Analyse der Möglichkeit einer automatischen Anreicherung von sprachwissenschaftlichen Texten durch Named Entity Recognition unter Verwendung von BERT mit Semantik aus der BLL Ontologie

BACHELORARBEIT

für die Prüfung zum

Bachelor of Science

des Studienganges Informatik

an der Dualen Hochschule Baden-Württemberg Mannheim

von

Jens Heinrich

Bearbeitungszeitraum	07.06.2022-19.09.2022
Matrikelnummer, Kurs	9016280, TINF19AI2
Ausbildungsfirma	Universitätsbibliothek J. C. Senckenberg
Betreuer der Ausbildungsfirma	Dr. Thomas Risse
Gutachter der Dualen Hochschule	Patrick Arras

Analyse der Möglichkeit einer automatischen Anreicherung von sprachwissenschaftlichen Texten durch Named Entity Recognition unter Verwendung von BERT mit Semantik aus der BLL Ontologie

Jens Heinrich

9016280,Universitätsbibliothek J. C. Senckenberg

J.Heinrich@ub.uni-frankfurt.de

Erstgutachter: **Thomas Risse**

Zweitgutachter: **Patrick Arras**

Ehrenwörtliche Erklärung

Ich versichere hiermit, dass ich diese Arbeit mit dem Thema

„Analyse der Möglichkeit einer automatischen Anreicherung von sprachwissenschaftlichen Texten durch Named Entity Recognition unter Verwendung von BERT mit Semantik aus der BLL Ontologie“

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich versichere zudem, dass die eingereichte elektronische Version mit der gedruckten Fassung übereinstimmt.

Ort	Datum	Unterschrift	*Jens Heinrich*

Danksagungen

Ich möchte an dieser Stelle allen meinen Korrekturlesern für ihre Verbesserungsvorschläge danken.

Meinen Kollegen Petra, Stefan und Thorsten danke ich für ihre Kommentare und Verbesserungsvorschläge. An Petra geht zusätzlich Dank für das Wissen über die Historie der Universitätsbibliothek Johann Christian Senckenberg und das Bild der themenspezifischen Zettelkästen. Ohne Thorstens technische Unterstützung wäre diese Arbeit auch nicht möglich gewesen.

Dank geht auch an meine Freunde und Partner Franzi, Hannah, Jan, Jessi, Katrin, Oli, Paul-Lilith und Romina, die mich teilweise sogar schon zum zweiten Mal bei einer Bachelorarbeit untersützen. Eure Korrekturen waren wertvoll und das Lob der Form hat gut getan.

An Jessi geht zusätzlicher Dank für die Hinweise auf geschlechtergerechte Sprache und an Jan für gute Argumente.

Romina und Mi möchte ich besonders für die Unterstützung in Form von Motivation danken. Ohne euch wäre mir das erheblich schwerer gefallen es durchzuziehen.

Des Weiteren danke ich auch meinen Betreuern Thomas Risse und Patrick Arras für die Unterstützung bei dieser Arbeit.

Abstract

This thesis aims to evaluate the feasibility of using BERT (Bidirectional Encoder Representations from Transformers) to enrich linguistic fulltexts with annotations based on the BLL-Ontologie

The primary archievement of this thesis is the implementation of the `OAIPMH` class.

Ideas for enhancements are documented at the end of this thesis and the code can be found at `github.com/JensHeinrich/Bachelor_INF`.

Zusammenfassung

Diese Arbeit versucht die Umsetzbarkeit der Verwendung von BERT zur Anreicherung linguistischer Volltexte mit Annotationen auf Basis der BLL-Ontologie zu evaluieren.

Die primäre Leistung dieser Arbeit besteht in der Implementierung der `OAIPMH` Klasse.

Ansätze für Verbesserungen sind am Ende dieser Arbeit dokumentiert und der Code ist auf `github.com/JensHeinrich/Bachelor_INF` verfügbar.

Inhaltsverzeichnis

1 Einleitung

Die Hauptaufgabe einer modernen Universitätsbibliothek ist es, ihre Nutzer*innen zu unterstützen. Dies geschieht vor allem dadurch, dass Informationen zugänglich gemacht werden. Früher wurde dies durch Zettelkataloge, in denen die Bücher verzeichnet waren, erreicht. Im Rahmen der Formalerschließung wurden hier Informationen, wie zum Beispiel der Titel, der Autor und die ISBN (Internationale Standardbuchnummer) dokumentiert. Für die Sacherschließung wurden diese formalen Informationen um inhaltliche Informationen ergänzt. Hierfür gab es dann nach Themen sortierte Zettelkästen, wie in Abbildung 1 zu sehen ist. Die Zettelkästen wurden an der Universitätsbibliothek Johann Christian Senckenberg, wie auch an vielen anderen Bibliotheken, durch einen OPAC (Online Public Access Catalogue) ersetzt, der den Zugriff auf die Übersicht von überall bietet. Als zusätzlicher Vorteil kommt bei digitalen Lösungen die einfachere Suche über verschiedene Felder eines Eintrags hinzu.

Eine manuelle Sacherschließung hängt von den Fähigkeiten der erfassenden Person ab. Somit ist, trotz bibliothekarischer Regelwerke, nicht sichergestellt, dass alle relevanten Schlagworte und Themenkomplexe verzeichnet werden. Die vollständige Verschlagwortung und Verknüpfung mit zusätzlichem Wissen ist der Schritt Richtung „Semantic Web“ und Zukunft der Bibliotheken. Durch solche Verknüpfungen wird die Information für die Nutzer noch besser zugänglich, denn vom Eintrag sind die Themenbereiche zu erreichen, welche auf alle Einträge verweisen. Dies vereinfacht die Suche.

Im Rahmen dieser Aufgabe betreut die Universitätsbibliothek Johann Christian Senckenberg mehrere FIDs (Fachinformationsdienste) mit finanzieller Förderung der DFG (Deutsche Forschungsgemeinschaft) über deren Förderprogramm „Fachinformationsdienste für die Wissenschaft“ [1]. Diese sind in Tabelle 1 dargestellt.

Der FID Linguistik ist „ein Kooperationsprojekt zwischen der Universitätsbibliothek Johann Christian Senckenberg und Prof. Christian Chiarcos vom Forschungsgebiet Angewandte Compu-

[1] https://www.dfg.de/foerderung/programme/infrastruktur/lis/lis_foerderangebote/fachinfodienste_wissenschaft/index.html

Abbildung 1: Zettelkasten aus der Universitätsbibliothek Johann Christian Senckenberg. Quelle: Petra Schneider

Afrikastudien `http://www.ilissafrica.de/`	ilissafrica
Allgemeine und Vergleichende Literaturwissenschaft `http://avldigital.de/`	avldigital.de
Biodiversitätsforschung `https://www.biofid.de/de/`	BIOfid.de
Darstellende Kunst `http://www.performing-arts.eu/`	www.performing-arts.eu
Germanistik `https://www.germanistik-im-netz.de/`	
Jüdische Studien `https://www.jewishstudies.de/`	Fachinformationsdienst Jüdische Studien
Linguistik `http://www.linguistik.de`	Lin|gu|is|tik

Tabelle 1: Übersicht der FIDs der Universitätsbibliothek Johann Christian Senckenberg [27]

terlinguistik (AcoLi) am Institut für Informatik der Goethe Universität“ ([26]) und betreibt das Lin|gu|is|tik-Portal, welches im folgenden Abschnitt beschrieben wird.

1.1 Das Portal

„Das Lin|gu|is|tik-Portal ist ein Fachportal für die allgemeine und vergleichende Sprachwissenschaft sowie die Linguistiken der europäischen und außereuropäischen Einzelphilologien.“ ([26]) Es weist die Titelmetadaten, wie z.B. den Autor, den Titel oder das Thema, verschiedener Publikationen nach. Um den Nutzen für die Forschung zu steigern, soll in Zukunft eine Volltextsuche über die Open Access-Texte hinzugefügt werden. Diese soll durch einen Index auf Basis von KBs (Knowledge Bases), wie z.B. der BLL-Ontologie, WikiData oder GND-Sachbegriffen unterstützt werden, indem die Volltexte mit semantischen Deskriptoren versehen werden.

Fachschule für Foto- und Medientechnik	Fixed Frequency Modem
Fast Food Musical	Frankfurt am Main
Fédération française de motocyclisme	Frankfurter Feldbahnmuseum
Fédération Française Motonautique	Fudbalska Federacija na Makedonija
Female Female Male	Fünf-Faktoren-Modell
Festival des Films du Monde	Full Face Mask
Fettfreie Masse	Fused Filament Fabrication
Firefly	Maasina-Fulfulde

Tabelle 2: Übersicht der Bedeutungen der Abkürzung *FFM*, bzw. *ffm* nach *FFM — Wikipedia, die freie Enzyklopädie* [84]

1.2 Die Ziele

Das Ziel dieser Arbeit ist die Implementierung eines mehrschrittigen Prozesses. Zuerst sollen BLL (Bibliography of Linguistic Literature)-Termini in linguistischen Texten erkannt werden (NER (Named Entity Recognition)). Diesen Schritt bezeichnet man teilweise als MD (Mention detection), denn die meisten Prozesse erkennen hier noch keine Entitäten. Stattdessen werden Kandidaten für Entitäten erkannt. Nachdem diese Kandidaten erkannt sind, werden im Schritt der NED (Named Entity Disambiguation) die gefundenen Kandidaten mit der KB abgeglichen. Um diesen Abgleich effizienter zu machen, wird oft für jede Mention eine Liste von möglichen Kanditation aus der KB extrahiert (English: „Candidate Generation“)[67] und der geeignetste Kanditat aus der KB ausgewählt. Dies ist insbesondere bei sehr breiten KBs wichtig, in denen Benennung von Entitäten mehrfach vorkommen; so listet Wikipedia für die Abkürzung *FFM*[84] neben der Bedeutung *Frankfurt am Main*[85] noch 14 weitere Bedeutungen, wie in Tabelle 2 zu sehen ist, zwischen denen aus dem Kontext entschieden werden müsste.

Im letzten Schritt wird die Mention im Text mit dem ausgewählten Kanditaten aus der KB annotiert bzw. verknüpft NEL (Named Entity Linking).

Durch eine solche Verlinkung können bei den Einträgen im Portal die Metadaten ergänzt werden und somit die Suche nach Artikeln zu einem bestimmten Thema vereinfacht werden. Es ist denkbar, die annotierten Texte direkt darzustellen und Nutzenden so ein „Nachschlagen“ von Begriffen in der BLL-Ontologie zu ermöglichen.

Im ersten Satz von *Frankfurt am Main — Wikipedia, die freie Enzyklopädie* werden alleine sieben

Frankfurt am Main	
(	
anhören	https://upload.wikimedia.org/wikipedia/commons/1/13/De-Frankfurt_a._M..ogg
?	https://de.wikipedia.org/wiki/Hilfe:Audio
/	
i	https://de.wikipedia.org/wiki/Datei:De-Frankfurt_a._M..ogg
)	
ist mit 759.224	
Einwohnern	https://de.wikipedia.org/wiki/Einwohnerentwicklung_von_Frankfurt_am_Main
(31. Dezember 2021) die	
bevölkerungsreichste Stadt	https://de.wikipedia.org/wiki/Liste_der_gr%C3%B6%C3%9Ften_St%C3%A4dte_in_Hessen
des Landes	
Hessen	https://de.wikipedia.org/wiki/Hessen
und die	
fünftgrößte	https://de.wikipedia.org/wiki/Liste_der_Gro%C3%9Fst%C3%A4dte_in_Deutschland
Deutschlands	https://de.wikipedia.org/wiki/Deutschland
.	

Tabelle 3: Übersicht der Verlinkungen des ersten Satzes von *Frankfurt am Main — Wikipedia, die freie Enzyklopädie*[85]

verschiedene weitere Wikipedia Entitäten und eine Wikimedia Entität verlinkt, wie in Tabelle 3 zu sehen ist.

Solch eine Verknüpfung von Informationen vereinfacht eine Auseinandersetzung mit den Inhalten und ermöglicht Metaanalysen.

1.3 Die Modelle

BERT ist ein state-of-the-art Modell für NLP (Verarbeitung natürlicher Sprache (Englisch: Natural Language Processing)) Aufgaben.[22] Es bietet sich somit für die in Abschnitt 1.2 beschriebene Aufgabe an. Es wurde in „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding“ von Devlin u. a. vorgestellt und basiert auf der Transformer Architektur.[22] BERT gehört zu einer neuen Generation von neuronalen Netzen, welche vortrainiert (English: „pretrained“) sind. Das bedeutet, dass der aufwendige Unsupervised Learning Teil des Trainings zwischen den verschiedenen Aufgaben geteilt wird. Für neue Aufgaben muss nur das finale Supervised Learning durchgeführt werden, damit das Modell auf die Aufgabe abgesimmt ist. Auf Englisch bezeichnet man ein abgestimmtes Modell als „fine-tuned“. Das GBERT Modell ist eine auf deutschsprachigen Daten trainierte Variante dieser Architektur von Chan, Schweter und Möller. [15] Die Transformer

Architektur ist für NLP vorteilhaft, weil sie beidseitigen Kontext für die Bewertung der Wörter nutzt. GBERT ist insbesondere geeignet, da dieses Modell bereits auf Deutsch als Sprache optimiert ist.

Dank exBERT kann man manuell das Modell erkunden und versuchen zu erkennen, welche Zusammenhänge bereits im Modell enthalten sind.

Sei der Beispielsatz „Frankfurt am Main (anhören?/i) ist mit 759.224 Einwohnern (31. Dezember 2021) die bevölkerungsreichste Stadt des Landes Hessen und die fünftgrößte Deutschlands“ ([85]) wieder gegeben und sei das betrachte Modell `bert-base-german-uncased`. Der Fokus wird auf den Token „Frankfurt“, „Main“ und „Hessen“ liegen. Ohne Maskierung wird der Token jeweils mit einer Sicherheit von 1 prognostiziert. Eine Maskierung eines einzelnen der drei Token führt nur bei „Hessen“ dazu, dass der Token als `[unused_punctuation2]` (also als „-“ nach Listing 1) vorhersagt werden würde[2]. Damit ist erkennbar, dass das `bert-base-german-uncased` bereits viele Informationen über Zusammenhänge erkennt. Dies liegt aber auch daran, dass dieses Modell unter anderem auf Wikipedia Daten trainiert wurde.

```
(                    |       [unused_punctuation4]
"                    |       [unused_punctuation3]
-                    |       [unused_punctuation2]
§                    |       [unused_punctuation8]
%                    |       [unused_punctuation10
,                    |       [unused_punctuation1]
&                    |       [unused_punctuation13
[                    |       [unused_punctuation14
.                    |       [unused_punctuation0]
)                    |       [unused_punctuation5]
:                    |       [unused_punctuation6]
<                    |       [unused_punctuation17
'                    |       [unused_punctuation12
/                    |       [unused_punctuation7]
```

[2]Eine Maskierung des Wortes „Frankfurt“ führt jedoch mit einer Konfidenz von 0.02 dazu, dass „Offenbach“ als Token vorhergesagt wird

```
+                    |        [unused_punctuation16
=                    |        [unused_punctuation18
#                    |        [unused_punctuation19
;                    |        [unused_punctuation9]
]                    |        [unused_punctuation15
*                    |        [unused_punctuation11
|                    |        [unused_punctuation20
##.                    |        .
##,                    |        ,
##-                    |        -
##"                    |        "
##(                    |        (
##)                    |        )
##:                    |        :
##/                    |        /
##§                    |        §
##;                    |        ;
##?                    |        ?
##%                    |        %
##*                    |        *
##'                    |        '
##&                    |        &
##!                    |        !
##_                    |        _
##[                    |        [
##]                    |        ]
##+                    |        +
##<                    |        <
##>                    |        >
##=                    |        =
###                    |        #
```

```
##@                    |       @
##$                    |       $
##|                    |       |
##\                    |       \
##{                    |       {
##}                    |       }
##~                    |       ~
##`                    |       `
##^                    |       ^
```

Listing 1: Änderungen im Vokabular des `bert-base-german-uncased` Tokenizers. Diff aus den beiden in [32] referenzierten `vocab.txt` Dateien. Links die Bezeichnungen wie sie ursprünglich verwendet wurden, rechts die aktuelle Darstellung der Token

2 Grundlagen und Definitionen

Im folgenden Abschnitt gibt es eine kurze Einführung zum Thema „maschinelles Lernen“. Danach werden die für diese Arbeit relevanten Problemstellungen formuliert, bevor die Datenquellen beschrieben werden. Nachdem die Datenquellen bekannt sind, wird die Erstellung von Trainingsdaten beschrieben, woraufhin kurz auf die Architektur von BERT (Bidirectional Encoder Representations from Transformers) eingegangen wird.

2.1 Maschinelles Lernen

2.1.1 Bewertung maschinellen Lernens

Die folgenden Metriken werden oft genutzt, um die Güte eines Modells darzustellen: Loss (auch Verlust- oder Kostenfunktion genannt), Recall (auch Sensitivität oder Empfindlichkeit genannt), Precision (auch Genauigkeit oder positiver Vorhersagewert genannt) und F1-Maß.

Zum besseren Verständnis werden im folgenden einige Begriffe eingeführt, bevor diese Metriken definiert werden.

Definition 1 (Typische Benennung bei binärer Kategorisierung)

Data ist hierbei die „Wahrheit“, während „Prediction“ die Kategorisierung durch das System ist.

		Prediction		
		Positive	***Negative***	$\sum$
Data	***Positive***	*True Positive*	*False Negative*	*Actual Positive*
	Negative	*False Positive*	*True Negative*	*Actual Negative*
	$\sum$	*Predicted Positive*	*Predicted Negative*	

Tabelle 4: Konfusionsmatrix (auch Wahrheitsmatrix genannt), welche die typischen Bezeichnungen der Merkmalsausprägungen für eine binäre Klassifikation zeigt

Für die Nachvollziehbarkeit wird in Tabelle 5 die Konfusionsmatrix mit beispielhaften Zahlen gefüllt.

		Prediction		
		Positive	**Negative**	$\sum$
Data	**Positive**	5	5	10
	Negative	90	9900	9990
	$\sum$	95	9905	

Tabelle 5: Beispielhafte Verteilung

Die Formeln für diese sind nach Shung wie folgt [72] definiert:

Definition 2 (Precision)

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} = \frac{True\ Positive}{Predicted\ Positive} \tag{1}$$

Somit wäre für Tabelle 5 die Precision $\frac{5}{95} \approx 5,3\%$.

