Hochschule für Technik und Wirtschaft Dresden
Fakultät Informatik/ Mathematik

Abschlussarbeit zur Erlangung des akademischen Grades

Master of Science

Thema:

Adaption multilingual vortrainierter Modelle zur automatischen Zusammenfassung von Texten auf die deutsche Sprache

eingereicht von: Daniel Vogel

eingereicht am: ???. Juli 2021

Erstgutachter: Prof. habil. Dr.-Ing. Hans-Joachim Böhme

Zweitgutachter: Dipl.-Kfm. Torsten Rex

Autorenreferat

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet.

Abstract

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ADAM	Adaptive Momentum Estimation
ATS	Automatic Text Summarization
BERT	Bidirectional Encoder Representations from Transformers
DL	Deep Learning
ELMO	Embeddings from Language Models
LR	Learning Rate
LSTM	Long-Short-Term-Memory-Networks
ML	Machine Learning
MLP	Multi-Layer-Perceptron
NLP	Natural Language Processing
RL	Reinforcement Learning
RNN	Recurrent Neural Networks
SOTA	State-of-the-Art
TL	Transfer Learning

Quellcodeverzeichnis

1 Einleitung

Die Automatic Text Summarization (ATS) ist dem Bereich des Natural Language Processing (NLP) zuzuordnen und gewinnt zunehmend an wissenschaftlicher Relevanz. Obgleich entsprechende Modelle mittlerweile nicht mehr völlig neuartig sind, weisen die Entwicklungen der vergangenen Jahre qualitativ noch viele Potenziale auf [Yang et al., 2019, S. 1-2]. Einsatzmöglichkeiten entsprechender ATS-Modelle sind beispielsweise die Zusammenfassung von Nachrichten, die Zusammenfassung von Gesprächsprotokollen oder auch die Generierung von Überschriften, um nur wenige zu nennen [Goncalves, 2020]. Ziel ist in jedem Fall die Verdichtung von Informationen und die Reduktion der Lesezeit, wie Abbildung 1.1 demonstriert.

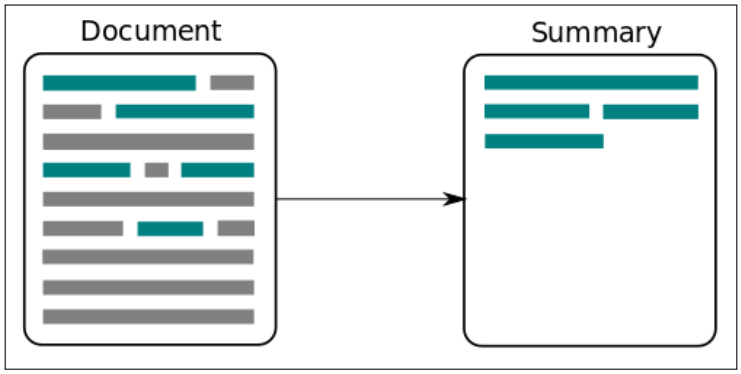

Abbildung 1.1: Ablauf einer automatischen Zusammenfassung [Thaker, 2019].

Mit besonderem Fokus auf das Gesundheitswesen lassen sich weiterhin zwei konkrete Einsatzgebiete konstruieren, in denen ein ATS-Modell in einem ganzheitlichen System als autarkes Modul implementiert werden könnte. Einerseits ist die Zusammenfassung von Patientengesprächen denkbar, wenn eine entsprechende Spracherkennung mit integrierter Sprechererkennung vorgeschaltet ist. Die verdichteten Informationen ließen sich anschließend zum Beispiel in Patientenakten exportieren oder anderweitig klassifizieren. Andererseits können Pflegeroboter, welche mitunter demente Patienten betreuen, durch ein ATS-Modell mit notwendigem Kontextwissen für die anstehenden Gespräche ausgestattet werden.

Die Anforderungen an ein ATS-Modell lassen sich aus dem individuell anvisierten Einsatzgebiet ableiten und können anhand verschiedener Faktoren klassifiziert werden. Demnach kann man prinzipiell zwischen dem extraktiven und dem abstraktiven Ansatz differenzieren [Gambhir et al., 2016, S. 5]. Extraktive Methoden bewerten die Sätze des ursprünglichen Textes anhand wort- und satzbezogener Attribute. Die Zusammenfassung entsteht sodann aus dem bewertungsgerechten Kopieren dieser Sätze [Kiani, 2017, S. 205-207]. Abstraktive Methoden hingegen verwenden Deep-Learning-Algorithmen, um Informationen zu identifizieren und entsprechende Zusammenfassungen mit völlig neuen Sätzen zu generieren [Nitsche, 2019, S. 1]. Weiterhin ist zu entscheiden, ob einzelne oder mehrere Dokumente zusammengefasst werden sollen, welcher Domäne diese Dokumente entstammen und ob möglicherweise eine Dialogorientierung vorliegt.

Aus technischer Sicht kommen zur ATS grundsätzlich sogenannte Sequence-to-Sequence-Modelle zum Einsatz. Dabei wird stets eine Eingabesequenz $x = [x_1, ..., x_n]$ in eine Ausgabesequenz $y = [y_1, ..., y_m]$ überführt, wobei n die Eingabelänge und m die Ausgabelänge ist. Die Sequenzen werden von Vektoren repräsentiert. Mithin wird bei der ATS $m < n$ intendiert. Entsprechende Architekturen modellieren also konsequenterweise die Wahrscheinlichkeit $P(y \mid x)$ [Nitsche, 2019, S. 32-33]. Die maßgebliche Herausforderung ist hierbei zum einen, dass ATS-Modelle tatsächlich die wichtigsten Informationen einer Eingabesequenz identifizieren. Zum anderen gilt es, diese Informationen in eine entsprechende Ausgabesequenz zu integrieren. Eben diese Ausgabesequenz ist zudem orthographisch und grammatikalisch korrekt zu generieren.

1.1 Zielsetzung

Das Ziel dieser Arbeit ist dementsprechend die abstraktive Zusammenfassung einzelner Dokumente, wobei multilingual vortrainierte Modelle mittels Transfer Learning (TL) auf die deutsche Sprache adaptiert werden. Die Arbeit ist somit außerdem eine potenzielle Grundlage für die beiden konstruierten Einsatzgebiete aus dem Gesundheitswesen. Die Adaption auf die Domäne oder auch die Dialogorientierung ist nicht Teil dieser Arbeit. Die Forschungsfragen lauten wie folgt:

- Wie lassen sich Texte automatisiert zusammenfassen?
- Wie können bereits existierende Modelle auf eine andere Sprache adaptiert werden?
- Wie qualitativ und skalierbar ist die Lösung?

1.2 Aufbau der Arbeit

Nach der Einleitung werden zunächst die Grundlagen des Deep Learning (DL) und des NLP offengelegt. Im Kapitel des DL werden neuronale Netze als solches definiert und ausgewählte Architekturen, welche auf die Zielerreichung einwirken, vorgestellt. Die Eigenschaften und die Relevanz von Hyperparametern und von TL schließen sich an. Im Kapitel des NLP werden neben der prinzipiellen Arbeit mit natürlicher Sprache und der entsprechenden Vorverarbeitung insbesondere sogenannte Word Embeddings und Deep Language Representations thematisiert.

