Bachelor Thesis

Named Entity Recognition

Entwicklung einer Maschinen-Lernen-Anwendung zur

Identifikation von Named Entities

Claudius Decker

Gutachter: Prof. Dr. Hannah Bast
Betreuer: Niklas Schnelle

Albert-Ludwigs-Universität Freiburg
Technische Fakultät
Institut für Informatik
Lehrstuhl für Algorithmen und Datenstrukturen

3. November 2017

Bearbeitungszeit

03. 08. 2017 – 03. 11. 2017

Gutachter

Prof. Dr. Hannah Bast

Betreuer

Niklas Schnelle

Erklärung

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

Ort, Datum	Unterschrift

Zusammenfassung

Das Thema dieser Bachelorarbeit ist die Entwicklung einer Anwendung, die Named Entities in natürlich-sprachlichen Texten erkennt und mit ihrem korrespondierenden Eintrag in einer Wissensbasis verbinden kann. Named Entities sind Objekte der realen Welt, wie Personen, Organisationen oder Orte, die mit einem Eigennamen bezeichnet werden. Die Erkennung der Named Entities wird mit zwei Algorithmen des maschinellen Lernens, dem Perzeptron und der logistischen Regression, umgesetzt. Dabei wird versucht, die Resultate durch ein Entity-Linking-Feature zu verbessern. Die beiden Algorithmen werden hinsichtlich ihrer Performanz ausgewertet und in ihrem theoretischen Hintergrund beleuchtet. Dabei zeigt sich, dass das Perzeptron in dieser speziellen Aufgabe der logistischen Regression überlegen ist. Während mit dem Perzeptron ein maximaler F1-Score von über 0,99 erreicht werden konnte, liegt der mit der logistischen Regression erreichte, maximale F1-Score bei 0.82.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Laut einer Studie der IDC aus dem Jahr 2009 wird sich die Menge der digitalen Inhalte im Zeitraum von 2009 bis 2020 mehr als vervierzigfachen. Die Investitionen, um diese Inhalte zu verwalten, werden im gleichen Zeitraum jedoch nur um das eineinhalbfache wachsen [GR10]. In Anbetracht dieser Zahlen, lässt sich erahnen, dass es in Zukunft eine Herausforderung sein wird, effektive Verfahren zur Informationsgewinnung und Strukturierung dieser Datenmengen zu entwickeln und zu optimieren. Für viele Anwendungen, die Informationen in natürlich-sprachlichen Texten auffinden und verarbeiten, bildet Named Entity Recognition die Grundlage, wie beispielsweise Question-Answering, Ontology-Learning und Opinion-Mining.

1.2 Ziele

Unter dem Begriff Named Entity Recognition versteht man die Aufgabe, Eigennamen in natürlich-sprachlichen Texten zu erkennen. Dabei lassen sich zwei Aufgabengebiete unterscheiden - Identifizierung und Klassifizierung von Eigennamen. Das Auffinden von Eigennamen in einem Text wird Named Entity Detection genannt. Die Klassifizierung, bei der die gefundenen Entitäten semantischen Kategorien, wie beispielsweise „Personen“, „Orte“ oder „Organisationen“ zugeordnet werden, wird Named Entity Classification genannt. Diese Arbeit will sich dem ersteren Aufgabengebiet, also dem Finden von Eigennamen widmen. Es soll ein System entwickelt werden, dass einen Text als Eingabe bekommt. Das Programm soll den selben Text mit getagten Entitäten ausgeben. Hierzu wird das IOB-Format verwendet. IOB steht für Inside, Outside, Beginning. *I* bedeutet, dass sich das Wort in einer Entität befindet. *O* heißt, dass das Wort keine Entität ist. *B* steht für den Beginn einer Entität. In *Abbildung 1* wird ein Beispielsatz im IOB-Format getagt.

Für das Identifizieren und Taggen der Eigennamen sollen zwei Algorithmen implementiert und evaluiert werden, das Perzeptron und die logistische Regression. Das

Wimbledon	bad	boy	Jeff	Tarango	caught	a	break	Monday	when	he	advanced	after	the
B	O	O	B	I	O	O	O	O	O	O	O	O	O

retirement	of	German	Alex	Radulescu	due	to	heat	exhaustion	.
O	O	B	I	I	O	O	O	O	O

Abbildung 1: Abbildung 1: Beispiel IOB-Tagging

Perzeptron ist ein Algorithmus, der die Funktion einer Nervenzelle abbilden soll. Die Eingaben werden gewichtet und aufsummiert. Sobald ein bestimmter Schwellwert überschritten wird, feuert das Perzeptron, das heißt, es gibt eine logische 1 aus. Das Verhalten des Perzeptrons wird durch das Lernen von Trainingsbeispielen optimiert. Die logistische Regression ist ein statistisches Verfahren. Dieser Algorithmus bestimmt die Wahrscheinlichkeit, dass die Ausgangsvariable in Abhängigkeit der Eingangsvariablen einen bestimmten Wert annimmt. In unserem Fall kalkuliert sie auf Grundlage bestimmmter Eigenschaften der untersuchten Wörter, die Wahrscheinlichkeit, dass es sich um einen Eigennamen handelt.
In einem zweiten Schritt soll versucht werden, die Resultate der Algorithmen mittels Entity Linking, also dem Nachschlagen der Entitäten in einer Wissensbasis, zu verbessern. Bei den erwähnten Algorithmen handelt es sich um binäre Klassifizier. Das heißt, die Anwendungen können nur zwei Klassen unterscheiden. Da wir in dieser Arbeit jedoch mit den IOB-Tags zwischen drei Klassen differenzieren wollen, behelfen wir uns mit einer Finite State Machine. Mit diesem Prinzip können zwei Perzeptronen beziehungsweise zwei logistische Regressionsanalysen koordiniert werden. Des Weiteren bietet dieses Vorgehen den Vorteil, dass unzulässige Tag-Kombinationen, wie I folgt auf O, ausgeschlossen werden können.

1.3 Aufbau der Arbeit

Im folgenden Kapitel wird ein Überblick über die Forschung im Bereich Named Entity Recognition gegeben. Dabei wird die Entwicklung dieses Forschungsgebietes von seinen Anfängen in den frühen 90ern bis heute skizziert. Hiernach werden die beiden verwendeten Algorithmen vorgestellt und deren Implementation besprochen. Im darauffolgenden Kapitel wird die Funktionsweise einer Finite State Machine erläutert und gezeigt, auf welche Weise sie die Zusammenarbeit zweier Perzeptronen bzw. zweier logistischer Regressionsanalysen koordiniert. Danach wird dargelegt, wie das erarbeitete Programm Methoden des Entity Linking nutzt, um die Leistung der beiden Klassifikatoren zu verbessern und die erkannten Entitäten in einer Wissensbasis aufzufinden. Im Kapitel Evaluation werden Instrumente zur Messung der Leistung der Klassifikatoren vorgestellt und die, mit dem Programm durchgeführten, Experimente besprochen. In den Schlussbetrachtungen werden mögliche, zukünftige Verbesserungen des Programms umrissen.