		Prediction		
		Positive	**Negative**	$\sum$
Data	**Positive**	10	0	10
	Negative	9990	0	9990
	$\sum$	10 000	0	

(a) Optimistisches Modell

		Prediction		
		Positive	**Negative**	$\sum$
Data	**Positive**	0	10	10
	Negative	0	9990	9990
	$\sum$	0	10 000	

(b) Pessimistisches Modell

Tabelle 6: Konstante Modelle

	Optimistic	Pessimistic	Example
Precision	$0,1\%$	NA	$5,3\%$
Recall	100%	0%	50%
F1	100%	NA	9.5%

Tabelle 7: Tabellarischer Vergleich der Metriken an einem optimistischen, einem pessimistischen und dem ursprünglichen-Beispiel Modell

Definition 3 (Recall)

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} = \frac{True\ Positive}{Actual\ Positive} \tag{2}$$

Für unser Beispiel Tabelle 5 ergibt sich ein Recall von $\frac{5}{10} = 50\%$.

Definition 4 (F1-Maß)

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3}$$

Der F1-Wert ist also $2 \times \frac{\frac{5}{95} \times \frac{5}{10}}{\frac{5}{95} + \frac{5}{10}} = \frac{2}{21} \approx 9,5\%$.

Sei ein Modell gegeben, welches immer positiv bzw. immer negativ vorhersagt, dann sehen die Matritzen wie in Tabelle 6a bzw. in Tabelle 6b aus. Diese erhalten die Werte in Tabelle 7.

Definition 5 (Loss[62])

Seien $\mathbf{X} \in \mathcal{X}$ *und* $Y \in \mathcal{Y}$ *Zufallsvariablen, wobei* $\mathcal{X} \subset \mathbb{R}^d$ *und* $\mathcal{Y} = \{0, 1\}$. *Eine Loss-Funktion ist eine Abbildung* $l : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}^+$. *Hierbei ist ein kleinerer Wert besser. „Loss is*

the penalty for a bad prediction“ ([21])

Bei Loss wird also nicht ein gutes Modell „belohnt“, sondern ein schlechtes Modell „bestraft“. Für eine solche Loss Funktion l lässt sich das l-Risiko wie folgt definieren:

Definition 6 (l-Risiko[62])

Seien $\mathbf{X} \in \mathcal{X}$ und $Y \in \mathcal{Y}$ Zufallsvariablen, wobei $\mathcal{X} \subset \mathbb{R}^d$ und $\mathcal{Y} = \{0, 1\}$. Für eine beliebige Loss-Funktion l und eine Klassifikationsfunktion f, wird das l-Risiko definiert als:

$$R_l(f) = \mathbb{E}_{\mathbf{X},Y} l\left(f(x) \mid y_x\right) \tag{4}$$

wobei $\mathbb{E}$ den Erwartungswert bezeichnet und der Index die Zufallsvariablen bezüglich derer er gebildet wird.

2.1.2 Arten maschinellen Lernens

Maschinelles Lernen ist eine Unterkategorie von AI (Artficial Intelligence). Das Ziel ist dem Computer die Möglichkeit zu geben zu lernen, ohne explizit dafür programmiert zu sein [87].

Wolfewicz unterteilt es in die drei folgenden großen Kategorien:

Definition 7 (Supervised Learning [87])

Maschinelles Lernen auf Basis von bereits annotierten Tupeln der Form $(X, f(X))$, wobei X eine Eingabe und $f(X)$ die erwartete Ausgabe ist, nennt man Supervised Learning. Solche Tupel werden oft auch als `labeled data` bezeichnet.

Hierbei sind nach Wolfewicz die Hauptaufgaben Regression und Klassifikation.

Bei Klassifikation ist die Funktion definiert als $f : \mathcal{X} \to \mathcal{Y}$, wobei $\mathcal{Y} = \left\{\texttt{LABEL}_i \mid i = 1 \vdots l\right\}$ die Menge der möglichen Klassen ist. Der Wertebereich ist also diskret.[1] Bei Regression hingegen ist der Wertebereich kontinuierlich und es gilt im Allgemeinen $f : \mathcal{X} \to \mathbb{C}^l$ bzw. $f : \mathcal{X} \to \mathbb{R}^l$.

Das so trainierte Modell ist ein Funktionsapproximation für f. Es gibt für eine Regressionsaufgabe

eine Fortsetzung und Vereinfachung der bisherigen Funktion und für Klassifikationsaufgaben Label für bisher unbekannte Eingaben.

Neuere Methoden reduzieren den Bedarf an Trainingsdaten, indem sie diese nach bestimmten Regeln erzeugen. Dies kann durch Einbindung externer Systeme geschehen, wie z.B. bei Distant Supervision und Reinforcement Learning oder nur auf Grundlage der Daten.

Definition 8 (Distant Supervision[23])
Unter Verwendung einer bestehenden KB (Knowledge Base) wird die Eingabe mit einer erwarteten Ausgabe kombiniert, um `labeled data` *für Supervised Learning zu erzeugen.*

Definition 9 (Reinforcement Learning [87])
Hierbei gibt im Gegensatz zu Supervised Learning keine Tupel, sondern eine bewertende Instanz, die mitteilt, ob eine bestimmte Aktion oder Entscheidung des Modells „gut“ oder „schlecht“ war.

Definition 10 (Unsupervised Learning [87])
Beim Unsupervised Learning sind die Daten nicht weiter klassifiziert. Die Tupel $(X, f(X))$ *können also nur automatisch erzeugt werden. Diese Art des Lernens nennt man auch „self-taught learning“ ([55]).*

Da die Modelle meist durch Tensorberechnungen abgebildet werden, ist eine Darstellung der Informationen als dünnbesetzter Vektor von Vorteil. So müssen weniger einzelne Berechnungen durchgeführt werden und gleichzeitig sinkt durch eine solche Darstellung der Speicherbedarf.[31] Die Darstellbarkeit von Informationen als dünnbesetzte Vektoren in einer entsprechenden Basis wurde von Olshausen und Field bereits 1996 in „Emergence of simple-cell receptive field properties by learning a sparse code for natural images“ gezeigt. [57] Solch eine Darstellung kann ein System durch Optimierung nach einer Loss-Funktion, die Informationserhaltung und Dünnbesetztheit betrachtet, im Rahmen des Unsupervised Learning einfach lernen. [57, Formel 2-4]

2.2 NLP (Verarbeitung natürlicher Sprache (Englisch: Natural Language Processing))

Bei NLP (Verarbeitung natürlicher Sprache (Englisch: Natural Language Processing)) geht es um die Verarbeitung und Interpretation von menschlicher Sprache. Das Ziel ist es Informationen zu gewinnen.

2.2.1 NER (Named Entity Recognition)

Der Prozess der NER (Named Entity Recognition) lässt sich nach [51] als Lösung des folgenden mathematischen Problems beschreiben:

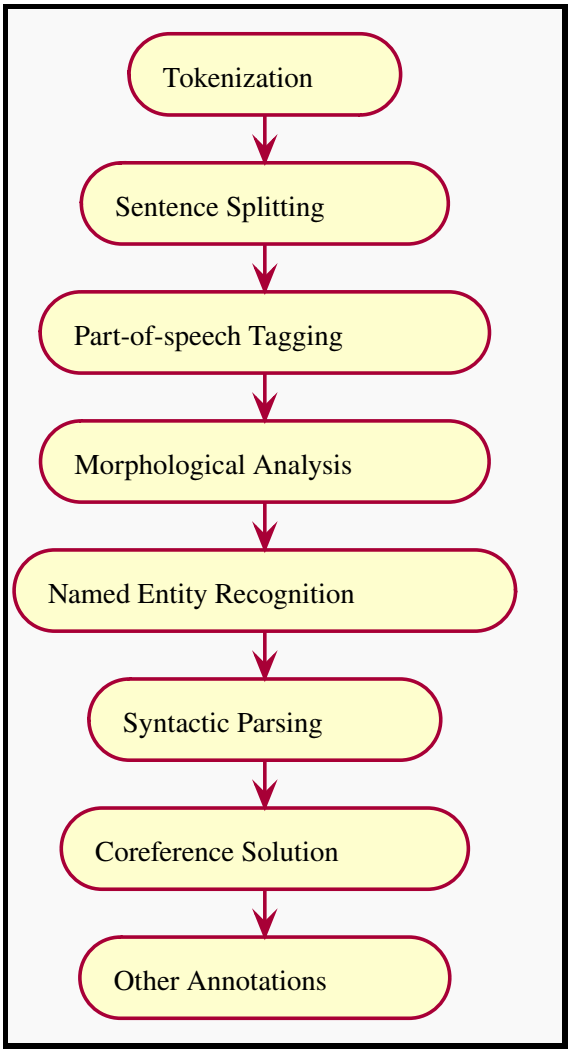
Problemstellung 1 (NER (Named Entity Recognition))

Für einen Satz $\mathbf{X} = [x_1, \ldots, x_N]$ *wird eine Beschriftung* $\mathbf{Y} = [y_1, \ldots, y_N]$ *gesucht. Die* `BIO`*-Klassifikation von Li u. a. [50] fordert hierbei, dass* $y_i \in \{$`B-X`, `I-X`, `O` : `X` *ist Entitätstyp*$\}$ *und dass gilt:*

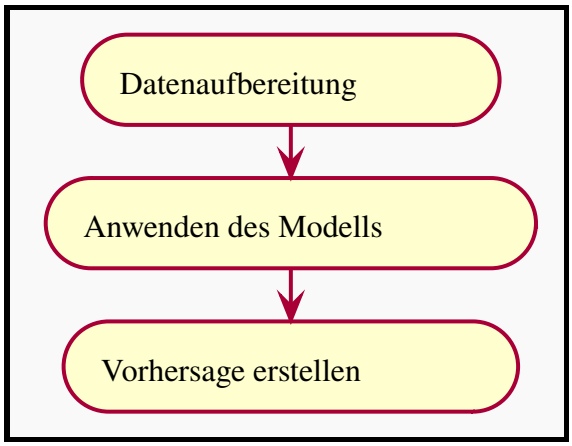
$$y_i = \begin{cases} \texttt{B-X}, \text{wenn } x_i \text{ der Beginn einer Entität vom Typ } \texttt{X} \text{ ist} \\ \texttt{I-X}, \text{wenn } x_i \text{ ein Token einer Entität vom Typ } \texttt{X} \text{ und nicht deren Beginn ist} \\ \texttt{O}, \text{sonst}. \end{cases}$$

Da die Sätze meistens in Tokens aufgeteilt werden, spricht man auch von einer Tokenklassifizierung-Aufgabe.

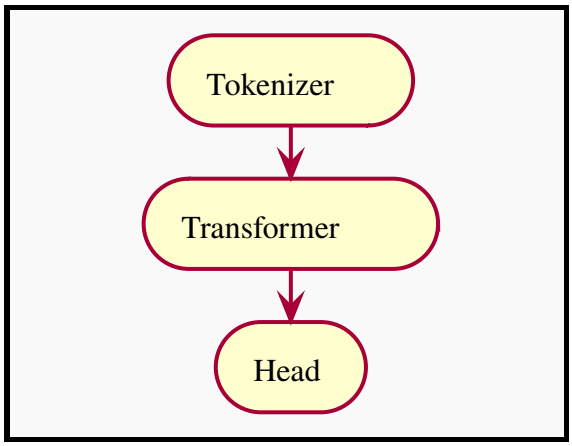
Der klassische Ansatz zur Erzeugung einer Beschriftung ist es, die Eingabe in mehreren Schritten die Eingabe um Informationen (Features) anzureichern. Diese werden in den jeweils darauf folgenden Schritten, unterstützend eingesetzt. Dazu gehören z.B. „häufig vorher genannte Wörter (sowie z.B. bei Orten das Wort „in"), Darstellungsformat (bei Daten so etwas wie Zahl Monat Jahr), Groß- und Kleinschreibung“ ([68, §19]) oder auch die „Position im Satz“ ([68, §19]). Eine Darstellung einer beispielhaften NLP-Architektur ist in Abbildung 2a zu sehen.



(a) Darstellung der Architektur von Stanford CoreNLP [53, Figure 1]



(b) Aufbau der Pipeline für die Datenverarbeitung



(c) Aufbau eines Transformer Modells: Der Tokenizer übersetzt die Eingabe in dünnbesetzte Vektoren über einer entsprechenden Basis. Diese Vektoren bzw. Tensoren werden dann durch den eigentlichen Transformer in eine kontextuelle Einbettung überführt. Diese wird dann durch den Head auf eine aufgabenspezifische Vorhersage umgerechnet.

Abbildung 2: Darstellung von Architekturen und Pipelines

Ein gängiger Aufbau für eine NLP Pipeline auf Basis von maschinellem Lernen ist nach [86] der in Abbildung 2b dargestellte. Hierbei werden keine weiteren Features definiert und das System muss diese selber erkennen. Tenney, Das und Pavlick stellen in „BERT Rediscovers the Classical NLP Pipeline“ jedoch fest, dass auch BERT teilweise ähnliche Zwischenschritte wie in der klassischen NLP-Pipeline abbildet.[77]

In der Regel besteht ein solches NLP-Modell aus drei Stufen, wie in Abbildung 2c dargestellt ist. Der Tokenizer wandelt den Text in eine numerische Repräsentation um. Diese wird durch den Transformer in einen Tensor aus Wahrscheinlichkeiten umgerechnet. Der head erstellt aus diesem Wahrscheinlichkeitstensor eine Interpretation für die Aufgabe.

Auf der Ebene des Tokenizers gibt es Unterschiede. In [90, Table 1] sind diese einfach zu erkennen,

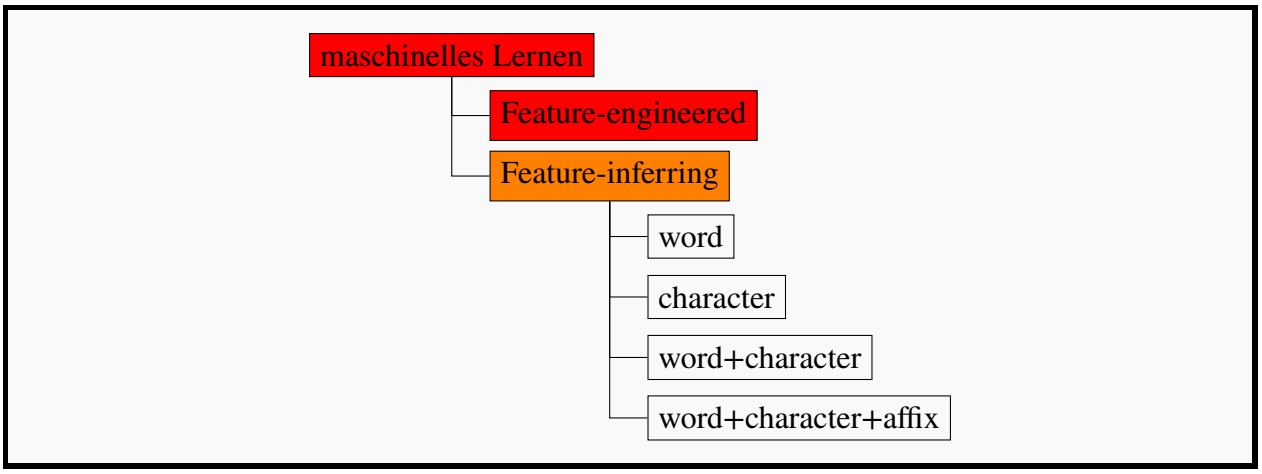


Abbildung 3: Übersicht der Systemtypen für maschinelles Lernen im Bereich NLP nach [90, Table 1]
Bei Feature-engineered Systemen werden die Features durch Menschen definiert, während Feature-inferring Systeme diese selbstständig aus den Trainingsdaten generieren.

Definition 11 (Güte eines Modells)

Seien M bereits annotierte Sätze gegeben und beschreibe $f(\mathbf{X};\Theta)$ ein NER-Modell, wobei $\mathbf{X}$ ein Satz und Θ die Parameter des Modells sind. Dann lässt sich anhand von Problemstellung 1 die Güte des Modells über eine Metrik bestimmen.

Eine beliebte Metrik im Rahmen des maschinelles Lernen ist die Kreuzentropie-Kosten-Funktion, wie sie in [43, Abschnitt 5.5] definiert wird: $l(y, \hat{y}) = -\left[y \log \hat{y} + (1 - y) \log (1 - \hat{y})\right]$. Hierbei wäre der ideale Fall, dass jeder Satz Labels entsprechend der vorgegeben (idealen) Annotation zugewiesen bekommt.

Theorem 1

Dementsprechend werden die Parameter so gesucht, dass sie diese Distanz über alle Trainingsdaten minimieren. Somit sind die idealen Parameter für die M Sätze durch die Formel [51, S. 1] gegeben:

$$\hat{\Theta} := \arg\min_{\Theta} \frac{1}{M} \sum_{m=1}^{M} l\left(\mathbf{Y}_m, f\left(\mathbf{X}_m; \Theta\right)\right) \tag{5}$$

2.2.2 NEL (Named Entity Linking)

Das Ziel von NEL (Named Entity Linking) ist es die die per NER erkannten Entitäten mit einer KB zuverknüpfen. Dies lässt sich als folgendes Problem darstellen.

Problemstellung 2 (NEL (Named Entity Linking))

Für einen Satz $\mathbf{X} = \left[x_1, \ldots, x_N\right]$ wird eine Beschriftung $\mathbf{Y} = \left[y_1, \ldots, y_N\right]$ gesucht, sodass $y_i \in \{Entitätsreferenz\} \cup \{NONE\}$.

Da es oftmals mehrere Kandidaten für die Verlinkung in der KB gibt, wird im Schritt der NED (Named Entity Disambiguation) eine ausgewählt.

2.3 Datenquellen

Im Folgenden werden die zwei Datenquellen, die für diese Arbeit verwendet wurden, beschrieben. Die erste Datenquelle ist die BLL-Ontologie, die für die Erstellung der Annotationen benötigt wird. Danach folgt das OAI–PMH (OAI – Protocol for Metadata Harvesting), welches die zu annotierenden Daten liefert.

2.3.1 BLL-Ontologie

Definition 12 (Thesaurus[7])

Ein Thesaurus ist eine Art von dokumentierender Sprache, die die konzeptuelle Struktur des Wissens eines bestimmten Fachgebietes darstellt. Er stellt den semantischen Zusammenhang zwischen verschieden Konzepten durch ein eingeschränktes Vokabular für die Beziehungen dar.

Das Ziel eines Thesaurus ist „synonymy“, also die Überlappung der Bedeutung von Wörtern [76], und „polysemy“, also die Mehrdeutigkeit der Bedeutung innerhalb eines Bereiches [64], zu reduzieren.

Im Folgenden werden Beispiele von Dateien in der turtle Syntax gezeigt [9]. Diese wird daher kurz erklärt.

Die Syntax besteht im Grunde nur aus Tripeln von *Subjekt*, *Prädikat* und *Objekt*. Zur Vereinfachung bleibt das aktuelle *Subjekt* bis zum nächsten *Punkt (.)* und das aktuelle *Prädikat* bis zum nächsten *Semikolon (;)* aktiv. Ein *Komma (,)* trennt in diesem Zusammenhang die *Objekte*. Bei StringLiterals ist eine I18N (Internationalization) möglich. `"TEXT"@de` ist somit als String für die Sprache Deutsch definiert.