Bevor die bis dahin behandelten Komponenten in ein tatsächliches Modell integriert werden können, ist die Beschreibung der Datengrundlage erforderlich. Zum daran anschließenden abstraktiven Ansatz gehört die Erläuterung der Architektur, die Beschreibung des Trainingsprozesses und die Evaluation der Ergebnisse. Bei der sprachtechnischen Adaption des Modells auf die deutsche Sprache werden zuerst entsprechende Anpassungen an der ursprünglichen Architektur konzipiert, bevor erneut der Trainingsprozess beschrieben wird und die dazugehörigen Ergebnisse evaluiert werden.

1.3 Forschungsstand & Referenzen

Aufgrund der stetig fortschreitenden Entwicklungen überholt sich der Forschungsstand der ATS regelmäßig. Dennoch haben sich in den vergangenen Jahren gewisse Tendenzen erkennen lassen. Bereits zur Jahrtausendwende existierten erste ATS-Systeme. Waren die ersten Ansätze zumeist noch extraktiv, wurde sich in den vergangenen Jahren mehr und mehr auf die abstraktiven Ansätze konzentriert. Vor 2016 schienen Ansätze mit Recurrent Neural Networks (RNN) und Long-Short-Term-Memory-Networks (LSTM) sehr populär [Nallapati et al., 2016]. In den Jahren 2016 und 2017 etablierten sich Ansätze, welche auf Reinforcement Learning (RL) basierten [Paulus et al., 2017]. Seit 2018 legten diverse Ansätze mit Encoder-Decoder-Architekturen die Grundlage des heutigen State-of-the-Art (SOTA) [Yang et al., 2019, Rothe et al., 2020], denn um den SOTA konkurrieren fast ausschließlich sogenannte Transformer. Diese basieren auf den Encoder-Decoder-Architekturen, implementieren verschiedenartige Attention-Mechanismen und haben sich sowohl unter qualitativen als auch unter ökonomischen und ökologischen Aspekten bewiesen [Zhang et al., 2020].

Die Qualität der ATS kann mithilfe des sogenannten ROUGE-Scores evaluiert werden. Dieser wird ebenso wie andere noch unerklärte Architekturen in einem nachfolgenden Kapitel dieser Arbeit umfangreich erläutert und kann zunächst als gegeben betrachtet werden. Die folgenden ROUGE-Scores können als Vergleichswerte verstanden werden: R-1: 40.10, R-2: 18.95, R-L: 37.39.

Weiterhin hat der Durchbruch frei verfügbarer vortrainierter Modelle die NLP-Welt revolutioniert, wie beispielsweise Bidirectional Encoder Representations from Transformers (BERT) [Devlin et al., 2019] oder auch Embeddings from Language Models (ELMO) [Peters et al., 2018] sowie deren Weiterentwicklungen. Verschiedenste NLP-Aufgaben wie die ATS konnten hiervon sehr stark profitieren. Die konkreten Funktionsweisen werden ebenfalls im Verlauf dieser Arbeit offengelegt. Wissenschaftliche Publikationen, welche mit dieser Arbeit vergleichbar sind und in dieser Arbeit referenziert oftmals werden, lauten wie folgt:

- Text Summarization with Pre-Trained Encoders [Yang et al., 2019]
- German Abstractive Text Summarization using Deep Learning [Nitsche, 2019]
- Leveraging Pre-Trained Checkpoints for Sequence Generation [Rothe et al., 2020]

2 Deep Learning

Deep Learning ist ein Teilbereich des Machine Learning (ML). ML-Algorithmen analysieren Daten automatisiert mittels mathematischer Methoden der Mustererkennung. DL-Algorithmen bedienen sich hingegen vielschichtiger und hoch parametrisierter neuronaler Netze, um dem menschlichen Gehirn bestmöglich nachzuempfinden [Khanna, 2019, S. 455-457]. Dabei werden sehr große Datenmengen verarbeitet und analysiert, um einen Lerneffekt zu erzielen. Neben einer Eingabe- und einer Ausgabeschicht sorgen insbesondere die verborgenen Schichten für die prädizierte Tiefe. Hier werden Informationen weiterverarbeitet, abstrahiert und reduziert [Zhang et al., 2020, S. 131]. Die potenziellen Einsatzmöglichkeiten gehen über die der ML-Algorithmen hinaus. Der Aufbau neuronaler Netze sowie deren Funktionsweise und ausgewählte Architekturen werden in diesen Kapitel thematisiert. Hyperparameter und TL schließen sich an.

2.1 Neuronale Netze

Um den Aufbau und die Funktionsweise neuronaler Netze verstehen zu können, bedarf es zunächst der Beschreibung von Neuronen. Diese können im biologischen Sinne als Schalter verstanden werden, welche verschiedene Signale empfangen können und aktiviert werden, sobald genug Signale registriert wurden. Diese Aktivierung sendet folglich weitere Signale an andere Neuronen, wie Abbildung 2.1 im technischen Sinne exemplarisch skizziert [Kriesel, 2005, S. 42]. Hierfür werden Aktivierungsfunktionen benötigt, welche die gewichteten Eingangssignale in ein Ausgangssignal konvertieren. Sie ermöglichen es, nicht-lineare Zusammenhänge zwischen den Eingangs- und den Ausgangsdaten herzustellen [Zhang et al., 2020, S. 134].

Die elementarste Form neuronaler Netze wird Multi-Layer-Perceptron (MLP) genannt. MLP bestehen aus mehreren Schichten, deren Neuronen jeweils vollständig mit den Neuronen der umliegenden Schichten verbunden sind [Zhang et al., 2020, S. 131]. Der Verständlichkeit halber veranschaulicht Abbildung 2.2 einen solchen Aufbau mit nur einer verborgenen Schicht, welche aus fünf Neuronen besteht.

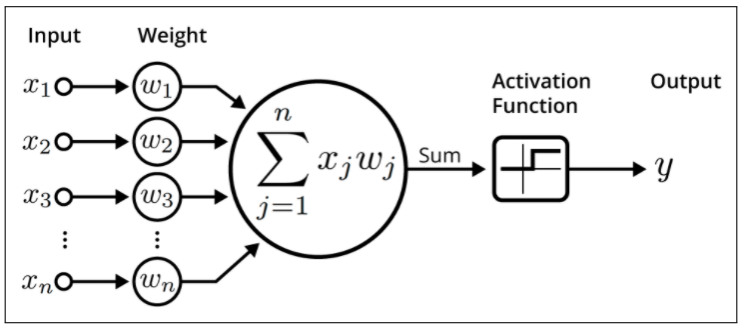

Abbildung 2.1: Aufbau eines künstlichen Neurons [McCullum, 2020].