2 Forschungsstand

2.1 Named Entity Recognition

1991 stellte Lisa F. Rau auf der siebten IEEE Conference on Artificial Intelligence Applications eine Methode vor, um Unternehmensnahmen aus Texten zu extrahieren. Ihr System arbeitete mit Heuristiken und Regeln. Nach dieser ersten bedeutenden Arbeit zur Named Entity Recognition bekam das Thema die nächsten Jahre kaum Aufmerksamkeit. In ihrer Untersuchung zur Entwicklung der Named Entity Recognition zählen David Nadeau und Sathosi Sekine nur eine wissenschaftliche Publikationen im Jahr 1991 und acht Veröffentlichungen im Jahr 1995 zu diesem Forschungsgebiet im englischsprachigen Raum [NS09].
1995 fand die sechste Message Understanding Conference (MUC-6) in Columbia, Maryland statt. Im Rahmen der Konferenz wurden eine Reihe von Wettbewerben durchgeführt, um die Entwicklung neuer und besserer Verfahren der Information Extraction zu fördern. Unter Information Extraction versteht man maschinelle Verfahren zur Verarbeitung von unstrukturierten Informationen. Auf dieser Konferenz wurde der Begriff Named Entity definiert als Bezeichnung für „unique identifiers“ von Entitäten [GS96] und als wichtige Teilaufgabe der Information Extraction bestätigt. Den bestehenden Wettbewerben wurde ein weiterer speziell für Named Entity Recognition hinzugefügt. Ab 1996 bis 2008 boten neben der MUC, die CoNLL (Computational Natural Language Learning) und die ACE-Konferenz (Automatic Content Extraction) mit Wettbewerben einen Rahmen, verschiedene NER-Methoden vorzustellen und zu evaluieren. Aufgabe bei diesen Wettbewerben war es, Elemente aus einem Text zu extrahieren, die zu einer gegebenen Menge von Semantiken passen. Kategorien denen die Entitäten zugeordnet werden sollten, waren beispielsweise „Personen“, „Organisationen“ und „Orte“. Die bei den Konferenzen zur Evaluation der NER-Methoden benutzten Korpora bestanden vorwiegend aus journalistischen Texten.
Bereits am Anfang erreichten die, an den Wettbewerben teilnehmenden, Verfahren eine sehr hohe Genauigkeit beim Erkennen von Named Entities. Schon bei der MUC-6 erreichten von den 20 teilnehmenden Systemen die Hälfte einen F1-Wert von

über 90 Prozent. Der niedrigste Wert, der gemessen wurde, war noch 84,95 Prozent [S95]. Trotz der sehr hohen F1-Werte, die heute erreicht werden können, ist das Problem der Named Entity Recognition noch nicht gelöst. Jedoch haben sich die Forschungsschwerpunkte verschoben. Während es zuvor um das Identifizieren und Klassifizieren von Eigennamen ging, gelten die Hauptanstrengungen nun spezielleren beziehungsweise weiterführenden Themen. Zu nennen sind hier beispielsweise die Reduzierung des Annotationsaufwands der Trainingkorpora durch halb-überwachtes Lernen [TRB10] oder themenunabhängige NER-Methoden [RR09] sowie Versuche mit einer sehr großen Anzahl von Entitätstypen zu arbeiten, wie Changki Lee und Kollegen, die zwischen 147 Entitätstypen unterscheiden [LHYL06].
Neue Konferenzen mit anderen Schwerpunkten entstanden, die die bisherigen Forschungen im Bereich des Named Entity Recognition mit neuen Verfahren des Entity Linking verbanden, wie beispielsweise die INEX Entity Ranking track (XER), die TREC Entity Track (ET) und die TAC Knowledge Base Population task (KBP). Die XER findet seit 2007 statt und hat das Ziel, Entitäten aus Wikipedia nach ihrer Relevanz auf eine Fragestellung zu gewichten. Die ET existiert seit 2009 und versucht Suchmaschinen zu entwickeln, die die Homepages der gesuchten Entitäten finden. Sie arbeitet mit Kategorien für Menschen, Organisationen, Orte und Produkte. KBP sammelt Attribute zu gegebenen Entitäten, wie Alias, Geburtsdatum und Todesursache. Sie ist bisher nur auf Menschen beschränkt.
Frühe NER-Systeme nutzten überwiegend regelbasierte Algorithmen. Zwar werden auch heute noch rein regelbasierte NER-Systeme entwickelt [ESK17], jedoch haben sich Verfahren des Maschinen Lernens mehr und mehr durchgesetzt [NS09]. Hier zu nennen sind beispielsweise Maximum Entropy Markov Models [BDD96], Hidden Markov Modelle [BSW99], Maximum Entropy Markov Models [BDD96], Conditional Random Fields [LMP01] und Support Vector Machines [KM01]. Häufig angewandt werden auch Verfahren, die regelbasierte Methoden und Maschinen Lernen kombinieren.

2.2 Definitionen

Eine verbreitete Defintion von Named Entity Recognition ist die von Alfonseca und Manandhar. Sie begreifen Named Entity Recognition als “the task of classifying unknown objects in known hierarchies that are of interest for us for being useful to solve a particular problem” [AM02]. Die Definition der NER-Task erfolgt hier von einem gegebenen Problem und nicht von den zu klassifizierenden Objekten aus. Dieser

Definition folgen auch die BN hierarchy und die GENIA ontology for Information Retrieval and Question Answering tasks [MUSMG13].
Uneinigkeit besteht darüber, was Named Entities sind und in welche Kategorien sie einzuteilen sind. CoNLL defniert sie folgendermaßen: “Named entities are phrases that contain the names of persons, organizations and locations” [TM03]. Auch die MUC und die ACE betrachten „Personen“, „Organisationen“ und „Orte“ als Named Entities. Jedoch werden bei der MUC und der ACE darüber hinaus auch zeitliche und numerische Ausdrücke zu den Entitäten gezählt. Außerdem werden bei der ACE auch Kategorien zu Typen von Gebäuden, Waffen, Fahrzeugen und geopolitischen Eigennamen, die wiederum in zahlreiche Unterkategorien gegliedert sind, benutzt.
Unser NER-Programm wird durch die unterschiedlichen Definitionen von Named Entities nicht eingeschränkt. Die verwendeten Algorithmen lernen die jeweilige Definition von Named Entities aus den verwendeten Trainingsbeispielen.

3 Features

Bevor Texte von unseren Algorithmen gelernt oder getaggt werden können, müssen sie erst in eine maschinenleserliche Abstraktion überführt werden. In einem ersten Schritt wird der Text hierfür in sogenannte Token unterteilt. Akil Rajdho und Marenglen Biba definieren Token als Instanz einer Sequenz von Zeichen in einem bestimmten Dokument, die, als für die Bearbeitung nutzbare semantische Einheit, gruppiert werden [RB13]. In unserem Fall sind die Token die einzelnen Wörter und Sätze aus denen sich die Trainings- und Testtexte zusammensetzen. Neben der Worte werden auch Satzzeichen wie Punkt, Komma, Semicolon und Klammern als eigenständige Token erfasst. Die genutzten Sätze sind dabei in den Trainingstexten so strukturiert, dass die einzelnen Worte und Sätze durch Leerzeichen beziehungsweise einen Zeilenumbruch angezeigt werden. Nachdem die Texte in Token separiert wurden, werden die Worte in sogenannte Features überführt. Christopher Bishop definiert ein Feature im Kontext des Maschinen Lernens als eine individuell messbare Eigenschaft oder Charakteristik einer beobachteten Erscheinung [B06].
Im Fall dieser Arbeit dienen Feature-Vektoren, um den Text zu abstrahieren. Um diese Features für unsere Algorithmen nutzbar zu machen, wird jedes Wort durch mehrere boolesche Werte repräsentiert und in einem eigenen Vektor gespeichert. Diese Vektoren werden dann als Eingabe für die Algorithmen genutzt. Dieser Vorbereitungsschritt wird feature extraction genannt. In *Tabelle 1* werden alle in unserem Programm verwendeten Features aufgelistet.