Damit die Konzepte eineindeutig sind, werden sie über `URI` (Uniform Resource Identifier) beschrieben; zur Vereinfachung können diese aber durch global erklärte Präfixe ersetzt werden. Ein Beispiel hierfür ist in Listing 2.

Nach dem die Präfixe erklärt sind, sind die zwei beispielhaften Einträge in Listing 3 eindeutig definiert. Der Eintrag `bllt:bll-133103862` ist eine Klasse und ein Konzept. Er wird durch `"Seneca"` sowohl auf Deutsch, als auch auf Englisch und sowohl für das Konzept, wie auch für die Anzeige klassifiziert. Er ist eine Unterklasse von `bllt:BLLConcept`. Als gröbere Klassifizierung gilt `bllt:bll-133108791` (Irokesisch). Die Notation `"02.25.01.047.06"` repräsentiert diese Hierarchie, wie in Abbildung 4 zu sehen ist.

```
@prefix :       <http://data.linguistik.de/bll/bll-thesaurus#> .
@prefix bllo:   <http://data.linguistik.de/bll/bll-ontology#> .
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl:    <http://www.w3.org/2002/07/owl#> .
@prefix xml:    <http://www.w3.org/XML/1998/namespace> .
@prefix xsd:    <http://www.w3.org/2001/XMLSchema#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix skos:   <http://www.w3.org/2004/02/skos/core#> .
@prefix blli:   <http://data.linguistik.de/records/> .
@prefix rdfs:   <http://www.w3.org/2000/01/rdf-schema#> .
@prefix vann:   <https://vocab.org/vann/> .
@prefix bllt:   <http://data.linguistik.de/bll/bll-thesaurus#> .
@prefix foaf:   <http://xmlns.com/foaf/0.1/> .
```

Listing 2: Erklärung der globalen Präfixe in der `bll-thesaurus.ttl`

```
bllt:bll-133103862  a     owl:Class , skos:Concept ;
        rdfs:label        "Seneca"@en , "Seneca"@de ;
        rdfs:subClassOf   bllt:BLLConcept ;
        skos:broader      bllt:bll-133108791 ;
        skos:notation     "02.25.01.047.06" ;
        skos:prefLabel    "Seneca"@en , "Seneca"@de .

bllt:bll-467296421  a     owl:Class , skos:Concept ;
        rdfs:label        "Adult ADHD Rating Scales"@en , "Adult ADHD
         ↪ Rating Scales"@de ;
        rdfs:subClassOf   bllt:BLLConcept ;
        skos:broader      bllt:bll-133123936 ;
        skos:notation     "03.27.09.523" ;
        skos:prefLabel    "Adult ADHD Rating Scales"@en , "Adult ADHD
         ↪ Rating Scales"@de .
```

Listing 3: Beispieleinträge aus `bll-thesaurus.ttl`

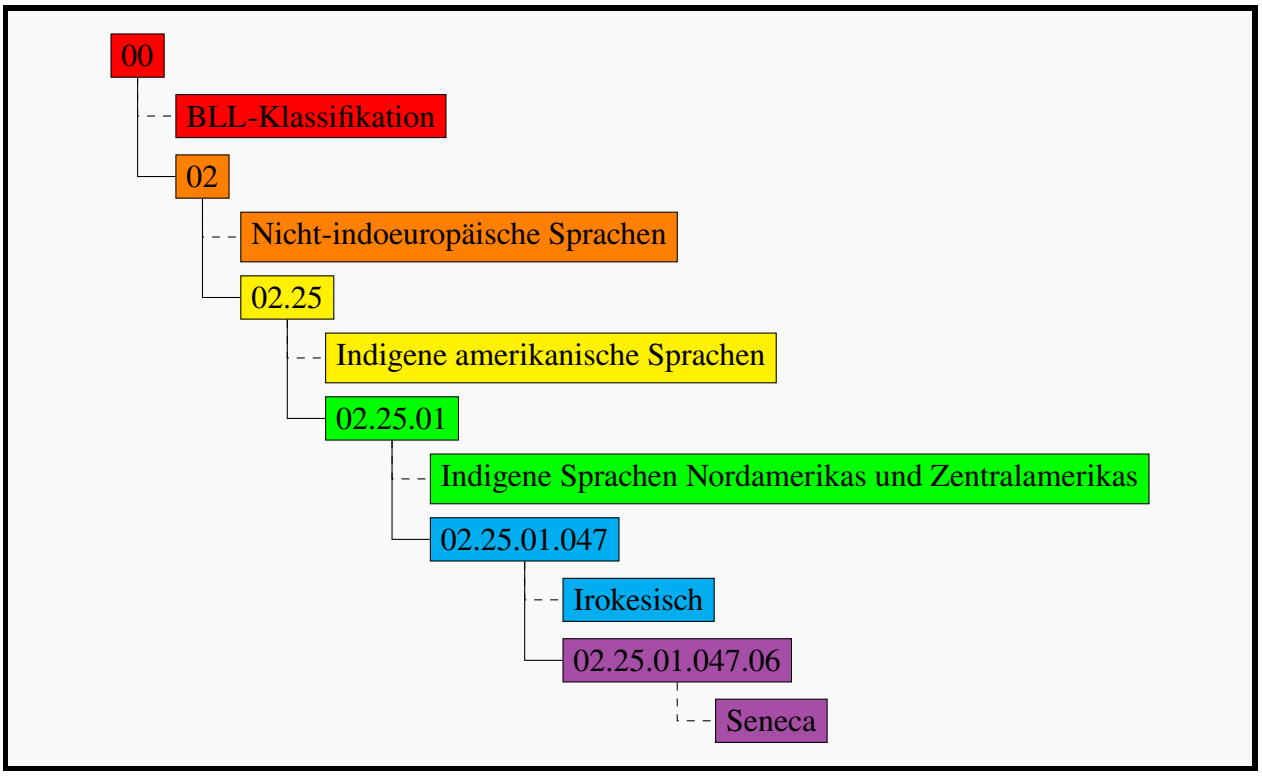


Abbildung 4: Darstellung der Notation des Eintrages Listing 3 als Baum mit den übergeordneten Klassen

Definition 13 (Ontologie[75][7])

„An ontology is a formal, explicit specification of a shared conceptualisation“ ([34, Abschnitt 1.] und [11, Abschnitt 2.1] via [75, Abschnitt 6.1]) wobei die folgenden Definitionen gelten:

1. *Konzeptualisierung : Ein abstraktes Modell eines Phänomens der realen Welt basierend auf den relevanten Konzepten des Phänomens*
2. *Explizit: der Typ des Konzepts und die Einschränkungen seiner Nutzing sind explizit definiert*
3. *Formal: Die Syntax ist präzise genug, um von einem Computer verstanden zu werden*
4. *Geteilt: das Wissen ist von einer Gruppe akzeptiert*

Die Konzeptualisierung wird von Arano noch weiter spezifiert, sodass sie eine Perspektive eine bestimmten Realität involvieren muss und diese auf der konzeptuellen Struktur einer KB begründet wird.
Das Ziel einer Ontologie ist das Teilen des Wissens, welches sie repräsentiert.

Da sich die Präfixe in der Ontologie von denen im Thesaurus unterscheiden, werden sie in Listing 4 noch einmal explizit dargestellt.

Nachdem die Präfixe erklärt sind, sind die zwei beispielhaften Einträge in Listing 5 eindeutig definiert. Das erste Beispiel ist wieder `"Seneca"`. Dieses Mal ist die Klassifizierung als „a

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix bllt: <http://data.linguistik.de/bll/bll-ontology#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

Listing 4: Erklärung der globalen Präfixe in der `bll-ontology.ttl`

```
bllt:bll-133103862 a owl:Class ;
    rdf:value "02.25.01.047.06" ;
    rdfs:label "Seneca"@de , "Seneca"@en ;
    rdfs:subClassOf bllt:BLLConcept, bllt:NorthernIroquoian ;
    skos:prefLabel "Seneca"@de , "Seneca"@en .

bllt:bll-133103870 a owl:Class ;
    rdf:value "04.03.13.004" ;
    rdfs:label "Capital letters/Small letters"@en ,
     ↪ "Grossschreibung/Kleinschreibung"@de ;
    rdfs:subClassOf bllt:AmbiguouslyDefinedConcept, bllt:BLLConcept ;
    owl:equivalentClass [
        a owl:Class ;
        owl:unionOf (bllt:CapitalLetter
            bllt:SmallLetter
        )
    ] ;
    owl:versionInfo "The primary BLL category Capital letters/Small
     ↪ letters was modeled as the disjunction of the newly created
     ↪ classes CapitalLetter and SmallLetter."@en ;
    skos:prefLabel "Capital letters/Small letters"@en ,
     ↪ "Grossschreibung/Kleinschreibung"@de .
```

Listing 5: Leicht umformatierte Beispieleinträge aus `bll-ontology.ttl` (Der Zeilenumbruch nach dem Subjekt ist jeweils entfernt, damit das Syntaxhighlighting mit Pygments funktioniert)

`skos:Concept`“ nicht mehr enthalten, dafür ist die neue Klassifikation als „`rdfs:subClassOf bllt:NorthernIroquoian`“ hinzugekommenen. Das Format dieser „Elternklasse“ zeigt an, dass es eine manuell angelegte Klasse ist, da der Name innerhalb des `bllt`-Namespaces keine Nummer ist. Das zweite Beispiel zeigt zusätzliche Prädikate in der OWL-Syntax (Web Ontology Language), wie Versionierungsinformationen und Äquivalenz.

Definition 14 (BLL (Bibliography of Linguistic Literature))

Die BLL (Bibliography of Linguistic Literature) ist eine Bibliographie, also ein Verzeichnis von Literaturnachweisen zu linguistischer Literatur, insbesondere der Allgemeinen Linguistik und „der anglistischen, germanistischen und romanistischen Sprachwissenschaft“ ([44]).

Zu dem oben genannten Verzeichnis existiert der BLL Thesaurus, welcher für die Indizierung

im Portal verwendet wird. Bei der Umwandelung des BLL Thesaurus in die BLL-Ontologie, die initial von Chiarcos u. a. durchgeführt wird [17], wird der Umfang an Begriffen eingeschränkt auf Konzepte aus den Zweigen „Syntax, Morphology, Lexicology and Phonology“ ([6]) und einigen zusätzlichen Einträgen aus den Zweigen „Graphemics and Semantics“ ([6]). Daher ist z.B. der Term `bllt:bll-467296421` aus Listing 3 nicht in der BLL-Ontologie verzeichnet.

2.3.2 OAI–PMH (OAI – Protocol for Metadata Harvesting)

Die OAI (Open Archives Initiative) hat mit OAI–PMH (OAI – Protocol for Metadata Harvesting) einen Standard für Interoperabilität zwischen verschiedenen Metadaten-Quellen und Dienstanbietern definiert. In Listing 6 ist der Anfang einer mit oaipmharvest erzeugten XML-Datei dargestellt.

```
<?xml version='1.0' encoding='UTF-8'?>
<?xml-stylesheet type="text/xsl" href="xsl/oai2.xsl"?>
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/"
 ↪ xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 ↪ xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
 ↪ http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
  <responseDate>2022-07-13T08:05:19Z</responseDate>
  <request verb="ListRecords" until="collections:Linguistik"
   ↪ metadataPrefix="oai_dc">http://publikationen.ub.uni-frankfurt.de/oai</
   ↪ request>
  <ListRecords>
    <record>
      <header>
        <identifier>oai:publikationen.ub.uni-frankfurt.de:744</identifier>
        <datestamp>2022-03-31</datestamp>
        <setSpec>doc-type:magisterthesis</setSpec>
        <setSpec>bibliography:false</setSpec>
        <setSpec>ddc</setSpec>
        <setSpec>ddc:400</setSpec>
        <setSpec>collections</setSpec>
        <setSpec>collections:Linguistik</setSpec>
        <setSpec>open_access</setSpec>
        <setSpec>open_access:open_access</setSpec>
      </header>
      <metadata>
        <oai_dc:dc xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
         ↪ xmlns:dc="http://purl.org/dc/elements/1.1/"
         ↪ xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/
         ↪ http://www.openarchives.org/OAI/2.0/oai_dc.xsd">
```

Listing 6: Beispiel Beginn eines OAI–PMH-Eintrags

Da der Inhalt innerhalb des `oai_dc:dc`-Tags durch das Dublin Core ™ Element Set[3] festgelegt ist, liegt hier der Fokus. Durch den Standard sind der Titel (`dc:title`), das Thema (`dc:subject`), welches oft als Liste von Schlagworten ausgedrückt wird, Beschreibungen (`dc:description`) und eindeutige Referenzen (`dc:identifier`) auf die Ressource als Einträge definiert. Auch die verwandten Ressourcen (`dc:relation`) und das Format (`dc:format`) sind enthalten.

Wenn der Volltext benötigt wird, müssen die `dc:identifier` oder die `dc:relation`-Einträge betrachtet werden. Je nach Anbieter können hier bereits Referenzen auf Volltext-Dokumente hinterlegt sein oder die Referenzen weisen nur auf Webseiten, auf welchen die Referenzen auf die Volltext-Dokumente gefunden werden müssen.

Sowohl für Problemstellung 1 als auch für Problemstellung 2 werden annotierte Daten benötigt.

2.4 Erstellung von Trainingsdaten

Da Trainingsdaten annotierte Daten sind, wird nachfolgend auf die Arten und die Erstellung von Annotationen eingegangen.

2.4.1 Arten der Annotation

Eine mögliche Annotation ist es, den Wörtern eines Satzes ihre grammatikalische Funktion oder andere semantische Informationen zuzuordnen. Wenn die semantische Information ist, dass die annotierte Entität eine „spezielle“ Entität ist, welche einen Namen hat, dann spricht man von NER. Auch eine Annotation über die Referenz auf die Entität ist möglich.

Da es keine bestehenden Annotationen für die Daten gibt, sollen hier Möglichkeiten für die automatische Erzeugung betrachtet werden.

2.4.2 Automatische Erzeugung von Annotationen

Liang u. a. listet als Möglichkeiten für die automatische Annotation „Stringmatching“, „`regex`“ und heuristische Verfahren [51].

[3] `https://www.dublincore.org/specifications/dublin-core/dces/1999-07-02/`

Bei ersterem wird ein Match nur erkannt, wenn es exakt ist. Dies führt dazu, dass die Wahrscheinlichkeit Wörter falsch zu erkennen, sehr niedrig ist. Die Rate der False Negative entsprechend steigt. Des Weiteren werden ausschließlich vorher bekannte Entitäten erkannt. So wird „Müller und Sohn“ erkannt, wenn es vorher aufgelistet wird. Der Transfer, dass „Meier und Sohn“ ebenfalls eine Entität ist, kann nicht stattfinden. Dazu kommt das Problem, dass insbesondere in Grammatiken, in denen Flexion wichtig ist, die gleiche Entität mit verschiedenen Strings repräsentiert werden kann. Die Laufzeit ist relativ hoch und nicht parallelisierbar. So liegt die Laufzeit von Algorithmus 1 in $\mathcal{O}\left(n \times k + l_E\right)$, wobei l_E die Anzahl der Wörter in der Wortliste beschreibt. Zusätzlich wird für das „Stringmatching“ eine Wortliste benötigt. Eine Liste von Entitäten kann z.B. durch Extraktion aus einer KB erstellt werden. In [51, A.1] von „BOND: BERT-Assisted Open-Domain Named Entity Recognition with Distant Supervision“ wird von Liang u. a. eine ganze Reihe von solchen KBs genannt. Eine solche Liste von Entitäten wird oft als Gazetteer (auch als Entitätsliste bezeichnet) bezeichnet (siehe z.B. [13, Introduction]).

Zur Erhöhung der Trefferrate kann die Liste erweitert werden. In [41, Abschnitt 3.4] werden Ansätze zur Erweiterung der Liste erklärt. Die Erkennungsaufgabe wird in der eben genannten Arbeit von FST (Finite-State Transducer) durchgeführt, welche eine Erkennung zuerst auf der Wortebene durchführen. Im Beispiel von oben würden so die Worte „Müller“, „und“ und „Sohn“ auf der Zeichenebene erkannt werden und danach die Entität „Müller und Sohn“ auf der Wortebene.

Der Begriff Wortebene passt nur eingeschränkt, denn in Sprachen, die wie das Deutsche von zusammengesetzen Wörtern leben, ist eine Einteilung in Wortbestandteile (diese werden in [41] als „lemma“ bezeichnet) zielführender.

Der Ansatz mit der Wortebene zeigt auf einen konzeptuell-anderen Ansatz: Es gibt bestimmte Formulierungen oder „Muster“, die oftmals eine Entität markieren. So ist ein „NAME und {Sohn,Söhne}“ insbesondere, wenn es in Anführungszeichen steht, oftmals eine Firma. Dementsprechend wird den Token dieser Sequenz die Kategorie ORG zugeordnet. Auch Ketten von Großbuchstaben sind oftmals Entitäten (Acronyme). Beide Beispiele lassen sich mit `regex` darstellen, siehe Listings 7a und 7b.

```
(?P<NAME>\w*)\s+und\s+(Sohn|Söhne)
```

(a) „Name und Sohn“ bzw. enquoteName und Söhne

```
(?P<acronym>[[:<:]][[:upper:]]+[[:>:]])\s+(?P<description>\((.|\s)+\))?
```

(b) Acronyme

Listing 7: Beispielhafte reguläre Ausdrücke

Die verschiedenen Ausgaben, der oben genannten FST, lassen sich als eine `regex` pro Ausgabe ansehen. Um die Ausführung zu beschleunigen, ist also eine parallele Prüfung mit verschiedenen `regex` denkbar. Das Problem, welches dabei auftritt, ist die Ambiguität, da den gleichen Token verschiedene Annotationen zugewiesen werden können.

Mit heuristischen Ansätzen, wie sie z.B. in [62] beschrieben werden, kann eine Entscheidung getroffen werden, welches Annotation die „beste“ ist. So könnte die häufigste oder auch die längste ausgewählt werden.

In dieser Arbeit wird aus Zeitgründen ausschließlich „Stringmatching“ implementiert.

2.4.3 Stringmatching

Stringmatching wird, wie in „Distantly Supervised Named Entity Recognition using Positive-Unlabeled Learning“ beschrieben, auch hier für die Generierung der Trainingsdaten verwendet.

Require: Wortliste D_E, Satz $s = \{w_1, \ldots, w_m\}$, Kontextlänge k

$i \leftarrow 1$

$l_1 \ldots l_m \leftarrow$ `unlabeled`

while $i < m$ **do**

  for $j = k; j--$ **do**

    if $w_i \ldots w_{i+j} \in D_E$ **then**

      $l_i, \ldots, l_{i+j} \leftarrow$ `Positive`

      $i \leftarrow i + j + 1$

      BREAK

    end if

    if $j == 0$ **then**

      $i \leftarrow i + 1$

      BREAK

    end if

  end for

end whilereturn teilweise annotierter Satz $\left(\{w_1, \ldots, w_m\}, \{l_1, \ldots, l_m\}\right)$

Algorithmus 1: Label mit Stringmatching: Longest Match nach [62]

Der ursprüngliche Algorithmus 1 wird in Algorithmus 2 wie folgt optimiert: Die verbleibende Länge des zu betrachtenden Strings wird in Betracht gezogen, um keine Matches zu suchen, die über den betrachteten String hinausragen. Zusätzlich werden nur Längen versucht, denen ein Wort in der Wortliste entspricht. Als Eingabe wird zusätzlich der Typ t, welcher für die Token genutzt werden soll, und ein Label l verwendet, sodass der gleiche Satz mit mehreren Wortlisten nacheinander annotiert werden kann. Hierbei werden bereits bestehende Label nicht überschrieben.

Require: Wortliste D_E, Satz $s = \{w_1, \ldots, w_m\}$,Label $l = \{l_1, \ldots, l_m\}$, Kontextlänge k, Token t

$lengths \leftarrow$ Liste der Wortlängen von D_E von gross nach klein sortiert

$lenghts \leftarrow lengths + 0$

while $i < m$ **do**

for $j \in lengths$ if $j < (m - 1)$ **do**

if $w_i, \ldots, w_{i+j} \in D_E$ **then**

if $l_v ==$ `unlabeled`$\forall v = i, \ldots, i + j$ **then**

$l_i \leftarrow \varepsilon B - \varepsilon + t$

$l_{i+1}, \ldots, l_{i+j} \leftarrow \ddot{I} - \varepsilon + t$

$i \leftarrow i + j + 1$

BREAK

end if

if $j == 0$ **then**

$i \leftarrow i + 1$

BREAK

end if

end if

end for

end whilereturn teilweise annotierter Satz $\left(\{w_1, \ldots, w_m\}, \{l_1, \ldots, l_m\}\right)$

Algorithmus 2: Label mit Stringmatching: Longest Match with Optimisations

Um das Named Entity Linking zu vereinfachen, wird für jeden Gazetteer eine eigene Klasse angelegt. So muss nach den jeweiligen Entitäten nur in einer KB gesucht werden. Für die Einheitlichkeit wird der Name des Gazetteer in Großbuchstaben verwendet. Z.B. `BLL-DE` für Einträge der BLL-Ontologie auf Deutsch. Inspiration hierfür sind die detaillierten Gazetteers von Leitner in „Eigennamen- und Zitaterkennung in Rechtstexten“ [48].

Dieser Algorithms wurde zur Vereinfachung von NEL um die Eingabe einer Liste von Entitätsreferenzen erweitert und die Wortliste D_E durch ein Wörterbuch ersetzt, sodass den Namen eine eindeutige Referenz auf die Entität zuordnet.

2.5 BERT

BERT (Bidirectional Encoder Representations from Transformers) ist ein Modell für NLP. Es wurde von Devlin u. a. auf Basis von Transformer entworfen, Diese Architektur beruht vor allem auf Aufmerksamkeitsmechanismen, welche in [81] vorgestellt wurde. Der Vorteil dieser Architektur bei NLP gegenüber den vorherigen LSTM-basierenden Architekturen ist es, dass beidseitiger Kontext für die Bewertung der Eingaben herangezogen werden kann.

2.5.1 Training des Modells

Das Modell wird auf `unlabeled data` mit zwei verschiedenen Aufgaben trainiert. Beim MLM (Masked Language Model) werden randomisiert 15% der Token maskiert. Da in späteren Aufgaben keine `[MASK]` Token mehr auftauchen, werden 10% der Token durch ein anderes Token und 10% der Token durch sich selbst „maskiert“. [22, 3.1 Task # 1]. Dieser Task ermöglicht dem Modell eine kontextuelles „Verständnis“ einzelner Wörter bzw. Token im Satz zu erlernen.

Die zweite Aufgabe ist NSP (Next Sentence Prediction). Hierbei werden zwei Sätze A und B aus dem Corpus gewählt, sodass in 50% der Fälle Satz B auf Satz A folgt und in 50% der Fälle B ein zufälliger Satz aus dem Corpus ist. [22, 3.1 Task # 2]. Die kontextuelle Einbettung des `[CLS]` Tokens wird hierbei für die Auswertung betrachtet. Das Modell lernt somit Zusammenhänge zwischen Sätzen.

Insgesamt entwickelt das Modell eine Darstellung der Informationen, wie sie in Abschnitt 2.1.2 beschrieben wird. Diese Darstellung wird von den `heads` verwendet, um die eigentlichen Aufgabenstellungen zu lösen. Verschiedenen `heads` sind am Beispiel der Transformers-Bibliothek in Tabelle 8 aufgelistet.

Beim „fine-tuning“ werden die einzelnen Parameter des bereits trainierten Netzwerks für den

Heads				
Name	Input	Output	Tasks	Ex. Datasets
Language Modeling	$x_{1:n-1}$	$x_n \in \mathcal{V}$	Generation	WikiText-103
Sequence Classification	$x_{1:N}$	$y \in \mathcal{C}$	Classification, Sentiment Analysis	GLUE, SST, MNLI
Question Answering	$x_{1:M}$,$x_{M:N}$	y span $[1 : N]$	QA, Reading Comprehension	SQuAD, Natural Questions
Token Classification	$x_{1:N}$	$y_{1:N} \in \mathcal{C}^N$	NER, Tagging	OntoNotes, WNUT
Multiple Choice	$x_{1:N}, \mathcal{X}$	$y \in \mathcal{X}$	Text Selection	SWAG, ARC
Masked LM	$x_{1:N\setminus n}$	$x_n \in \mathcal{V}$	Pretraining	Wikitext, C4
Conditional Generation	$x_{1:N}$	$y_{1:M} \in \mathcal{V}^M$	Translation, Summarization	WMT, IWSLT, CNN/DM, XSum

Tabelle 8: Übersicht der verschiedenen „Heads“ aus [86, Figure 2, top]. Die Token Sequenzen $x_{1:N}$ stammen hierbei aus einem Vokabular $\mathcal{V}$, während die y z.B. aus einer Menge von Klassen $\mathcal{C}$ stammen kann.

expliziten Task optimiert. [30] Dies kann durch verschiedene Mechanismen geschehen. Der einfachste Ansatz sind randomisierte *kleine* Änderungen der Parameter. Modernere Ansätze gehen hier jedoch gezielter vor und betrachten z.B. Gradienten.

2.5.2 Eingaben des Modells

Mögliche Eingaben in das Modell sind in Tabelle 8 zu sehen. Die Token x_i werden durch Zahlen identifiziert in welche sie von einem `Tokenizer` umgewandelt werden. Der BERT-Tokenizer benutzt sogenannte „sub word“-Token. Hierbei kann ein Wort aus mehreren Token bestehen. Die typische `##ing` des englischen Gerund, wird hierbei der z.B. der Zahl 270 zugeordnet.

Da die tieferen Funktionsweisen von BERT durch die verwendeten Bibliotheken abstrahiert sind, wird nicht weiter auf die Aufmerksamkeitsmechanismen und die Interpretation durch die heads eingegangen.

```
Abkürzung        http://data.linguistik.de/bll/bll-ontology#bll-133075222
Abstraktum        http://data.linguistik.de/bll/bll-ontology#AbstractWord
Akkusative Ausrichting        http://data.linguistik.de/bll/bll-
 ↪  ontology#AccusativeAlignment
Akkusativ (Kasus)        http://data.linguistik.de/bll/bll-ontology#AccusativeCase
Adamawa        http://data.linguistik.de/bll/bll-ontology#Adamawa
```

Listing 8: Beispielhafte Einträge aus dem `bll.de.dict` Gazetteer, welches zusätzlich zum Wort noch eine `URI` für das NEL enthält

3 Umsetzung

Die Verarbeitung der verschiedenen Eingaben geschieht für das Training in der in Abbildung 5 dargestellten Pipeline, während die Verarbeitung der produktiven Daten in einer Pipeline, wie sie in Abbildung 6 dargestellt ist, erfolgen würde.

Im Folgenden werden die einzelnen Abschnitte der Implementierung näher erläutert.

3.1 Datenquellen

Zuerst werden hier die beiden Quellen der Eingaben erklärt.

3.1.1 Erstellung eines Gazetteer aus der BLL-Ontologie

Um eine initiale Annotation der Texte durchzuführen, wird aus der BLL-Ontologie ein Gazetteer der Anzeigenamen und der Namen der Objekte erstellt.

Das Format für die Gazetteers, welches in Listing 8 zu sehen ist, besteht aus der Named Entity, einem Tabulator und der `URI`, welche die Named Entity eindeutig bezeichnet. Der Dateimame in Großbuchstaben wird als Tag-Typ verwendet.

Für einen vereinfachen Umgang mit den Gazetteers sind in `src/oaipmh/gazetteer.py` Helferfunktionen `read_gazetteers`, `read_gazetteer` und `write_gazetteer` definiert, deren Signatur in Listings 9a bis 9c dargestellt ist.

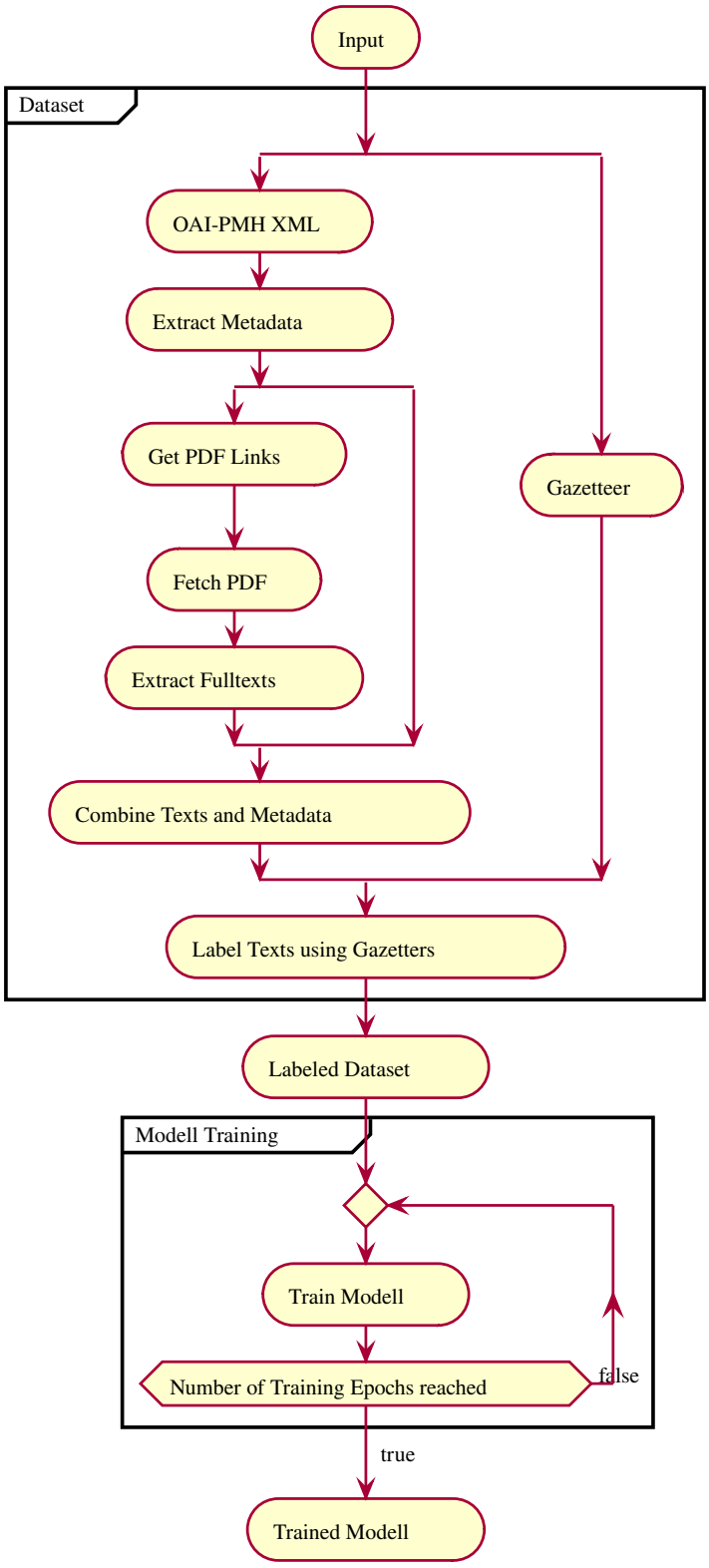


Abbildung 5: Aufbau der Pipeline für das Training des Modells

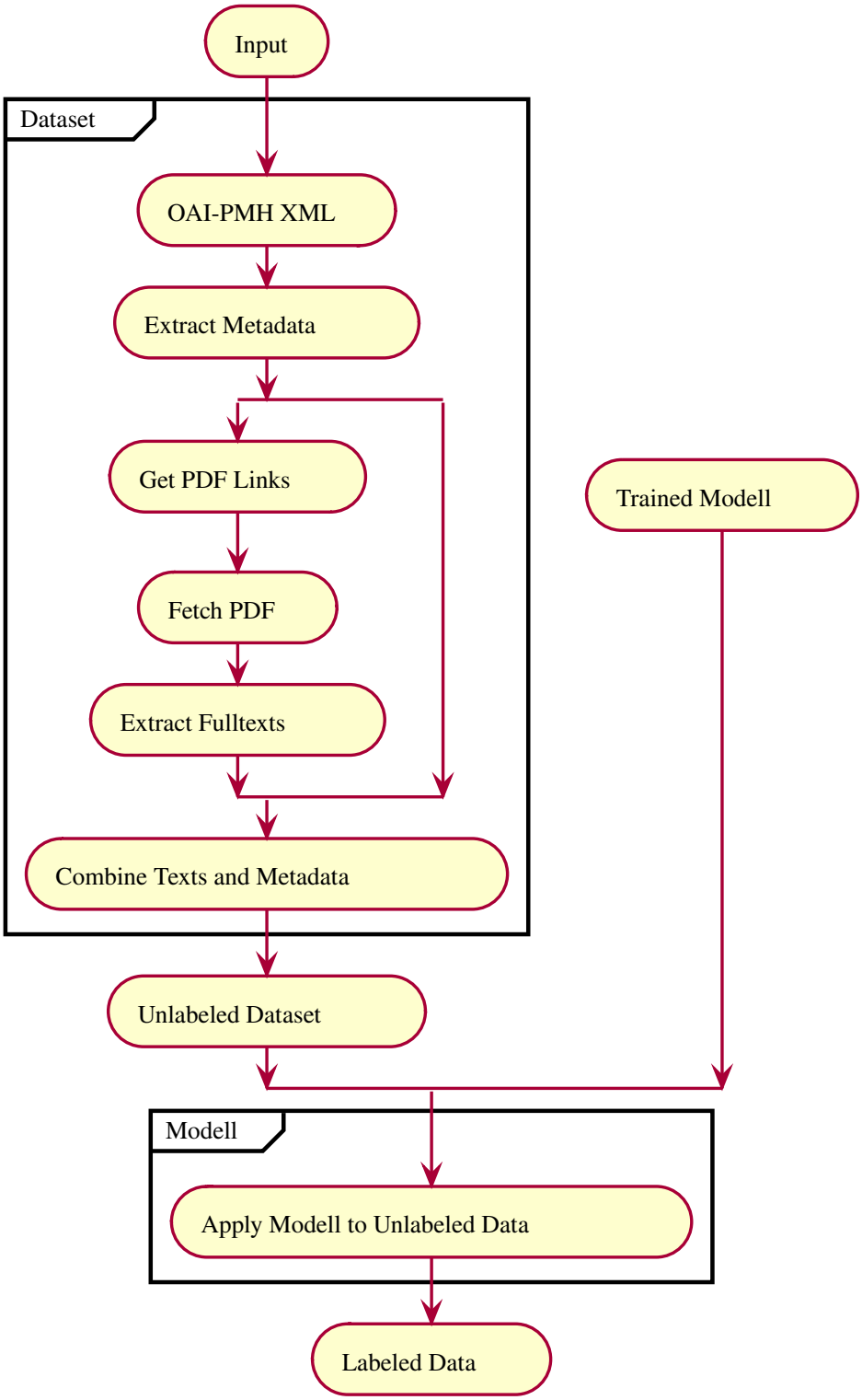


Abbildung 6: Aufbau der Pipeline für die Datenverarbeitung

```python
def read_gazetteers(
    dir_or_files: Union[list[Union[str, Path]], Union[str, Path]]
) -> dict[str, Gazetteer]:
```

(a) `read_gazetteers`

```python
def read_gazetteer(file: Union[str, Path]) -> Gazetteer:
```

(b) `read_gazetteer`

```python
def write_gazetteer(file: Union[str, Path], gazetteer: Gazetteer):
```

(c) `write_gazetteer`

Listing 9: Signaturen der Helferfunktionen in `src/oaipmh/gazetteer.py` für den Umgang mit Gazetteers

3.1.2 Extraktion der Metadaten

Statt die Metadaten aus dem Lin|gulis|tik-Portal zu extrahieren, werden die Daten mit Hilfe von oaipmharvest manuell direkt von den Anbieterseiten extrahiert. Die so gewonnenen XML-Dateien werden im nächsten Schritt weiter bearbeitet. Durch diesen Weg kann auf die Besonderheiten jedes Anbieters eingegangen werden.

```
        <dc:title xml:lang="deu">Explizit performative Äußerungen : Deklarationen,
         ↪ Feststellungen oder indirekte Sprechakte?</dc:title>
        <dc:creator>Runkel, Andreas</dc:creator>
        <dc:subject>Pragmatik </dc:subject>
        <dc:subject>Sprechakttheorie </dc:subject>
        <dc:subject>Austin, John L. </dc:subject>
        <dc:subject>Searle, John R.</dc:subject>
        <dc:subject>Illokutiver Akt </dc:subject>
        <dc:subject>Illokutionäres Verb</dc:subject>
        <dc:subject>ddc:400</dc:subject>
```

```
<dc:description xml:lang="deu">Die vorliegende Arbeit befasst sich mit
 ↪ einem spezifischen Phänomen innerhalb der Sprachphilosophie
 ↪ [...]</dc:description>
<dc:date>2007-07-31</dc:date>
<dc:type>magisterthesis</dc:type>
<dc:type>doc-type:magisterthesis</dc:type>
<dc:format>application/pdf</dc:format>
<dc:identifier>http://publikationen.ub.uni-frankfurt.de/frontdoor/index/
 ↪ index/docId/744</dc:identifier>
<dc:identifier>urn:nbn:de:hebis:30-47103</dc:identifier>
<dc:identifier>http://nbn-resolving.de/urn/
 ↪ resolver.pl?urn:nbn:de:hebis:30-47103</dc:identifier>
<dc:identifier>http://publikationen.ub.uni-frankfurt.de/files/744/
 ↪ Magisterarbeit_Andreas_Runkel.pdf</dc:identifier>
<dc:language>deu</dc:language>
<dc:rights>Deutsches Urheberrecht</dc:rights>
```

Listing 10: Beispiel Metadaten eines OAI–PMH-Eintrags von `ubffm` innerhalb des `oai_cd:dc`-Tags

Da das OAI–PMH-Protokoll standardisiert ist, treten Fallunterscheidungen erst auf, wenn die eigentlichen Volltexte geladen werden.

Des Weiteren nutzt die Universitätsbibliothek Johann Christian Senckenberg auch die `URL` (Uniform Resource Locator) der PDF-Dateien als `dc:identifier`, was dazu führt, dass die Extraktion sehr einfach ist. Andere Verlage haben die Links zu den eigentlichen PDFs nicht oder zumindest nicht klar erkennbar in die `oai:metadata` Einträge ihrer `oai:record` codiert.

3.2 Erstellung einer Datasets-Klasse

Nachdem im vorherigen Schritt die Daten gewonnen wurden, werden sie nun vorverarbeitet. Hierfür werden Helferfunktionen erstellt und in eine Datasets-Klasse integiert.

3.2.1 Extraktion der Daten aus den OAI–PMH XML-Dateien

Obwohl der Standard die Beschreibung der Elemente durch das Dublin Core ™ Element Set[4] vorschreibt, gibt es z.B. bei `dc:format` nur eine Empfehlung für die Art des Inhalts. Die Universitätsbibliothek Johann Christian Senckenberg folgt der Empfehlung und liefert den MIME-Typ (Multipurpose Internet Mail Extensions) der Dateien, während Language Science Press dies nicht tut.

Da bei `lang-sci-press` und per Definition nicht zwingend eine `URL` als `dc:identifiers` hinterlegt sein muss, muss auch hier nachgefiltert werden.

Die Extraktion ist sowohl für `ubffm` als auch für `lang-sci-press` als Quellen implementatiert. Um nicht für alle Einträge des html-Quellcodes der `lang-sci-press` Webseite durchsuchen zu müssen, werden auch die verwandten Resourcen (`dc:relation`) extrahiert, da hier bei `lang-sci-press` teilweise die PDF-Ressource enthalten ist (siehe Listing 11).

Ein Auswahl der Einträge aufgrund des Dateiformats ist bei den `ubffm` Einträgen möglich, da als `dc:format` entsprechend der Empfehlung der MIME-Typ der Ressource angegeben wird. Da `lang-sci-press` das Format von Einträgen, wie in Listing 12 zu sehen ist, nicht als MIME-Typ angibt, ist dieser Filter nicht implementiert.

[4] `https://www.dublincore.org/specifications/dublin-core/dces/1999-07-02/`

```
    <record>
      <header>
        <identifier>oai:publikationen.ub.uni-frankfurt.de:24325</identifier>
        <datestamp>2022-03-31</datestamp>
        <setSpec>doc-type:workingpaper</setSpec>
        <setSpec>bibliography:false</setSpec>
        <setSpec>collections</setSpec>
        <setSpec>collections:Linguistik</setSpec>
        <setSpec>open_access</setSpec>
        <setSpec>open_access:open_access</setSpec>
      </header>
      <metadata>
        <oai_dc:dc xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
        ↪ xmlns:dc="http://purl.org/dc/elements/1.1/"
        ↪ xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/
        ↪ http://www.openar
chives.org/OAI/2.0/oai_dc.xsd">
          <dc:title xml:lang="deu">Der irokesische Sprachtyp</dc:title>
          <dc:creator>Sasse, Hans-Jürgen</dc:creator>
          <dc:subject>Irokesische Sprachen</dc:subject>
          <dc:subject>ddc:497</dc:subject>
          <dc:description xml:lang="deu">Im folgenden wird versucht, eine
          ↪ Interpretation der funktionalen Gesamtzusammenhänge des irokesischen
          ↪ Sprachbaus zu geben.</dc:description>
          <dc:date>1988</dc:date>
          <dc:type>workingpaper</dc:type>
          <dc:type>doc-type:workingpaper</dc:type>
          <dc:format>application/pdf</dc:format>
          <dc:identifier>http://publikationen.ub.uni-frankfurt.de/frontdoor/index/⌋
          ↪ index/docId/24325</dc:identifier>
          <dc:identifier>urn:nbn:de:hebis:30:3-243259</dc:identifier>
          <dc:identifier>http://nbn-resolving.de/urn/⌋
          ↪ resolver.pl?urn:nbn:de:hebis:30:3-243259</dc:identifier>
          <dc:identifier>http://publikationen.ub.uni-frankfurt.de/files/24325/⌋
          ↪ AP09NF-Sasse%281988%29.pdf</dc:identifier>
          <dc:language>deu</dc:language>
          <dc:rights>Deutsches Urheberrecht</dc:rights>
        </oai_dc:dc>
      </metadata>
    </record>
```

Listing 11: Beispielhafter `dc:record` aus einer OAI–PMH-Datei von `ubffm`

```
    <record>
      <header>
        <identifier>oai:omp.langsci-press.org:publicationFormat/22</identifier>
        <datestamp>2015-03-05T08:26:54Z</datestamp>
        <setSpec>LangSci-Press:silp</setSpec>
      </header>
      <metadata>
        <oai_dc:dc xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
         ↪ xmlns:dc="http://purl.org/dc/elements/1.1/"
         ↪ xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         ↪ xsi:schemaLocation="http://
www.openarchives.org/OAI/2.0/oai_dc/
 ↪ http://www.openarchives.org/OAI/2.0/oai_dc.xsd">
          <dc:title xml:lang="0">Linguistic variation, identity construction and
           ↪ cognition</dc:title>
          <dc:creator>Drager, Katie K.; University of Hawai‘i at Mānoa</dc:creator>
          <dc:description xml:lang="en-US">Speakers use a variety of different
           ↪ linguistic resources in the construction of their identities
           ↪ [...]</dc:des
cription>
          <dc:publisher xml:lang="en-US">Language Science Press</dc:publisher>
          <dc:date>2015-11-03</dc:date>
          <dc:type xml:lang="en-US">Book</dc:type>
          <dc:format>Digital (DA)
                      </dc:format>
          <dc:identifier>https://langsci-press.org/catalog/book/75</dc:identifier>
          <dc:identifier>10.17169/langsci.b75.22</dc:identifier>
          <dc:identifier>978-3-946234-24-1</dc:identifier>
          <dc:source xml:lang="en-US">Language Science Press; </dc:source>
          <dc:language>eng</dc:language>
          <dc:relation>https://langsci-press.org/index.php/
           ↪ Language%20Science%20Press/catalog/view/75/22/295-1</dc:relation>
          <dc:relation>https://langsci-press.org/index.php/
           ↪ Language%20Science%20Press/catalog/view/75/22/320-1</dc:relation>
          <dc:relation>https://langsci-press.org/index.php/
           ↪ Language%20Science%20Press/catalog/view/75/22/321-2</dc:relation>
        </oai_dc:dc>
      </metadata>
    </record>
```

Listing 12: Beispielhafter gekürzter `dc:record` aus einer OAI–PMH-Datei von `lang-sci-press`

Das initiale Auslesen geschieht mittels der lxml Bibliothek, wie in Listing 13 zu sehen ist.

```
def load_xml_file(
    xml_file: typing.Union[str, bytes, os.PathLike]
) -> list[OAIXMLRecordDict]:
    """Return the records contained in a OAI PMH XML file

    Args:
        xml_file (typing.Union[str, bytes, os.PathLike]): Path to an OAI PMH XML
 ↪  file

    Raises:
        UserWarning: Warning in wrong xml format
        UserWarning: Warning on a parsing error

    Returns:
        list[OAIXMLRecordDict]: List of the extracted OAI PMH records
    """

    with open(xml_file) as file:
        tree = etree.parse(file)

    if tree.getroot().tag != "{http://www.openarchives.org/OAI/2.0/}OAI-PMH":
        raise UserWarning(f"The file {file} is not in the OAI-PMH format.")

    records: list[etree._Element] = get_record_metadata_nodes_for_lang(tree,
     ↪  lang="deu")
    _records: list[OAIXMLRecordDict] = [_parse_record(r) for r in records]

    if len(records) != len(_records):
        raise UserWarning("Length mismatch")
```

```
    return _records
```

Listing 13: Code zum Laden einer OAI–PMH XML-Datei. `src/oaipmh/xml_loader.py`

Hierbei werden die eigentlichen Einträge jeweils mit der in Listing 14 in das in Listing 15 dargestellte `OAIXMLRecordDict` Format umgewandelt.

```
def _parse_record(record: etree._Element) -> OAIXMLRecordDict:
    """Parse an xml element to a dict

    Args:
        record (lxml.etree._Element): OAI PMH record

    Returns:
        OAIXMLRecordDict: dictionary containing the metadata of an OAI PMH record
    """
    id = get_identifiers_from_record(record)[0]
    id_ = id.split(":")[-1]
    identifiers = get_metadata_identifiers(record)
    subject = get_metadata_subject(record)
    title = get_metadata_title(record)
    description = get_metadata_description(record, lang="eng")
    relations = get_metadata_relations(record)

    return OAIXMLRecordDict(
        id=id,
        id_=id_,
        identifiers=identifiers,
        subject=subject,
        title=title,
        description=description,
        relations=relations,
```

```
    )
```

Listing 14: Code zum Umwandeln eines in Listing 13 gewonnen OAI–PMH-Record in das in Listing 15 definierte Format. `src/oaipmh/xml_loader.py`

```python
@dataclass(kw_only=True)
class OAIXMLRecordDict:
    """Class for the OAI PMH records"""

    id: str
    id_: str
    identifiers: list[str]
    subject: list[str]
    title: list[str]
    description: list[str]
    relations: list[str]

    def keys(self) -> list[str]:
        return ["id", "id_", "identifiers", "relations", *self.text_keys()]

    def text_keys(self) -> list[str]:
        """ "return the keys holding the NER relevant data"""
        return ["subject", "title", "description"]

    # https://stackoverflow.com/questions/62560890/how-to-make-custom-data-class-
    ↪ subscriptable
    def __getitem__(self, item):
        return getattr(self, item)
```

Listing 15: Dedizierte Datenklasse für die einheitliche Typisierung der OAI–PMH-Records. `src/oaipmh/xml_loader.py`

Die verschiedenen `get_...` Funktionen, die in Listings 13 bis 15 verwendet werden, sind `XPath` ausdrücke, welche eine direkte Suche im XML-Baum darstellen. Mehr zur Syntax dieser Ausdrücke

kann in der Dokumentation[5] von lxml gefunden werden.

In einem nächsten Schritt kann hier auch eine Extraktion des Volltextes geschehen. Dafür muss je nach OAI-Anbieter (Open Archives Initiative) der Link zur PDF-Datei noch mittes Beautiful Soup aus den in den `dc:identifier` referenzierten Webseiten extrahiert werden. Der Code dafür ist in Listing 16 dargestellt.

```
def get_pdf_links(record: OAIXMLRecordDict, publisher: str = "") -> list[str]:
    """Get the potential fulltext links for an OAI PMH Record
    Args:
        record (OAIXMLRecordDict): Dict representing the record
        publisher (str, optional): Publisher of the record (Used to handle link
 ↪  extraction). Defaults to "".

    Returns:
        list[str]: URLs pointing to the potential fulltexts
    """
    if publisher == "ubffm":

        return [x for x in record["identifiers"] if x.lower().endswith(".pdf")]

    elif publisher == "lang-sci-press":

        return [
            # check if the pdf is in the relations
            req.url
            for rel in record["relations"]
            if (
                (
                    req := _peak_at_link(  # save the request in a temporary
                     ↪  variable  # only do a HEAD not a GET; no need to download
                     ↪  the pdf now
```

[5] https://lxml.de/xpathxslt.html

```
                        rel.replace("index.php/Language%20Science%20Press/", "")
                    )
                )
                and _check_pdf_request(req=req)
            )
        ] or [
            # fall back to extracting the urls from the page
            req.url
            for a in BeautifulSoup(requests.get(record["id"]).content).select(
                ".item.files > .pub_format_single > .pdf"
            )
            if (
                (
                    req := _peak_at_link(  # save the request in a temporary
                     ↪ variable
                        href
                        if isinstance(
                            str,
                            href := a[
                                "href"
                            ],  # pyright: ignore [reportGeneralTypeIssues]
                        )
                        else href[0]  # pyright: ignore [reportGeneralTypeIssues]
                    )
                )
                and _check_pdf_request(req=req)
            )
        ]

    else:
```

```
        logger.warning("Unsupported publisher")
        return []
```

Listing 16: Code zum Finden der URL, welche auf PDFs verweisen, in Abhängigkeit des Anbieters. src/oaipmh/publishers.py

Da nicht alle Referenzen valide Weblinks sind, wird mit Listing 17 eine Wrapper definiert, der None für diese zurückgibt.

```
def _peak_at_link(
    link: Union[Text, bytes],
) -> Union[requests.Response, None]:  # TODO replace with _peak_at_links
    """Peak at the provided link in an exception safe way

    Args:
        link (str): url to peak at

    Returns:
        Union[requests.Response,None]: Response of the request or None if it failed
    """
    req = None
    try:
        req = requests.head(link, allow_redirects=True)
    except Exception as N:
        logger.warning(f"Exception occured handling {link}: {N}")

    return req
```

Listing 17: Wrapper, um Requests in List Comprehensions mit Listen invalider Links zu nutzen src/oaipmh/helpers.py

Da bei den in Tabelle 9 aufgeführten sieben der 1027 Einträgen des ubffm-Providers keine URL gefunden wird, die auf eine entsprechende PDF Datei verweist, ist Fehlerbehandlung wichtig.

3.2.2 Extraktion der Volltexte

Da die Volltexte als PDF vorliegen, müssen die eigentlichen Texte extrahiert werden. Hierfür gibt es verschiedenen Werkzeuge und Bibliotheken, wie z.B. PyPDF2, pdfminer.six oder pdftotext.

Der erste Versuch der Extraktion wird mit PyPDF2 durchgeführt. Dieses analysiert die PDF seitenweise. Wenn dies nicht funktioniert, wird ein erneuter Versuch mit pdfminer.six gestartet. Sollte dies auch fehlschlagen, wird ein leerer Text zurückgegeben. Die Implementierung ist in Listing 18 dargestellt.

```python
from pathlib import Path
from typing import Union

import datasets

from PyPDF2 import PdfReader

# https://pypdf2.readthedocs.io/en/latest/user/suppress-warnings.html#exceptions
from pdfminer.high_level import extract_text as fallback_text_extraction


logger = datasets.utils.logging.get_logger("PdfExtractor")


def get_text_from_pdf(ft: Union[str, Path]) -> str:
    """Extract text from a pdf file

    Args:
        ft (Union[str,Path]): Path to the pdf file

    Returns:
        str: Text content of the pdf file or empty string if both extractors fail
    """
```

```
    text = ""

    try:
        reader = PdfReader(ft)
        logger.info(f"Handling {ft}")

        pbar = datasets.utils.logging.tqdm(
            reader.pages,
            unit="pages",
            # total=len(reader.pages),
            disable=not datasets.utils.logging.is_progress_bar_enabled(),
        )
        mapped = [page.extract_text() for page in pbar]
        text = "\n\n".join(mapped)

    except Exception as exc:
        logger.error(f"Error processing {ft} with pypdf2: {exc}")

        try:
            text = fallback_text_extraction(ft)

        except Exception as exc:
            logger.error(f"Error processing {ft} with pdfminer.six: {exc}")

    finally:  # ensure the text is always returned
        return text
```

Listing 18: Code zur Extraktion des Textes aus einer PDF-Datei. `src/oaipmh/extract_pdf.py`

3.2.3 Implementierung des Stringmatching

In Listing 19 ist die Implementierung des Algorithmus, der am Ende von Abschnitt 2.4.3 beschrieben wird, zu sehen.

```
    for i in range(start_idx, end_idx):
        sentence_label[i] = label



from collections.abc import Callable, Sequence
import typing

from logging import Logger

import datasets

from .gazetteer import Gazetteer

T = typing.TypeVar("T")  # Declare type variable

logger = datasets.utils.logging.get_logger("Matcher")



def matching_and_linking_BIO(
    gazetteer: Gazetteer,  # dictionary returning the entity identifier
    tokens: typing.Union[list[str], str],
    token_type: str,
    context_size: typing.Union[int, None] = None,
    tokenizer: Callable[[str], list[str]] = lambda x: x.split(),
    detokenizer: Callable[[list[str]], str] = lambda x: " ".join(x),
    label: typing.Union[list[str], None] = None,
    links: typing.Union[list[str], None] = None,
) -> tuple[list[str], list[str], list[str]]:
```

```
    """Dict matching based on arXiv:1906.01378

    Args:
        gazetteer (Gazetteer): Entries to match
        token_type (str): base label to use for the tokens
        context_size (typing.Union[int, None], optional): number of tokens to match.
↪   Defaults to None.
        tokenizer (function(str -> list[str]) , optional): functionn to create
↪   tokens from a String. Defaults to lambdax:x.split().
        detokenizer (function(list[str] -> str), optional): function to create
↪   strings from tokens. Defaults to lambdax:" ".join(x).
        label (typing.Union[list[str], None], optional): list of current labels.
↪   Defaults to None.
        links (typing.Union[list[str], None], optional): list of current links.
↪   Defaults to None.

    Returns:
        tuple[list[str], list[str], list[str]]: _description_
    """
    # TODO Add support for multiple gazetteers by overloading and removing the
     ↪  token_type

    if isinstance(tokens, str):
        _tokens = tokenizer(tokens)
    else:
        _tokens = tokens

    # normalize gazetteer for use with tokenizer and detokenizer
    _gazetteer = {detokenizer(tokenizer(key)): gazetteer[key] for key in gazetteer}

    PLACEHOLDER_LABEL = "O"
```

```
    lengths = list(set([len(tokenizer(l)) for l in _gazetteer]))
    lengths.append(0)
    lengths.sort(reverse=True)

    if context_size == None:
        context_size = max(lengths)

    # TODO use trie or other data structore to optimise matching

    i = 0
    n = len(_tokens)

    if not label:
        label = [
            PLACEHOLDER_LABEL.upper(),
        ] * n
    else:
        assert len(label) == n, "Wrong length of label list"

    if not links:
        links = [
            "",
        ] * n
    else:
        assert len(links) == n, "Wrong length of links list"

    while i < n:
        for j in lengths:
            if j <= n:
                # TODO Remove dependency on string like behaviour
                if (
```

```
                    s := detokenizer(_tokens[i : (upper_bound := min(i + j, n))])
                ) in _gazetteer:
                    logger.info(f"""Found: {s}""")
                    if label[i:upper_bound] == [PLACEHOLDER_LABEL.upper(),] * (
                        upper_bound - i
                    ):  # This way a shorter match can still be found
                        label[i:upper_bound] = [f"B-{token_type}".upper()] + [
                            f"I-{token_type}".upper()
                        ] * (upper_bound - i - 1)
                        links[i] = _gazetteer[s]
                        i = i + j + 1
                        break
                    else:
                        logger.info(
                            f"""Did not overwrite labels {label} and links {links}
                             ↪  for {(s,gazetteer[s],token_type)} in {_tokens}"""
                        )
                        continue
                if j == 0:
                    i += 1
                    break

    return _tokens, label, links
```

Listing 19: Implementierung des Stringmatching basierend auf Algorithmus 2. `src/oaipmh/dict_matcher.py`

3.2.4 Einbettung in Datasets

Um die Verwendung von Transformers zu vereinfachen, werden die bisher definierten Methoden in Datasets eingebunden.

Hierfür wird zunächst die `OAIPMHConfig` Konfigurationsklasse erstellt, deren Parameter in

Listing 20 dargestellt sind.

```
    def __init__(
        self,
        # citation,
        # url,
        publisher: str = "",
        extract_fulltexts: bool = False,
        do_string_match: bool = False,
        language: typing.Union[str, None] = "all",
        size: int = 100,
        token_label_classes: Union[List[str], None] = None,
        link_label_classes: Union[List[str], None] = None,
        gazetteers: Union[dict[str, Gazetteer], None] = None,
        oaipmh_xml_files: Union[list[Path], list[str], None] = None,
        gazetteer_files: Union[list[Path], list[str], None] = None,
        time_log: Union[Path, str, None] = None,
        **config_kwargs,
    ):
        """BuilderConfig for OAI PMH datasets class

        Args:
            publisher (str): publisher of the records (used to find the pdf)
            extract_fulltexts (bool): toggle to enable text extraction from the pdf
↪   files
            do_string_match (bool): toggle to enable creation of labels by using
↪   string matching
            language (typing.Union[str, None], optional): _description_. Defaults
↪   to "all".
            size (int, optional): number of entries to use. Defaults to 100.
            token_label_classes (Union[List[str], None], optional): list of labels
↪   used by the dataset for NER tags. Defaults to None.
```

```
            link_label_classes (Union[List[str], None], optional): list of labels
↪  used by the dataset for NER links. Defaults to None.
            gazetteers (Union[dict[str, Gazetteer], None], optional): gazetteers
↪  used for string matching. Defaults to None.
            oaipmh_xml_files (Union[list[Path],list[str],None]), optional): OAI PMH
↪  XML Files containing the records. Defaults to None.
            gazetteer_files (Union[list[Path],list[str],None]), optional): Files
↪  containing the gazetteers. Defaults to None.
            time_log (Union[Path,str,None]): Path to log timings to. Defaults to
↪  None.

            **kwargs: Arguments passed to the parent class


        Raises:
            UserWarning: StringMatching not possible without gazetteer
            UserWarning: labels could not be found

        """
```

Listing 20: Beginn der Konfigurationsklasse `OAIPMHConfig` für das `OAIPMH` Dataset. `src/oaipmh/oaipmh.py`

Die Fehlerbehandlung in Listing 18, die immer mindestens den leeren Text zurückgibt, der am Anfang der Funktion definiert wird, ist an dieser Stelle wichtig, da die Funktion aus Listing 16 bei 8 der 1027 Einträgen auf Deutsch der `ubffm` keine `URL` gefunden wird. Die betreffenden `dc:identifier` sind in Tabelle 9 aufgelistet.

Zur Implementierung der `OAIPMH` Klasse muss zum einen der Typ der Konfigurationsklasse definiert werden und zum anderen müssen die `_info`, die `_split_generators` und die `_generate_examples` Methoden überschrieben werden. In der `_info` Methode werden hierbei die Features festgelegt, die das Dataset zurückgibt. Die `_split_generators` Methode liest die XML-Dateien mit der in Listing 13 dargestellten Methhode ein und lädt potentiell die PDFs herunter. Die Volltextextraktion

dc:identifier	Besonderheit
oai:publikationen.ub.uni-frankfurt.de:7101	Besteht aus einem Hauptband und einem Materialien Teil, die auf der Webseite referenziert werden
oai:publikationen.ub.uni-frankfurt.de:47810	Ist eine Zeitschrift
oai:publikationen.ub.uni-frankfurt.de:23611	Ist eine Zeitschrift
oai:publikationen.ub.uni-frankfurt.de:4620	Ist eine Zeitschrift
oai:publikationen.ub.uni-frankfurt.de:30655	Ist eine Zeitschrift
oai:publikationen.ub.uni-frankfurt.de:48364	Ist eine Zeitschrift
oai:publikationen.ub.uni-frankfurt.de:33792	Ist eine Zeitschrift
oai:publikationen.ub.uni-frankfurt.de:34780	Besteht aus mehreren Dateien, die auf der Webseite referenziert werden

Tabelle 9: OAI–PMH-Records der ubffm, bei denen die Funktion aus Listing 16 keinen Treffer liefert, und die zusätzliche Information über die Besonderheiten des Record

und das Stringmatching werden potentiell in der _generate_examples Methode durchgeführt, bevor die einzelnen Datensätze zurückgegeben werden.

Diese können danach durch die Transformers-Bibliothek verwendet werden.

3.3 Verwendung von Transformers

3.3.1 Vorbereitung der Eingaben

Zuerst wird der Text in Tokens aufgeteilt, welche in numerische Werte übersetzt werden können. Diese numerischen Werte können durch das neuronale Netzwerk verarbeitet werden, da dieses aber eine Eingabe fixer Länge erwartet, müssen zu lange Sequenzen abgeschnitten („truncate“) und zu kurze Sequenzen ergänzt („pad“) werden. Die so bearbeiteten Werte werden anschießend in einen Tensor umgewandelt, mit dem die weiteren Berechnungen durchgeführt werden können. Diese werden wird von einem „preprocessor“ bzw. „tokenizer“ ausgeführt. [65]

3.3.2 Training

Da potentiell die Grenzen der so erzeugten Token nicht mit den Grenzen der Token des Datasets übereinstimmen, müssen die Annotationen angepasst werden. Die tokenize_and_align_labels Funktion, die in Listing 21 dargestellt ist, iteriert über die Token und weisst dem jeweils ersten Token

eines Worts die Annotation des Worts zu.[80]

```
# https://huggingface.co/docs/transformers/perf_train_gpu_one
import numpy as np
from datasets import Dataset, load_from_disk

ds = load_from_disk("../data/datasets/ubffm_ft_debug_dataset")

MODEL_NAME = "bert-base-german-cased"

from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)

CATEGORY_FIELD = f"all_ner_tags"



def tokenize_and_align_labels(examples):
    tokenized_inputs = tokenizer(
        examples["all_tokens"], truncation=True, is_split_into_words=True
    )

    labels = []
    for i, label in enumerate(examples[CATEGORY_FIELD]):
        word_ids = tokenized_inputs.word_ids(
            batch_index=i
        )  # Map tokens to their respective word.
        previous_word_idx = None
        label_ids = []
        for word_idx in word_ids:  # Set the special tokens to -100.
            if word_idx is None:
                label_ids.append(-100)
```

```
            elif (
                word_idx != previous_word_idx
            ):  # Only label the first token of a given word.
                label_ids.append(label[word_idx])
            else:
                label_ids.append(-100)
            previous_word_idx = word_idx
        labels.append(label_ids)

    tokenized_inputs["labels"] = labels
    return tokenized_inputs


tokenized_ds = ds.map(tokenize_and_align_labels, batched=True)

from transformers import AutoModelForTokenClassification
from gpu_utls import print_gpu_utilization, print_summary

print_gpu_utilization()

model = AutoModelForTokenClassification.from_pretrained(
    MODEL_NAME, num_labels=len(ds.features[CATEGORY_FIELD].feature.names)
)

print_gpu_utilization()

from transformers import DataCollatorForTokenClassification

data_collator = DataCollatorForTokenClassification(tokenizer=tokenizer)

# https://huggingface.co/docs/transformers/training
```

```
import numpy as np
import evaluate

metric = evaluate.load("accuracy")



def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    return metric.compute(predictions=predictions, references=labels)



default_args = {
    "output_dir": "tmp",
    "evaluation_strategy": "steps",
    "num_train_epochs": 1,
    "log_level": "error",
    "report_to": "none",
}

from transformers import TrainingArguments, Trainer, logging

logging.set_verbosity_error()

training_args = TrainingArguments(
    per_device_train_batch_size=1,
    # optim="adafactor",
    optim="adamw_torch",
    gradient_accumulation_steps=4,
    gradient_checkpointing=True,
    do_eval=False,
```

```
    output_dir="lang-sci-press_ft_debug_training",
    no_cuda=True,
)
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_ds,
    eval_dataset=tokenized_ds,
    tokenizer=tokenizer,
    data_collator=data_collator,
    compute_metrics=compute_metrics,
)

result = trainer.train()
print_summary(result)
```

Listing 21: Code zum Trainieren eines Modells `src/train.py`

4 Diskussion und Herausforderungen

Die Bewertung der Ergebnisse und die Herausforderungen der Arbeit werden hier dargestellt.

4.1 Bewertung der Implementierung

In Abschnitt 3.1.2 hätte statt einer gemeinsamen Klasse für alle Anbieter eine Klasse für jeden einzelnen die Implementierung und das Testen der Komponenten vereinfacht. Für den `ubffm` Provider fehlen noch ein Handler für `<dc:format>application/octet-stream</dc:format>`-Einträge, wie z.B. `oai:publikationen.ub.uni-frankfurt.de:7101`, und ein Filter für Einträge mit `<dc:type>periodical</dc:type>`-Tag, wie z.B. `oai:publikationen.ub.uni-frankfurt.de:23611`. Dies wurde bei der Erstellung von Tabelle 9 festgestellt.

	duration
count	1101.000000
mean	31.476439
std	66.191223
min	0.078202
25%	5.321660
50%	11.187993
75%	26.971283
max	1007.566026

Tabelle 10: Statistische Bewertung der Dauer der PDF-Extraktionen

In Tabelle 10 wird die durchschnittliche Dauer der Extraktion der PDF mit weniger als 32 Sekunden angegeben. In einzelnen Fällen dauert sie jedoch fast 17 Minuten.

4.2 Bewertung der Trainingsdaten

Die Trainingsdaten sind stark verezrrt, wie an der Dominanz ersten Balken in Abb. 7 und 8 trotz logarithmischer Skalierung zu erkennen ist.

Peng u. a. Bei einer zu ungleichmässigen Verteilung der Label kann mit Gewichten, wie in [62, Loss Definition] beschrieben, ein Ausgleich geschaffen werden.

4.3 Bewertung des Modells

Ein klassischer Ansatz zur Bewertung eines Modells ist die sogenannte Konfusionsmatrix bei der die Wahrscheinlichkeit, dass ein Token einer bestimmten Klasse zugeordnet wird, in Abhängigkeit von der echten Klasse des Tokens visualisiert wird.

Da kein Modell mit den finalen Daten trainiert ist, findet hier keine weitere Auswertung statt.

4.4 Allgemeine Herausforderungen

Die grundsätzlichen Probleme von Unsupervised Learning auf Basis von ungefilterten Daten, wie die Reproduktion von Vorurteilen, wie z.B. Sexismus oder Rassismus, welche in der Datenbasis

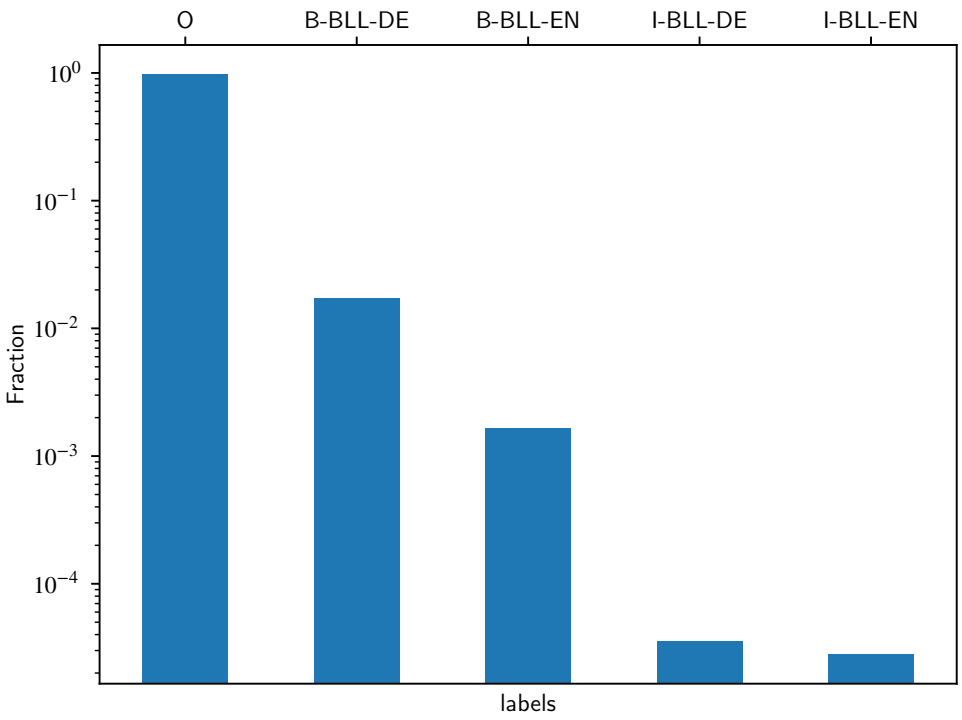


Abbildung 7: Darstellung der Verteilung der Tags im `SPEEDTEST_lang-sci-press_ft_debug`-Dataset

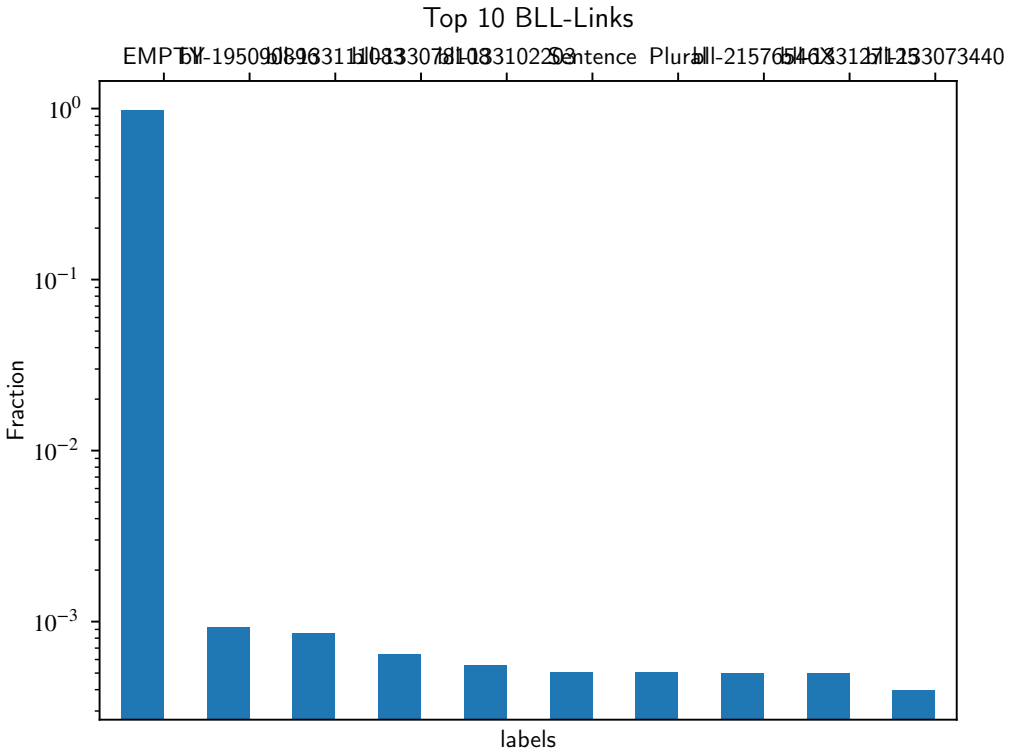


Abbildung 8: Darstellung der Verteilung der Top 10 Links im `SPEEDTEST_lang-sci-press_ft_debug`-Dataset

abgebildet sind, betreffen auch die hier verwendeten Modelle. Chan, Schweter und Möller haben versucht diese bei GBERT zu reduzieren [15, S. 3.1], aber warnen:

> The main portion (89%) of our training data, namely the OSCAR dataset, uses texts scraped from the internet, which is in some respects problematic. First off, this dataset contains a lot of explicit and indecent material. While we filtered out many of these documents through keyword matching, we cannot guarantee that this method was successful in every case ([15, S. 3.1])

Da hier jedoch keine Bewertung vorgenommen wird, sollte der Effekt gering bleiben.

4.5 Verschwundene Daten

Viele der Experimente der Quellen ließen sich nicht mehr reproduzieren, da selbst wenn der Code auf GitHub verfügbar war, die eigentlichen Datendateien nicht abrufbar waren. Im folgenden werden zwei Beispiele explizit aufgeführt.

The Annotated Transformer ist eine mit Code annotierte Fassung von „Attention Is All You Need“. Letzteres ist das Paper, welches die Transformer Architektur vorgestellt hat. Ersteres wäre also ein guter Ansatz für die Einarbeitung in diese Architektur gewesen. Da jedoch das in [38, Data Loading] verwendete `Multi30k` Dataset nicht geladen werden konnte, war diese Einarbeitung nicht so interaktiv wie geplant. Der Fehler beim Laden lag daran, dass PyTorch die Datei mit den Trainingsdaten[6] nicht herunterladen konnte. Das Problem war im Nachhinein bereits am 2022-06-01 bekannt, wie die nachträgliche Recherche zeigt[89] und die Datei ist mit dem Stand 2022-09-05 wieder verfügbar.

Bei AutoNER wird in den Beispielskripten auf ein bestimmtes Embedding [7] verwiesen, welches Stand 2022-09-05 immer noch nicht verfügbar ist[8]. Durch den GitHub Issue [82] von victorconan

[6] `http://www.quest.dcs.shef.ac.uk/wmt16_files_mmt/training.tar.gz`

[7] `http://dmserv4.cs.illinois.edu/bio_embedding.pk`

[8] Eine E-Mail des Autors an die IT-Abteilung des „Department of Computer Science“ des „Grainger College of Engineering“ der „University of Illinois“ wurde Stand 2022-09-06 nicht beantwortet

wird zumindest das Format aufgeklärt und ein Testlauf mit den originalen GloVe-Embeddings konnte durch geführt werden. [63]

4.6 Veränderung der Software

Die API der Software Bibliotheken hat sich geändert, insbesondere die API von PyTorch und auch das Paket selbst wurde von `pytorch` in `torch` umbenannt.

4.7 Eingeschränkte Hardware

Durch das Fehlen von CUDA-Unterstützung auf der Entwicklungshardware, wurde das Testen bestehender Lösungen erschwert, da bei Python-Programmen, welche nicht entsprechend der aktuellen „Best practices“ in [18, Device-agnostic code], der Quellcode angepasst werden musste, um auf Lenovo T14 lauffähig zu sein.

Auch mit Dell Precision 5510 waren nicht alle Aufgaben ausführbar, da der Grafikspeicher zu beschränkt war, um den `AdamW` Optimierer zu laden. Diese Grenzen der Hardware werden mit `src/check_gpu.ipynb` ermittelt. [24]

4.8 Fehlerhafte Annahmen

Im Rahmen dieser Arbeit wurden an einigen Stellen fehlerhafte Annahmen getroffen, welche den Gesamtfortschritt eingeschränkt haben. Hier ist zu erst die Annahme der Existenz eines bereits annotierten Korpus zu nennen. Daher konnte nicht mit den klassischen Ansätzen zum Training von neuronales Netz gearbeitet werden. Infolgedessen wurden Ansätze zur NER auf Basis von Distant Supervision recherchiert. Die nächste falsche Annahme betraf die Funktionsweise von Distant Supervision; hierbei wurde zunächst angenommen, dass nicht die Eingabe selber annotiert wird, sondern dass die Annotationen in Form von Tags als Eingabe in das Modell eingehen. Bereits in Abschnitt 1.2 war die Annahme, dass diese drei Schritte gemacht werden müssen, nicht zielführend, da das eigentliche Ziel das NEL ist. Ansätze für das direkte Training eines Modells für NEL finden

sich in [16], [67], [12].

5 Fazit und Ausblick

Da dieses Projekt der Test für eine vollständige Implementierung ist, werden hier Verbesserungen für ein künftiges Projekt aufgeführt

5.1 Verbesserung der Datasets-Klasse

Hier werden Verbesserungen vorgestellt, die an der Datasets-Klasse vorgenommen werden können.

Durch die direkte Nutzung der Application-Programming-Interface (API) statt der durch oaipmharvest generierten XML-Dateien könnte der Prozess vereinfacht werden. Der OAI–PMH-Standard bietet die Möglichkeit Treffer nach Datum einzuschränken. Die kontinuierliche Verarbeitung, der seit dem letzen Verarbeitungslauf hinzugekommenen Einträge, würde somit vereinfacht. Auch der zusätzlicher Schritt der Nutzung von oaipmharvest würde wegfallen.

In dem Zuge wäre auch die Umstellung auf Streaming denkbar. Das so erstellte `IterableDataset` lädt neue Daten stückweise, während auf die Einträge zugegriffen wird. Dadurch wird der Speicherbedarf reduziert und der Ladevorgang beschleunigt.

Ein alternativer Ansatze wäre, anstatt die Einträge mit der Entität zu labeln, eine Liste von Labeln zu nutzen, ähnlich zum Ansatz von [51].

Das Stringmatching in Listing 19 könnte stark verbessert werden durch die Verwendung einer besseren Datenstruktur. Eine Struktur die sinnvoll erscheint, wäre der Präfixbaum (auch Trie genannt).

Alternativ würde sich die Verwendung von MLFST (Multi-Layer FST), wie in „Inflection-Tolerant Ontology-Based Named Entity Recognition for Real-Time Applications“ beschrieben, anbieten.[41] Hierfür werden verschiedene FST hintereinander kaskadiert; dieser Ansatz beruht auf „Partial parsing via finite-state cascades“ von Abney [2] mit dem Ziel keine vollständige Liste aller möglichen Kombinationen von Fällen, sondern nur solcher, die auch grammatikalisch vorkommen können, zu erstellen.

5.2 Verbesserung der Textextraktion

Die eigentliche Textextraktion könnte durch Wechsel auf eine Optical Character Recognition-basierende (OCR) Methode verbessert werden. Diese bringt jedoch auch Schwierigkeiten mit sich, wie schon die Studie *Volltext via OCR* aus dem Jahr 2013 von Federbusch und Polzin in [28, S. 6.2] aufzeigt. Hier wurden teilweise weniger als 90% Erkennungsgüte erreicht. Dennoch wäre eine Verbindung von OCR-D mit den semantischen Annotationen vorstellbar. Aber auch eine Verbesserung von PyPDF2 durch die Implementierung der erweiterten Encodings `/B5pc-H` und `/90msp-RKSJ-H` wäre denkbar. Die lösten bei der Entwicklung `NotImplementedError`s aus.

5.3 Verbesserung der Daten

Verbesserung der eigentlichen Daten ist auch eine Möglichkeit bessere Ergebnisse zu erzielen.

Ein Hinzufügen von Kontext zu den einzelnen klassifizierenden Schritten, wie in „FLERT: Document-Level Features for Named Entity Recognition“ beschrieben, ist vorstellbar.[69] Naheliegende Kandidaten wären hier der eigentliche Dokumentenkontext, aber auch eine Kontextualisierung durch die `dc:subject`-Tags wäre vorstellbar.

Es könnte auch ein an ERNIE (Enhanced Language RepresentatioN with Informative Entities) orientiertes Modell trainiert werden, welches das Wissen der verschiedenen KBs enthält. [92] Dafür muss jedoch die Rechenleistung entsprechend vorhanden sein: „We use 8 NVIDIA-2080Ti to pre-train our model and there are 32 instances in each GPU. It takes nearly one day to finish the training (1 epoch is enough).“ ([91]) Damit die mit TransE erzeugten Einbettungen kongruent sind, müssen die Entitäten zusätzlich vereinheitlicht werden oder das Training auf eine KB beschränkt werden. Für die Verknüpfung der verschiedenen KBs könnte, `DeepOnto`[9] ein hilfreiches Werkzeug für die Identifizierung der Entitäten sein, wenn die KBs jeweils bereits als Ontologie vorliegen,

Ebsenso könnten Konzepte der BLL-Ontologie zur Verbesserung der Daten mit der DDC (Dewey Decimal Classification) verknüpft werden. So hat z.B. der Eintrag aus Listing 11 als `dc:subject` den Wert `dcc:497`, also nach [20] „North American native languages“ als Thema.

[9] `https://github.com/KRR-Oxford/DeepOnto`

Da sowohl die BLL-Ontologie als auch ein Teil der `dc:description` bzw. der `dc:title` der OAI–PMH-Records sowohl auf English als auch auf Deutsch vorliegen, könnte ein Ansatz, wie in „Joint Bilingual Name Tagging for Parallel Corpora“ beschrieben, dafür verwendet werden, die Trainingsdaten zu verbessern. [50] Wenn ein Begriff aus der BLL-Ontologie in einer der Sprachen vorkommt, ist die Wahrscheinlichkeit hoch, dass er auch in der anderen Sprache vorkommt. Sollte ein Begiff mit einer Levenshtein-Distanz unter einer gewissen Schwelle zu dem erwarteten Begriff vorkommen, ist dies ein Kanditat für die Erweiterung des Gazetteer. Ein Beispiel ist in Listing 22 zu sehen.

```
<dc:title xml:lang="eng">Intervention effects in questions</dc:title>
<dc:title xml:lang="deu">Interventionseffekte in Fragen</dc:title>
```

Listing 22: Beispiel für die Parallelen zwischen den `dc:title` Einträgen am Beispiel von `<identifier>oai:publikationen.ub.uni-frankfurt.de:6099</identifier>`

Literatur

[1] Education Ecosystem (LEDU). *Regression Versus Classification Machine Learning: What's the Difference?* 11. Aug. 2018. URL: https://medium.com/quick-code/regression-versus-classification-machine-learning-whats-the-difference-345c56dd15f7 (besucht am 07. 09. 2022).

[2] Steven Abney. „Partial parsing via finite-state cascades". In: *Natural Language Engineering* 2.4 (1996), S. 337–344. DOI: 10.1017/S1351324997001599.

[3] Alejandro und community wiki. *Command for argmin or argmax?* URL: https://tex.stackexchange.com/questions/5223/command-for-argmin-or-argmax (besucht am 30. 07. 2022).

[4] *Algorithms*. URL: https://de.overleaf.com/learn/latex/Algorithms.

[5] alpenwasser und user133284. *Pagebreak for minted in figure*. 9. Mai 2017. URL: https://tex.stackexchange.com/questions/368864/pagebreak-for-minted-in-figure (besucht am 12. 09. 2022).

[6] *Annotations*. URL: https://data.linguistik.de/bll/ontology-doc/index.html (besucht am 12. 09. 2022).

[7] Silvia Arano. *Thesauruses and ontologies [en linea]*. 2005. URL: http://eprints.rclis.org/8972/2/12.pdf (besucht am 05. 08. 2022).

[8] Aditya Basu. *Versioning Resume (& LaTeX docs) using Git SHA*. 5. Juli 2019. URL: https://www.adityabasu.me/blog/2019/07/resume-with-git-sha-version/ (besucht am 26. 09. 2020).

[9] David Beckett u. a. *RDF 1.1 Turtle*. Hrsg. von Eric Prud'hommeaux und Gavin Carothers. 25. Feb. 2014. URL: http://www.w3.org/TR/2014/REC-turtle-20140225/ (besucht am 12. 09. 2022).

[10] Antoine Bordes u. a. „Translating Embeddings for Modeling Multi-Relational Data“. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'13. Lake Tahoe, Nevada: Curran Associates Inc., 2013, S. 2787–2795.

[11] Willem Nico Borst. „Construction of Engineering Ontologies for Knowledge Sharing and Reuse“. Undefined. Diss. Netherlands: University of Twente, Sep. 1997. ISBN: 90-365-0988-2. URL: https://research.utwente.nl/en/publications/construction-of-engineering-ontologies-for-knowledge-sharing-and-.

[12] Samuel Broscheit. „Investigating Entity Knowledge in BERT with Simple Neural End-To-End Entity Linking“. In: *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, S. 677–685. DOI: 10.18653/v1/K19-1063. URL: https://aclanthology.org/K19-1063.

[13] Andrew Carlson, Scott Gaffney und Flavian Vasile. „Learning a Named Entity Tagger from Gazetteers with the Partial Perceptron“. In: Jan. 2009, S. 7–13.

[14] Josh Cassidy. *How to Write a Thesis in LaTeX (Part 3): Figures, Subfigures and Tables*. Aug. 2013. URL: https://www.overleaf.com/learn/latex/How_to_Write_a_Thesis_in_LaTeX_(Part_3)%3A_Figures%2C_Subfigures_and_Tables (besucht am 06. 09. 2022).

[15] Branden Chan, Stefan Schweter und Timo Möller. „German's Next Language Model“. In: *CoRR* abs/2010.10906 (2020). arXiv: 2010.10906. URL: https://arxiv.org/abs/2010.10906.

[16] Haotian Chen u. a. „YELM: End-to-End Contextualized Entity Linking“. In: *CoRR* abs/1911.03834 (2019). arXiv: 1911.03834. URL: http://arxiv.org/abs/1911.03834.

[17] Christian Chiarcos u. a. „Lin|gu|is|tik: Building the Linguist's Pathway to Bibliographies, Libraries, Language Resources and Linked Open Data“. In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. European

Language Resources Association (ELRA), Mai 2016, S. 4463–4471. URL: https://aclanthology.org/L16-1707.

[18] *CUDA semantics*. URL: https://pytorch.org/docs/1.12/notes/cuda.html (besucht am 02. 08. 2022).

[19] dani, Schroöder, Martin und user. *How do I check if a document is "oneside" or "twoside"?* 2017. URL: https://tex.stackexchange.com/questions/360785/how-do-i-check-if-a-document-is-oneside-or-twoside (besucht am 11. 08. 2021).

[20] *DDC 23 Summaries*. URL: https://www.oclc.org/content/dam/oclc/dewey/ddc23-summaries.pdf (besucht am 19. 08. 2022).

[21] *Descending into ML: Training and Loss*. URL: https://developers.google.com/machine-learning/crash-course/descending-into-ml/training-and-loss (besucht am 03. 09. 2022).

[22] Jacob Devlin u. a. „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[23] *Distant supervision*. URL: http://deepdive.stanford.edu/distant_supervision (besucht am 06. 09. 2022).

[24] *Efficient Training on a Single GPU*. URL: https://huggingface.co/docs/transformers/perf_train_gpu_one (besucht am 17. 09. 2022).

[25] *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. 17. Feb. 2013. URL: https://www.w3.org/TR/REC-xml/ (besucht am 28. 08. 2021).

[26] *Fachinformationsdienst Linguistik*. URL: https://www.ub.uni-frankfurt.de/projekte/fid-linguistik.html (besucht am 18. 08. 2022).

[27] *Fachinformationsdienste (FID)*. URL: https://www.ub.uni-frankfurt.de/sammlungen/fid.html (besucht am 07. 09. 2022).

[28] Maria Federbusch und Christian Polzin. *Volltext via OCR. Möglichkeiten und Grenzen*. 2013. URL: https://staatsbibliothek-berlin.de/fileadmin/user_upload/zentrale_Seiten/historische_drucke/pdf/SBB_OCR_STUDIE_WEBVERSION_Final.pdf.

[29] Ulrike Fischer und samcarter_is_at_topanswers.xyz. *Make bibitemsep stretchy*. 2. März 2017. URL: https://tex.stackexchange.com/questions/356550/make-bibitemsep-stretchy.

[30] Samuel Flender. *What exactly happens when we fine-tune BERT? A closer look into some of the recent BERTology research*. 21. Feb. 2022. URL: https://towardsdatascience.com/what-exactly-happens-when-we-fine-tune-bert-f5dc32885d76 (besucht am 15. 09. 2022).

[31] P. Földiák. „Forming sparse representations by local anti-Hebbian learning“. In: *Biological Cybernetics* 64.2 (Dez. 1990), S. 165–170. ISSN: 0340-1200, 1432-0770. DOI: 10.1007/bf02331346. URL: https://doi.org/10/dqcdfm.

[32] *German Bert Tokenization produces [UNK] for non alpha-numeric symbols like: "?" "!" "." "@" #60*. URL: https://github.com/deepset-ai/FARM/issues/60 (besucht am 10. 09. 2022).

[33] GitHub. *GitHub: The Largest and Most Advanced Open Source Development Platform - Secure Public & Private Repositories - Code Review - Codespaces - Actions - CI CD - Project Management - DevOps · GitHub*. 2020. URL: https://github.com/ (besucht am 23. 09. 2020).

[34] Thomas R. Gruber. „A translation approach to portable ontology specifications“. In: *Knowledge Acquisition* 5.2 (1993), S. 199–220. ISSN: 1042-8143. DOI:

https://doi.org/11.1006/knac.1993.1008. URL: https://www.sciencedirect.com/science/article/pii/S1042814383710083.

[35] Jens Heinrich. „Analyse der Implementierung von algebraisch-geometrischen Codes“. 2019. URL: https://github.com/s0160187/bachelor_thesis.

[36] Benjamin Hoover, Hendrik Strobelt und Sebastian Gehrmann. „exBERT: A Visual Analysis Tool to Explore Learned Representations in Transformer Models“. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Online: Association for Computational Linguistics, Juli 2020, S. 187–196. URL: https://www.aclweb.org/anthology/2020.acl-demos.22.

[37] MaPePeR und user36296. *qrcode in bibliography margin/bibliography entry with height of marginpar-content1*. 15. Juli 2016. URL: https://tex.stackexchange.com/questions/319524/qrcode-in-bibliography-margin-bibliography-entry-with-height-of-marginpar-conten (besucht am 02. 09. 2019).

[38] Austin Huang u. a. *The Annotated Transformer. Attention is All You Need*. Mit Code annotierte Version von [81]. 2022. URL: http://nlp.seas.harvard.edu/annotated-transformer/.

[39] The IEEE und The Open Group. *The Open Group Base Specifications Issue 6 – IEEE Std 1003.1, 2004 Edition*. New York, NY, USA: IEEE, 2004. URL: http://www.opengroup.org/onlinepubs/009695399/.

[40] Jesse und Michael Osl. *Use algorithm2e inside tcolorbox*. 19. Juli 2014. URL: https://tex.stackexchange.com/questions/192280/use-algorithm2e-inside-tcolorbox (besucht am 12. 09. 2022).

[41] Christian Jilek u. a. „Inflection-Tolerant Ontology-Based Named Entity Recognition for Real-Time Applications“. In: *2nd Conference on Language, Data and Knowledge (LDK 2019)*. Hrsg. von Maria Eskevich u. a. Bd. 70. OpenAccess Series in Informatics (OASIcs).

Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, 11:1–11:14. ISBN: 978-3-95977-105-4. DOI: 10.4230/OASIcs.LDK.2019.11. URL: http://drops.dagstuhl.de/opus/volltexte/2019/10375.

[42] Duff Johnson. *About the Portable Document Format*. 2021. URL: https://www.pdfa.org/about-us/the-portable-document-format/ (besucht am 29. 08. 2021).

[43] D Jurafsky und James H Martin. „Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition“. Version Third Edition draft. In: *Pearson education, Asia* (12. Jan. 2022). URL: https://web.stanford.edu/~jurafsky/slp3/ed3book_jan122022.pdf.

[44] *Kataloge*. URL: https://www.linguistik.de/kataloge/info/ (besucht am 19. 08. 2022).

[45] Jun'ichi Kazama und Kentaro Torisawa. „Inducing Gazetteers for Named Entity Recognition by Large-Scale Clustering of Dependency Relations“. In: *Proceedings of ACL-08: HLT*. Columbus, Ohio: Association for Computational Linguistics, Juni 2008, S. 407–415. URL: https://aclanthology.org/P08-1047.

[46] Kent u. a. *Error message: There's no line here to end*. 11. Okt. 2018. URL: https://tex.stackexchange.com/a/454804 (besucht am 26. 09. 2020).

[47] *Learn How to Draw Trees in TikZ*. 17. Apr. 2021. URL: https://latexdraw.com/draw-trees-in-tikz/.

[48] Elena Leitner. „Eigennamen- und Zitaterkennung in Rechtstexten“. Masterthesis. Potsdam: Universität Potsdam, Feb. 2019. URL: https://raw.githubusercontent.com/elenanereiss/Legal-Entity-Recognition/master/docs/Leitner_LER_BA.pdf.

[49] V.I. Levenshtein. „Binary Codes Capable of Correcting Deletions, Insertions and Reversals“. In: *Soviet Physics Doklady* 10 (Feb. 1966), S. 707.

[50] Qi Li u. a. „Joint Bilingual Name Tagging for Parallel Corpora“. In: *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*. CIKM ’12. Maui, Hawaii, USA: Association for Computing Machinery, 2012, S. 1727–1731. ISBN: 9781450311564. DOI: 10.1145/2396761.2398506. URL: https://doi.org/10/ghhxpp.

[51] Chen Liang u. a. „BOND: BERT-Assisted Open-Domain Named Entity Recognition with Distant Supervision“. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*. ACM, Aug. 2020. DOI: 10.1145/3394486.3403149. URL: https://doi.org/10.1145%5C%2F3394486.3403149.

[52] Universitätsbibliothek Frankfurt am Main. *Über die Universitätsbibliothek*. 2020. URL: https://www.ub.uni-frankfurt.de/ueber/ (besucht am 21. 09. 2020).

[53] Christopher Manning u. a. „The Stanford CoreNLP Natural Language Processing Toolkit“. In: *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Baltimore, Maryland: Association for Computational Linguistics, Juni 2014, S. 55–60. DOI: 10.3115/v1/P14-5010. URL: https://aclanthology.org/P14-5010.

[54] newbedev. *How do I create a LaTeX package? | Newbedev*. 2021. URL: https://newbedev.com/how-do-i-create-a-latex-package (besucht am 12. 08. 2021).

[55] Andrew Ng. *Deep Learning, Self-Taught Learning and Unsupervised Feature Learning*. 27. Okt. 2014. URL: https://www.datascienceassn.org/content/deep-learning-self-taught-learning-and-unsupervised-feature-learning.

[56] Niko und egreg. *How to stack two subfigures next to a third subfigure?* 17. Nov. 2012. URL: https://tex.stackexchange.com/questions/83069/how-to-stack-two-subfigures-next-to-a-third-subfigure (besucht am 14. 09. 2022).

[57] Bruno A Olshausen und David J Field. „Emergence of simple-cell receptive field properties by learning a sparse code for natural images“. In: *Nature* 381.6583 (1996), S. 607–609. URL: http://www.cns.nyu.edu/~tony/vns/readings/olshausen-field-1996.pdf.

[58] Piet van Oostrum. *The fancyhdr and extramarks packages*. version v3.10. Dept. of Computer Science - Utrecht University. 31. Jan. 2019.

[59] Overleaf. *Headers and footers - Overleaf, Online LaTeX Editor*. URL: https://www.overleaf.com/learn/latex/Headers_and_footers (besucht am 19. 03. 2020).

[60] Overleaf. *Writing your own package*. 2021. URL: https://de.overleaf.com/learn/latex/Writing_your_own_package (besucht am 12. 08. 2021).

[61] Adam Paszke u. a. „PyTorch: An Imperative Style, High-Performance Deep Learning Library“. In: *Advances in Neural Information Processing Systems 32*. Hrsg. von H. Wallach u. a. Curran Associates, Inc., 2019, S. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[62] Minlong Peng u. a. „Distantly Supervised Named Entity Recognition using Positive-Unlabeled Learning“. In: *CoRR* abs/1906.01378 (2019). DOI: 10.48550/ARXIV.1906.01378. arXiv: 1906.01378. URL: http://arxiv.org/abs/1906.01378.

[63] Jeffrey Pennington, Richard Socher und Christopher D. Manning. „GloVe: Global Vectors for Word Representation“. In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, S. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

[64] *Polysemy*. URL: https://www.oxfordbibliographies.com/view/document/obo-9780199772810/obo-9780199772810-0259.xml.

[65] *Preprocess*. URL: https://huggingface.co/docs/transformers/preprocessing (besucht am 20. 06. 2022).

[66] Rachel Pries und Beth Malmskog. „Maximal Curves and Application to Coding Theory“. 2008. eprint: https://www.math.colostate.edu/pries/costaricalectures.pdf. URL: https://www.math.colostate.edu/~pries/costaricalectures.pdf.

[67] Amund Faller Råheim. „Joint Entity Linking with BERT“. Magisterarb. Albert-Ludwigs-University Freiburg, 10. Mai 2022. URL: https://ad-publications.cs.uni-freiburg.de/theses/Master_Amund_Faller_Raheim_2022.pdf (besucht am 07. 09. 2022).

[68] Mareike Schumacher. „Named Entity Recognition (NER)“. In: *forTEXT. Literatur digital erforschen*. 2018. URL: https://fortext.net/routinen/methoden/named-entity-recognition-ner (besucht am 05. 08. 2022).

[69] Stefan Schweter und Alan Akbik. „FLERT: Document-Level Features for Named Entity Recognition“. In: *CoRR* abs/2011.06993 (2020). arXiv: 2011.06993. URL: https://arxiv.org/abs/2011.06993.

[70] Jingbo Shang u. a. „Learning Named Entity Tagger using Domain-Specific Dictionary“. In: *CoRR* abs/1809.03599 (2018). arXiv: 1809.03599. URL: http://arxiv.org/abs/1809.03599.

[71] *shell escape - How do you detect restricted write18 support - TeX - LaTeX Stack Exchange*. URL: https://tex.stackexchange.com/questions/88614/how-do-you-detect-restricted-write18-support (besucht am 13. 08. 2021).

[72] Koo Ping Shung. *Accuracy, Precision, Recall or F1?* 15. März 2018. URL: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9 (besucht am 03. 09. 2022).

[73] Stackexchange. *STACKEXCHANGE*. 1. Jan. 1800. URL: https://stackexchange.com (besucht am 22. 09. 2019).

[74] Stackexchange. *STACKEXCHANGE*. 1. Jan. 1800. URL: https://stackexchange.com.

[75] Rudi Studer, V.Richard Benjamins und Dieter Fensel. „Knowledge engineering: Principles and methods“. In: *Data Knowledge Engineering* 25.1 (1998), S. 161–197. ISSN: 0169-023X. DOI: https://doi.org/10.1016/S0169-023X(97)00056-6. URL: https://www.sciencedirect.com/science/article/pii/S0169023X97000566.

[76] *Synonymy*. URL: https://www.oxfordbibliographies.com/view/document/obo-9780199772810/obo-9780199772810-0220.xml.

[77] Ian Tenney, Dipanjan Das und Ellie Pavlick. „BERT Rediscovers the Classical NLP Pipeline“. In: *CoRR* abs/1905.05950 (2019). arXiv: 1905.05950. URL: http://arxiv.org/abs/1905.05950.

[78] Th und abyshukla. *Read a file and store its contents in a variable*. 28. Juni 2017. URL: https://tex.stackexchange.com/questions/377118/read-a-file-and-store-its-contents-in-a-variable (besucht am 26. 09. 2020).

[79] thegeekgreek und user31729. *Default frame around theorems in amsart*. 30. Dez. 2016. URL: https://tex.stackexchange.com/questions/346392/default-frame-around-theorems-in-amsart (besucht am 01. 09. 2019).

[80] *Token classification*. URL: https://huggingface.co/course/chapter7/2 (besucht am 17. 09. 2022).

[81] Ashish Vaswani u. a. „Attention Is All You Need“. In: *CoRR* abs/1706.03762 (2017). URL: https://arxiv.org/pdf/1706.03762.pdf.

[82] victorconan. *bio$_e$mbedding.txtlinknotworking*#44. 10. März 2022. URL: https://github.com/shangjingbo1226/AutoNER/issues/44.

[83] Judy L. Walker. *Codes and Curves*. Bd. 7. Student mathematical library. American Mathematical Society, 17. Mai 1999. ISBN: 978-0-8218-2628-7. DOI: 10.1090/stml/007. URL: http://www.ams.org/bookstore-getitem/item=STML-7.

[84] Wikipedia. *FFM — Wikipedia, die freie Enzyklopädie*. 2021. URL: https://de.wikipedia.org/w/index.php?title=FFM&oldid=209960843 (besucht am 10. 09. 2022).

[85] Wikipedia. *Frankfurt am Main — Wikipedia, die freie Enzyklopädie*. 2022. URL: https://de.wikipedia.org/w/index.php?title=Frankfurt_am_Main&oldid=225740004 (besucht am 10. 09. 2022).

[86] Thomas Wolf u. a. „HuggingFace's Transformers: State-of-the-art Natural Language Processing". In: *CoRR* abs/1910.03771 (2019). arXiv: 1910.03771. URL: http://arxiv.org/abs/1910.03771.

[87] Arne Wolfewicz. *How Do Machines Learn? A Beginners Guide*. URL: https://levity.ai/blog/how-do-machines-learn.

[88] Joseph Wright und Raaja_is_at_topanswers.xyz. *Using stix fonts with pdfLaTeX: "Option clash for package textcomp."* 7. Dez. 2016. URL: https://tex.stackexchange.com/questions/342797/using-stix-fonts-with-pdflatex-option-clash-for-package-textcomp (besucht am 12. 09. 2022).

[89] xuzhao9. *Multi30K dataset link is broken #1756*. 1. Juni 2022. URL: https://github.com/pytorch/text/issues/1756 (besucht am 06. 09. 2022).

[90] Vikas Yadav und Steven Bethard. „A Survey on Recent Advances in Named Entity Recognition from Deep Learning models". In: *CoRR* abs/1910.11470 (2019). arXiv: 1910.11470. URL: http://arxiv.org/abs/1910.11470.

[91] Zhengyan Zhang u. a. *ERNIE*. 4. Juli 2022. URL: https://github.com/thunlp/ERNIE (besucht am 13. 09. 2022).

[92] Zhengyan Zhang u. a. „ERNIE: Enhanced Language Representation with Informative Entities“. In: *CoRR* abs/1905.07129 (2019). arXiv: 1905.07129. URL: http://arxiv.org/abs/1905.07129.

Verzeichnis der Listings

Tabellenverzeichnis

Abbildungsverzeichnis

Liste der Algorithmen

Glossar

AI Artficial Intelligence 11

Alphabet ($\mathcal{A}$) Menge von Zeichen als Grundlage eine Codes oder einer (geschriebenen) Sprache xviii

API Schnittstelle, über welche mit einer Application kommuniziert werden kann. 63, xv

BERT Transformer-basiertes Modell für NLP-Aufgaben [22] I, 5, 8, 15, 29, 30, xv

Beschriftung ($\mathbf{Y}$) 13, 16

BLL Bibliography of Linguistic Literature 4, 21, 22

BLL-Ontologie I, 3, 4, 16, 22, 28, 31, 64, 65

Kreuzentropie-Kosten (l) Differenz zwischen der erwarteten Verteilung $\hat{y}$ und der echten Verteilung y 16

CUDA von Nvidia entwickelte API zum Ausführen von Berechnungen auf der GPU 62

DDC Dewey Decimal Classification 64

DFG „ist die zentrale Selbstverwaltungsorganisation der Wissenschaft in Deutschland“ (dfg:about) 1

Distant Supervision 12, 62, xviii

Dublin Core ™ Element Set standardisiert Menge von Termen zur Beschreibung einer Ressource 24, 36

ERNIE Modell, welches auf mit TransE aus KBs erstellten Einbettungen mittels MLM und NSP trainiert wurde [92] 64

exBERT Visuelles Analysewerkzeug zum Untersuchen gelernter Repräsentationen von Transformer-Modellen[36] 6

FID Fachinformationsdienst 1, 3, xiii

FST Finite-State Transducer 25, 26, 63, xvii

Gazetteer Wörterbuch, welches Entitäten enthält[45] 25, 28, 31, 34, 65, xii

GBERT BERT-basierendes Modell, welches auf Deutsch trainiert wurde [15] 5, 6, 61

GitHub „ GitHub is a development platform inspired by the way you work “ ([33]) 61

GloVe Unsupervised Learning Algorithmus zum Finden von Vektorrepräsentationen von Wörtern 62

html hypertext markup language 36

HTTP generisches zustandsloses objektorientiertes Protokoll auf der Anwendungsebene zur Übertragung von Daten xix

I18N Internationalization 17

ISBN Internationale Standardbuchnummer 1

KB Knowledge Base 3, 4, 12, 16, 20, 25, 28, 64, xv

Konfusionsmatrix Spezialfall der Kontingenztafel (auch Kontingenztabelle oder Kreuztabelle genannt) 9, 58, xiii

Kontingenztafel (auch Kontingenztabelle oder Kreuztabelle genannt) Tabelle, die Häufigkeiten bestimmter Merkmalsausprägungen enthält xvi

Kreuzentropie

Label (y) 13, 16, 58

  ORG 25

Lin|gu|is|tik 34

maschinelles Lernen Trainieren von Maschinen auf Datensätzen I, 8, 11, 15, 16, xix

MD Mention detection 4

Metriken

  F1-Maß Harmonisches Mittel von Precision und Recall 8

  Loss (l) Funktion, die jeder Entscheidung, die vom wahren Parameter abweicht, einen Schaden zuordnet 8, 10–12

  Precision Wahrscheinlichkeit, dass ein positiv identifiziertes Objekt positiv ist 8, xvi

  Recall Wahrscheinlichkeit, dass ein positives Objekt als solches identifiziert wird 8, xvi

MIME Multipurpose Internet Mail Extensions 36

MLFST Verknüpfung kaskadierter FST 63

MLM Aufgabe 15% der Tokens vorherzusagen; diese werden dafür in der Eingabe durch ein Token ersetzt(80% MASK-Token, 10 % zufälliges Token, 10% gleiches Token)[22, 3.1 Task # 1] 29, xv

NLP-Modell (f) 15, 16

Parameter (Θ) Parameter des Models 15, 16

neuronales Netz an die Funktionsweise biologischer neuronaler Netzwerke angelehntes System zur Funktionsapproximation 5, 62, xvii, xix

RNN

LSTM neuronales Netz, welches die Aktivierungsregeln einmal pro Zeitschritt und die Verbindungsgewichte einmal pro Lernzyklus verändert 29

Transformer Architektur eines neuronalen Netzwerks, welches statt Faltung oder Rückkopplung nur Aufmerksamkeitsmechanismen verwendet[81] 5, 14, 15, 29, 61, xv, xix

NLP Verarbeitung natürlicher Sprache (Englisch: Natural Language Processing) 5, 6, 13, 15, 29, xv

NED Named Entity Disambiguation 4, 16

NEL Named Entity Linking 4, 16, 29, 31, 62, xii

NER Named Entity Recognition 4, 13, 15, 16, 24, 62

(B) Label für Token am Anfang einer Entität 13

(I) Label für Token innerhalb einer Entität 13

(O) Label für Token ausserhalb einer Entität 13

NSP Hierbei werden zwei Sätze A und B aus dem Corpus gewählt, sodass in 50% der Fälle Satz B auf Satz A folgt und in 50 % der Fälle B ein zufälliger Satz aus dem Corpus ist. [22, 3.1 Task # 2] 29, xv

OAI Open Archives Initiative 42

OAI–PMH OAI – Protocol for Metadata Harvesting II, 22, 23, 35–38, 40, 41, 53, 63, 65, xii–xiv

oaipmharvest 22, 34, 63

Ontologie 20, 64

OPAC Online Public Access Catalogue 1

OWL eine Sprache die genutzt werden kann, um Klassen und ihre Beziehungen zueinander zu beschreiben 21

Reinforcement Learning 12

Satz (**X**) Liste aus N Tokens $\mathbf{X} = \left[x_1, \ldots, x_N\right]$ 13, 15, 16, xviii

Supervised Learning 5, 11, 12

Tensor Erweiterung des Konzeptes eines Vektors ins mehrdimensionale 53

Token (x) 13, 16, xviii

TransE Modell für Einbettungen, welches darauf beruht, dass eine Entität t, die zu einer Entität h in einer Relation r steht, in der Nähe der Position von $h + v_r$ ist, wobei v_r direkt von r abhängt [10] 64, xv

turtle 17

Universitätsbibliothek Johann Christian Senckenberg „Seit dem 1.1.2005 sind Stadt- und Universitätsbibliothek Frankfurt am Main (StUB) und Senckenbergische Bibliothek (SeB) gemeinsam die zentrale Bibliothek der Universität Frankfurt am Main mit dem neuen Namen »Universitätsbibliothek Johann Christian Senckenberg«.“ ([52]) III, 1–3, 35, 36, xiii, xiv

Unsupervised Learning 5, 12, 58, xvi

`URI` Uniform Resource Identifier 17, 31, xii, xviii, *see also* `URL`

`URL` Spezialfall einer `URI` 35, 36, 44, 52, xiii

XSLT Sprache um XML-Dokumente in andere XML-Dokumente zu überführen xix

AutoNER trainiert Named Entity Tagger mit Distant Supervision basierend auf dem `Tie or Break` Schema [70] 61

Beautiful Soup Python Bibliothek zur Extraktion von Informationen aus Webseiten 42

Code Ein *Alphabet* $\mathcal{A}$ ist eine Menge von Zeichen und ein *Code* von *Länge n* ist eine Teilmenge von $\mathcal{A}^n$ [66] [83, S. 1.1, 1.2,] [35, S. 2.1] xv, xviii, xix

Datasets Bibliothek, welche Zugriff auf den HuggingFace Datasets Hub und effiziente Datenvorverarbeitung bietet II, 35, 50, 53, 63

Levenshtein-Distanz Die Levensthein-Distanz zwischen zwei (Code-)Wörtern, ist die minimale Anzahl von Zeichen, die ersetzt („reversal“ ([49])), gelöscht („deletion“ ([49])), bzw. hinzugefügt („insertion“ ([49])) werden müssen, um eines der Worte in das andere zu überführen.
Dies ist symmetrisch, da eine Ersetzung in beide Richtungen eine Ersetzung erfordert und jede Löschung einem Hinzufügen in die andere Richtung entspricht. Sie beruht auf dem Artikel [49] von Levenshtein, in dem er über eine Klasse von Codes schreibt, die auch solche Störungen korrigieren können. 65

libxml2 in C implementiertes XML Toolkit xix

libxslt in C implementierte XSLT (XSL Transformations) Bibliothek xix

lxml Python interface für libxml2 und libxslt 39, 42

OCR Optical Character Recognition 64

OCR-D Koordinierte Förderinitiative zur Weiterentwicklung von Verfahren für die Optical-Character-Recognition (OCR) 64

pdfminer.six Python-Paket für die Extraktion von Informationen aus PDF 45

Poppler

- **pdftotext** 45

Pygments in Python geschriebene Syntaxhighlighting-Engine 21, xii

PyPDF2 PDF-Bibliothek in reinem Python 45, 64

Python 62, xviii, xix

PyTorch Python Paket, das Tensorberechnungen mit GPU-Unterstützung und Deep neuronale Netze auf Basis von einem Speicherband bietet [61] 61, 62

Requests Python-Bibliothek für HTTP (Hypertext Transfer Protocol) 44, xiii

Transformers Bibliothek für maschinelles Lernen mit Transformer Modellen II, 29, 50, 53

XML „ The Extensible Markup Language (XML) is a subset of SGML that is completely described in this document “ ([25]) II, 22, 34, 36, 40, 41, 52, 63, xii, xviii, xix

Open Access „meint den unbeschränkten und kostenlosen Zugang zu wissenschaftlicher Information“ (openaccess) 3

PDF Dateiformat zum Teilen von Dokumenten unter Beibehaltung der Formatierung und der eingebetteten Bilder[42] 35, 36, 42, 44, 45, 52, 58, xiii, xiv, xix

Regulärer Ausdruck (`regex`) „Reguläre Ausdrücke sind eine kontext-unabhängige Syntax, welche eine greosse Vielfalt an Zeichensätzen und Zeichensatzordnungen repräsentieren kann“ ([39]) 24–26, xii

SGML standard generalized markup language

Hardware

Dell Precision 5510 Model: Precision 5510

CPU: Intel Xeon E3-1505M v5 (8) 3.700GHz

GPU: Intel HD Graphics P530 (256M)

GPU: NVIDIA Quadro M1000M (2048MiB)

MEM: 64161MiB

Lenovo T14 Model: 20S1S06B00 ThinkPad T14 Gen 1

CPU: Intel i5-10210U (8) 4.200GHz

GPU: Intel CometLake-U GT2 [UHD Graphics] (256M)

MEM: 15661MiB