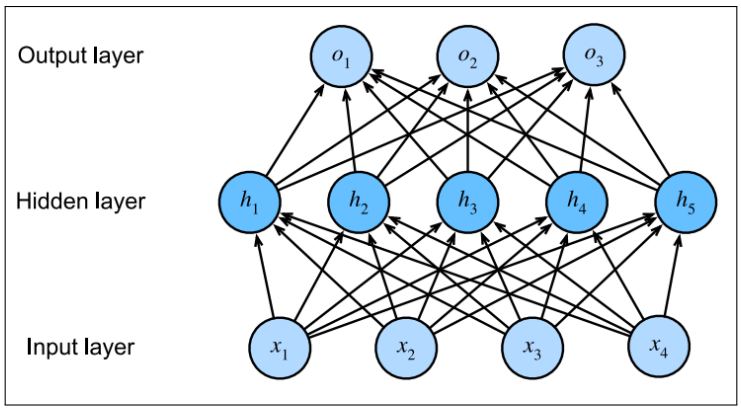

Abbildung 2.2: Aufbau eines MLP [Zhang et al., 2020, S. 133].

Ziel der hoch parametrisierten neuronalen Netze ist es, komplexe Funktionen hohen Grades bestmöglich zu approximieren und so verschiedenste Probleme zu lösen. Der anvisierte Lerneffekt wird mithilfe des sogenannten Backpropagation-Algorithmus erreicht. Hierbei werden Eingangsdaten zunächst vorwärts durch ein neuronales Netz hindurch propagiert. Mithilfe einer Fehlerfunktion wird sodann die erwartete mit der tatsächlichen Ausgabe verglichen und bewertet. Über das Gradientenverfahren werden die Fehler nun rückwärts durch das neuronale Netz propagiert und somit die Gewichte in den Neuronen angepasst, insbesondere in den verborgenen Schichten. Ziel ist die Minimierung der Fehlerfunktion und letztlich die Optimierung der durch das neuronale Netz approximierten Funktion [Zhang et al., 2020, S. 140, 169].

Der Trainingsprozess erfolgt optimalerweise über mehrere sogenannte Epochen. Hier werden dem neuronalen Netz verschiedene Eingangsdaten zugeführt und beidseitige Propagationen ausgeführt. Wichtig ist dennoch, kein Overfitting beziehungsweise Underfitting zu erzeugen. Dies würde bedeuten, dass das trainierte Modell zu sehr beziehungsweise zu wenig auf die Trainingsdaten angepasst ist. Ziel ist ein möglichst hoher Generalisierungs-

effekt des Modells, wie Abbildung 2.3 zeigt. Das Modell sollte den Lernfortschritt auf noch unbekannte Daten adaptieren können und darauf eine hohe Genauigkeit erreichen. Es gibt verschiedene Ansätze, um beispielsweise Overfitting vorzubeugen. Hier seien insbesondere Batch Normalization, Dropout und Early Stopping genannt, wobei entsprechende Mechanismen an anderweitiger Stelle erläutert werden [Zhang et al., 2020, S. 143-149].

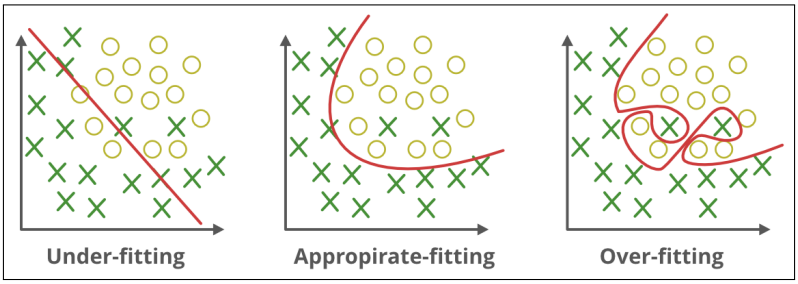

Abbildung 2.3: Typen von Generalisierungseffekten [Edpresso, O. J.].

2.2 Architekturen

Um mithilfe neuronaler Netze die ATS zu modellieren, werden nun ausgewählte Architekturen vorgestellt. Diese gehen weit über die als Grundlage beschriebenen MLP hinaus und verdeutlichen die Varietät neuronaler Netze.

2.2.1 Recurrent Neural Networks

[Zhang et al., 2020] ab Seite 361, 354, sequenzielle Daten fokussieren

2.2.2 Encoder-Decoder Networks

[Zhang et al., 2020] ab Seite 377, 375, YAN19 S. 3 links unten und rechts unten, siehe außerdem `https://colab.research.google.com/drive/1WIk2bxglElfZewOHboPFNj8H44_VAyKE?usp=sharing#scrollTo=Gw3IZYrfKl4Z` für Encoder-Decoder-Rechtfertigung im entsprechenden Theorie-Teil, Skizzen von dort nutzen

2.2.3 Attention in Neural Networks

[Zhang et al., 2020] ab Seite 389, 394 mit Self-Attention, Multi-Head-Attention, [Vaswani et al., 2017] nutzen, um Attention wissenschaftlich zu beschreiben, notfalls NLP für DL abhandeln, falls Inhalte hier erforderlich sind, dann entsprechend die Einleitungstexte anpassen

2.2.4 Transformer Networks

[Zhang et al., 2020] ab Seite 398 mit MH-Attention, Encoder, Decoder, Training etc. + Transformer-based Encoder-Decoder-Models erwähnen, schon mal [Rothe et al., 2020] bzgl. der Möglichkeit von Ersetzen des Encoder und und des Decoder durch vortrainierte multilinguale Modelle erwähnen, ebenfalls geeignet für sequenzielle Daten, Grenzen im nachfolgenden Kapitel bzgl. Transfer Learning genauer offenlegen, an [Raffel et al., 2020] orientieren, ggf. `https://arxiv.org/pdf/1912.08777.pdf` referenzieren, oder auch `https://arxiv.org/pdf/2007.14062.pdf`

2.3 Hyperparameter

Hyperparameter sind Parameter einer Architektur, die bereits vor dem eigentlichen Trainingsprozess definiert werden. Sie bedürfen einer separaten Optimierung, da sie eben dieses Training und folglich auch die Qualität des entstehenden Modells enorm beeinflussen. Ziel ist es hierbei, die beste Kombination aller Hyperparameter zu finden, um weiterhin die Fehlerfunktion zu minimieren [Yang et al., 2020, S. 1]. Im weiteren Verlauf werden ausgewählte Hyperparameter peripher vorgestellt.

2.3.1 Learning Rate

Die Learning Rate (LR) ist ein Hyperparameter, der bestimmt, wie viel Einfluss jede einzelne Epoche im Trainingsprozess auf die Anpassung der Gewichte nimmt. Sie gilt mithin als wichtigster Hyperparameter einer Architektur [Zhang et al., 2020, S. 428].

Dabei ist zunächst das Gradientenverfahren offenzulegen. Dieses dient der methodischen Lösung von Optimierungsproblemen. Entlang des negativen Gradienten wird das globale Minimum einer entsprechenden Fehlerfunktion gesucht, bis keine numerische Verbesserung mehr zu verzeichnen ist.

Eine zu niedrige LR kann den Trainingsprozess entweder stark verlangsamen oder dafür sorgen, dass kein Lernfortschritt mehr erzielt wird, da lokale Minima der Fehlerfunktion nicht übersprungen werden können und fälschlicherweise als globales Minimum interpretiert werden. Eine zu hohe LR kann hingegen sehr abrupte Anpassungen der Gewichte verursachen, sodass potenziell auch das globale Minimum übersprungen werden kann [Zhang et al., 2020, S. 414-415]. Abbildung 2.4 verdeutlicht diese Bedingungen. Ziel ist

allgemein eine möglichst schnelle Konvergenz.

