Überprüfung von Freitextargumenten auf syntaktische Korrektheit

Bachelorarbeit

von

Alexander Hering

aus

Düsseldorf

vorgelegt am

Lehrstuhl für Rechnernetze
Prof. Dr. Martin Mauve
Heinrich-Heine-Universität Düsseldorf

August 2021

Betreuer:
Markus Brenneis, M. Sc.

Zusammenfassung

Das Ziel dieser Bachelorarbeit besteht darin ein Verfahren zu entwickeln, um zu überprüfen ob, gegebene Freitextargumente syntaktisch korrekte kausale Nebensätze sind.
Dies geschieht unter dem Hintergrund der Überprüfung von Textsegmenten, welche Nutzer im Online-Diskussionssystem „D-BAS“ an vorgegebene Textbausteine anhängen.
Die vorgegebenen Textbausteine enthalten dabei bereits Einleitungswörter für Kausalsätze („weil“, „da“, etc.), welche üblicherweise für eine einfache Klassifizierung von Kausalsätzen verwendet werden. Ein Beispiel für einen solchen Textbaustein wäre: „Ich stimme dem Argument zu, weil ...“.

Da die Einleitungswörter bereits in den Textbausteinen vorgegeben werden, stehen sie allerdings in den zu untersuchenden Textsegmenten nicht zur Klassifizierung zur Verfügung.
Der Fokus dieser Ausarbeitung liegt darauf ein neuronales Netz zur Erkennung von kausalen Nebensätzen zu trainieren und dessen Genauigkeit zu maximieren. Hierzu werden sowohl verschiedene Datenbasen als auch Methodiken zur Vorabüberprüfung der Eingabedaten untersucht.
Das Ergebnis ist ein neuronales Netz, welches mit einer Genauigkeit von ca. 94% erkennen kann, ob es sich bei einem gegebenen Eingabetext um einen Kausalsatz handelt.

Die finalen Probleme, welche eine Erhöhung der Genauigkeit verhindern, bestehen in syntaktisch inkorrekten Eingabedaten sowie einer sehr geringen Menge an Trainingsdaten im exakt zu erkennenden Format.

Inhaltsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Das vom Fachbereich „Rechnernetze“ entwickelte Onlinediskussionsportal D-BAS bietet die Möglichkeit, unter Beachtung verschiedener Standpunkte, über beliebige Themen zu diskutieren. Zur besseren Verwaltung der Diskussionen werden hierbei die von Nutzern eingegebenen Daten automatisiert verarbeitet und wieder in verschiedenen Kontexten angezeigt.

Bei einer Diskussion mit dem Thema „Soll eine Anwesenheitspflicht an der Universität für Seminare eingeführt werden?“ könnte ein Nutzer z. B. zustimmen, dass keine Anwesenheitspflichten für Seminare eingeführt werden sollten.
Wählt ein Nutzer diese Option, so generiert das System einen Satz wie „Ich stimme zu, dass Anwesenheitspflichten nicht eingeführt werden sollten.“ und fragt den Nutzer „Was ist Ihr wichtigster Grund dafür, dass Anwesenheitspflichten nicht eingeführt werden sollten?“

Hierauf hat der Nutzer die Möglichkeit entweder etwas zu einem bisherigen Argument eines Themas zu sagen oder ein neues Argument anzugeben.
Zu diesem Zweck gibt das System eine Satzeinleitung wie „Ich akzeptiere, dass Anwesenheitspflichten nicht eingeführt werden sollten, weil ...“ vor, welche der Nutzer vervollständigen kann, um ein neues Argument in die Diskussion einzubringen. Grammatikalisch korrekt wäre z. B. „für manche Studenten ein Vollzeitstudium nicht möglich ist.“
Dieses Argument wird vom System mit den bisherigen ausgewählten Optionen zusammengeführt und anderen Nutzern z. B. als „Ich argumentiere, dass Anwesenheitspflichten nicht

eingeführt werden sollten, weil für manche Studenten ein Vollzeitstudium nicht möglich ist.“ angezeigt.

Um die eingegebenen Daten grammatikalisch korrekt in andere Kontexte einzubinden ist es wichtig, dass diese in der korrekten grammatikalischen Form vorliegen. Im häufigsten Fall wird den Nutzern ein Kausalsatzbeginn im Stil von „Ich stimme dem Argument zu, weil ...“ vorgegeben, welcher vervollständigt werden muss. Hierbei kann es aus verschiedenen Gründen dazu kommen, dass ungültige Eingaben, z. B. vollständige Sätze, übermittelt werden, welche bei der automatischen Verarbeitung ebenfalls falsch an anderer Stelle eingesetzt werden.

Um dies zu verhindern, beschäftigt sich diese Bachelorarbeit damit, Methoden zu untersuchen, mit denen falsch formulierte Eingaben an dieser Stelle erkannt werden können.

1.2 Lösungsansatz

Die Erkennung von Kausalsätzen im Deutschen erfolgt üblicherweise durch die Suche nach den Kausalsatz einleitenden Stichwörtern. Diese sind z. B. „weil“, „darum“, „deshalb“, usw. Da in den meisten Fällen nur eine sehr geringe Anzahl dieser Stichwörter verwendet wird, gestaltet sich die Erkennung von Kausalsätzen in üblichen Texten als nicht besonders schwierig. Im Fall der Eingabenüberprüfung für D-BAS wird das Initialwort „weil“ jedoch bereits vom System vorgegeben, so dass die Nutzereingabe dieses nicht enthält. Dieser Umstand erschwert die korrekte Erkennung von Kausalsätzen massiv, da die Satzstruktur eines Kausalsatzes ohne Initialwort von der anderer Satzstrukturen abweichen kann, aber nicht muss. So ist z. B. „ ... , weil ich denke, dass das Thema wichtig ist“ ein korrekt formulierter kausaler Nebensatz, aber „Ich denke, dass das Thema wichtig ist“ ein vollständiges Satzkonstrukt aus Hauptsatz und Konsekutivsatz. Im Gegensatz dazu ist jedoch „Ich bin der Meinung, dass das Thema wichtig ist“ wieder ein korrektes Satzkonstrukt aus Hauptsatz und Konsekutivsatz, während „ ... , weil ich bin der Meinung, dass das Thema wichtig ist“, kein korrekt formulierter Kausalsatz ist.

Aufgrund der Vielzahl an möglichen gültigen und ungültigen Satzkonstruktionsmöglichkeiten der deutschen Sprache, ist eine sichere Identifikation von Kausalsätzen ohne Initialwort, nach aktuellem Stand der Technik, allerdings äußerst schwierig.

Es ist jedoch durch den Vergleich mit bestehender Literatur möglich zu untersuchen, wie stark sich der zu untersuchende Satz von bekannten, kategorisierten Sätzen unterscheidet und anzugeben, wie wahrscheinlich der zu untersuchende Satz ein Kausalsatz ist.

Um eine möglichst hohe Präzision zu erzielen, wurde ein neuronales Netz auf diversen Textkorpora trainiert und anschließend untersucht, wie genau die untersuchten Nutzereingaben kategorisiert wurden.

1.3 Aufbau der Arbeit

Die folgenden Kapitel führen in die verwendeten Methoden zur Lösung dieses Problems ein, beschreiben die Ergebnisse und geben ein abschließendes Fazit zu den Ergebnissen.

In Kapitel 2 wird das verwendete neuronale Netz genauer beschrieben. Es wird auf die allgemeine Art des neuronalen Netzes, die verwendete Bibliothek, die verwendete Trainingsmethode sowie den Aufbau der Trainingsdaten eingegangen.

Kapitel 3 beschreibt den Ablauf der verwendeten Arbeitsschritte sowie deren Ergebnisse. Es wird ebenfalls auf die Quellen und Aufbereitungsmethoden der Trainingsdaten eingegangen.

Kapitel 4 zieht ein abschließendes Fazit, stellt vor welche Trainingsdaten die höchste Genauigkeit versprechen und gibt Vorschläge zur weiteren Verbesserung der Genauigkeit.

Kapitel 2

Methodik

In diesem Kapitel wird auf den Aufbau des verwendeten neuronalen Netzwerks, die verwendete Bibliothek sowie den Aufbau der Trainingsdaten eingegangen.
Diese Form des maschinellen Lernens wurde klassischeren Methoden, aufgrund der großen Fortschritte, welche neuronale Netze in den letzten Jahren gemacht haben, sowie der guten Verfügbarkeit freier Codebibliotheken, vorgezogen.
Gerade im Bereich der Sprachanalyse hebt sich dieser Ansatz aufgrund seiner weitaus höheren Genauigkeit von bisherigen Verfahren ab. [Dee] [Hea]

2.1 Aufbau des neuronalen Netzes

Dieser Abschnitt geht auf den Aufbau des verwendeten neuronalen Netzes ein, um einen Überblick über die Struktur und Methoden zu geben, welche verwendet werden um die im nächsten Abschnitt besprochenen Daten zu verarbeiten.

2.1.1 Auswahl der Bibliothek

Zur Entwicklung neuronaler Netze stehen aktuell diverse low-level APIs zur Verfügung wie z. B. TensorFlow, PyTorch, Torch oder Keras. Diese bieten tiefgehende Kontrolle über den Aufbau neuronaler Netze, wurden stark auf Performance optimiert und sind in den verwendeten Datenstrukturen zu vielen großen Bibliotheken kompatibel.

Der Nachteil, eine dieser Bibliotheken direkt einzusetzen, besteht jedoch darin, dass diese darauf ausgelegt sind, komplett neue neuronale Netze zu erstellen und von Grund auf zu trainieren und zu optimieren.
Da für das hier beschriebene Problem jedoch notwendig ist, dass das verwendete neuronale Netz in der Lage ist, mit in deutscher Sprache verfassten Texten umzugehen, wäre eine signifikante (Rechen-)Zeitinvestition nötig, um ein passendes neuronales Netz von Grund auf neu zu trainieren, welche sinnvoller zur Lösung des eigentlichen Problems als zur Erarbeitung der Grundlagen verwendet werden kann.

Um die aufzuwendende Zeit zu reduzieren, wird die Bibliothek FastAI in Version 2 verwendet.[Fas]
Diese Bibliothek setzt auf PyTorch auf und bietet alle von PyTorch bereitgestellten Möglichkeiten, konzentriert sich allerdings auf die in den letzten Jahren immer stärker verwendete Methode des Transferlearnings.

2.1.2 Transferlearning

Beim Transferlearning wird ein neuronales Netz verwendet, welches zuvor zur Lösung eines ähnlichen Problems verwendet wurde. Dieses wird dann dem aktuellen Problem angepasst.
Ein Beispiel hierfür wäre die Verwendung eines Netzes, welches darauf trainiert wurde zu erkennen, ob sich auf einem vorgelegten Bild ein Hund befindet. Dieses Netz könnte umtrainiert werden, damit es erkennen kann, ob auf vorgelegten Bildern Katzen vorhanden sind.
In beiden Fällen muss das Netzwerk in der Lage sein gewisse Formen oder Farbzusammenhänge zu erkennen. Ist dies erst einmal gelungen, ist der Schritt zur Erkennung von Hunden oder Katzen weitaus weniger rechenintensiv.
Wurden so z. B. in das erste neuronale Netz („Hundeerkennung“) einige hundert GPU-Jahre gesteckt um eine Erkennungsrate von 95 % zu erreichen, so ist es denkbar, dass das zweite neuronale Netz („Katzenerkennung“) dieselbe Erkennungsrate bereits nach einigen GPU-Tagen oder Stunden erreicht, sofern das erste neuronale Netz umtrainiert wurde.

Eine Variante dieses Ansatzes wurde zum Training des hier verwendeten neuronalen Netzes verwendet.

2.1.3 Art des neuronalen Netzes

Im Laufe der starken Verbreitung neuronaler Netze in den letzten Jahren haben sich verschiedene Arten von neuronalen Netzen für verschiedene Einsatzzwecke durchgesetzt. Diese unterscheiden sich vor allem in der Art wie einzelne Neuronen miteinander verknüpft sind.
Bei klassischen neuronalen Netzen wird davon ausgegangen, dass alle Eingabedateien eine uniforme Länge und Dimension haben. Da in unserem Fall jedoch die Eingabedaten beliebig kurz oder lang sein können, benötigen wir eine spezielle Form von neuronalen Netzen.

Die üblicherweise zur Verarbeitung von Texten genutzte Form ist ein „rekurrentes neuronales Netz“ (eng. Recurrent neural network) oder kurz RNN. Bei RNNs sind die Neuronenemitter mit den Neuroneninputs derselben Schicht verbunden. Je nach Variante geschieht dies mehr oder weniger vollständig.
Dies bietet vor allem zwei für diesen Anwendungsfall nützliche Eigenschaften.

Zum einen ist es somit möglich, die Eingabedaten in kleine sinnvolle Token zu unterteilen (z. B. Wörter oder Silben) und davon beliebig viele an das neuronale Netz zu übermitteln. Das Netzwerk verwendet dann pro Schritt nur einen Token sowie die Ausgabe der Verarbeitung des letzten Tokens. Dies löst das Problem, dass Eingabedaten für neuronale Netze üblicherweise eine fixe Größe besitzen müssen.
Zum anderen bietet die sequentielle Verarbeitung der Daten die Möglichkeit, alle bisherigen Daten des aktuellen Inputs bei der Verarbeitung jedes Tokens mit einzubeziehen.
Im Gegensatz zu anderen Typen neuronaler Netzwerke, welche komplette Inputdatensätze in einem Schritt einlesen, ist dieser Ansatz besser für Daten geeignet, die in ihrer Reihenfolge voneinander abhängen, wie sie grade in der Sprachanalyse vorkommen.

Ein Nachteil einfacher RNNs ist jedoch, dass sie ein „kurzes Gedächtnis“ haben. Dies bedeutet, dass Daten, welche früher in der Eingabe verarbeitet werden, immer weniger beachtet werden je länger der Eingabedatenstrom ist. Dieses Problem entsteht, da die Ausgaben der Neuronen in jedem Schritt mit einer neuen Eingabe und der bisherigen Ausgabe kombiniert und wieder an die Neuroneneingabe gesendet werden. Je mehr Schritte folgen, desto weniger relevant sind ältere Eingaben, da sie in den gesendeten Ausgabedaten nach und nach untergehen.

Eine Variante eines RNNs, welches dieses Problem löst wird „langes Kurzzeitgedächtnis“ (eng. Long short-term memory), kurz LSTM genannt [Gre+17]. In diesem neuronalen Netz

wird durch eine zusätzliche Ebene bestimmt, welche Daten wie relevant sind. Dies wird üblicherweise an der Stärke des Ausschlags der betreffenden Neuronen festgemacht. Je weniger relevant die aktuellen Eingabedaten sind, desto weniger stark fließen sie in die Ausgabedaten für die nächste Iteration ein.
Dies verbessert am Ende die Fähigkeit des neuronalen Netzes insbesondere lange Sätze korrekt zu klassifizieren.

Aufgrund dieser Eigenschaften bot es sich an ein LSTM einzusetzen, um eine möglichst hohe Genauigkeit bei der Klassifizierung der Kausalsätze zu erhalten.

LSTMs werden von FastAI direkt unterstützt und stehen, in einer weiterentwickelten Form, zur Verwendung bereit.
Die konkrete Variante, welche sowohl standardmäßig von FastAI, als auch in dieser Ausarbeitung verwendet wird, trägt den Namen AWD-LSTM [MKS17]. Dies steht für „ASGD Weight-Dropped LSTM“. ASGD wiederum steht für „Averaged stochastic gradient descent“ (deu. etwa: „gemittelter, stochastischer Gradientenabstieg“).
Dieses Netz ist intern wie ein normales LSTM aufgebaut, optimiert jedoch den Trainingsprozess um mit möglichst wenigen Trainingsiterationen eine möglichst genaue Erkennung zu ermöglichen.

2.2 Aufbau der Trainingsdaten

Die Auswahl der Trainingsdaten entscheidet maßgeblich über die Klassifizierungsgenauigkeit neuronaler Netze.
Im Laufe dieser Ausarbeitung wurden Trainingsdaten aus verschiedenen Quellen, alleinstehend oder in Kombination, verwendet. In diesem Abschnitt werden Quellen und Aufbau der verwendeten Trainingsdaten behandelt. Die konkrete Verwendung wird in Kapitel 3 angesprochen.

2.2.1 Genereller Aufbau der Trainingsdaten

Alle Trainingsdatensätze wurden dem neuronalen Netz als UTF-8 formatierte, zweispaltige, tabulatorseparierte CSV-Dateien zur Verfügung gestellt. Die Daten enthielten in der ersten

Spalte jeweils einen zu untersuchenden Text, üblicherweise einen oder mehrere Sätze oder einen Nebensatz, und in der 2. Spalte eine Zuordnung als „correct“ oder „incorrect“.

„correct“ bezeichnet hierbei einen gültigen Kausalsatz, „incorrect“ einen beliebigen anderen Text. Von der üblicherweise verwendeten Klassifizierung mit „true“ und „false“ wurde abgesehen, da diese von einigen Funktionen der verwendeten Bibliotheken als Strings und anderen als boolsche Werte interpretiert werden, was zu schwierig vorherzusehenden Problemen geführt hat. Ähnliche Probleme traten bei der Klassifizierung mit 0 und 1 auf.

Aufgrund des großen Umfangs einiger automatisiert vorbereiteter Datensätze, war es generell nicht auszuschließen, dass einzelne Daten falsch klassifiziert wurden, jedoch haben stichprobenartige manuelle Tests ergeben, dass dies nur in einer insignifikanten Anzahl von Fällen vorkam.
Einzelne Fehlklassifizierungen gehen somit in der Masse an Daten unter und sind für die letztendliche Genauigkeit des neuronalen Netzes nicht von Belang.

2.2.2 Datenquelle: D-BAS

Zum Zwecke der Verwendung in dieser Arbeit wurde ein Dump sämtlicher Argumentationsdaten der öffentlichen D-BAS-Testinstanz bereitgestellt. Da diese in einem JSON-Format vorliegen, wurden die JSON-Struktur sowie alle nicht deutschen oder nicht Argumentationsdaten entfernt und die restlichen Daten in ein passendes CSV-Format gebracht. Da keine Klassifizierung der Daten vorlag und eine automatische Klassifizierung bisher nicht möglich war, wurden die Daten manuell klassifiziert.
Grammatikalisch inkorrekte, aber von der Absicht als Kausalsatz geschriebene, Sätze wurden hierbei als inkorrekt markiert. Sätze mit Rechtschreibfehlern wurden klassifiziert als ob sie keine Rechtschreibfehler enthielten.

Die Anzahl der von dieser Datenquelle letzten Endes verwendbaren Datensätze beträgt 608.

Da im Verlauf der Ausarbeitung festgestellt wurde, dass dies eine unzureichende Datenmenge ist, um ein neuronales Netzwerk mit hoher Genauigkeit zu trainieren, wurden für das Training des neuronalen Netzes die in den folgenden Abschnitten beschriebenen Datenquellen hinzugezogen.

2.2.3 Datenquelle: DER STANDARD

Die österreichische Tageszeitung „DER STANDARD“ bietet über ihre Webpräsenz [Der] die Möglichkeit an, Nachrichtenartikel zu lesen und zu kommentieren. Über eine Million der so entstandenen Kommentare wurden im „One Million Posts Corpus“ [Omp] gesammelt, aufbereitet und zur Verfügung gestellt.

Dieser Textkorpus [SST17; SS18] enthält somit eine große Menge Texte, deren Schreibstil dem Stil von D-BAS Nutzern ähneln sollte. Nachteilig ist, dass Artikelkommentare häufig hastig geschrieben werden und oftmals Rechtschreib- oder Grammatikfehler enthalten. Dem wurde versucht mit der im nächsten Abschnitt beschriebenen Datenquelle entgegenzuwirken. Da die Satzstruktur kein Teil der Annotationen ist, wurden diese nicht verwendet. Stattdessen wurde nur der reine, unannotierte Textkorpus verwendet.

Eine manuelle Klassifizierung des Datensatzes war aufgrund dessen Umfangs nicht möglich, daher wurden die Daten automatisch annotiert. Hierzu wurde jeder einzelne Datensatz mit der Bibliothek „nnsplit“ [Min] in einzelne Sätze aufgespalten.
Kausalsätze können in der deutschen Sprache nur von einer sehr geringen Anzahl an Wörtern eingeleitet werden, deshalb wurde jeder dieser Sätze nach den häufigsten Einleitungswörtern für Kausalsätze (weil, da, zumal) durchsucht.

Wurde eines dieser Einleitungswörter nach einem Komma gefunden, so wurde der Satzteil nach dem Einleitungswort als „correct“ in die Trainingsdaten aufgenommen und der vorhergehende Satzteil, bis einschließlich dem Einleitungswort, verworfen. Das Einleitungswort zu verwerfen, obwohl es Teil des Nebensatzes ist, ist essentiell, da in den zu überprüfenden Daten bereits Einleitungswörter vorgegeben werden und sie somit nicht zur zu überprüfenden Nutzereingabe zählen.
Wurde keines dieser Einleitungswörter nach einem Komma gefunden, so wurde der Satzteil nach dem Einleitungswort als „incorrect“ in die Trainingsdaten aufgenommen.

Die Annahme hinter diesem Ansatz war, dass Personen, welche von sich aus einen Nebensatz mit einem der oben genannten Einleitungswörter beginnen, diesen auch als größtenteils korrekten kausalen Nebensatz formulieren.
Weiter wurde vermutet, dass auf diese Weise viel mehr korrekt als inkorrekt formulierte Sätze extrahiert werden, dass die inkorrekten Sätze beim Training in der Masse untergehen.

Die Anzahl der von dieser Datenquelle letzten Endes verwendbaren Datensätze beträgt 2.477.745.

2.2.4 Datenquelle: Wikipedia

Der deutsche Teil der freien Onlineenzyklopädie „Wikipedia“ [Wik] enthält eine große Menge an Texten zu verschiedensten Themen. Diese weichen zwar vom zu untersuchenden Kommentarformat ab, aber sind dafür üblicherweise in gehobener, grammatik- und rechtschreibfehlerfreier Form vorhanden. Wenngleich die Rohdaten frei und öffentlich verfügbar sind, wurde in dieser Ausarbeitung nicht auf selbige, sondern auf den „German Wikipedia Text Corpus“ [May] zurückgegriffen. Dieser enthält sämtliche Artikel der deutschen Wikipedia vom 01.10.2018 zusammen mit ihren Kommentaren aus den Artikeldiskussionen.
Da auch die Kommentare enthalten sind, ist der Gesamtstil dieses Textkorpus den zu untersuchenden Daten ähnlicher, als es die reinen Enzyklopädieeinträge wären. Da die Kommentare jedoch erfahrungsgemäß ebenfalls in einer höheren Sprachqualität verfasst werden, sollte der sprachlich gehobene Standard des Textkorpus bestehen bleiben.

Diese Daten wurden bereits gesäubert, in einzelne Sätze aufgeteilt und in ihrer Reihenfolge randomisiert.
Die Aufbereitung der Daten zum Training des neuronalen Netzes wurde wie bei der Datenquelle „DER STANDARD“ vorgenommen, wobei der erste Schritt, der Aufteilung der Texte in einzelne Sätze, in diesem Fall entfallen konnte, da bereits nur einzelne Sätze vorhanden waren.

Die Anzahl der von dieser Datenquelle letzten Endes verwendbaren Datensätze beträgt 44.751.113.

2.2.5 Übersicht Datenquellen

Insgesamt stehen an dieser Stelle die in Tabelle 2.1 angegebenen Mengen an Trainingsdaten aus den verschiedenen Datenquellen zur Verfügung.

Je nach Trainingsmethode wurden verschieden große Teile dieser Daten für den Trainingsprozess verwendet. Genauere Informationen zur Verwendung finden sich in den entsprechenden Abschnitten in Kapitel 3.

Quelle	Anzahl Datensätze
D-BAS	608
DER STANDARD	2.477.745
Wikipedia	44.751.113

Tabelle 2.1: Übersicht der verfügbaren Trainingsdaten

Kapitel 3

Ablauf

In diesem Kapitel wird auf den Aufbau des verwendeten neuronalen Netzwerks, die verwendete Bibliothek sowie den Aufbau der Trainingsdaten eingegangen.

3.1 Ansatz: Syntaxbäume

Zu Beginn wurde davon ausgegangen, dass zum Training des neuronalen Netzes nur die Daten aus D-BAS zur Verfügung stehen. Aufgrund des geringen Umfangs dieser Trainingsdaten im Vergleich zur Komplexität der zu lernenden Aufgabe, wurde die Erarbeitung einer Lösung direkt an diesen Daten als unwahrscheinlich angesehen.

Beim initialen Ansatz wurde deshalb versucht, die Komplexität der Trainingsdaten zu reduzieren, um die Erkennung gleichartiger Strukturen in den Trainingsdaten zu erleichtern.
Zu diesem Zweck wurden die zu untersuchenden Sätze, vor der Eingabe in das neuronale Netz, mittels eines Syntaxbaumgenerators in eine Textrepräsentation des entsprechenden Syntaxbaums umgewandelt.
Syntaxbäume repräsentieren einen analysierten Text nicht anhand der eigentlich verwendeten Wörter, sondern anhand einer Abhängigkeitsstruktur der einzelnen Satzbestandteile.
Somit wird eine wesentlich geringere Tokenmenge verwendet, als wenn jedes Wort der deutschen Sprache als eigener Token verwendet würde. Ein neuronales Netz hätte somit ein wesentlich kleineres Arbeitsvokabular, welches zur Erkennung von Satzstrukturen verstanden werden muss.

Auch wenn für die so entstandenen Trainingsdaten kein bestehendes Sprachmodell zur Anwendung von Transferlearning vorhanden war, wurde untersucht, ob die Verringerung des Wortschatzes ausreicht, um die Anzahl der korrekten Fälle auf ein Maß zu beschränken, für welches ein erfolgreiches Training, anhand der aus D-BAS extrahierten Trainingsdaten, möglich ist.

Zur Generierung der Syntaxbäume wurde der „Berkeley Parser“ [Pet+06] verwendet.

Allen verwendeten Texten wurde vor der Generierung der Syntaxbäume der Text „Ich stimme dem Argument zu, weil“ vorangestellt. Dies sollte sicherstellen, dass bei korrekt formulierten Eingabetexten ein kompletter, gültiger Satz vom Syntaxbaumgenerator verarbeitet wird und so im Fall korrekter Eingaben ähnliche Strukturen entstehen.

Eine Analyse der Genauigkeit, des mit diesen Daten angelernten neuronalen Netzes, ergab jedoch, dass die getroffenen Vorhersagen nicht von zufälligem Raten unterschieden werden konnten.

Nach einer Untersuchung der Ergebnisse wurden hierfür 2 Gründe ausgemacht.
Zum einen führten selbst kleine Änderungen der verwendeten Wörter zu starken Unterschieden in den erzeugten Syntaxbäumen.
Zum anderen war das verwendete neuronale Netz, welches für die Verarbeitung von eindimensionalen Eingabewertfolgen (Sätze) gedacht war, nicht in der Lage, die Zusammenhänge in der als zweidimensionalen Baumstruktur gedachten Daten (Syntaxbaum) korrekt in Relation zueinander zu setzen.

Da die Erstellung eines Netzes, zur Analyse zweidimensionaler Strukturen unbekannter Länge und Breite, jedoch wesentlich mehr Aufwand erfordert hätte als durch die erwarteten Ergebnisse zu rechtfertigen war, wurde der Versuch, die Komplexität der Eingabedaten durch die Verwendung von Syntaxbäumen zu reduzieren, abgebrochen.

3.2 Ansatz: Naives Training an D-BAS Daten

Im Folgeschritt wurde zur Erstellung einer minimalen Baseline, anhand derer sich alle weiteren Versuche messen lassen müssen, ein, wie in 2.1.3, AWD-LSTM ausschließlich mit den

vorhandenen Daten aus D-BAS trainiert.

Wird in FastAI ein AWD-LSTM Modell erstellt, so wird automatisch, falls nicht explizit deaktiviert, ein vortrainiertes Basismodell verwendet, welches bereits in der Lage ist mit Wörtern im Allgemeinen umzugehen.

Dieses Modell wurde mittels der D-BAS Daten weiter trainiert.
Die anhand des Validationsdatenanteils der Trainingsdaten berechnete Genauigkeit lag nach 5 Epochen bereits bei über 98 %. Ein Test mit manuell erstellten Verfikationsdaten ergab jedoch, dass das neuronale Netzwerk ausschließlich die trainierten Texte korrekt klassifizierte und bei anderweitig erstellten Texten wieder zufällige Vorhersagen traf.

Es wurde die Vermutung aufgestellt, dass die Menge der vorhandenen Trainingsdaten aus D-BAS nicht ausreichend war, um dem neuronalen Netz sowohl ein Verständnis der deutschen Sprache, als auch ein Verständnis für die Art der zu klassifizierenden Texte beizubringen.

3.3 Ansatz: ULMFiT auf D-BAS Daten

Um den Ansatz der ULMFiT-Methode besser erklären zu können, wird in diesem Abschnitt zunächst erklärt, wie die ULMFiT-Methode funktioniert und dann auf die tatsächliche Anwendung eingegangen.

3.3.1 Erklärung

Um die Erkennung syntaktischer Strukturen der deutschen Sprache zu verbessern, wurde der Trainingsansatz geändert.
2018 wurde der Ansatz des „Universal Language Model Fine-tuning for Text Classification“ [HR18] (kurz ULMFiT) entwickelt.
Dieser Ansatz versucht die Vorteile des immer stärker verbreiteten Transferlearnings in der Textklassifizierung zu maximieren und so bei geringer Rechenzeit und Trainingsdatengröße eine hohe Genauigkeit der Klassifizierung zu erzielen.

Diese Methode fand auch z. B. beim Training des GPT-2 Models Anwendung, welches An-

fang 2019 mit seiner Fähigkeit Schlagzeilen machte, anhand eines kurzen Einleitungstextes komplette Fachartikel zu verfassen. [KH]

Nach der ULMFiT-Methode wird als Basis des Trainings ein vortrainiertes Sprachmodell der verwendeten Sprache verwendet. Dieses Modell wird auf klassische Weise mit einem größeren Datensatz, z. B. einer Million Sätze, trainiert.
Zudem muss es sich um ein generatives Modell handeln, welches in der Lage ist, Vorhersagen über das vermutlich nächste verwendete Wort eines Satzes zu machen. Dieses Basismodell muss jedoch nur einmal pro Sprache erstellt werden und kann anschließend für beliebige weitere Zwecke umfunktioniert werden.
Im zweiten Schritt wird dieses Netzwerk, für einige wenige Epochen, mit den für das Training eines Klassifizierers vorgesehenen Trainingsdaten nachtrainiert. Die Klassifizierung ist für diesen Schritt irrelevant. Nur die reinen Textdaten sind von Interesse.
Auf diese Weise lernt das neuronale Netz, welches bereits mit einem großen Teil der verwendeten Sprache umgehen kann, auch das Vokabular der verwendeten Trainingsdaten und kann sein Verständnis der Sprache so erweitern.

Anschließend werden das generative neuronale Netz sowie das verwendete Vokabular als Basis für einen Klassifizierer verwendet.
Dieser wird dann mit den vorhandenen Trainingsdaten trainiert, wobei wesentlich weniger Trainingsdaten und Rechenzeit benötigt werden, als wenn dieser von Grund auf neu trainiert werden würde.

3.3.2 Anwendung

Zur Anwendung der ULMFiT-Methode, ohne ein Sprachmodell von Grund auf neu zu trainieren, wurde ein vortrainiertes Sprachmodell auf einem möglichst großen Textkorpus der deutschen Sprache benötigt. Zudem musste das Modell in einem Format vorliegen, welches mit den verwendeten Bibliotheken kompatibel ist.
Im ersten Ansatz wurde das Modell „ULMFIT for German“ [Fil] verwendet. Leider war dieses Modell für Version 1 der FastAI-Bibliothek konzipiert. Der Versuch, das Modell anhand des angegebenen Beispielcodes zu verwenden, schlug somit fehl, da diverse Funktionen nicht mehr oder in veränderter Form zur Verfügung standen.
Um dies zu umgehen, wurden alle im Beispielcode angegebenen Aufrufe auf neuere Versio-

nen umgebogen und alle verwendeten Parameter in aktuellere Varianten umgeschrieben.
Nachdem jedoch alle vorbereiteten Funktionen ohne Fehler ausgeführt werden konnten und ein erster Versuch unternommen worden war das Modell weiter zu trainieren, wurde die Programmausführung mit einem nicht zu behebenden Fehler beendet.
Eine Ursachenanalyse ergab, dass mit Version 1 der FastAI-Bibliothek erstellte Sprachmodelle auch nach Adaption der verwendeten Parameter und Funktionsaufrufe nicht mit Version 2 der Bibliothek kompatibel sind.
Da Version 1 von FastAI nicht mehr weiterentwickelt wird und Version 2 große Geschwindigkeitsvorteile und Unterstützung für modernere Sprachmodelle mitbringt, wurde der Ansatz, dieses Modell zu verwenden, nicht weiter verfolgt.

Im weiteren Verlauf der Ausarbeitung wurde das Projekt „fast.ai ULMFiT with SentencePiece from pretraining to deployment“ [Leu] gefunden. Dieses bietet vortrainierte ULMFiT kompatible Sprachmodelle für eine Vielzahl an Sprachen an.
Hiervon wurde das deutsche Sprachmodell mit einem Vokabular von 15.000 Tokens verwendet. Dieses Modell wurde gegenüber dem Modell mit 30.000 Tokens bevorzugt, da es eine geringere Perplexität besitzt und somit voraussichtlich eine höhere Genauigkeit besitzt.

Dieses Modell wurde, wie im vorherigen Abschnitt beschrieben, mit den aus D-BAS verfügbaren Daten nachtrainiert und anschließend verwendet, um einen Klassifizierer zu trainieren.

Das Ergebnis war leider immer noch nicht zufriedenstellend. Es war eindeutig ersichtlich, dass der Klassifizierer von den Trainingsdaten gelernt hatte, jedoch wurde das falsche Merkmal gelernt.
Aufgrund des Aufbaus der Trainingsdaten haben die meisten korrekten Kausalsätze eine kurze Länge, während die meisten fehlerhaften Eingaben aus mehreren Sätzen bestehen und somit eine größere Länge besitzen. Aus der geringen Anzahl an Trainingsdaten war dies das Merkmal, welches der Klassifizierer gelernt hatte.
In anschließenden Tests wurden beliebige kurze Texte als korrekt und beliebige lange Texte als inkorrekt klassifiziert.

Obwohl es somit durch die ULMFiT-Methode möglich war, dem Klassifizierer ein grobes Verständnis der deutschen Sprache zu geben, war die Trainingsdatenbasis immer noch zu klein, um die in D-BAS eingegebenen Nutzereingaben korrekt zu klassifizieren.

		Gesamt		Kategorie	
Ansatz	Genauigkeit	false pos.	false neg.	false pos.	false neg.
OMPC	0,7034	0,1264	0,1678	0,4911	0,2267

Tabelle 3.1: Auswertung: ULMFiT auf One Million Posts Corpus

3.4 Ansatz: ULMFiT auf One Million Posts Corpus

Um die Genauigkeit des Netzwerks zu erhöhen wurde im nächsten Schritt der, aus Kommentaren von DER STANDARD bestehende, „One Million Posts Corpus“ (OMPC) zum Training des Netzwerks verwendet. Aufgrund der Ähnlichkeit dieser Daten mit den zu untersuchenden Daten wurde erwartet, dass sich das resultierende neuronale Netz auch auf die Nutzereingaben aus D-BAS anwenden lässt.
Vor dem Training wurden die Daten, wie in Abschnitt 2.2.3 beschrieben, aufbereitet. Das neuronale Netz wurde anschließend für 10 Epochen ausschließlich auf diesen Daten trainiert. Die aus D-BAS exportierten Daten wurden bei diesem Ansatz nicht zum Training, sondern nur zur finalen Validierung der Ergebnisse verwendet.

Wie in Tabelle 3.1 zu sehen, ergab sich auf diese Weise eine Genauigkeit von ca. 70 %.

Dieser Ansatz funktionierte somit, musste aber noch feiner abgestimmt werden.

3.5 Ansatz: ULMFiT auf One Million Posts Corpus, Wikipedia und D-BAS

Da der „One Million Posts Corpus“ zwar eine große Menge an Datensätzen zum Training beinhaltet, die Qualität der Datensätze aber stark schwankt, wurde im nächsten Schritt versucht das Ergebnis des letzten Ansatzes dadurch zu verbessern, dass den Trainingsdaten weitere Datensätze aus Wikipedia und D-BAS beigemischt werden.
Zudem wurde festgestellt, dass in den vorherigen Trainingsdaten das Verhältnis von als korrekt und inkorrekt zu klassifizierenden Daten bei 1:40 lag. Somit waren nur 2,5 % der Trainingsdaten von der Form, die als Nutzereingabe gewünscht ist.
Hierdurch besteht die Gefahr, dass das neuronale Netz zu stark auf inkorrekte Eingaben trai-

Quelle	Anzahl Datensätze
D-BAS	300
DER STANDARD	360.522
Wikipedia	4.819.427
Summe	5.180.549

Tabelle 3.2: Übersicht über verwendete Trainingsdaten

		Gesamt		Kategorie	
Ansatz	Genauigkeit	false pos.	false neg.	false pos.	false neg.
OMPC	0,7034	0,1264	0,1678	0,4911	0,2267
+ Wikipedia und D-BAS	0,7890	0,1812	0,0291	0,6087	0,0417

Tabelle 3.3: Auswertung: ULMFiT auf One Million Posts Corpus, Wikipedia und D-BAS

niert wird und Textstücke, bei denen sich das Netzwerk in der Klassifizierung unsicher ist, eher als inkorrekt als als korrekt klassifiziert werden. Dies ist gut an den Ergebnissen des vorherigen Ansatzes zu sehen, in welchem Textstücke fast 50 % häufiger fälschlicherweise als inkorrekt als fälschlicherweise als korrekt klassifiziert wurden.

Aus diesem Grund wurden in diesem Ansatz die Daten des „One Million Text Corpus“ sowie die Daten aus Wikipedia zuerst in korrekte und inkorrekte Datensätze aufgeteilt und anschließend zufällig so viele als inkorrekt klassifizierte Datensätze verworfen, bis das Verhältnis von korrekten und inkorrekten Datensätzen bei 1:5 lag.
Ein Teil der Ungleichverteilung wurde beibehalten, da es wesentlich weniger Möglichkeiten gibt einen kausalen Nebensatz zu formulieren, als es Möglichkeiten gibt einen beliebigen anderen Text zu formulieren.
Somit soll das neuronale Netzwerk für inkorrekt formulierte Texte mehr Trainingsdaten erhalten, ohne jedoch so viel mehr Daten zu erhalten, dass die Klassifizierungswahrscheinlichkeiten in die falsche Richtung verschoben werden.

Die Anzahl an Trainingsdaten, die aus den in Abschnitt 2.2 genannten Quellen verwendet wurde, wird in Tabelle 3.2 veranschaulicht.

Die so angepassten Datensätze wurden zusammen mit ca. der Hälfte der Daten aus D-BAS zusammengeführt und zum Training des neuronalen Netzes verwendet.

Wie in Tabelle 3.3 zu sehen, ist hierbei eine eindeutige Verbesserung der Erkennungsgenauigkeit zu sehen.

Die Erkennungsrate inkorrekter Satzstrukturen hatte sich deutlich verbessert, während sich die Erkennungsrate korrekter Satzstrukturen verschlechtert hatte. Eine Analyse der fehlerhaft als korrekt klassifizierten Daten ergab, dass ein Großteil (87,5 %) der Erkennungsprobleme bei Texten auftraten, die aus mehreren Sätzen bestanden. Da eine Eingabe mehrerer Sätze während der Vervollständigung eines Satzbausteines in D-BAS allerdings ohnehin inkorrekt ist, wäre es ideal, wenn diese direkt als inkorrekt klassifiziert werden könnten ohne vom neuronalen Netz klassifiziert zu werden.

3.6 Ansatz: ULMFiT auf One Million Posts Corpus, Wikipedia und D-BAS mit Sentencesplitter

Sentencesplitter teilen einen beliebigen Eingabetext einer unterstützten Sprache in einzelne Sätze auf. In diesem Abschnitt wird auf die beiden verwendeten Sentencesplitter eingegangen. Zudem wird untersucht, wie sich die Genauigkeit des neuronalen Netzes durch Vorfilterung der Eingaben durch die Sentencesplitter ändert.

3.6.1 nnsplit

Der Sentencesplitter nnsplit [Min] verwendet ein neuronales Netz um Satzgrenzen zu erkennen. Die Struktur ist ein LSTM und ähnelt somit dem Aufbau des in dieser Ausarbeitung verwendeten neuronalen Netzes.
Über die genaueren Trainingsparameter macht der Autor von nnsplit keine Angaben.
In einem stichprobenartigen Test wurden alle getesteten Texte korrekt in ihre Sätze aufgeteilt. Der Autor selbst gibt an, dass Satzgrenzen mit einer Genauigkeit von 87,85 % erkannt werden.

Zur Verringerung der false positive Klassifizierungen des neuronalen Netzwerks wurde nnsplit der Klassifizierung des neuronalen Netzwerks vorgeschaltet.
Jeder Eingabetext wurde zuerst durch nnsplit in Sätze aufgeteilt. Bestand der Text aus mehr

		Gesamt		Kategorie	
Ansatz	Genauigkeit	false pos.	false neg.	false pos.	false neg.
OMPC	0,7034	0,1264	0,1678	0,4911	0,2267
+ Wikipedia und D-BAS	0,7890	0,1812	0,0291	0,6087	0,0417
+ nnsplit	0,8896	0,0421	0,0680	0,1413	0,0972

Tabelle 3.4: Auswertung: ULMFiT auf One Million Posts Corpus, Wikipedia und D-BAS mit Sentencesplitter nnsplit

als einem Satz, so wurde die Eingabe direkt als inkorrekt klassifiziert. Ansonsten wurde die Klassifizierung wie im vorherigen Abschnitt durch das neuronale Netzwerk vorgenommen.

Dieser Ansatz verringerte die Menge an false positives massiv, sorgte aber zugleich für mehr als eine Verdoppelung der Rate an false negatives, wie in Tabelle 3.4 dargestellt.

Angewendet auf den kompletten Testdatensatz stellte sich heraus, dass nnsplit alle Texte, die aus mehreren Texten bestanden, auch als solche erkannte.
Allerdings wurden einige einzelne Sätze fälschlicherweise in mehrere aufgeteilt. Dies betraf besonders solche, welche entweder mehrere Punkte (z. B. „etc..“ oder „z. B.“) oder Kommafehler enthielten (z. B. „das Studenten die ProPra 2 schon gehört haben nichts bringt“).

Da nnsplit keine Möglichkeit anbietet diese Fehler zu korrigieren, wurde versucht die Fehlerrate durch Verwendung einer anderen Bibliothek zu verbessern.

3.6.2 SoMaJo

SoMaJo [PU16] [Pro] ist ein regelbasierter Tokenizer. Dieser wendet eine Kaskade von regulären Ausdrücken auf den Eingabetext an und erstellt hierdurch eine Tokenrepräsentation des Textes. Es werden zuerst möglichst spezifische reguläre Ausdrücke verwendet um Spezialfälle wie URLs, E-Mailadressen etc. abzufangen. In weiteren Schritten werden immer generelle reguläre Ausdrücke verwendet.

Für diese Ausarbeitung ist dabei besonders relevant, dass Satzzeichen, die das Ende eines Satzes markieren, erst in einem der letzten Schritte durch Token umgewandelt werden. Zu

		Gesamt		Kategorie	
Ansatz	Genauigkeit	false pos.	false neg.	false pos.	false neg.
OMPC	0,7034	0,1264	0,1678	0,4911	0,2267
+ Wikipedia und D-BAS	0,7890	0,1812	0,0291	0,6087	0,0417
+ nnsplit	0,8896	0,0421	0,0680	0,1413	0,0972
+ SoMaJo statt nnsplit	0,9383	0,0356	0,0259	0,1158	0,0376

Tabelle 3.5: Auswertung: ULMFiT auf One Million Posts Corpus, Wikipedia und D-BAS mit Sentencesplitter SoMaJo

diesem Zeitpunkt wurden bereits alle Spezialfälle, welche Punkte enthalten, wie URLs, Abkürzungen, etc., abgefangen und es bleiben nur noch relevante Satzzeichen an Satzgrenzen über.
Somit lässt sich eine sehr hohe Genauigkeit in der Abgrenzung verschiedener Sätze erzielen ohne dass die konkrete Syntax des Satzes analysiert werden muss.
Für die allgemeine Tokenisierung mittels dieses Ansatzes wurde von den Autoren SoMaJos eine Genauigkeit von 99.73 % angegeben.

SoMaJo wurde, wie zuvor nnsplit, der Klassifizierung vorgeschaltet und erzielte eine fast genauso gute Rate von false negatives bei einer stark reduzierten Rate an false positives. Eine Übersicht über die Ergebnisse ist in Tabelle 3.5 zu sehen.

3.7 Ergebnisse

Die Erkennung korrekt formulierter Kausalsätze ohne Einleitungswort ist definitiv möglich und der Einsatz eines neuronalen Netzwerks hierfür geeignet. Wie an den Ergebnissen der verschiedenen Ansätze ersichtlich ist, ist es jedoch wichtig, sowohl eine große Menge an Trainingsdaten verwenden zu können als auch an Texten verschiedener Sprachstile zu trainieren.

Zudem sollte eine Vorüberprüfung der zu klassifizierenden Eingabedaten vorgenommen werden, wobei ungültige Daten direkt verworfen oder als ungültig markiert werden müssen.

Die Filterung von Mehrsatzeingaben kann, wie in den Ergebnissen der letzten drei Ansätze zu sehen ist, die Genauigkeit stark erhöhen.

Weiterhin war die Menge an Trainingsdaten im konkret zu klassifizierenden Fall unzureichend. Mit nur 608 Textsegmenten, die zum Training und zur Validation zur Verfügung standen, musste auf andere Datenquellen zurückgegriffen werden, welche die zu untersuchenden Textsegmente nicht perfekt abbildeten.
Hier könnte zu einem späteren Zeitpunkt eine höhere Genauigkeit erzielt werden, sobald eine ausreichende Datenmenge, z. B. 100.000 Datensätze, zur Verfügung steht.

In einer abschließenden Auswertung der falsch klassifizierten Sätze wurde zudem festgestellt, dass diese zu 30 % syntaktische oder Rechtschreibfehler enthielten. Hier wäre es denkbar eine weitere Voranalyse der Eingabedaten vorzunehmen und den Nutzer zur Korrektur der Eingabe zu bitten.

64 % der false positives wiederum sind syntaktisch korrekt, wurden aber als kausaler Hauptsatz formuliert. Da diese syntaktisch sehr nah an kausalen Nebensätzen liegen, ist es schwierig diese mit absoluter Genauigkeit durch das neuronale Netz erkennen zu lassen. Eine Möglichkeit zur Verbesserung der Erkennungsrate in diesem Fall wäre eine zusätzliche Datenquelle, welche eine größere Anzahl, z. B. 10.000, von als inkorrekt klassifizierten kausalen Hauptsätzen enthält. Diese könnten dem Trainingsdatensatz hinzugefügt werden, um die Erkennungsrate zu verbessern.

Auch ohne diese Anpassungen sollte die Genauigkeit jedoch ausreichen, um ein Vorschlagsystem zu implementieren, welches den Nutzer auf mögliche Unstimmigkeiten in seiner Eingabe hinweist.
Ob die Genauigkeit ausreicht, um als ungültig erkannte Eingaben direkt abzulehnen, muss im Rahmen des Einsatzzweckes abgewogen werden, da aufgrund der false negative Rate von ca. 4 % im Schnitt jede 25. korrekte Eingabe abgelehnt würde.

Kapitel 4

Fazit

Die Erkennung der gesuchten Nebensätze durch ein neuronales Netz ist mit einer durchaus akzeptablen Genauigkeit möglich. Die Schwierigkeiten liegen jedoch einerseits darin eine ausreichende Menge geeigneter Trainingsdaten zu finden und andererseits in der Qualität der Nutzereingaben.

Die Menge an Trainingsdaten konnte durch die Extraktion passender Satzstrukturen aus bestehenden großen Textkorpora erhöht werden. Während mit kleineren Datenmengen keine sinnvolle Klassifizierung möglich war, konnte die Genauigkeit durch Vergrößerung des Umfangs der Trainingsdaten deutlich erhöht werden.

Da die Trainingsdaten nur einzelne (Teil-)Sätze enthielten, ist die Erkennungsgenauigkeit von Mehrsatzeingaben gering, was allerdings durch die Vorfilterung der Eingabedaten durch Sentencesplitter umgangen werden kann. Die so resultierende Genauigkeit von fast 94 %, kann durchaus in Produktivsystemen zur Generierung von Vorschlägen verwendet werden.

Tabelle 4.1 zeigt hierbei wie die Vergrößerung der Trainingsdatengrundlage sowie die Verwendung eines hochwertigen Sentencesplitters die Genauigkeit verbessern.

Die größten Mengen an noch bestehenden Fehlklassifizierungen bestehen aus Eingaben, die entweder syntaktisch fehlerhaft oder als kausale Hauptsätze formuliert sind.
Hier könnte man noch für eine Verbesserung ansetzen.

Zur besseren Klassifizierung syntaktisch fehlerhafter Satzstrukturen wäre eine weitere Vorfilterung durch einen Syntaxvalidator denkbar, welcher ungültige Sätze direkt als inkorrekt

Ansatz	Genauigkeit
ULMFiT auf One Million Posts Corpus	0,7034
ULMFiT auf One Million Posts Corpus, Wikipedia und D-BAS	0,7890
ULMFiT auf One Million Posts Corpus, Wikipedia und D-BAS mit nnsplit	0,8896
ULMFiT auf One Million Posts Corpus, Wikipedia und D-BAS mit SoMaJo	0,9383

Tabelle 4.1: Klassifizierungsgenauigkeit verschiedener Ansätze

markiert.

Um die Klassifizierungsgenauigkeit kausaler Hauptsätze zu verbessern wäre der Aufbau eines weiteren Trainingskorpus möglich, welcher ausschließlich als inkorrekt markierte kausale Hauptsätze enthält. Dieser Korpus sollte eine ausreichende Menge an Datensätzen, mindestens einige tausend, enthalten und kann den anderen Trainingsdaten beigemischt werden. Dies sollte die Erkennungsgenauigkeit weiter erhöhen.

Abschließend wäre eine größere Menge an Trainingsdaten wünschenswert, welche tatsächlich aus dem System stammen, für welches später die Klassifizierung erfolgen soll. Während ein Training auf ähnlichen Daten, wie gezeigt, möglich ist, können diese nicht die konkrete Art der zu untersuchenden Nutzereingaben widerspiegeln.

Ein erneutes Training des Netzwerks sowie eine erneute Auswertung der Genauigkeit ist also zu empfehlen, wenn zu einem späteren Zeitpunkt mehr Trainingsdaten aus dem System zur Verfügung stehen, dessen Daten klassifiziert werden sollen.

Literatur

[Dee] *Der DeepL Übersetzer im Vergleich zur Konkurrenz*. https://www.deepl.com/quality.html.

[Der] *DER STANDARD*. URL: https://www.derstandard.at.

[Fas] *FastAI*. URL: http://docs.fast.ai/.

[Fil] Johannes Filter. *ULMFIT for German*. URL: https://github.com/jfilter/ulmfit-for-german.

[Gre+17] Klaus Greff u. a. „LSTM: A Search Space Odyssey“. In: *IEEE Transactions on Neural Networks and Learning Systems* 28.10 (2017), 2222–2232. ISSN: 2162-2388. DOI: 10.1109/tnnls.2016.2582924. URL: http://dx.doi.org/10.1109/TNNLS.2016.2582924.

[Hea] Will Douglas Heaven. *Sprach-KI GPT-3: Schockierend guter Sprachgenerator*. URL: https://www.heise.de/hintergrund/GPT-3-Schockierend-guter-Sprachgenerator-4867089.html.

[HR18] Jeremy Howard und Sebastian Ruder. *Universal Language Model Fine-tuning for Text Classification*. 2018. arXiv: 1801.06146 [cs.CL].

[KH] Wolfgang Stieler Karen Hao. *Künstliche Intelligenz: Fortsetzung folgt*. URL: https://www.heise.de/hintergrund/Kuenstliche-Intelligenz-Fortsetzung-folgt-4403643.html.

[Leu] Florian Leuerer. *fast.ai ULMFiT with SentencePiece from pretraining to deployment*. URL: https://github.com/floleuerer/fastai_ulmfit.

[May] Philip May. *German Wikipedia Text Corpus*. URL: https://github.com/t-systems-on-site-services-gmbh/german-wikipedia-text-corpus.

[Min] Benjamin Minixhofer. *nnsplit*. URL: https://github.com/bminixhofer/nnsplit.

[MKS17] Stephen Merity, Nitish Shirish Keskar und Richard Socher. „Regularizing and Optimizing LSTM Language Models“. In: (2017). arXiv: 1708.02182 [cs.CL].

[Omp] *One Million Posts Corpus*. URL: https://ofai.github.io/million-post-corpus.

[Pet+06] Slav Petrov u. a. „Learning Accurate, Compact, and Interpretable Tree Annotation“. In: *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*. https://github.com/slavpetrov/berkeleyparser. Sydney, Australia: Association for Computational Linguistics, Juli 2006, S. 433–440. DOI: 10.3115/1220175.1220230. URL: https://aclanthology.org/P06-1055.

[Pro] Thomas Proisl. *SoMaJo*. URL: https://github.com/tsproisl/SoMaJo.

[PU16] Thomas Proisl und Peter Uhrig. „SoMaJo: State-of-the-art tokenization for German web and social media texts“. In: *Proceedings of the 10th Web as Corpus Workshop*. Berlin: Association for Computational Linguistics, Aug. 2016, S. 57–62. DOI: 10.18653/v1/W16-2607. URL: https://aclanthology.org/W16-2607.

[SS18] Dietmar Schabus und Marcin Skowron. „Academic-Industrial Perspective on the Development and Deployment of a Moderation System for a Newspaper Website“. In: *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC)*. Miyazaki, Japan, Mai 2018, S. 1602–1605. URL: http://www.lrec-conf.org/proceedings/lrec2018/summaries/8885.html.

[SST17] Dietmar Schabus, Marcin Skowron und Martin Trapp. „One Million Posts: A Data Set of German Online Discussions“. In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. Tokyo, Japan, Aug. 2017, S. 1241–1244. DOI: 10.1145/3077136.3080711.

[Wik] *Wikipedia*. URL: https://de.wikipedia.org/wiki.

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Düsseldorf, 27. August 2021 Alexander Hering