Martin-Luther-Universität Halle-Wittenberg
Institut für Informatik
Studiengang Informatik, M.Sc.

Extraktion von Metadaten aus wissenschaftlichen Artikeln mittels Transformern

Masterarbeit

Christian Peters
geb. am: 09.12.1990 in Menden

Matrikelnummer 212217274

1. Gutachter: Dr. habil. rer. nat. Alexander Hinneburg
2. Gutachter: Ferdinand Schlatt

Datum der Abgabe: 5. Januar 2022

Erklärung

Hiermit versichere ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Halle (Saale), 5. Januar 2022

. . . . . .
Christian Peters

Inhaltsverzeichnis

Kapitel 1

Einleitung

Auf ArXiv, einer digitalen Bibliothek für Preprints verschiedener naturwissenschaftlicher Disziplinen, werden pro Monat über 10.000 neue Artikel hochgeladen[1]. Diese Datenmenge kann manuell nicht organisiert werden. Dieses Problem tritt auch schon bei einzelnen wissenschaftlichen Disziplinen auf, zum Beispiel den Bodenwissenschaften. Für diese stellt das Projekt „BonaRes“ eine Artikeldatenbank zur Verfügung, die Artikel nach bestimmten Daten gruppiert und bereitstellt[2]. Die in der Datenbank bereitgestellten Daten sind bibliographische Metadaten, also Daten über Artikel. Solche Metadaten sind zum Beispiel Autor*in und Titel des Artikels, oder das Veröffentlichungsdatum. Diese Metadatensammlung soll erweitert werden um Metadaten, die den Inhalt des jeweiligen Artikels genauer beschreiben.

Relevante Daten für die Bodenwissenschaften sind beispielsweise im Artikel vorkommende Bodenarten, Nutzpflanzen, Bodentexturen, oder Koordinaten. Diese vier Metadatenklassen sollen in dieser Arbeit extrahiert werden. Durch die Erweiterung der Metadaten um diese inhaltlichen Metadaten wird die Suche nach bestimmten Themen und die Aggregation von Artikeln stark vereinfacht. Da den Artikelautor*innen im Allgemeinen nicht klar ist, welche inhaltlichen Metadaten ihrer Artikel relevant werden könnten, werden sie in der Regel nicht bereitgestellt. Das heißt, die Artikel müssen für eine Klassifizierung selbst durchsucht werden. Aufgrund der eingangs erwähnten Menge an Veröffentlichungen ist das aber nicht manuell möglich, es wird also eine automatisierte Lösung für die Extraktion der Metadaten gesucht.

Die Artikel liegen als PDF-Dateien vor. Um eine Extraktion durchzuführen, müssen die Artikel in ein strukturiertes Format umgewandelt werden, beispielsweise XML. Diese Konvertierung funktioniert bei Texten größtenteils fehlerfrei, bei Symbolen treten aber oft Fehler auf (vgl. Tabelle 1.1). Die Extraktion von

[1] https://archive.md/DxVVv
[2] https://www.bonares.de/

Tabelle 1.1: Vergleich Originaltext (oben) und konvertierter Text (unten). Text aus „Effects of fertility management strategies on phosphorus bioavailability in four West African soils“ von Sinaj et al. (2001). Zeilenumbrüche im Originaltext sind übernommen worden.

The soils were Psammentic Paleustalfs at Kara Bedji (13°15′N, 2°32′E) and Goberi (12°58′N, 2°50′E) in the southern Sahelian zone and an Arenic Kandiustalf at Gaya-Bengou (11°59′N, 3°32′E) in the Sudanian zone of Niger and a Haplustalf at Fada-Kouaré (11°59′N, 0°19′E) in the Sudanian zone of Burkina Faso. With their range of soil chemical properties they are typical for this region.
The soils were Psammentic Paleustalfs at Kara Bedji (13 • 15 N, 2 • 32 E) and Goberi (12 • 58 N, 2 • 50 E) in the southern Sahelian zone and an Arenic Kandiustalf at Gaya-Bengou (11 • 59 N, 3 • 32 E) in the Sudanian zone of Niger and a Haplustalf at Fada-Kouaré (11 • 59 N, 0 • 19 E) in the Sudanian zone of Burkina Faso. With their range of soil chemical properties they are typical for this region.

Bodenarten, Nutzpflanzen und Bodentexturen kann in diesen strukturierten Artikeln mit Hilfe von Wortlisten durchgeführt werden, da diese Datenklassen jeweils nominale Daten beinhalten. Mit diesen von BonaRes bereitgestellten Wortlisten können durch Volltextsuche in den Artikeln die jeweiligen relevanten Metadaten gefunden werden. Dafür müssen allerdings reguläre Ausdrücke verwendet werden, da die Wörter auf der Wortliste nicht unbedingt den Wörtern, die im Artikel vorkommen, exakt entsprechen. Es kann beispielsweise ein Plural-S angehangen werden, es kann einen Schreibfehler geben, oder ein Zeilenumbruch im Wort zerlegt es so, dass es nicht mehr direkt gefunden wird. Reguläre Ausdrücke sind auch notwendig, um Koordinaten zu finden, da hier keine Wortlisten existieren können, weil Koordinaten metrische Daten sind. Zusätzlich dazu tauchen Koordinaten in den konvertierten Artikeln in immer anderen Formen auf (vgl. Tabelle 1.2).

Durch diese großen Unterschiede im Auftauchen der Koordinaten ist es schwierig, diese mit regulären Ausdrücken zu finden. Ist der Ausdruck zu allgemein, werden viele falsch-positive Treffer gefunden. Ein Beispiel für einen allgemeinen regulären Ausdruck, der alle Koordinaten aus Tabelle 1.2 findet, ist

Tabelle 1.2: Beispiele für Koordinaten und deren Konvertierungen

Zu extrahierende Koordinate	Konvertierter Originaltext
35° 05’ S, 147° 20’ E	35805 H S, 147820 H E
13° 15’ N, 2° 32’ E	13 • 15 N, 2 • 32 E
52° 52’ N, 13° 53’ E	52852VN, 13853VE
43° 56’ N, 71° 45’ W	43j56VN; 71j45VW
43° 43’ 17”, 4° 10’ 25” E	43°43¢17¢¢N-4°10¢25¢¢E
31° 56’ S, 115° 20’ E	31.56 S, 115.20 E
43° 58’ 16” N, 5° 0’ 25” E	438589160 N, 5809250 E
2° 47’ - 3° 15’ N, 10° 24’ - 10° 51’ E	2847V -3815VN and 10824V -10851VE
124° 27’ W, 47° 15’ N	124 • 27 W, 47 • 15 N

```
(.{0,6}(\d.{0,6}\d.{0,6}(?:N|S|W|E))){2}
```

Wird dieser Ausdruck auf alle Paragraphen des in dieser Arbeit benutzten Datensatzes angewandt, findet man beispielsweise folgende Ausdrücke:

```
W19 W20 W5
6.5 AE 1.0 10.0 AE
4306 AE 750 3469 AE 591 ns N
1.67-mM NaHCO 3 /1.67-mM N
3276 AE 513 1839 AE
The 87 Sr/ 86 S
VOL. 169 ˜ NO. 11 S
The 87 Sr/ 86 S
3.2.1. Winter 1994W
VOL. 169 ˜ NO. 10 FLUORE
```

Insgesamt liegt die Falsch-Positiv-Rate für diesen Ausdruck bei über 90%, er ist also für die automatische Extraktion nicht geeignet. Wird der Ausdruck strenger genommen, werden allerdings nicht alle Koordinaten gefunden. Es müssten also verschiedene reguläre Ausdrücke verwendet werden, wobei jeder einzelne Ausdruck eng gewählt werden muss, um alle Koordinaten zu finden. Das würde bedeuten, dass es reguläre Ausdrücke geben könnte, die nur für wenige Koordinaten nötig wären. Um eine Liste von diesen regulären Ausdrücken zu erstellen, müsste man also erst einmal viele Koordinaten finden. Weiterhin ist nicht klar, ob wirklich alle Koordinaten gefunden worden sind, außer, wenn alle Artikel manuell durchgeschaut wurden. Dieses Problem tritt auch bei Wortlisten auf: Es ist nicht unbedingt klar, ob die Listen vollständig

sind, oder ob noch Wörter fehlen. Dieses Problem der Vollständigkeit ist also eine Schwierigkeit der Datenextraktion und soll in dieser Arbeit bearbeitet werden.

Um dieses Problem zu lösen, werden in dieser Arbeit künstliche neuronale Netze eingesetzt. Diese können sowohl Paragraphen, als auch einzelne Worte oder Wortteile vorher festgelegten Klassen zuordnen. So könnte beispielsweise ein Paragraph, der Koordinaten enthält, als „Koordinatenparagraph“ klassifiziert werden, und das Wort „Vertisol“, welches eine Bodenart ist, könnte als „Bodenart“ klassifiziert werden. Sowohl die Sequenzklassifikation, bei der Paragraphen klassifiziert werden, als auch die Tokenklassifikation, bei der einzelne Wörter oder Wortteile klassifiziert werden, benötigen Trainingsdaten. Das sind in dieser Arbeit für die Sequenzklassifikation Paragraphen, deren Klasse bekannt sind, und für die Tokenklassifikation Paragraphen, bei denen jedes enthaltende Wort einer Klasse zugewiesen wurde. Die Klassifikation der Trainingsdaten wird mit einer Volltextsuche durchgeführt, mit Wortlisten und regulärem Ausdruck für die nominalen Daten, und einem regulären Ausdruck für Koordinaten. Dabei wird ein strenger Ausdruck eingesetzt, damit die Zahl der falsch-positiven Klassifizierungen gering gehalten wird, um die Klassifikation durch das Netz nicht negativ zu beeinflussen. Dadurch ist die Menge der Trainingsdaten allerdings im Verhältnis zum gesamten Datensatz klein.

Eine größere Menge von Trainingsdaten ist hilfreich, um eine Überspezialisierung des Netzes zu verhindern. Es ist allerdings nicht immer möglich, genug klassifizierte Paragraphen zu finden, um eine ausreichend große Trainingsdatenmenge zu erzeugen. Um dieses Problem der Trainingsdatenerzeugung zu lösen, werden verschiedene Techniken der Data Augmentation verwendet. Bei dem in dieser Arbeit genutzten Datensatz gibt es nur wenige Paragraphen, in denen Koordinaten vorhanden sind, weswegen diese Techniken nur auf das Training der Klassifikation von Koordinaten angewandt werden. Dabei wird unter anderem untersucht, ob das Kürzen von Paragraphen oder das Austauschen von ganzen Koordinaten oder Koordinatenanteilen hilft, ein Netz genauer klassifizieren zu lassen. Ein auf diesen Trainingsdaten trainiertes Netz soll in der Lage sein, Paragraphen und einzelne Wörter beziehungsweise Wortteile in die richtigen, relevanten Klassen einzuordnen. Mit einem solchen Netz kann dann das Problem der Vollständigkeit bearbeitet werden. Da ein Netz keine Wörter auswendig lernt, sondern den Kontext mit in Betracht zieht, sollten auch relevante Wörter gefunden werden können, die sich nicht durch einen regulären Ausdruck oder mit Wortlisten finden lassen. Ob das gelingt, wird in dieser Arbeit anhand der Tokenklassifikation untersucht. Dabei ist die Genauigkeit der Klassifikation auf Tokenebene wichtig, da nur Wörter und Wortteile extrahiert werden, die auch einer relevanten Klasse zugeordnet werden. Um eine möglichst hohe Genauigkeit zu erzielen, wurden noch andere Varianten

getestet, um die Genauigkeit der Tokenklassifikation zu erhöhen: Änderungen an der Loss-Funktion des Netzes, und Änderungen an der Wortzerlegung für das Netz. Zusätzlich wurde untersucht, ob Multi-Task-Learning, also das Klassifizieren von mehreren Klassen mit einem Modell, oder Single-Task-Learning, also die Klassifikation mit nur einer Klasse von einem Modell, bessere Ergebnisse erzielt.

Zuletzt wird noch das Problem der Extraktion untersucht, also die eigentliche Extraktion der relevanten Daten. Hat man ein Netz, welches mit hoher Präzision einzelne Wörter und Paragraphen klassifizieren kann, müssen die Daten mit Hilfe dieses Netzes noch extrahiert werden. Auf Tokenebene ist das einfach: Hat ein Wort die Klasse „Bodenart“, kann es direkt als Bodenart extrahiert werden. Bei der Sequenzklassifikation ist das nicht so einfach: Weiß man, dass ein Paragraph beispielsweise eine Nutzpflanze enthält, weiß man nicht, welches Wort aus dem Paragraphen diese Pflanze ist. Um dieses Extraktionsproblem zu lösen, werden in dieser Arbeit drei Varianten untersucht, die mit der Sequenzklassifikation einzelne Wörter extrahieren sollen.

Die drei vorgestellten Probleme werden durch die Ansätze dieser Arbeit verbessert. Die trainierten Netze finden viele relevante Daten, die vorher nicht klassifiziert wurden, und können somit mehr Daten extrahieren als eine reine Volltextsuche. Die Netze verallgemeinern also über die Trainingsdaten hinaus, allerdings ist unklar, ob so wirklich alle relevanten Daten gefunden wurden. Das ist eine Verbesserung des Problemes der Vollständigkeit. Das Problem der Trainingsdatenerzeugung konnte auch verbessert werden. Die einzelnen hier angewendeten Data Augmentation-Techniken konnten jeweils in Teilbereichen Verbesserungen erzielen, es gab aber keine Technik, die die Genauigkeit in allen Teilbereichen verbessert hat. Auch bei dem Problem der Extraktion gab es Verbesserungen. Mit den in dieser Arbeit vorgestellten Methoden können über 95% der in den Artikeln vorhandenen Bodenarten extrahiert werden. Der Code, der für diese Arbeit geschrieben wurde, ist auf Github [3] verfügbar.

[3] https://github.com/Xerono/Data_Extraction

Kapitel 2

Verwandte Arbeiten

In dieser Arbeit wird BERT verwendet, um Metadaten aus bodenwissenschaftlichen Artikeln zu extrahieren. Dabei werden die Trainingsdaten verändert, um die Trainingsdatenmenge zu vergrößern. Zusätzlich wird ein Vergleich zwischen Multi- und Single-Task-Learning durchgeführt.

Gupta and Nishu [2020] nutzten BERT, um Ortsangaben aus Nachrichtentexten zu extrahieren. Da allerdings in Nachrichtentexten im Allgemeinen keine Koordinaten auftauchen, fokussierten sie sich auf einen Named-Entity-Recognition-Ansatz, mit Erfolg: Sie erzielten einen F1-Wert von über 70%. Zhang et al. [2020] nutzten BERT zur Extraktion von domänenspezifischen Daten, indem sie Daten aus wirtschaftlichen Verträgen extrahierten. Dabei wurde festgestellt, dass weniger als 100 Dokumente ausreichend sind, um eine verhältnismäßig gute Genauigkeit zu erhalten. Allerdings bezieht sich das auf zu extrahierende Daten, die immer in ähnlicher Form auftauchen und somit durch das Netz leichter zu identifizieren sind. BERT wurde erfolgreich eingesetzt, um Schlüsselwörter und -Phrasen aus Texten zu extrahieren (Sharma and Li [2019]). Auch im Topic Modeling-Bereich gibt es Anwendungsbereiche für BERT (Grootendorst and Reimers [2021]). Auch wurde BERT genutzt, um Definitionen zu extrahieren (Kumar et al. [2020]). Dabei wurde ein F1-Wert von 0,974 erzielt.

Domänenspezifische Metadatenextraktion, zum Beispiel mit dem „Word2vec“-Algorithmus, erzielte auf geowissenschaftlichen Artikeln einen F1-Wert von 40,7% in der Arbeit von Qiu et al. [2019]. Dort wurde auch angemerkt, dass ein größerer Trainingsdatensatz die Ergebnisse der Extraktion verbessert. sheng WANG et al. [2019] extrahierten verschiedene Metadaten aus geowissenschaftlichen Artikeln. Sie nutzten dafür einen regelbasierten Ansatz, der dem in dieser Arbeit verwendeten Ansatz der Volltextsuche ähnelt. Sie konnten damit einen F1-Wert von über 95% auf Bodenarten erzielen. Das Problem der Extraktion von uneinheitlichen Daten tritt nicht nur in den Bodenwissenschaften

auf (Boukhers et al. [2019]). Ma et al. [2021] nutzten BERT, um Textzusammenfassungen zu bodenwissenschaftlichen Artikeln zu erzeugen. Neben BERT wurden auch andere Deep Learning-Verfahren genutzt, um Daten aus bodenwissenschaftlichen Artikeln zu extrahieren (unter anderem Qiu et al. [2020], Luo et al. [2018], Wang et al. [2021], und Holden et al. [2019]).