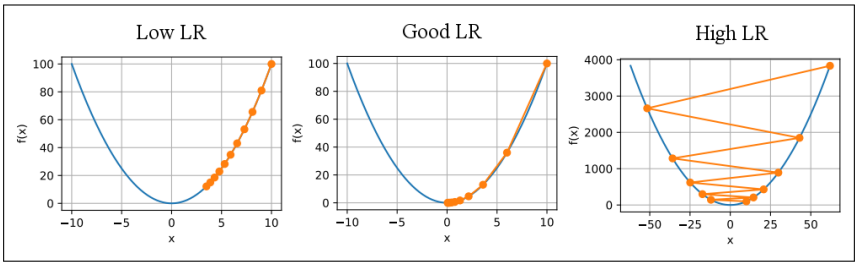

Abbildung 2.4: Konvergenz und Divergenz im Gradientenverfahren [Zhang et al., 2020, S. 429-430].

Neben der sorgfältigen manuellen Auswahl der LR, etwa mithilfe eines sogenannten LR-Schedule, ist es weiterhin möglich, eine adaptive LR einzuführen. Hierbei wird die LR in jeder Epoche verändert. Üblich ist hier die Reduktion der LR, wenn bereits akzeptable Ergebnisse erreicht wurden [Zhang et al., 2020, S. 433]. Außerdem existiert das stochastische Gradientenverfahren, welches pro Epoche nur eine Stichprobe der verfügbaren Trainingsdaten berücksichtigt. Hier ist von einem generalisierenden Effekt auszugehen [Zhang et al., 2020, S. 437].

2.3.2 Momentum

Das Momentum unterstützt die bereits beschriebene LR auf der Suche nach dem globalen Minimum in der Fehlerfunktion. Dabei berücksichtigt es den Durchschnitt vorheriger Gradienten. Auf dieser Grundlage wird entstiegen, in welche Richtung das stochastische Gradientenverfahren weiter absteigen soll, wie Abbildung 2.5 zeigt. Das Momentum ist somit potenziell in der Lage, lokale Minima zu überspringen und die Suche erst im tatsächlichen globalen Minimum zu beenden [Zhang et al., 2020, S. 453-456].

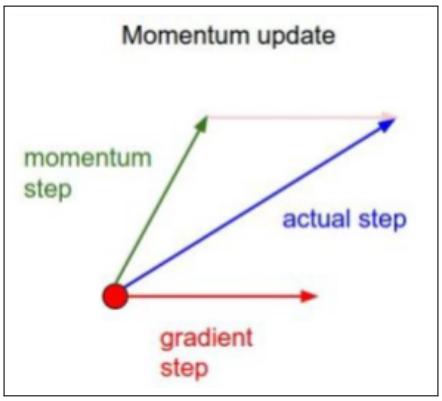

Abbildung 2.5: Gradientenverfahren unter Einfluss eines Momentums [CS231N, O. J.].

Bei der Auswahl eines hohen Momentums sollte die LR eher niedriger sein, oder anders herum. Eine Möglichkeit der stochastischen Optimierung ist hierbei Adaptive Momentum Estimation (ADAM). Dieser Algorithmus übernimmt nicht nur die Auswahl der adaptiven LR, sondern auch die Auswahl des entsprechenden Momentums. ADAM arbeitet somit effizient für daten- und parameterintensive Probleme. Dabei konvergiert der Algorithmus üblicherweise schneller als vergleichbare Optimierungsalgorithmen [Kingma et al., 2017, S. 1-2].

2.3.3 Weight Decay

Weight Decay meint die Multiplikation der Gewichte einer Architektur nach jeder Epoche mit einem Faktor kleiner als eins, um sehr große Gewichte zu verhindern. Die Gefahr von Overfitting wird hierbei verringert, während sich die Generalisierung des Modells verbessert [Zhang et al., 2020, S. 154].

2.3.4 Batch Size

Die Batch Size definiert die Größe der Stichprobe, welche je Epoche durch eine Architektur propagiert wird, ehe die entsprechenden Gewichte der Neuronen angepasst werden. Die Einführung einer Batch Size reduziert insbesondere den Rechenaufwand und den Speicherbedarf des Trainingsprozeses [Zhang et al., 2020, S. 446]. Grundsätzlich lässt sich die optimale Kombination aller Hyperparameter durch Techniken wie Grid Search annähern [Yang et al., 2020, S. 24].

2.4 Transfer Learning

TL ist in den letzten Jahren wissenschaftlich immer bedeutsamer geworden, da DL-Modelle heutzutage sehr komplex und Trainingsprozesse sehr zeit- und rechenintensiv sind. Unter TL versteht man das Wiederverwenden bereits vortrainierter neuronaler Netze für die Lösung neuartiger Probleme. Das initiale Training obliegt hierbei meist großen Unternehmen oder Institutionen. Dabei werden die erprobten Modelle sodann als Startpunkt genutzt und nur noch auf die neuen Probleme adaptiert, anstatt eigene Modelle von Grund auf neu zu trainieren. Anwender profitieren hier zeitlich, qualitativ und technisch. Zumeist sind architektonische Anpassungen in den hinteren Schichten der vortrainierten Modelle erforderlich, sodass sie sich für die Lösung der neuen Probleme eignen, wie Abbildung 2.6 veranschaulicht. Zudem ist ein gezieltes weitergehendes Training mit entsprechenden Daten notwendig. Inwieweit die neuen Daten auf die

vortrainierten Modelle einwirken sollen, ist individuell zu erproben [Zhang et al., 2020, S. 554].

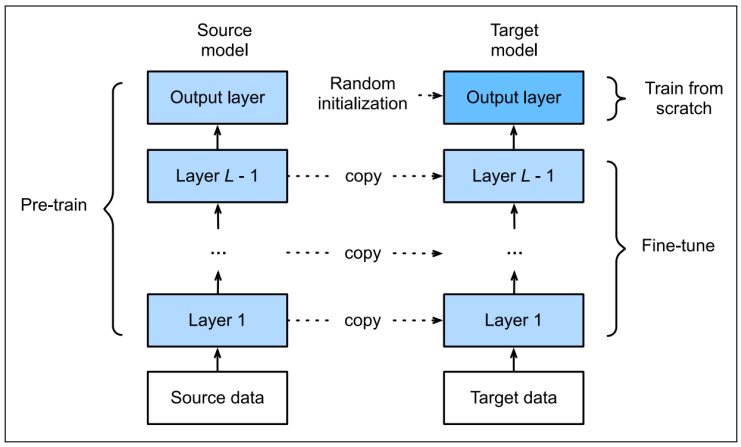

Abbildung 2.6: Fine-Tuning vortrainierter Modelle [Zhang et al., 2020, S. 555].

