INSTITUT FÜR INFORMATIK
Datenbanken und Informationssysteme

Universitätsstr. 1 D–40225 Düsseldorf

Heinrich Heine Universität Düsseldorf

Projektarbeit

Unterstützung der Auswertung von verkehrsbezogenen Bürger*innenbeteiligungsverfahren durch die Automatisierte Erkennung von Verortungen

Suzan Padjman

1 Zielsetzung der Projektarbeit

Durch Bürger*innenbeteiligungsverfahren haben Bürger*innen die Möglichkeit, ihre Interessen und Anliegen bei öffentlichen Vorhaben zu vertreten und einzubringen. Ziel ist es, die Qualität der politischen Entscheidungen der Kommunen durch die Einbeziehung von Bürger*innen zu verbessern. Um jedoch die Meinungen und Positionen der Bürger*innen in den Entscheidungsverfahren zu berücksichtigen, müssen die Beiträge der Bürger*innen zunächst ausgewertet werden. Da eine manuelle Auswertung und Analyse von Bürger*innenbeteiligungsverfahren einen hohen Aufwand und viel Zeit in Anspruch nimmt, wäre es hilfreich, die Auswertungen der Beiträge maschinell zu unterstützen. Die Forschungsgruppe CIMT[1] (Citizen Involvement in Mobility Transitions) erforscht konkrete Probleme und automatisierte Unterstützungsmöglichkeiten von Bürger*innenbeteiligungsverfahren der Verkehrswende zum Ausbau nachhaltiger Mobilität. Gerade das Thema Mobilität spielt eine große Rolle bei der Entwicklung einer nachhaltigeren Gesellschaft, da der Verkehrssektor zu enormen Treibhausgasemmissionen beiträgt. Untersuchungen der Forschungsgruppe haben ergeben, dass ein Bedarf zur (teil-)automatisierten Unterstützung der Analyse der Beiträge besteht [RE20]. Ein wichtiger Aspekt bei der Analyse der Beiträge ist z.B., welche Orte, Straßen, Kreuzungen, Radwege etc. Probleme aufweisen und verbesserungswürdig sind, um die Mobilität nachhaltiger zu gestalten. Die automatisierte Identifikation von solchen Verortungen könnte somit die Arbeit und den Zeitaufwand bei der Analyse reduzieren.
Ziel dieser Arbeit ist es daher, mithilfe von Methoden aus dem *Natural-Language-Processing* (NLP) eine automatisierte Lösung zur Identifikation von Verortungen zu finden. Diese Aufgabe wird als eine *Sequence-Labeling*-Aufgabe betrachtet, da Verortungen häufig aus mehreren hintereinanderfolgenden Token bestehen: Es sollen Verortungen in den Beiträgen identifiziert werden und die entsprechenden Wortsequenzen der Klasse „Verortung“ zugeordnet werden. Dabei ist in dieser Arbeit eine Verortung als die Beschreibung eines konkreten Ortes einer Maßnahme oder eines Zustandes definiert, welche eventuell auf einer Karte markiert werden könnte. Beispiele für Verortungen sind Straßennamen, Stadtteile und eindeutig zuordenbare Plätze, wie z.B. „in der Innenstadt“ oder „am Ausgang des Hauptbahnhofs“. Reine Lagebeschreibungen, wie z.B. „in der Einbahnstraße“, ohne eine konkrete Ortsbeschreibung werden dagegen nicht als Verortung betrachtet.
Es gibt verschiedene Ansätze für Sequence-Labeling Aufgaben, die entweder auf *Supervised-Learning*- oder *Unsupervised-Learning*-Methoden basieren. Letztere Methoden, insbesondere auf Transformer basierende *Deep-Learning*-Methoden, gewannen in den letzten Jahren zunehmend an Bedeutung. Transformer-basierte Modelle werden zunächst mit großen Mengen an ungelabelten Textdaten vortrainiert, um ein Sprachrepräsentationsmodell zu erlernen, und werden dann anschließend mit Hilfe eines *Fine-Tunings* an die jeweilige Aufgabe, wie z.B. an eine Klassifikations- oder eine *Named-Entity-Recognition*-Aufgabe, angepasst. Ein Beispiel für eine Transformer-basierte Methode ist das von Google-Entwicklern bereitgestellte Open-Source Modell **BERT** (Bidirectional Encoder Representations from Transformers) [DCLT19]. BERT verwendet für das Vortrainieren die sogenannte *Masked-Language-Modeling*-Methode. Bei dieser Art des Vortrainierens werden zunächst beliebige Token „maskiert“ und daraufhin lernt das Modell, diese anhand des Kontextes vorherzusagen. BERT erzielte bei mehreren NLP-Aufgaben, wie z.B. bei Named-Entity-Recognition (NER), *Question-Answering* und Textklassifikation, *State-of-the-Art* Ergebnisse. In [GN20] wurde BERT außerdem zur Extraktion von Verortungen aus Nachrichtentexten angewendet und mit herkömmlichen NER-Taggern wie spaCy [HMVLB20], NLTK [BKL09] und Stanford NLP [FGM05] verglichen. Während die herkömmlichen NER-Tagger dazu tendierten, Adressen, Straßennamen und sonstige Ortsangaben zu separieren, war dagegen BERT in der Lage, diese vollständig als eine komplette Ortsangabe zu erkennen. Weitere Transformer-basierte Methoden sind z.B. **RoBERTa** [LOG+19] und **ELECTRA** [CLLM20], dessen Architekturen ebenfalls auf BERT basieren. Im Unterschied zu BERT, wo die Maskierung der Token nur einmal zu Beginn des Trainings stattfindet, verwendet RoBERTa ein *dy-*

[1] `https://www.cimt-hhu.de/`

namisches Masked-Language-Modeling, d.h. beim Trainieren werden in jeder Epoche andere Token maskiert. ELECTRA dagegen verwendet die *Replaced-Token-Detection*-Strategie zum Trainieren, bei der beliebige Token durch andere Token ersetzt werden und das Modell erlernt, diese zu erkennen.
In dieser Arbeit liegt der Fokus auf der Anwendung der bereits erwähnten Transformer-basierten Methoden, um die Verortungen in den Beiträgen zu identifizieren. Zusätzlich wird ein Vergleich zu einem vortrainierten spaCy Modell gezogen. Der Aufbau dieser Arbeit lautet wie folgt: In Kapitel 2 wird der verwendete Datensatz und die Vorverarbeitung des Datensatzes beschrieben. In Kapitel 3 wird erläutert, inwiefern die Aufgabe dieser Projektarbeit eine NER-Aufgabe ist. In Kapitel 4 wird auf die verwendeten Methoden eingegangen. In Kapitel 5 werden die angewendeten Methoden evaluiert und in Kapitel 6 folgt schließlich ein Fazit.

2 Datensatz

Der verwendete Datensatz besteht aus 261 annotierten Beiträgen des Bonner Raddialogs[2] und 45 annotierten Beiträgen des Ehrenfelder Raddialogs[3] aus dem Jahr 2017. Insgesamt bestehen die 306 Beiträge aus 1444 Sätzen bzw. 22758 Token. Alle Beiträge wurden jeweils von drei Annotator*innen mithilfe des Annotationstools INCEpTION[4] annotiert. Zur Annotation wurden die folgenden beiden Klassen verwendet: Verortung („*“) und Keine-Verortung („-“).
Wie bereits in Kapitel 1 erwähnt wurde, ist in dieser Arbeit eine Verortung als die Beschreibung eines genauen Ortes einer Maßnahme oder eines Zustandes definiert. Wichtig dabei ist, dass es einen Fixpunkt gibt, der eine konkrete Ortzuschreibung zulässt und z.B. auf einer Karte markiert werden könnte. Dazu gehören z.B. Straßennamen, Stadtteile und eindeutig zuordenbare Plätze. Dabei ist zu beachten, dass Artikel, Pronomen und zusammenhängende Streckenverläufe mit mehreren Fixpunkten ebenfalls mit markiert werden und zur Verortung dazugehören, weil sie zur Konkretisierung und Nachvollziehung des Fixpunktes bzw. des Streckenverlaufs dienen. Einige Beispiele für Verortungen sind z.B. „in der Innenstadt“, „am Ausgang des Hauptbahnhofs“ und „diagonal in den Wiesenweg rein“. Zu beachten ist, dass reine Lagebeschreibungen, wie z.B. „in der Einbahnstraße“ oder „in Richtung Norden“ nicht als Verortung betrachtet werden, da eine Konkretisierung des Ortes bzw. ein Fixpunkt fehlt.
Die einzelnen annotierten Beiträge liegen als TSV-Datei im Format WebAnno TSV 3.2 vor. Dabei ist jeder einzelne Token in den Beiträgen mit einer der beiden Klassen deklariert. Zusammenhängende Tokensequenzen, die eine Einheit bilden, sind zusätzlich mit Nummern in eckigen Klammern versehen. Innerhalb eines Beitrags haben Token, die zur selben Einheit gehören, die gleiche Nummer.
Im Folgenden ist ein Beispiel-Abschnitt der TSV-Datei. Die erste Zeile beinhaltet einen kompletten Satz eines Beitrags. Die darauf folgenden Zeilen beinhalten je ein Token des Satzes und eine Markierung. Die ersten drei Ziffern zu Beginn der Zeilen geben die Beitragsnummer, Satznummer und Wortnummer des entsprechenden Tokens an. Die nachfolgenden zwei Ziffern markieren die Nummer der Anfangs- und Endzeichen des Tokens in dem entsprechenden Beitrag.

#Text=Es fehlen Parkplätze für Radfahrer an beiden Seiten des Hauptbahnhofs.
4-2-1 15-17 Es -
4-2-2 18-24 fehlen -
4-2-3 25-35 Parkplätze -
4-2-4 36-39 für -
4-2-5 40-49 Radfahrer -
4-2-6 50-52 an *[2]
4-2-7 53-59 beiden *[2]

[2] `https://www.raddialog.bonn.de/`
[3] `www.raddialog-ehrenfeld.koeln`
[4] `https://inception-project.github.io/`

4-2-8 60-66 Seiten *[2]
4-2-9 67-70 des *[2]
4-2-10 71-84 Hauptbahnhofs *[2]
4-2-11 84-85 . -

Aus diesem Beispiel ist zu entnehmen, dass die Sequenz „an beiden Seiten des Hauptbahnhofs“ als eine Verortung annotiert wurde.
Um die Verarbeitung der Daten zu vereinfachen, wurden zunächst alle einzelnen TSV-Dateien zu einer großen TSV-Datei zusammengeführt, die alle Beiträge enthält. Zusätzlich wurden für die später folgende Evaluation zwei weitere zusammengeführte TSV-Dateien erstellt, die nur aus den Beiträgen des Bonner Raddialogs bzw. nur aus den Beiträgen des Ehrenfelder Raddialogs bestehen.
Da allerdings Verortungssequenzen mit unterschiedlichen Nummern gekennzeichnet sind und sich somit ihre Tags voneinander unterscheiden, würden diese von den Algorithmen als unterschiedliche Klassen betrachtet werden. Um die Tags der Klasse Verortung zu vereinheitlichen, wurde das sogenannte **IOB**(Inside–Outside–Beginning)-Schema [RM99] gewählt, welches häufig für *Chunking*-Aufgaben verwendet wird. Das IOB-Schema markiert mithilfe von Präfixen, ob es sich bei einem Token um das erste, ein mittleres oder das letzte Token einer markierten Sequenz, auch *Chunk* genannt, handelt. Steht das Präfix „**B-**“ vor einem Tag, dann handelt es sich hierbei um das erste Token des Chunks bzw. um einen Chunk, das aus einem einzigen Token besteht. Das Präfix „**I-**“ gibt an, dass es sich um ein inneres bzw. das letzte Token eines Chunks handelt. Folgt nach einem mit „B-“ markierten Token kein mit „I-“ markiertes Token, dann handelt es sich bei dem Chunk um ein einzelnes Token. Token außerhalb eines Chunks, die also keiner Klasse angehören, werden lediglich mit einem „**O**“ markiert. Im Folgenden ist ein Beispielsatz aus dem Bonner Raddialog gegeben, welcher mit dem IOB-Schema (in Klammern) markiert wurde, wobei die Abkürzung „V“ für eine Verortung steht:

Die(O) Anbindung(O) an(O) das(B-V) Beueler(I-V) Zentrum(I-V) auch(O) wenig(O) angenehm(O) zu(O) fahren(O) durch(O) die(B-V) Obere(I-V) Wilhelmsstraße(I-V) .(O)

Nach dem IOB-Schema sind in dem Beispielsatz „das Bueler Zentrum“ und „die Obere Wilhelmsstraße“ als Verortungen markiert. Im kompletten Datensatz sind insgesamt 555 Token mit „B-V“, 2130 Token mit „I-V“ und die restlichen 20073 Token mit „O“ markiert. Hierbei lässt sich erkennen, dass ein extremes Klassenungleichgewicht herrscht. Vor allem die Klasse „B-V“ hat einen sehr geringen Support, was dazu führen kann, dass die Klasse weniger gut erkannt wird im Vergleich zu den anderen beiden Klassen. Mit insgesamt 2685 Token, die zu einer Verortung gehören, machen alle Verortungen etwa 11,8 % des Datensatzes aus.
Um die Daten für die Anwendung der Algorithmen vorzubereiten, wurden alle TSV-Dateien in CSV-Dateien umgewandelt, welche die Beitragsnummer, die Satznummer, das Token selbst und den zugehörigen IOB-Tag für jedes Token erfassen. Speziell für spaCy 3.0 und höhere Versionen müssen die Daten in dem neuen binären spaCy-Format vorliegen. Dazu wurden die Dateien zunächst in das IOB-Format umgewandelt und anschließend in das spaCy-Format konvertiert.

3 Named-Entity-Recognition

Die Erkennung von Verortungen kann als eine spezielle Art von *Named-Entity-Recognition*-Aufgabe betrachtet werden. Mit Named-Entity-Recognition (NER) bezeichnet man die Aufgabe, Eigennamen (Named-Entities) in einem Text zu identifizieren und diese in vordefinierte Klassen zu kategorisieren. Klassische NER-Tagger identifizieren z.B. Personennamen, Firmennamen, Zeitangaben und auch Ortsnamen. Verortungen unterscheiden sich jedoch von der „klassischen“ Definition von Orten, welche von den üblichen NER-Taggern verwendet wird. Bei der klassischen Definition beschränken sich Orte nur auf den öffentlichen Namen des Ortes und beinhalten im Gegensatz zu Verortungen z.B. keine

näheren Lagebeschreibungen. Außerdem werden Verortungen oder Teile von Verortungen, wie z.B. Adressen, von den klassischen NER-Taggern häufig in kleinere Einheiten unterteilt und nicht als eine zusammenhängende Ortsangabe identifiziert. [GN20] Daher ist es notwendig, entweder ein neues Modell zur Erkennung von Verortungen zu erstellen oder ein bereits votrainiertes NER-Modell an die Erkennung von Verortungen anzupassen. Letztere Methode ist aufgrund der geringen Anzahl an gegebenen Trainingsdaten für diese Arbeit besser geeignet und effizienter.

4 Methode

Im Folgenden wird erläutert, welche vortrainierten Deep-Learning-Methoden angewendet wurden, um die automatisierte Identifizierung von Verortungen in den Beiträgen in Form einer NER-Aufgabe zu lösen. Es wurden verschiedene BERT-Modelle, ein deutsches ELECTRA-Modell und zum Vergleich ein spaCy-Modell als Baseline angewendet.

4.1 spaCy

SpaCy ist eine Python-Bibliothek, die verschiedene Funktionen zur Verarbeitung natürlicher Sprache, unter anderem auch für NER, anbietet. Die vortrainierten NER-Tagger von spaCy sind in der Lage, viele gängige Arten von Named-Entities zu klassifizieren, wie z.B. Personennamen, Firmennamen und „klassisch“ definierte Orte. SpaCy bietet außerdem die Möglichkeit, ein bereits vortrainiertes NER-Modell an die Erkennung von neuen benutzerdefinierten Named-Entities anzupassen.
Für die Erkennung von Named-Entities in der deutschen Sprache bietet spaCy mehrere vortrainierte Modelle verschiedener Größe an. Für diese Arbeit wurde das größte deutsche Modell *de_core_news_lg*[5] gewählt, da das größte Modell laut Release-Details[6] eine bessere Performance als das kleine und das mittelgroße Modell erzielen kann. Das Modell wurde auf einer Sammlung von deutschen Wikipediatexten, Nachrichtentexten und Mediatexten trainiert, wie z.B. wie dem TIGER Korpus [BDE+04]. Der NER-Tagger des Modells basiert wie die meisten spaCy-Modelle auf einem Convolutional-Neural-Network (CNN). CNN-basierte Methoden lernen im Gegensatz zu z.B. Long-Short-Term-Memory(LSTM)-Architekturen nur die Abhängigkeiten zwischen Wörtern innerhalb eines Satzes statt des kompletten Dokuments und haben dadurch den Vorteil, dass sie deutlich effizienter sind. [ZZL15] Im Gegensatz zu Transformer-basierten Architekturen sind jedoch CNNs nicht in der Lage, die Relevanz und die Positionen der einzelnen Wörter in einem Satz zu berücksichtigen und somit den Fokus auf wichtige Teile eines Satzes zu setzen. Des Weiteren könnte die Erfassung der Abhängigkeiten für längere Sätze schwieriger werden, da die Abstände zwischen den Wörtern zu groß werden. Das spaCy-Modell wird daher als Vergleichsmodell angewendet, um die Performance der Transformer-basierten Modelle besser vergleichen und bewerten zu können. SpaCy stellt auch ein Transformer-basiertes Modell für die deutsche Sprache zur Verfügung, welches jedoch keinen NER-Tagger bietet und daher für diese Aufgabe nicht angewendet werden konnte.
Um das vortrainierte Modell an die Erkennung von Verortungen anzupassen, wurde der NER-Tagger des beschriebenen Modells mit unseren neuen benutzerdefinierten Tags („B-V“, „I-V“ und „O“) trainiert.

4.2 BERT-Modelle

BERT (Bidirectional Encoder Representations from Transformers) ist ein Transformer-basiertes Sprachrepräsentationsmodell, welches auf ungelabelten Textdaten für verschiedene NLP-Aufgaben vortrainiert wird. Das resultierende Modell kann anschließend anhand eines Fine-Tunings an eine spezifische NLP-Aufgabe, wie z.B. an eine NER-Aufgabe, angepasst werden. Dazu wird lediglich ein

[5] `https://spacy.io/models/de`
[6] `https://github.com/explosion/spacy-models/releases/tag/de_core_news_lg-3.1.0`

zusätzlicher Output-Layer kreiert, d.h. die restliche Modell-Architektur bleibt unverändert und muss nicht für die jeweilige Aufgabe modifiziert werden. Beim Training verwendet BERT die *Masked-Language-Modeling*(MLM)-Methode, bei welcher beliebige Token des Inputs (üblicherweise 15% der Token) maskiert werden und das Ziel ist, diese anhand des Kontextes zu erraten. Dazu wird sowohl der vorherige als auch der nachfolgende Kontext des Tokens berücksichtigt. Neben der MLM-Methode wird BERT auch für eine *Next-Sentence-Prediction*-Aufgabe trainiert, um anhand eines gegebenen Kontextes den nächsten Satz vorherzusagen und somit auch Repräsentationen von Textpaaren zu erlernen. [DCLT19]
Für diese Arbeit wurden insgesamt sieben verschiedene vortrainierte BERT-Modelle verwendet. Bei BERT-Modellen wird zwischen *cased* und *uncased* Modellen unterschieden. Bei uncased Modellen wird die Groß- und Kleinschreibung nicht berücksichtigt, da vor dem Training alle Großbuchstaben in Kleinbuchstaben umgewandelt werden, um Speicherbedarf und Performance zu verbessern. Namhafte Orte, wie z.B. Straßen und Plätze, werden zwar in der Regel groß geschrieben, allerdings existieren für diese üblicherweise keine Synonyme, daher würde die Groß- und Kleinschreibung vermutlich in diesem Fall keine große Rolle spielen. Da jedoch im Allgemeinen die Groß- und Kleinschreibung in der deutschen Sprache eine größere Rolle spielt als z.B. im Englischen, gibt es für die deutsche Sprache mehr cased als uncased Modelle. In dieser Arbeit wurden daher überwiegend cased Modelle ausgewählt. Um jedoch die Performance zwischen cased und uncased Modellen zu vergleichen, wurden für ein deutsches und für ein mehrsprachiges Modell beide Versionen getestet.
Für die deutsche Sprache existieren mehrere BERT-Modelle. Das bekannteste deutsche Modell ist GermanBERT[7] von deepset[8], einem Open-Source-Framework für NLP. Dieses Modell wurde auf deutschen Wikipedia-Texten, juristischen Dokumenten aus OpenLegalData[9] und Nachrichtentexten trainiert. Ende 2020 veröffentlichte deepset ein weiteres deutsches BERT-Modell, GBERT [CSM20], welches zusätzlich auf dem deutschen OSCAR Korpus [SSR19] und dem OPUS Datensatz [Tie12], einer Sammlung von Texten aus Filmen, Parlament-Vorträgen und Büchern, trainiert wurde. GBERT ist zudem in einer größeren Version mit deutlich mehr Parametern, etwa drei Mal so vielen wie das normale Modell, verfügbar. Die Performance des größeren Modells war jedoch in [CSM20] nicht bemerkenswert höher als die des normalen Modells, daher wurde in dieser Arbeit das normale Modell verwendet. Außerdem wurde in [CSM20] angemerkt, dass die großen Modelle aufgrund ihres unterschiedlichen Trainingsvorgangs auf viel mehr Token trainieren als die Basis-Modelle. Dadurch lassen sich die Vorteile der großen Modelle nicht so leicht quantifizieren. Ein weiteres deutsches BERT-Modell, DBMDZ BERT[10], lieferte in [CSM20] ähnlich gute Ergebnisse wie GermanBERT und wurde ebenfalls in dieser Arbeit angewendet. Im Gegensatz zu den anderen deutschen BERT-Modellen ist das DBMDZ-Modell auch in einer uncased Version verfügbar, welches ebenfalls angewendet wurde.
Obwohl alle Beiträge in deutscher Sprache vorliegen und die bereits genannten Modelle auf die deutsche Sprache spezialisiert sind, wurden dennoch zum Vergleich einige mehrsprachige BERT-Modelle ausgewählt. Das mehrsprachige BERT-Modell M-BERT (Multilingual BERT) [DCLT19] wurde auf Wikipedia-Texten aus insgesamt 104 Sprachen trainiert und deckt ebenfalls eine große Menge deutscher Texte ab. In [CSM20] wurde gezeigt, dass GBERT das mehrsprachige Modell M-Bert auf deutschen Korpora häufig leicht übertreffen konnte, allerdings war der Unterschied nicht gravierend. Zudem ist M-BERT sowohl in einer cased als auch uncased Version verfügbar. Des Weiteren konnte in [CSM20] das von Facebook AI[11] bereitgestellte mehrsprachige Modell XLM-RoBERTa [CKG+20] die deutschen BERT-Modelle in einigen Fällen sogar übertreffen. XLM-RoBERTa wurde auf dem Common Crawl Korpus [WLC+19] für 100 verschiedene Sprachen trainiert und ist im Vergleich zu den anderen erwähnten BERT-Modellen mit Abstand das größte Modell mit den meisten Parametern. Außerdem unterscheiden sich RoBERTa-Modelle von den herkömmlichen BERT-Modellen in der Art

[7] `https://www.deepset.ai/german-bert`
[8] `https://www.deepset.ai`
[9] `https://de.openlegaldata.io/`
[10] `https://github.com/dbmdz/berts`
[11] `https://ai.facebook.com/`

des Trainings. Anstelle die Token nur zu Beginn des Trainings zu maskieren, verwendet RoBERTa ein *dynamisches* Masked-Language-Modeling, welches während des Trainings in jeder Epoche andere Token auswählt, die maskiert werden sollen.
Alle erwähnten BERT-Modelle sind auf Hugging Face[12] verfügbar, einer Open-Source Plattform für implementierte NLP-Methoden.

4.3 GELECTRA

ELECTRA (Efficiently Learning an Encoder that Classifies Token Replacements Accurately) ist ein BERT-basiertes Modell, das sich in der Form des Vortrainierens von den herkömmlichen BERT-Modellen unterscheidet. Die Motivation von ELECTRA ist, dass das MLM-Verfahren mit hohen Rechenkosten einhergeht, da das Modell nur von einer geringen Menge der Token lernt. Im Falle von BERT wird nur von 15 % der Token pro Trainingsbeispiel gelernt. Als Alternative zur MLM-Methode verwendet ELECTRA das effizientere *Replaced-Token-Detection*-(RTD)-Verfahren zum Vortrainieren. Hier werden beliebige Token durch alternative und plausible „fake“ Token ersetzt, anstelle sie zu maskieren. Diese Token werden von einem kleinen Generator-Netzwerk ausgegeben. Anschließend wird das Modell trainiert, um vorherzusagen, welche Token ersetzt wurden. Es wird dabei aber nicht gelernt, das „originale“ Token vorherzusagen. Mit Hilfe von Experimenten wurde in [CLLM20] gezeigt, dass die RTD-Methode effizienter als die MLM-Methode ist, da die Aufgabe über alle Token definiert ist und nicht nur über die maskierten Token.
Für diese Arbeit wurde das deutsche ELECTRA-Modell GELECTRA [CSM20] ausgewählt, welches ebenfalls von deepset entwickelt wurde. GELECTRA ist auch in einer größeren Version mit etwa drei Mal so vielen Parametern verfügbar. In [CSM20] performten GBERT und GELECTRA ähnlich gut. Die Basis-Version von GBERT konnte GELECTRA leicht übertreffen, jedoch konnte das große Modell von GELECTRA das große Modell von GBERT ebenfalls leicht übertreffen. Zudem konnte in [CSM20] bestätigt werden, dass GELECTRA aufgrund der RTD-Methode viel effizienter lernt als GBERT. Im Vergleich zum großen GBERT-Modell sah das große Modell von GELECTRA in [CSM20] am Ende von einer Millionen Trainingsschritten nur die Hälfte der Anzahl der Token wie GBERT. Für diese Arbeit wurde auch für GELECTRA das Basis-Modell statt dem größeren gewählt, da auch hier die Performance des größeren Modells in [CSM20] nicht bemerkenswert höher war und die Vorteile der großen Modelle aufgrund ihres unterschiedlichen Trainingsvorgangs noch unklar sind.

5 Evaluation

In diesem Kapitel werden die in Kapitel 4 erwähnten Modelle evaluiert. In [DCLT19] werden für BERT einige Hyperparameterwerte vorgeschlagen: Für die Lernrate werden die Werte 5e-5, 4e-5, 3e-5 oder 2e-5 vorgeschlagen, für die Anzahl der Epochen 2 bis 4 und für die Größe der Batches 16 oder 32. Für alle BERT-Modelle und auch für GELECTRA wurden stets Parameterwerte aus diesen Wertebereichen ausgewählt, da die Modellarchitektur und die meisten Hyperparameter übereinstimmen. Außerdem haben die Entwickler von ELECTRA in [CLLM20] ebenfalls dieselben Parameterwerte für beide Modelle verwendet. Da die Anzahl der Trainingsdaten gering ist, wurde die Epochenanzahl auf 4 und die Größe der Batches auf 16 festgelegt. Hierbei ist anzumerken, dass eine höhere Batchgröße mit mehr Speicherbedarf verbunden ist und mit herkömmlichen CPUs schwer zu realisieren ist. Die Lernrate aller Modelle wurde auf den Wert 3e-5 festgelegt, welcher einen guten Basiswert darstellte. Eine zu hohe Lernrate kann zu einem instabilen Trainingsprozess führen, in welchem das Modell zu sensibel auf kleine Veränderungen reagiert, während eine zu niedrige Lernrate zu einem trägen Trainingsprozess führen kann, in welchem das Modell sich zu langsam an das Problem anpasst.
In Tabelle 1 sind für jedes Modell die F_1-Scores pro Klasse und der F_1-Macro angegeben. Zum Trainieren wurden hier beide Datensätze, der Bonner und Ehrenfelder Raddialog, verwendet. Um stabilere

[12] `https://huggingface.co/`

F_1-Scores				
Modell	B-V	I-V	O	Macro
spaCy	0.262	0.420	0.955	0.545
GermanBERT	0.590	0.876	0.983	**0.817**
GBERT	0.564	0.879	0.983	0.809
DBMDZ-cased	0.565	0.879	0.983	0.807
DBMDZ-uncased	0.515	0.875	0.983	0.791
M-BERT-cased	0.551	0.873	0.984	0.802
M-BERT-uncased	0.553	0.862	0.983	0.799
XLM-RoBERTa	0.336	0.864	0.982	0.727
GELECTRA	0.255	0.864	0.982	0.700

Tabelle 1: F1-Scores der verschiedenen Modelle für die Klassen „B-V“, „I-V“ und „O“.

und zuverlässigere Ergebnisse zu erhalten, wurde eine 5-fache Kreuzvalidierung angewendet und anschließend der Durchschnitt der jeweiligen Scores berechnet. Anhand der Ergebnisse in Tabelle 1 ist zu erkennen, dass im Schnitt GermanBERT mit einem F_1-Macro von 0.817 am besten performt hat. Entgegen den Erwartungen lieferte das neuere und verbesserte Modell GBERT nicht bessere Ergebnisse als das ältere Modell GermanBERT, performte aber mit einem F_1-Macro von 0.809 ähnlich gut. Die deutschen BERT-Modelle lieferten alle ähnlich gute Ergebnisse. Den niedrigsten F_1-Macro von den deutschen Modellen erreichte die uncased Version von DBMDZ mit einem Wert von 0.791. Die Performance der cased und uncased Versionen von DBMDZ unterscheiden sich nur minimal, somit ist kein klarer Vorteil der cased Version gegenüber der uncased Version zu erkennen. Im Gegenteil, die uncased Version ist schneller und effizienter, da sie weniger Speicherbedarf benötigt. Die beiden M-BERT-Modelle schneiden ähnlich gut wie die deutschen Modelle ab. Hier unterscheiden sich die Ergebnisse der uncased und der cased Version ebenfalls nur minimal. Das von Facebook AI bereitgestellte mehrsprachige Modell XLM-RoBERTa performte entgegen den Erwartungen mit einem F_1-Macro von 0.727 schlechter als M-BERT. Aufgrund der Größe der mehrsprachigen BERT-Modelle wird deutlich, dass sie keinen Vorteil bei der Erkennung von Verortungen gegenüber den deutschen Modellen haben. Von den Transformer-basierten Modellen lieferte GELECTRA mit einem F_1-Macro von 0.7 das schlechteste Ergebnis ab. Das spaCy-Modell schneidet den Vermutungen nach mit einem F_1-Macro von 0.545 mit großem Abstand am schlechtesten ab.
Es ist außerdem zu erkennen, das die Klasse „B-V“ deutlich schlechter erkannt wird als die anderen beiden Klassen. Hierbei ist jedoch zu berücksichtigen, dass ein extremes Klassenungleichgewicht im Datensatz herrscht, was bereits in Kapitel 2 erwähnt wurde. Es scheint sich somit als schwierig zu erweisen, den genauen Anfang einer Verortungssequenz zu erkennen. Wie bereits in Kapitel 2 erwähnt wurde, gehören in dieser Arbeit per Definition und Annotationsguidelines zuvorstehende Pronomen, Artikel und genauere Lagebeschreibungen zur Verortung dazu, wie z.B. bei „an beiden Seiten des Hauptbahnhofs“. Um beurteilen zu können, wie gut Verortungen oder Teile von Verortungen im Allgemeinen erkannt werden, werden im Folgenden die Klassen „B-V“ und „I-V“ als eine gemeinsame Klasse „V“ (Verortung) betrachtet. In den meisten Anwendungsfällen für die Erkennung von Verortungen ist es bereits hilfreich zu wissen, wo etwa im Text Verortungen bzw. Teile von Verortungen stehen. Der genaue Anfang bzw. das genaue Ende einer Sequenz, die eine Verortung darstellt, ist nicht immer relevant. Einige Beispiele für weniger relevante Bestandteile einer Verortung sind z.B. zuvorstehende Artikel, da sich ohne diese dennoch ableiten lässt, um welchen Standort es sich handelt. Werden jedoch der Beginn und das Ende der Verortungssequenzen strikt wie in Tabelle 1 bewertet, so ist eine höhere Fehlerwahrscheinlichkeit gegeben. Es wird nämlich nicht berücksichtigt, dass z.B. Token, welche zur Klasse „I-V“ gehören, aber als „B-V“ vorhergesagt werden (oder umgekehrt), dennoch korrekt als Teil einer Verortung erkannt wurden.
In Tabelle 2 wurden die Ergebnisse der Modelle aus Tabelle 1 für die zusammengefasste Klasse „V“

F_1-Scores			
Modell	V	O	Macro
spaCy	0.583	0.955	0.769
GermanBERT	0.904	0.983	0.944
GBERT	0.906	0.983	**0.945**
DBMDZ-cased	0.903	0.983	0.943
DBMDZ-uncased	0.903	0.983	0.943
M-BERT-cased	0.898	0.984	0.941
M-BERT-uncased	0.894	0.983	0.939
XLM-RoBERTa	0.891	0.982	0.936
GELECTRA	0.897	0.982	0.940

Tabelle 2: F1-Scores der Modelle für die zusammengefasste Klasse „V“ (Verortung) und für die Klasse „O“.

und „O“ angepasst. Hier wird deutlich, dass bei der Erkennung von Verortungen tatsächlich ein sehr großer Anteil an Token, die Bestandteil einer Verortung sind, erkannt werden. Die F_1-Scores der jeweiligen Modelle für die zusammengefasste Klasse „V“ sind stets deutlich höher als für die einzelnen Klassen „B-V“ und „I-V“ (vgl. Tabelle 1). Dies bestätigt die Vermutung, dass Token häufig korrekt als Bestandteil einer Verortung identifiziert werden, jedoch der genaue Beginn und das Ende einer Verortungssequenz weniger gut erkannt werden. Zudem ist in Tabelle 2 zu erkennen, dass die Unterschiede zwischen den Ergebnissen der Transformer-basierten Modelle deutlich kleiner sind als in Tabelle 1. Den besten F_1-Macro erreicht GBERT mit einem Wert von 0.945. Von den Transformer-basierten Modellen schneidet XLM-RoBERTa mit einem F_1-Macro von 0.936 am schlechtesten ab. Hier ist zu erkennen, wie klein der Unterschied der Performance des besten und des schlechtesten Transformer-basierten Modells ist. Im Allgemeinen performten die deutschen BERT-Modelle auch hier im Schnitt besser als die mehrsprachigen BERT-Modelle. Das spaCy-Modell lieferte mit einem F_1-Macro von 0.769 erneut ein deutlich schlechteres Ergebnis als die Transformer-basierten Methoden. Im Allgemeinen lässt sich schließen, dass die Transformer-basierten Modelle sehr gut in der Lage sind, Token zu identifizieren, die Teil einer Verortung sind. Um jedoch die Erkennung der genauen Sequenzgrenzen zu verbessern, könnte man z.B. mehr Trainingsdaten verwenden.
Ein weiterer Aspekt, der zu beachten ist, ist dass in Datensätzen aus verschiedenen Raddialogen einige Unterschiede auftreten können z.B. in der Struktur der Beiträge. Daher ist es sinnvoll zu untersuchen, ob die beiden besten Modelle GBERT und GermanBERT für die Klassen „B-V“, „I-V“ und „O“ auch auf anderen ungesehenen Datensätzen ähnlich gut performen können und robust sind. Dazu werden im Folgenden GBERT und GermanBERT nur auf dem Bonner Datensatz trainiert und anschließend auf dem Ehrenfelder Datensatz getestet. Mit dem Ehrenfelder Raddialog wird nicht trainiert, da dieser Datensatz mit nur 45 Beiträgen deutlich kleiner ist als der Bonner Raddialog, welcher aus 261 Beiträgen besteht. In Tabelle 3 sind die F_1-Scores von GBERT und GermanBERT für die einzelnen Klassen angegeben und der F_1-Macro. Sowohl GermanBERT als auch GBERT schneiden mit einem jeweiligen F_1-Macro von 0.86 und 0.84 sogar etwas besser ab wie für den gemischten Datensatz (vgl. Tabelle 1). Daraus kann man ableiten, dass die beiden besten Modelle in der Lage sind, auch auf Daten aus ungesehenen Raddialogen gut zu performen.

6 Fazit und Ausblick

Im Rahmen dieser Projektarbeit wurden verschiedene deutsch- und mehrsprachige BERT-Modelle, ein ELECTRA-Modell und zum Vergleich ein spaCy-Modell zur Erkennung von Verortungen in Beiträgen aus Bürger*innenbeteiligungsverfahren zur Verkehrswende angewendet und miteinander ver-

F_1-Scores				
Modell	B-V	I-V	O	Macro
GermanBERT	0.69	0.90	0.98	0.86
GBERT	0.63	0.91	0.98	0.84

Tabelle 3: F1-Scores der beiden besten Modelle GBERT und GermanBERT, trainiert auf dem Bonner Raddialog und getestet auf dem Ehrenfelder Raddialog für die Klassen „B-V“, „I-V“ und „O“.

glichen. Dabei wurde festgestellt, dass die BERT-Modelle im Allgemeinen besser als das ELECTRA-Modell performten. Die beiden besten Modelle, GermanBERT und GBERT, erzielten jeweils einen F_1-Macro von etwa 0.82 und 0.81. XLM-RoBERTa performte von den BERT-Modellen am schlechtesten. Deutlich schlechter als die Transformer-basierten Modelle schneidete das spaCy-Modell ab. Dies könnte daran liegen, dass im Gegensatz zu den Transformer-basierten Architekturen das CNN-basierte spaCy-Modell nicht in der Lage ist, die Relevanz und die Positionen der einzelnen Wörter eines Satzes zu berücksichtigen und somit den Fokus auf wichtige Teile des Satzes zu setzen. Des Weiteren wurde festgestellt, dass die deutschen BERT-Modelle GBERT, GermanBERT und die DBMDZ-Modelle für die Erkennung von Verortungen in deutschen Texten besser geeignet sind als die mehrsprachigen BERT-Modelle M-BERT und XLM-RoBERTa, da die deutschen Modelle im Vergleich zu den mehrsprachigen Modellen deutlich kleiner und effizienter sind. Es wurden außerdem die cased und uncased Versionen von DBMDZ und M-BERT miteinander verglichen. Dabei wurde beobachtet, dass die Performance der cased und uncased Versionen etwa gleich gut waren. Daraus lässt sich schließen, dass die Groß- und Kleinschreibung für die Erkennung von Verortungen keine große Rolle spielt. Zudem hat die uncased Version gegenüber der cased Version bezüglich Effizienz und Speicherbedarf einen klaren Vorteil. Im Allgemeinen wurde bei der Verwendung des IOB-Schemas außerdem festgestellt, dass die Anfangstoken einer Verortungssequenz häufig nicht als solches erkannt wurden. Dies ist wahrscheinlich auf das extreme Klassenungleichgewicht und die geringe Anzahl an Trainingsdaten zurückzuführen. Daher wurde zusätzlich untersucht, wie gut die Verortungen unabhängig von den Sequenzanfängen und Sequenzenden erkannt werden. Hierbei wurde festgestellt, dass viele Token, die nicht korrekt als Anfangstoken oder inneres bzw. letztes Token einer Verortung klassifiziert wurden, dennoch korrekt als Teil einer Verortung erkannt wurden. Außerdem wurde beobachtet, dass die verschiedenen Transformer-basierten Modelle bei der Erkennung von Verortungen unabhängig von den Sequenzanfängen und Sequenzenden nahezu gleich gut performt haben. Dabei lagen die F_1-Macros von allen Transformer-basierten Modellen bei über 0.936. Ein weiterer untersuchter Aspekt war, wie gut die beiden besten Modelle GermanBERT und GBERT auf ungesehenen Daten, z.B. aus anderen Raddialogen, performen. Dazu wurden beide Modelle nur auf dem Bonner Raddialog trainiert und anschließend auf dem Ehrenfelder Raddialog getestet. Hier waren die Ergebnisse sogar knapp besser bei Verwendung des gemischten Datensatzes. Hieraus ließ sich ableiten, das die Modelle robust sind und auch auf anderen Raddialogen ähnlich gut performen können.

Nach Beendigung der Projektarbeit sind mir einige Punkte bewusst geworden, die ich anders machen würde, wenn ich die Projektarbeit wiederholen würde. Dazu gehört z.B. den Fokus auf die eigentliche Projektarbeit-Aufgabe von Anfang an zu setzen bzw. die Aufgabe von Beginn an deutlich zu formulieren, um überflüssigen Zeitaufwand zu Beginn der Arbeit zu vermeiden. Außerdem würde ich bei einem erneuten Versuch in Betracht ziehen, die Transformer-basierten Modelle über eine leistungsstärkere GPU statt dem Laptop-eigenen CPU laufen zu lassen, da die Modelle viel Zeit und Speicher in Anspruch nehmen. Dadurch wäre es auch möglich mit verschiedenen Parametern, wie z.B. mit höheren Batchgrößen, zu experimentieren und diese zu vergleichen. Dementsprechend wäre ein Parameter-Tuning einfacher zu realisieren.

In zukünftige Arbeiten über die automatisierte Erkennung von Verortungen könnte man zusätzlich versuchen, die erkannten Verortungen in einer Karte zu markieren und die Koordinaten der gefundenen Verortung mit den Koordinaten der tatsächlichen Verortung vergleichen. Des Weiteren könnte

die Verwendung von Datensätzen aus weiteren Raddialogen in Betracht gezogen werden, um das Modell möglichst robust zu machen. Ein weiterer Aspekt, der in dieser Arbeit nicht in Betracht gezogen wurde ist, dass Verortungen nicht immer notwendigerweise in Form einer nacheinander folgenden Sequenz von Token auftreten. Zusammenhängende Verortungen können auch über Sequenz- und Satzgrenzen hinweg auftreten. Des weiteren werden in Raddialogen häufig auch Beschreibungen über den zu verbessernden Zustand der erwähnten Verortungen und eventuell auch Vorschläge zur Verbesserung zu den Verortungen gemacht. Für eine automatisierte Analyse der Beiträge wäre es hilfreich, diese zu ebenfalls identifizieren und den Verortungen zuzuordnen.

Literatur

[BDE+04] Brants, Sabine ; Dipper, Stefanie ; Eisenberg, Peter ; Hansen-Schirra, Silvia ; König, Esther ; Lezius, Wolfgang ; Rohrer, Christian ; Smith, George ; Uszkoreit, Hans: TIGER: Linguistic interpretation of a German corpus. In: *Research on language and computation* 2 (2004), Nr. 4, S. 597–620

[BKL09] Bird, Steven ; Klein, Ewan ; Loper, Edward: *Natural Language Processing with Python*. O'Reilly Media Inc., 2009

[CKG+20] Conneau, Alexis ; Khandelwal, Kartikay ; Goyal, Naman ; Chaudhary, Vishrav ; Wenzek, Guillaume ; Guzmán, Francisco ; Grave, Edouard ; Ott, Myle ; Zettlemoyer, Luke ; Stoyanov, Veselin: Unsupervised Cross-lingual Representation Learning at Scale. In: *arXiv preprint arXiv:1911.02116* (2020)

[CLLM20] Clark, Kevin ; Luong, Minh-Thang ; Le, Quoc V. ; Manning, Christopher D.: Electra: Pre-training text encoders as discriminators rather than generators. In: *arXiv preprint arXiv:2003.10555* (2020)

[CSM20] Chan, Branden ; Schweter, Stefan ; Möller, Timo: German's Next Language Model. In: *arXiv preprint arXiv:2010.10906* (2020)

[DCLT19] Devlin, Jacob ; Chang, Ming-Wei ; Lee, Kenton ; Toutanova, Kristina: 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* (2019)

[FGM05] Finkel, Jenny R. ; Grenager, Trond ; Manning, Christopher D.: Incorporating non-local information into information extraction systems by gibbs sampling. In: *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)* (2005), S. 363–370

[GN20] Gupta, Sarang ; Nishu, Kumari: Mapping Local News Coverage: Precise location extraction in textual news content using fine-tuned BERT based language model. In: *Proceedings of the Fourth Workshop on Natural Language Processing and Computational Social Science* (2020), S. 155–162

[HMVLB20] Honnibal, Matthew ; Montani, Ines ; Van Landeghem, Sofie ; Boyd, Adriane: spaCy: Industrial-strength natural language processing in python. In: *Zenodo* (2020)

[LOG+19] Liu, Yinhan ; Ott, Myle ; Goyal, Naman ; Du, Jingfei ; Joshi, Mandar ; Chen, Danqi ; Levy, Omer ; Lewis, Mike ; Zettlemoyer, Luke ; Stoyanov, Veselin: RoBERTa: A Robustly Optimized BERT Pretraining Approach. In: *arXiv preprint arXiv:1907.11692* (2019)

[RE20] Romberg, Julia ; Escher, Tobias: Analyse der Anforderungen an eine Software zur (teil-)automatisierten Unterstützung bei der Auswertung von Beteiligungsverfahren. (2020). `https://www.cimt-hhu.de/2020/erster-praxisworkshop/`

[RM99] Ramshaw, Lance A. ; Marcus, Mitchell P.: Text chunking using transformation-based learning. In: *Natural language processing using very large corpora* (1999), S. 157–176

[SSR19] Suárez, Pedro Javier O. ; Sagot, Benoît ; Romary, Laurent: Asynchronous pipeline for processing huge corpora on medium to low resource infrastructures. In: *7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7)* (2019)

[Tie12] Tiedemann, Jörg: Parallel data, tools and interfaces in OPUS. In: *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)* (2012)

[WLC+19] Wenzek, Guillaume ; Lachaux, Marie-Anne ; Conneau, Alexis ; Chaudhary, Vishrav ; Guzmán, Francisco ; Joulin, Armand ; Grave, Edouard: Ccnet: Extracting high quality monolingual datasets from web crawl data. In: *arXiv preprint arXiv:1911.00359* (2019)

[ZZL15] Zhang, Xiang ; Zhao, Junbo J. ; LeCun, Yann: Character-level Convolutional Networks for Text Classification. In: *CoRR* abs/1509.01626 (2015)