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

ATS	Automatic Text Summarization
BERT	Bidirectional Encoder Representations from Transformers
DL	Deep Learning
ELMO	Embeddings from Language Models
LSTM	Long-Short-Term-Memory-Networks
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]. Modelle sind prinzipiell als das Ergebnis des Trainingsprozesses neuronaler Netze zu verstehen. 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.

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 [Gambhir et al., 2016, S. 5]. Demnach kann man prinzipiell zwischen dem extraktiven und dem abstraktiven Ansatz differenzieren. 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. 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 vielversprechenden 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], bis 2018 verschiedenste Ansätze mit Encoder-Decoder-Architekturen die Grundlage des heutigen State-of-the-Art (SOTA) legten [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 insbesondere in der Trainingsgeschwindigkeit 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.

Nicht zuletzt sind die Ergebnisse positiv bedingt durch den Durchbruch und die kostenlose Bereitstellung vortrainierter Modelle, 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 konnten hiervon sehr stark profitieren, darunter auch die ATS. Die konkreten Funktionsweisen werden ebenfalls im Verlauf dieser Arbeit offengelegt.

Wissenschaftliche Publikationen, welche mit dieser Arbeit vergleichbar sind und in dieser Arbeit referenziert 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 Tasks [Rothe et al., 2020]

2 Deep Learning

Notizen:

- Quelle: [Zhang et al., 2020]
- Deep Learning definieren
- Machine Learning erwähnen
- Siehe Abstract im Exposé
- ATS on GitHub: `https://github.com/mathsyouth/awesome-text-summarization#corpus`

2.1 Neuronale Netze

Notizen:

- Neuronale Netze definieren
- Historie beschreiben
- Funktionsweise und ausgewählte Komponenten beschreiben

2.2 Architekturen

Notizen:

- Existenz und Notwendigkeit verschiedener Architekturen ankündigen, ggf. in spätere Kapitel verlegen, bspw. zum abstraktiven Ansatz
- Später benötigte Architekturen hier beschreiben
- Diversität der existierenden Architekturen (wie im Forschungsstand bereits erwähnt) hervorheben

- "Reinforcement Learning comes to the rescueäus https://towardsdatascience.com/deep-learning-models-for-automatic-summarization-4c2b89f2a9ea einbinden
- Encoder/ Decoder, Self-Attention, Seq to Seq, Transformer Model (Recherche + Vergleich)
- Transformer, bestehend aus Seq-to-Seq-Model mit Encoder-/ Decoder-Architektur, gut erklärt: https://medium.com/inside-machine-learning/what-is-a-transformer-d wissenschaftliche Paper hierzu: https://arxiv.org/abs/1706.03762, https://wiki.pathmind.com/, https://nlp.stanford.edu/pubs/emnlp15_attn.pdf, Struktur ggf. überarbeiten, d.h. langsam an Seq to Seq, Encoder, Decoder heranführen
- PyCharm-Rebuild: Seq-to-Seq with Local Attention-: https://github.com/JRC1995/Abstractive-Summarization, https://nlp.stanford.edu/projects/glove/, https://nlp.stanford.edu/pubs/emnlp15_attn.pdf, https://arxiv.org/abs/1409.3215, https://arxiv.org/abs/1409.0473
- PyCharm-Rebuild: Bert-Encoder with Transformer-Decoder: https://github.com/santhoshkolloju/Abstractive-Summarization-With-Transfer-Learning
- PyCharm-Rebuild: RL-Seq-to-Seq: https://github.com/yaserkl/RLSeq2Seq, https://arxiv.org/abs/1805.09461

2.2.1 Feed-Forward Networks

[Zhang et al., 2020] ab Seite 331

2.2.2 Recurrent Neural Networks

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

2.2.3 Encoder-Decoder Networks

[Zhang et al., 2020] ab Seite 377, 375, YAN19 S. 3 links unten, S. 3 rechts unten

2.2.4 Attention in Neural Networks

[Zhang et al., 2020] ab Seite 389, 394 mit Self-Attention, Multi-Head-Attention

2.2.5 Transformer Networks

[Zhang et al., 2020] ab Seite 398 mit MH-Attention, Encoder, Decoder, Training etc.

2.3 Hyperparameter

Notizen:

- [Zhang et al., 2020] ab Seite 413 in den Unterkapiteln schauen
- Hyperparameter vorstellen
- Notwendigkeit und Einfluss von Hyperparametern beschreiben
- Batch-Size, e.g. Mini-Batch vs. Stochastic Batch: `https://stats.stackexchange.com/questions/153531/what-is-batch-size-in-neural-network`

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. Dabei werden die erprobten Modelle 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 2.1 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 und zu entscheiden [Zhang et al., 2020, S. 554].

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 wie folgt zusammengefasst werden:

- Zeitersparnis durch Überspringen des initialen Trainings
- Qualitätsanstieg und Generalisierung durch Berücksichtigung massenhafter Daten
- Reduktion der hardwaretechnischen Anforderungen und des Stromverbrauches

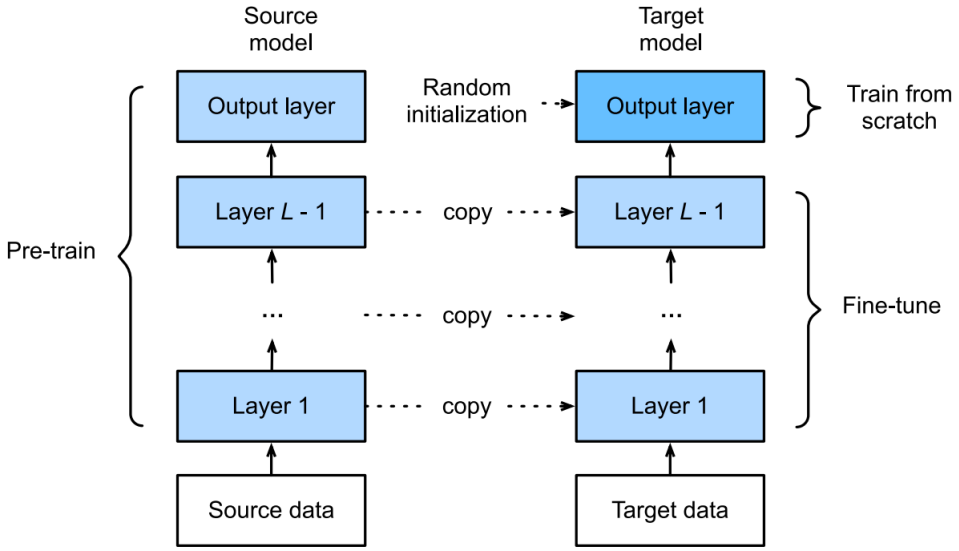

Abbildung 2.1: Fine-Tuning vortrainierter Modelle [Zhang et al., 2020, S. 555]

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-langua` `https://www.analyticsvidhya.com/blog/2019/09/introduction-to-pytorch-from-scratch/?utm_source=blog&utm_medium=Natural_Language_Generation_System_using_PyTorch`, `https://courses.analyticsvidhya.com/courses/natural-language-processing-nlp?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, TL aufgreifen (S. 1 in YAN19)
- 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
- 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:

- 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:

- 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“)
- Verschiedene Ansätze `https://medium.com/analytics-vidhya/deep-reinforcement-learning-de` evtl. im Forschungsstand erwähnen
- Siehe Abstract im Exposé

5.1 Architektur

Notizen:

- YAN19 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
- Netzwerk des abstraktiven Ansatzes als Pipeline skizzieren
- Transformers-Library -¿ Scores in Excel, funktioniert gut als Benchmark/ "Nullfall"
- 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-abstrac` -¿ 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
- 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-4c2b89f2a9ea`

- 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 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

7 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.

[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.

[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.

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

[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.

[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.

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

[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.