TL wird auch in dieser Arbeit genutzt. Einige Komponenten der bereits vorgestellten Architekturen, wie beispielsweise der Encoder oder auch der Decoder, können durch vortrainierte Modelle repräsentiert werden. Hier wird inhaltlich sowie kontextuell in den folgenden Kapiteln angeknüpft, da zunächst die Einführung weiterer NLP-Grundlagen erforderlich ist. Die angeführten Vorteile von TL können nichtsdestotrotz folgendermaßen zusammengefasst werden:

- Zeitersparnis durch Überspringen des initialen Trainings
- Qualitätsanstieg und Generalisierung durch Berücksichtigung massenhafter Daten
- Reduktion von hardwaretechnischen Anforderungen, entsprechenden Kosten und dem Stromverbrauch

3 Natural Language Processing

Notizen:

- Quelle: [Nitsche, 2019]
- Natural Language Processing definieren, e.g. Natural Language Understanding?
- NLP ist Optimierungslösung, d.h. es gibt keine eindeutige und damit im mathematischen Sinne analytische Lösung, Beispiel bei der Textzusammenfassung: Selbst Menschen können Texte auf verschiedene Arten und Weisen zusammenfassen, und verschiedene Varianten können korrekt sein
- NLU ist Teilgebiet des NLP
- Umfang der Anwendungsgebiete andeuten
- Natural Language Generation bspw. zum Generieren von Texten anhand von Stichworten benutzen, sollte bereits in gutem Zustand implementierfähig sein, möglicherweise Strukturen hiervon für die Generierung der Zusammenfassung verwenden, NLP-Links: `https://www.analyticsvidhya.com/blog/2020/08/build-a-natural-la` `https://www.analyticsvidhya.com/blog/2019/09/introduction-to-pytorch-from-scratc` `?utm_source=blog&utm_medium=Natural_Language_Generation_System_using_` `PyTorch`, `https://courses.analyticsvidhya.com/courses/natural-language-processing-` `utm_source=blog&utm_medium=Natural_Language_Generation_System_using_` `PyTorch`
- `https://github.com/adbar/German-NLP#Data-acquisition`
- `https://github.com/JayeetaP/mlcourse_open/tree/master/jupyter_english`
- Spacy: `https://spacy.io/usage/processing-pipelines#pipelines`
- Lemmatizer: `https://github.com/Liebeck/spacy-iwnlp`

- Transfer Learning with German BERT? `https://deepset.ai/german-bert` -¿ Modell muss die deutsche Sprache nicht alleine und von neu mit den Trainingsdaten lernen, sondern erhält einen großen Vorsprung, BERT ist Modell, welches der Transformer-Architektur nachkommt, d.h. Transformer sind bestimmte Architekturen, eventuell hiermit die Struktur dieses Kapitels überarbeiten, hier für vor allem aus meinem privaten Verzeichnis das Paper "Pre-Training of Deep Bidirectional Transformers for Language Understanding using BERT"nutzen
- GLoVe-Embeddings nutzen, weil TF-IDF etc. nicht den Kontext eines Satzes betrachten
- Supervised Learning nutzen, aber es ist eventuell nicht genug, hier kommt bspw. Transfer Learning mit BERT zur Abhilfe, zudem bspw. semi-supervised Learning mit Auto-Encoders? Self-supervised Training
- Siehe Abstract im Exposé

3.1 Vorverarbeitung

Notizen:

- Pipeline der Vorverarbeitung als Voraussetzung hervorstellen
- Relevanz von Capitalization, Punctuation, Zeilenumbrüchen klären, auch im Negativfall begründen und belegen, Satzzeichen für die Minimierung von Zwei- oder Uneindeutigkeiten berücksichtigen

3.1.1 Textbereinigung

Notizen:

- Siehe vergangene Aufgaben in GitHub

3.1.2 Tokenisierung

Notizen:

3.1.3 POS-Tagging

Notizen:

3.1.4 Lemmatisierung

Notizen:

- Lemmatisierung eventuell irrelevant, weil Wort-Tokenisierung bei modernen Architekturen und Modellen oftmals ausreicht
- Nach erfolgreichem Aufsetzen der Pipeline kann man die Eingangsdaten testweise immer noch der Lemmatisierung oder weiteren Vorverarbeitungsschritten unterziehen, um deren Auswirkungen zu messen

3.1.5 Entfernen von Stoppwörtern

Notizen:
Weitere Notizen, die eingearbeitet werden sollten:

- Relevanz für extraktiven Ansatz beschreiben (vgl. Paper: „Automatic Text Summarization“)
- Relevanz für abstraktiven Ansatz, falls vorhanden, beschreiben
- Metriken selbst weiterentwickeln und ausreifen
- Siehe: `https://scikit-learn.org/stable/modules/feature_extraction.html`
- Übereinstimmung mit dem Titel, Satzposition, Satzähnlichkeit, Satzlänge, domänenspezifische Wörter, Eigennamen, numerische Daten

3.2 Word Embeddings

Notizen:

- Bereich des Language Modeling
- Word2Vec
- BOW
- BPE
- GloVe

3.3 Deep Language Representations

Notizen:

- BERT als Encoder & Decoder nutzen, Architekturen und TL dementsprechend aufgreifen (S. 1 in YAN19), bspw. als Encoder oder/ und Decoder verwenden, siehe ROT20 auf S. 2 rechts und S. 6 unten, Encoder zur NLU und Decoder zur NLG, d.h. BERT oder andere Transformer als vortrainiertes multilinguales Modell für Encoder/ Decoder nutzen, Ausblick auf Adaption von EN-¿DE: Fine-Tuning der Modelle, verschiedene Experimente, aber dazu im späteren Kapitel mehr, BERT ist außerdem austauschbar, durch sowohl größere als auch kleinere Modelle, Multilingualität architektonisch ergründen, erwähnen, dass diese Modelle die Encoder oder auch Decoder ersetzen können, hierzu populäre Ansätze wie [Rothe et al., 2020], Trainingsvorgehensweise von BERT und ELMo beschreiben, wie im Forschungsstand bereits erwähnt, konnten diese vortrainierten Language Models die NLU-Welt revolutionieren, wie viele Parameter wurden genutzt? Verschiedene Modelle vergleichen, bspw. das Notebook `https://colab.research.google.com/drive/1WIk2bxglElfZewOHboPFNj8H44_VAyKE?usp=sharing#scrollTo=4M2uzGLV9a_O` unter Änalysis"referenzieren, jeweils maximale Token-Länge oder verschieden trainierte Versionen hervorheben, ggf. LongFormer als Encoder benutzen
- ELMo
- GPT
- Transfer Learning mit BERT hier sinnvoll, sodass das Modell die Sprache nicht in einer bestimmten Domain oder mit zu wenigen Texten neu erlernen muss, TL durch Encoder, Decoder etc. auf jeden Fall hier aufgreifen
- BERT zunächst in Englisch nutzen, weil SOTA, ggf. Grafik aus Oli's VL integrieren, irgendwie in Abstractive Summarization Pipelines integrieren
- BERT ist auch multilingual, d.h. englisches Modell mit deutschem Fine-Tuning vermutlich sogar brauchbar
- Modell beschreiben, d.h. Datensätze und Parametrisierung, SOTA für verschiedene NLP-Tasks, von Kapazitäten profitieren, hat viel Kontextwissen, Fine-Tuning für eigenes Problem, d.h. domainspezifisch o.ä.