Die verwendeten Features lassen sich in drei Kategorien unterteilen. Zum einen Features, die das Wort selbst charakterisieren, wie beispielsweise die ersten oder letzten drei Buchstaben des Wortes. Zum anderen Features, die die Wortumgebung charakterisieren, wie beispielsweise welches Wort vorausgeht oder folgt.
Zu den Features gehören auch die POS-Tags, welche in den Trainingstexten zur Verfügung gestellt werden. Hierunter versteht man die Zuordnung der Token zu Wortarten. Unser Programm kann jedoch auch ohne dieses Feature arbeiten, falls Trainings- oder Testdaten keine POS-Tags enthalten.
Das Feature "Wort in Wissensbasis enthalten"dient dem Entity Linking. Wird das

Tabelle 1: Verwendete Features

Feature	***boolescher Wert***
Wortname lautet „Meier“	True
Erster Buchstabe ist groß	True
Alle Buchstaben sind groß	False
Das Wort besteht aus Zahlen	False
Erste drei Buchstaben des Token sind „Mei“	True
Letzte drei Buchstaben des Token sind „ier“	True
Namen des Tokens davor ist „Herr“	True
Erste drei Buchstaben des Token davor sind „Her“	True
Letzte drei Buchstaben des Token davor sind „err“	True
Namen des Token danach ist „schläft“	True
Erste drei Buchstaben des Token danach sind „sch“	True
Letzte drei Buchstaben des Token danach sind „äft“	True
Wort in Wissensbasis enthalten	True
POS-Tag ist „NNP“	True

jeweilige Wort in der Wissensbasis gefunden, wird dieser Wert *True* gesetzt.
In dem Beispiel ist ersichtlich, wie die Charakteristika der Token in boolesche Werte überführt werden. Statt das Feature an sich zu speichern, wird nur das Vorhandensein des Features mit einer logischen 1 beziehungsweise *True* markiert. Die Feature-Vektoren besitzen für jedes im Trainingstext vorkommende Feature eine eigene Dimension. Die Feature-Vektoren sind dünnbesetzte Vektoren, im Englischen auch *sparse vectors* genannt. Das heißt, sie bestehen fast ausschließlich aus Nullen. Um Speicherplatz zu sparen, werden nur die Stellen an denen eine logische 1 steht, im Feature-Vektor gespeichert. Die Umwandlung eines Wortes in einen solchen Vektor wird in *Abbildung 2* dargestellt. Der Beispielsatz lautet "Herr Meier schläft".

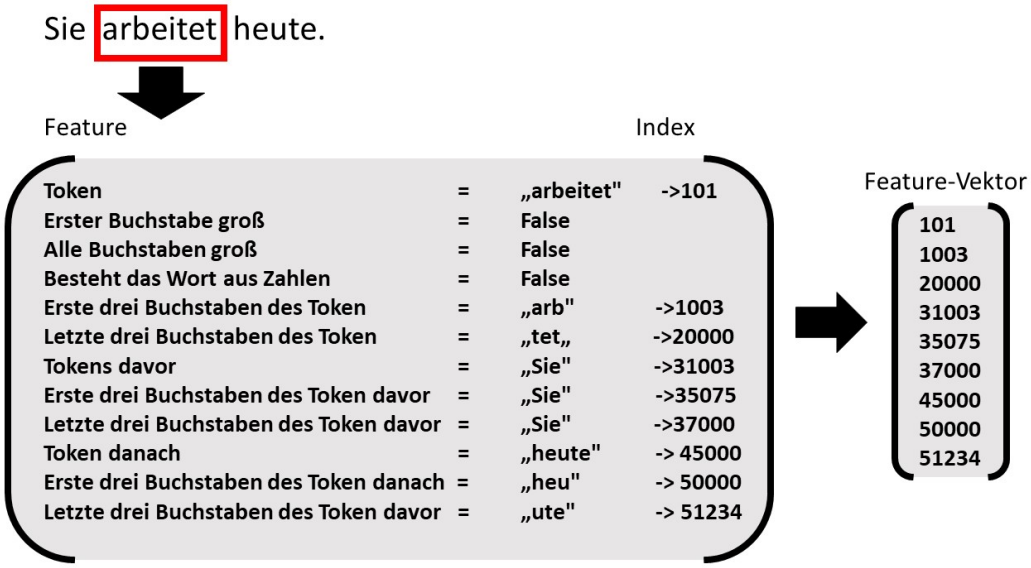

Abbildung 2: Umwandlung in Vektoren

4 Verwendete Algorithmen

4.1 Machine Learning

Tom M. Mitchell definiert ein lernendes Programm folgendermaßen „A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" [Mit97]. Ein lernendes Programm ist demnach ein System, das mit zunehmender Erfahrung eine Aufgabe besser löst.
In diesem Teil der Arbeit werden einige wichtige Grundlagen des maschinellen Lernens in Hinsicht auf die beiden genutzten Algorithmen, Perzeptron und logistische Regression, vorgestellt. In dieser Arbeit wird überwachtes Lernen angewandt, das heißt, die Klassifikation sämtlicher Trainingsdaten ist zu Beginn des Lernprozesses bereits bekannt. Des Weiteren lernen die beiden Algorithmen in dieser Arbeit online. Beim Online Lernen steht ein konstanter Strom von Trainingsdaten bereit. Zu einem Zeitpunkt wird jeweils nur ein Beispiel gelernt. Das zu lernende Modell wird infolge der neu eintreffenden Daten stetig weiter angepasst. Online Lernen eignet sich gut für große Trainingsdaten, da nicht die gesamten Daten gleichzeitig in den Speicher geladen werden müssen. Beide Algorithmen sind binäre Klassifikatoren, dass heißt sie können nur Daten lernen, die linear separierbar sind. Linear separierbare Probleme können durch eine Hyperebene in zwei Klassen separiert werden. Diese Eigenschaft machen sich das Perzeptron und die logistische Regression zunutze und versuchen den Eingangsraum mit einer Ebene in zwei Teile zu teilen. Eine solche Trennebene, wie sie die Algorithmen finden könnten, ist in *Figur 1* dargestellt.

Laut dem Cover-Theorem sind Klassifikationsprobleme mit höherer Wahrscheinlichkeit im höher-dimensionalen Raum lösbar als im nieder-dimensionalen Raum [C65]. Das heißt, dass Klassifikationsprobleme, die nicht linear separierbar sind, oftmals in einem höher-dimensionalen Raum gelöst werden können. In *Abbildung 4* sieht man ein Klassifikationsproblem, dass im 2-dimensionalen Raum nicht linear separierbar ist, jedoch im 3-dimensionalen Raum. Das dargestellte Klassifizierungsproblem ist bei einer Projektion auf die xy-Ebene nicht lösbar. Durch Hinzunahme eines neuen

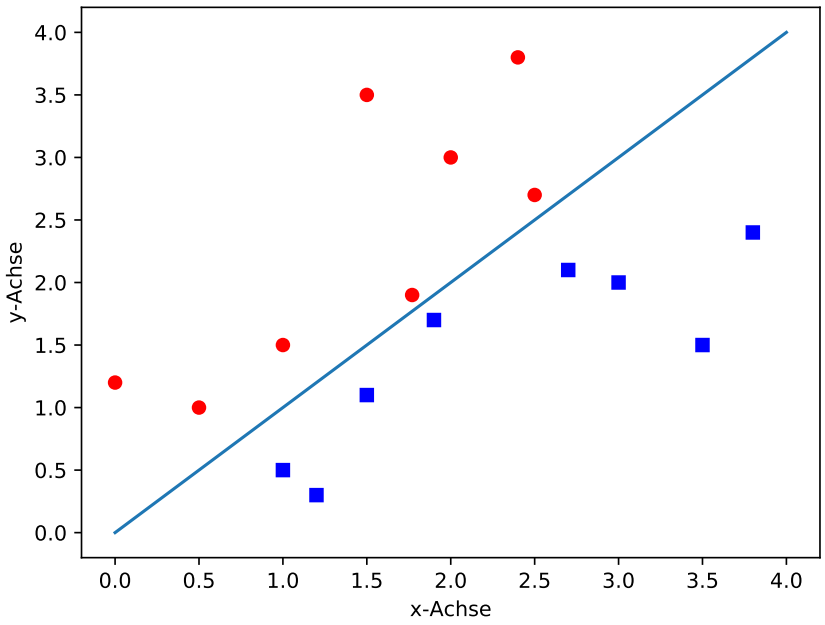

Abbildung 3: lineare Separierbarkeit im 2-dimensionalen Raum

Features $z = -x^2$ wird es lösbar.

Wie in Kapitel 3 gezeigt, wird jedem Feature eine eigene Dimension zugeordnet. Da diese Vektoren, wie ebenda gezeigt, komprimiert werden können, kann eine hohe Anzahl von Features verwendet und damit eine sehr hohe Dimensionalität erreicht werden, womit wir eine sehr hohe Wahrscheinlichkeit erreichen, dass ein Klassifikationsproblem linear separierbar ist und somit von den genutzten Algorithmen gelöst werden kann.

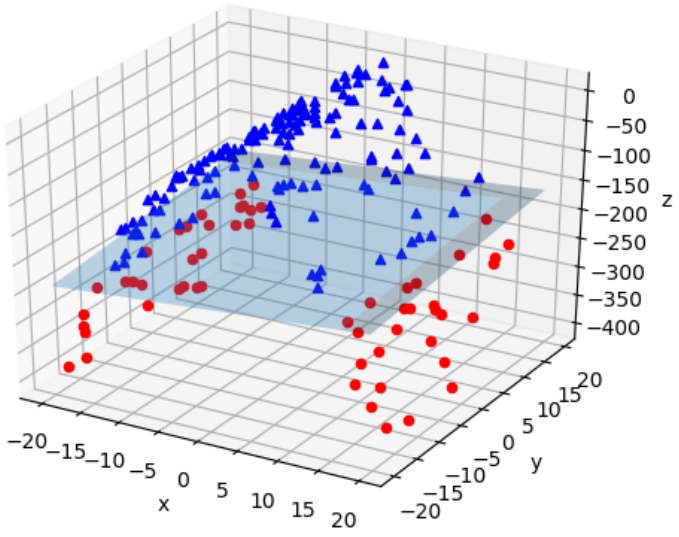

Abbildung 4: lineare Separierbarkeit im 3-dimensionalen Raum

4.2 Perzeptron

Das Perzeptron ist ein vereinfachtes künstliches neuronales Netz. Es wurde von Frank Rosenblatt 1958 im Rahmen der neurologischen Forschung entwickelt. Es wurde als eines der ersten maschinellen Lernverfahren für Klassifikation verwendet und stellt bis heute die Grundlage künstlicher neuronaler Netze dar. Dieser Abschnitt befasst sich mit dem Perzeptron-Algorithmus. Das Perzeptron ist ein Algorithmus, der einen binären Klassifikator lernen kann, das heißt eine Funktion, die einen Inputvektor x auf einen binären Wert abbildet. Somit hat das Ergebnis nur zwei Ausprägungen: beispielsweise eine 1 für "das Beispiel gehört zur Klasse“ oder eine 0 für "das Beispiel gehört nicht zur Klasse". In *Formel 1* wird der Algorithmus dargestellt, mit dem das Perzeptron einen Input mit dem Inputvektor x klassifiziert. Es gibt n Dimensionen. Dabei ist $w \in \mathbb{R}^n$, der Gewichtsvektor. $x \in \{0,1\}^n$. $w \bullet x$ ist das Skalarprodukt $\sum_{i=1}^{n} w_i x_i$. Der Wert b ist ein Bias-Wert. $b \in \mathbb{Z}$ sofern b mit einem Wert $\in \mathbb{Z}$ initialisiert wurde.

$$f(x) = \begin{cases} 1 & \text{wenn } w \bullet x + b > 0 \\ 0 & \text{sonst} \end{cases} \tag{1}$$

Überschreitet die Summe $w \bullet x$ einen bestimmten Schwellenwert, in diesem Beispiel den Wert 0, wird das Trainingsbeispiel als der Klasse zugehörig klassifiziert. Im Lernalgorithmus gilt es einen Gewichtsvektor w zu finden, der den Input so gewichtet, dass bei der Klassifikation der Trainingsdaten möglichst wenige Fehler gemacht werden. Dies geschieht, indem das Perzeptron beim Lernen über alle Trainingsbeispiele iteriert und bei jeder Fehlklassifizierung das Modell entsprechend anpasst. Der Gewichtsvektor w wird nach jedem Trainingsbeispiel aktualisiert. Sei x der Input-Vektor und b ein Bias-Wert. Die Aktualisierung des Gewichtsvektor w im Schritt k verläuft dann wie folgt:

1. $w_{k+1} = w_k$
bei korrekter Ausgabe

2. $w_{k+1} = w_k + x_k$ und $b_{k+1} = b_k - 1$
bei tatsächlicher Ausgabe 0 und korrekter Ausgabe 1

3. $w_{k+1} = w_k - x_k$ und $b_{k+1} = b_k + 1$
bei tatsächlicher Ausgabe 1 und korrekter Ausgabe 0

4.3 Logistische Regression

Bei der Regressionsanalyse wird versucht den Zusammenhang zwischen einer Eingangsgröße x und einer Ausgangsgröße $y = f(x) + \epsilon$ herzustellen. Hierbei ist ϵ ein stochastischer Fehler.

Die einfachste Form ist die sogenannte lineare Regression. Man geht dabei von einem linearen Zusammenhang zwischen x und y aus. Es gibt also Konstanten w, b so dass $y = wx + b + \epsilon$. Sind mehrere Messpunkte $y_1, \ldots, y_n$ zu Eingangsgrößen $x_1, \ldots, x_n$ bekannt, so kann z.B. mit der Methode der kleinsten Quadrate versucht werden, eine möglichst gute Wahl für w und b zu finden.

Die logistische Regression hingegen betrachtet Fälle bei denen y nur Werte zwischen 0 und 1 annehmen kann. Insbesondere ist sie damit prädestiniert für den Fall, dass die Ausgangsgröße eine Wahrscheinlichkeit beschreibt. Sie geht von einem Verhalten $y(x) = S(wx + b)$ aus, wobei $S(t) := \frac{1}{1+e^{-t}}$ die sogenannte Sigmoid-Funktion ist. Dabei kann $y(x)$ als die Wahrscheinlichkeit angesehen werden, dass x ein Erfolg ist. Um gute Werte für w und b zu finden, bietet es sich an, sie mithilfe des Maximum Likelihood Estimators abzuschätzen.

In unserem konkreten Fall, ist $y(x)$ somit die Wahrscheinlichkeit, dass x als $+1$ kategorisiert wird. Der Maximum Likelihood Estimator führt zu folgendem Algorith-

mus:
Sei w der Gewichtsvektor
Sei x der Input-Vektor und b ein Bias-Wert.
α ist ein Lernfaktor

Die Aktualisierung des Gewichtsvektor w im Schritt k verläuft dann wie folgt:

1. $w_{k+1} = w_k + \alpha x_k \cdot (1 - S(w_k \bullet x_k)) \cdot x_k$
bei tatsächlicher Ausgabe +1 und korrekter Ausgabe -1

2. $w_{k+1} = w_k - \alpha x_k \cdot (S(w_k \bullet x_k)) \cdot x_k$
bei tatsächlicher Ausgabe -1 und korrekter Ausgabe +1

Im Gegensatz zum Perzeptron wird der Gewichtsvektor bei der logistischen Regression auch dann verändert, wenn das Trainingsbeispiel richtig klassifiziert wurde.

5 Finite-State-Machine

Die Aufgabe einen gegebenen Text mit IOB-Tags zu versehen, kann von den gewählten Algorithmen nicht ohne Weiteres gelöst werden. Sowohl das Perzeptron als auch die binäre logistische Regression können nur zwischen zwei möglichen Klassen unterscheiden.
Dieses Problem kann gelöst werden, indem man zwei Perzeptronen beziehungsweise logistische Regressionsanalysen implementiert und mittels einer Finite State Machine steuert. Die Finite State Machine ist ein mathematisches Modell einer einfachen idealen Rechenmaschinen. Es ist eine abstrahierte Maschine, die sich zu einem bestimmten Zeitpunkt nur in exakt einem Zustand befinden kann. Die Anzahl der möglichen Zustände ist endlich.
Die Finite State Machine kann von einem Zustand infolge eines externen Inputs in einen anderen übergehen und dabei eine Ausgabe generieren. Der nächste Zustand und die Ausgabe der Finite State Machine ergeben sich aus der Eingabe und dem momentanen Zustand.
Die in dieser Arbeit implementierte Finite State Machine wird in *Abbildung 5* dargestellt. Beim iterieren über die Wörter eines Textes kann sich unser Programm in exakt zwei Zuständen befinden: Entweder es ist in einer Named Enitiy oder nicht.
Jedem der beiden Zustände ist ein Perzeptron beziehungsweise eine logistische Regression mit einem eigenen Gewichtsvektor zugeordnet. Der Startzustand ist der *Outside-State*, was bedeutet, dass sich das Programm aktuell nicht in einer Named Entity befindet. Wird nun ein Wort vom Perzeptron bez. der logistischen Regression nicht als Named Entity klassifiziert, bleibt das System in diesem Zustand und gibt ein O für *outside* aus, womit angezeigt wird, dass es sich beim aktuellen Wort nicht um eine Entität handelt.
Erkennt der Algorithmus hingegen das Wort als Named Entity, wird in den Zustand *Inside-State* gewechselt. In diesem Fall wird ein B für *beginn* ausgegeben und somit angezeigt, dass das Wort der Beginn einer Named Entity ist. Nun befindet sich das System im *Inside-State* und das zweite Perzeptron bez. logistische Regression übernimmt die Klassifikation des nächsten Wortes. Wird das Wort als Named Entity

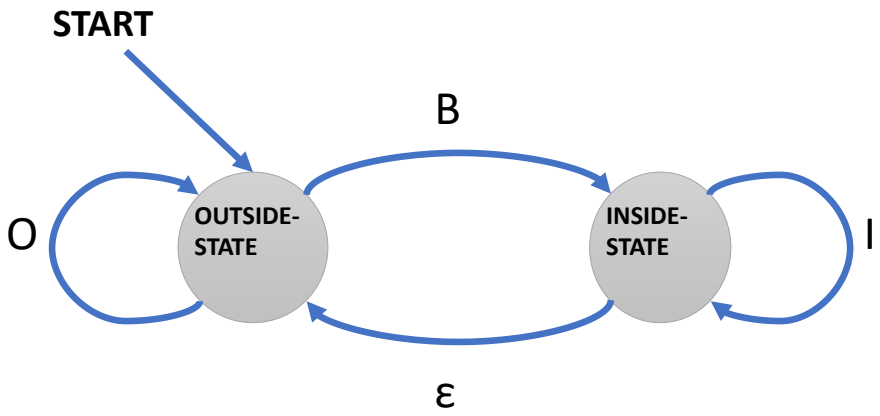

Abbildung 5: Finite State Machine

erkannt, bleibt das System in diesem Zustand und gibt ein I für *inside* aus, was anzeigt, dass das aktuelle Wort Teil einer Named Entity ist.
Wird das Wort jedoch nicht als Named Entity identifiziert, geht das Programm wieder in den *Outside-State* über, wobei es ein O für *outside* als Output generiert und somit anzeigt, dass das aktuelle Wort nicht zu einer Named Entity gehört. Hier übernimmt wieder das erste Perzeptron die Berechnung des nächsten Wortes. Ein Vorteil dieser Finite State Machine ist, dass eine Ausgabe von ungültigen Kombinationen von IOB-Tags ausgeschlossen ist. Die Tag-Folge $O - I$ kann beispielsweise nicht generiert werden, da die Finite State Machine nicht vom *Outside-State* zum *Inside-State* übergehen kann, ohne ein B auszugeben.

6 Entity Linking

Wie Named Entity Recgnition gehört das Entity Linking zu den zentralen Aufgaben der Information Extraction. Beim Entity Linking werden in einem natürlichsprachlichen Text Named Entities identifiziert und mit den korrespondierenden Einträgen einer Wissensbasis verbunden.
In unserem Programm wird der Prozess des *Entity Linking* zweimal unabhängig voneinander durchgeführt. In der Trainingsphase wird jedes Wort in der Wissensbasis gesucht. In dieser Arbeit wird ein Teil von CrossWikis als Wissensbasis genutzt [CrossWiki]. Das Wissen, ob das Wort in der Wissensbasis enthalten ist, wird dem Perzeptron bez. der logistischen Regression für den Klassifikationsprozess in Form eines Features zur Verfügung gestellt.
Ein zweites Entity Linking wird in einer Testphase, die über eine Web-App gestartet werden kann, also beim Klassifizieren bisher unbekannter Texte, durchgeführt. Nachdem der Algorithmus die Worte des gegebenen Textes klassifiziert hat, werden diejenigen Phrasen die als Named Entities identifiziert wurden in der Wissensbasis gesucht. Wird die Named Entity gefunden, wird die Freebase-ID zurückgegeben, welche das Wort einer Seite in der Freebase-Datenbank zuordnet. Diese Suche hat keinen Einfluss auf die Klassifikation des Wortes.
Sehr oft kommt es vor, dass ein Eintrag mehrfach in der Wissensbasis vorkommt, weil es verschiedene Entitäten mit dem gleichen Namen gibt. Mit dem Namen *Washington* kann beispielsweise George Washington, der erste Präsident der vereinigten Staaten, ein Staat oder eine Stadt gemeint sein. In diesem Fall gibt es in der Wissensbasis mehrere gleichlautende Einträge, denen verschiedene Freebase-IDs zugeordnet sind. In der Ausgabe des Programms werden alle IDs aufgeführt.
Die in der Wissensbasis vorkommenden Entitäten wurden in einer Hashtabelle als Schlüssel gespeichert, sodass eine Suche in $\mathcal{O}(1)$ durchgeführt werden kann. Die Hashtabelle befindet sich während des gesamten Programmablaufs im Arbeitsspeicher. Wodurch die Größe der Wissensbasis, die man bei dieser Vorgehensweise nutzen kann, beschränkt ist.

7 Evaluation

7.1 Precision, Recall and F1-Score

Zum Evaluieren der Ergebnisse werden drei Werte ermittelt *Precision*, *Recall* und *F1-Score*, welche sich wie in *Formel* 2, 3 und 4 gezeigt, berechnen. Folgende Fälle können bei der Klassifizierung auftreten:

True Positive Das Wort ist eine *Named Entity* und wurde als *Named Entitiy* klassifiziert

False Positive: Das Wort ist keine *Named Entity* und wurde als *Named Entitiy* klassifiziert

True Negative: Das Wort ist keine *Named Entity* und wurde nicht als *Named Entitiy* klassifiziert

False Negative: Das Wort ist eine *Named Entity* und wurde nicht als *Named Entitiy* klassifiziert

$$precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{2}$$

$$recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{3}$$

$$F1 - Score = 2 \cdot \frac{1}{\frac{1}{recall} + \frac{1}{precision}} = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{4}$$

Der Precision-Wert zeigt für unser Programm das Verhältnis von korrekt erkannten Named Entities zur Anzahl aller klassifizierten Named Entities an.
Der Recall-Wert gibt Auskunft darüber. wie viele der im Text vorhandenen Named

Entities gefunden wurden. Die Werte für Recall, Precision und F1-Score liegen zwischen 0 und 1. Ein Recall von 1 bedeutet, dass alle Named Entities gefunden wurden. Ein Precision von 1 bedeutet, dass alle identifizierten Named Entities tatsächlich Named Entities sind.
Beide Werte haben nur eine bedingte Aussagekraft darüber, wie gut der Klassifikator arbeitet. Beispielsweise würde ein Algorithmus, der einfach alle im Text vorkommenden Wörter als Named Entities klassifiziert, einen optimalen Recall-Wert von 1 erreichen.
Eine Methode, die nur eine Named Entity richtig identifiziert und alle anderen nicht erkennt, würde einen optimalen Precision-Wert von 1 erreichen. Deshalb wurde der F-Score eingeführt, der den ungewichteten harmonischer Mittelwert zwischen Precision und Recall berechnet.

7.2 Experimente

Als Trainingstexte wurden zwei Texte verwendet. Die Datei eng.train stammt aus der CoNLL-2003 Shared Task [ConLL]. Die Datei train wird vom Data-Science-Portal Kaggle zur Verfügung gestellt [train]. Beide Texte wurden in einer gekürzte Version verwendet.
Für die Evaluation wurden 90 Prozent der Texte zum Training benutzt. Die übrigen 10 Prozent wurden zum Testen verwendet. Es wurden beide Texte mit der logistischen Regression und dem Perzeptron gelernt und im Anschluss getestet. Dabei wurde jeweils einmal mit und einmal ohne Entity Linking gearbeitet. Hier findet nur der F1-Score nähere Betrachtung. Alle gemessenen Werte befinden sich im Anhang. Mit dem Perzeptron-Algorithmus konnte ein sehr hoher F1-Score von bis zu über 0,99 erreicht werden. Die logistische Regression schnitt mit einem F1-Score von bis knapp über 0,82 wesentlich schlechter ab. Ein Lernfortschritt über die Epochen ist nicht festzustellen.
Der F1-Score konnte durch Entity Linking nicht verbessert werden. Wie man in den *Abbildungen 1-7* sieht sind die F1-Werte mit Enity Linking und ohne oftmals deckungsgleich. Dies gilt für beide Algorithmen auf beiden Texten.
Dies liegt daran, dass die verwendete Wissensbasis sehr umfangreich ist. Deshalb werden viele Wörter gefunden, die nur sehr selten die Bedeutung einer Named Entity annehmen. Im Test zum Kaggle-Text wurden von den vorkommenden Wörtern 2271 in der Wissensbasis gefunden, während 321 der Wörter nicht gefunden werden konnten. Beim CoNLL-Text wurden 755 Wörter in der Wissensbasis gefunden und

nur 144 nicht. Somit erhöht das Vorhandensein eines Wortes in der Wissensbasis die Wahrscheinlichkeit kaum, dass es sich dabei um eine Named Entity handelt.

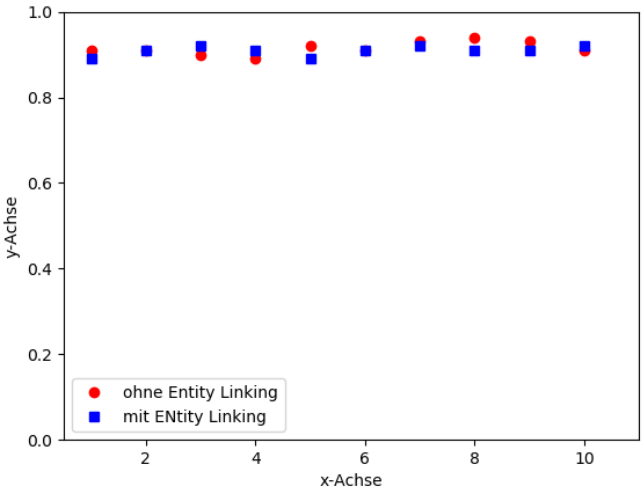

Abbildung 6: Perceptron Kaggle-Text

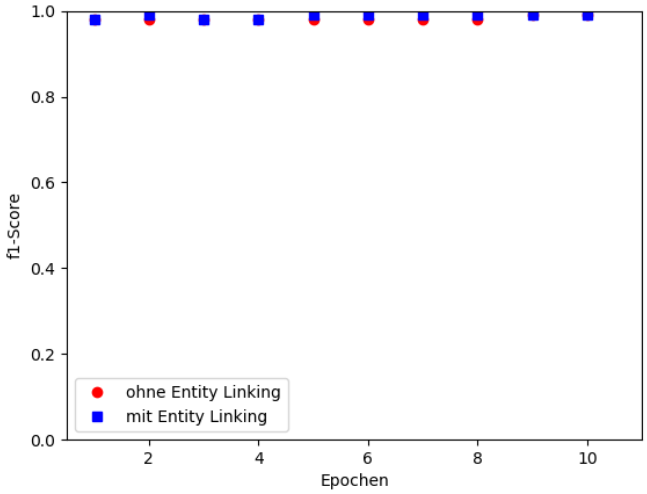

Abbildung 7: Perceptron CoNLL-Text

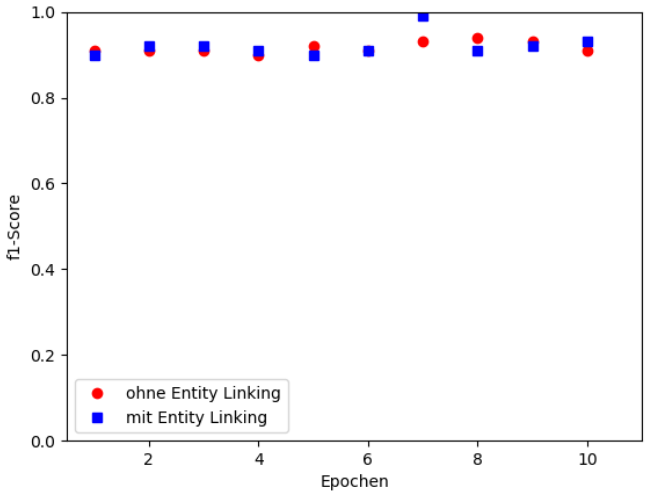

Abbildung 8: logistische Regression Kaggle-Text

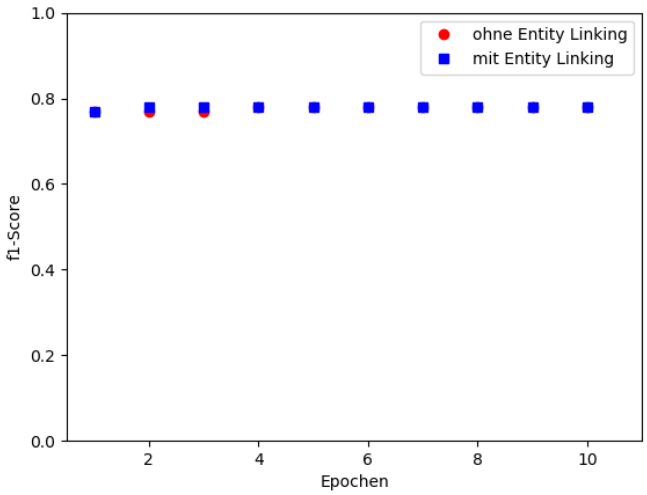

Abbildung 9: logistische Regression CoNLL-Text

8 Schlussbetrachtungen und zukünftige Arbeit

Im Rahmen dieser Bachelorarbeit wurden erfolgreich zwei unterschiedliche Algorithmen zur Klassifikation implementiert und evaluiert. Darüber hinaus wurde getestet, ob die Ergebnisse durch das hinzunehmen von Entity Linking verbessert werden können.
Des Weiteren wurde eine Web-App entwickelt, die in eingegebenen Sätzen Named Entities erkennt, in einer Wissensbasis sucht und im gegebenen Fall den gefunden Eintrag ausgibt.
Es stellt sich heraus, dass der Perzeptron-Algorithmus bei unserer Aufgabe wesentlich bessere Ergebnisse erreicht als die logistische Regression.
Durch die Hinzunahme von Named-Entity-Verfahren konnte keine signifikante Verbesserung erreicht werden. Dies könnte daran liegen, dass die verwendete Wissensbasis sehr umfangreich ist und ein Großteil, der in den Texten vorkommenden Worte, dort als Named Entities gelistet werden. Die meisten dieser Worte haben aber im normalen Gebrauch nur selten die Bedeutung dieser Named Entities.
Ein Mögliche Verbesserung könnte eventuell mit einer kleineren Wissensbasis erreicht werden, in der nur diejenigen Named Entities geführt werden, die ausschließlich als Named Entities vorkommen. Eine weitere Möglichkeit, wäre die Wahrscheinlichkeit, mit der ein Wort als Named Entity vorkommt, mit in die Wissensbasis aufzunehmen und in den Entscheidungsprozess der Algorithmen mit einfließen zu lassen.

9 Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich während der Anfertigung dieser Bachelorarbeit unterstützt haben.

Zuerst gebührt mein Dank Frau Prof. Hannah Bast, die meine Bachelorarbeit betreut und begutachtet hat. Des Weiteren möchte ich mich herzlich bei Herrn Niklas Schnelle bedanken für die hilfreichen Anregungen und die konstruktive Kritik.

Literaturverzeichnis

[GR10] Gantz, John u. Reinsel, David: Digital Universe Decade, Are You Ready? IDC White Paper. 2010 https://www.emc.com/collateral/analyst-reports/idc-digital-universe-are-you ready.pdf (30.10.2017)

[NS09] Nadeau, David u. Sekine, Satoshi: survey of named entity recognition and classificationin. Named Entities. Recognition, classification and use. Amsterdam, Philadelphia 2009

[GS96] Grishman, R. u. Sundheim, B.: Message Understanding Conference. 6: A Brief History. In: COLING.[1996 S. 466-471.

[MUSMG13] Marrero, Mónica; Urbano, Julián; Sánchez-Cuadrado, Sonia; Morato, Jorge u. Gómez-Berbís, M. Juan: Named Entity Recognition. Fallacies, Challenges and Opportunities. In: Journal of Computer Standards u. Interfaces (35:5). 2013. S. 482–489.

[S95] S95] Sundheim, M. Beth: Overview of results of the MUC-6 evaluation. In: MUC6 95 Proceedings of the 6th conference on Message understanding. 1995. S. 13-31.

[TRB10] Turian, Joseph; Ratinov, Lev; Bengio, Yoshua: Word representations: A simple and general method for semi-supervised learning. In: Proc. Association for Computational Linguistics, 2010. S. 384–394.

[RR09] Ratinov, Lev u Roth, Dan: Design challenges and misconceptions in named entity recognition. In: Conference on Computational Natural Language Learning (CoNLL). Association for Computational Linguistics, 2009. S. 147–155.

[LHYL06] Lee, Changki; Hwang, Yi-Gyu; Oh, Hyo-Jung; Jeong Heo, Soojong Lim; Lee, Chung-Hee; Kim, Hyeon-Jin; Wangm Ji-Hyun Wang u.

Jang, Myung-Gil: Fine-grained named entity recognition using conditional random fields for question answering. In: Information Retrieval Technology (4182). 2006. S. 581–587.

[ESK17] Eftimov, Tome; Seljak, Barbara; Korošec, Peter: A rule-based named-entity recognition method for knowledge extraction of evidence-based dietary recommendations. PLoS ONE12(6): e0179488. https://doi.org/10.1371/journal.pone.0179488 (28.10.2017)

[BSW99] Bikel, M. Daniel; Schwartz, Richard; Weischedel, M. Ralph: An Algorithm that Learns What's in a Name. In: Machine Learning. (34, 1-3) 1999 S. 211–231. http://dx.doi.org/http://dx.doi.org/10.1023/A:1007558221122 (28.10.2017)

[BDD96] Berger L. Adam; Della Pietra, A. Stephen; Della Pietra; J. Vicent: A Maximum Entropy Approach to Natural Language Processing. In: Computional Linguistics (22) 1996, S. 39–71.

[LMP01] Lafferty, John; McCallum, Andrew; Pereira, Fernando: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the Eighteenth International Conference on Machine Learning. 2001. S. 282–289.

[KM01] Kudo, Taku; Matsumoto, Yuji: Chunking with support vector machines. In: NAACL 01: Second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies 2001. Morristown, NJ, USA : Association for Computational Linguistics, 2001, S. 1–8.

[AM02] Alfonseca, Enrique u. Manandhar, Suresh "An unsupervised method for general named entity recognition and automated concept discovery," In: 1st International Conference on General WordNet, 2002.

[TM03] Tjong Kim Sang, F. Erik u. De Meulder, Fien: Introduction To The CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. International Conference on Computational Natural Language Learning. 2003.

[MUSMG13] Marrero, Mónica; Urbano, Julián, Sánchez-Cuadrado, Sonia; Morato, Jorge; Gómez-Berbís, M. Juan: Entity Recognition. Fallacies,

Challenges and Opportunities. In: Journal of Computer Standards u. Interfaces (35:5). 2013. S. 482–489.

[B06] Bishop, Christopher: Pattern recognition and machine learning. 2006

[RB13] Rajdho, Akil, Biba, Marenglen: Plugging Text Processing and Mining in a Cloud Computing Framework. In: Internet of Things and Intercooperative Computational Technologies for Collective Intelligence. 2013 S. 369–390.

[M97] Mitchell, M. Tom: Machine Learning. McGraw-Hill, New York, 1997.

[C65] Cover, M. Thomas: Geometrical and Statistical properties of systems of linear inequalities with applications in pattern recognition. IEEE Transactions on Electronic Computers. EC-14. 1965. S. 326–334.

Trainingsdaten:

[CoNLL] CoNLL 2003 eng.train http://github.com/synalp/NER/blob/master/corpus/CoNLL-2003/eng.train (30.10.2017)

[train] train https://www.kaggle.com/velavok/nercorpus/data (30.10.2017)

Wissensbasis:

[CrossWiki] V. I. Spitkovsky, I. Valentin u. Chang X. Angel. A Cross-Lingual Dictionary for English Wikipedia Concepts. In: LREC. 2012 S. 3168–3175.

10 Messergebnisse

Tabelle 2: Perceptron Text Kaggle ohne Entity Linking

Epoche	***Precision***	***Recall***	***F1-score***
1	0.9858757062146892	0.8470873786407767	0.9112271540469974
2	0.9633802816901409	0.8636363636363636	0.9107856191744341
3	0.9515669515669516	0.8675324675324675	0.9076086956521738
4	0.8863636363636364	0.9043478260869565	0.8952654232424677
5	0.9915254237288136	0.8645320197044335	0.9236842105263158
6	0.9631728045325779	0.8651399491094147	0.9115281501340482
7	0.9943661971830986	0.8781094527363185	0.9326287978863937
8	0.9943661971830986	0.8825000000000000	0.9350993377483444
9	0.9943661971830986	0.8694581280788177	0.9277266754270697
10	0.9943342776203966	0.8357142857142857	0.908150064683053

Tabelle 3: Perceptron Text Kaggle mit Entity Linking

Epoche	***Precision***	***Recall***	***F1-score***
1	0.9567723342939481	0.8341708542713567	0.8912751677852349
2	0.9830985915492958	0.8574938574938575	0.916010498687664
3	0.9971830985915493	0.8613138686131386	0.9242819843342036
4	0.9463276836158192	0.881578947368421	0.9128065395095367
5	0.8806818181818182	0.9064327485380117	0.893371757925072
6	0.9971830985915493	0.8509615384615384	0.9182879377431907
7	0.9971830985915493	0.855072463768116	0.9206762028608583
8	0.9463276836158192	0.8769633507853403	0.9103260869565217
9	0.9943502824858758	0.8585365853658536	0.9214659685863874
10	0.9886685552407932	0.8768844221105527	0.929427430093209

Tabelle 4: Perceptron Text CoNLL03 ohne Entity Linking

Epoche	***Precision***	***Recall***	***F1-score***
1	0.9921052631578947	0.9641943734015346	0.9779507133592736
2	0.9973753280839895	0.9669211195928753	0.9819121447028423
3	1.0000000000000000	0.9547738693467337	0.9768637532133677
4	1.0000000000000000	0.9547738693467337	0.9768637532133677
5	1.0000000000000000	0.9547738693467337	0.9768637532133677
6	1.0000000000000000	0.9523809523809523	0.975609756097561
7	1.0000000000000000	0.9523809523809523	0.975609756097561
8	1.0000000000000000	0.9523809523809523	0.975609756097561
9	1.0000000000000000	0.9744245524296675	0.9870466321243523
10	0.994750656167979	0.9768041237113402	0.9856957087126138

Tabelle 5: Perceptron Text CoNLL03 mit Entity Linking

Epoche	***Precision***	***Recall***	***F1-score***
1	0.9973684210526316	0.989501312335958	0.9780645161290323
2	0.9921052631578947	0.8574938574938575	0.9908015768725361
3	0.9921259842519685	0.9593908629441624	0.975483870967742
4	0.9947368421052631	0.9717223650385605	0.9830949284785436
5	1.0000000000000000	0.9718670076726342	0.9857328145265889
6	1.0000000000000000	0.9743589743589743	0.9870129870129869
7	1.0000000000000000	0.9743589743589743	0.9870129870129869
8	1.0000000000000000	0.9743589743589743	0.9870129870129869
9	1.0000000000000000	0.9743589743589743	0.9870129870129869
10	1.0000000000000000	0.9794344473007712	0.9896103896103896

Tabelle 6: logistische Regression Text CoNLL03 ohne Entity Linking

Epoche	***Precision***	***Recall***	***F1-score***
1	0.6353887399463807	0.9753086419753086	0.7694805194805194
2	0.6390374331550802	0.9795081967213115	0.7734627831715211
3	0.6417112299465241	0.9523809523809523	0.7667731629392971
4	0.6417112299465241	0.9836065573770492	0.7766990291262135
5	0.6417112299465241	0.9836065573770492	0.7766990291262135
6	0.6417112299465241	0.9836065573770492	0.7766990291262135
7	0.6417112299465241	0.9836065573770492	0.7766990291262135
8	0.6417112299465241	0.9836065573770492	0.7766990291262135
9	0.6417112299465241	0.9836065573770492	0.7766990291262135
10	0.6417112299465241	0.9836065573770492	0.7766990291262135

Tabelle 7: logistische Regression Text CoNLL03 mit Entity Linking

Epoche	***Precision***	***Recall***	***F1-score***
1	0.6327077747989276	0.989501312335958	0.7674796747967481
2	0.6390374331550802	0.9835390946502057	0.7747163695299838
3	0.6390374331550802	0.9835390946502057	0.7747163695299838
4	0.6417112299465241	0.9836065573770492	0.7766990291262135
5	0.6417112299465241	0.9836065573770492	0.7766990291262135
6	0.6417112299465241	0.9836065573770492	0.7766990291262135
7	0.6417112299465241	0.9836065573770492	0.7766990291262135
8	0.6417112299465241	0.9836065573770492	0.7766990291262135
9	0.6417112299465241	0.9836065573770492	0.7766990291262135
10	0.6417112299465241	0.9836065573770492	0.7766990291262135

Tabelle 8: logistische Regression Text Kaggle ohne Entity Linking

Epoche	***Precision***	***Recall***	***F1-score***
1	0.6753623188405797	0.8534798534798534	0.7540453074433657
2	0.7327586206896551	0.8732876712328768	0.7968750000000000
3	0.7335243553008596	0.8951048951048951	0.8062992125984252
4	0.7306590257879656	0.8793103448275862	0.7981220657276995
5	0.7507163323782235	0.891156462585034	0.8149300155520995
6	0.7564469914040115	0.8918918918918919	0.8186046511627907
7	0.7593123209169055	0.8952702702702703	0.8217054263565892
8	0.7593123209169055	0.8952702702702703	0.8217054263565892
9	0.7593123209169055	0.8952702702702703	0.8217054263565892
10	0.7593123209169055	0.8983050847457628	0.8229813664596274

Tabelle 9: logistische Regression Text Kaggle mit Entity Linking

Epoche	***Precision***	***Recall***	***F1-score***
1	0.7159420289855073	0.872791519434629	0.7866242038216561
2	0.7463976945244957	0.8720538720538721	0.8043478260869567
3	0.7191977077363897	0.8964285714285715	0.7980922098569158
4	0.7335243553008596	0.89198606271777	0.8050314465408804
5	0.7421203438395415	0.8931034482758621	0.8106416275430359
6	0.7449856733524355	0.8934707903780069	0.8125000000000000
7	0.7449856733524355	0.896551724137931	0.8137715179968701
8	0.7478510028653295	0.900000000000000	0.8169014084507041
9	0.7507163323782235	0.9003436426116839	0.818750000000000
10	0.7507163323782235	0.9003436426116839	0.818750000000000

Abbildungsverzeichnis

Tabellenverzeichnis