Die in dieser Arbeit verwendeten Data Augmentation-Methoden wurden teilweise als „Easy Data Augmentation Techniques“ von Feng et al. [2021] genutzt. Die dort vorgestellte Methode „Random Deletion“ wird in dieser Arbeit als „Löschen“ verwendet, allerdings nur direkt auf Koordinaten, nicht auf ganze Sätze. Zusätzlich dazu ist das Austauschen von Koordinaten mit Koordinaten aus anderen Paragraphen, was in dieser Arbeit als „Koordinatenersetzung“ bezeichnet wird, ähnlich zu der „Synonym Replacement“-Methode der Arbeit von Feng et al. [2021]. Bei dieser werden Wörter durch Synonyme ersetzt, um neue Paragraphen zu erzeugen. Auch „Random Swap“ aus der Arbeit von Feng et al. [2021] wird in dieser Arbeit ähnlich als Data Augmentation-Methode „Permutation“ genutzt, bei welcher einzelne Koordinatenanteile miteinander tauscht. Die dort verwendeten Methoden verbessern die Klassifikationsfähigkeit jeweils leicht. Wullach et al. [2021] konnten mit der Synthese neuer Trainingsbeispiele mit Hilfe von GPT-2 den F1-Wert von BERT bei der Klassifikation von Hassrede um etwa 0,1 erhöhen.

BERT wurde schon im Multi-Task-Learning eingesetzt, bei medizinischen Daten, von Mulyar and McInnes [2020]. Dort wird festgestellt, dass Single-Task-Modelle im Allgemeinen bessere Ergebnisse liefern als Multi-Task-Modelle. Schulz et al. [2018] haben gezeigt, dass Multi-Task-Learning-Methoden bessere Ergebnisse als Single-Task-Learning-Methoden erzielen können, wenn eine der Klassen nur selten vertreten ist. Dieses Ergebnis ist für diese Arbeit relevant, da auch die Klasse der Koordinaten verhältnismäßig selten im dieser Arbeit vorliegendem Datensatz vorhanden ist. Allerdings wurde das Ergebnis mit LSTM-Modellen zur Extraktion von Argumentationsstrukturen erzielt. Ob sich die Ergebnisse auf Klassifikationen mit Transformern übertragen lassen, ist unklar. Peng et al. [2020] zeigten, dass BERT im medizinischen Kontext durch Multi-Task-Learning eine Verbesserung von bis zu 2% erzielen konnte, allerdings nicht mit allen Trainingsdatensätzen. Jia et al. [2021] stellen fest, dass ein Multi-Task-Learning-Ansatz mit BERT und DistilBERT in vielen Fällen bessere Ergebnisse liefert als der jeweilige Single-Task-Learning-Ansatz.

Kapitel 3

Methoden

In dieser Arbeit werden zwei verschiedene Arten von Metadaten untersucht - metrische Daten und nominale Daten. Für eine Extraktion gesuchte Daten werden in dieser Arbeit „relevante Daten“ genannt. Metrische Daten sind Metadaten, deren allgemeine Form bekannt ist, sich aber bei jedem Auftauchen unterscheiden kann und deren Werte sich nicht alle konkret auflisten lassen. Beispiele für metrische Daten sind Koordinaten, Temperaturen, Datumsangaben, Längenangaben, oder Zeitangaben. Metrische Daten folgen also oft keinem klaren Muster und können daher nicht einfach in Texten gefunden werden. Nominale Daten sind Metadaten, deren Form und Werte im Voraus bekannt und auflistbar sind. Beispiele für nominale Daten sind Nutzpflanzen, Länder, Bodenarten, Rohstoffe, oder Monatsnamen. Da diese auflistbar sind, sind Wortlisten verfügbar, die relevante Metadaten auflisten. Teilweise ist die Vollständigkeit solcher Listen allerdings unklar. In dieser Arbeit werden Koordinaten als metrische Daten, und Bodenarten, Nutzpflanzen und Bodentexturen als nominale Daten untersucht. Um die Güte einer Klassifizierung zu bewerten, wird die F1-Funktion genutzt.

In dieser Arbeit wird eine teilwortbasierte Zerlegung durchgeführt, das heißt, Wörter im Paragraphen werden teilweise in mehrere Teilwörter zerlegt. Der dafür genutzte Algorithmus ist der WordPiece-Algorithmus (Wu et al. [2016]). Diese Zerlegung wird durchgeführt, damit ein künstliches neuronales Netz mit den Paragraphen arbeiten kann. Jedes Netz hat ein eigenes Vokabular. Das Vokabular besteht aus Wörtern, die den Anfang eines zu zerlegenden Wortes bilden können, und Wörtern, die nicht zu Beginn eines zu zerlegenden Wortes stehen können. Dadurch kann man erkennen, wie die Teilwörter in Relation zueinander stehen - falls ein Teilwort aus den Wörtern, die nur zu Beginn stehen können, stammt, gehört das vorige Teilwort zu einem anderen Wort. Die Wörter, die nicht zu Beginn stehen können, sind meistens durch eine zusätzliche Zeichenkette erweitert, die in normalen Texten in der Regel nicht

vorkommt, zum Beispiel „##“. So würde mit dem Beispielvokabular

$$a, \#\#a, b, \#\#b, c, \#\#c, d, \#\#d, ba, \#\#cd$$

der Textabschnitt „ab. bacd cd aba!“ wie folgt in einzelne Teilworte zerlegt werden:

$$a, \#\#b, ba, \#\#cd, c, \#\#d, a, \#\#b, \#\#a$$

Die kleinste Einheit im Beginn- und Folgevokabular müssen jeweils alle möglichen, einzelnen Zeichen der Sprache sein, um alle Wörter abbilden zu können. Die Teilwörter werden auch Token genannt.

3.1 Künstliche neuronale Netze

Künstliche neuronale Netze suchen nicht nach einzelnen Wörtern einer Liste. Sie arbeiten stattdessen mit ganzen Paragraphen, um entweder den Paragraph selbst oder einzelne Token davon zu klassifizieren. Daher sind sie geeignet, sowohl nominale Daten als auch metrische Daten zu extrahieren. In dieser Arbeit wird als Grundlage für die verwendeten Netze BERT verwendet (Devlin et al. [2019]). BERT ist ein Transformer, das heißt ein Netz, das ausschließlich mit „Attention“ (Aufmerksamkeit) arbeitet (Vaswani et al. [2017]). Die Attention zwischen zwei Token ist ein Skalar, der angibt, wieviel Aufmerksamkeit der Zieltoken im Bezug auf die Klassifikation vom Starttoken erhalten hat. Jeder Token hat zu jedem anderen Token einen Attentionwert. Bei der Zerlegung eines Paragraphen in Token entstehen zusätzlich ein CLS-Token, welches bei der Sequenzklassifizierung des gesamten Paragraphen genutzt wird, und ein SEP-Token, welches für das Ende des Paragraphen steht.

Ein BERT-Modell besteht aus mehreren Schichten, wobei jede Schicht aus verschiedenen „Köpfen“ besteht, welche parallel die Berechnungen zur Attention durchführen. Die Ergebnisse der Köpfe werden zusammengefasst als Ergebnis der Schicht betrachtet. Da jeder Kopf eine eigene Attention berechnet, kann die Attention potentiell in jeder Schicht des Transformers unterschiedlich sein. Das in dieser Arbeit verwendete BERT-Modell nutzt sechs Schichten und pro Schicht jeweils 12 Köpfe. Die Attentionwerte werden in dieser Arbeit genutzt, um die Extraktion aus Paragraphen, die als „Relevante Daten enthaltend“ klassifiziert sind, zu ermöglichen. Es soll damit überprüft werden, ob die relevanten Daten hohe Aufmerksamkeit durch das Netz bekommen, und falls ja, ob diese Daten direkt extrahiert werden können. Damit künstliche neuronale Netze überhaupt mit hoher Genauigkeit klassifizieren können, müssen sie trainiert werden. Die dafür benötigten Trainingsdaten sind Paragraphen, für die eine Klassifizierung bekannt ist. Um diese erste Klassifizierung durchzuführen, wird die Volltextsuche genutzt.

3.2 Volltextsuche

Bei der Volltextsuche wird der vollständige Text nach bekannten, relevanten Wörtern durchsucht, und falls es eine Übereinstimmung gibt, wird das gefundene Wort ausgegeben. Konkret ist die Volltextsuche nach nominalen Daten einfacher als nach metrischen Daten, da sich für Erstere Wortlisten erstellen lassen. Für metrische Daten sind reguläre Ausdrücke notwendig, die allerdings auch falsch-positive Ergebnisse liefern können. In dieser Arbeit wird die Volltextsuche verwendet, um eine grundlegende Klassifizierung des Datensatzes zu erzeugen. Daher ist es wichtig, die Grenzen dieser Variante zu beachten. Ein Problem der Volltextsuche ist die Abhängigkeit von Wortlisten oder Suchmustern. Falls es keine Wortliste gibt, ist eine Volltextsuche nicht durchführbar. Auch ist die Vollständigkeit dieser Listen ein Problem, da es nicht direkt auffällt, falls eine Liste unvollständig ist. In diesem Fall würden relevante Wörter einfach nicht gefunden werden. Auch hat die einfache Volltextsuche eine geringe Fehlertoleranz, ein falschgeschriebenes Wort wird mit einer einfachen Volltextsuche nicht gefunden. Werden reguläre Ausdrücke genutzt, können Fehler gefunden werden, es steigert aber auch die Falsch-Positiv-Rate. Auch werden Wörter unabhängig vom Kontext gefunden. So würde eine Suche nach „Wetter“ neben dem meteorologischen Phänomen auch die Stadt „Wetter“ in Nordrhein-Westfalen finden. Bei der Volltextsuche ist garantiert, dass ein gefundenes Wort auch immer in einer Wortliste steht. Somit kann ein gefundenes Wort direkt für die Weiterverarbeitung ausgegeben werden.

Eine Extraktion ausschließlich mit einer Volltextsuche ist also möglich, aber nicht unbedingt vollständig. Dabei wird in jedem Paragraphen nach den relevanten Wörtern gesucht. Dieser Ansatz funktioniert allerdings nicht nur auf Paragraphenebene, sondern auch auf Tokenebene. Eine Extraktion auf Tokenebene arbeitet mit dem in die Grundwörter des Netzes zerlegten Paragraphen. Dabei werden einzelne Token in vordefinierte Klassen eingeteilt. Um Token mit der Volltextsuche zu finden, wird in dem in Token zerlegten Artikel nach den in Token zerlegten Wörtern der Wortlisten gesucht. Durch die Volltextsuche auf Tokenebene können Trainingsdaten bereitgestellt werden, die direkt einzelne Wörter klassifizieren, anstatt nur ganze Paragraphen. Damit ist eine direkte Extraktion möglich. Sowohl die Extraktion auf Paragraphenebene als auch die Extraktion auf Tokenebene arbeitet mit in Token zerlegten Paragraphen. Bei dieser Zerlegung werden in dieser Arbeit zwei Varianten genutzt.

3.3 Textzerlegung

Ein Problem beim Zerlegen von Textabschnitten in Token ist die Zuweisung von Labeln zu einzelnen Token. Im Allgemeinen funktioniert das gut, falls

aber mehrere Abschnitte eines Wortes unterschiedlich gelabelt werden sollen (zum Beispiel, weil durch Artefakte in der Umwandlung ein Leerzeichen nicht erkannt wurde, und so mehrere Wörter als ein Wort erkannt werden), ist es schwierig, diese korrekt zu trennen. Ein konkretes Beispiel ist die Koordinate „23° 24'“, die im umgewandelten Text als „23824'“ vorkommt, das Gradzeichen und das folgende Leerzeichen wurde also als „8“ identifiziert. Diese „8“ sollte nicht als Koordinate klassifiziert werden, da sie nicht zur Koordinate gehört. Um dieses Problem zu lösen, werden in dieser Arbeit zwei Ansätze verwendet.

Leerzeichentrennung

Eine Lösung, die bei Koordinaten funktioniert, ist die Leerzeichentrennung. Leerzeichentrennung wird in dieser Arbeit durchgeführt, indem alle Zeichenfolgen, in denen Zahlen vorkommen, durch Leerzeichen getrennt werden. Dadurch ist garantiert, dass jedes Zeichen als eigener Token erkannt wird. Diese Token können dann einzeln mit einem Label versehen werden. Nachteil dieser Methode ist, dass alle Wörter mit Zahlen getrennt werden. Als Beispiel würde aus einer korrekten „12“ eine „1“ und „2“. Aus der Koordinate „23824'“ mit fehlerhafter „8“] würde allerdings «2 3 8 2 4 ‘ ‘», wobei jedes Zeichen ein eigener Token ist. Dadurch lässt sich die falsche Acht für die Trainingsdaten einfach anders klassifizieren.

Ziffern-Tokenizer

Die zweite in dieser Arbeit verwendete Lösung ist die Modifizierung des Tokenizers, also des Algorithmus, der die Texte zerlegt. Dabei wurde das Vokabular verkleinert, so dass Zahlen nur noch einstellig vorkommen können. Dafür werden Token entfernt, die sowohl Zahlen als auch Buchstaben enthalten, und auch alle Token, die mehr als eine Zahl enthalten. Die zerlegte „12“ ist somit „1“ und „##2“. Problematisch an dieser Umsetzung ist, dass Artefakte, die als Zahlen erkannt wurden, nicht aussortiert werden können, sondern als zum Wort gehörend betrachtet werden. Als konkretes Beispiel kann wieder die Koordinate „23824'“ dienen, die mit dem Ziffern-Tokenizer umgewandelt die Folge «2, ##3, ##8, ##2, ##4, ##'» ergibt. Da der Tokenizer modifiziert wurde, um Ziffern deutlicher zu zerlegen, wird er in dieser Arbeit „Ziffern-Tokenizer“ (Z-Tokenizer) genannt. Der unmodifizierte Tokenizer wird in dieser Arbeit Standard-Tokenizer genannt.

Somit können durch die Volltextsuche relevante Wörter und relevante Paragraphen gefunden werden. Die so klassifizierten Paragraphen können dann als Trainingsdaten für künstliche neuronale Netze verwendet werden. Dabei ist es nicht allzu wichtig, dass wirklich alle Vorkommen von relevanten Wörtern korrekt klassifiziert sind, sondern nur, dass ein vergebenes Label auch ein

Token dieser Klasse klassifiziert, damit das Netz keine falschen Wörter und Kontexte lernt. Die Falsch-Positiv-Rate soll daher gering gehalten werden. Die Probleme der Volltextsuche sind also für eine erste Klassifizierung kaum nachteilig, die Falsch-Negativ-Rate für relevante Token ist potentiell etwas höher, was aber bei der letztlichen Klassifikation durch ein Netz keine Probleme bereiten sollte. Durch die potentiell wenigen korrekt-positiven Klassifikationen gibt es allerdings Probleme: Wenn die Trainingsdatenmenge klein ist, kann das Netz eventuell keine korrekte Klassifikation durchführen. Das war in dieser Arbeit bei Koordinaten der Fall. Daher werden verschiedene Methoden der Trainingsdatenerzeugung untersucht, die die Klassifikation verbessern sollen.

3.4 Trainingsdatenerzeugung

Das Erzeugen von Trainingsdaten für die Metadatenextraktion ist ein Problem - gäbe es ein verlässliches Verfahren, um beliebige Metadaten korrekt zu klassifizieren, könnte man diese Methode verwenden, statt künstliche neuronale Netze dafür zu nutzen. Die Volltextsuche liefert in dieser Arbeit die Grundlage der Trainingsdaten. Diese können aber noch auf verschiedene Arten modifiziert werden.

Auswahl

Bei der endgültigen Auswahl der Textabschnitte für das Training werden in dieser Arbeit zwei Varianten genutzt. Bei der ersten Mtehode werden alle Paragraphen genutzt, also Paragraphen, die relevante Daten enthalten, und solche, die keine enthalten. Der Vorteil ist, dass die Trainingsmenge hier großtmöglich ist. Allerdings können hier die relevanten Paragraphen/Token eventuell nur sehr selten auftreten, so dass das künstliche neuronale Netz zu wenig tatsächlich relevante Beispiele sieht. Bei der anderen Methode werden nur Paragraphen gewählt, die für das Problem relevante Token enthalten. Der Vorteil ist, dass das Netz dann mit jedem Paragraphen lernt, in welchem Kontext die relevanten Token auftreten. Bei wenig relevanten Daten kann dadurch die Trainingsdatenmenge klein werden. Das kann zu einer Überspezialisierung führen. Dieser Nachteil lässt sich aber durch nachträgliche Bearbeitung des Trainingsdatensatzes vermeiden.

Nachträgliche Bearbeitung

Es gibt verschiedene Möglichkeiten, die Trainingsdaten zu verändern, um aus wenigen Daten einen größeren Datensatz zu erhalten. In dieser Arbeit werden

diese Varianten beim Trainieren von künstlichen neuronalen Netzen zur Koordinatenerkennung genutzt, da die Anzahl der Paragraphen mit Koordinaten, die in der Volltextsuche gefunden wurden, relativ gering ist. Aus einem Paragraphen können mehrere entstehen, wenn diese in ihrer Form und Platzierung von einzelnen Token verändert werden. Dadurch kann das künstliche neuronale Netz nicht die Positionen der Label für einen gewissen Textabschnitt auswendig lernen, wodurch eine Überspezialisierung verhindert wird. Auch wird die Menge der Trainingsdaten dadurch größer. In dieser Arbeit soll überprüft werden, ob diese Änderungen auch einen positiven Effekt auf die Klassifizierung durch die Modelle haben. Konkret werden dafür drei Methoden genutzt, die in dieser Arbeit auf ihre Auswirkungen auf die Klassifikationsgenauigkeit untersucht werden sollen.

Kürzen Da bekannt ist, welche Token im Paragraphen relevant sind, können Paragraphen vor und nach diesen Token gekürzt werden. Dafür wird der Paragraph von einem String in eine Liste von Wörtern überführt, wobei alles, was zwischen zwei Leerzeichen steht, als ein Wort betrachtet wird. So würde aus dem String

`Goberi (12 • 58 N, 2 • 50 E).`

die folgende Wortliste entstehen:

- Goberi
- (12
- •
- 58
- N,
- 2
- •
- 50
- E).

Nachdem der Paragraph so umgewandelt wurde, werden die konkreten Trennungsstellen zufällig gewählt, jedoch wird beachtet, dass Koordinaten zusammenhängend bleiben. Dadurch bleibt der Kontext für die relevanten Token erhalten, der Paragraph sieht allerdings anders aus. Damit soll verhindert werden, dass sich das Netz überspezialisiert und die Paragraphen auswendig lernt. Als Beispiel sei hier der Paragraph aus Tabelle 1.1 genutzt.

> The four experimental sites selected for the present study were part of a large factorial crop rotation experiment arranged in a split-plot design with two replications. The experiment was conducted under rainfed conditions from 1995 to 1999 in the Sahelian, Sudanian and Guinean zones of sub-Saharan West Africa (Bagayoko et al., 2000;Buerkert et al., 2000). The soils were Psammentic Paleustalfs at Kara Bedji (13 • 15 N, 2 • 32 E) and Goberi (12 • 58 N, 2 • 50 E) in the southern Sahelian zone and an Arenic Kandiustalf at Gaya-Bengou (11 • 59 N, 3 • 32 E) in the Sudanian zone of Niger and a Haplustalf at Fada-Kouaré (11 • 59 N, 0 • 19 E) in the Sudanian zone of Burkina Faso. With their range of soil chemical properties they are typical for this region (Table 1).

Bei diesem Beispiel ist

> (13 • 15 N, 2 • 32 E) and Goberi (12 • 58 N, 2 • 50 E) in the southern Sahelian zone and an Arenic Kandiustalf at Gaya-Bengou (11 • 59 N, 3 • 32 E) in the Sudanian zone of Niger and a Haplustalf at Fada-Kouaré (11 • 59 N, 0 • 19 E)

der Abschnitt des Strings mit relevanten Daten, welcher nicht zerlegt wird, um den Kontext für die Koordinaten zu erhalten. Falls es nur eine Koordinate im Paragraphen gibt, besteht der Abschnitt mit relevanten Daten nur aus diesen Koordinaten. Nach einem Kürzen kann beispielsweise folgender Paragraph übrig bleiben:

> in a split-plot design with two replications. The experiment was conducted under rainfed conditions from 1995 to 1999 in the Sahelian, Sudanian and Guinean zones of sub-Saharan West Africa (Bagayoko et al., 2000;Buerkert et al., 2000). The soils were Psammentic Paleustalfs at Kara Bedji (13 • 15 N, 2 • 32 E) and Goberi (12 • 58 N, 2 • 50 E) in the southern Sahelian zone and an Arenic Kandiustalf at Gaya-Bengou (11 • 59 N, 3 • 32 E) in the Sudanian zone of Niger and a Haplustalf at Fada-Kouaré (11 • 59 N, 0 • 19 E) in the Sudanian zone of Burkina Faso. With their range of soil chemical properties they are

Sei P ein in eine Wortliste zerlegter Paragraph mit $P = p_1, p_2, ..., p_m$, wobei Wort p_i an Stelle i steht. Diese wird in drei Teillisten zerlegt: P_v, die Teilliste vor den relevanten Daten, P_r, die Teilliste mit relevanten Daten, und P_n, die Teilliste nach relevanten Daten.

$$P \mapsto P_v, P_r, P_n$$

wobei

$$P_v = p_1, p_2, ..., p_v$$

$$P_r = p_{v+1}, p_{v+2}, ..., p_{v+j}$$

$$P_n = p_{v+j+1}, p_{v+j+2}, ..., p_{v+j+b}$$

mit p_i wieder das Wort an Stelle i in der Wortliste des Originalparagraphens P, weswegen $p_{v+j+b} = p_m$ und $v + j + b = m$ gilt. Dann werden zwei Zahlen $x_v \in \{1, 2, ..., v\}$ und $x_n \in \{v + j + 1, v + j + 2, ..., v + j + b - 1, v + j + b\}$ zufällig gewählt, und P_v und P_n wie folgt geändert:

$$P_v \mapsto p_{x_v}, p_{x_v+1}, ..., p_v = P_v^e$$

$$P_n \mapsto p_{v+j+1}, p_{v+j+2}, ..., p_{x_n} = P_n^e$$

Anschließend werden die neuen Einzelteile mit dem unveränderten relevanten Teil wieder zu einer Liste zusammengesetzt:

$$P_v^e, P_r, P_n^e \mapsto p_{x_v}, p_{x_v+1}, ..., p_{v+1}, p_{v+2}, ..., p_{v+j}, p_{v+j+1}, ..., p_{x_n}$$

Da die p_i aus einer Wortliste stammen, die durch Leerzeichentrennung entstanden ist, lässt sich aus dieser Liste wieder leicht ein String konstruieren, indem man die einzelnen Wörter der Liste mit Leerzeichen konkateniert:

$$p_{x_v} + „“ + p_{x_v+1} + „“ + ... + „“ + p_{x_n} = P^e$$

P^e kann dann als Trainingsdatum genutzt werden, da eine Klassifizierung der einzelnen Wörter schon für P bekannt war.

Permutieren Da die Reihenfolge von einzelnen Koordinatenanteilen wichtig ist, kann die Reihenfolge von Zeichen innerhalb einer Koordinate gemischt werden, um die Trainingsdatenmenge mit ähnlichen Negativbeispielen zu erweitern. Zusätzlich verlieren sie ihre Koordinaten-Klassifizierung. Dadurch soll das Netz erkennen, dass die Reihenfolge der Zeichen relevant ist, und nicht nur ihr reines Vorkommen. Ein Beispiel hierfür sind die Koordinaten

13 • 15 N, 2 • 32 E

Diese könnten nach Anwenden dieser Methode beispielsweise wie folgt wieder in den Paragraphen eingefügt werden:

N• 1 E2 •,5 312 3

Konkret wird also eine Koordinate als String $C = c_1, c_2, ..., c_n$ betrachtet, wobei c_i das Zeichen an Stelle i ist. Dieser wird dann umgewandelt in einen String $C_S = c_{S(1)}, c_{S(2)}, ..., c_{S(n)}$, wobei S eine zufällige Permutation ist. Die Permutation wird für jede Koordinate neu zufällig gewählt, so dass die gleiche Koordinate nicht immer gleich permutiert wird.

$$C \mapsto S(C) = C_S$$

Löschen Analog zur Permutation werden einzelne Zeichen aus Koordinaten gelöscht, und der Rest der Koordinate verliert sein relevantes Label. Dadurch lernt das Netz, dass unvollständige Koordinaten nicht als relevant markiert werden sollen. So könnte aus

13 • 15 N, 2 • 32 E

13 • N, • 32

werden. Auch hier wird wieder eine Koordinate als String $C = c_1, c_2, ..., c_n$ (mit c_i das Zeichen an Stelle i) abgebildet auf einen String $L = c_{u_1}, c_{u_2}, ..., c_{u_j}$, wobei $u_i \in \{1, ..., n\} \setminus \{r_1, r_2, ..., r_t\}$ mit $r_i \in \{1, ..., n\}$ und $u_1 < u_2 < .... < u_j$, sowie $j = n - t$. Dabei werden sowohl die r_i als auch t mit $n \geq t \geq 1$ zufällig gewählt. Es wird also eine zufällige Anzahl von Zeichen gelöscht, und nur die u_i bleiben übrig.

Erzeugung von künstlichen Daten

Falls die Anzahl der Koordinaten zu gering ist, können neue Paragraphen erzeugt werden, die den Trainingsdatensatz erweitern. Um den Kontext für Koordinaten zu erhalten, können die Paragraphen nicht komplett zufällig erzeugt werden. Daher wurden in dieser Arbeit zwei Varianten verwendet.

Vollständig zufällige Koordinaten Bereits gefundene Koordinaten werden auf ihre Störzeichen hin analysiert, also Zeichen, die innerhalb der Koordinate stehen, aber selbst kein Teil der Koordinate sind. Jede Art von Störzeichen werden extrahiert. Diese werden verwendet, um künstlich neue Koordinaten zu erzeugen, indem sie mit zufälligen Zahlen und Längengrad/Breitengrad-Kennzeichnern gemischt werden und in Koordinatenform gebracht werden. Die so erzeugten Koordinaten werden dann in einen Paragraphen eingefügt, der bereits Koordinaten enthält. Diese werden dafür entfernt, und die neuen Koordinaten ersetzen sie an ihrem Platz. Dadurch sieht das Netz andere relevante Daten, die aber in Kontext und Aufbau realen Daten ähnlich sind. Diese

Variante funktioniert nur mit metrischen Daten, da zufällig neue Kombinationen von Daten entstehen, und keine ganzen Wörter ausgetauscht werden.

Sei N die Menge aller im Datensatz vorkommenden Störzeichen, wobei N_i das i-te Störzeichen ist. $N_0 = N \cup \{„“\}$ ist die Menge der Störzeichen, erweitert um ein leeres Element. Sei $Z = \{0, 1, 2, ..., 89\}$ die Menge der als Koordinate möglichen Zahlen. Und sei $F(K)$ eine Funktion, die zufällig ein Element der Menge K auswählt. Seien $L = \{„N“, „S“\}$ und $O = \{„W“, „E“\}$ Mengen, die die Breiten- und Längen-Kennzeichner beinhalten. Zuletzt sei $E = F(N), F(N_0)$ die Kombination eines zufälligen Störzeichens, und einem zufälligen Zeichen aus N_0, das wieder ein Störzeichen oder nichts sein kann. Dann hat eine vollständig zufällige Koordinate ohne Sekundenstelle die Form

$$F(Z), E, F(Z), E, F(L), E, F(Z), E, F(Z), E, F(O)$$

Eine vollständig zufällige Koordinate mit Sekundenstelle hat dementsprechend die Form

$$F(Z), E, F(Z), E, F(Z), E, F(L), E, F(Z), E, F(Z), E, F(Z), E, F(O)$$

Eine solche Koordinate könnte beispielsweise so aussehen, falls $N = \{@, ¶\}$:

4¶8¶@15@N¶16@23¶¶42@W

Teilweise zufällige Koordinaten Im Gegensatz zur vollständig zufälligen Erzeugung von Koordinaten werden nur einzelne Anteile der Koordinaten ausgetauscht. Konkret werden Zahlen und Breiten-/Längen-Kennzeichner aus dem passenden Bereich zufällig erzeugt. Die Form inklusive Störzeichen der originalen Koordinaten bleibt somit erhalten, der Inhalt selbst ist allerdings anders, wodurch der Paragraph für das Netz etwas anders aussieht. So könnte beispielsweise aus der Koordinate

13 • 15 N, 2 • 32 E

die teilweise zufällig erzeugte Koordinate

0 • 9 S, 1 • 2 E

erzeugt werden.

Austauschen von Koordinaten Anstatt künstlich Koordinaten zu erzeugen, werden alle Koordinaten aus den Trainingsdaten extrahiert. Beim Training selbst wird dann zufällig eine Koordinate aus dieser Menge gewählt und in den Trainingsparagraph eingesetzt, an die Stelle, an der vorher eine Koordinate stand. Dadurch bleibt der Kontext erhalten, und auch die Koordinaten sind garantiert realistisch. Als Beispiel könnte der Paragraph

> The four experimental sites selected for the present study were part of a large factorial crop rotation experiment arranged in a split-plot design with two replications. The experiment was conducted under rainfed conditions from 1995 to 1999 in the Sahelian, Sudanian and Guinean zones of sub-Saharan West Africa (Bagayoko et al., 2000;Buerkert et al., 2000). The soils were Psammentic Paleustalfs at Kara Bedji (13 • 15 N, 2 • 32 E) and Goberi (12 • 58 N, 2 • 50 E) in the southern Sahelian zone and an Arenic Kandiustalf at Gaya-Bengou (11 • 59 N, 3 • 32 E) in the Sudanian zone of Niger and a Haplustalf at Fada-Kouaré (11 • 59 N, 0 • 19 E) in the Sudanian zone of Burkina Faso. With their range of soil chemical properties they are typical for this region (Table 1).

nach dem Austauschen wie folgt aussehen:

> The four experimental sites selected for the present study were part of a large factorial crop rotation experiment arranged in a split-plot design with two replications. The experiment was conducted under rainfed conditions from 1995 to 1999 in the Sahelian, Sudanian and Guinean zones of sub-Saharan West Africa (Bagayoko et al., 2000;Buerkert et al., 2000). The soils were Psammentic Paleustalfs at Kara Bedji (52852VN, 13853VE) and Goberi (13 • 15 N, 2 • 32 E) in the southern Sahelian zone and an Arenic Kandiustalf at Gaya-Bengou (42°07J N and 74°15J W) in the Sudanian zone of Niger and a Haplustalf at Fada-Kouaré (72.37N, 126.47E) in the Sudanian zone of Burkina Faso. With their range of soil chemical properties they are typical for this region (Table 1).

Diese drei Methoden können genutzt werden, um die Trainingsdatenmenge zu vergrößern. Mit diesen Änderungen an den Trainingsdaten kann begonnen werden, sich Netze genauer anzuschauen.

3.5 Extraktion durch künstliche neuronale Netze

Mit Hilfe der so erzeugten Trainingsdaten können künstliche neuronale Netze trainiert werden, um Metadaten aus Paragraphen zu extrahieren. Dafür wurden in dieser Arbeit zwei verschiedene Ansätze untersucht: Netze, die eine Sequenzklassfikation durchführen, und Netze, die eine Tokenklassifikation durchführen.

3.5.1 Sequenzklassifikation

Bei der Sequenzklassifikation wird einem Paragraphen ein einzelnes Label zugewiesen - entweder er enthält die gesuchten Daten, oder er enthält diese Daten nicht. Die Sequenzklassifikation ist ursprünglich nicht dazu gedacht, einzelne, relevante Wörter zu extrahieren. In dieser Arbeit wird allerdings untersucht, ob es dennoch erfolgreich möglich ist. Dafür werden drei verschiedene Methoden getestet. Die Sequenzklassifikation benötigt mit diesen Methoden noch immer manuelle Nachbearbeitung, aber sie kürzt die Zeit, die auf einen Artikel verwendet werden muss.

Die Attention-Methode

Für diese Methode werden die Attentionwerte pro Token pro Schicht berechnet. Da eine Sequenzklassifikation durchgeführt wird, wird der CLS-Token genutzt. Der CLS-Token gibt die Klassifikation des Paragraphen an. Wird also ein Paragraph durch das Netz als „Relevante Daten enthaltend“ klassifiziert, haben Token des Paragraphen Einfluss auf das CLS-Token genommen, so dass dieser diese Klassifizierung angibt. Daher wird jeweils der Attentionwert genutzt, der angibt, wieviel Einfluss ein entsprechender Token auf das CLS-Token hatte, für alle Token je Schicht. Sei $A^{(j)}(i)$ dieser skalare Attentionwert von Token i in Schicht j. Sei $R_r^{(j)}(i)$ der skalare Attentionwert von Kopf r auf Schicht j von Token i, und Q die Anzahl der Köpfe pro Schicht, gilt dann:

$$A^{(j)}(i) = \sum_{r=1}^{Q} R_r^{(j)}(i)$$

Um diese Attentionwerte dann für die Extraktion nutzen zu können, werden die zehn Token mit den höchsten Attentionwerten pro betrachteter Schicht markiert. Dann wird das Vorkommen der einzelnen markierten Token über alle betrachteten Schichten gezählt. Die Token, die hierbei die maximale Anzahl aufweisen, werden als relevantes Datum extrahiert. Damit soll untersucht werden, ob sich relevante Token stärker auf die Klassifizierung des Paragraphen

auswirken. Es wird erwartet, dass das Netz bei als relevante Daten enthaltend klassifizierten Paragraphen die größte Aufmerksamkeit auf eben diese relevanten Daten gelegt hat.

Die Halbierungs-Methode

Wird ein Paragraph als relevante Daten enthaltend klassifiziert, wird der Paragraph in zwei Teile geteilt, welche wieder durch das Netz klassifiziert werden. Dieser Vorgang wird so lange wiederholt, bis beide neuen Teilparagraphen nicht mehr als relevant klassifiziert werden, und die Zusammensetzung dieser Teilparagraphen wird als relevantes Datum betrachtet.

Sei $P = p_1, p_2, ..., p_n$ ein Paragraph, der in eine Wortliste überführt wurde, indem alle Wörter an Leerzeichen getrennt wurden, p_i ist dann das Wort an Stelle i. Sei H_u eine Halbierungsfunktion mit $H_u(P) = W_1, W_2$, wobei $W_1 = p_1, p_2, ..., p_u$ und $W_2 = p_{u+1}, p_{u+2}, ..., p_n$. Der Paragraph wird also nach Wort u getrennt. Sei $K(P)$ die Klassifikationsfunktion, die 1 ergibt, falls P relevante Daten enthält, sonst 0, und sei $M(P)$ die Ausgabefunktion, die relevante Daten liefert, falls $K(P) = 1$ gilt. Dann lautet die Formel der Halbierungs-Methode:

$$M(W_1, W_2) = \begin{cases} M(H_{\lceil \frac{n}{2} \rceil}(W_1)), M(H_{\lceil \frac{n}{2} \rceil}(W_2)) & : \ K(W_1) = 1, \ K(W_2) = 1 \\ M(H_{\lceil \frac{n}{2} \rceil}(W_1)) & : \ K(W_1) = 1, \ K(W_2) = 0 \\ M(H_{\lceil \frac{n}{2} \rceil}(W_2)) & : \ K(W_1) = 0, \ K(W_2) = 1 \\ W_1, W_2 & : \ K(W_1) = 0, \ K(W_2) = 0 \end{cases}$$

Es soll untersucht werden, ob auf diese Weise relevante Daten automatisiert extrahiert werden können.

Die Attention-Halbierungs-Methode

Die letzte hier vorgestellte Methode ist die Attention-Halbierungs-Methode. Sie kombiniert die Halbierungs-Methode mit den Attentionwerten des Netzes. Diese Methode arbeitet mit den Token des Paragraphen, nicht mit Worten. Zusätzlich wird der Paragraph nicht in zwei etwa gleich große Teile geteilt, sondern der Paragraph wird in zwei Teile geteilt, die etwa gleich große summierte Attentionwerte haben. Dafür werden die Attentionwerte für jeden einzelnen Token berechnet (vgl. Attention-Methode, 3.5.1), und dann wird für jeden möglichen Teilungspunkt die Differenz der Summen der beiden so erzeugten Teilparagraphenattentionwerte ermittelt. Die betragsmäßig kleinste Differenz wird als Teilungspunkt genutzt. Bei den Beispielattentionwerten von 2, 2, 3, 1, 1, und 1, würde dieser Algorithmus diesen Paragraphen nach dem zweiten

Tabelle 3.1: Berechnung des Trennungspunktes am Beispiel. AWS: Attentionwertsumme, Resultat: Teilparagraph 1: 1 & 2, Teilparagraph 2: 3 & 4 & 5 & 6

Token-ID	Attentionwert	Schnitt nach diesem Token		Differenz
		AWS vorne	AWS hinten	
1	2	2	8	-6
2	2	4	6	**-2**
3	3	7	3	4
4	1	8	2	6
5	1	9	1	8
6	1	10	0	10

Token aufteilen (vgl. Tabelle 3.1). Mit dieser Teilung wird dann wieder die Halbierungs-Methode ausgeführt.

Sei also $A(i)$ der Attentionwert von Token i und T der gesuchte Teilungspunkt, also der Token, nach welchem der Paragraph in zwei Teile zerlegt wird. Sei $B \in \{1, ..., n\}$ jeweils ein möglicher Trennungspunkt. Dann gilt:

$$T = \arg\min_{B} \left(\left| \sum_{i=1}^{B} A(i) - \sum_{i=B+1}^{n} A(i) \right| \right)$$

$P = p_1, p_2, ..., p_n$ ist ein Paragraph, der in eine Tokenliste zerlegt wurde, p_i der Token an Stelle i. Dann kann H_u und M analog zur Halbierungs-Methode definiert werden: $H_u(P) = W_1, W_2$ mit $W_1 = p_1, p_2, ..., p_u$ und $W_2 = p_{u+1}, p_{u+2}, ..., p_n$, und $K(P)$ die Klassifikationsfunktion, die 1 ergibt, falls P relevante Daten enthält. Dann gilt, falls $K(P) = 1$:

$$M(W_1, W_2) = \begin{cases} M(H_T(W_1)), M(H_T(W_2)) & : \ K(W_1) = 1, \ K(W_2) = 1 \\ M(H_T(W_1)) & : \ K(W_1) = 1, \ K(W_2) = 0 \\ M(H_T(W_2)) & : \ K(W_1) = 0, \ K(W_2) = 1 \\ W_1, W_2 & : \ K(W_1) = 0, \ K(W_2) = 0 \end{cases}$$

Mit dieser Methode soll untersucht werden, ob die Halbierungs-Methode mit informierterer Berechnung des Teilungspunktes bessere Ergebnisse erzielen kann.

3.5.2 Tokenklassifikation

Auch die Tokenklassifikation wird in dieser Arbeit genutzt, um relevante Metadaten zu extrahieren. Auch dabei werden Modelle genutzt, die auf BERT

beruhen, wie schon bei der Sequenzklassifikation. Grundsätzlich werden in dieser Arbeit noch einmal zwei Varianten der Tokenklassifikation unterschieden: Single-Task-Learning und Multi-Task-Learning. Beim Single-Task-Learning wird ein Modell so trainiert, dass es für genau eine Klasse eine Klassifizierung durchführen kann. Beim Multi-Task-Learning können gleichzeitig mehrere Klassen erkannt werden. Der Unterschied dieser beiden Varianten wird in dieser Arbeit untersucht.

Da es bei der Tokenklassifikation möglich ist, Label an einzelne Token zu vergeben, wird mit dieser zusätzlich noch untersucht, ob sich Koordinatenanteile konkret extrahieren lassen. Dafür werden die einzelnen Koordinatenanteile mit den Labeln „Grad“ für die Grad-Anteile, „Min“ für die Minuten-Anteile, „Sek“ für die Sekunden-Anteile, „Lat“ für den Breiten-Indikator, und „Long“ für den Längen-Indikator gekennzeichnet. Zusätzlich dazu wird noch eine weitere Variante getestet, bei der die Breiten-Koordinaten und die Längen-Koordinaten getrennt gelabelt werden (also „Grad-Lat“ und „Grad-Long“, Rest analog). In dieser Arbeit wird überprüft, mit welcher Genauigkeit einzelne Token extrahiert werden können - nur das allgemeine Label „Koordinate“, genaue Label („Grad“), oder exakte Label („Grad-Latitude“). Außerdem gibt es jeweils ein Label „Noise“ für Störzeichen. Da die einzelnen Klassen große Unterschiede im Vorkommen aufweisen können, können beim Training Gewichte eingeführt werden, die den Einfluss der Klassen auf das Training beeinflussen. Falls Gewichte G genutzt werden, werden diese in dieser Arbeit wie folgt berechnet:

$$G = \frac{D + Y}{D}$$

Dabei ist D die Anzahl der Nicht-Koordinaten-Token, und Y die Anzahl der Koordinatentoken, beides jeweils bezogen auf den Trainingsdatensatz.

Die letzte Änderung, die eine bessere Klassifikation ermöglichen soll, wird bei der Loss-Funktion durchgeführt. Die standardmäßige Loss-Funktion beim Klassifizieren von Paragraphen mit nur einer Klasse mit BERT ist die mittlere quadratische Abweichung, bei mehreren Klassen die Kreuzentropie. Die Standard-Loss-Funktion kann auch modifiziert werden. Dadurch kann man die Berechnung des Losses näher an die konkrete Aufgabenstellung bringen. In dieser Arbeit wird die Standard-Loss-Funktion durch folgende Änderungen erweitert, und wird somit zur modifizierten Loss-Funktion:

Falsche Klassifizierungen Falls das Netz eine Koordinatenklasse klassifiziert, der dazugehörige Token aber kein Koordinatenanteil ist, wird dieses Label mit einem höheren Strafwert belegt, als standardmäßig vergeben wird.

Tabelle 3.2: Beispiele für fehlerhafte Klassifizierung von Token

Koordinatenanteil	13	•	15	N	2	•	32	E
Korrektes Label	Grad	Noise	Min	Lat	Grad	Noise	Min	Long
Fehlende Label	Grad	Noise	**Grad**	Lat	Grad	Noise	**Grad**	Long
Reihenfolge	**Min**	Noise	Grad	Lat	**Min**	Noise	**Grad**	Long

Labelkorrektheit Da Koordinaten im gesamten Datensatz eher selten vorkommen, wird das falsche Verteilen eines solchen Labels in einem Paragraphen, in dem keine Koordinaten vorkommen, stärker bestraft, um Korrektheit zu garantieren.

Fehlende Label Da Koordinatentoken nur in Gruppen auftreten, führt das Fehlen einer notwendigen Klasse zu einer höheren Strafe (vgl. Tabelle 3.2).

Reihenfolge Da Koordinatenanteile immer in bestimmten Reihenfolgen auftreten, und diese Reihenfolge vom Netz nicht eingehalten wurde, wird auch hier ein höherer Strafterm addiert (vgl. Tabelle 3.2).

Durch die modifizierte Loss-Funktion sollen Fehler, die häufiger vorkommen könnten, reduziert werden. Die genauen Auswirkungen dieser Änderungen werden in dieser Arbeit untersucht.

Kapitel 4

Experimente

Die in dieser Arbeit genutzten vortrainierten BERT-Modelle sind „distilbert-base-uncased“ (Sanh et al. [2019]) bei der Sequenzklassifikation und „bert-base-cased“ (Devlin et al. [2019]) bei der Tokenklassifikation. Sie werden bereitgestellt durch die Transformers-Bibliothek von Huggingface (Wolf et al. [2019]).

In dieser Arbeit wird, nachdem die Datengrundlage beschrieben wurde, mit der Sequenzklassifikation begonnen. Dort werden die drei in Abschnitt 3.5.1 vorgestellten Methoden getestet. Bei der Tokenklassifikation wird zuerst der Multi-Task-Learning-Ansatz für die vier zu extrahierenden Metadatenklassen untersucht, und dieser Ansatz mit den jeweiligen Ergebnissen der Single-Task-Learning-Modellen verglichen. Da die Koordinaten hier einen Sonderfall darstellen, wird daraufhin noch untersucht, ob sich durch Änderungen an den Trainingsdaten oder dem Tokenizer die Ergebnisse noch verbessern lassen. Dabei werden die verschiedene Data Augmentation-Methoden untersucht, die in Abschnitt 3.4 vorgestellt wurden. Danach werden die Koordinaten selbst bearbeitet, indem sie durch andere Koordinaten ersetzt werden. Auch dabei werden die Data Augmentation-Methoden aus dem Abschnitt 3.4 genutzt. Zuletzt werden diese Änderungen auch noch mit vollständig randomisierten Koordinaten getestet. Dabei werden zusätzlich die Auswirkungen von Gewichten und des Z-Tokenizers auf die Ergebnisse untersucht.

4.1 Datengrundlage

Die Datengrundlage dieser Arbeit bildet eine Menge von 6.345 englischen, bodenwissenschaftlichen Artikeln aus verschiedenen Magazinen und Zeiträumen[1].

[1] Agriculture, Ecosystems and Environment (2001-2005), Biogeochemistry (2001-2005), Geoderma (2001-2005), Plant & Soil (2001-2005), Soil & Tillage Research (1998-2002), Soil Science (2001-2004, 2010), SSSAJ (2001-2005), Vadose Zone Journal (2002-2006)

Die Artikel liegen als PDF-Dateien vor und werden daher erst in ein strukturiertes Format konvertiert, damit die Arbeit mit Paragraphen möglich ist. Dafür wurde GROBID (GRO [2008–2021]) verwendet, welches die Artikel als XML-Dateien bereitstellt. Diese können dann leicht nach dem Tag für Paragraphen durchsucht werden, um diese für die Verarbeitung vorzubereiten. Paragraphen, die durch ihre Länge nicht mit BERT verarbeitet wurden konnten, wurden ignoriert. Insgesamt wurden so 214.734 Paragraphen gefunden. Durch die Konvertierung traten Konvertierungsfehler auf, wie zum Beispiel der Fehler in Tabelle 1.1.

Aus diesen Daten sollen Metadaten extrahiert werden. In dieser Arbeit wird sich auf folgende Metadatenklassen konzentriert:

- Bodenart
- Textur
- Nutzpflanzen
- Koordinaten

Da Bodenarten, Texturen und Nutzpflanzen nominale Daten sind, existieren dafür Wortlisten, die in dieser Arbeit genutzt werden. Für Koordinaten, die metrische Daten sind, existieren keine Listen. Um aus den Paragraphen Trainingsdaten für die Sequenzklassifikation zu erzeugen, werden für die Kategorien Bodenart, Textur und Nutzpflanze die bereitgestellten Wortlisten genutzt. Diese Wörter werden (unabhängig von Groß- und Kleinschreibung) in einem Paragraphen gesucht. Dabei wird darauf geachtet, dass das gesuchte Wort nicht in einem anderen Wort vorkommt, also tatsächlich eigenständig ist (beispielsweise würde „Unicorn“ keinen Treffer für „corn“ liefern). Paragraphen, in denen Wörter gefunden werden, werden dann für das eigentliche Training des Netzes vorgemerkt.

Da bei Koordinaten keine allgemeinen Wortlisten zur Verfügung stehen, werden hier reguläre Ausdrücke verwendet, um diese zu finden. Die regulären Ausdrücke sind

```
(\d{2}.{1,4}){2}(?:N|S).{1,5}\d.{0,10}\d{2}.{1,4}(?:W|E)
```

und

```
(\d{2}.{1,4}){3}(?:N|S).{1,5}\d.{0,6}\d{2}.{1,4}(?:W|E)
```

Der erste Ausdruck sucht Zeichenkombinationen, die mit genau zwei Zahlen beginnen. Es folgen ein bis vier beliebige Zeichen, danach stehen wieder

zwei Zahlen und vier beliebige Zeichen. Anschließend stehen entweder ein „N“ oder „S“, und dann wird, nach einem bis fünf beliebigen Zeichen, nach einer Zahl gesucht. Zuletzt werden null bis zehn beliebige Zeichen, gefolgt von zwei Zahlen, gesucht, wobei am Ende mit „W“ oder „E“ abgeschlossen wird. Dadurch werden Koordinaten der Form

$$51°29′N, 11°58′E$$

gefunden, also Koordinaten mit sechs Einträgen (Grad, Minute, Breitengrad, Grad, Minute, Längengrad). Der zweite Ausdruck findet Koordinaten mit acht Einträgen, also Grad, Minute, Sekunde, Breitengrad, Grad, Minute, Sekunde und Längengrad, beispielsweise

$$51°29′46″N, 11°56′09″E$$

Durch diese recht eng gewählten Ausdrücke sind falsch-positive Treffer so gut wie ausgeschlossen, wodurch garantiert ist, dass ausschließlich Koordinaten auch als solche klassifiziert werden. In Anschluss an diese Suchen erhält man eine Liste von Paragraphen und den relevanten Metadaten, die jeweils in ihnen gefunden wurden.

Für die Tokenklassifikation wird zuerst genauso vorgegangen. Da man für jeden relevanten Paragraphen eine Liste aller relevanter Daten in diesem Paragraph hat, wird der Paragraph in Token zerlegt, und die gefundenen Daten werden auch in Token zerlegt. Dann wird im in Token zerlegten Paragraphen nach der Folge von Token, die bei der Zerlegung der relevanten Daten entstanden ist, gesucht, und diese Token erhalten das jeweilige Label der Klasse der gefundenen Daten (zum Beispiel „Koordinate“ oder „Textur“). Das wird für alle relevanten Wörter des Paragraphen durchgeführt, für alle Paragraphen. Alle Token, die danach kein Label erhalten haben, erhalten das Nulllabel, das besagt, dass der Token kein relevantes Datum ist. Mit den so gelabelten Daten kann dann das eigentliche Training für die Klassifikation starten.

4.2 Extraktion durch Sequenzklassifikation

Für die Sequenzklassifikation wird „distilbert-base-uncased“ (Sanh et al. [2019]) genutzt, welches eine Sequenz von Wörtern (in diesem Fall den Paragraphen) klassifizieren kann. Als korrekt gewertet wird ein extrahierter Textabschnitt, wenn in diesem ein relevantes Metadatum enthalten ist. Dadurch wird die automatische Extraktion ohne zusätzlichen Aufwand nicht garantiert, aber da die Sequenzklassifikation im Allgemeinen hauptsächlich als Hilfe für die manuelle Extraktion vorgesehen ist, ist dort ist eine Verkürzung des zu lesenden Textes wünschenswert.

Tabelle 4.1: Allgemeine Ergebnisse der Sequenzklassifikation. Betrachtet werden alle Paragraphen.

Modell	Echt-Positiv	Falsch-Positiv	Falsch-Negativ	Echt-Negativ
Soils	3.188	117	38	138.862
Soil-less	1.551	83	1.675	138.896
Koordinaten	422	104	125	141.554

4.2.1 Modelle

Die Sequenzklassifikation wird mit zwei verschiedenen Modellen durchgeführt: einem Soils-Modell und einem Koordinaten-Modell. Da Bodenarten, Texturen, und Nutzpflanzen jeweils auf Wortlisten basieren, werden hier beliebig die Bodenarten als Vertreter dieser Gruppe der nominalen Daten gewählt. Zusätzlich wird bei zwei Methoden noch mit einem Soil-less-Modell getestet, welches überprüfen soll, ob das Modell jeweils nur die Wörter auswendig lernt, oder ob der Kontext, in welchem die Bodenarten auftauchen, auch relevant ist. Dafür werden in den Trainingsparagraphen die Bodenarten vorher entfernt.

Jedes der Modelle wird jeweils nur mit Paragraphen trainiert, die auch das jeweilige Metadatum enthalten. Auf diesen Daten gibt es einen Trainings/Test-Split von 0,75, das heißt, drei Viertel der Daten werden zum Training genutzt, ein Viertel für das Testen. Das Training wird in sechs Epochen mit einer Gruppengröße von vier durchgeführt. Der Tokenizer ist der Standard-Tokenizer. Die Trainingsdaten sind nicht weiter modifiziert worden. Die Auswertung der Daten findet mit den gesamten Daten statt. Ein Paragraph wird als einer Klasse angehörig gelabelt, falls er einen Wert von über 0,9 bei den Bodenartenmodellen und über 0,7 bei dem Koordinatenmodell erreicht.

Betrachtet man die allgemeinen Ergebnisse der Sequenzklassifikation, sieht man, dass das Soils-Modell 3.188 Paragraphen korrekt als Bodenart enthaltend klassifiziert, und es in 38 Paragraphen, die eine Bodenart enthalten, keine Bodenart sieht. Im Vergleich dazu erkennt das Soil-less-Modell 1.551 Paragraphen korrekt als Bodenart enthaltend, 1.675 allerdings nicht. Beide haben eine ähnliche Zahl falsch-positiver Klassifikationen (vgl. Tabelle 4.1).

4.2.2 Halbierungs-Methode

Wie in Abschnitt 3.5.1 beschrieben, wird hier die Halbierungs-Methode genutzt, es wird also ein Paragraph so lange in halbiert, bis man keine relevanten Teile mehr findet, um dann die letzte relevante Sequenz als Metadatum auszugeben. Für die Auswertung werden alle Paragraphen, die relevante Daten enthalten, mit der Halbierungsmethode bearbeitet, unabhängig davon, wie das

Tabelle 4.2: Auswertung der Halbierungs- und der Attention-Methode. Betrachtet werden nur Paragraphen, die ein relevantes Datum enthalten. Ext: Extrahiert, N.: Nicht.

Methode	Klassifikation Modell	Enthaltend		Nicht enthaltend	
		Ext	N. ext	Ext	N. ext
Halbierung	Soils	5.841	139	42	0
	Soil-less	2.934	339	2.722	27
	Koordinaten	282	221	89	43
Attention	Soils	5.433	547	40	2
	Soil-less	2.461	812	2.389	360
	Koordinaten	0	503	0	132

Modell sie vorher klassifiziert hat.

Das Soils-Modell kann 5.883 Bodenarten erfolgreich mit dieser Methode extrahieren, davon sind 42 in Paragraphen, die als „Nicht enthaltend“ bewertet wurden. Das Soil-less-Modell kann 5.656 Koordinaten auf diese Weise extrahieren, davon allerdings 2.722 aus Paragraphen, denen das Modell die Klassifikation „Nicht enthaltend“ gab. Das Koordinaten-Modell kann auf diese Weise 282 Koordinaten extrahieren (vgl. Tabelle 4.2).

4.2.3 Attention-Methode

Als nächstes wird die Attention-Methode untersucht (vgl. Abschnitt 3.5.1). Die Auswertung erfolgt, indem überprüft wird, ob die extrahierten Token in dem relevanten Metadatum auftauchen. Ist das der Fall, wird das Datum als „Extrahiert“ gewertet.

Mit der Attention-Methode kann das Soils-Modell 5.433 Koordinaten korrekt extrahieren. Das Soil-less-Modell kann 2.934 extrahieren. Betrachtet man zusätzlich die Extraktionen aus falsch klassifizierten Paragraphen, kann das Soil-less-Modell 4.850 Koordinaten extrahieren. Das Koordinatenmodell kann mit dieser Methode keine der 635 Koordinaten extrahieren (vgl. Tabelle 4.2).

4.2.4 Attention-Halbierungs-Methode

Als letzte Methode wird hier die Attention-Halbierungs-Methode untersucht (vgl. Abschnitt 3.5.1). Die Auswertung funktioniert analog zur Attention-Methode, da auch hier mit Token gearbeitet wird. Untersucht werden die Attentionwerte der ersten Schicht, der letzten Schicht, und die Summe aller Schichten.

Tabelle 4.3: Auswertung der Attention-Halbierungs-Methode. Betrachtet werden nur Paragraphen, die ein relevantes Datum enthalten. Ext: Extrahiert, N.: Nicht.

Klassifikation		Enthaltend		N. enthaltend	
Modell	Schicht	Ext	N. ext	Ext	N. ext
	1	4.381	1.599	39	3
Soils	6	4.381	1.599	39	3
	1-6	4.381	1.599	39	3
	1	281	222	90	42
Koordinaten	6	277	226	91	41
	1-6	275	228	88	44

Das Soils-Modell kann, unabängig von den Schichten, aus denen die Attention übernommen wurde, jeweils 4.381 Bodenarten aus Paragraphen extrahieren, die als „Enthaltend“ klassifiziert wurden. Beim Koordinaten-Modell gibt es unterschiedliche Werte, abhängig davon, von welcher Schicht die genutzte Attention entnommen wurde. Schicht 1 führt zu 281 extrahierten Koordinaten aus Paragraphen, die als „Enthaltend“ klassifiziert wurden. Aus diesen kann das Modell der Schicht 6-Attention 277 Koordinaten extrahieren. Auch aus „Enthaltend“en Paragaphen kann das Modell, das Attention aller Schichten nutzte, 275 Koordinaten extrahieren (vgl. Tabelle 4.3). Die Unterschiede zwischen den Schichten sind also vorhanden, aber nicht groß.

4.3 Extraktion durch Tokenklassifikation

Die Tokenklassifikation wird mit dem BERT-Grundmodell „bert-base-cased“ (Devlin et al. [2019]) durchgeführt. Mit der Tokenklassifikation werden verschiedene Varianten der Trainingsdatenerzeugung und der Anwendung der Netze untersucht. Die Lernrate ist bei jedem Modell $5 * 10^{-5}$, als Optimizer wird jeweils der AdamW-Optimizer verwendet. In dieser Arbeit steht „TP“ für „Echt-Positiv“, „FP“ für „Falsch-Positiv“, „FN“ für „Falsch-Negativ“, und „TN“ für „Echt-Negativ“.

4.3.1 Unveränderte Trainingsdaten

Zuerst wird die Tokenklassifikation an unveränderten Trainingsdaten getestet, das heißt, keine der Modell-Optionen aus Abschnitt 3.4 werden angewandt. Allerdings wird jedes nicht-relevante Token mit einer Chance von 0.25 maskiert, um zusätzlich Varianz in die Daten zu bringen. Das Training läuft pro Modell jeweils acht Stunden. Die Gruppengröße beim Training beträgt acht.

Es gibt einen Trainings-/Test-Split von 90/10. Die Tokenklassifikation wird in dieser Arbeit unterschieden in Multi-Task-Learning und Single-Task-Learning (vgl. Abschnitt 3.5.2).

Multi-Task-Learning

Beim Multi-Task-Learning werden alle Klassen mit einem einzigen Modell extrahiert. Dafür werden als Trainingsdaten alle Daten genutzt, in denen durch eine Volltextsuche oder mit regulären Ausdrücken relevante Metadaten gefunden werden. Dadurch gibt es für jede der vier gesuchten Klassen Trainingsbeispiele, aber nicht jede Klasse ist in jedem Paragraph vertreten. Als Tokenizer wird der Z-Tokenizer verwendet. Es werden verschiedene Schwellenwerte getestet, ab denen ein Token einer Klasse zugehörig ist. Als Schwellenwerte genutzt werden 0,4, 0,5, 0,6, 0,7, 0,8, 0,9, 0,95, 0,99, 0,999, 0,9999, 0,99999 und 0,999999.

Wird angenommen, dass alle Klassen den gleichen Grenzwert nutzen, ist der höchste Makro-F1-Wert 0,896, beim Schwellenwert 0,8. (vgl. Tabelle 5.1). Die Auswertung der einzelnen Klassen liefert dann ein genaueres Bild. Die nominalen Klassen erzielen jeweils auf allen Daten einen F1-Wert von über 0,96 (Nutzpflanzen: 0,966, Texturen: 0,994, Bodenarten: 0,967) (vgl. Tabellen 5.2, 5.3, und 5.4), die Koordinaten als metrische Klasse erzielen lediglich einen F1-Wert von etwa 0,656 (vgl. Tabelle 5.5). Der Makro-F1-Wert beträgt für die Testdaten 0,914 (vgl. Tabelle 5.6). Durch die guten Ergebnisse bei den Testdaten kann eine Überspezialisierung ausgeschlossen werden.

Werden verschiedene Schwellenwerte für die unterschiedlichen Klassen verwendet, erzielt die Koordinatenklassifizierung einen F1-Wert von 0,678 beim Schwellenwert 0,95. Bei diesem Schwellenwert erzielt auch die Klassifizierung der Nutzpflanzen ihren höchsten F1-Wert, mit 0,976. Das Maximum der Texturen, 0,994, wird beim Schwellenwert 0,7 erzielt, und das Maximum der Bodenarten, 0,971, wird beim Schwellenwert 0,6 erzielt. Auf diese Weise beträgt der Makro-F1-Wert beim gesamten Datensatz 0,905.

Single-Task-Learning

Beim Single-Task-Learning wird für jede zu extrahierende Klasse ein eigenes Modell erzeugt (vgl. Abschnitt 3.5.2). Um einen direkten Vergleich zum Multi-Task-Learning-Modell zu ermöglichen, werden die gleichen Verfahren angewandt und die gleichen Parameter genutzt. Auch werden die gleichen Trainings- und Testdaten verwendet, so dass im Training auch Paragraphen genutzt werden, die keine relevanten Daten für das jeweilige Modell enthalten.

Das Nutzpflanzen-Modell kann beim Schwellenwert 0,4 einen F1-Wert von 0,989 erzielen (vgl. Tabelle 5.7). Das Texturen-Modell erreicht einen F1-Wert

Tabelle 4.4: Vergleich der verschiedenen Varianten. Abgebildet sind jeweils F1-Werte. Beim Multi-Task-Learning-Modell wird sich auf die F1-Werte bei unterschiedlichen Grenzwerten bezogen.

Modell	Nutzpflanzen	Texturen	Bodenarten	Koordinaten
Multi	0,9761	0,9948	0,9713	0,6783
Single	0,9892	0,9941	0,9805	0,0

von 0,994 beim Schwellenwert 0,7 (vgl. Tabelle 5.8). Beim Bodenarten-Modell liegt der höchste F1-Wert bei 0,980, beim Schwellenwert 0,7 (vgl. Tabelle 5.9). Bei dem Koordinaten-Modell kann kein Schwellenwert positive Ergebnisse liefern, es wird nicht ein einziger Token als Koordinate klassifiziert (vgl. Tabelle 5.10).

Vergleicht man die Werte des Multi-Task-Learnings mit denen vom Single-Task-Learning, sieht man, dass das Single-Task-Learning bei Nutzpflanzen und Bodenarten um jeweils etwa 0,1 besser ist. Bei Texturen ist das Single-Task-Learning um 0,0007 schlechter, und auf Koordinaten funktioniert das Single-Task-Learning auf diese Weise nicht (vgl. Tabelle 4.4).

4.3.2 Methodische Änderungen des Trainings

Um die Extraktion von Koordinaten beim Single-Task-Learning zu verbessern, werden zwei weitere Methoden ausprobiert. Methode 1 beschränkt die Trainingsdaten auf Paragraphen mit für das Modell relevanten Daten, also nur Paragraphen, in welchen Koordinaten gefunden wurden (vgl. Abschnitt 3.4). Methode 2 führt Gewichte ein, so dass Token mit dem Label „Koordinate“ für das Training höher gewichtet werden, damit sich das Training stärker an diesen orientiert (vgl. Abschnitt 3.5.2).

Beide Methoden werden kombiniert, wodurch drei neue Modelle entstehen. Die Gewichte haben keine Auswirkungen auf die Ergebnisse, es werden weiterhin keine Token als Koordinaten gelabelt, egal bei welchem Schwellenwert. Lediglich die Beschränkung der Trainingsdaten verbessert das Modell, der höchste F1-Wert wird bei einem Schwellenwert von 0,999 erreicht, er beträgt 0,5345 (vgl. Tabelle 4.5). Dieser Wert liegt unter dem erzielten F1-Wert für Koordinaten beim Multi-Task-Learning.

4.3.3 Grundlegende Änderungen der Wortzerlegung

Ein weiterer Ansatz, um Koordinaten zu extrahieren, ist die Möglichkeit der Leerzeichentrennung, wie in Abschnitt 3.3 beschrieben. Da dadurch die Zahlen

Tabelle 4.5: Methodische Änderungen des Trainings - Vollständige Daten - Schwellenwert 0.999. Relevant - Nutzt Modell nur relevante Trainingsdaten? Gewichte - Nutzt das Modell Gewichte?

Relevant	Gewichte	TP	FP	FN	F1-Wert
Nein	Nein	0	0	7.636	0,0
Nein	Ja	0	0	7.636	0,0
Ja	Nein	4.304	4.163	3.332	0,5345
Ja	Ja	0	0	7.636	0,0

nicht mehr in Gruppen auftreten können, wird hier nur der Standard-Tokenizer verwendet. Weiterhin werden die drei grundlegenden Modifikationen, die in Abschnitt 3.4 beschrieben sind, implementiert: Kürzen, Permutieren und Löschen. Es werden exakte und genaue Klassifikationen gesucht (vgl. Abschnitt 3.5.2). Zusätzlich wird, im Vergleich zum vorigen Modell, das Label „Koordinate“ fallen gelassen, so dass jeder relevante Token in exakt einer Klasse liegt - die Klassen bezeichnen wieder die Einzelteile der Koordinate. Dadurch kann der Standardloss von BERT verwendet werden, es wird keine zusätzliche Schicht benötigt. Das ermöglicht, die Loss-Funktion zu erweitern, wie in Abschnitt 3.5.2 beschrieben. Durch diese fünf Optionen gibt es für diese Variante 32 Modelle.

Den höchsten F1-Wert erzielt das Modell 11110, er beträgt 0,213. 11110 heißt, dass alle Modifikationen bis auf die modifizierte Loss-Funktion genutzt werden. Insgesamt kann das Modell in 41 Paragraphen Koordinatenanteile korrekt klassifizieren, in zwei Paragraphen werden sogar jeweils sieben Koordinatenanteile korrekt identifiziert. Das Modell kann allerdings keine Koordinate vollständig korrekt identifizieren. Auffällig ist noch das Modell 10000, also das Modell, bei welchem nur die Modifikation genutzt wird, bei der die Paragraphen gekürzt werden. Hier werden in 295 der 331 Paragraphen Koordinatenanteile gefunden, davon 16 mal acht Koordinatenanteile, was einer vollständigen Extraktion entspricht. Der F1-Wert beträgt jedoch nur 0,090, die Precision ist mit 0,047 sehr viel niedriger als die Precision des Modells mit dem besten F1-Wert. Dort beträgt die Precision 0,271. In den sechs Modellen mit den besten F1-Werten dieser Variante ist jede Modifikation mindestens einmal aktiv und mindestens einmal nicht aktiv. Damit lassen sich also keine allgemeine Aussage über die Güte der Modell-Optionen treffen. Allerdings sind die Optionen „Kürzen“ und „Löschen“ bei fünf der sechs Modellen mit den höchsten F1-Werten aktiv, und „Permutieren“ ist nur bei einem der sechs aktiv. Das Modell ohne Modifikationen erzielt einen F1-Wert von 0,056 (vgl. Tabelle 4.6).

Tabelle 4.6: Auswertung der Methode mit grundlegenden Änderungen der Wortzerlegung, absteigend nach F1-Wert sortiert, Top 6. Modell-Optionen: 1, falls Modifizierung aktiv, 0 sonst. Prec - Precision, Rec - Recall. Paragraphen: Anzahl Paragraphen, in denen n Koordinatenanteile korrekt gefunden wurden. Gesamtzahl Paragraphen: 331. Vollständige Tabelle: Tabelle 5.11.

F1-W.	Prec	Rec	Kürzen	Permutieren	Löschen	Mod. Loss	Exakte Label	1	2	3	4	5	6	7	8
,213	,271	,176	1	1	1	1	0	12	7	9	5	6	0	2	0
,201	**,485**	,127	1	0	1	0	0	1	2	1	4	8	22	0	4
,180	,454	,112	1	0	1	1	1	9	7	1	5	6	6	4	4
,177	,340	,119	1	0	1	0	1	12	9	5	3	0	3	0	4
,090	,047	**,947**	1	0	0	0	0	50	35	20	31	41	95	7	16
,083	,302	,048	0	0	1	1	0	0	1	0	5	1	4	0	4
...															
,056	,029	,857	0	0	0	0	0	41	22	28	42	33	86	4	22
...															

4.3.4 Koordinatenersetzung

Für diese Variante werden, wie in Abschnitt 3.4 beschrieben, Koordinaten über eine Volltextsuche gesucht, und jedes gefundene Ergebnis wird extrahiert. Das Netz lernt dann mit Paragraphen, die Koordinaten enthalten, jedoch sind alle Koordinaten, die in diesem Paragraphen vorkommen, durch andere, im ersten Schritt extrahierte, Koordinaten ersetzt. Die Auswahl der Koordinaten, die eingefügt werden, ist zufällig. Bei den einzufügenden Koordinaten werden dann noch zufällig die Zahlen geändert, so dass nicht immer die gleichen Koordinaten in der gleichen Form auftauchen. Es werden keine Token maskiert. Auch hier werden die Modifikationen aus Abschnitt 3.4 umgesetzt. Weiterhin werden alle Varianten sowohl mit dem Standard- als auch mit dem Z-Tokenizer getestet. Somit gibt es fünf Optionen, die für dieses Experiment in jeder Kombination getestet werden. Es werden also 32 Modelle erzeugt, wobei jedes bei der Klassifizierung mit den Schwellenwerten 0,4, 0,5, 0,6, 0,7, 0,8, 0,9, 0,95, und 0,99 getestet wird.

Den höchsten F1-Wert erzielt das Modell 0111, bei dem also die Änderungen „Permutieren“, „Löschen“ und „Exakte Label“ aktiv sind, mit dem Standard-Tokenizer. Der höchste erreichte F1-Wert von einem Modell mit dem Z-Tokenizer ist das Modell 1011, es sind also die Optionen „Kürzen“, „Löschen“,

Tabelle 4.7: Auswertung der Methode mit Koordinatenersetzung, sortiert absteigend nach F1-Wert des Standard-Tokenizers. Top 6 und Grundmodell. Modell-Optionen: 1, falls Modifizierung aktiv, 0 sonst. Vollständige Tabellen: Tabellen 5.12 und 5.13

Modell-Optionen				Tokenizer	
Kürzen	Permutieren	Löschen	Exakte Label	Standard	Z
0	1	1	1	**0,7996**	0,7639
1	0	1	0	0,7987	0,7522
1	1	1	1	0,7792	0,7433
0	1	1	0	0,7630	0,7342
1	0	1	1	0,7500	**0,7709**
0	0	1	0	0,7450	0,6253
...					
0	0	0	0	0,5273	0,4891

und „Exakte Label“ aktiv. Der erreichte F1-Wert ist 0,770, er ist also um etwa 0,02 niedriger als mit dem Standard-Tokenizer. Das Modell ohne Optionen erzielt F1-Werte von 0,527 (Standard-Tokenizer) beziehungsweise 0,489 (Z-Tokenizer). Insgesamt erzielt der Standard-Tokenizer also leicht bessere Ergebnisse (vgl. Tabelle 4.7).

4.3.5 Randomisierte Koordinaten

Diese Variante wendet die Methode aus Abschnitt 3.4 an: Koordinaten werden komplett zufällig erzeugt, inklusive Artefakten, die durch die Konvertierung entstehen. Es wird wieder die Leerzeichentrennung aus Abschnitt 3.3 genutzt, somit also auch der Standard-Tokenizer. Es werden auch die drei Modifikationen aus Abschnitt 3.4 genutzt, sowie exakte und genaue Label aus Abschnitt 3.5.2. Die Klasse „Koordinate“ wird zusätzlich eingeführt, wobei die Einteilung in Koordinatenanteile bestehen bleibt. Da Token also zu zwei Klassen gehören können, wird hier eine Extraschicht auf das BERT-Standardmodell aufgesetzt.

Der höchste erzielte F1-Wert von allen Modellen ist 0,556 von Modell 1011, also das Modell, welches die Modifikationen „Kürzen“, „Löschen“ und „Exakte Label“ nutzt. Jedes Modell, bei dem die Modifikation „Löschen“ aktiv ist, hat einen höheren F1-Wert, als jedes Modell, das diese Modifikiation nutzt (vgl.

Tabelle 4.8: Auswertung der Methode mit randomisierten Koordinaten, sortiert absteigend nach F1-Wert. Ausgewählt ist jeweils der Schwellenwert mit dem höchsten F1-Wert pro Modell. Gesamtzahl Token: 15.239.597, davon relevante Token: 3.046. Modell-Optionen: 1, falls Modifizierung aktiv, 0 sonst. Korrekte Token gibt in Klammern an, wieviele davon relevant waren. Prec - Precision, Rec - Recall.

F1-W.	Kürzen	Permutieren	Löschen	Exakte Label	Korrekte Token	FP	Prec	Rec
,5563	1	0	1	1	15.229.877 (2.155)	2.546	**,4584**	,7074
,5364	1	0	1	0	15.226.801 (2.236)	3.054	,4226	,7340
,5341	0	0	1	1	15.232.159 (2.103)	2.725	,4355	,6904
,5310	1	1	1	0	13.349.484 (2.123)	2.826	,4289	,6969
,5287	0	1	1	1	15.232.140 (2.369)	3.546	,4005	,7777
,5016	0	0	1	0	15.225.763 (2.374)	4.044	,3698	,7793
,4954	1	1	1	1	**15.234.060** (1.777)	2.350	,4305	,5833
,4545	0	1	1	0	15.231.211 (1.812)	3.115	,3677	,5948
,3958	0	1	0	1	15.230.871 (1.252)	2.027	,3818	,4110
,3881	0	1	0	0	15.210.141 (1.123)	1.617	,4098	,3686
,3700	1	1	0	0	15.214.916 (1.044)	**1.552**	,4021	,3427
,3628	1	1	0	1	15.233.003 (0.970)	2.076	,4215	,3184
,1941	0	0	0	1	14.964.445 (2.404)	19.310	,1107	,7892
,1616	1	0	0	0	15.154.188 (2.430)	24.592	,0899	,7977
,1546	0	0	0	0	15.064.476 (**2.448**)	26.166	,0855	**,8036**
,1532	1	0	0	1	14.462.535 (2.261)	24.208	,0854	,7422

Tabelle 4.9: Auswertung der Methode mit randomisierten Koordinaten und Gewichten, sortiert absteigend nach F1-Wert. Ausgewählt ist jeweils der Schwellenwert mit dem höchsten F1-Wert pro Modell. Gesamtzahl Token: 15.239.597, davon relevante Token: 3.046. Modell-Optionen: 1, falls Modifizierung aktiv, 0 sonst. Korrekte Token gibt in Klammern an, wieviele davon relevant waren. Prec - Precision, Rec - Recall.

F1-Wert	Kürzen	Permutieren	Löschen	Exakte Label	Korrekte Token	FP	Prec	Rec
,5283	0	1	0	1	15.090.884 (2.590)	4.169	,3831	,8502
,5190	1	1	1	0	14.982.490 (2.014)	**2.700**	**,4272**	,6611
,4680	1	1	0	1	**15.199.591** (1.852)	3.015	,3805	,6080
,4582	1	1	1	1	14.827.538 (2.592)	5.674	,3135	,8509
,4394	1	1	0	0	15.162.868 (2.620)	6.258	,2951	,8601
,4313	0	1	1	1	15.165.345 (1.652)	2.962	,3580	,5423
,2865	0	1	1	0	08.810.695 (2.063)	9.289	,1817	,6772
,2330	1	0	1	0	14.629.441 (2.201)	13.641	,1389	,7225
,1948	1	0	1	1	13.602.244 (2.382)	19.026	,1112	,7820
,1747	0	1	0	0	14.949.380 (1.872)	16.511	,1018	,6145
,1115	0	0	1	0	14.474.908 (2.168)	33.642	,0605	,7117
,0845	0	0	0	0	14.948.854 (**2.842**)	61.356	,0442	**,9330**
,0826	0	0	0	1	14.939.060 (2.750)	60.727	,0433	,9028
,0721	0	0	1	1	14.789.349 (2.780)	71.190	,0375	,9126
,0469	1	0	0	0	14.662.050 (2.797)	113.283	,0240	,9182
,0306	1	0	0	1	14.594.915 (2.757)	174.164	,0155	,9051

Tabelle 4.8).

4.3.6 Randomisierte Koordinaten und Gewichte

Diese Variante funktioniert exakt so wie die im Abschnitt 4.3.5, sie wird jedoch erweitert um Gewichte, um das Ungleichgewicht zwischen den Klassen auszugleichen (vgl. Abschnitt 3.5.2).

Das Modell mit dem höchsten F1-Wert von allen Modellen ist Modell 0101, also das Modell, bei dem „Permutieren“ und „Exakte Label“ aktiv sind. Der F1-Wert beträgt 0,528. Bei dieser Variante erzielen Modelle, die „Permutieren“ nutzen, höhere F1-Werte als Modelle, die nicht „Permutieren“ nutzen, bis auf

Tabelle 4.10: Auswertung der Methode mit randomisierten Koordinaten und Z-Tokenizer, sortiert absteigend nach F1-Wert. Ausgewählt ist jeweils der Schwellenwert mit dem höchsten F1-Wert pro Modell. Gesamtzahl Token: 35.252.681, davon relevante Token: 4.342. Modell-Optionen: 1, falls Modifizierung aktiv, 0 sonst. Modell 0001 nicht aufgeführt, da das Modell nichts als Koordinate klassifizierte.

F1-Wert	Kürzen	Permutieren	Löschen	Exakte Label	Korrekt	FP	Precision	Recall
0,4753	1	0	1	0	2.453	3.525	**0,4103**	0,5649
0,4625	1	0	1	1	2.454	3.815	0,3914	0,5651
0,4611	1	1	1	1	2.692	4.642	0,3670	0,6199
0,4460	0	0	1	0	2.509	4.399	0,3632	0,5778
0,4419	0	0	1	1	2.646	4.986	0,3466	0,6093
0,4303	0	1	1	0	2.467	4.657	0,3462	0,5681
0,3535	0	1	1	1	1.657	3.374	0,3293	0,3816
0,3460	1	1	1	0	1.510	2.874	0,3444	0,3477
0,3141	0	1	0	0	1.788	5.254	0,2539	0,4117
0,3091	0	0	0	0	2.738	10.632	0,2047	0,6305
0,2941	0	1	0	1	1.138	2.257	0,3351	0,2620
0,2695	1	1	0	0	1.128	2.900	0,2800	0,2597
0,2601	1	1	0	1	970	**2.144**	0,3114	0,2233
0,1401	1	0	0	0	919	7.858	0,1047	0,2116
0,1012	1	0	0	1	**3.016**	52.213	0,0546	**0,6946**

das Modell 0100, welches ausschließlich „Permutieren“ nutzt (vgl. Tabelle 4.9).

4.3.7 Randomisierte Koordinaten und Z-Tokenizer

Diese Variante funktioniert so wie die im Abschnitt 4.3.5, allerdings wird hier der Z-Tokenizer statt des Standard-Tokenizers genutzt (vgl. Abschnitt 3.3).

Das Modell mit dem höchsten F1-Wert ist das Modell 1011, bei dem also die Änderungen „Kürzen“ und „Löschen“ aktiv sind, sowie exakte Label genutzt werden. Das Modell hat einen F1-Wert von 0,462. Bei dieser Variante hat jedes Modell, bei dem die Änderung „Löschen“ aktiv ist, einen höheren F1-Wert als jedes Modell, bei dem „Löschen“ nicht aktiv ist (vgl. Tabelle 4.10).

Tabelle 4.11: Auswertung der Methode mit randomisierten Koordinaten ohne allgemeine Klasse, Top 7. Modell: 1, falls Modifizierung aktiv, 0 sonst. Paragraphen: Anzahl Paragraphen, in denen n Koordinatenanteile korrekt gefunden wurden (331 Paragraphen).Vollständige Tabelle: Tabelle 5.14

	Modell-Optionen					Paragraphen							
F1-Wert	Kürzen	Permutieren	Löschen	Mod. Loss	Exakte Label	1	2	3	4	5	6	7	8
0,3942	1	0	1	1	1	1	6	4	5	17	47	2	12
0,1589	1	0	0	1	1	23	28	38	54	43	99	6	15
0,1123	0	0	1	1	1	9	5	7	5	12	13	0	4
0,1010	1	0	0	1	0	2	6	21	50	89	133	9	20
0,0983	1	0	1	0	0	1	0	0	0	3	10	2	6
0,0892	0	0	1	0	1	2	0	0	2	6	15	0	4
0,0871	0	0	0	0	0	0	3	10	43	72	176	7	20

4.3.8 Randomisierte Koordinaten ohne allgemeine Klasse

Die letzte durchgeführte Variante setzt wieder auf den Standard-Tokenizer. Es wird Leerzeichentrennung durchgeführt (vgl. 3.3). Die Koordinaten werden vollständig zufällig generiert (vgl. Abschnitt 3.4). Es werden die drei Modifikationen der Trainingsdaten, die in 3.4 beschrieben sind, implementiert: Kürzen, Permutieren und Löschen, sowie exakte und genaue Label aus Abschnitt 3.5.2. Auch kann die modifizierte Loss-Funktion genutzt werden, wie in 3.5.2 beschrieben. Die einzigen Label, die vergeben werden, sind genaue oder exakte Label (je nach Modell), so dass es keine allgemeine Klasse für Koordinaten gibt. Durch diese Modifikationen werden 32 Modelle erzeugt.

Nur vier Modelle können einen F1-Wert von über 0,1 erzielen, und zwar die Modelle 10111 (aktive Modifizierungen: „Kürzen“, „Löschen“, „Modifizierte Loss-Funktion“ und „Exakte Label“), 10011 (aktive Modifizierungen: „Kürzen“, „Modifizierte Loss-Funktion“ und „Exakte Label“), 00111 (aktive Modifizierungen: „Löschen“, „Modifizierte Loss-Funktion“ und „Exakte Label“), und 10100 (aktive Modifizierungen: „Kürzen“ und „Löschen“), jeweils mit den F1-Werten von 0,394, 0,158, 0,112 und 0,101. Das Modell mit dem höchsten F1-Wert kann in zwölf Paragraphen alle acht Koordinatenanteile korrekt extrahieren, in 237 Paragraphen kann kein einziger Koordinatenanteil extrahiert werden. Aus den

meisten Paragraphen kann das Modell 10010 korrekt extrahieren: In nur einem Paragraphen kann kein Koordinatenanteil korrekt gefunden werden. Das Modell 00000 ohne Änderungen am Trainingsdatensatz oder am Loss erzielt einen F1-Wert von 0,087, und ist damit besser als 25 andere Modelle. Unter den sieben Modellen mit den höchsten F1-Werten findet sich kein Modell, bei dem „Permutieren“ aktiv ist (vgl. Tabelle 4.11).

4.4 Bewertung

Die Extraktion mit Sequenzklassifikation lieferte für nominale Daten gute Ergebnisse. Das Soil-less-Modell bestätigte, dass die Modelle die Ausdrücke nicht auswendig lernen, sondern der Kontext, in welchem die Wörter auftreten, gelernt wurde. Die Extraktion mit diesem Modell funktionierte zwar etwas schlechter als die Extraktion mit dem Soils-Modell, aber ein klarer Trend ist dennoch erkennbar. Die Extraktion von Koordinaten nur durch Attention ist nicht gelungen, durch die Halbierungs-Attention-Methode konnten aber auch hier die Ergebnisse verbessert werden. Die zahlenmäßig beste Extraktionsmethode war jedoch die reine Halbierungsmethode. Die Extraktion von nominalen Daten durch die Tokenklassifikation funktionierte sowohl mit einem Multi-Task-Learning-Ansatz, als auch mit einem Single-Task-Learning-Ansatz. Da der Wert für Koordinaten beim Multi-Task-Learning-Ansatz im Vergleich zu den nominalen Daten deutlich schlechter war, wurde versucht, durch Modifikationen die Klassifikationsrate zu verbessern. Das ist auch gelungen, durch Ersetzung von Koordinaten konnte ein F1-Wert von 0,799 erreicht werden. Eine direkte Extraktion der einzelnen Koordinatenanteile mit den hier vorgestellten Methoden ist zu ungenau für den praktischen Einsatz, das reine Labeln als Koordinate ermöglichte aber die automatische Extraktion von Koordinatentoken. Der Standard-Tokenizer lieferte im Allgemeinen etwas bessere Ergebnisse als der Z-Tokenizer aus Abschnitt 3.3.

Die einzelnen Data Augmentation-Modifikationen sind unterschiedlich effektiv, in den verschiedenen getesteten Varianten gab es keine Modifikation, die den F1-Wert ausschließlich verschlechtert hat. Manche Kombinationen waren allerdings in einzelnen Varianten schlechter als keine Modifikationen. So war zum Beispiel bei den Modellen aus Abschnitt 4.3.8 keine einzelne Modifikation besser als die Variante ohne Modifikationen, aber eine Kombination von fast allen Modifikationen (bis auf „Permutieren“) ist mehr als vier mal so gut wie die Variante ohne Modifikation. Jede der grundlegenden Data Augmentation-Methoden war mindestens einmal besser als die Variante ohne Änderungen. Es lässt sich aber kein klares Muster erkennen, wann welche Modifikation besser oder schlechter ist. Die Modifikationen „Modifizierte Loss-Funktion“ und

Tabelle 4.12: Vergleich der verschiedenen Data Augmentation-Varianten. WZ - Wortzerlegung, KE - Koordinatenersetzung, K. a. Klasse - Keine allgemeine Klasse. In Klammern jeweils das Modell mit dem höchsten F1-Wert.

			Randomisierte Koordinaten			
Augmentation	WZ	KE	Standard	Gewichte	Z-Tok.	K. a. Klasse
Keine	0,056	0,527	0,154	0,084	0,309	**0,087**
Kürzen	**0,090**	0,564	0,161	0,046	0,140	0,0
Permutieren	0,000	0,687	0,388	**0,174**	0,314	0,0
Löschen	0,039	**0,718**	**0,501**	0,111	**0,446**	0,049
Mod. Loss	0,056	-	-	-	-	0,074
Ex. Label	0,050	0,593	0,194	0,082	0,0	0,047
Beste	0,213	0,799	0,556	0,528	0,475	0,394
	(11110)	(0111)	(1011)	(0101)	(1010)	(10111)

„Exakte Label“ waren in keinem untersuchten Fall die beste einzelne Variante, sie werden aber in den Modellen mit den höchsten F1-Werten pro Variante oft genutzt (vgl. Tabelle 4.12).

Vollständige Randomisierung der zu extrahierenden Daten im Training liefert schlechte Ergebnisse, vermutlich, weil die Daten zu weit von realen Daten entfernt sind. Rein auf realen Daten zu trainieren liefert allerdings bei zu kleinen Trainingsmengen auch keine guten Ergebnisse. Die Variante mit dem Ersetzen von Koordinaten in einem Paragraphen durch Koordinaten aus einem anderen Paragraphen, inklusive zufälliger Zahlen in diesen Koordinaten, scheint hier ein guter Kompromiss zu sein, da die F1-Werte hier verbessert werden konnten.

Betrachtet man die falsch-positiven Label des Multi-Task-Learning-Ansatzes, stellt man fest, dass etwa 75% der Token zu Koordinaten gehören, sie wurden nur von der Volltextsuche nicht erkannt. Auch bei Bodenarten (42%), Texturen (58%), und Nutzpflanzen (36%) fanden sich noch viele jeweils relevante Token unter den falsch-positiven Labeln. Für diese Auswertung wurden jeweils 50 Paragraphen angeschaut, in welchen falsch-positive Label der jeweiligen Kategorie aufgetaucht sind. Damit erweitert der Multi-Task-Learning-Ansatz die Ergebnisse der Volltextsuche und kann verallgemeinern.

Kapitel 5

Abschluss

Alles in allem kann aus den Experimenten geschlossen werden, dass BERT für die Extraktion von nominalen Daten und Koordinaten aus bodenwissenschaftlichen Artikeln geeignet ist. Die hohen Werte der falsch-positiven Ergebnisse lassen sich durch Fehler beim originalen Labeling durch die Volltextsuche erklären, die Netze arbeiten in dieser Hinsicht also besser. Allerdings ist eine hundertprozentige Genauigkeit nicht gegeben. Falls eine sehr hohe Exaktheit benötigt wird, kann ein BERT-Netz genutzt werden, um die Textmengen, die manuell verarbeitet werden müssten, zu verringern, indem potentiell relevante Paragraphen bereitgestellt werden. Die Extraktion einzelner Koordinatenanteile hat nur mit sehr niedriger Genauigkeit funktioniert. In der Praxis ist diese Variante so nicht verwendbar, da sowohl die tatsächliche Falsch-Positiv-, als auch Falsch-Negativ-Rate sehr hoch ist. Die Ergebnisse wurden auch durch verschiedene Modifikationen nicht deutlich verbessert. Die verschiedenen Modifikationen, die durchgeführt wurden, um das Trainingsdatenset diverser zu gestalten, haben gemischte Ergebnisse geliefert. Manche Modifikationen waren in einigen Konfigurationen vorteilhaft, in anderen wiederrum nicht. Es lässt sich also keine allgemeine Aussage über die Effektivtät dieser Modifikationen treffen. Es gab allerdings bei jeder Variante Möglichkeiten, den F1-Wert gegenüber der Variante ohne Modifikationen zu verbessern. Den größten Effekt hatte das Verändern der relevanten Daten. Eine vollständig zufällige Generation der Koordinaten hat zu schlechteren Ergebnissen geführt, ein Vertauschen von Koordinaten zwischen Paragraphen, und zufälliges Ersetzen der einzelnen Koordinatenanteile hat die Ergebnisse deutlich verbessert. Offene Fragen dieser Arbeit sind, in welchem Kontext welche Modifikationen von Vorteil sind, und ob sich das verallgemeinern lässt. Weiterhin unklar ist, wie andere Data Augmentation-Methoden Einfluss auf das Training haben könnten, und welche Modifikationen überhaupt noch möglich sind. Es besteht ebenfalls Unklarheit, ob sich die hier verwendeten Verfahren für andere wissenschaftliche Domä-

nen und andere Arten von Metadaten übertragen lassen. Die Frage, ob sich Koordinatenanteile effektiv direkt mit BERT extrahieren lassen, bleibt unbeantwortet. Auch die Extraktion mithilfe der Sequenzklassifikation wurde hier eher oberflächlich behandelt.

Literaturverzeichnis

Grobid. https://github.com/kermitt2/grobid, 2008–2021.

Zeyd Boukhers, Shriharsh Ambhore, and Steffen Staab. An end-to-end approach for extracting and segmenting high-variance references from PDF documents. jun 2019. doi: https://doi.org/10.1109/JCDL.2019.00035.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. 2019.

Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. A survey of data augmentation approaches for nlp, 2021.

Maarten Grootendorst and Nils Reimers. Maartengr/bertopic: v0.9.4. 2021. doi: https://doi.org/10.5281/zenodo.4381785.

Sarang Gupta and Kumari Nishu. Mapping local news coverage: Precise location extraction in textual news content using fine-tuned BERT based language model. 2020. doi: 10.18653/v1/2020.nlpcss-1.17.

Eun-Jung Holden, Wei Liu, Tom Horrocks, Rui Wang, Daniel Wedge, Paul Duuring, and Trevor Beardsmore. GeoDocA – fast analysis of geological content in mineral exploration reports: A text mining approach. *Ore Geology Reviews*, 111:102919, aug 2019. doi: 10.1016/j.oregeorev.2019.05.005.

Qinjin Jia, Jialin Cui, Yunkai Xiao, Chengyuan Liu, Parvez Rashid, and Edward F. Gehringer. ALL-IN-ONE: multi-task learning BERT models for evaluating peer assessments. *CoRR*, abs/2110.03895, 2021. URL https://arxiv.org/abs/2110.03895.

Priyanshu Kumar, Aadarsh Singh, Pramod Kumar, and Chiranjeev Kumar. An explainable machine learning approach for definition extraction. pages 145–155, 2020. doi: doi:10.1007/978-981-15-6318-8_13.

Xiong Luo, Wenwen Zhou, Weiping Wang, Yueqin Zhu, and Jing Deng. Attention-based relation extraction with bidirectional gated recurrent unit and highway network in the analysis of geological data. *IEEE Access*, 6: 5705–5715, 2018. doi: 10.1109/ACCESS.2017.2785229.

Kai Ma, Miao Tian, Yongjian Tan, Xuejing Xie, and Qinjun Qiu. What is this article about? generative summarization with the BERT model in the geosciences domain. *Earth Science Informatics*, sep 2021. doi: 10.1007/s12145-021-00695-2.

Andriy Mulyar and Bridget T. McInnes. Mt-clinical BERT: scaling clinical information extraction with multitask learning. *CoRR*, abs/2004.10220, 2020. URL `https://arxiv.org/abs/2004.10220`.

Yifan Peng, Qingyu Chen, and Zhiyong Lu. An empirical study of multi-task learning on BERT for biomedical text mining. 2020. doi: 10.18653/v1/2020.bionlp-1.22.

Qinjun Qiu, Zhong Xie, Liang Wu, and Wenjia Li. Geoscience keyphrase extraction algorithm using enhanced word embedding. *Expert Systems with Applications*, 125:157–169, jul 2019. doi: https://doi.org/10.1016/j.eswa.2019.02.001.

Qinjun Qiu, Zhong Xie, Liang Wu, and Liufeng Tao. Dictionary-based automated information extraction from geological documents using a deep learning algorithm. *Earth and Space Science*, 7(3), mar 2020. doi: 10.1029/2019EA000993.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019. URL `http://arxiv.org/abs/1910.01108`.

Claudia Schulz, Steffen Eger, Johannes Daxenberger, Tobias Kahse, and Iryna Gurevych. Multi-task learning for argumentation mining in low-resource settings. 2018. doi: 10.18653/v1/N18-2006.

Prafull Sharma and Yingbo Li. Self-supervised contextual keyword and keyphrase retrieval with self-labelling. aug 2019. doi: 10.20944/preprints201908.0073.v1.

De sheng WANG, Jun zhi LIU, A xing ZHU, Shu WANG, Can ying ZENG, and Tian wu MA. Automatic extraction and structuration of soil–environment relationship information from soil survey reports. *Journal of Integrative Agriculture*, 18(2):328–339, feb 2019. doi: https://doi.org/10.1016/S2095-3119(18)62071-4.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL `http://arxiv.org/abs/1706.03762`.

Bin Wang, Liang Wu, Wenjia Li, Qinjun Qiu, Zhong Xie, Hao Liu, and Yuan Zhou. A semi-automatic approach for generating geological profiles by integrating multi-source data. *Ore Geology Reviews*, 134:104190, jul 2021. doi: 10.1016/j.oregeorev.2021.104190.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771, 2019. URL `http://arxiv.org/abs/1910.03771`.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Googles neural machine translation system: Bridging the gap between human and machine translation, 2016.

Tomer Wullach, Amir Adler, and Einat Minkov. Fight fire with fire: Fine-tuning hate detectors using large samples of generated hate speech, 2021.

Ruixue Zhang, Wei Yang, Luyun Lin, Zhengkai Tu, Yuqing Xie, Zihang Fu, Yuhao Xie, Luchen Tan, Kun Xiong, and Jimmy Lin. Rapid adaptation of bert for information extraction on domain-specific business documents, 2020.

Anhang

Tabelle 5.1: Auswertung des Multi-Task-Learning, Übersicht. Gesamtzahl Token: 34.835.392, davon relevante Token: 325.354. In Klammern jeweils die Anzahl der jeweiligen Label bei relevanten Daten.

Schwellenwert	Korrekte Label	Falsche Label	Makro-F1-Wert
0,4	34.778.530 (322.825)	56.862 (002.529)	0,8715
0,5	34.789.255 (322.281)	46.137 (003.073)	0,8793
0,6	34.797.772 (321.566)	37.620 (003.788)	0,8859
0,7	34.804.539 (320.553)	30.853 (004.801)	0,8918
0,8	34.809.488 (318.609)	25.904 (006.745)	**0,8962**
0,9	34.810.859 (312.954)	245.33 (012.400)	0,8960
0,95	34.804.650 (303.057)	30.742 (022.297)	0,8838
0,99	34.766.643 (260.359)	68.749 (064.995)	0,7965
0,999	34.581.116 (71.109)	254.276 (254.245)	0,1355
0,9999	34.510.038 (0)	325.354 (325.354)	0,0
0,99999	34.510.038 (0)	325.354 (325.354)	0,0
0,999999	34.510.038 (0)	325.354 (325.354)	0,0

Tabelle 5.2: Auswertung des Multi-Task-Learning, Nutzpflanzen. Schwellenwert 0,8. Token: Anzahl relevanter Token dieser Art.

Daten	Token	TP	FP	FN	Precision	Recall	F1-Wert
Training	170.980	168.679	748	2.301	0,995	0,986	0,991
Test	19.460	18.868	100	592	0,994	0,969	0,981
Vollständig	191.101	188.206	10.212	2.895	0,948	0.984	0,966

Tabelle 5.3: Auswertung des Multi-Task-Learning, Texturen. Schwellenwert 0,8. Token: Anzahl relevanter Token dieser Art.

Daten	Token	TP	FP	FN	Precision	Recall	F1-Wert
Training	48.538	48.190	159	348	0,996	0,992	0,994
Test	5.144	5.103	16	41	0,996	0,992	0,994
Vollständig	53.705	53.315	239	390	0,995	0,992	0,994

Tabelle 5.4: Auswertung des Multi-Task-Learning, Bodenarten. Schwellenwert 0,8. Token: Anzahl relevanter Token dieser Art.

Daten	Token	TP	FP	FN	Precision	Recall	F1-Wert
Training	65.769	63.161	411	2.608	0,993	0,960	0,976
Test	6.574	6.574	90	481	0,986	0,931	0,958
Vollständig	69.822	69.822	1.574	3.090	0,977	0,957	0,967

Tabelle 5.5: Auswertung des Multi-Task-Learning, Koordinaten. Schwellenwert 0,8. Token: Anzahl relevanter Token dieser Art.

Daten	Token	TP	FP	FN	Precision	Recall	F1-Wert
Training	6.940	6.623	2.723	317	0,708	0,954	0,813
Test	670	617	417	53	0,596	0,920	0,724
Vollständig	7.636	7.266	7.229	370	0,501	0,951	0,656

Tabelle 5.6: Auswertung des Multi-Task-Learning, durchschnittliche F1-Werte. Schwellenwert 0,8.

Daten	Durchschnittlicher F1-Wert
Training	0,943951
Test	0,914749
Vollständig	0,896201

Tabelle 5.7: Single-Task-Learning, Nutzpflanzen. Vollständige Daten, sortiert absteigend nach F1-Wert. 191.101 relevante Token.

F1-Wert	TP	FP	FN	Precision	Recall	Schwellenwert
0,9892	189.056	2.066	2.045	0,989	0,989	0,4
0,9891	188.769	1.818	2.332	0,990	0,987	0,5
0,9887	188.400	1.589	2.701	0,991	0,985	0,6
0,9881	187.998	1.399	3.103	0,992	0,983	0,7
0,9871	187.411	1.186	3.690	0,993	0,980	0,8
0,9845	186.142	868	4.959	0,995	0,974	0,9
0,9786	183.578	580	7.523	0,996	0,960	0,95
0,9362	168.364	201	22.737	0,998	0,881	0,99
0,4019	48.071	7	143.030	0,999	0,251	0,999
0,0	0	0	191.101	0,0	0,0	0,9999
0,0	0	0	191.101	0,0	0,0	0,99999

Tabelle 5.8: Single-Task-Learning, Texturen. Vollständige Daten, sortiert absteigend nach F1-Wert. 53.705 relevante Token.

F1-Wert	TP	FP	FN	Precision	Recall	Schwellenwert
0,99417	53.492	414	213	0,992	0,996	0,7
0,99408	53.439	370	266	0,993	0,995	0,8
0,99405	53.521	456	184	0,991	0,996	0,6
0,99386	53.547	503	158	0,990	0,997	0,5
0,99364	53.578	558	127	0,989	0,997	0,4
0,99349	53.309	302	396	0,994	0,992	0,9
0,99176	53.070	246	635	0,995	0,988	0,95
0,97398	51.081	105	2.624	0,997	0,951	0,99
0,00282	76	7	53.629	1,0	0,001	0,999
0,0	0	0	53.705	0,0	0,0	0,9999
0,0	0	0	53.705	0,0	0,0	0,99999

Tabelle 5.9: Single-Task-Learning, Bodenarten. Vollständige Daten, sortiert absteigend nach F1-Wert. 72.912 relevante Token.

F1-Wert	TP	FP	FN	Precision	Recall	Schwellenwert
0,9805	71.686	1.618	1.226	0,977	0,983	0,7
0,9804	71.935	1.891	977	0,974	0,986	0,6
0,9797	71.290	1.318	1.622	0,981	0,977	0,8
0,9795	72.086	2.179	826	0,970	0,988	0,5
0,9783	72.193	2.483	719	0,966	0,990	0,4
0,9737	70.052	918	2.860	0,987	0,960	0,9
0,9583	67.669	632	5.243	0,990	0,928	0,95
0,8627	55.490	232	17.422	0,995	0,761	0,99
0,0011	43	0	72.869	1,0	0,001	0,999
0,0	0	0	72.912	0,0	0,0	0,9999
0,0	0	0	72.912	0,0	0,0	0,99999

Tabelle 5.10: Single-Task-Learning, Koordinaten. Vollständige Daten, sortiert absteigend nach F1-Wert. 7.636 relevante Token.

Schwellenwert	TP	TN	FP	FN	Precision	Recall	F1-Wert
0,4	0	34.827.756	0	7.636	0,0	0,0	0,0
0,5	0	34.827.756	0	7.636	0,0	0,0	0,0
0,6	0	34.827.756	0	7.636	0,0	0,0	0,0
0,7	0	34.827.756	0	7.636	0,0	0,0	0,0
0,8	0	34.827.756	0	7.636	0,0	0,0	0,0
0,9	0	34.827.756	0	7.636	0,0	0,0	0,0
0,95	0	34.827.756	0	7.636	0,0	0,0	0,0
0,99	0	34.827.756	0	7.636	0,0	0,0	0,0
0,999	0	348.27.756	0	7.636	0,0	0,0	0,0
0,9999	0	34.827.756	0	7.636	0,0	0,0	0,0
0,99999	0	34.827.756	0	7.636	0,0	0,0	0,0

Tabelle 5.11: Auswertung der Methode mit grundlegenden Änderungen der Wortzerlegung, absteigend nach F1-Wert sortiert. Paragraphen: Anzahl Paragraphen, in denen n Koordinatenanteile korrekt gefunden wurden (331 Paragraphen maximal). Modelle 11000, 11001, 01010, 11011, 11100, 11101, und 11111 labelten jeweils nichts als Koordinate und sind daher nicht mit aufgeführt.

	Model-Optionen:					Paragraphen:								
F1-Wert	Kürzen	Permutieren	Löschen	Exakte Label	Mod. Loss	0	1	2	3	4	5	6	7	8
0,2136	1	1	1	1	0	290	12	7	9	5	6	0	2	0
0,2017	1	0	1	0	0	289	1	2	1	4	8	22	0	4
0,1801	1	0	1	1	1	289	9	7	1	5	6	6	4	4
0,1772	1	0	1	0	1	295	12	9	5	3	0	3	0	4
0,0902	1	0	0	0	0	36	50	35	20	31	41	95	7	16
0,0838	1	0	1	1	0	316	0	1	0	5	1	4	0	4
0,0739	1	0	0	1	0	139	9	18	27	21	29	70	2	16
0,0629	1	0	0	0	1	79	86	37	58	43	15	6	1	6
0,0620	1	0	1	1	0	317	0	1	0	1	0	8	0	4
0,0564	0	0	0	1	0	17	5	29	42	49	85	76	14	14
0,0562	0	0	0	0	0	53	41	22	28	42	33	86	4	22
0,0529	0	0	0	1	1	15	42	80	82	79	12	13	2	6
0,0509	1	0	0	1	1	63	30	25	47	79	40	29	10	8
0,0505	0	0	0	0	1	2	14	22	81	130	45	27	2	8
0,0390	0	0	1	0	0	321	0	0	0	0	1	5	0	4
0,0335	0	0	1	1	1	321	2	2	0	0	0	2	0	4
0,0288	0	1	1	1	1	325	0	0	1	0	0	1	0	4
0,0256	0	0	1	0	1	323	0	0	0	2	0	2	0	4
0,0217	0	1	1	1	0	327	0	0	0	0	0	2	0	2
0,0171	0	1	1	0	1	325	0	0	0	0	0	2	0	4
0,0148	0	1	1	0	0	328	0	0	0	0	0	1	0	2
0,0029	0	1	0	1	1	328	0	3	0	0	0	0	0	0
0,0015	0	1	0	0	1	328	0	2	0	0	0	1	0	0
0,0011	1	1	0	1	0	331	0	0	0	0	0	0	0	0
0,0008	0	1	0	0	0	329	0	0	0	2	0	0	0	0

Tabelle 5.12: Auswertung der Methode mit Koordinatenersetzung mit dem Standard-Tokenizer, sortiert nach F1-Wert. Koordinatentoken gesamt: 4.255. Modell-Optionen: 1, falls Modifizierung aktiv, 0 sonst.

F1-Wert	Kürzen	Permutieren	Löschen	Exakte Label	Korrekt	FP	KKR	Precision	Recall
0,7996	0	1	1	1	2.353	**904**	0,9996	**0,7385**	0,8719
0,7987	1	0	1	0	2.490	1.046	0,9995	0,7089	**0,9145**
0,7792	1	1	1	1	2.289	932	0,9997	0,7219	0,8464
0,7630	0	1	1	0	2.372	1.161	**0,9998**	0,6778	0,8728
0,7500	1	0	1	1	2.427	1.415	0,9996	0,6453	0,8953
0,7450	0	0	1	0	**2.558**	1.650	0,9989	0,6170	0,9399
0,7364	0	1	0	1	2.399	1.523	0,9995	0,6294	0,8871
0,7218	1	1	1	0	2.470	1.691	0,9995	0,5992	0,9074
0,7121	0	0	1	1	2.404	1.737	0,9994	0,5947	0,8872
0,6876	0	1	0	0	2.398	1.947	0,9996	0,5627	0,8836
0,6799	1	1	0	0	2.282	1.864	0,9993	0,5683	0,8462
0,6102	1	1	0	1	2.257	2.832	0,9994	0,4772	0,8459
0,5973	1	0	0	1	2.364	3.157	0,9444	0,4527	0,8776
0,5938	0	0	0	1	2.417	3.233	0,9964	0,4448	0,8928
0,5640	1	0	0	0	1.176	2.124	0,9971	0,5282	0,6050
0,5273	0	0	0	0	1.496	3.400	0,9735	0,4318	0,6771

Tabelle 5.13: Auswertung der Methode mit Koordinatenersetzung mit dem Z-Tokenizer, sortiert nach F1-Wert. Koordinatentoken gesamt: 2.728. Modell-Optionen: 1, falls Modifizierung aktiv, 0 sonst.

F1-Wert	Kürzen	Permutieren	Löschen	Exakte Label	Korrekt	FP	KKR	Precision	Recall
0,7709	1	0	1	1	3.756	1.958	**0,9998**	0,6786	0,8923
0,7639	0	1	1	1	3.631	**1.909**	0,9997	**0,6823**	0,8679
0,7522	1	0	1	0	3.741	2.182	**0,9998**	0,6522	0,8884
0,7433	1	1	1	1	3.688	2.225	0,9997	0,6450	0,8770
0,7342	0	1	1	0	3.836	2.540	0,9974	0,6167	0,9070
0,7170	0	0	1	1	**3.852**	2.913	0,9997	0,5905	**0,9124**
0,7133	1	1	1	0	3.463	2.182	0,9995	0,6291	0,8237
0,6762	1	1	0	0	3.741	3.467	0,9984	0,5453	0,8899
0,6440	1	1	0	1	3.683	4.047	0,9995	0,5079	0,8795
0,6253	0	0	1	0	3.617	4.036	0,9990	0,4914	0,8594
0,6252	0	1	0	0	3.724	4.415	0,9970	0,4830	0,8859
0,6126	1	0	0	1	3.051	3.939	0,9956	0,5079	0,7715
0,5923	0	1	0	1	3.730	5.242	0,9994	0,4442	0,8886
0,5737	0	0	0	1	1.551	2.692	0,9531	0,5742	0,5731
0,4891	0	0	0	0	1.694	5.286	0,9940	0,4154	0,5946
0,4356	1	0	0	0	1.761	7.490	0,9786	0,3397	0,6071

Tabelle 5.14: Auswertung der Methode mit randomisierten Koordinaten ohne allgemeine Klasse, absteigend nach F1-Wert sortiert. Paragraphen: Anzahl Paragraphen, in denen n Koordinatenanteile korrekt gefunden wurden (331 Paragraphen maximal). Modell-Optionen: 1, falls Modifizierung i aktiv, 0 sonst. Prec - Precision, Rec - Recall. Modelle 11000, 01000, 10000, und 11001 labelten jeweils nichts als Koordinate und sind daher nicht mit aufgeführt.

	Modell-Optionen					Paragraphen:									
F1-W.	Kürzen	Permutieren	Löschen	Exakte Label	Mod. Loss	1	2	3	4	5	6	7	8	Prec	Rec
0,39	1	0	1	1	1	1	6	4	5	17	47	2	12	0,55	0,30
0,15	1	0	0	1	1	23	28	38	54	43	99	6	15	0,08	0,96
0,11	0	0	1	1	1	9	5	7	5	12	13	0	4	0,08	0,18
0,10	1	0	0	1	0	2	6	21	50	89	133	9	20	0,05	0,99
0,09	1	0	1	0	0	1	0	0	0	3	10	2	6	0,39	0,05
0,08	0	0	1	0	1	2	0	0	2	6	15	0	4	0,09	0,08
0,08	0	0	0	0	0	0	3	10	43	72	176	7	20	0,04	**1,0**
0,08	0	1	1	1	0	2	0	1	3	1	5	0	8	0,17	0,05
0,07	0	0	0	1	1	1	9	40	103	75	85	3	15	0,03	**1,0**
0,07	0	0	0	1	0	0	2	2	17	94	174	16	26	0,03	**1,0**
0,05	1	0	1	1	0	0	1	1	1	2	4	0	4	0,53	0,02
0,04	0	0	1	0	0	0	1	0	0	0	6	0	4	0,14	0,02
0,04	1	0	1	0	1	1	0	0	1	0	4	0	4	0,11	0,02
0,04	0	0	0	0	1	0	4	66	100	77	60	8	16	0,02	**1,0**
0,04	1	1	0	1	1	5	5	1	0	3	0	0	5	0,04	0,05
0,04	0	1	1	0	1	0	0	0	0	2	2	0	4	0,75	0,02
0,04	0	1	1	1	1	3	0	0	0	0	2	0	4	0,33	0,02
0,04	0	1	1	0	0	0	0	0	0	1	2	0	6	0,18	0,02
0,03	1	1	1	0	1	0	0	0	0	1	2	0	4	**0,71**	0,01
0,03	1	1	1	0	0	0	0	1	0	0	2	0	4	0,18	0,01
0,02	1	1	1	1	0	0	0	0	0	0	2	2	2	0,66	0,01
0,02	0	0	1	1	0	0	0	0	0	0	2	0	4	0,30	0,01
0,02	1	1	1	1	1	0	0	0	0	0	2	0	4	0,25	0,01
0,01	0	1	0	1	1	3	1	0	0	1	1	0	4	0,01	0,02
0,01	1	0	0	0	1	0	1	57	127	82	45	12	7	0,00	**1,0**
0,01	1	1	0	1	0	6	0	1	1	0	0	0	2	0,01	0,01
0,01	0	1	0	0	1	4	0	0	0	0	1	0	0	0,01	0,01
0,00	0	1	0	1	0	0	1	0	1	0	1	0	0	0,00	0,00