- Am besten direkt ein vortrainiertes Transformer-Modell nutzen (extra für Summarization-Tasks), BERT und RL bspw. in der Pipeline integrieren, Ziel wäre dann: Verbesserung im Score erzielen
- BERT vielleicht durch andere (teils bessere und neuere) Transformer ersetzen? Transformer in NLP recherchieren, LSTM als veraltet bezeichnen

4 Datengrundlage

Notizen:

- Modelle erfordern keine gelabelten Daten, wohl aber gesichtete Daten
- Siehe Abstract im Exposé

4.1 Akquise

Notizen:

- ROT20 auf S. 6 unten und rechts
- Wikihow- und CNN-Dailymail-Korpora beschreiben, bspw. sind etwa 230.000 Wikihow-Paare zu erwarten, aber per Skript auswerten, Texte mit unter 1.000 Wörtern ausschließen, d.h. 444.365 Paare aus beiden Korpora schrumpfen um 33.177 Paare auf 411.188 in die Trainingsdaten eingehende Paare
- Data Collection: Akquise mittels Skripten in Python, zunächst mit grober Vorverarbeitung, noch nicht entsprechend der NLP-Pipeline
- Zielform der Textdateien beschreiben, Ablagestruktur ebenfalls
- Datenquellen: Wikipedia-API (`https://pypi.org/project/Wikipedia-API/`, rekursiv für 263.000 Texte), OpenLegalData-Dumps (`https://de.openlegaldata.io/pages/api/`, `https://static.openlegaldata.io/dumps/de/2019-10-21/` für 100.000 Texte), tensorflow-datasets (use latex-boxes when using bib), also `https://www.tensorflow.org/datasets/catalog/wikihow` mit 157.252 Texten, in denen Themen beantwortet werden, `https://www.tensorflow.org/datasets/catalog/gigaword` mit 3.803.957 Sätzen, `https://zenodo.org/record/1168855#.X75WfmhKiUk` mit 3.084.410 Sätzen und `https://www.tensorflow.org/datasets/catalog/cnn_dailymail` mit 287.113 Newsartikel, jeweils mit entsprechender Zusammenfassung), Unterschiede und Dokumentation siehe Excel (bspw. EN-DE)

- Datenherkünfte beschreiben, d.h. Dateiformat, Größe, Sprache etc. beschreiben
- Von den Datenquellen wird vermutet und nach manueller Einsicht bestätigt, dass Texte dort grammatikalisch korrekt sind, außerdem allgemeinsprachlich und ausreichend lang (¿ 1000 Wörter, es wird angenommen, dass 1000 Wörtern vorliegen müssen, um eine Zusammenfassung erforderlich zu machen) sind und möglichst diversifizierten Themengebieten entstammen
- Testdaten aus anderen Domänen vorbereiten und dokumentieren
- V1: Englischsprachige Korpora aus verschiedenen Branchen aus Text-Zusammenfassung-Paaren beschaffen und übersetzen
- V2: Deutschsprachige Korpora aus Text-Zusammenfassung-Paaren anfragen
- V3: Englischsprachige Korpora verwenden, um Modellarchitektur zu entwickeln und Modell zu trainieren, Adaption auf die deutsche Sprache als separates anschließendes Arbeitspaket, NLP-Vorverarbeitung überarbeiten und Modell neu trainieren
- V3 nutzen, bei Erfolg auf V1 ummünzen, oder: Eigenen deutschen Korpus aus den lokalen Agenturen aufbereiten und nutzen, Herkunft und Struktur beschreiben, Vorgang der Akquise ebenfalls, dann Fine-Tuning mit diesen Daten

4.2 Vorverarbeitung

Notizen:

- Vorgehen wie im Notebook beschreiben, d.h. warum cased/ uncased? Data Preprocessing beschreiben, bereits Anforderungen an Encoder/ Decoder stellen, d.h. Token-Länge, durchschnittliche Länge der Texte und der Zusammenfassungen gleichzeitig hierfür analysieren
- Daten iterieren, jeweils die Klassen zum Data Cleaning, Tokenisierung, Lemmatisieren etc. für einen einzelnen Text aufrufen, ggf. per weiteren Exporten zwischenspeichern, zuvor alle möglichen Dateien sichten und möglichst viele Fehler im Laufe des erneuten Exportes eliminieren, Ablageorte und Textdateiversionen beschreiben, dann Train-Test-Split, Übergabe der vorverarbeiteten Daten an die Modelle, welche den Korpus von einer Klasse namens NLP-Pipeline bekommt

- Weitergehende Besonderheiten innerhalb der Texte werden toleriert, da diese auch im Praxisbetrieb auftreten könnten und somit gekannt werden sollten
- Möglicherweise Spell Checking von Google RS für die deutsche Sprache einbinden
- Interne Pipeline: Skripte zum Herunterladen erledigen Data Cleaning, NLP-Pipeline erledigt Tokenisierung und Lemmatisierung, Lemmatisierung ausschließen, mit der Vermutung, dass neuartige Verfahren ohne viele Vorverarbeitungsschritte auskommen, außerdem Notiz zu Capitalization, Punctuation, Zeilenumbrüchen: Stark modellabhängig, tiefe Modelle wie bspw. Transformer-Architekturen (BERT) kommen damit ganz gut klar, d.h. an denen orientieren, vermutlich Plain-Text reingeben, "die machen nicht mal lower-case", Annahme: Älles was ich reinstecke, kann ein potenzielles Feature sein", andere Vorverarbeitungsschritte verfälschen das Ergebnis insofern, als dass das Training anders erfolgt, als das Modell selbst trainiert wurde

4.3 Datensatz

Notizen:

- Datengrundlage besteht aus frei verfügbaren allgemeinsprachlichen, ausreichend langen und deutschsprachigen Daten, verschiedene Herkünfte
- Auf Grundlage dieser Allgemeinsprache und den eben genannten Vorhaben, sollte ein grundlegendes Modell trainiert werden und später für den Use Case eine Art Adaptive Learning betrieben werden, d.h. wenn bekannt ist, dass das Modell für medizinische Texte angewandt werden soll, sollte man vorher die Parameter des Modells finetunen
- Später dann zwecks Adaption auch unternehmensinterne fachspezifische Daten notwendig, genauer beschreiben, perspektivisch sogar fachspezifische, dialogorientierte oder auch mehrsprachige Modelle möglich, dementsprechend mehr Daten benötigt, ggf. erst im Ausblick erwähnen
- Ähneln medizinische Texte "normalen"Texten? Gefahr: Hohe Informationsdichte bei Diktaten - "Was fällt raus?"
- Ergebnisse beim Domänenübergriff? "falsch-positiv"?
- Skript zum Einlesen entwickeln, bspw. `data_loader`

- Sätze nur in geringem Anteil verwenden, d.h. knapp unter 500.000 realen Trainings- und/ oder Testdaten

5 Abstraktiver Ansatz

Notizen:

- Quelle: [Nitsche, 2019]
- Abgrenzung zum extraktiven Ansatz beschreiben
- Vorteile gegenüber referenzierten Modellen herausstellen
- Generierung neuer Sätze sowohl mit vorkommenden als auch mit nicht-vorkommenden Wörtern (vgl. Paper: „Automatic Text Summarization with Machine Learning“)
- Siehe Abstract im Exposé

5.1 Architektur

Notizen:

- [Rothe et al., 2020] um Encoder und Decoder zu ersetzen, laut beschriebener Encoder-Decoder-Architektur + Language Model + Transfer Learning in Kombination, also die drei Grundlagenkapitel werden hier zusammengeführt, sogenannter Warm-Start des Encoder-Decoder-Models, d.h. kein initiales Training erforderlich, welches im Bereich von NLP-Tasks ein Neuerlernen einer Sprache bedeuten würde, Architektur skizzieren, in der sowohl Encoder als auch Decoder "warm gestartet"werden, Fine-Tuning auch skizzieren, Warm-Starting beschreiben, wie hier in der Theorie beschrieben: `https://colab.research.google.com/drive/1WIk2bxglElfZewOHboPFNj8H44_VAyKE?usp=sharing#scrollTo=Gw3IZYrfKl4Z`, Skizzen von dort nutzen, Encoder-Decoder-Weight-Sharing beschreiben, zuvor muss hier erstmal beschrieben werden, wie die Encoder-Decoder-Architektur konkret aussieht, also BERT o.ä., dann verschiedene Skizzen und Dinge wie Weight-Sharing beschreiben, prinzipiell das Colab-Notebook nutzen, Multilingualität behandeln
- [Yang et al., 2019] S. 4 rechts, S. 5 oben für Evaluation, S. 6 links unten für Konfiguration

- Modellauswahl begründen, d.h. warum "Transformer"? Warum genau dieses vortrainierte Modell? Auf vorherige Inhalte der Masterarbeit verweisen
- Transformer-to-Transformer-Architektur beschreiben, Grundlagen des TL, der Encoder/ Decoder und der Embeddings/ Language Model Representations etc. aufgreifen, bspw. Bert2Bert, Longformer2Roberta wie in `https://huggingface.co/patrickvonplaten/longformer2roberta-cnn_dailymail-fp16/blob/main/README.md`
- Netzwerk des abstraktiven Ansatzes als Pipeline skizzieren
- Vorgehen bei der Datenverarbeitung/ -vorbereitung wie im Notebook beschreiben, d.h. warum cased/ uncased? Token-Länge von BERT o.ä. als Begründung erwähnen
- Transformers-Library -¿ Scores in Excel, funktioniert gut als Benchmark/ "Nullfall"
- Komponenten wie den Encoder oder den Decoder mit der behandelten Theorie auf die durch die Datengrundlage gestellten Anforderungen mappen, d.h. tatsächlich eine Auswahl aus den vorgestellten theoretischen Inhalten treffen und stets ausführlich begründen
- Seq2Seq-Trainer wie im Notebook beschreiben, Notebook nochmal durchlesen und wichtige Informationen sinnvoll einbinden
- Seq2Seq `https://github.com/yaserkl/RLSeq2Seq#dataset` -¿ Fehler
- Seq2Seq-Library: `https://github.com/dongjun-Lee/text-summarization-tensorflow` -¿ Done, aber Scores auf meinem Datensatz nicht evaluierbar, da Aufbau des Vocabularies und Training auf meinem Korpus ausstehend ist, aber zu rechenintensiv ist, alternativ nur Scores auf anderem Korpus auswertbar, Azure ML vs. AWS SageMaker?
- Deep Reinforcement Learning (DeepRL) for Abstractive Text Summarization `https://medium.com/analytics-vidhya/deep-reinforcement-learning-deeprl-for-abstractive` -¿ Rouge-Scores ausschließlich auf dem CNN-Korpus berechnet, Anpassungen an den Daten, an der Code-Architektur und an den Modellen möglich -¿ Zurückgestellt aber bei Bedarf mit Potenzial

- BERT-Encoder Transformer-Decoder: Paper `https://arxiv.org/pdf/2008.09676.pdf`, Code `https://github.com/nlpyang/PreSumm`, Results `https://paperswithcode.com/paper/abstractive-summarization-of-spoken#code` -¿ In Progress...
- Deep Reinforced Model with PyTorch: `https://github.com/rohithreddy024/Text-Summarizer-Pytorch` -¿ TBD

5.2 Training

Notizen:

- Konfiguration
- Training verschiedener Modelle
- Kompressionsrate der Referenzzusammenfassungen in Bezug auf die Originaltexte liegt bei 12 Prozent, d.h. mit einer gewissen Toleranz wird die maximale Zusammenfassungslänge auf 15 Prozent des Originaltextes festgelegt

5.3 Evaluation

Notizen:

- ROUGE vorstellen, evtl. BLEU, auch die Implementierung beschreiben, ROUGE-Score umstritten, daher evtl. alternativen Score nutzen, Kritik inkl. Grenzen nennen, dennoch weitreichend genutzt und daher vergleichbar, wichtig für die Interpretation, ROUGE evaluiert uncased, `https://huggingface.co/metrics/rouge`
- Kompressionsrate messen
- Qualität der Zusammenfassung messen (BLEU `https://en.wikipedia.org/wiki/BLEU`, ROUGE `https://en.wikipedia.org/wiki/ROUGE_(metric)`, evtl. Funktionen fusionieren)
- Evaluation verschiedener Modelle mit geeigneter Vergleichstabelle
- Vergleich mit SOTA-Modellen
- Praktische Nutzung durch Implementation eines vortrainierten Modells in ein Skript oder eine Software

- Es muss eine Metrik existieren, mit der man die Genauigkeit bzw. Qualität der Zusammenfassung messen kann, d.h. man möchte die Texte nicht mit menschlich generierten Zusammenfassungen vergleichen, sondern automatisiert lernen, ggf. sollte man auch Grammatik und Inhalt separat prüfen

- For a given document there is no summary which is objectively the best. As a general rule, many of them that would be judged equally good by a human. It is hard to define precisely what a good summary is and what score we should use for its evaluation. Good training data has long been scarce and expensive to collect. Human evaluation of a summary is subjective and involves judgments like style, coherence, completeness and readability. Unfortunately no score is currently known which is both easy to compute and faithful to human judgment. The ROUGE score [6] is the best we have but it has obvious shortcomings as we shall see. ROUGE simply counts the number of words, or n-grams, that are common to the summary produced by a machine and a reference summary written by a human. `https://towardsdatascience.com/deep-learning-models-for-automatic-summarization-4c2b89f2`

- Bei der Anwendung einer Architektur, in der das Modell durch Reinforcement Learning trainiert wird, braucht man keine massenhaft menschlich generierten Referenztexte, sondern eine wohlbedachte Kostenfunktion, der ein entsprechender Aufwand entgegen gebracht werden muss, d.h. die Herausforderung liegt beim RL eher darin, eine Umwelt und eine geeignete Funktion zum Belohnen und Bestrafen zu konstruieren, hier sind bspw. auch Evaluationsmetriken notwendig

- Rouge-Score in Python: `https://pypi.org/project/rouge-score/`

- Typisches Diagramm zur Visualisierung des Trainingsprozesses anfügen

6 Sprachtechnische Adaption

- Konzeption der Adaption
- Adaption von EN-¿DE: Fine-Tuning der Modelle, verschiedene Experimente, umfangreiche Tabellen aufstellen

6.0.1 Konzeption

Text...

6.0.2 Training

Text...

6.0.3 Evaluation

Text...

7 Zusammenfassung

Notizen:

- Methoden und Ergebnisse zusammenfassen
- Bewertung der Zielerreichung
- Beantwortung der Forschungsfragen
- Lösung eingangs beschriebener Szenarien
- Welche Domäne wird am besten erkannt? Funktionieren Modelle mit gemischten Korpora?
- Siehe Abstract im Exposé

8 Diskussion und Ausblick

Notizen:

- Adaptive Learning für die Modelle ansatzweise vorstellen
- Modelle für mehrere Sprachen trainieren
- Modell auf Dialogcharakter adaptieren, um es in der Verdichtung von Protokollen einer Videosprechstunde zu nutzen, bzw. generell bspw. Meetings zusammenzufassen
- Forschungsstand und SOTA-Modelle hierfür im einleitenden Kapitel beschreiben, hier aufgreifen (vgl. Paper: „Abstractive Dialogue Summarization with Sentence-Gated Modeling Optimized by Dialogue Acts“ und “Using a KG-Copy Network for Non-Goal Oriented Dialogues”), bereits Architekturen vorstellen (vgl. „Automatic Dialogue Summary Generation of Customer Service“ und „Dialogue Response Generation using Neural Networks and Background Knowledge“ und „Global Summarization of Medical Dialogue by Exploiting Local Structures”)
- Modell ohne Anpassungen auf Konversationen anwenden: `https://www.isi.edu/natural-language/people/hovy/papers/05ACL-email_thread_summ.pdf`
- Modell nutzen, um Zusammenfassungen für Texte zu generieren und damit neue Datensätze für neue Modelle zu generieren, aber stark von der Qualität abhängig
- Gelb markierte Literatur sichten und verwenden, Datumsangaben aktualisieren
- Ausblick: Zusammenfassungen formatieren, also Großschreibung nach Satzenden oder auch Leerzeichenentfernung vor Punkten, Adaption auf Sprache und/ oder Domain
- Siehe Abstract im Exposé

Literaturverzeichnis

[Bird et al., 2009] Bird, Steven & Klein, Ewan & Loper, Edward: Natural Language Processing with Python, Verlag O'Reilly, Sebastopol, Vereinigte Staaten, 2009.

[CS231N, O. J.] Convolutional Neural Networks for Visual Recognition: Nesterov Momentum, in: `https://cs231n.github.io/neural-networks-3/`, Aufruf am 01.03.2021.

[Devlin et al., 2019] Devlin, Jacob & Chang, Ming-Wei & Lee, Kenton & Toutanova, Kristina: Pre-training of Deep Bidirectional Transformers for Language Understanding, Google AI Language, 2019.

[Edpresso, O. J.] Edpresso: Overfitting and Underfitting, in: `https://www.educative.io/edpresso/overfitting-and-underfitting`, Aufruf am 01.03.2021.

[Gambhir et al., 2016] Gambhir, Mahak & Gupta, Vishal: Recent Automatic Text Summarization Techniques, University of Panjab in Chandigargh, 2016.

[Goncalves, 2020] Goncalves, Luis: Automatic Text Summarization with Machine Learning, in: `https://medium.com/luisfredgs/automatic-text-summarization-with-machine-learning-an-overview-68ded5717a25`, Aufruf am 01.03.2021.

[Khanna, 2019] Khanna, Sachin: Machine Learning vs. Deep Learning, Department of Computer Science Engineering in India, 2019.

[Kiani, 2017] Kiani, Farzad: Automatic Text Summarization, University of Arel in Istanbul, 2017.

[Kingma et al., 2017] Kingma, Diederik & Ba, Jimmy: Adam - A Method for Stochastic Optimization, University of Amsterdam and Toronto, 2017.

[Kriesel, 2005] Kriesel, David: Ein kleiner Überblick über neuronale Netze, in: `http://www.dkriesel.com/science/neural_networks`, Aufruf am 01.03.2021.

[McCullum, 2020] McCullum, Nick: Deep Learning Neural Networks Explained, in: `https://www.freecodecamp.org/news/deep-learning-neural-networks-explained-in-plain-english/`, Aufruf am 01.03.2021.

[Nallapati et al., 2016] Nallapati, Ramesh & Zhou, Bowen & Dos Santos, Cicero & Gulcehre, Caglar & Xiang, Bing: Abstractive Text Summarization using Sequence-to-Sequence RNNs, Conference on Computational Natural Language Learning, 2016.

[Nitsche, 2019] Nitsche, Matthias: Towards German Abstractive Text Summarization using Deep Learning, HAW Hamburg, 2019.

[Paulus et al., 2017] Paulus, Romain & Xiong, Caiming & Socher, Richard: A Deep Reinforced Model for Abstractive Summarization, in: https://arxiv.org/pdf/1705.04304v3.pdf, Aufruf am 01.03.2021.

[Peters et al., 2018] Peters, Matthew & Neumann, Mark & Iyyer, Mohit & Gardner, Matt & Clark, Christopher & Lee, Kenton & Zettlemoyer, Luke: Deep Contextualized Word Representations, Allen Institute of AI in Washington, 2018.

[Raffel et al., 2020] Raffel, Colin & Shazeer, Noam & Roberts, Adam & Lee, Katherine & Narang, Sharan & Matena, Michael & Zhou, Yanqi & Li, Wei & Lio, Peter: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, Google Research, 2020.

[Raschka et al., 2019] Raschka, Sebastian & Mirjalili, Vahid: Machine Learning and Deep Learning with Python, Verlag Packt, Birmingham, Vereinigtes Königreich, 2019.

[Rothe et al., 2020] Rothe, Sascha & Narayan, Shashi & Severyn, Aliaksei: Leveraging Pre-Trained Checkpoints for Sequence Generation Tasks, Google Research, 2020.

[Thaker, 2019] Thaker, Madhav: Comparing Text Summarization Techniques, in: https://towardsdatascience.com/comparing-text-summarization-techniques-d1e2e465584e, Aufruf am 01.03.2021.

[Vaswani et al., 2017] Vaswani, Ashish & Shazeer, Noam & Parmar, Niki & Uszkoreit, Jakob & Jones, Llion & Gomez, Aidan & Kaiser, Lukasz & Polosukhin, Illia: Attention Is All You Need, Google Research, 2017.

[Yang et al., 2019] Yang, Liu & Lapata, Mirella: Text Summarization with Pretrained Encoders, Institute for Language, Cognition and Computation in Edinburgh, 2019.

[Yang et al., 2020] Yang, Li & Shami, Abdallah: On Hyperparameter Optimization of Machine Learning Algorithms, University of Western Ontario, 2020.

[Zhang et al., 2020] Zhang, Aston & Lipton, Zachary & Li, Mu & Smola, Alexander: Dive into Deep Learning, in: https://d2l.ai/, Aufruf am 01.03.2021.

Thesen

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet.

Selbstständigkeitserklärung

Hiermit erkläre ich, Daniel Vogel, die vorliegende Masterarbeit selbstständig und nur unter Verwendung der von mir angegebenen Literatur verfasst zu haben. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegen.

Dresden, den ???. Juli 2021

Daniel Vogel

A Erster Anhang

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet.

B Zweiter Anhang

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet.