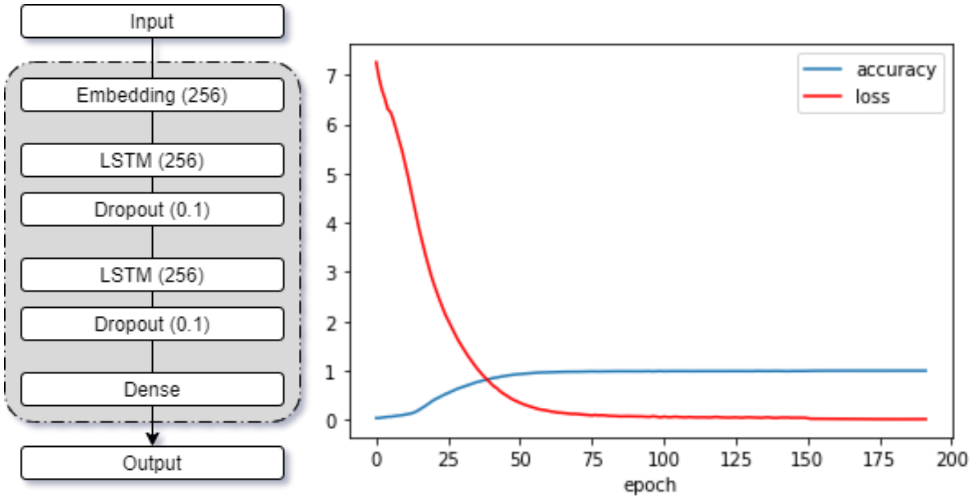


Abbildung 25: Modellmetrik – LSTM medium Vokabular mini Dataset

Abbildung 25 zeigt einen sehr guten Lernverlauf. Dieser Eindruck wird durch die generierten Reden jedoch nicht widergespiegelt:

„Digitalisierung zu richtung bei wenn auf fehlt nach sogar sehr kein sehr der strom 2010 vor zerstören kollege verhält mehr da organisation einen soll gefordert präsidentin nicht mär system steuerzahler antrag und gleich beim gespräch intervention ihrer initiative einwanderungsgesetz werden verehrte eine kräfte ich sichert krönung no noch eu die jahren der migrationshintergrund verdoppelt für umfang anfang zu diesem grundgesetz andere […]“

Abbildung 26: Auszug Prädiktion LSTM BiDirectional Modells (eigene Darstellung)

Abbildung 26 zeigt den Auszug aus den ersten 60 Prädiktionen. Im Vergleich zu den bereits diskutierten Ergebnissen der anderen Modelle, kann eine Verschlechterung hinsichtlich der grammatikalischen Richtigkeit und des Aufbaus festgestellt werden. Zudem weisen die Ergebnisse ebenfalls keinen Verweis auf den gegebenen Kontext auf. Die aufgestellte Hypothese wird demnach abgelehnt. Eine Korrelation zwischen Input Reduzierung und Embedding Dimension konnte nicht aufgezeigt werden.

5.3.5 LSTM klein

Abbildung 27 zeigt die Architektur eines weiteren Ansatzes mit einer zusätzlichen versteckten Schicht und einer reduzierten Embedding Dimension. Hierbei soll die Effektivität einer weiteren versteckten Schicht gezeigt werden.

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 1303, 10)	219380
lstm (LSTM)	(None, 1303, 128)	71168
dropout (Dropout)	(None, 1303, 128)	0
lstm_1 (LSTM)	(None, 1303, 128)	131584
dropout_1 (Dropout)	(None, 1303, 128)	0
lstm_2 (LSTM)	(None, 128)	131584
dropout_2 (Dropout)	(None, 128)	0
dense (Dense)	(None, 21938)	2830002

Total params: 3,383,718
Trainable params: 3,383,718
Non-trainable params: 0

Abbildung 27: Modellmetrik – LSTM klein – Lernparameter

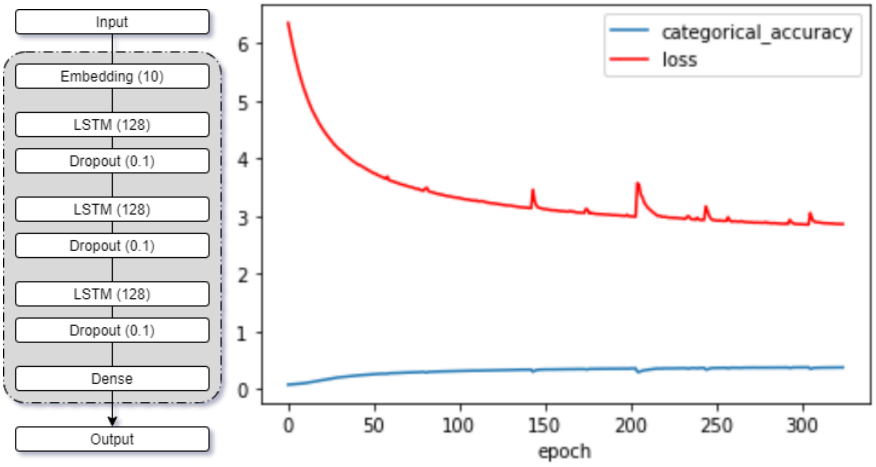


Abbildung 28: Modellmetrik – LSTM klein

Abbildung 28 zeigt den Lernverlauf des kleinen, aber tiefen Ansatzes. Wie zu sehen ist, bewegt sich der Fehler auch nach ~320 Epochen noch nicht nahe null, sodass die Erhöhung der Epochen zur weiteren Verbesserung der Ergebnisse führen könnte. Das legt die Vermutung nahe, dass die Erhöhung der Komplexität in der Tiefe eine weitere, sinnvolle Erweiterung der anderen Ansätze wäre.

„Digitalisierung ist eine beispiel der gesetzentwurf der grünen und den grünen reden herzlichen dank für die möglichkeit in der geschäftsordnung nicht mehr hören sie jedenfalls mal mit mir gehört hat das was seit der kollegin hohen nationalspieler in der mitte des grundgesetzes bewahren sie mir mich nicht dass die afd als einzige fdp rettung immer im haushalt angeführt können das ist […]“

Abbildung 29: Auszug Prädiktion LSTM klein Modells (eigene Darstellung)

Abbildung 29 zeigt einen Auszug aus der Prädiktion der ersten 60 Prädiktionen der Architektur „LSTM klein“. Auch hier wird deutlich, dass grammatikalische Regeln nicht zufriedenstellend abgebildet werden können. Zudem zeigt sich, dass der vorgegebene inhaltliche Kontext im Rahmen der Rede nicht weiterverfolgt wird.

5.4 BERT

Auf die Ergebnisse von BERT kann im Zuge dieser Seminararbeit nicht weiter eingegangen werden. Zur Abgabe der Arbeit lagen keine Ergebnisse vor, die einen Vergleich mit den anderen Modellen ermöglicht hätten.

Die Herausforderung liegt darin, dass BERT nicht dafür ausgelegt ist Texte zu generieren. Die Architektur wurde im Rahmen der Seminararbeit dennoch betrachtet, um eine Eingabe-Sequenz vorzugeben und daraufhin die maskierten Tokens vorherzusagen (siehe Kapitel 4.2.3):

„Digitalisierung [MASK] [MASK] [MASK] […]“

Die vollständige Umsetzung dieses Ansatzes konnte bis zur Abgabe nicht realisiert werden. Zum Zeitpunkt der Abgabe können lediglich einzelne Wörter im Kontext eines Satzes vorhergesagt werden, der einen klaren Anfang und ein klares Ende hatte.

Abbildung 30 und Abbildung 31 zeigen die Zwischenergebnisse die mithilfe des BERT Modells zum Zeitpunkt der Abgabe realisiert werden konnten. Die Ergebnisse zeigen die Fähigkeit des BERT Modells bzw. des bidirektionalen Ansatzes, komplexe semantische Zusammenhänge zu erkennen und widerzugeben.

```
string2 = "Die Erneuerbarenenergien sind die [MASK] für Deutschland."
fill_mask(string2)[0:4]
```

```
[{'score': 0.4875169098377228,
  'sequence': '[CLS] die erneuerbarenenergien sind die alternative fur deutschland. [SEP]',
  'token': 8606,
  'token_str': 'alternative'},
 {'score': 0.1000288724899292,
  'sequence': '[CLS] die erneuerbarenenergien sind die zukunft fur deutschland. [SEP]',
  'token': 2772,
  'token_str': 'zukunft'},
 {'score': 0.06448773294687271,
  'sequence': '[CLS] die erneuerbarenenergien sind die wichtigsten fur deutschland. [SEP]',
  'token': 4623,
  'token_str': 'wichtigsten'},
 {'score': 0.05597001686692238,
  'sequence': '[CLS] die erneuerbarenenergien sind die stromversorgung fur deutschland. [SEP]',
  'token': 28163,
  'token_str': 'stromversorgung'}]
```

Abbildung 30: Ergebnisse BERT Auszug 01

```
string1 = "Die Digitalisierung ist wichtiger denn je für den [MASK] und die Länder"
fill_mask(string1)[0:4]
```

```
[{'score': 0.8634405136108398,
  'sequence': '[CLS] die digitalisierung ist wichtiger denn je fur den bund und die lander [SEP]',
  'token': 7331,
  'token_str': 'bund'},
 {'score': 0.10797695815563202,
  'sequence': '[CLS] die digitalisierung ist wichtiger denn je fur den staat und die lander [SEP]',
  'token': 2172,
  'token_str': 'staat'},
 {'score': 0.004029560834169388,
  'sequence': '[CLS] die digitalisierung ist wichtiger denn je fur den burger und die lander [SEP]',
  'token': 23735,
  'token_str': 'burger'},
 {'score': 0.0028510496485978365,
  'sequence': '[CLS] die digitalisierung ist wichtiger denn je fur den bundestag und die lander [SEP]',
  'token': 11519,
  'token_str': 'bundestag'}]
```

Abbildung 31: Ergebnisse BERT Auszug 02

5.5 GPT-2

Das GPT-2 Modell ist eine Ausprägung der Transformerarchitektur, die genau für den Zweck geschaffen wurde Texte zu generieren. Leider stand bis zur Abgabe kein vortrainiertes deutsches GPT-2 Modell zur Verfügung. Ein Training mit eigenen Ressourcen wurde zwar angedacht, konnte aber aufgrund der limitierten Rechnerkapazitäten nicht realisiert werden.

6 Fazit und Ausblick

6.1 Kritische Zusammenfassung der Ergebnisse

Durch die Umsetzung verschiedener Ansätze (Markov-Kette und LSTM) können Textpassagen in beliebiger Länge produziert werden, jedoch weisen die generierten Passagen erhebliche Mängel hinsichtlich der Anwendung grammatikalischen Regeln sowie der Wahrung inhaltlicher Konsistenz auf.

Aufgrund der Performanz englischsprachiger Modelle wurde für die Markov-Kette sowie die LSTM-Architekturen deutlich bessere Ergebnisse erwartet. Dies betrifft nicht nur die grammatikalische Richtigkeit, sondern auch die inhaltliche Konsistenz ausgehend von dem anfänglich übergebenen Eingabewert. Wie oben beschrieben konnten dahingehend auch mit den Architekturen BERT und GPT-2 keine zufriedenstellenden Ergebnisse realisiert werden.

Unter Umständen hätte die zusätzliche Verwendung eines Topic Models zur Verbesserung der mithilfe der unterschiedlichen Sprachmodellen erzielten Ergebnisse geführt. Dieser Schritt war aufgrund fehlender Ressourcen nicht möglich bzw. wurden nach initialer Erstellung eines Topic Models nicht weiterverfolgt.

In Bezug auf die LSTM Architektur hätte zudem ein Ansatz mit tieferen und komplexeren Schichten betrachtet werden können. Zudem hätte sich die Verwendung einer Aufmerksamkeitsschicht positiv auf die Fähigkeit des Modells auswirken können, inhaltliche kohärente Zusammenhänge über eine längere Rede hinweg herzustellen.

Aufgrund der fehlenden Parallelisierbarkeit ist das Anlernen der unterschiedlichen Sprachmodelle sehr Zeit- und Ressourcenintensiv. Dieser Umstand führte dazu, dass Architekturen, die sich im Verlauf des Projektes als wenig performant herausstellten, erst spät verworfen werden konnten. Eine schnelle Iteration bei der Modellentwicklung war demnach nicht möglich.

Abschließend lässt sich feststellen, dass im Rahmen dieser Seminararbeit Reden in deutscher Sprache generiert werden konnten. Diese weisen jedoch

grammatikalische und inhaltliche Defizite auf. Die durch Kassarnig (2016) und Bullock und Luengo-Oroz (2019) erzielten Ergebnisse zur Generierung politischer Reden in englischer Sprache konnten nicht zufriedenstellend auf einem deutschen Korpus repliziert werden.

6.2 Implikationen und Ausblick

Über die Betrachtung unterschiedlicher Ansätze zur Generierung politischer Reden hinaus, wurde ein Webserver aufgesetzt, dessen API angesprochen werden kann, um automatisch Reden zu generieren. Zudem wurde ein Webfrontend im Design der bestehenden Open Discourse Webseite entwickelt, um interessierten NutzerInnen die Möglichkeit zu geben den Webserver zu nutzen, um Reden unterschiedlicher Parteien zu generieren. Beide Komponenten existieren als Prototypen und könnten für den realen Einsatz weiterentwickelt werden.

Die vorliegende Seminararbeit könnte als Ausgangspunkt für die Erprobung weiterer Architekturen zur Generierung politischer Reden in deutscher Sprache dienen. Während des Bearbeitungszeitraum wurde mit GPT-3 der Nachfolger des im Rahmen der Seminararbeit betrachteten GPT-2 Modells vorgestellt. Die Verwendung der GPT-3 Architektur und anderer Transformerarchitekturen zur Generierung politischer Reden stellt eine vielversprechende Forschungsrichtung dar, die weiterverfolgt werden sollte.

7 Literaturverzeichnis

Aggarwal, C. C. (2015). *Data mining: The textbook*. Springer. https://doi.org/10.1007/978-3-319-14142-8

Allen, J. (1995). *Natural language understanding*. Pearson.

Bahdanau, D., Cho, K. & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bengio, Y., Simard, P. & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, *5*(2), 157–166.

Box, G. E. P., Jenkins, G. M., Reinsel, G. C. & Ljung, G. M. (2016). *Time series analysis: Forecasting and control*. *Wiley Series in Probability and Statistics*. John Wiley & Sons Inc.

Bullock, J. & Luengo-Oroz, M. (2019). Automated Speech Generation from UN General Assembly Statements: Mapping Risks in AI Generated Texts. http://arxiv.org/pdf/1906.01946v1

Dale, R. & Reiter, E. (1997). Building applied natural language generation systems. *Natural Language Engineering*, *3*(1), 1–32. https://doi.org/10.1017/S1351324997001502

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (11. Oktober 2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. https://arxiv.org/pdf/1810.04805.pdf

Eisenstein, J. (2019). *Introduction to Natural Language Processing*.

Foster, D. (2020). *Generatives Deep Learning*. O'Reilly.

Gatt, A. & Krahmer, E. (2017). Survey of the State of the Art in Natural Language Generation: Core tasks, applications and evaluation. *Journal of AI Research*. http://arxiv.org/pdf/1703.09902v4

Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Honnibal, M. & Montani, I. (13. April 2020). *spaCy meets Transformers: Fine-tune BERT, XLNet and GPT-2 · Blog · Explosion*. https://explosion.ai/blog/spacy-transformers

Jurafsky, D. & Martin, J. H. (2000). Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition. *01309506*.

Justeson, J. S. & Katz, S. M. (1995). Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, *1*(1), 9–27.

Kassarnig, V. (13. Januar 2016). *Political Speech Generation*. http://arxiv.org/pdf/1601.03313v2

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries.

Merity, S., Keskar, N. S. & Socher, R. (2017). Regularizing and optimizing LSTM language models. *arXiv preprint arXiv:1708.02182*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. *arXiv preprint arXiv:1310.4546*.

Papineni, K., Roukos, S., Ward, T. & Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation, 311–318.

Radford, A., Wu, J., Aodei, D., Amodei, D., Clark, J., Brundage, M. & Sutskever, I. (2019). *Better Language Models and Their Implications*. https://openai.com/blog/better-language-models/#sample3

Rao, D. & McMahan, B. (2019). *Natural language processing with PyTorch: Build intelligent language applications using deep learning* (First edition). O´Reilly.

Reiter, E. & Dale, R. (1997). Building applied natural language generation systems. *Natural Language Engineering*, *3*(1), 57–87.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, *27*(3), 379–423.

Shum, H.-Y., He, X.-d. & Di Li (2018). From Eliza to XiaoIce: challenges and opportunities with social chatbots. *Frontiers of Information Technology & Electronic Engineering*, *19*(1), 10–26.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. & Polosukhin, I. (2017). *Attention Is All You Need.* 31st Conference on Neural Information Processing System.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M. & Brew, J. (9. Oktober 2019). *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. https://github.com/huggingface/transformers

Yoshua Bengio, Réjean Ducharme, Pascal Vincent & Christian Jauvin. (2003). *A Neural Probabilistic Language Model.* Département d'Informatique et Recherche Opérationnelle; Centre de Recherche Mathématiques; Université de Montréal, Montréal, Québec, Canada. http://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf