HOCHSCHULE für Angewandte Wissenschaften
Fachhochschule LANDSHUT · UNIVERSITY of Applied Sciences
Hochschule für Technik, Sozialwesen, Wirtschaft

FAKULTÄT Informatik

BettercallxPaul

Masterarbeit

Coreference Resolution auf Deutsch mit externen Splits der Datensätze und deutschsprachiges word-level Modell

Daniel Schmuckermeier

Betreuer: Prof. Dr. Abdelmajid Khelil (HAW Landshut)
Betreuer: Clemens Gutknecht (BettercallPaul)
Betreuer: David Jenkins (BettercallPaul)

ERKLÄRUNG ZUR MASTERARBEIT

Schmuckermeier, Daniel

Hochschule Landshut
Fakultät Informatik

Hiermit erkläre ich, dass ich die Arbeit selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benützt, sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

14.10.2022
(Datum)

Schmuckermeier Daniel
(Unterschrift des Studierenden)

Abstract

Das Ziel dieser Arbeit ist es, die beiden deutschen state-of-the-art Modelle zur Coreference Resolution mit dem aktuellen state-of-the-art Modell (word-level) zu vergleichen. Ebenfalls soll untersucht werden, wie die Datensätze bestmöglich angepasst werden können, um die Performance der Modelle zu maximieren. Hierfür werden die Performance, die Trainingsdauer und der Peak-Speicherverbrauch experimentell ermittelt und miteinander verglichen. Die Datensätze werden vor dem Training in kleinere Dokumente gesplittet, um die Speicherproblematik zu umgehen. Ebenfalls haben vorherige Untersuchungen gezeigt, dass kürzere Dokumente besser für die Coreference Resolution geeignet sind. Zusätzlich werden verschiedene Kombinationen von Dokumentengrößen im train, dev und test Datensatz trainiert und der Einfluss von Sprecherinformationen untersucht. Als Ergebnis lässt sich festhalten, dass sowohl Performance als auch Speicherverbrauch von kürzeren Dokumenten stark profitieren. Bemerkenswert ist der Fakt, dass man die Trainingsdauer reduzieren kann, wenn man die Dokumentenlängen anpasst. Es ist davon auszugehen, dass die schlechtere Performance des deutschsprachigen word-level Modells darin begründet liegt, dass aufgrund der Domänen der Datensätze nicht für logisch abgeschlossene Dokumente garantiert werden kann.

Inhaltsverzeichnis

1 Einleitung

Virtuelle, digitale Assistenten nehmen immer mehr und mehr Einfluss in unserem Leben. Die Nutzung der Siris, Alexas und Co nimmt weltweit stark zu. 2015 gab es über 390 Millionen Nutzer. Sechs Jahre später werden es bereits viereinhalb Mal so viele sein (vgl. Richter 2016). Der große Vorteil dieser Assistenten ist, dass sie auf fast alle Fragen antworten, mit denen sie konfrontiert werden. Doch die allgemeinen Wissensassistenten unterliegen auch deutlichen Einschränkungen. Folgendes Beispiel veranschaulicht die Grenzen der gängigen Systeme:

- Wer ist unser Kanzler? Olaf Scholz
- Wer leitet unsere Vertriebsabteilung? 10 Tipps, um ein erstklassiges Vertriebsteam aufzubauen
- Wie ist der aktuelle Händlerbestand für den X253? United States Steel Corporation stieg gestern um 1,02 %

Sofort erkennt man, dass die erste Frage korrekt beantwortet wurde, die letzten beiden jedoch nicht. Das liegt daran, dass diese Assistenten Informationen für die Masse bereithalten. Möchte man unternehmensinterne Informationen erfahren, versagen sie. Hier zeigt sich, dass die gängigen allgemeinen Wissensassistenten auf öffentliche Daten zurückgreifen müssen und keine unternehmensspezifischen Antworten liefern können. Diesem Problem hat sich BettercallPaul mit ihrem Projekt XNEXT angenommen. XNEXT ist ein Frage-Antwort-System für Unternehmensdaten. Hierbei sollen mehrere Komponenten mithilfe eines trainierten Sprachmodells eine natürliche Frage (“Wo muss ich meine Reisekostenabrechnung einreichen? “) interpretieren und in unternehmensinternen Dokumenten nach passenden Antworten suchen (z. B. “reisekosten@bcxp.de“) (vgl. Gutknecht, Jenkins und L. Schröder 2022).

XNEXT ist eine Informationsextraktionspipeline. In Abbildung 1.1 wird ein einfacher, ähnlicher Modellaufbau dargestellt. Die Input-Daten sind unstrukturierter Text, z. B. Geschäftsunterlagen aus PDFs oder Text, der mithilfe Computer Vision aus Bildern extrahiert wurde. Die Coreference Resolution wandelt Pronomen in referenzierte Entitäten

um. So ist in einem Geschäftsbericht eine Passage “Johannes ist Leiter dieser Abteilung“ und etwas weiter im Geschäftsbericht “er leitet die Vertriebsabteilung“ vorzufinden. Nun kann man den Pronomen “er“ durch “Johannes“ ersetzen und hat die Information, wer die Abteilung leitet. Im nächsten Schritt geht es um Named Entity Linking. Je nach Domäne sind andere Entitäten von Bedeutung. In der Regel sind es Personen, Orte und Organisationen. Für Unternehmen gibt es weitere Entitäten, um eine interne Wissensdatenbank aufzubauen. So könnte der Begriff “Reisekosten“ direkt mit “reisekosten@bcxp.de“ gelinkt sein und passende Dokumente bereitstellen. Diese Entitäten müssen in der Regel selbst in einer Datenbank gepflegt werden. Im dritten Schritt wird versucht, Beziehungen zwischen den Entitäten zu erkennen. So sollte die Pipeline im Idealfall die Beziehung zwischen Johannes und der Vertriebsabteilung erkennen (vgl. Bratanic 2022).

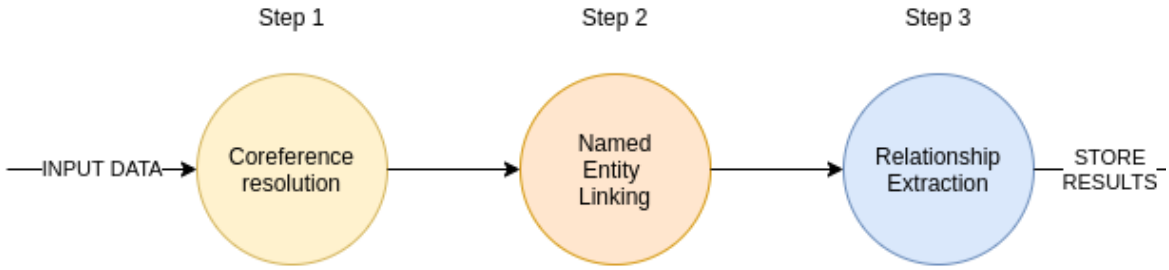

Abb. 1.1: Informationsextraktionspipeline (Bratanic 2022, gekürzt)

In dieser Arbeit wird der erste Schritt, die Coreference Resolution, genauer untersucht.

1.1 Motivation

Koreferenzen auflösen ist ein elementarer Schritt im Natural Language Understanding (NLU), um den Kontext von Texten maschinell besser erfassen zu können. Bei der Auflösung von Koreferenzen geht es um ein oder mehrere Ausdrücke in einem Dokument, die auf dieselbe Entität verweisen. Als Baustein größerer Natural Language Pipelines werden sie eingesetzt, um automatisiert Texte zu übersetzen, der Beantwortung von Fragen oder dem Erstellen von Zusammenfassungen. Durch diesen Baustein werden die Ergebnisse besser. Dazu ist es wichtig, dass dieser Baustein zeit- und speichereffizient arbeitet.

Explizit können durch das Auflösen von Koreferenzen Pronomen in ihre Entitäten aufgelöst werden. So wird “Er ist Gründer von SpaceX.“ aufgelöst zu “Elon Musk ist

Gründer von SpaceX.“ Ebenso können alle Informationen zu der Entität gesammelt und weiterverarbeitet werden. Man hätte für Elon Musk zum Beispiel automatisiert die weiteren Informationen “Gründer von SpaceX“, “Geburtsort: Pretoria, Südafrika“ oder “Körpergröße: 189 cm“.

Es lässt sich zeigen, dass ein greedy Ansatz zur Coreference Resolution eine Modellkomplexität von $O(n^4)$ aufweist, wobei n die Anzahl der Wörter im Dokument repräsentiert. Für ein 100 Wörter langes Dokument gibt es 100 Millionen mögliche Spannenpaare. Bei diesem Ansatz wird jede mögliche Spanne an Wörtern mit jeder möglichen anderen Spanne verglichen.

1.2 Ziel der Arbeit

Die beiden besten deutschsprachigen Modelle sollen mit dem aktuellen state-of-the-art Modell (word-level) in Bezug auf Speicherverbrauch, Trainingsdauer und der Performance verglichen werden (vgl. F. Schröder, Hatzel und Biemann 2021, vgl. Dobrovolskii 2021). Ebenfalls soll der Datensatz so optimiert werden, dass der Speicherverbrauch sinkt, die Performance möglichst erhalten bleibt oder verbessert wird. Auch der Einfluss von Metadaten wie Sprecher-Informationen wird analysiert. Eine Aussage über die Eigenschaften eines perfekten Datensatzes soll darüber hinaus getätigt werden können.

1.3 Struktur der Arbeit

Im Kapitel Grundlagen werden die wichtigsten Begriffe rund um Koreferenzen erklärt. Ebenso wird aufgezeigt, wie die hohe Modellkomplexität von $O(n^4)$ zustande kommt. Im Weiteren werden die technischen Grundlagen rund um Transformer erläutert und die Besonderheiten der verschiedenen Modelle dargestellt. In Methodik werden die Forschungsschwerpunkte dieser Arbeit erläutert, die verwendeten Datensätze beschrieben und die durchgeführten Experimente aufgezeigt. Daraufhin werden in der Evaluierung die Ergebnisse vorgestellt und anschließend diskutiert.

2 Grundlagen

In diesem Kapitel werden die verschiedenen Begrifflichkeiten zur Koreferenz und die grundlegende Theorie, auf der die Arbeit aufbaut, erläutert. Dies umfasst die Herleitung der Modellkomplexität bei einem greedy Ansatz, den Einfluss von Koreferenzauflösung in NLP-Modellen und die wichtigsten Bausteine werden ebenso in diesem Abschnitt erklärt: Transformer, Attention, BERT und ELECTRA. Zum Schluss wird der Algorithmus hinter Coreference Resolution vorgestellt und die Ansätze der drei unterschiedlichen Modelle dargestellt.

2.1 Definitionen der Koreferenzbegriffe

Der Begriff Koreferenz wird im *Collins Wörterbuch* folgend definiert: "a relationship between two words or phrases in which both refer to the same person or thing and one stands as a linguistic antecedent of the other, as the two pronouns in *She taught herself* but not in *She taught her*" (dt.: eine Beziehung zwischen zwei Wörtern oder Sätzen, in der sich beide auf dieselbe Person oder Sache beziehen und das eine als sprachliches Antezedens des anderen steht, wie die beiden Pronomen in *Sie hat es sich selbst beigebracht* aber nicht in *Sie hat es ihr beigebracht*). Coreference Resolution ist die Aufgabe, alle Ausdrücke in einem Text zu finden, dessen Erwähnungen sich zueinander auf dieselbe Entität beziehen, also koreferent sind. Weitere wichtige Begriffe sind Cluster, Anaphorik und Singleton-Entitäten. Alle Erwähnungen werden in Cluster aufgeteilt. Pro Entität existiert ein Cluster mit allen dazugehörigen Erwähnungen. Anaphorik ist der Gegenspieler zur Antezedens. Die Antezedens ist ein Ausdruck, der einem Pronomen seine Bedeutung gibt. Die Anaphorik dagegen ist ein Ausdruck, dessen Interpretation von einem vorangehenden Ausdruck abhängt. In der folgenden Abbildung ist *Sally* die Antezedens und gibt dem Pronomen *she* ihre Bedeutung. Die Interpretation der Anaphorik *she* ist im vorangehenden Teil des Satzes zu finden: *Sally*. Eine Singleton-Entität oder auch Nicht-Anaphorik bezeichnet, ist eine Erwähnung, die nur einmal im Text vorkommt, also keine koreferente Beziehung aufbauen kann (vgl. O'Brien 2019).

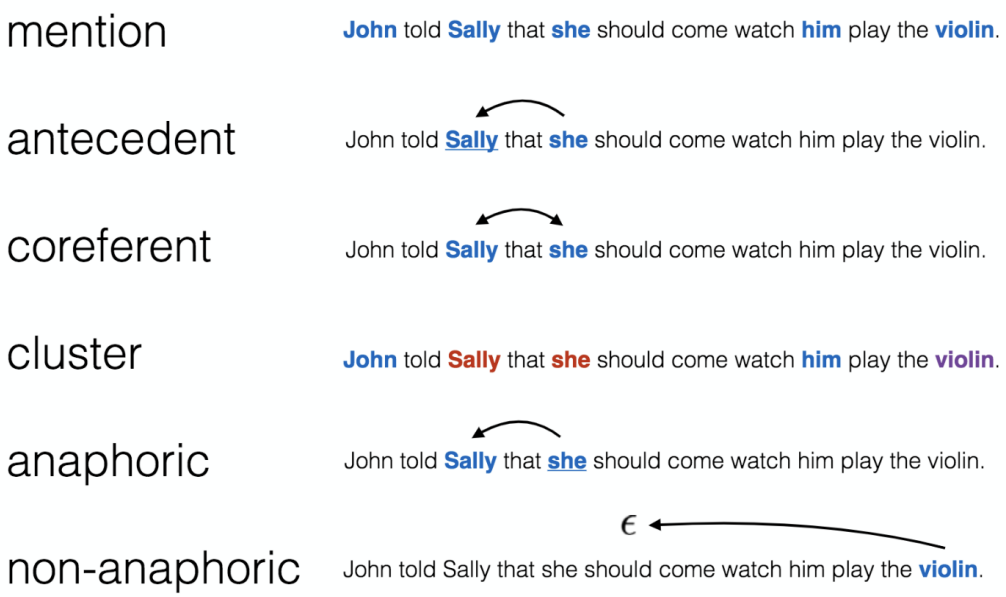

Abb. 2.1: Visualisierung der Definitionen (O'Brien 2019)

2.2 Der greedy Ansatz

Die Komplexität dieser Aufgabe wird im Folgenden bei einem greedy Ansatz erläutert. Es liegt ein Dokument mit n Wörtern vor. Diese Wörter müssen in jede mögliche Spanne aufgeteilt werden. Die erste Spanne ist das erste Wort, die zweite Spanne beinhaltet die ersten beiden Wörter. Dies wird so lange fortgesetzt, bis man beim letzten Wort ist. Dann bildet das zweite Wort alleine eine Spanne, die Wörter zwei und drei eine neue Spanne und das für alle Wörter. Die Gesamtzahl der Spannen ist genauso groß wie das Ergebnis der Gaußschen Summenformel $\frac{n^2+n}{2}$. Das resultiert zu $O(n^2)$ viele mögliche Spannen. Nachdem alle möglichen Spannen erzeugt wurden, müssen deren koreferenten Partner ermittelt werden. Dafür wird jede Spanne mit all seinen Vorgänger-Spannen verglichen. Die Anzahl der möglichen Beziehungen entspricht wieder dem Wert der Gaußschen Summenformel und somit einer Komplexität von $O(n^2)$. Insgesamt führt das zu einer Komplexität von $O(n^4)$. Daraus resultiert, wie viele mögliche Spannen existieren und multipliziert mit dem Abgleich jeder Spanne mit seinem potenziellen Partner. Ein kurzer Zeitungsartikel enthält ca. 100 Wörter. Daraus ergeben sich insgesamt 100 Millionen Spannenpaare, die verglichen werden müssen.

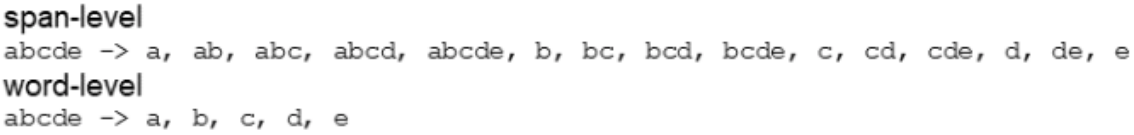

Abb. 2.2: Visualisierung der Spannenbildung beider Ansätze, wobei ein Buchstabe ein Wort repräsentiert

2.3 Einfluss der Koreferenzauflösung in NLP Aufgaben

Viele NLP-Aufgaben erfassen Attribute, Aktionen und Beziehungen zwischen den Entitätsclustern (2.1). Um alle Informationen über eine bestimmte Entität aufzudecken, müssen die Erwähnungen dieser Entität zusammen gefasst werden. Dadurch ist die Koreferenz eine wichtige Voraussetzung für Aufgaben wie textuelles Entailment und Informationsextraktion, um nur einige zu nennen. Textuelles Entailment beschreibt die Aufgabe, auf ein Statement eine automatisierte logische Schlussfolgerung zu tätigen. Zum Statement "Durch den Klimawandel schmilzt immer mehr Eis in der Arktis." wird beispielsweise vom Modell die Aussage "Die Arktis wird immer wärmer." erzeugt. Informationsextraktion dagegen ist der Prozess, aus unstrukturierten Daten (Dokumenten), die essenziellen Informationen in ein strukturiertes Format zu überführen. Dies kann in zweierlei Verwendung finden. Einmal für den menschlichen Betrachter, der in Google "Wer hat Apple gegründet? " sucht und die extrahierte Information "Steve Jobs, Ron Wayne, Steve Wozniak" in der Infobox mitgeteilt bekommt. Oder es ist Teil einer Pipeline, um Prozesse zu optimieren. Zum Beispiel werden Datenbanken mit fehlenden Informationen automatisiert aufgefüllt oder Texte durch die gewonnenen Informationen kategorisiert.

2012 wurden die NLP-Aufgaben Textzusammenfassung, textuelles Entailment und Textkategorisierung mit und ohne dem BART-Koreferenzauflösungssystem experimentell auf die Performance Unterschiede untersucht. BART (Beautiful Anaphora Resolution Toolkit) ist ein End-to-End System, welches nicht annotierten Text vorverarbeitet. Daraus werden Erwähnungen generiert, die semantische Informationen (Geschlecht, Nummer, etc.) beinhalten. Mittels maschinellem Lernen werden syntaktische und semantische Merkmale verwendet. Diese erzeugen Paarinstanzen (Anapher, Antezedens), die Koreferenzketten. BART bot damals nachweislich die beste Leistung mit einem F1-Score von 64 % (vgl. Versley u. a. 2008). Das Ergebnis war, dass es keine signifikante Verbesserung oder Verschlechterung gegenüber den Modellen ohne Coreference Resolution

gab. Überraschend war, dass Textzusammenfassungen bis zu 5 % vom Modell ohne Coreference Resolution nach unten abwichen. Die anderen beiden Aufgaben erzielten ähnlich gute Ergebnisse wie ihr Referenzmodell. Die Autoren mutmaßen für die fehlende positive Auswirkung der Koreferenzmodelle in der zu geringen Erfolgsrate des BART-Systems (vgl. Mitkov u. a. 2012).

Eine weitere experimentelle Untersuchung wurde 2017 veröffentlicht. Die TAC (Text Analysis Conference) Slot Filling Aufgabe, eine Informationsextraktionsaufgabe, wurde auf den Einfluss der Koreferenzen beleuchtet. Bei dieser Aufgabe werden Informationen über Personen, Organisationen oder geopolitische Einheiten aus großer Sammlung von Nachrichten, Web- und Diskussionsforen extrahiert. Der Ansatz ist es, die Wissensbasis zu erweitern, indem Beziehungen aus den Textdaten extrahiert werden. Dazu gibt es eine Abfragedatei, um die gewünschte Beziehung zu ermitteln. So muss für die Entität "Apple" und die Relation "org:founded by" das System "Steve Jobs, Ron Wayne, Steve Wozniak" zusammen mit passenden Textpassagen ausgeben. Es konnte bereits zuvor in Studien gezeigt werden, dass die zweithäufigste Fehlerquelle dieser Aufgabe auf Koreferenzfehler zurückzuführen ist (vgl. Min und Grishman 2012). Dies wurde in diesem Experiment erneut bestätigt. Das Modell mit der Coreference Resolution schnitt um sechs F1-Punkte besser ab als das Pendant ohne Coreference Resolution. Es wurden hierbei zwei Hauptaspekte für die Verbesserung identifiziert. 1) Pronomen und Namen können durch andere Namen/Pronomen ersetzt werden("Bill ist der Vater von Jane." Ersetzen von Jane "Bill ist ihr Vater."). 2) Zusätzliche Informationen über die Entitäten ("Die auf Hawaii Geborenen" – zusätzliche Information des Geburtsortes) (vgl. Adel und Schütze 2017).

Pink, Nothman und Curran (2014) untersuchten die Ursachen und den Einfluss für den Recall-Verlust bei der Slot Filling Aufgabe. Sie kommen zum Schluss, dass das Auflösen von Koreferenzen die Trefferquote erhöht, aber möglicherweise zu zeit- und ressourcenintensiv ist. Ebenfalls argumentieren sie, dass ohne die Coreference Resolution das Gesamtergebnis besser sein könnte, da die Coreference Resolution sich negativ auf die Präzision auswirken kann.

2.4 Aktueller Stand der Forschung

In diesem Abschnitt werden die technischen Grundlagen erläutert. Angefangen mit den Vorgängern RNNs und LSTMs Modellen und deren Nachteilen. Die Einführung der Transformer Architektur mit dem Attention-Layer folgt. Ebenso werden zwei konkrete

Transformer Modelle vorgestellt.

Der Weg zum Transformer – RNN und LSTM

Vor dem Transformer waren Recurrent Neural Networks (RNNs) die beste Möglichkeit Sequence-to-Sequence Daten zu trainieren. Ein Sequence-to-Sequence Modell bekommt als Eingabe eine Sequenz und gibt eine Sequenz zurück. Als Beispiel für einen Übersetzer könnte ein englischer Satz als Eingabe dienen und die deutsche Übersetzung als Ausgabe. Die RNNs sind sehr zeit- und ressourcenintensiv im Training. Dazu haben die RNNs Probleme bei langen Sequenzen, da es bei der Berechnung des Fehlergradienten zu verschwindenden oder explodierenden Gradienten kommen kann. Ein verschwundener Gradient bedeutet, dass das Modell ohne Anpassung der Gewichte weiter trainiert und ein explodierender Gradient führt zu Abbruch des Trainingsprozesses, da der Gradient den Wert NaN (not a number) erhält (vgl. Parikh 2021).

Um die beiden Probleme entgegenzuwirken, baut man Long-Short Term Memory (LSTM) Zellen ein. Durch ihre besondere Architektur können diese Zellen selbst entscheiden, ob es vorherige Informationen im Kurzzeitgedächtnis behält oder sie verwirft. Das ermöglicht es, längere Abhängigkeiten in Sequenzen zu erkennen. Die LSTM Zellen sind in der Berechnung komplexer als die normalen Zellen des RNN. Das führt dazu, dass das Trainieren eines Modells mit LSTM Zellen noch länger dauert.

Für RNNs und LSTMs müssen die Daten sequentiell verarbeitet werden, da immer Werte von vorherigen Zellen benötigt werden. Das führt dazu, dass heutige GPUs nicht sinnvoll verwendet werden können, da diese auf Parallelisierung ausgelegt sind. Die Frage ist nun: “Wie können wir die Parallelisierung für sequentielle Daten erreichen? “

Die Transformer Architektur und die Attention-Layer

2017 wurde die Transformer Architektur vorgestellt (vgl. Vaswani u. a.). Der Transformer ist eine Encoder-Decoder-Architektur, die ähnlich wie RNNs funktionieren. Anstatt nur Wort für Wort können Transformer ganze Sequenzen auf einmal trainieren.

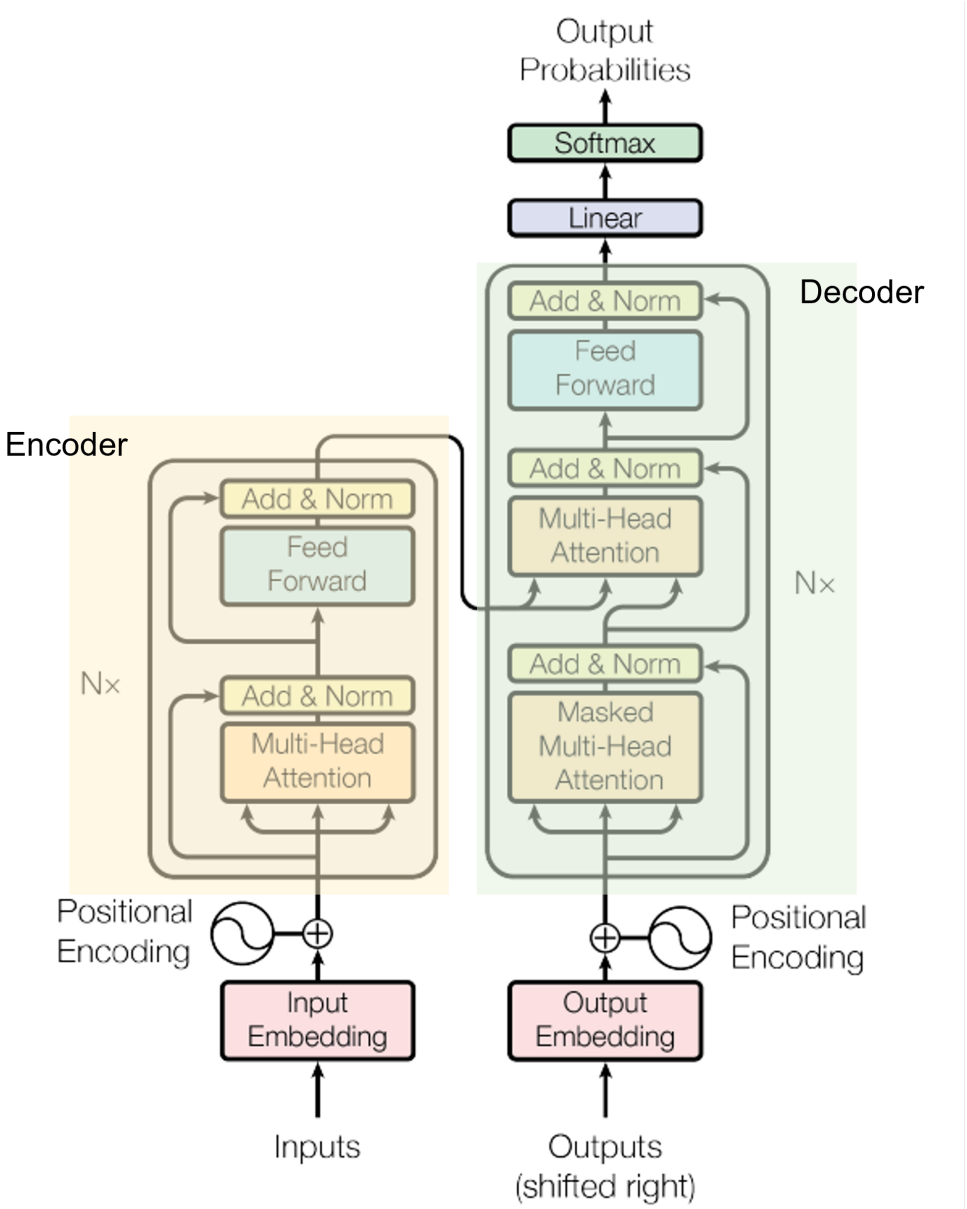

Abb. 2.3: Transformer – die Modell Architektur (Vaswani u. a. 2017, modifiziert)

Der Computer versteht keine Wörter, sondern nur Zahlen. So werden im ersten Schritt die Wörter anhand eines Embedding Space zu einem Vektor umgewandelt. Der Embedding Space ist eine hochdimensionale Abbildung aller Wörter. Alle Wörter mit ähnlicher Bedeutung befinden sich in der Abbildung nah an einem Ort. Ist die Bedeutung zweier Wörter unterschiedlich, werden diese weit auseinander liegen. Im nächsten Schritt wird dieser Vektor mit einem Positionsvektor kodiert, da die Position eines Wortes im Satz eine relevante Rolle spielt. Wir erhalten einen Vektor, bei dem der Wortvektor mit seiner

Information zur Position kodiert ist.

Diese Vektoren werden dem Encoder-Block übergeben. Zuerst kommt es zum mehrkopfigen Attention-Layer. Bevor die mehrkopfigen Attention-Layer behandelt werden, betrachten wir die Self-Attention. Die Aufgabe der Self-Attention ist es herauszufinden, wie relevant das ite Wort für die anderen Wörter im Satz ist. Wird ein englischer Satz ins Deutsche übersetzt, so ist die Self-Attention zu erkennen, da man ganz auf den Satz fokussiert ist. Für jedes Wort wird ein Attention-Vektor erstellt. In Abbildung 2.4 erkennt man, dass das ite Wort sich selbst die meiste Aufmerksamkeit schenkt. Dies ist nicht erwünscht, da die Beziehungen zu den anderen Wörtern interessanter ist. Dafür berechnen wir weitere Attention-Vektor mit anderen Gewichten. Zur Berechnung des endgültigen Attention-Vektors wird der gewichtete Durchschnitt für jedes Wort berechnet. Die Berechnung mehrere Attention-Vektoren auf denselben Satz ist die Aufgabe der mehrkopfigen Attention-Layer.

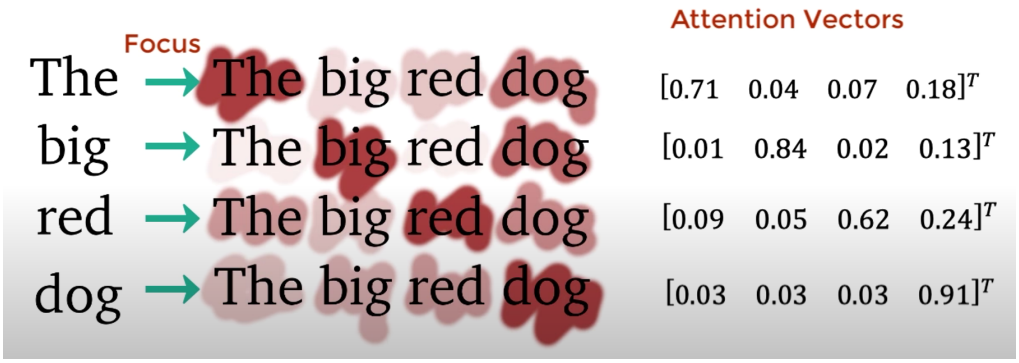

Abb. 2.4: Self-Attention (CodeEmporium 2020)

Die andere wichtige Einheit ist ein Feed-Forward-Netz. Dieses Netz wird auf jedem Attention-Vektor angewandt und verwandelt diese Vektoren in eine Form, damit diese für den nächsten Encoder-Block oder Decoder-Block weiterverarbeitet werden kann.

Als nächstes betrachten wir den Decoder-Block. Während der Trainingsphase wird der Decoder mit dem übersetzten deutschem Satz trainiert. Die ersten Schritte sind exakt gleich wie vorhin. Durch den Embedding Space wird das Wort in einen Wortvektor umgewandelt und mit Positionsinformationen ergänzt. Dann wird es dem Decoder-Block übergeben. Zwei der drei Schichten arbeiten so ähnlich wie im Encoder-Block. Mithilfe der Self-Attention wird jedes Wort im deutschen Satz auf die Relevanz zu den anderen Wörtern im Satz bewertet. Der daraus resultierende Attention-Vektor wird mit dem passenden englischen Attention-Vektor in einen neuen Attention-Vektor überführt. Der neue Encoder-Decoder-Attention-Vektor bestimmt, in welchem Verhältnis die ein-

zelnen Wortvektoren zueinander in Beziehung stehen. An dieser Stelle findet die Wortzuordnung zwischen Englisch und Deutsch statt. Jeder Wortvektor repräsentiert die Beziehungen zu den jeweils anderen Wörtern in den beiden Sprachen.

Wie im Encoder-Block kommt der neue Attention-Vektor in ein Feed-Forward-Netz und wird auf den nächsten Decoder Schritt vorbereitet oder geht zur linearen Schicht über. Die lineare Schicht dient dazu, den Vektor auf die Anzahl der deutschen Wörter zu erweitern. Die Softmax-Schicht verwandelt dies in eine Wahrscheinlichkeitsverteilung und gibt das nächste vorhergesagte Wort aus.

Die erste Schicht im Decoder-Block ist der Masked-Layer. Um das nächste deutsche Wort zu generieren, darf auf alle englischen Wörter zugegriffen werden, aber im deutschen Satz nur auf alle vorherigen Wörter im Satz. Könnte es auf alle Wörter zugreifen, dann gäbe es kein Lernen mehr, da es einfach das nächste deutsche Wort nimmt.

Der große Vorteil ist, dass jedes der Attention-Netze unabhängig voneinander ist. Das erlaubt uns alle Wörter gleichzeitig in den Encoder-Block zu geben und man erhält als Ausgabe eine Menge an kodierter Vektoren für jedes Wort.

Die Sprachmodelle BERT und ELECTRA

In den letzten Jahren haben Forscher aus dem Bereich der Computervision wiederholt den Wert des Transferlernens gezeigt. Das heißt: Ein Modell wird vortrainiert auf eine allgemeine Aufgabe, indem es zum Beispiel eine große Menge an Bilder bekommt. Anschließend wird dieses Modell feintrainiert. Dabei wird auf das Modell aufbauend eine neue zweckspezifische Aufgabe trainiert. So wird auf dem Modell die Aufgabe “erkenne Tiere“ feintrainiert. Das Feintraining benötigt nur einen Bruchteil der Zeit und Ressourcen als das erste allgemeine Modelltraining. Forscher zeigten, dass eine ähnliche Technik bei vielen Aufgaben in natürlicher Sprache sinnvoll sein kann (vgl. Horev 2018).

Bidirectional Encoder Representations from Transformers (kurz: BERT) ist ein Paper, das von Forschern bei Google AI Language veröffentlicht wurde (vgl. Devlin u. a. 2019). Es erregte großes Aufsehen, indem es Top-Ergebnisse in einer Vielzahl von NLP-Aufgaben erzielte (vgl. Devlin u. a. 2019, S.5 ff.).

BERT ist ein Sprachmodell der Transformer-Architektur. In der Abbildung 2.3 kann man eine physische Trennung erkennen. Es herrscht an dieser Stelle eine Aufgabenteilung vor. Bezogen auf das Beispiel oben lernt der Encoder die englische Sprache, deren Grammatik und noch wichtiger: den Kontext der Sätze. Der Decoder lernt, wie man die englischen Wörter auf die deutschen Wörter übertragen kann. Beide Blöcke lernen ein Verständnis von Sprache. Deshalb können wir den Encoder vom Decoder trennen und

erhalten BERT. Das E in BERT steht für Encoder.

Aufgrund des sequentiellen Worts für Wort lernen, können RNNs und LSTMs in eine Richtung lernen (von links nach rechts oder von rechts nach links) oder bidirektional (von links nach rechts und rechts nach links). Dennoch wird immer nur ein Wort und nicht die gesamte Sequenz erfasst. Im Gegenzug kann der Transformer-Encoder die gesamte Wortfolge auf einmal erfassen. Daher spricht man hier ebenfalls von bidirektionalem Lernen, wobei der Ausdruck "nicht gerichtet" besser passen würde. Jedes Wort der Wortfolge wird gleichzeitig trainiert. Diese Eigenschaft erlaubt es dem Modell, den Kontext eines Wortes basierend auf seiner gesamten Umgebung zu lernen.

Die Herausforderung beim Training von Sprachmodellen besteht darin, ein Vorhersageziel zu definieren. Viele Modelle sagen das nächste Wort einer Folge hervor, was wiederum ein gerichteter Ansatz wäre. Dies schadet dem Kontextlernen. BERT verwendet hierbei die Masked-Language Modeling (MLM) Methode. Bevor die Sequenzen in BERT eingespeist werden, werden 15 % der Wörter durch ein [MASK]-Token ersetzt. Das sieht so aus: *["the", "chef", "cooked", "the", "meal"]* wird zu *["[MASK]", "chef", "[MASK]", "the", "meal"]* Hierbei versucht das Modell, basierend auf dem Kontext der umliegenden Wörter, das maskierte Wort vorherzusagen.

Es gibt mittlerweile einige Modelle, die sich der Transformer-Architektur zunutze machen, so RoBERTa, XLNet oder GPT-3. Der große Erfolg wird dem MLM zugeschrieben. Für das Training dieser Modelle ist man auf riesige Datenmengen angewiesen. Die Rechenkosten sind ebenso enorm. So schätzt man, dass für das Training eines GPT-3 Modells 4,6 Millionen US-Dollar benötigt werden (vgl. Briggs 2021).

Mit Efficiently Learning an Encoder that Classifies Token Replacements Accurately oder in kurz ELECTRA wurde ein neuer erfolgreicher Ansatz vorgestellt (vgl. Clark u. a. 2020). Anstatt wie BERT eine zufällige Auswahl an Wörter zu maskieren, wird hierbei ein weiteres neuronales Netzwerk verwendet. Dieses Netzwerk hat die Aufgabe, unser Modell zu täuschen, indem es zufällige Wörter durch gefälschte Wörter ersetzt. So ähnlich wie Generative Adversarial Netzwerke (GAN), wo zwei Netzwerke gegeneinander arbeiten. Der Generator ist darauf optimiert, den Diskriminator mit zunehmend überzeugenden, dennoch gefälschten Daten in die Irre führen. Der Diskriminator muss feststellen, welche Daten wahr oder gefälscht sind.

Der Ansatz von ELECTRA verfolgt einen ähnlichen Ansatz wie den von GANs. ELECTRA nimmt die Rolle des Diskriminator ein und muss feststellen, welche Wörter echt und welche vom Generator gefälscht sind. Es ist deshalb kein GAN, da der Generator während dem Training nicht optimiert wird, um die Fehlerrate des Diskriminator

zu steigern. Der Generator wird wie ein typisches MLM-Modell trainiert, indem er die [MASK]-Werte erraten muss. Dieser Ansatz hat den Vorteil, dass das Modell jedes einzelne Wort, in jedem Durchlauf, berücksichtigen muss, während MLM nur die Fokussierung auf das [MASK]-Token erfordert. Durch das Konzentrieren auf jedes Wort erzeugt das Modell ein besseres Verständnis des Kontexts.

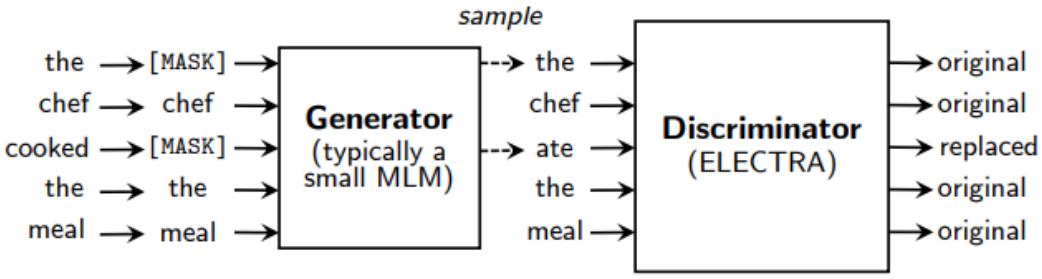

Abb. 2.5: Der Trainingsansatz von ELECTRA (Clark u. a. 2020)

Nach dem Training kann der Generator verworfen werden und mit dem Feintraining begonnen werden. ELECTRA ist kleiner als BERT und konnte nachweislich in den meisten Disziplinen, bessere Ergebnisse bei weniger Rechenleistung erzielen (vgl. Clark u. a. 2020, S.4 ff.).

Nun wurden beide Sprachmodelle vorgestellt, aber das Feintraining fehlt noch. Das kann so für beide Sprachmodelle umgesetzt werden. Die letzte Schicht im Modell wird angepasst und das befähigt es unterschiedliche Sprachaufgaben zu lösen. Zum Beispiel bei der Frage-Antwort Aufgabe bekommt das Modell eine Sequenz, die Frage, und muss die passende Antwort in der Sequenz markieren. Dies erreicht man, indem man zwei weitere Vektoren trainiert, die den Anfang und das Ende der Antwort repräsentieren.

2.5 Modelle

In diesem Kapitel werden das Coarse-to-fine (c2f), das Incremental (increm) (beide vgl. F. Schröder, Hatzel und Biemann 2021) und das word-level (wl) Modell (vgl. Dobrovolskii 2021) vorgestellt. Die ersten beiden sind state-of-the-art im deutschen Bereich. Das letzterer ist das aktuelle state-of-the-art Modell und erreichte mit dem englischen OntoNotes Datensatz (vgl. Pradhan u. a. 2012) einen F1-Score von 81. Neben den Besonderheiten der einzelnen Modelle wird auch der Prozess der Coreference Resolution genauer betrachtet.

Prozess der Coreference Resolution

Der Prozess der Coreference Resolution ist bei jedem der drei Modelle gleich. Die Unterschiede und Besonderheiten der Modelle werden im Abschnitt Beschreibung der drei Modelle erläutert.

Für jede Spanne i lernt das Modell eine bedingte Wahrscheinlichkeitsverteilung $P(y_i|D)$ über die vorherigen Spannen y aus dem Dokument D, deren wahrscheinlichste Konfiguration das korrekte Clustering ergibt.

$$P(y_i) = \frac{e^{s(i,j)}}{\sum_{y' \in Y} e^{s(i,j')}} \tag{2.1}$$

Hierbei stellt die Gleichung 2.2 einen paarweisen Coreference Score zwischen den Spannen i und j dar. Diese Gleichung lässt sich in drei Faktoren aufteilen: (1) ist Spanne i eine Erwähnung, (2) ist Spanne j eine Erwähnung und (3) ist j eine Antezedenz von i. Der dritte Faktor gibt an, wie wahrscheinlich es ist unter der Annahme, dass beide Spannen Erwähnungen sind, dass sie sich auf dieselbe Entität beziehen. Zusätzlich gibt es ein Dummy-Antezedenz ϵ mit einem festen Score $s(i, \epsilon) = 0$ für alle i. Dadurch, dass der Coreference Score mit der Dummy-Antezedenz auf 0 festgelegt wird, sagt das Modell das am besten bewerte Antezedenz voraus, wenn die Nicht-Dummy-Antezedenz einen positiven Score aufweist. Aufgrund der Modellkomplexität von $O(n^4)$ und der Abhängigkeit der Dokumentenlänge werden die Spannen beschnitten, die gemäß der Erwähnungsbewertung $s_m(i)$ unwahrscheinlich zu einem Koreferenzcluster gehören. Hier ist g_i die Vektordarstellung der Spanne i und $\phi(i, j)$ ein Vektor von paarweisen Metadaten Merkmalen. Diese Merkmale können Sprecherinformationen, das Genre des Dokuments, die Distanz zwischen der beiden Spannen, usw. sein. FFNN stellt ein standardmäßiges neuronale Feed-Forward-Netz dar, das eine nichtlineare Abbildung von Eingabe- auf Ausgabevektoren berechnet (vgl. Lee u. a. 2017, vgl. Joshi u. a. 2019).

$$s(i, j) = s_m(i) + s_m(j) + s_a(i, j) \tag{2.2}$$

$$s_m(i) = \text{FFNN}_m\left(\mathbf{g_i}\right) \tag{2.3}$$

$$s_a(i, j) = \text{FFNN}_a\left(\left[\mathbf{g_i}, \mathbf{g_j}, \mathbf{g_i} \circ \mathbf{g_j}, \phi(i, j)\right]\right) \tag{2.4}$$

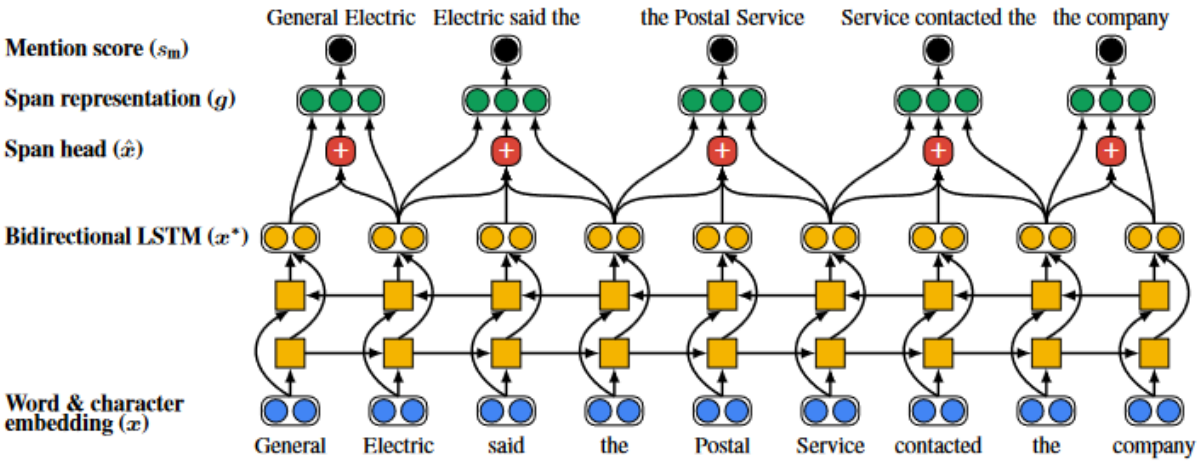

Abb. 2.6: Koreferenzauflösungsprozess Teil 1 - die LSTM werden durch die Transformer BERT oder ELECTRA ersetzt (Lee u. a. 2017)

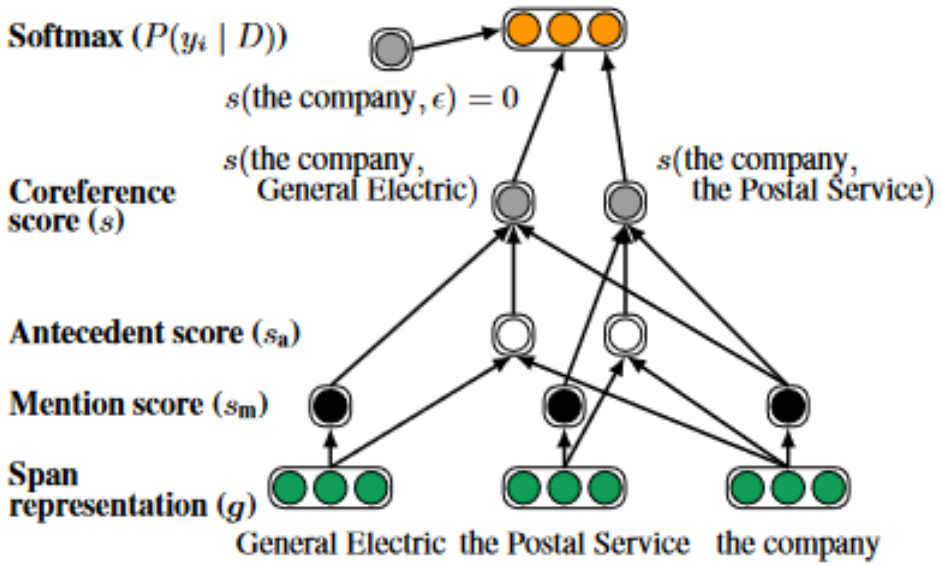

Abb. 2.7: Koreferenzauflösungsprozess Teil 2 (Lee u. a. 2017)

In der Abbildung 2.6 werden die ersten Schritte des Prozesses anhand des Satzes *General Electric said the Postal Service contacted the company* aufgezeigt. Zuerst wird jedes einzelne Wort in einem Embedding-Schritt zu einem Wortvektor transformiert. Daraufhin werden die Wortvektoren in den Transformer gespeist. Dadurch erhält man für jedes Wort einen Attention-Vektor. Die werden dazu verwendet, die Spannen zu bilden und die Spannenrepräsentation g zu berechnen. Im nächsten Schritt wird für

jede Spanne die Erwähnungsbewertung s_m durchgeführt. Jede Spanne, die nicht als Erwähnung klassifiziert wird, wird beschnitten. In Abbildung 2.7 bemerkt man, dass aus den ursprünglichen fünf Spannen nur drei als Erwähnung klassifiziert werden. Nun wird für die Spanne *the company* geprüft, ob eine der anderen beiden Spannen zu *the company* koreferent ist. Dazu wird mit jeder Vorgängerspanne der Antezedenz-Score s_a berechnet und das Ergebnis des Antezedenz-Scores zusammen mit dem Score aus der Erwähnungsbewertung, der Coreference Score s ermittelt. Zu guter Letzt wird mithilfe des Dummy-Antezedenz ϵ und allen Scores aus der Koreferenzbewertung, die bedingte Wahrscheinlichkeitsverteilung $P(y_i|D)$ ermittelt. Diese wird mit einem Softmax auf einen Wertebereich zwischen Null und Eins normalisiert.

Beschreibung der drei Modelle

Jetzt werden die Besonderheiten der drei Modelle besprochen. Alle Modelle haben dieselben Grundschritte, die man auch an der Abbildung 2.6 sehr gut erkennen kann:

1. Zuallererst findet ein Embedding der Wörter statt und
2. dann werden die neuen Wortvektoren in den Transformer gespeist.
3. Danach werden immer die Spannenrepräsentation gebildet.
4. Darauf folgt der Pairwise-Schritt. Dort werden die Metadatenmerkmale verarbeitet und
5. letztendlich wird der endgültige Score berechnet. Dieser Score ist die bedingte Wahrscheinlichkeitsverteilung von 2.1 für eine Spanne i über alle möglichen Vorgängerspannen. Diese Wahrscheinlichkeitsverteilung wird mit einem Softmax auf einen Wert zwischen Null und Eins normalisiert. So kann der Spanne i eine Entität zugeordnet werden.

Das Coarse-to-fine Modell besticht mit zwei Techniken. Zuallererst werden nur Spannen mit einer maximalen Anzahl an Wörter berücksichtigt. Laut einer Studie haben die Spannen von Koreferenzen im Durchschnitt weniger als fünf Wörter (vgl. Kunz, Lapshinova-Koltunski und Martínez 2016). Die Berechnung der Erwähnungsbewertung wird bei jedem möglichen Spannenpaar angewandt. So besteht die zweite Technik darin, alle Spannen zu beschneiden, die anhand der Erwähnungsbewertung s_m keine Erwähnungen repräsentieren. Um die Anzahl der zu berücksichtigen Spannen weiter zu reduzieren, werden nur die besten k Antezedenzien für jede Erwähnung gespeichert.

Die beiden Techniken machen es erst möglich, größere Dokumente zu verarbeiten. Sie sind elementar wichtig, sodass sie beim Incremental Modell zum Einsatz kommen und beim word-level Modell in abgeschwächter Form.

Beim Incremental Modell teilt man größere Dokumente in kleinere Segmente auf und/oder nur die aktuell relevantesten Erwähnungen werden im Speicher behalten. Dies soll dem Speicherproblem zugutekommen. Der Ansatz basiert auf der Frage: Ist es notwendig, eine unbegrenzte Anzahl von Erwähnungen zu speichern? Psycholinguistische Erkenntnisse deuten darauf hin, dass die menschliche Sprachverarbeitung inkrementell erfolgt und das Arbeitsgedächtnis begrenzt ist (vgl. Tanenhaus u. a. 1995, vgl. Keller 2010, vgl. Baddeley 1986). Die Praxis zeigt, dass die Anzahl der Wörter von der ersten bis zur letzten Erwähnung einer Entität eine geringe Streuung aufweist. Die Streuungen betragen bei fast allen Entitäten einen Dokumentenbereich von ca. 2000 bis 5000 Wörter. Das heißt, dass nicht alle Erwähnungen ständig im Speicher gehalten werden müssen (vgl. Toshniwal u. a. 2020).

Das word-level Modell setzt nicht auf mehrere Wörter pro Spanne, sondern auf einzelne Wörter. Im letzten Schritt wird bei allen möglichen Koreferenz-Kanditaten auf eine mögliche mehrwortige Spanne rekonstruiert. Dies gelingt mit einem weiteren neuronalen Feed-Forward-Netzwerk und einem Faltungsblock mit zwei Ausgängen. Diese Ausgänge sollen die besten Start und Endpunkte vorhersagen. Der Unterschied zum Coarse-to-fine Modell, hinsichtlich der Beschneidung bei der Erwähnungsbewertung ist, dass für alle Spannen nur die k besten Antezedenzien gespeichert werden. Es findet keine Beschneidung anhand der Erwähnungsbewertung statt.

3 Problemstellung, Methodik und Benchmarking

Der folgende Part beinhaltet die Forschungsschwerpunkte, die verwendeten Datensätze und die Experimente. Ebenso wird aufgezeigt, wie das englische word-level Modell für die deutsche Sprache angepasst wurde.

3.1 Forschungsschwerpunkte

Die Ergebnisse der deutschen state-of-the-art Modelle sollen mit dem word-level Modell, dem aktuellen state-of-the-art, in deutscher Sprache verglichen werden. Ebenso sollen Anpassungen am Datensatz untersucht werden. Es wurde mehrfach festgestellt, dass kürzere Dokumente für ein besseres Modell-Ergebnis sorgen (vgl. Krug 2020, vgl. Joshi u. a. 2019, vgl. F. Schröder, Hatzel und Biemann 2021). Dies kann so interpretiert werden, dass die inhärente Schwierigkeit der Koreferenzaufgabe mit zunehmender Dokumentenlänge wächst. Die Länge, die sich in der Praxis als optimal erweist, liegt bei ca. 500 Wörtern. Für das Coarse-to-fine Modell und das Incremental Modell existiert eine Split-Option. Hierbei wird das Dokument während dem Training in kleinere Segmente unterteilt. Haben alle Segmente eines Dokuments das Training durchlaufen, werden die Ergebnisse zusammengetragen. Danach erfolgt der Recall-Prozess, indem die Gewichte im Modell angepasst werden. Einer der verwendeten Datensätze (3.1) hat im Schnitt über 4.000 Wörter bei nur 90 Dokumenten. Um die Performance zu maximieren, sollen die Dokumente in kleinere Segmente unterteilt werden. Für diese Arbeit fungieren diese Segmente als eigenständige Dokumente. So werden aus 90 4.000er-Dokumente, 720 500er-Dokumente. Dadurch wirkt man der Schwierigkeit der Koreferenzaufgabe entgegen und profitiert vom regelmäßigen Einsatz des Recalls. Des Weiteren soll der Einfluss der Sprecherinformationen auf die verschiedene Datensatz-Aufteilungen untersucht werden.

Dazu werden als Vergleichsgrößen, die Performance mit dem F1-Score, die Trainings-

dauer in Minuten und der Peak-Speicherverbrauch der GPU ermittelt. Der F1-Score repräsentiert die Trefferquote des Modells und zeigt, wie gut das Modell die Koreferenzen erkennt. Dabei werden True Positives als auch False Positives und False Negatives betrachtet. Die Messung der Trainingszeit ist von Bedeutung, inwiefern sich die verschiedenen gesplitteten Datensätze sich auf die Trainingszeit auswirken. Da der Peak-Speicherverbrauch stark mit der Anzahl der Wörter im Dokument korreliert, ist das Verhalten bei gesplitteten Datensätzen ebenso interessant. Vor allem da der Speicher ein großes Problem darstellt und der Grund ist, weshalb man Inkrementelle Modelle entwickelt hat.

3.2 Datensätze

An dieser Stelle werden die verschiedenen Datensätze und ihre Besonderheiten betrachtet. Der Discourse Information Radio News Database for Linguistic Analysis oder kurz DIRNDL-Datensatz besteht aus stündlich gesendeten deutschen Radionachrichten (vgl. Roesiger und Riester 2015). Dieser Datensatz unterscheidet sich zu den anderen Datensätzen, dass gesprochene Nachrichten als Text transkribiert wurden. Die ursprüngliche Form war demnach eine gesprochene Variante. Dies unterscheidet sich zu rein textuell verfassten Inhalten, die man bereits an Versprechern oder den Füllwörtern zum Beispiel “ähm“ erkennen kann. Ebenfalls wurde der Datensatz mit lebendigen und toten Koreferenzen annotiert. Lebendige Koreferenzen sind welche, die sich auf lebendige Entitäten, wie Menschen oder Tiere beziehen. Tote Koreferenzen sind dagegen alle anderen Koreferenzen. Zu den toten Koreferenzen zählen Ausdrücke wie “diese Nacht“ oder “das Feuer“. Toten Koreferenzen sind deutlich schwieriger als Erwähnung zu erkennen und stellen eine weitere Herausforderung dar.

Ein weiterer Datensatz ist der Deutsche Roman Corpus (DROC) (vgl. Krug 2018). Dort sind 90 zufällig ausgewählte und manuell annotierte Fragmente deutschsprachiger Romane aus dem 17. bis ins 20. Jahrhundert vorzufinden. Hierbei wurden schwerpunktmäßig Figurenreferenzen annotiert. Dies fällt unter lebendige Koreferenzen.

Der dritte und letzte Datensatz kommt aus einer ähnlichen Domäne wie DROC. GerDraCor steht für German Drama Corpus for Coreference und umfasst etwa 40 Akte aus 30 deutschen Dramen/Theaterstücke (vgl. Pagel und Reiter 2014). Die Zeitspanne der Dramen/Theaterstücke erstreckt sich von 1730 bis 1920. Zu jedem Dokument wurden auch die Sprecherinformationen mit aufgeführt.

In der Abbildung 3.1 wird ein exemplarischer Auszug aus dem DROC-Datensatz

gezeigt. Er beinhaltet zwei Sätze, die durch eine Leerzeile voneinander getrennt werden. Die Sätze sind “Marie, meine geliebte Marie! “ und “Du bist sehr krank! “. Dieses Beispiel-Dokument beinhaltet zwei Cluster, einmal die Nummer sechs und einmal die Nummer 25. Somit sind die drei Spannen mit der sechs koreferent zueinander und beziehen sich auf dieselbe Entität. Die 25 stellt in diesem Fall eine Singleton-Entität dar, da sie keine Beziehung zu einer anderen passenden Erwähnung eingehen kann. Solche Singletons werden in den Datensätzen nicht annotiert. Bei der Aufgabe der Coreference Resolution können Singletons nicht mitwirken, da sie keine Koreferenzen darstellen.

```
#begin document Ahlefeld,-Charlotte-von__Marie_Müller.00058
Ahlefeld,-Charlotte-von__Marie_Müller.00058  24  1  Marie  -  -  Marie  -  -  -  -  -  -  (6)
Ahlefeld,-Charlotte-von__Marie_Müller.00058  24  2  ,      -  -  ,      -  -  -  -  -  -
Ahlefeld,-Charlotte-von__Marie_Müller.00058  24  3  meine  -  -  mein   -  -  -  -  -  -  (25)
Ahlefeld,-Charlotte-von__Marie_Müller.00058  24  4  geliebte -  -  geliebt -  -  -  -  -  -  -
Ahlefeld,-Charlotte-von__Marie_Müller.00058  24  5  Marie  -  -  Marie  -  -  -  -  -  -  (6)
Ahlefeld,-Charlotte-von__Marie_Müller.00058  24  6  !      -  -  !      -  -  -  -  -  -

Ahlefeld,-Charlotte-von__Marie_Müller.00058  25  1  Du  -  -  du  -  -  -  -  -  -  (6)
Ahlefeld,-Charlotte-von__Marie_Müller.00058  25  2  bist -  -  sein -  -  -  -  -  -
Ahlefeld,-Charlotte-von__Marie_Müller.00058  25  3  sehr  -  -  sehr  -  -  -  -  -  -  -
Ahlefeld,-Charlotte-von__Marie_Müller.00058  25  4  krank -  -  krank -  -  -  -  -  -  -
Ahlefeld,-Charlotte-von__Marie_Müller.00058  25  5  !     -  -  !     -  -  -  -  -  -
#end document
```

Abb. 3.1: Ausschnitt eines Dokuments aus dem DROC-Datensatz im conll-Format mit zwei Sätzen. (Krug 2018, gekürzt)

Daten	DIRNDL	DROC	GerDraCor
Anz. der Dokumente	382	90	526
Anz. der Wörter	31.942	393.164	790.249
Anz. der Cluster	1.176	5.275	24.753
Anz. der Koreferenzen	2.828	51.797	173.371
Wörter pro Dokument	83,62	4.368,49	1.502,37
Maximum an Wörter	216	15.718	10.022

Tab. 3.1: Ein Überblick über die Datensätze

3.3 Experimente

Hier werden die verschiedenen Experimente, die im Zuge dieser Masterarbeit untersucht werden, erläutert. Das erste Experiment ist das Benchmark-Experiment. Dort

werden die Datensätze in ursprünglicher Form mit den zwei verschiedenen Transformer, einen mit dem BERT-Ansatz und einen mit dem ELECTRA-Ansatz, trainiert. Die Datensätze werden in einem 60-20-20 Verhältnis in train-, dev- und test-Datensätze aufgeteilt. Die Ergebnisse aus diesem Experiment dienen als Benchmark, an denen die anderen Konstellationen bewertet werden.

Das nächste Experiment ist, die Dokumente in verschiedene Dokumentengrößen zu splitten. Hierbei sollen die Dokumente im Vorfeld, nicht während dem Training, auf eine maximale Dokumentenlänge aufgeteilt werden. Dafür soll in verschiedenen Split-Größen der Effekt der Performance, der Trainingsdauer und dem Speicherverbrauch untersucht werden. Auch das interne Splitten, welches die Modelle Coarse-to-fine und Incremental innehatten, wird zu Vergleichszwecken mittrainiert. Das Splitten läuft bei beiden Verfahren gleich ab. Nach einer bestimmten Anzahl an Wörtern (intern 512) wird nach dem nächsten Satzende gesplittet. Dies kann man, wie in Abbildung 3.1 gut zu sehen ist, an der Leerzeile im Dokument festmachen. Dies führt unweigerlich dazu, dass es Dokumente gibt, die deutlich kleiner sind als die zuvor definierte Split-Größe. Das liegt daran, dass es immer einen Rest gibt. Bei einem Dokument der Länge 600 würde man nach dem Splitten bei Split-Größe von 500, ein 500er und ein 100er-Dokument erhalten. Ebenso können einzelne Dokumente nach dem Split dennoch länger ausfallen. Das liegt daran, dass kein hard-cut stattfindet, sondern bis zum nächsten Satzende, mit dem Split, abgewartet wird.

Das dritte Experiment soll die Frage beantworten, ob ein Modell, welches mit einer kleinen Dokumentenlänge trainiert wurde, dennoch gute Ergebnisse liefert, wenn die test-Dokumente ihre originale Länge beibehalten. Dafür sollen Modelle mit einem 500er-Split trainiert werden und mit der originalen Länge der test-Dokumente evaluiert werden. Um die Performance zu steigern, werden unterschiedliche dev-Dokumentenlängen trainiert. Hierbei stellt sich die Frage, ob die Ergebnisse besser werden, wenn die dev-Dokumente sich zwischen der train und test Dokumentenlängen befinden. Als Zwischenschritt, um das Modell bei der Evaluierung auf längere Dokumente vorzubereiten, aber dennoch während dem Training mit kürzeren Dokumenten, den Trainingsprozess unterstützen zu können.

Beim letzten Experiment wird der Einfluss der Sprecherinformation auf die Modelle untersucht. Hierfür werden die Modelle in verschiedene Split-Größen, mit und ohne Sprecherinformationen trainiert und der Einfluss dieser Metadaten überprüft.

In allen weiteren Experimenten werden die Modelle in ihrer bestmöglichen Konfiguration und ohne Änderung der Hyperparameter trainiert. Insgesamt werden 20 Epochen

auf einer Ampere A6000 GPU mit 45 GB Grafikspeicher trainiert. Es wird beim Coarse-to-fine Modell mit zwei verschiedenen Varianten trainiert: einmal mit und einmal ohne der internen Split-Funktion, die das Modell bereits innehat. So können die Unterschiede der beiden Split Varianten differenzierter betrachtet werden und der Einfluss des öfters auftretenden Recall-Prozess analysiert werden.

3.4 Das word-level Modell auf Deutsch

Um das word-level Modell für die deutsche Sprache anzupassen, muss es an zwei Stellen verändert werden. Einmal beim Erstellen der preprocessten Datensätze und einmal beim Modell selbst. Für die Datensätze wird in der originalen Implementierung der Stanford Parser verwendet. Dieser ist nicht mehr aktuell und durch das Tool CoreNLP von der Stanford Universität ersetzt worden. Das Tool CoreNLP unterstützt acht Sprachen: Arabisch, Chinesisch, Englisch, Französisch, Deutsch, Ungarisch, Italienisch und Spanisch (vgl. Manning u. a. 2014). Für die Rekonstruktion von koreferenten Wörtern zu Spannen werden zusätzliche Informationen benötigt. Diese Informationen erhält man durch das CoreNLP. Durch das Tool kann für jeden Satz ein Dependency Parsing durchgeführt werden. Hierbei wird für jedes Wort im Satz, die Abhängigkeit zu den anderen Wörtern aus diesem Satz bewertet (vgl. Sharma 2020).

Dem englischsprachigen word-level Modell liegt ein mit englischen Daten trainiertes Transformer-Modell zugrunde. Für BERT wurde das *bert-base-german-cased* Modell verwendet. Dieses Modell basiert auf über sechs GB an Wikipedia Daten. Zusätzlich wurden Daten von OpenLegalData (2,4 GB) und Nachrichtenartikel (3,6 GB) verwendet. Das gesamte Training dauerte mit einer einzelnen Cloud-TPU v2 über neun Tage (vgl. Chan u. a. 2019). Für ELECTRA wurde das Modell *electra-base-german-uncased* von der *GERMAN NLP Group* verwendet. Insgesamt wurden über 70 GB an Daten verarbeitet. Darunter Wikipedia-Datensätze und Cleaned Common Crawl Corpus 2019-09 German. Über sieben Tage dauerte das Training auf einer TPU V3-8 (vgl. May, Reißel und German-NLP-Group 2021).

4 Evaluierung

An dieser Stelle werden die Ergebnisse der Experimente vorgestellt und diskutiert. Zuerst werden die Ergebnisse vom Benchmark Experiment bewertet, anschließend die Ergebnisse mit dem externen Split in drei verschiedenen Split-Größen. Danach folgt das Experiment mit unterschiedlichen Datensatzlängen und beim letztes Experiment wird der Einfluss von Sprecherinformationen untersucht. Zum Schluss wird noch auf die Problematik mit dem deutschsprachigen word-level Modell eingegangen.

4.1 Benchmark

Beim ersten Experiment, dem Benchmark, ist sofort ersichtlich, dass der DIRNDL Datensatz im Vergleich zu den anderen Datensätzen sehr schlecht abschneidet. Die Gründe hierfür dürften sein, dass dieser Datensatz im Vergleich zum zweitkleinsten Datensatz mehr als 10x kleiner ist. Zusätzlich dürften die toten Koreferenzen einen negativen Einfluss haben. Das liegt daran, dass Wörter wie “Nacht“ oder “Feuer“ nur unzureichend als Erwähnung erkannt werden. Wie groß der Einfluss von toten Koreferenzen ist, kann letztendlich nicht abschließend geklärt werden.

Neben der schlechten Performance fällt auf, dass ELECTRA des Öfteren einen niedrigeren Peak-Speicherverbrauch hat, gleichzeitig aber auch immer einen besseren F1-Score aufweisen kann als BERT. Dies bestätigt die Erkenntnisse rund um ELECTRA. Bei den unterschiedlichen Modellen ist auffällig, dass das word-level Modell bei größeren Datensätzen nur halb so lange trainiert werden muss. Bezüglich des Speicherverbrauchs schneiden die Modelle besser ab, die intern gesplittet haben.

Benchmark	F1-Score	Zeit (min)	Speicher (GB)	F1-Score	Zeit (min)	Speicher (GB)
DROC	bert			electra		
c2f		error			error	
c2f (split)	44,57	24	18,1	57,82	24	17,63
increm		error			error	
wl	0,61	10	39,59	31,83	11	39,58
GerDraCor						
c2f		error		59,27	62	39,63
c2f (split)	55,01	61	17,31	59,81	55	17,34
increm		error		59,31	63	39,65
wl	20,26	32	37,43	41,07	31	35,00
DIRNDL						
c2f	0,00	5	2,97	0,00	5	2,99
c2f (split)	0,00	5	2,97	0,00	5	2,99
increm	0,00	5	2,97	0,00	5	2,99
wl	7,08	4	2,66	2,65	4	2,68

error steht für Cuda Error: Out of memory, der Speicher der GPU ist zu klein

Tab. 4.1: Benchmark aller Modelle mit originalen Datensätzen

4.2 Externer Split

Beim zweiten Experiment wurden der DROC und der GerDraCor Datensatz in bestmögliche 5000er, 2500er und 500er-Dokumentenlängen aufgeteilt. Es wurden dieselben Modellkonstellationen wie im Benchmark-Experiment trainiert, der einzige Unterschied liegt darin, dass nur ELECTRA verwendet wurde. Das Benchmark-Experiment hat gezeigt, dass ELECTRA immer eine bessere Performance geliefert hat.

In der Tabelle 4.2 sieht man sofort die steigende Dokumentenanzahl, je kleiner die Dokumente werden. Die Anzahl der Wörter nimmt bei kleiner werdenden Split-Größen ab. Das liegt daran, dass Dokumente entstehen, die keine Koreferenzen enthalten. Da sie keinen Mehrwert bieten, werden sie im Vornherein aussortiert. Das führt zu einem kleinen Informationsverlust. Beim DROC Datensatz verlieren wir bis zum 500er Split ca. 1000 Wörter an Informationen. Bei GerDraCor sogar ca. 15.000 Wörter. Die ebenfalls steigende Anzahl der Cluster liegt darin, dass die Cluster nicht global, sondern

dokumentenweise agieren. Das heißt, kommt in einem Dokument eine Person vor und dieselbe Person ebenfalls in einem anderen Dokument, sind es zwei verschiedene Entitäten und somit zwei Cluster. Das Dokument mit den meisten Wörtern im Datensatz liegt immer sehr nahe an der Split-Größe. Einzig bei den 500er-Versionen sind die Dokumente ein großes Stück länger als die vorgegebene Länge von 500 Wörtern. Das liegt daran, dass manche Dokumente sehr lange Sätze, mit über 100 Wörtern, beinhalten. Ebenfalls zu beachten ist es, dass die neuen Datensätze Singleton-Entitäten beinhalten. Das sind Entitäten, die einmal im Dokument vorkommen, aber keine koreferente Beziehung eingehen können, da die passende zweite Entität fehlt. Durch das Splitten ist es möglich, dass Singleton-Entitäten entstehen können. Die ursprüngliche Variante der beiden Datensätze beinhaltet keine annotierten Singleton-Entitäten.

Da es bei der Coreference Resolution um das Auflösen der Koreferenzen geht und weniger um das Erkennen von Singleton-Entitäten. Das lässt den Schluss zu, dass der Score etwas höher ausfällt, als er ursprünglich ist (vgl. Kubler und Zhekova 2011).

Split	DROC5000	DROC2500	DROC500	GDC5000	GDC2500	GDC500
Anz. der Dokumente	116	207	829	592	707	1.950
Anz. der Wörter	393.164	393.164	392.150	790.249	790.244	774.289
Anz. der Cluster	5.490	6.135	8.786	26.164	28.100	38.006
Anz. der Koreferenzen	51.797	51.796	51.795	173.371	173.371	173.368
Wörter pro Dokument	3.389,34	1.899,23	473,04	1.334,88	1.117,74	397.07
Maximum an Wörter	5.018	2.560	681	4.760	2.435	635

GDC steht für GerDraCor

Tab. 4.2: Datensätze nach dem Split

Split	F1-Score	Zeit (min)	Speicher (GB)	F1-Score	Zeit (min)	Speicher (GB)
	DROC5000			GDC5000		
c2f	58,56	23	18,71	60,83	53	17,57
c2f (split)	59,08	23	17,35	61,09	52	16,82
increm	58,96	23	18,68	60,79	52	17,70
wl	37,89	10	21,02	28,34	31	18,28
	DROC2500			GDC2500		
c2f	55,82	18	10,44	64,00	44	10,26
c2f (split)	61,99	18	10,44	63,50	44	10,20
increm	62,39	18	10,40	63,85	44	10,22
wl	47,74	10	11,12	17,90	27	10,51
	DROC500			GDC500		
c2f	69,27	21	4,83	67,39	36	4,77
c2f (split)	69,29	22	4,84	**68,13**	36	4,75
increm	**69,52**	22	4,83	67,69	28	4,77
wl	46,18	16	4,48	4,06	46	4,25

GDC steht für GerDraCor

Tab. 4.3: Performance der verschiedenen Modelle mit den Datensätzen, nachdem die Datensätze gesplittet wurden

In der Tabelle 4.3 werden die Modelle mit verschiedenen Split-Varianten vorgestellt. Wie zu erwarten, sinkt der Peak-Speicherverbrauch bis auf unter fünf Gigabyte. Interessant ist der Fakt, dass die Dauer des Trainings bei allen Modellen nicht länger als bei dem Benchmark dauert. Einzig die beiden word-level Modelle in der 500er-Version fallen aus dem Muster. Das liegt vermutlich am word-level Modell selbst. Die Erklärung für die Vermutung für das schlechte Abschneiden des word-level Modell folgt im Abschnitt: Problem des deutschsprachigen word-level Modell. Bei DROC2500 ist die Zeit um 22 % kürzer als bei der Benchmark. Wiederum hat DROC500 die Zeitwerte aus dem Benchmark. Bei GerDraCor ist dieser Effekt ebenfalls zu beobachten. GerDraCor500 spart gegenüber der Benchmark-Varianten um bis zu 56 Prozent an Trainingszeit ein. Vor allem stellt man beim GerDraCor Datensatz eine stetige Zeitreduzierung fest, je kleiner die Dokumentenlänge wird. Was die Performance des F1-Scores angeht, steigt sie sukzessive an, je kleiner die Dokumente werden. So bessert sich die Performance beim DROC Datensatz um über 11 Punkte auf 69,52 %. Ebenfalls steigt beim GerDraCor Datensatz der F1-Score um über 8 Punkte auf 68,13 %.

In Tabelle 4.4 wird nochmal der Unterschied zwischen dem internen und externen

Split aufgezeigt.

intern vs. extern	DROC			GerDraCor		
origin c2f (split)	57,82	24	17,63	59,81	55	17,34
500 c2f	69,27	21	4,83	67,39	36	4,77

Tab. 4.4: Vergleich zwischen dem internen Split und der externen Split-Variante, der Dokumentenaufteilung

Um den Einfluss der beiden verschiedenen Split-Variationen zu zeigen. In der ersten Zeile der Tabelle 4.4 haben wir den internen Split von 512 auf die Original-Dokumente. In der zweiten Zeile wurde die interne Split-Funktion deaktiviert und dafür die Dokumente auf 500 Wörter gesplittet. Man erkennt ganz klar, dass die zweite Variante besser abschneidet. Hat man, wie beim DROC Datensatz, wenige Dokumente, die aber sehr lange sind, kann man sie splitten. Dennoch darf man den Werten nicht blind folgen, da Singleton-Entitäten mit in die Auswertung flossen und somit den F1-Score positiv beeinflussen.

4.3 Unterschiedliche Längen der einzelnen Datensätze

Im dritten Experiment wurden die Modelle mit unterschiedlicher train, dev und test-Dokumentenlängen trainiert. Hierbei sollte die Frage beantwortet werden, inwiefern sich die Ergebnisse ändern, wenn man das Modell im 500er-Split trainiert, die test-Datensätze aber in ursprünglicher Länge belässt. Sollten die neu zu annotierenden Dokumente im Vorfeld gesplittet werden oder können sie ihre Originallänge beibehalten? Um diesen Sachverhalt zu lösen wurden in diesem Experiment, die Modelle mit dem 500er train-Datensatz trainiert und mit der originalen Dokumentenlänge beim test-Datensatz evaluiert. Um die Ergebnisse zu verbessern, wurde der dev-Datensatz in allen Split-Größen trainiert. Die Frage hierbei war: “Wird das Ergebnis besser, wenn das Modell mit einem dev-Datensatz trainiert wird, der zwischen der train und test Dokumentenlängen liegt? “ Trainiert wurden die Modelle mit ELECTRA und dem Coarse-to-fine mit internen Split.

Dev-Länge	F1-Score	Zeit (min)	Speicher (GB)	F1-Score	Zeit (min)	Speicher (GB)
	DROC			GerDraCor		
DevOrigin	57,23	17	17,20	58,50	37	10,92
Dev5000	57,35	15	6,59	58,85	23	5,01
Dev2500	57,50	21	4,83	58,52	30	4,75
Dev500	57,55	22	4,82	58,32	36	4,75

Tab. 4.5: Verschiedene Dokumentenlängen bei dev, wobei die Modelle immer mit 500er-Split trainiert wurden und mit originalem test Datensatz ausgewertet wurden

Beim vorletzten Experiment lässt sich festhalten, dass das Auswählen unterschiedlicher Datensatz Längen keinen positiven Effekt erzeugt. Im Gegenteil sind dies die schlechtesten Ergebnisse. Auffallend ist die große Schwankung der Trainingsdauer. Wodurch dies zustande kommt, kann keine Aussage getroffen werden.

4.4 Einfluss von Sprecherinformationen

Im letzten Experiment wurden der Einfluss von Sprecherinformationen auf das Modell untersucht, mit den verschiedenen Split-Varianten. Die Modelle wurden mit dem GerDraCor-Datensatz trainiert, einmal mit Sprecherinformationen und einmal ohne Sprecherinformationen. Wie im dritten Experiment wurden die Modelle mit ELECTRA und dem Coarse-to-fine mit internen Split trainiert.

Sprecher	F1-Score	F1-Score
	mit Sprecher	ohne Sprecher
origin	59,81	58,72
5000	61,09	59,33
2500	63,50	61,42
500	68,13	65,39

Tab. 4.6: Performance-Unterschiede der Modelle mit und ohne Sprecherinformationen

Bemerkenswert ist, dass jedes Modell mit Sprecherinformationen besser als sein Pendant ohne Sprecherinformationen abschneidet. Ebenfalls nimmt die Einflussnahme je kleiner die Dokumente werden stärker zu. In der ursprünglichen Version gibt es einen Performance-Anstieg von etwas mehr als einem Prozent zu verzeichnen. Bei der 500er-Variante sind es bereits fast drei Prozent.

4.5 Problem des deutschsprachigen word-level Modell

Für das word-level Modell werden zusätzliche Informationen benötigt, um die Rekonstruktionen der Spannen zu ermöglichen. Diese Informationen erhält man durch das Dependency Parsing. Hierbei geht es um die Analyse der grammatikalischen Struktur eines Satzes basierend auf den Abhängigkeiten zwischen den Wörtern in diesem Satz (vgl. Sharma 2020). Für diese Informationen gibt es das Tool CoreNLP, welches unter anderem für einen Satz, die entsprechenden Informationen liefert (vgl. Manning u. a. 2014). Aufgrund der speziellen Domänen aller Datensätze kann dieses Tool nicht alle Sätze korrekt verarbeiten. Das liegt einerseits an Versprechern und Dialekten, aber auch an Ausdrücken wie "Wasser-Fall", die vom Tool falsch verarbeitet werden. Um dennoch dieses neue Modell zu testen, werden die fehlerhaften Sätze entfernt. Das führt zu weiteren Informationsverlusten, die in 4.7 aufgezeigt werden. Neben den teils großen Verlusten darf eine weitere Sache nicht vergessen werden. Es sind Dokumente vorhanden, wo mittendrin Sätze fehlen. Das heißt, dass diese Dokumente an sich logisch nicht vollkommen abgeschlossen sind, weil mittendrin ganze Sätze fehlen. Beim Splitten der Dokumente kann es auch zu Informationsverlust kommen, im Gegensatz dazu ist aber jedes Dokument an sich vollkommen logisch abgeschlossen. Die Verluste sind ganze Dokumente und keine Sätze. Da beim Lernen von Transformer, die Umgebung der Wörter eine wichtige Rolle spielt, wird der Verlust von vielen Sätzen einen negativen Einfluss auf das Modell haben.

Anz. Wörter mit CoreNLP	origin	wl
DIRNDL	31.942	19.804
DROC	393.164	353.822
DROC5000	393.164	353.822
DROC2500	393.141	353.799
DROC500	392.150	352.883
GerDraCor	790.249	781.199
GerDraCor5000	790.249	781.199
GerDraCor2500	790.244	781.194
GerDraCor500	774.289	765.671

Tab. 4.7: Der zusätzliche Verlust beim word-level Modell, der aufgrund der speziellen Domänen der Datensätze und dem Tool CoreNLP nicht zu vermeiden ist

5 Fazit und Ausblick

Ziel dieser Arbeit war es, das englische state-of-the-art Modell mit den deutschen state-of-the-art Modellen zu vergleichen. Dazu wurde eine weitere Möglichkeit untersucht, wie man mit langen Dokumenten verfahren kann. Vor allem unter dem Gesichtspunkt, dass man nur wenige Dokumente insgesamt hat.

Das word-level Modell kann, was die Performance angeht, den beiden deutschen Modellen nicht das Wasser reichen. Der große Pluspunkt hiervon ist, dass eine deutliche Zeitreduzierung gegenüber den beiden anderen Modellen zu erkennen. Vermutet wird für das schlechte Abschneiden des word-level Modell, dass es am Zusammenspiel von speziellen Domänen der Datensätze (alte Literatur und mündliche Texte mit Dialekt) und dem Tool CoreNLP. Dieses Tool wird benötigt, um weitere Informationen zu erhalten, die es ermöglichen aus den bestmöglichen Wörtern die Spannen zu rekonstruieren. Hierbei müssen ganze Sätze verworfen werden, da die speziellen Domänen mit dem CoreNLP nicht harmonieren. Das führt dazu, dass die Dokumente an sich nicht mehr vollständig logisch abgeschlossen sind und das schadet dem Kontextverstehen der Modelle.

Das Splitten der Dokumente auf kürzere Dokumentenlängen hat gezeigt, dass das ein sinnvoller Ansatz ist, um die Speicherprobleme aufgrund der Dokumentenlänge in den Griff zu bekommen. Inwiefern die in dieser Arbeit vorgestellten Performance-Gewinne dem Splitten der Dokumente zuzuschreiben sind, kann nicht endgültig beantwortet werden, da sich Singleton-Entitäten mit in der Auswertung befanden. Auffallend ist die teils starke Reduzierung der Trainingszeit, ohne dass man viel Informationen durch das Splitten verloren hat. Das lässt den Schluss zu, bei gleichbleibender Anzahl der Wörter im Datensatz, sinkt die Trainingsdauer bei kürzeren Dokumenten gegenüber den längeren. Dies zeigt, dass zukünftige Datensätze, eher auf kürzere Dokumente setzen sollten und die Verwendung von Metadaten wie den Sprecherinformationen vom Vorteil sind.

Für die Zukunft wäre es im Bezug auf das word-level Modell ratsam, andere Tools oder Datensätze zu verwenden, um zu überprüfen, inwiefern sich die fehlenden Sätze mitten im Dokument auswirken. Für das Splitten der Dokumente wäre vor allem interessant,

den Einfluss der Singleton-Entitäten zu bewerten. Weiterhin wären andere Splitting-Strategien vorstellbar, mit dem Fokus auf möglichst wenig Informationsverlust oder dem Vermeiden von Singleton-Entitäten. Ebenso wäre eine Untersuchung bedeutsam, inwiefern die Rest-Dokumente, die nur wenige Wörter beinhalten, eine Rolle spielen. Diese Rest-Dokumente entstehen, durch das naive Splitten der Dokumente.

Literaturverzeichnis

Adel, Heike und Hinrich Schütze (Okt. 2017). „Impact of Coreference Resolution on Slot Filling“. In:

Baddeley, Alan D. (1986). *Working Memory*. en. Clarendon Press. ISBN: 978-0-19-852116-7.

Bratanic, Tomaz (Mai 2022). *Extract knowledge from text: End-to-end information extraction pipeline with spaCy and Neo4j*. de. URL: https://towardsdatascience.com/extract-knowledge-from-text-end-to-end-information-extraction-pipeline-with-spacy-and-neo4j-502b2b1e0754 (besucht am 29. 08. 2022).

Briggs, James (Sep. 2021). *ELECTRA is BERT — Supercharged*. de. URL: https://towardsdatascience.com/electra-is-bert-supercharged-b450246c4edb (besucht am 11. 09. 2022).

Chan, Branden u. a. (2019). *German BERT — State of the Art Language Model for German NLP*. de. URL: https://www.deepset.ai/german-bert (besucht am 01. 09. 2022).

Clark, Kevin u. a. (März 2020). „ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators“. en. In: URL: https://arxiv.org/abs/2003.10555v1 (besucht am 11. 09. 2022).

CodeEmporium (Jan. 2020). *Transformer Neural Networks - EXPLAINED! (Attention is all you need)*. URL: https://www.youtube.com/watch?v=TQQlZhbC5ps (besucht am 10. 09. 2022).

Collins Wörterbuch (2019). en. URL: https://www.collinsdictionary.com/de/worterbuch/englisch/coreference (besucht am 20. 08. 2022).

Devlin, Jacob u. a. (Juni 2019). „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding“. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, S. 4171–4186. URL: https://aclanthology.org/N19-1423 (besucht am 10. 09. 2022).

Dobrovolskii, Vladimir (Nov. 2021). „Word-Level Coreference Resolution“. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online und Punta Cana, Dominican Republic: Association for Computational Linguistics,

S. 7670–7675. URL: https://aclanthology.org/2021.emnlp-main.605 (besucht am 12. 08. 2022).

Gutknecht, Clemens, David Jenkins und Leon Schröder (Juni 2022). *BettercallPaul + Forsche Forschung – Ihre Ai-Assistenz für's Büro*. de-DE. URL: https://bcxp.de/forschungsprojekte-ihre-ai-assistenz-fuers-buero/ (besucht am 29. 08. 2022).

Horev, Rani (Nov. 2018). *BERT Explained: State of the art language model for NLP*. de. URL: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270 (besucht am 10. 09. 2022).

Joshi, Mandar u. a. (Nov. 2019). „BERT for Coreference Resolution: Baselines and Analysis". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, S. 5803–5808. URL: https://aclanthology.org/D19-1588 (besucht am 12. 09. 2022).

Keller, Frank (Juli 2010). „Cognitively Plausible Models of Human Language Processing". In: *Proceedings of the ACL 2010 Conference Short Papers*. Uppsala, Sweden: Association for Computational Linguistics, S. 60–67. URL: https://aclanthology.org/P10-2012 (besucht am 12. 09. 2022).

Krug, Markus (2018). „Description of a Corpus of Character References in German Novels - DROC [Deutsches ROman Corpus]". en. In: S. 18.

– (2020). „Techniques for the Automatic Extraction of Character Networks in German Historic Novels". eng. Diss. Universität Würzburg. DOI: 10.25972/OPUS-20918. URL: https://opus.bibliothek.uni-wuerzburg.de/frontdoor/index/index/docId/20918 (besucht am 14. 09. 2022).

Kubler, Sandra und Desislava Zhekova (2011). „Singletons and Coreference Resolution Evaluation". en. In: S. 7. URL: https://aclanthology.org/R11-1036.pdf.

Kunz, Kerstin, Ekaterina Lapshinova-Koltunski und José Manuel Martínez (Juni 2016). „Beyond Identity Coreference: Contrasting Indicators of Textual Coherence in English and German". In: *Proceedings of the Workshop on Coreference Resolution Beyond OntoNotes (CORBON 2016)*. San Diego, California: Association for Computational Linguistics, S. 23–31. URL: https://aclanthology.org/W16-0704 (besucht am 12. 09. 2022).

Lee, Kenton u. a. (Dez. 2017). *End-to-end Neural Coreference Resolution*. arXiv:1707.07045 [cs]. URL: http://arxiv.org/abs/1707.07045 (besucht am 12. 09. 2022).

Manning, Christopher u. a. (2014). „The Stanford CoreNLP Natural Language Processing Toolkit". en. In: *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Baltimore, Maryland: Association for Computatio-

nal Linguistics, S. 55–60. URL: http://aclweb.org/anthology/P14-5010 (besucht am 23. 09. 2022).

May, Philip, Philipp Reißel und German-NLP-Group (2021). *german-nlp-group/electra-base-german-uncased · Hugging Face*. URL: https://huggingface.co/german-nlp-group/electra-base-german-uncased (besucht am 01. 09. 2022).

Min, Bonan und Ralph Grishman (Mai 2012). „Challenges in the Knowledge Base Population Slot Filling Task". In: *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*. Istanbul, Turkey: European Language Resources Association (ELRA), S. 1137–1142. URL: http://www.lrec-conf.org/proceedings/lrec2012/pdf/256_Paper.pdf (besucht am 24. 09. 2022).

Mitkov, Ruslan u. a. (2012). „Coreference Resolution: To What Extent Does It Help NLP Applications?" en. In: *Text, Speech and Dialogue*. Hrsg. von Petr Sojka u. a. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, S. 16–27. ISBN: 978-3-642-32790-2. DOI: 10.1007/978-3-642-32790-2_2.

O'Brien, Shayne (Aug. 2019). *Koreferenzauflösung*. URL: https://github.com/shayneobrien/coreference-resolution (besucht am 20. 08. 2022).

Pagel, Janis und Nils Reiter (2014). „GerDraCor-Coref: A Coreference Corpus for Dramatic Texts in German". en. In: S. 10. URL: https://aclanthology.org/2020.lrec-1.7/.

Parikh, Dishant (Jan. 2021). *Disadvantages of RNN*. de. URL: https://iq.opengenus.org/disadvantages-of-rnn/ (besucht am 09. 09. 2022).

Pink, Glen, Joel Nothman und James R. Curran (Okt. 2014). „Analysing recall loss in named entity slot filling". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, S. 820–830. URL: https://aclanthology.org/D14-1089 (besucht am 06. 09. 2022).

Pradhan, Sameer u. a. (Juli 2012). „CoNLL-2012 Shared Task: Modeling Multilingual Unrestricted Coreference in OntoNotes". In: *Joint Conference on EMNLP and CoNLL - Shared Task*. Jeju Island, Korea: Association for Computational Linguistics, S. 1–40. URL: https://aclanthology.org/W12-4501 (besucht am 12. 09. 2022).

Richter, Felix (2016). *Infografik: Siri und Co. – Stets zu Diensten*. de. URL: https://de.statista.com/infografik/5627/nutzung-von-digitalen-virtuellen-assistenten/ (besucht am 29. 08. 2022).

Roesiger, Ina und Arndt Riester (Juli 2015). „Using prosodic annotations to improve coreference resolution of spoken text". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on*

Natural Language Processing (Volume 2: Short Papers). Beijing, China: Association for Computational Linguistics, S. 83–88. URL: https://aclanthology.org/P15-2014 (besucht am 16. 09. 2022).

Schröder, Fynn, Hans Ole Hatzel und Chris Biemann (2021). „Neural End-to-end Coreference Resolution for German in Different Domains“. In: *Proceedings of the 17th Conference on Natural Language Processing (KONVENS 2021)*. Düsseldorf, Germany: KONVENS 2021 Organizers, S. 170–181. URL: https://aclanthology.org/2021.konvens-1.15 (besucht am 11. 09. 2022).

Sharma, Abhishek (Juli 2020). *Part-of-Speech(POS) Tag — Dependency Parsing — Constituency Parsing*. de. URL: https://www.analyticsvidhya.com/blog/2020/07/part-of-speechpos-tagging-dependency-parsing-and-constituency-parsing-in-nlp/ (besucht am 23. 09. 2022).

Tanenhaus, M. K. u. a. (Juni 1995). „Integration of visual and linguistic information in spoken language comprehension“. eng. In: *Science (New York, N.Y.)* 268.5217, S. 1632–1634. ISSN: 0036-8075. DOI: 10.1126/science.7777863.

Toshniwal, Shubham u. a. (Nov. 2020). *Learning to Ignore: Long Document Coreference with Bounded Memory Neural Networks*. arXiv:2010.02807 [cs]. URL: http://arxiv.org/abs/2010.02807 (besucht am 12. 09. 2022).

Vaswani, Ashish u. a. (Juni 2017). „Attention Is All You Need“. en. In: URL: https://arxiv.org/abs/1706.03762v5 (besucht am 07. 09. 2022).

Versley, Yannick u. a. (2008). „BART: a modular toolkit for coreference resolution“. en. In: *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies Demo Session - HLT '08*. Columbus, Ohio: Association for Computational Linguistics, S. 9–12. URL: http://portal.acm.org/citation.cfm?doid=1564144.1564147 (besucht am 25. 08. 2022).

Abbildungsverzeichnis

Tabellenverzeichnis

Digitaler Anhang

In diesem Abschnitt werden die Quellen gezeigt, um die Ergebnisse selbst reproduzieren zu können. In diesem GitHub-Repository [1] befinden sich:

- alle drei Modellimplementationen und die entsprechenden Skripte für das Preprocessen der Datensätze
- die Masterarbeit (PDF)

Auf der schreibgeschützten SD-Karte befinden sich zusätzlich zum GitHub-Repository noch:

- die Masterarbeit (LaTeX)
- der Folien des Kolloquium (pptx und PDF)
- die Modelldateien und die Logs des Trainingsprozesses
- neben den Skripten und Modellimplementationen befinden sich zusätzlich noch die gesplitteten und preprocessten Datensätze mitsamt Logs und den benötigten Tools

[1] https://github.com/sch2da/Coreference-Resolution-with-extern-splits-and-german-word-level-model