Universität Paderborn

Masterarbeit

Named Entity Extraction auf Archivdaten

von
Daniel Vollmers

vorgelegt bei
Prof. Dr. Axel-Cyrille Ngonga Ngomo
betreut von
Dr. Ricardo Usbeck

24. August 2018

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Ausführungen, die fremden Quellen wörtlich oder sinngemäß entnommen wurden, sind kenntlich gemacht. Die Arbeit war in gleicher oder ähnlicher Form noch nicht Bestandteil einer Studien- oder Prüfungsleistung.

Paderborn, 24. August 2018

_____

Daniel Vollmers

Abstract - deutsch

Ziel dieser Arbeit ist es Entitäten in einer großen Menge von Archivdaten zu finden und zu den Wissensbasen Wikidata und der Gemeinsamen Normdatei der deutschen Nationalbibliothek zu verlinken. Dafür wird zunächst ein Named Entity Recognition (NER) Ansatz basierend auf einem Conditional Random Field entwickelt und anschließend das Linking mit einer Erweiterung von AGDISTIS[1] durchgeführt. Die Erweiterung von AGDISTIS umfasst ein neues Distanzmaß für den Vergleich zweier Entitäten, neue Entity-spezifische Features, sowie das gleichzeitige Linking von Entitäten durch zwei Wissensbasen. Abschließend wird das neue Verfahren auf einem neu erstellten Goldstandard evaluiert

Stichworte: Named Entity Recognition, Entity Linking, Semantic Web, Archivdaten,

Abstract - english

The goal of this thesis is to find entities in a large set of archival data and to link those entities to the knowledge bases GND and Wikidata. For the task of Named Entity Recognition (NER) a Conditional Randomn Field is used and the linking is implemented with a extension of AGDISTIS. The extension of AGDISTIS contains a new measure for calculating the distance of two entities and new entity type specific features. On top of that a new technique was developed to link entities over two knowledgebases at the same time. Finally the framework was evaluated with a newly generated gold standard for archival data.

Keywords: Named Entity Recognition, Entity Linking, Semantic Web, archival data

1 http://aksw.org/Projects/AGDISTIS.html (abgerufen am 23.8.2018)

Inhaltsverzeichnis

1. Einleitung

Das Konzept des Sematic Webs hat das Ziel Informationen im Web maschinenlesbar zur Verfügung zu stellen, damit sie einfacher von Applikationen verarbeitet werden können. Anwendern bietet dies weitreichende Möglichkeiten zur Suche und Filterung einer großen Menge von Daten. Bislang sind viele Informationen im Web nur unstrukturiert enthalten. Deshalb ist eine Vielzahl von NLP Anwendungen notwendig, um diese Informationen zu strukturieren.
Im Bereich des Webs gibt es bislang schon viele Ansätze zur Verarbeitung von Webseiten. Weit erforscht sind hier beispielsweise Wikipedia Einträge, die im Rahmen des DBpedia Projekts automatisch verarbeitet und in einer Wissensbasis frei zur Verfügung gestellt werden.
Neben der Betrachtung von Webseiten ist es aber ebenso erstrebenswert andere unstrukturierte Datenquellen so zu strukturieren, dass sie leicht von einem Rechner verarbeitet werden können. Ein Beispiel hierfür sind Nachrichtenartikel.
Ein weitgehend unerforschter Bereich ist das Verarbeiten von Archivdaten. Archivdaten reichen oft über einen langen Zeitraum zurück und liegen oft teilweise strukturiert in tabellarischer Form vor. Diese Daten enthalten eine Große Menge an Entitäten, die allerdings nur als reiner Text vorliegen ohne zusätzliche standardisierte Annotationen.
Das Ziel dieser Arbeit, welche Teil des DIESEL-Projekts[1] ist, ist es nun diese Entitäten aus den Archivdaten zu extrahieren und mit Hilfe von bereits existierenden Wissensbasen zu verlinken. Dafür liegen eine große Menge von circa 60.000 Datensätzen eines Archivs mit Daten über Theater-, Opern- und anderer Aufführungen vor.
Ziel ist es somit für folgende Probleme neue Ansätze zu entwickeln

- Named Entity Recognition (NER) auf Archivdaten
- Entity Linking (EL) auf Archivdaten
- Entity Linking durch die Wissensbasen Wikidata und GND
- Entity Linking über mehrere Wissensbasen
- Generierung eines Goldstandards für Archivdaten

1 http://aksw.org/Projects/DIESEL.html (abgerufen am 23.8.2018)

Zur Identifizierung und Verlinkung der Archivdaten kann auf bereits bekannte Konzepte für die Named Entity Recognition (NER) und das Entity Linking (EL) zurückgegriffen werden. Die Anwendung von NER und EL auf Archivdaten ist dabei noch ein unerforschtes Gebiet, für das bislang keine Ansätze entwickelt wurden. Deshalb soll in dieser Arbeit auch erforscht werden, in wie weit bisherige Konzepte der NER und des ELs auf Archivdaten übertragbar sind.
Die im Bereich des Entity Linkings am häufigsten genutzte Ressource ist Wikipedia. Neben Wikpipedia wurden auch die Wissensbasen DBpedia häufig in Entity Linking Ansätzen verwendet. Archivdaten enthalten jedoch häufig viele weitgehend unbekannte Entitäten, die noch nicht in DBpedia enthalten sind und für die noch keine Wikipediaseite angelegt wurden. Ziel ist es deshalb, für das Entity Linking die bisher kaum verwendete Gemeinsame Normdatei der Deutschen Nationalbibliothek und Wikidata als Wissensbasen zu nutzen, da hier die Abdeckung der in den Archivdaten enthaltenen Entitäten deutlich höher ist. Ebenso soll ein Verfahren entwickelt werden, welches Entitäten gleichzeitig zur GND und zu Wikidata verlinkt.
Abschließend soll eine Evaluation der neu entwickelten Funktionen durchgeführt werden. Aus diesem Grund ist es notwendig aus einem Teil der Archivdaten einen Goldstandard zu erstellen, auf dem der neue Ansatz getestet werden kann. Ziel ist es daher, zunächst Entitäten mit Hilfe eines NER-Verfahrens zu identifizieren und anschließend durch ein Linking-Verfahren passende Ressourcen aus Wissensbasen zuzuordnen.
Linking-Verfahren verwenden dabei in der Regel nur eine Wissensbasis, zu der Entitäten verlinkt werden. Ziel dieser Arbeit ist es deshalb, auch ein Verfahren zu entwickeln, welches Entitäten gleichzeitig zu mehreren Wissensbasen verlinkt.

2. Grundlagen

In diesem Kapitel sollen die wesentlichen Grundlagen für diese Masterarbeit vorgestellt werden. Dies ist zunächst das Semantic Web, da es das Ziel ist Entitäten aus Archivdaten maschinenlesbar mit Webressourcen zu verlinken. Das Extrahieren dieser Daten erfordert einen Named Entity Recognition Ansatz, zur Identifikation von Entitäten. Hierfür wird das sequenzielle Modell CRF (Kapitel 2.3.3) verwendet. Dazu werden zunächst die Modelle HMM (Kapitel 2.3.1) und MEMM (Kapitel 2.3.2) eingeführt, da diese die Grundlage für CRFs bilden und darauf aufbauend CRFs beschrieben. Als letztes sollen die Daten verlinkt werden. Das Linkingverfahren nutzt dafür den HITS-Algorithmus (Kapitel 2.4), der abschließend ebenfalls kurz beschrieben wird.

2.1. Semantic Web

Das Ziel dieser Arbeit ist es Archivdaten zu Webressourcen in Wissensbasen zu verlinken, um damit die maschinelle Verarbeitung dieser Daten zu ermöglichen. Die grundlegende Technologie zur maschinellen Verarbeitung von Webdaten ist das Semantic Web. Das Ziel des Sematic Webs, welches auch Data Web genannt wird, ist es Webdokumente so zu erweitern, dass sie leichter von Rechnern verarbeitet werden können. Dabei werden Webdokumente durch Metadaten ergänzt, sodass ein Rechner Informationen extrahieren kann.
Während es für Menschen einfach ist anhand des Kontextes zu erkennen, ob es sich etwa bei dem Wort Washington um eine Person oder eine Stadt handelt, ist dies für eine Maschine nur schwer umsetzbar.
Aus diesem Grund erweitert das Sematic Web das klassische Web mit Hilfe verschiedener Frameworks, um Rechnern Daten in einem strukturierten Format zur Verfügung zu stellen (siehe Abschnitt 2.1.1).
Im Folgenden wird zunächst das RDF beschreiben, welches grundlegende Funktionalitäten für die Generierung von Linked Open Data zur Verfügung stellt. Besonders relevant für diese Arbeit ist darüber hinaus das NIF-Datenformat, welches zum Austausch von Informationen zwischen NLP-Anwendungen entwickelt wurde.

2.1.1. RDF

Das Ressource Description Framework (RDF) wurde vom World Wide Web Consortium(W3C)[1] ursprünglich als standardisiertes Modell für Metadaten entwickelt und ist heute ein grundlegender Bestandteil des Sematic Webs. RDF dient dabei zur strukturierten Darstellung von Informationen. Das Grundlegende Element bei der Modellierung mit dem RDF sind Triple. Ein Triple besteht aus folgenden Bestandteilen:

Subject, **Predicate**, **Object**

Graphisch betrachtet stellen *Subjects* und *Objects* jeweils Knoten dar. *Predicates* dagegen sind gerichtete Kanten, von einem *Subject* zu einem Object (siehe Abbildung 2.1). Sie dienen also dazu Relationen zwischen *Subjects* und *Objects* zu beschreiben (W3C, 2004b).

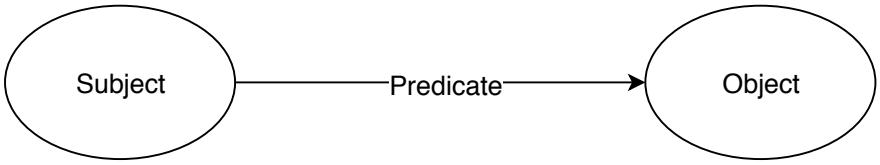

Abbildung 2.1.: RDF-Triple Quelle: (W3C, 2004b)

Eine Menge von Triples bildet einen RDF Graphen. *Subjects*,*Predicates* und *Objects* können folgende Arten von Werten annehmen:

- Subject: RDF-URI-Referenz oder Blank-Node
- Predicate: RDF-URI-Referenz
- Object: RDF-URI-Referenz, Blank-Node oder Literal

Abbildung 2.2 zeigt beispielhaft einen RDF-Graphen, der die Funktion dieser unterschiedlichen Elemente verdeutlicht. Rechtecke stellen Laterale dar. Der mittlere runde Knoten ist ein Blank-Node und der erste Knoten stellt eine URI Referenz dar.

1 https://www.w3.org/RDF/ (abgerufen am 23.8.2018)

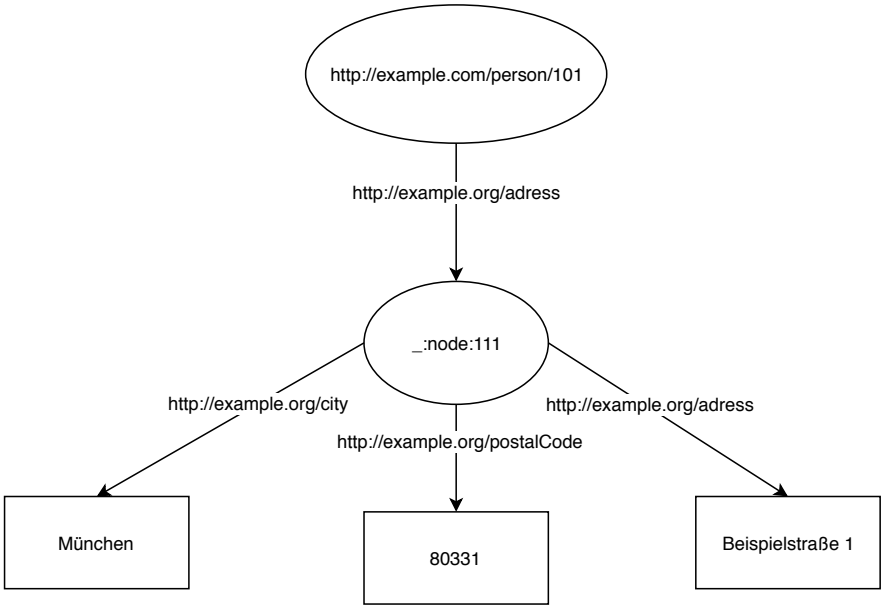

Abbildung 2.2.: RDF-Beispiel

RDF-URI-Referenz

Eine URI-Referenz ist ein Unicodestring, welcher keine *control charcaters* enthält und eine valide Sequenz von Zeichen nach *RFC 2396*[2] generiert. URIs dienen dazu *Subjects*, *Predicates* und *Objects* eindeutig zur identifizieren. Diese Objekte werden auch Ressourcen genannt (W3C, 2004b).

Blank-Nodes

Blank-Nodes sind *Subjects* und *Objects* die weder URIs noch Literale sind. Zur lokalen Identifizierung von Blank-Nodes erhalten Blank-Nodes-Identifier, welche mit der Zeichensequenz *_:* beginnen (W3C, 2004b). Blank-Nodes werden zum Beispiel verwendet, um Daten zu modellieren, welche aus mehreren separaten Teilinformationen bestehen. Beispielsweise kann die Adresse einer Person aus einer Straße, einer Postleitzahl und einer Stadt bestehen. Um diese Informationen zu aggregieren, wird die Adresse als ein Blank-Node modelliert, da dieser Knoten nur zur zum zusammenfassen mehrerer Literale verwendet wird und daher nicht global identifizierbar sein muss (W3C, 2004a).

2 http://www.faqs.org/rfcs/rfc2396.html (abgerufen am 23.8.2018)

Literale

Literale werden verwendet um Werte wie Zahlen oder Datumsangaben oder Bezeichnungen zu beschreiben. Literale sind entweder ein einfacher Unicode String mit einem optionalem *Language Tag* nach RFC 3066[3] (*plain Literal*) oder ein Unicode String kombiniert mit einer URI, welche einen Datentyp identifiziert (*typed Literal*) (W3C, 2004b).

Vokabulare

Vokabulare sind eine Menge von RDF-Ressourcen, deren URI-Referenzen alle mit dem selben Substring beginnen. Beispielsweise beginnen alle Ressourcen, welche eine Entität in der Wissensbasis Wikidata identifizieren mit dem Präfix

http://www.wikidata.org/entity/.

Diese Präfixe werden auch *Namespaces* genannt (W3C, 2004b).
Das W3C stellt für die Modellierung mit RDF die Vokabulare RDF, RDF-Schema (RDFS) und Web Ontologie Language (OWL) zur Verfügung(W3C, 2004b). Darauf aufbauend können weitere Vokabulare definiert werden. Beispielsweise enthält das Vokabular Friend of a Friend (FOAF)[4] Klassen und Relationen zur Modellierung von Beziehungen zwischen Personen (W3C, 2004b).

RDF-Datenformate

Zur Speicherung der RDF-Daten gibt es im wensentlichen drei Formate. Dies sind Turtle (Terse RDF Triple Language)[5], N-Triples[6] und RDF-XML[7]. In dieser Arbeit wird hauptsächlich mit dem Turtle-Format gearbeitet. Auflistung 2.1 zeigt beispielhaft ein kurzes Turtle-Dokument (Quelle: W3C (2014)).

3 http://www.i18nguy.com/unicode/language-identifiers.html (abgerufen am 23.8.2018)
4 http://xmlns.com/foaf/spec/ (abgerufen am 23.8.2018)
5 https://www.w3.org/TR/turtle/ (abgerufen am 23.8.2018)
6 https://www.w3.org/TR/n-triples/ (abgerufen am 23.8.2018)
7 https://www.w3.org/TR/rdf-syntax-grammar/ (abgerufen am 23.8.2018)

```
@base <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rel: <http://www.perceive.net/schemas/relationship/> .

<#green-goblin>
    rel:enemyOf <#spiderman> ;
    a foaf:Person ;    # in the context of the Marvel universe
    foaf:name "Green Goblin" .

<#spiderman>
    rel:enemyOf <#green-goblin> ;
    a foaf:Person ;
    foaf:name "Spiderman".
```

Auflistung 2.1: Turtle-Beispiel

2.1.2. NIF

Das NLP-Interchange-Format(NIF)[8] von Hellmann u. a. (2013) ist ein Vokabular, welches für die Interaktion von Natural Language Processing Tools entwickelt wurde. Das Ziel von NIF ist es den Austausch von Annotationen in Texten zwischen verschiedenen Anwendungen so zu erleichtern, dass NLP-Tools einfacher miteinander interagieren können. Hierfür ist es notwendig, Text anhand von URIs identifizieren zu können. NIF verwendet dafür ein URI-Scheme basierend auf *RFC 5147*[9].
Im NIF-Format wird zwischen einem Dokument d, einem Text t und Substrings s_t dieses Textes unterschieden. Eine URI wird aus der Dokument-URI d_u gebildet, indem das Trennzeichen # und die Start- und Endindizes des (Sub)-Textes hinzugefügt werden (Hellmann u. a., 2013).
Zur Beschreibung von Relationen zwischen Strings definieren Hellmann u. a. (2013) die NIF Core Ontologie. Siehe dazu Auflistung 2.2 als Beispiel für ein NIF-Dokument im Turtle-Format.

8 http://aksw.org/Projects/NIF.html (abgerufen am 23.8.2018)
9 https://tools.ietf.org/html/rfc5147 (abgerufen am 23.8.2018)

```
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix itsrdf: <http://www.w3.org/2005/11/its/rdf#> .
@prefix nif:   <http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .

<http://example.org/document75#char=13,25>
        a                       nif:RFC5147String , nif:String ;
        nif:anchorOf            "Stadttheater"^^xsd:string ;
        nif:beginIndex          "13"^^xsd:nonNegativeInteger ;
        nif:endIndex            "25"^^xsd:nonNegativeInteger ;
        nif:referenceContext    <http://example.org/document75#char=0,347> ;
        itsrdf:taClassRef       <http://example/entity> ;
        itsrdf:taIdentRef       <http://d-nb.info/gnd/2115459-4> , <http://www.wikidata.org/entity/Q391991> .

<http://example.org/document75#char=0,347>
        a               nif:RFC5147String , nif:String , nif:Context ;
        nif:beginIndex  "0"^^xsd:nonNegativeInteger ;
        nif:endIndex    "347"^^xsd:nonNegativeInteger ;
        nif:isString    "50997,5,,2,D,Stadttheater,Wallensteins Tod. Trauerspiel in fünf Akten...
```

Auflistung 2.2: NIF-Beispiel

2.2. Wissensbasen

Das Ziel dieser Masterarbeit ist es Entitäten aus Archivdaten zu Wisssensbasen zu verlinken. Die beiden betrachteten Wissensbasen sind die Gemeinsame Normdatei der deutschen Nationalbibliothek und Wikidata. Diese beiden Wissensbasen werden in diesem Abschnitt kurz vorgestellt.

2.2.1. Wikidata

Wikidata[10] (Wikimedia Foundation, 2012) ist eine Wissensbasis, die von der Wikimedia Foundation entwickelt und im Jahr 2012 gestartet wurde. Wikidata ist unter der public domain license verfügbar und kann in Standardformaten wie Turtle oder N-Triples abgerufen werden. Wikidata speichert Informationen in strukturierter Form, sodass diese einfach von anderen Anwendungen verarbeitet werden können. Inhalte werden sowohl von Experten als auch durch automatische Skripte generiert. Daten in Wikidata sind in verschiedenen Sprachen verfügbar. Jedes Objekt (Item) enthält eine Bezeichnung (Label), eine Beschreibung (Description) und Alternativbezeichnungen (Aliase), welche jeweils in verschiedenen Sprachen verfügbar sein können. Diese Objekte werden eindeutig durch ein Q gefolgt von einer Nummer identifiziert (Beispiel Q2971 für die Stadt Paderborn).
Zur Beschreibung der Charakteristika von Objekten werden Statements verwendet, welche eindeutig durch ein P, gefolgt von einer Nummer identifiziert werden. Beispielsweise ordnet die Eigenschaft (property) P17 (country) der Stadt Paderborn das Land

10 https://www.wikidata.org (abgerufen am 23.8.2018)

Deutschland (Q183) zu. Wikidata erstellt regelmäßige Dumps, die frei heruntergeladen werden können. In Kapitel 4.2.2 wird beschrieben, wie aus einem Wikidata-Dump ein Suchindex generiert wird.

2.2.2. Gemeinsame Normdatei der deutschen Nationalbibliothek

Die Gemeinsame Normdatei (GND)[11] (Deutsche National Bibliothek, 2012) enthält Personen, Körperschaften, Konferenzen, Geografika, Sachschlagwörter sowie Werktitel und dient zur Katalogisierung von Literatur. Sie wird aber auch in Webanwendungen oder Archiven verwendet. Die GND enthält vor allem Daten aus dem deutschsprachigem Raum und eignet sich deshalb im Rahmen dieser Arbeit gut als Ergänzung zu Wikidata, da sie beispielsweise auch Informationen über wenig bekannte Theaterschauspieler enthält, die noch nicht von Wikidata erfasst wurden.

2.3. Sequence Labeling

NER wird häufig als Sequence Labeling Aufgabe angesehen. Das derzeit am häufigsten verwendete Modell in diesem Bereich ist das Conditional Random Field (CRF), welches in diesem Kapitel eingeführt wird. Dazu werden zunächst die Verfahren Hidden Markov Model (HMM) und Maximum Entropie Markov Model (MEMM) eingeführt, welche als Vorläufer der CRFs gelten und von denen viele Konzepte auch für das CRF relevant sind. Zum Abschluss wird dann basierend auf HMMs und MEMMs das CRF-Modell beschrieben.

2.3.1. Hidden Markov Model

Hidden Markov Models von Rabiner und Juang (1986), sind ein sequenzielles Modell zur Berechnung der Wahrscheinlichkeit einer Label Sequenz für eine gegebene Beobachtungssequenz. Beispielsweise kann für einen Satz, der aus einer Menge von Wörtern besteht die wahrscheinlichste Sequenz von *Part-of-speech Tags* ermittelt werden. HMMs basieren auf Markov-Ketten, welche eine Erweiterung von endlichen Automaten darstellen, bei denen die Kantengewichte Wahrscheinlichkeiten sind und sich die Gewichte aller ausgehenden Kanten eines Zustands auf eins summieren.
Durch eine Markov-Kette kann die Wahrscheinlichkeit für eine bekannte Sequenz von Labeln einfach durch Multiplikation der Kantengewichte berechnet werden. Bei HMMs dagegen ist die Sequenz der Label nicht bekannt. Sie ist versteckt. Bekannt ist nur eine

11 ww.dnb.de/DE/Standardisierung/GND/gnd_node.html (abgerufen am 23.8.2018)

Sequenz von Beobachtungen V, der eine gleich lange Sequenz von Labels zugeordnet werden soll. Ein HMM λ ist nach Rabiner und Juang (1986) wie folgt definiert:

$Q = \{q_1, ..., q_n\}$ Menge der Zustände
$V = \{v_1, ..., v_n\}$ Menge der Beobachtungen
$\pi = \{\pi_i\}, \pi_i = P(q_i \text{ in } t = 1)$ initaiale Verteilung der Zustände
$A = \{a_{i,j}\}, a_{i,j} = P(q_j \text{ in } t + 1 | q_i \text{ in } t)$ Wahrscheinlichkeitsverteilung der Zustandsübergänge
$B = \{b_j(k)\}, b_j(k) = P(v_k \text{ in } t | q_j \text{ in } t)$ Wahrscheinlichkeitsverteilung der Beobachtungen in Zustand j

Im folgenden wird für ein HMM λ die verkürzte Schreibweise $\lambda = (A, B, \pi)$ verwendet. Rabiner und Juang (1986) beschreiben drei Probleme für HMMs, die auch für alle anderen sequentiellen Modelle gelöst werden müssen:

1. Für eine Sequenz $O = O_1, O_2, ..., O_T$ und ein HMM $\lambda = (A, B, \pi)$: Wie kann die Wahrscheinlichkeit für eine Beobachtungssequenz $P(O|\lambda)$ berechnet werden?
2. Für eine Sequenz $O = O_1, O_2, ..., O_T$ und ein HMM $\lambda = (A, B, \pi)$: Wie kann die optimale Sequenz von Zuständen $I = i_1, i_2..., i_t$ berechnet werden?
3. Wie können die Parameter des HMM $\lambda = (A, B, \pi)$ berechnet werden um $P(o|\lambda)$ zu optimieren?

Im Folgenden wird nun beschrieben, wie alle drei Probleme gelöst werden können.

Problem 1: Evaluation

Das erste Problem wird auch Evaluation genannt. Eine einfache Lösung für das Problem ist es die Wahrscheinlichkeit für jede mögliche Sequenz von Zuständen für O zu berechnen und aufzusummieren. Für jede Sequenz von Zuständen $I = i_1, i_2, ..., i_T$ gilt für die Wahrscheinlichkeit der Beobachtungssequenz O:

$$P(O|I, \lambda) = b_{i1}(O_1), b_{i2}(O_2), ...b_{iT}(O_T)$$

Für die Wahrscheinlichkeit der Zustandssequenz I gilt:

$$P(I|\gamma) = \pi_{i_1}, a_{i_1,i_2}, a_{i_2,i_3}, ...a_{i_{t-1},i_t}$$

Zur Berechnung von $P(O, I|\lambda)$, also dass die Beobachtungssequenz und Zustandssequenz zusammen auftreten, müssen die beiden Terme multipliziert werden. Somit gilt:

$$P(O, I|\lambda) = P(O|I, \lambda) * P(I|\lambda)$$

Die Wahrscheinlichkeit für O ergibt sich dann durch das Aufsummieren aller möglicher Zustandssequenzen $P(O|\lambda) = \sum_{allI} P(O, I|\lambda)$. Für N Zustände gibt es somit N^T mögliche Sequenzen. Insgesamt sind somit $2TN^T$ Berechnungsschritte notwendig (Rabiner und Juang, 1986). Die Methode ist somit nicht effizient und für besonders große Modelle mit vielen Zuständen und langen Beobachtungssequenzen nicht anwendbar. Eine effizientere Methode zum Lösen des Problems ist der Forward-Algorithmus (Rabiner und Juang, 1986). Dafür werden spezielle Forward-Variablen $\alpha_t(i)$, zur Berechnung der Wahrscheinlichkeit einer Teilsequenz eingeführt, welche zum Zeitpunkt t nach t Zustandsübergängen in Zustand S_i endet:

$$\alpha_t(i) = P(O_1, O_2, ..., O_t, q_t = S_i|\lambda)$$

Die Wahrscheinlichkeit für eine Beobachtungssequenz O für ein Modell λ kann mit Hilfe der Forward-Variablen induktiv wie folgt berechnet werden:

Inititalisierung

$$\alpha_1(i) = \pi_i b_i(O_1), \quad 1 \leq i \leq N$$

Induktion

$$\alpha_{t+1}(j) = \left(\sum_{i=1}^{N} \alpha_t(i) a_{i,j} \right) b_j(O_{t+1}), \quad 1 \leq t \leq T-1, 1 \leq j \leq N$$

Terminierung

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

Im Gegensatz zum ersten beschriebenen Ansatz braucht dieses Verfahren nur N^2T Schritte zur Berechnung von $P(O|\lambda)$ und kann somit auch für große Modelle angewendet werden (Rabiner und Juang, 1986).

Problem 2: Dekodierung

Das zweite Problem beschäftigt sich damit für eine gegebene Beobachtungssequenz $O = O_1, O_2, ..., O_T$ und ein gegebenes Modell λ die optimale Zustandssequenz zu er-

mitteln. Dabei können verschiedene Sequenzen von Zuständen als optimal betrachtet werden. Zum Beispiel der zu einem Zeitpunkt t individuell wahrscheinlichste Zustand. Das am häufigsten verwendete Optimalitätskriterium ist die wahrscheinlichste Sequenz von Zuständen Q. Ziel ist es also $P(Q|O,\lambda)$ zu maximieren, was äquivalent zur Maximierung von $P(Q,O|\lambda)$ ist (Rabiner und Juang, 1986). Berechnet werden kann dies durch den Viterbi-Algorithmus, welcher nachfolgend beschrieben wird. Zunächst wird die Größe $\delta_t(i)$ definiert:

$$\delta_t(i) = \max_{q_1,q_2,...,q_n} P(q_1, q_2, ..., q_t = i, O_1, O_2...O_t|\lambda)$$

δ ist die höchste Wahrscheinlichkeit, dass ein einzelner Pfad, zum Zeitpunkt t nach t Beobachtungen im Zustand S_i endet. Dies ist induktiv wie folgt definiert:

$$\delta_{t+1}(j) = [\max_i \delta_t(i) * a_{i,j}] * b_i(O_{t+1})$$

Die Berechnung erfolgt durch folgendes induktives Verfahren. Dabei wird ein $\psi_t(j)$ zur Speicherung, der maximalen Elemente für $\delta_{t+1}(j)$ eingeführt, damit später durch Backtracking der optimale Pfad berechnet werden kann.

Initialisierung

$\delta_1(i) = \pi_i b_i(O_1), \quad 1 \leq i \leq N$

$\psi_1(i) = 0$

Rekursion

$\delta_t(j) = \max_{1\leq i\leq N}[\delta_{t-1}(i) * a_{i,j}] * b_j(O_t), \quad 2 \leq t \leq T, 1 \leq j \leq N$

$\psi_t(j) = \arg\max_{1\leq i\leq N}[\delta_{t-1}(i) * a_{i,j}], \quad 2 \leq t \leq T, 1 \leq j \leq N$

Terminierung

$P^* = \max_{1\leq i\leq N}[\delta_T(i)]$

$q_T^* = \arg\max_{1\leq i\leq N}[\delta_T(i)]$

Backtracking der optimalen Zustandssequenz

$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, ..., 1$

Problem 3: Optimierung der Parameter

Die Verfahren zum Lösen der ersten beiden Probleme können in leicht angepasster Form auch auf für MEMMs und CRFs angewandt werden (siehe die Abschnitte 2.3.2 und 2.3.3). Zur Lösung des dritten Problems wird eine Variante der Expactation-Modification-Methode (EM) verwendet.
Diese ist unter dem Namen Baum-Welch-Algorithmus bekannt, siehe dafür Rabiner und Juang (1986).

2.3.2. Maximum Entropy Markov Model

HMMs haben zwei entscheidende Nachteile für NLP-Applikationen. Ein Problem ist, dass Wahrscheinlichkeiten der Beobachtungen als Multinomialverteilung über ein diskretes abgeschlossenes Vokabular repräsentiert werden. (McCallum u. a., 2000). Für viele NLP-Applikationen, insbesondere auch für NER ist es aber sinnvoll, Beobachtungen über mehrere überlappende Features zu modellieren. So können beispielsweise Features über *Part-of-speech Tags*, Wortendungen, oder die Großschreibung von Wörtern gebildet werden.
Des Weiteren ist es möglich, das kein abgeschlossenes Vokabular vorliegt. Es können beispielsweise ganze Sätze als Beobachtungen betrachtet werden. Das Berechnen einer Multinomialverteilung ist jedoch auf Grund der großen Menge von möglichen Sätzen nicht möglich (McCallum u. a., 2000).
Das zweite Problem von HMMs ist, dass die Parameter durch den Baum-Welch-Algorithmus so gelernt werden, dass die Wahrscheinlichkeit für eine Beobachtungssequenz maximiert wird. Jedoch wird in den meisten Anwendungen verlangt eine Zustandssequenz für eine gegebene Beobachtungssequenz zu berechnen. HMMs benutzen hier ein Modell, welches *Generative Joint Model* genannt wird. MEMMs und auch CRFs verwenden dagegen ein *Discriminative Conditional Model*, also ein Model, welches die bedingte Wahrscheinlichkeit für eine gegebene Zustandssequenz direkt berechnet (McCallum u. a., 2000).
Im Allgemeinen berechnen generative Modelle die gemeinsame Wahrscheinlichkeit $p(x, y)$, aus der mittels Bayes Theorem $p(y|x)$ die bedingte Wahrscheinlichkeit für y gegeben x abgeleitet wird. Diskriminative Modelle berechnen $p(y|x)$ direkt (Ng und Jordan, 2001). In der Praxis funktionieren diskriminative Modelle besser als generative Modelle (Ng und Jordan, 2001) und werden deshalb gegenüber generativen Modellen bevorzugt. Auch für NER sind diskriminative Modelle in Form von CRFs der bevorzugte Ansatz.
Die Idee der MEMMs ist es die Wahrscheinlichkeit für das Erreichen eines Zustandes als bedingte Wahrscheinlichkeit gegeben einer Beobachtung und des vorherigen Zustands

zu berechnen. Dafür wird ein exponentielles Modell verwendet. Die Wahrscheinlichkeitsverteilungen der Beobachtungen und Zustände wird durch die Funktion $P(s|s', o)$, mit s als momentanem Zustand, s' als vorherigem Zustand und o als momentaner Beobachtung, ersetzt. Diese Wahrscheinlichkeit repräsentiert in MEMMs die Wahrscheinlichkeit für die Transition von s' nach s für die Beobachtung o (McCallum u. a., 2000).
Für jeden Zustandsübergang von einem Zustand s' nach einem Zustand s für eine Beobachtung o wird in MEMMs jeweils eine Transitionsfunktion $P_{s'}(s|o)$ definiert. Diese Transitionsfunktionen werden durch n separate nicht unabhängige binäre Features modelliert. Jedes Feature a definiert eine Funktion $f_a(o, s)$ mit den Parametern o für die momentane Beobachtung und s für einen möglichen Folgezustand. Dabei besteht a aus einem binären Feature b und dem gewünschtem Folgezustand s und für $f_{<b,s>}(o_t, s_t) = f_a(o_t, s_t)$ gilt:

$$f_{<b,s>}(o_t, s_t) = \begin{cases} 1 \text{ wenn } b(o_t) \text{ ist wahr und } s = s_t \\ 0 \text{ sonst} \end{cases} \quad \text{(McCallum u. a., 2000)}$$

Der erwartete Wert jedes Features in der zu lernenden Verteilung soll dabei dem Durchschnitt auf der Beobachtungssequenz entsprechen. Die Maximum-Entropie-Verteilung, die diesem Durchschnitt entspricht hat die Form

$$p_{s'}(s, o) = \frac{1}{Z(o, s')} exp\left(\sum_w \lambda_a f_a(o, s)\right)$$

mit den Parametern λ_a, welche gelernt werden müssen und dem Normalisierungsfaktor $Z(o, s')$, da sich die Verteilungen über alle Zustände auf eins summieren müssen (siehe dazu McCallum u. a. (2000)). Die Berechnung der wahrscheinlichsten Sequenz funktioniert analog wie für HMMs indem die Forward-Variablen wie folgt berechnet werden:

$$a_{t+1}(s) = \sum_{s' \in S} \alpha_t(s') * P'_s(s|o_{t+1})$$

Zum trainieren der Parameter für eine Menge von Beobachtungen mit einer gegebenen Sequenz von Zuständen kann das Verfahren Generalized-Iterative-Scaling (N. Darroch und Ratcliff, 1972) verwendet werden. Siehe dazu McCallum u. a. (2000).

2.3.3. Conditional Random Field

Das derzeit dominante Vefahren für Named Entity Recognition sind Conditional Random Fields Lafferty u. a. (2001), welche eine Erweiterung zu den HMMs und MEMMs darstellen. CRFs lösen dabei das Label Bias Problem, welches für MEMMs besteht und an dieser Stelle kurz beschrieben wird.
Das Problem von MEMMs ist, dass der Übergang von einem Zustand c zu einem Zustand c^* nur von c selbst und der Beobachtung o_c abhängt. Spätere Zustände werden aber nicht betrachtet. Dies führt dazu, dass Zustände, mit vielen möglichen Nachfolgern einen höheren Einfluss auf die Zustandssequenz haben als Zustände mit weniger möglichen Nachfolgern. Im Extremfall hat ein Zustand nur einen Nachfolger, was dazu führt, dass die Beobachtung an dieser Stelle komplett ignoriert wird. Die Markov Annahme von MEMMs, dass der Folgezustand nur vom momentanem Zustand und der momentanen Beobachtung abhängt bildet nicht mit ab, dass der Folgezustand auch von einer späteren Beobachtung oder einem späteren Zustand abhängen kann (Lafferty u. a., 2001). Lafferty u. a. (2001) beschreibt dazu das Beispiel, welches in Abbildung 2.3 dargestellt ist.

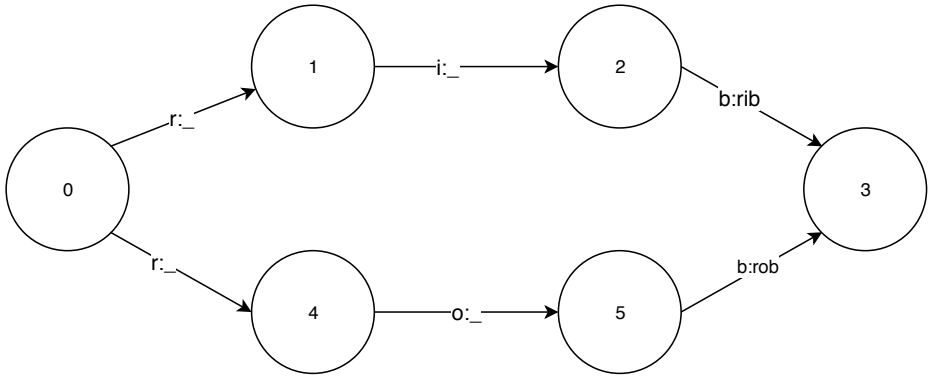

Abbildung 2.3.: Label Bias Problem Quelle: Lafferty u. a. (2001)

Das Modell in Abbildung 2.3 ist ein einfaches Modell, welches zwischen den Strings rib und rob unterscheiden soll. Für die Sequenz *rib* geschieht nun folgendes: r passt zu beiden Transitionen, also werden beide Transitionen gleich wahrscheinlich ausgewählt. Da Knoten eins und vier jeweils nur eine ausgehende Kante haben, wählen sie beide unabhängig von der Beobachtungssequenz diese Kante aus, obwohl Knoten 4 das *i:_* nicht beobachtet. Beide Pfade sind bei der Dekodierung also gleich wahrscheinlich. Falls beim Trainieren des Modells eines der beiden Wörter häufiger vorkommt, hätte entweder die Kante $(0, 1)$ oder die Kante $(0, 4)$ eine höhere Wahrscheinlichkeit, sodass eine der beiden Sequenzen immer gewinnen würde (Lafferty u. a., 2001).

Conditional Random Fields lösen dieses Problem, indem in jedem Zustand die gesamte Beobachtungssequenz betrachtet wird und Folgezustände auch von späteren und früheren Zuständen abhängen können. Lafferty u. a. (2001) definieren ein Conditional Random Field folgendermaßen:

Sei $G = (V, E)$ ein Graph, sodass $Y = (Y_v)_{v \in v}$ durch die Kanten E repräsentiert wird. Dann ist (X, Y) ein Conditional Random Field, wenn die Zufallsvariablen Y_v bedingt durch X der Markov Eigenschaft gehorchen, sodass $p(Y_v|X, Y_w, w \neq v) = p(Y_v|X, Y_w, w \sim v)$.
($w \sim v$ = bedeutet, dass w und v Nachbarn in G sind)

Ein CRF ist also ein Zufallsfeld, welches global von einer Beobachtungssequenz X abhängt. Eine häufig verwendetet Variante von CRFs sind Linear-Chain-CRFs bei denen die Knoten in einer linearen Kette verbunden sind. Nach Mccallum und Li (2003) definieren Linear-Chain-CRFs die bedingte Wahrscheinlichkeit $P(s|o)$ für eine Zustandssequenz s für eine gegebene Beobachtungssequenz o wie folgt:

$$p(s, o) = \frac{1}{Z_x} exp\left(\sum_{t=1}^{T} \sum_{k} \lambda_k f_k(s_{t-1}, s_t, o, t)\right)$$

$f_k(s_{t-1}, s_t, o, t)$ ist dabei eine Feature Funktion für das Feature k und λ_k sind zu lernende Gewichte für jede Feature Funktion. Z_O ist ein Normalisierungsfaktor über alle möglichen Zustandssequenzen. Für Z_X gilt:

$$Z_O = \sum_{s \in s^T} exp\left(\sum_{t=1}^{T} \sum_{k} \lambda_k f_k(s_{t-1}, s_t, o, t)\right)$$

Der Normalisierungsfaktor Z_O kann mit modifizierten Forward-Variablen durch den Viterbi-Algorithmus berechnet werden. Die Forward-Variablen werden wie folgt definiert:

$$\alpha_{t+1}(s) = \sum_{s'} exp\left(\sum_{k} f_k(s_{t-1}, s_t, o, t)\right)$$

für Z_0 gilt dann $Z_O = \sum_s \alpha_T(s)$ (Mccallum und Li, 2003). Das Lernen der Parameter erfolgt wie bei den MEMMs über eine Variante des Iterative-Scaling-Verfahrens, siehe dazu Lafferty u. a. (2001), weitere Möglichkeiten zum Lernen der Parameter sind das Quasi-Newton-Verfahren Mccallum und Li (2003), sowie Gradientenverfahren.

2.4. HITS-Algorithmus

Wie in Kapitel 2.1.1 beschrieben bilden eine Menge von RDF-Triplen einen RDF-Graphen. Dieser hat prinzipiell die gleichen Eigenschaften, wie eine Menge von Webseiten, die durch Hyperlinks verbunden wurden.
Für das Finden relevanter Webseiten gibt es im Bereich des Information Retrievals-Verfahren, welche anhand eines durch Hyperlinks induzierten Graphen relevante Ressourcen finden. Die beiden bekanntesten derartigen Verfahren sind zum einen der PageRank-Algorithmus von Brin und Page (1998) und zum anderen der HITS-Algorithmus von Kleinberg (1999).
Diese Verfahren sind im wesentlichen auch auf RDF-Grapen anwendbar. Der HITS-Algorithmus wird deshalb im Rahmen des Linkings dazu verwendet den am besten passenden Kandidaten zu finden (Kapitel 4.2.1). Der HITS-Algorithmus beschäftigt sich mit der Filterung von relevanten Seiten unter Verwendung der Hyperlink Struktur. Der Algorithmus wird in Information Retrieval-Systemen zur Generierung eines Rankings einer Menge von Webseiten verwendet und besteht im wesentlichen aus zwei Schritten:

1. Konstruktion eines Subgraphen aller (Web)-Seiten
2. Berechnung von *Hubs* und *Authorities*

Diese beiden Schritte sollen nun im Weiteren beschrieben werden.

2.4.1. Konstruktion eines Subgraphen

Eine Menge von verlinkten Webseiten kann als gerichteter Graph $G = (V, E)$ betrachtet werden. Die Knoten Menge V repräsentiert die Menge der Webseiten und die Menge der gerichteten Kanten $(p, q) \in E$ die Links zwischen je zwei Seiten. Kleinberg (1999) definiert den *out degree* eines Knoten p als die Anzahl der ausgehenden Kanten von p und den *in degree* als die Anzahl der eingehenden Kanten des Knotens p. Eine Teilmenge W der Knoten V mit $W \subseteq V$ induziert einen Teilgraphen $G[W]$ mit der Knotenmenge W und allen Kanten aus E zwischen den Knoten aus W.
Das Ziel ist es nun einen Teilgraphen G_σ für eine Query σ zu konstruieren, mit dem durch Analyse der Linkstruktur die *Authorities* identifiziert werden können. *Authorities* sind Seiten, welche relevant sind und einen hohen *in degree* besitzen.

Die Menge S_ω der Knoten von G_σ sollte folgende Bedingungen erfüllen:

1. S_σ soll relativ klein sein
2. S_σ soll möglichst relevante Knoten enthalten
3. S_σ soll die meisten *authorities* enthalten

Zur Erfüllung der ersten beiden Bedingungen wird eine Basismenge R_σ berechnet. Hierzu wird ein Parameter t festgelegt und die t höchst gerankten Seiten einer Query σ ausgewählt. R_σ enthält somit die t höchst gerankten Seiten. R_σ erfüllt zwar die ersten beiden Bedingungen, jedoch ist die dritte Bedingung nicht zwangsläufig erfüllt, da eine *Authority* nicht unbedingt eine der t relevantesten Seiten ist. Zudem enthält R_σ oft sehr wenige Kanten. Vergleiche dazu Kleinberg (1999).
Zur Erfüllung der dritten Bedingung wird die Menge R_σ erweitert, indem Knoten, die bislang nicht in R_σ enthalten, aber mit einem Knoten in R_σ verbunden sind hinzugefügt werden. Dies geschieht mit folgender Methode:

$\sigma : eineQuery$

$E : eineSuchmaschine$

$t, d \in \mathbb{N}$

R_σ : *top* t Ergebnisse aus E für σ

Algorithm 1: Subgraph(R_σ, d)

```
S_σ ⟵ Rσ;
foreach p ∈ R_σ do
    Γ⁺(p) = alle Kanten e für die gilt e = (p, q) ∈ E;
    Γ⁻(p) = alle Kanten e für die gilt e = (q, p) ∈ E;
    S_σ.add_all(Γ⁺(p));
    if |Γ⁻(p)| ≤ d then
        S_σ.add_all(Γ⁻(p));
    end
    else
        S_σ.add_abitrary_Set(Γ⁻(p),d);
    end
end
return D[K, L]/max(K, L)
```

Diese Methode fügt alle Seiten hinzu, auf die mindestens eine Seite aus R_σ zeigt. Gleichzeitig werden umgekehrt eingehenden Knoten in R_0 hinzugefügt, wobei ein Knoten p maximal d Knoten hinzufügen darf. Das Ergebnis ist die Basismenge S_σ, welche den Graphen G_σ induziert, mit welchem im nächsten Kapitel *Authorities* identifiziert werden.
Zusätzlich zur Konstruktion dieses Teilgraphen G_σ beschreibt Kleinberg (1999) Heuristiken zur Eliminierung von Navigationslinks einer Domain und von Werbeseiten. Diese sind allerdings nicht für AGDISTIS relevant, da dieses Problem in einer Wissensbasis nicht auftritt und werden dementsprechend an dieser Stelle nicht weiter erläutert.

2.4.2. Berechnung von *Hubs* und *Authorities*

Die einfachste Variante zur Berechnung der *Authorities* ist es, die Knoten einfach nach ihrem *in degree*, also nach der Menge der eingehenden Kanten zu ordnen. Das Problem ist allerdings, dass mache Seiten an sich schon sehr populär sind und unabhängig vom Themengebiet, nach dem in der Query σ gesucht wurde, generell viele eingehende Kanten haben. Kleinberg (1999) hat jedoch festgestellt, dass sich bei *Authorities*, welche relevant für die Query sind, die Menge der Seiten, die auf diese *Authorities* zeigen überschneiden. Aus diesem Grund werden neben den *Authorities* *Hubs* berechnet, welche Knoten sind, die Links zu möglichst vielen *Authorities* haben. Gute *Hubs* besitzen dabei Links zu möglichst vielen *Authorities* und gute *Authorities* werden von möglichst vielen *Hubs* verlinkt, Kleinberg (1999) beschreibt dies als *mutually reinforcing relationship*.
Zur Berechnung der *Hubs* und *Authorities* wird ein iterativer Algorithmus verwendet. Dabei wird zunächst jedem Knoten $p \in G_\sigma$ ein *Authority-Weight* x^p und ein *Hub-Weight* y^p zugeordnet, diese beiden Gewichte werden normalisiert, sodass ihre Quersumme jeweils gleich eins ist:

$$\sum_{p \in S_\sigma E} (x^p)^2 = 1$$
$$\sum_{p \in S_\sigma E} (y^p)^2 = 1$$

Seiten mit hohen Werten für x sind dabei gute *Authorities* und Seiten mit hohen Werten für y gute *Hubs*.
Das bedeutet, dass eine Seite p einen hohen Wert für y haben soll, wenn sie auf viele Seiten mit einem hohem Wert für x zeigt, beziehungsweise p einen hohen Wert für x haben soll, wenn viele Seiten mit hohem x-Wert auf sie zeigen.

Kleinberg (1999) beschreibt dafür die beiden Operationen I und O welche für eine gegebenen Mengen von Gewichten $\{x^p\}$ und $\{y^p\}$ neue Gewichte berechnen:

$$I : x^p \longleftarrow \sum_{q:(q,p)\in E} y^q$$
$$O : y^p \longleftarrow \sum_{q:(q,p)\in E} x^q$$

Basierend auf diesen beiden Operationen kann der Algorithmus nun Werte für $\{x^p*\}$ und $\{y^p*\}$ berechnen (siehe Algorithmus 2). Die Gewichte aller Knoten p aus G_σ werden dafür in den Vektoren x und y gespeichert.

Algorithm 2: Iterate(G, k)

```
x ⟵ [1, 1, ..., 1] (Authority Werte der Knoten) ;
y ⟵ [1,1,...,1] (Hub Werte der Knoten) ;
for i → k do
    foreach x_j ∈ x do
        I(x_j,y,G) ;
    end
    foreach y_j ∈ y do
        O(y_j,x,G) ;
    end
    normalize(x) ;
    normalize(y) ;
end
```

Anschließend können die besten *Hubs* und *Authorities* ausgegeben werden. Für ein ausreichend großes k konvergieren die Werte. Siehe dazu Kleinberg (1999).

3. Stand der Forschung

In diesem Kapitel soll der aktuelle Stand in allen für diese Masterarbeit relevanten Forschungsbereichen dargestellt werden. Dies umfasst die Bereiche Named Entity Recognition (NER) und Entity Linking (EL), da es das Ziel dieser Arbeit ist benannte Entitäten aus Archivdaten zu extrahieren und zu verlinken.
Zunächst wird der aktuelle Stand im Bereich NER beschrieben, dies umfasst zunächst eine Definition des NER-Problems, die Beschreibung aller Arten von Ansätzen und eine kurze Übersicht der am häufigsten verwendeten Systeme. Abschließend wird beschrieben warum diese Ansätze nicht für die in dieser Arbeit betrachtete Archivdaten geeignet sind.
Des Weiteren wird auf den Bereiche Entity Linking eingegangen. Dies umfasst ebenfalls die Definition des Problems, die Beschreibung des grundlegenden Aufbaus eines EL-Systems und die Nennung populärer Ansätze. Abschließend wird auch hier das in dieser Arbeit entwickelte System von den bisherigen Ansätzen abgegrenzt.

3.1. Named Entity Recognition (NER)

Named Entity Recognition ist ein Teilproblem der Informationsextraktion und beschäftigt sich mit der Erkennung und Kategorisierung von benannten Entitäten wie Personen und Städten in Texten (Mansouri u. a., 2008).
NER ist eine wichtige Grundlage für verschiedene NLP-Aufgaben wie dem maschinellen Übersetzen von Texten oder dem Information Retrieval (Mansouri u. a., 2008).
Das Erkennen von Entitäten ist eine für Menschen intuitiv recht einfach lösbare Aufgabe, dennoch ist die Automatisierung von NER sehr schwierig, da keine eindeutige Regeln zur Identifikation von benannten Entitäten definiert werden können und aufgrund von Ambiguität auch das Klassifizieren von Entitäten in Kategorien wie Personen oder Orte nicht trivial ist. Named Entity Recognition umfasst im wesentlichen zwei Teilaufgaben: Im ersten Schritt sollen Entitäten in einem Dokument identifiziert werden. Anschließend erfolgt die Einteilung der gefunden Entitäten in Kategorien, wie Personen, Organisationen oder Orte. Je nach Ansatz erfolgt die Erkennung und Klassifizierung von Entitäten in einem Schritt oder nacheinander (Mansouri u. a., 2008).

Die Klassifizierung der Entitäten ist dabei keine einfache Aufgabe, da häufig Ambiguitäten vorliegen. Beispielsweise kann mit dem Wort Washington je nach Kontext eine Person, die Hauptstadt der USA oder ein Bundesstaat der USA gemeint sein (Mansouri u. a., 2008).
In Kapitel 3.2 wird dazu das verwandte Problem des Entity Linkings beschrieben, welches sich mit Zuordnung von Entitäten zu Ressourcen in einer Wissensbasis befasst.
NER-Systeme werden meist für eine bestimmte Sprache entwickelt und sind somit nicht ohne weiteres auf andere Sprachen übertragbar, da Sprachen unterschiedliche grammatikalische und syntaktische Eigenschaften haben (Mansouri u. a., 2008; Konkol, 2015). Beispielsweise beginnen in der deutschen Sprache alle Nomen mit einem Großbuchstaben, während im Englischen nur Eigennamen großgeschrieben werden. Die Eigenschaft, dass ein Wort mit einem Großbuchstaben beginnt ist somit für die Erkennung von Entitäten bei der englischen Sprache ein wichtiges Kriterium, während es in für die deutschen Sprache eine geringere Bedeutung hat (Konkol, 2015).
Die englische Sprache ist im NER-Bereich die am meisten untersuchte Sprache, da es bereits eine große Zahl an Systemen und Textkorpora zur Evaluation dieser Systeme gibt. Erstmals wurde Englisch bei der MUC-6 Konferenz behandelt, bei welcher auch erstmals das NER-Problem sowie der Begriff Named Entity definiert wurden (Sundheim, 1995).
Im Rahmen dieser Arbeit werden Archivdaten in überwiegend deutscher Sprache betrachtet. Ein System für die deutsche Sprache wurde erstmals im Rahmen der CoNLL-2003 (Tjong Kim Sang und De Meulder, 2003) Konferenz entwickelt. Das beste System von Florian (2002) erreichte ein F1-Measure von 78 %, neuere Systeme wie FOX (Speck und Ngonga Ngomo, 2017) erreichen je nach Domäne bis zu 80 % F1-Measure. Für die deutsche Sprache gibt es neben dem CoNLL-2003 Korpus bisher nur wenige Textkorpora. Insbesondere gibt es bisher keinen Korpus für Archivdaten. Aus diesem Grund muss für diese Arbeit ein neuer Korpus erstellt werden, was in Kapitel 6.2 näher erläutert wird.
Neben einsprachigen Systemen gibt es auch multilinguale Systeme, bei denen auf sprachabhängige Tools wie Lemmatisierung und *Part-of-speech Tagging* verzichtet wird. Dies erhöht die Adaptierbarkeit auf neue Sprachen (Konkol, 2015).
Neben der Sprache hat auch die Domäne, auf die ein System angewendet wird einen hohen Einfluss auf die Performance. In der Literatur wurden bereits verschiedene Domänen wie Nachrichtenartikel, Social-Media-Beiträge oder Wikipedia-Artikel untersucht, wobei NER-Systeme, die auf Grundlage von Daten eines bestimmten Bereichs entwickelt wurden oft eine schlechtere Performance erreichen, wenn sie auf eine andere Domäne

angewendet werden (Konkol, 2015).
Das NER-Problem ist dabei für einige Domänen wie Nachrichten einfacher zu lösen als für andere und je nach Bereich können Dokumente mehr oder weniger strukturiert oder umfangreich sein. Längere Texte weisen meist einen größeren Kontext auf, was die Klassifizierung von Entitäten in Kategorien wie Personen oder Städte erleichtert (Mansouri u. a., 2008). Zudem können für gut strukturierte Texten wie Wirtschaftsdaten leichter Regeln zum Finden von Entitäten gefunden werden.
Die in dieser Arbeit betrachtete Daten sind tabellarische Archivdaten, für welche es bislang noch kein NER System gibt (Mansouri u. a., 2008). Da bisherige Systeme, wie zuvor beschrieben, nicht ohne weiteres auf diese Art von Daten angewendet werden können, ist es notwendig ein neues Modell für die Extraktion von Entitäten zu entwickeln. Dieses wird in Kapitel 4.1 beschrieben.
Im wesentlichen gibt es drei verschiedene Arten von Ansätzen für das Named Entity Recognition Problem:
Zunächst sind dies regelbasierte Ansätze, welche eine Menge von Experten entwickelten Regeln zur Erkennung von Entitäten verwenden. Die zweite Klasse von Ansätzen nutzt Verfahren des maschinellen Lernens zur Erkennung von Entitäten. Des Weiteren gibt es noch hybride Verfahren, welche versuchen die Vorteile von regelbasierten- und Machine Learning-Ansätzen zu kombinieren.
Im Folgenden werden zunächst in Abschnitt 3.1.1 kurz die regelbasierten Ansätze vorgestellt, welche aktuell vor allem dann genutzt werden, wenn nicht ausreichend Trainingsdaten vorhanden sind oder es nicht möglich ist Trainingsdaten zu generieren. Anschließend werden in Abschnitt 3.1.2 die Machine-Learning basierten Ansätze beschrieben, welche sich gegenüber regelbasierten Verfahren durchgesetzt haben und von einem großen Teil aller derzeitigen NER-Systeme angewendet werden.

3.1.1. Regelbasierte Ansätze

Ein erster Ansatz ist die regelbasierte Erkennung von Entitäten welcher eine Menge von Experten entwickelten Regeln in Kombination mit Wörterbüchern verwendet (Mohit, 2014). Ziel ist die Abbildung von grammatikalischen und orthografischen Mustern, mit denen Entitäten möglichst eindeutig identifiziert werden können. Zusätzlich zu den Regeln werden oft externe Ressourcen wie Gazetteers oder Wörterbücher verwenden, die eine Menge von Eigennamen enthalten. Diese werden von vielen Ansätzen verwendet, um in einem ersten Schritt mögliche Entitäten zu markieren, um anschließend unter Anwendung von Regeln zu entscheiden, ob es sich um eine Entität handelt (Mohit, 2014). Beispielsweise deutet das Vorkommen eines Vornamens auf eine Person hin.

Die Regeln können dann die bereits markierten Schlüsselbegriffe nutzen, um darauf aufbauend Entitäten zu identifizieren. Durch eine Regel können dann Namenszusätze wie Dr. oder der Nachname einer Person identifiziert werden. Es ist jedoch auch möglich Regeln zu definieren, welche allein die syntaktische und grammatikalische Struktur eines Dokuments nutzen und keine externen Ressourcen benötigen. Eine Folge von Großbuchstaben deutet beispielsweise auf eine Abkürzung einer Entität hin.
Regeln können manuell oder durch einen Bootstrapping-Ansatz gefunden werden und je nach Klasse wie Personen oder Orten können unterschiedliche Regeln definiert werden (Mansouri u. a., 2008).
Ein großer Nachteil regelbasierter Ansätze ist es, dass eine Menge von Regeln im allgemeinen nicht auf andere Domänen adaptiert werden können und die Regeln auch von der Sprache abhängig sind. Zudem werden je nach Domäne auch unterschiedliche Wörterbücher und Gazetteers benötigt. Die Anpassung der Regeln ist oft sehr aufwendig und je nach Anwendung eine äußerst schwierige Aufgabe, zu der auch ein gutes Wissen über die Domäne erforderlich ist (Mansouri u. a., 2008).

3.1.2. Maschinelles Lernen

Ein weiterer Ansatz ist die Anwendungen von maschinellem Lernen zur Erkennung benannter Entitäten. Die Ansätze können im wesentlichen in überwachte, semiüberwachte und unüberwachte Methoden unterteilt werden, wobei der überwachte Ansatz das derzeit dominierende Vorgehen ist. Das Ziel der überwachten Ansätze ist das Lernen von Features aus einer großen Menge von Trainingsdaten.
Das NER-Problem kann dabei als eine Klassifikationsaufgabe angesehen werden, bei der jedem Token ein Label aus einer Klasse zugeordnet werden soll. Das Problem an diesem Ansatz ist, dass Abhängigkeiten zwischen Tokens nicht oder nur unzureichend berücksichtigt werden. (Mansouri u. a., 2008)
Deshalb wird das NER-Problem meist als ein Problem des sequenziellen Lernens aufgefasst, bei dem einer Sequenz von Tokens eine gleich lange Sequenz von Labeln zugeordnet wird. Häufig verwendete Methoden sind die bereits in Kapitel 2.3 beschriebenen Modelle HMM, MEMM sowie CRF. Andere Ansätze verwenden auch künstliche neuronale Netze zur Vorhersage einer Sequenz von Named-Entity-Labels (Ratinov und Roth, 2009).
Ein weiteres Verfahren des maschinellen Lernens sind semiüberwachte Verfahren, die im wesentlichen auf Bootstrapping basieren (Mansouri u. a., 2008). Das System bekommt dabei eine kleine Menge an Beispielen und versucht dann weitere Entitäten zu finden, die in einem ähnlichen Kontext verwendet werden.

Des Weiteren gibt es unüberwachte Lernverfahren, die komplett ohne Trainingsdaten auskommen und häufig eine Form des Clusterings oder lexikalische Ressourcen verwenden. Unüberwachte und semiüberwachte Verfahren benötigen wenige oder kaum annotierte Trainingsdaten, erreichen jedoch bisher nicht die Präzision überwachter Ansätze (Mansouri u. a., 2008).

3.1.3. Ansätze

Nachfolgend werden nun einige der bekanntesten NER-Syteme vorgestellt und abschließend miteinander verglichen und anschließend erläutert, warum ein neuer Ansatz zur Annotierung der vorliegenden Archivdaten notwendig ist.

Illinois ist ein NER-Systme entwickelt von (Ratinov und Roth, 2009). Dieses System basiert auf einem Perceptron, also einem künstlichen neuronalem Netz. Illinois verwendet *greedy left-to-right decoding* statt des Viterbi-Algorithmus 2.3.1 und das BILOU-Schema siehe 4.1.2 zur Repräsentation von Text-Chunks. Der Ansatz verwendet neben lokalen Features auch globale Features sowie externes Wissen von nicht annotierten Daten und Gazetteers. Illionois ist nur für die englische Sprache verfügbar.

Stanford Core NLP (Manning u. a., 2014) ist ein Framework für verschiedene NLP-Aufgaben wie Tokenization, Sentence Splitting und *Part-of-speech Tagging*. Das NER-Tool verwendet neben einem CRF auch ein regelbasiertes System zur Erkennung von numerischen Entitäten. Neben der englische Sprache unterstützt Stanford durch die Erweiterung von Faruqui und Padó (2010) auch Deutsch. Neben Englisch und Deutsch werden auch Spanisch und Chinesisch unterstützt.

Weitere bekannte Frameworks für verschiedene NLP-Aufgaben sind OpenNLP und Balie. Neben reinen NER-Systemen werden auch zunehmend Systeme entwickelt, die gleichzeitig auch das Entity Linking behandeln (Abschnitt 3.2). Beispiele sind hier DBpedia Spotlight (Daiber u. a., 2013) oder FOX (Speck und Ngonga Ngomo, 2017).

Fox von Speck und Ngonga Ngomo (2017) ist ein NER-Framework, das auf Ensemble-Learning basiert. FOX nutzt dafür insgesamt fünf verschiedene NER-Ansätze: Stanford, Illinois, Balie, OpenNLP und Spotlight. Das Ensamle-Leraning funktioniert dabei wie folgt: Zunächst wird ein Text von allen NER-Tools gleichzeitig bearbeitet. Anschließend wird ein Perceptron verwendet, der jedem Token des Textes auf Grundlage der Ergebnisse aller verwendeten NER-Tools eine Klasse zuweist.

Für das EL verwendet FOX AGDISTIS, welches auch für diese Arbeit verwendet wird (siehe Kapitel 4.2.1).

Wie bereits beschrieben haben die Domäne und die Sprache, auf denen ein NER System angewendet und gelernt wird einen enormen Einfluss auf die Qualität. Eine Adaption auf eine neue Domäne ist dabei meist sehr schwierig.
Da es bislang keinen geeigneten Korpus für tabellarische Archivdaten gibt und tabellarische Archivdaten generell im NER-Bereich kaum erforscht sind, gibt es weder ein regelbasiertes NER-System noch ein Machine Learning basiertes NER-System, welches geeignet ist die vorliegenden Archivdaten zu annotieren.
Ziel dieser Arbeit ist es deshalb ein NER-System zu entwickeln, welches auf einem Trainingsdatensatz mit bereits annotierten Daten gelernt wird. Hierzu können im wesentlichen die Konzepte der NER-Systeme, welche auf überwachtem Lernen basieren übernommen werden. In Kapitel 4.1 wird ein NER-System beschrieben, welches auf einem CRF basiert, das auf Archivdaten trainiert wurde.

3.2. Entity Linking (EL)

Entity Linking wird häufig auch Named Entity Disambiguation (NED) genannt genannt und beschreibt das Verlinken von Entitäten zu einer Wissensbasis. Konkret soll allen gefunden Entitäten in einem Dokument genau ein Link zu einer äquivalenten Ressource aus einer Wissensbasis zugeordnet werden oder eine *NIL-Referenz*, falls die Wissensbasis keine äquivalente Ressource enthält (Dai u. a., 2012).
Zur besseren Unterscheidung von Entitäten, die in einem Text gefunden werden und Ressourcen aus einer Wissensbasis werden die in einem Text gefundenen Entitäten im Folgenden *Entity Mentions* genannt. EL besteht dabei nach Shen u. a. (2015) im wesentlichem aus vier Schritten:

1. Identifikation von *Entity Mentions* (NER)
2. Kandidaten Generierung
3. Linking
4. Identifikation von *Entity Mentions*, die nicht in der KB enthalten sind (*NIL-Mentions*)

Die Identifikation von *Entity Mentions* ist identisch mit dem NER-Problem wie in Kapitel 3.1 beschrieben. Die restlichen vier Schritte sollen im Folgenden nun näher erläutert werden.

3.2.1. Kandidaten Generierung

Für jede *Entity Mention*, welche durch ein NER-System gefunden wurde, muss zunächst eine Kandidatenmenge aus einer Wissensbasis generiert werden. Formal soll für jede *Entity Mention* m_i aus einem Dokument D die Menge $C_i = \{c_{i_1}, ...c_{i_k}\}$ der k besten Kandidaten berechnet werden.
Ein häufig verwendeter Ansatz sind Named Dictionaries (ND), bei welchen einem Label eine Menge relevanter Entitäten zugeordnet sind. Das Ziel ist es dabei eine Wissensbasis nach *Entity Mentions* durchsuchbar zu machen. (Shen u. a., 2015)
Eine Methode zur Generierung von NDs ist die Nutzung verschiedener Wikipedia Features. Wikipedia enthält neben Entity-Pages, welche Entitäten beschreiben, auch Redirect-Pages, auf welchen alternative Namen für eine Entität beschreiben werden, Disambiguation-Pages auf denen unterschiedliche Entitäten für einen Namen unterschieden werden und Hyperlinks deren Ankertexte oft Synonyme für Entitäten sind. (Shen u. a., 2015)
Neben Wikipedia werden häufig auch Ontologien wie Yago oder DBpedia verwendet, welche bereits Daten in strukturierter Form enthalten und so die Generierung eines NDs erleichtert. (Shen u. a., 2015) Das Suchen von Kandidaten erfolgt durch Vergleich des Textes der *Entity Mention* mit dem Label einer Entität aus dem Dictionary. Hierbei können verschiedene Distanzmaße wie die N-Gram-Distanz (Kondrak, 2005) zur Berechnung des Abstandes der beiden Strings verwendet werden.
Da *Entity Mentions* auch Akronyme oder nur Nachnamen von Personen sind, werden häufig zusätzlich Surface Form Expansion Techniken angewandt, um weitere mögliche Kandidaten zu finden.
Eine weitere Technik ist das Verwenden von Suchmaschinen. Beispielsweise kann durch die Google Api[1] nach möglichen Kandidaten gesucht werden. Die gefundenen Seiten werden anschließend gefiltert, sodass nur noch Wikipedia Ergebnisse als Kandidaten betrachtet werden. (Shen u. a., 2015)

1 https://developers.google.com/custom-search/json-api/v1/overview (abgerufen am 23.8.2018)

3.2.2. Linking

Aus der im zweiten Schritt generierten Kandidatenmenge muss für jede *Entity Mention* ein Kandidat ausgewählt werden, was häufig durch ein Ranking der Kandidaten geschieht. Hierzu gibt es sowohl überwachte als auch unüberwachte Verfahren. Des Weiteren werden je nach System auch Abhängigkeiten zwischen den *Entity Mentions* betrachtet.
Hier kann zwischen Systemen unterschieden werden, die *Entity Mentions* als unabhängig voneinander ansehen und Systemen, die Abhängigkeiten betrachten, sodass *Entity Mentions*, die ähnlich zueinander sind und im gleichen Kontext verwendet werden der selben Ressource zugeordnet werden können.
Linking-Verfahren verwenden sowohl kontextunabhängige Features, wie die Ähnlichkeit zwischen zwei Strings, als auch kontextabhängige Features zur Berechnung des Rankings. Ein kontextabhängige Features ist beispielsweise Bag-of-Words, bei dem der Kontext durch alle Wörter um eine Entität oder im gesamten Dokument beschrieben wird. (Shen u. a., 2015)
Die genannten Features werden sowohl von überwachten als auch von unüberwachten Verfahren verwendet (Shen u. a., 2015). Überwachte Verfahren greifen dabei auf eine große Menge von Trainingsdaten zurück. Ein einfacher Ansatz ist die Anwendung von binären Klassifikationsalgorithmen wie SVM, Naive Baies und k-Nearest-Neighbors, um zu klassifizieren, ob eine gefundene Entität zu einem Kandidaten passt. Ein Nachteil ist hier, dass die Trainingsdaten sehr schlecht ausbalanciert sind, da es mehr negative als positive Beispiele gibt. Gleichzeitig können auch mehrere Kandidaten als positiv klassifiziert werden, wodurch ein weiter Linking-Schritt notwendig wird. (Shen u. a., 2015)
Ein besseres Vorgehen ist hier das Lernen von Rankings. Ein Algorithmus lernt dabei anhand von Trainingsdaten, wie eine Menge von Kandidaten gerankt sein muss. Dies löst die Probleme der binären Klassifikation, da die Trainingsdaten so ausbalanciert sind und eindeutig ist, welcher Kandidat am besten passt (Shen u. a., 2015).
Neben den überwachten Ranking Methoden gibt es ebenfalls unüberwachte Methoden, welche ohne Trainingsdaten auskommen. Diese Modelle verwenden häufig ein Vektor-Space-Model mit dem Ziel ein Ranking der Kandidaten anhand ihrer Ähnlichkeit zu einer *Entity Mention* zu berechnen.
Andere Ansätze betrachten das Ranking Problem als ein Information Retrieval-Problem, bei dem jeder Kandidat als ein Dokument indiziert wird und *Entity Mentions* in Queries umgewandelt werden (Shen u. a., 2015).

Eine weitere Klasse von Ansätzen sind graphenbasierte Methoden. Hoffart u. a. (2011) verwenden beispielsweise einen Kohärenzgraphen und berechnen anschließend den dichtesten Teilgraphen, der für jede *Entity Mention* nur noch genau einen Knoten enthält. In Kapitel 4.2.1 wird eine weitere unüberwachte Methode beschrieben, die graphenbasiert ist und den HITS-Algorithmus verwendet.

3.2.3. *NIL-Mentions*

Da eine Wissensbasis nicht alle *Entity Mentions* eines Dokuments enthält, muss abschließend für jede *Entity Mention* beurteilt werden, ob ein Kandidat zugeordnet werden kann, oder ob eine spezielle NIL-Referenz zugeordnet werden soll. Dieser Schritt wird von einigen EL-Systemen weggelassen, da oft nur Entitäten annotiert werden, welche auch in einer Wissensbasis enthalten sind. Andere Methoden annotieren nur dann eine *NIL-Referenz*, wenn im Schritt der Kandidatengeneriereung keine Kandidaten gefunden wurden. (Shen u. a., 2015)
Allerdings ist es möglich, dass eine Wissensbasis keine passende Ressource für eine *Entity Mention* enthält und dennoch im Rahmen der Kandidatengenerierung einen Kandidaten findet. Aus diesem Grund verwenden einige Verfahren zusätzlich einen binären Klassifizierer, um zu entscheiden, ob der im Linking-Schritt beste gefundene Kandidat annotiert wird oder eine *NIL-Referenz* zurückgegeben wird. (Shen u. a., 2015)

3.2.4. Ansätze

In diesem Abschnitt werden einige bekannte EL-Ansätze beschrieben und miteinander verglichen. Abschließend werden die Nachteile dieser Ansätze für das Linking des Archivdatensatzes beschrieben.

AIDA (Hoffart u. a., 2011) verwendet vor YAGO als Wissensbasis und einen Kohärenzgraphen für das Linking. Die Kandidaten werden durch die *YAGO means-realation* generiert. Das Linking erfolgt durch schrittweises Entfernen von Knoten aus dem Kohärenzgraphen.

DBpedia Spotlight (Daiber u. a., 2013) kombiniert NER und EL. Als Wissensbasis wird ein verlinktes Wörterbuch verwendet, welches aus Wikipedia-Seiten generiert wird. Spotlight durchsucht zunächst ein Dokument nach Wikipedia-Artikeln. Zusätzlich können *Entity Mentions*, die keine Nomen enthalten ausgefiltert werden. Für die gefundenen *Entity Mentions* werden anschließend Kandidaten gesucht. Das Linking geschieht durch ein Vector-Space-Modell verwendet.

TagMe von Ferragina und Scaiella (2012) verwendet einen Lucene Index für Ankertexte aus Wikipedia und einen Index, welcher Wikipedia Seiten enthält. Zusätzlich wird ein Graph konstruiert, dessen Knoten die Wikipedia Seiten sind und dessen Kanten die Links zwischen den Wikipedia Seiten darstellen. Für das Linking wird zunächst mit Hilfe des Ankerindexes nach Vorkommen in einem Text gesucht. Durch das Linking werden dann mit Hilfe eines graphenbasierten Algorithmus allen gefundenen Vorkommen eine Wikipedia-Seite zugeordnet.

Der größte Teil aller EL-Ansätze verwenden Wikipedia als Wissensbasis. Zwei der beschriebenen Ansätze nutzen Ontologien wie Yago oder DBpedia zur Verlinkung von Entitäten. Das Ziel dieser Arbeit ist es jedoch die Archivdaten zur Gemeinsamen Norm Datei (GND) der deutschen Nationalbibliothek und nach Wikidata zu verlinken, da die Abdeckung der Entitäten hier größer ist als bei Wikipedia, DBpedia oder Yago. Des Weiteren soll der Ansatz in der Lage sein über mehrere Wissensbasen gleichzeitig zu verlinken um dabei die Vorteile beider Wissensbasen zu kombinieren. Zum Erreichen dieses Ziels wird der AGDISTIS-Ansatz von Usbeck u. a. (2014) erweitert.

4. Konzept

In diesem Kapitel soll das Konzept des entwickelten Ansatzes vorgestellt werden. Das Verfahren besteht grundsätzlich aus zwei Schritten. Der erste Schritt umfasst die Erkennung von Entitäten (NER). Im zweiten Schritt wird das Entity Linking durchgeführt (EL). Dementsprechend wird in diesem Kapitel zunächst das Vefahren für die *Named Entity Recognition* beschrieben, welches auf *Conditional Random Fields* basiert. Abschließend erfolgt die Beschreibung des Konzepts des Linking-Verfahrens, welches eine Erweiterung des AGDISTIS-Ansatzes verwendet. Hier werden im wesentlichen die notwendigen Anpassungen des Linking-Ansatzes im Vergleich zu AGDISTIS beschrieben.

4.1. Named Entity Recognition

Zur Erkennung der Entitäten wird das maschinelle Lernverfahren Conditional Random Field (CRF) eingesetzt. Es handelt sich hierbei um ein überwachtes Lernverfahren, welches Trainingsdaten benötigt, um ein Modell zu lernen, mit welchem dann Entitäten in Texten erkannt werden können. Nachfolgend wird zunächst der Traingsprozess des Modells beschrieben. Anschließend erfolgt die Beschreibung aller Funktionen des NER-Moduls.

4.1.1. Rasa NLU

Rasa NLU[1] ist ein Open Source Tool zur Extraktion von Entitäten und zur Absichtserkennung von Sätzen. Es wurde von der Rasa Technologies GmbH (2017) entwickelt. Das Tool stellt weitreichende Funktionalitäten zur Entwicklung von Chatbots und weiterer NLP Anwendungen bereit. Vor allem können CRFs zur Erkennung und Klassifizierung von Entitäten trainiert werden. Diese Arbeit verwendet die Implementierung von CRFs aus Rasa NLU, welche auf der CRF-Suite von Sklearn[2] basiert, sowie die für die Rasa Pipeline notwendigen Funktionalitäten. Die Erkennung von Entitäten ist hierbei ausreichend, die Klassifizierung der Entitäten wird nicht benötigt, da AGDISTIS für das Linking nur den gefundenen String, nicht aber die Klassifizierung berücksichtigt.

1 https://nlu.rasa.com/ (abgerufen am 23.8.2018)
2 https://sklearn-crfsuite.readthedocs.io/en/latest/ (abgerufen am 23.8.2018)

Rasa NLU bietet die Möglichkeit ein Modell zu trainieren, um es in einer Applikation zu verwenden. Für das Trainieren eines Modells werden eine Konfiguration und eine Menge von Trainingsdaten benötigt, welche nachfolgend vorgestellt werden.

4.1.2. Konfiguration

Rasa NLU verwendet eine JSON Konfigurationsdatei, in der festgelegt wird, welche Komponenten verwendet werden sollen. Anschließend ist die für diese Arbeit verwendete Konfiguration dargestellt.

```
{
  "pipeline":["nlp_spacy",
        "tokenizer_spacy",
        "ner_crf"],
  "language": "de",
  "num_threads": 1,
  "max_training_processes": 1,
  "max_number_of_ngrams": 7,
  "ner_crf": {
    "BILOU_flag": true,
    "features": [
      ["low", "title", "upper", "pos", "pos2"],
      ["bias", "low", "word3", "word2", "upper", "title", "digit", "pos", "pos2"],
      ["low", "title", "upper", "pos", "pos2"]],
    "max_iterations": 50,
    "L1_c": 1,
    "L2_c": 1e-3
  }
}
```

Auflistung 4.1: Rasa Konfiguration

Diese Konfiguration enthält zunächst die Pipeline, welche festlegt, in welcher Reihenfolge die verwendeten Komponenten ausgeführt werden. Die wichtigste Komponente ist neben dem CRF der Tokenizer, welcher aus einem String ein Tokenarray generiert. Diese Tokens werden als Beobachtungen x von den Feature-Funktionen f verwendet. Im Element *ner-crf* wird das CRF konfiguriert. Zunächst kann eingestellt werden, ob das *BILOU*-Schema verwendet werden soll. Das *BILOU* Schema wird von NER-Tools verwendet, um Entitäten im Text zu markieren.

BILOU ordnet dabei jedem Token ein Tag zu:

- Beginning: Starttoken einer Entität
- Inside: Token innerhalb einer Entität
- Last: Token am Ende einer Entität
- Outside: Token außerhalb einer Entität
- Unique: Einzelnes Token, das eine Entität darstellt

Wenn dieses Einstellung nicht verwendet wird, markiert Rasa NLU nur Entitäten, welche aus einem Token bestehen. Da Entitäten in den Archivdaten aber fast immer aus mehreren Tokens bestehen, wird das BILOU Schema benötigt. Ein anderes häufig verwendetes Schema ist das BIO-Schema, welches nur die Zustände *Beginning*, *Inside* und *Outside* verwendetet. Dieses wird von Rasa NLU jedoch nicht unterstützt.
Als nächstes werden die Features festgelegt, die das CRF verwendet soll. Dabei können für das betrachtete Wort selber, das vorherige Wort und das nachfolgende Wort Features hinzugefügt werden.
Folgende Features stehen zur Verfügung:

- *low*: Wort in Kleinbuchstaben,
- *title*: Boolean Wert, der angibt, ob ein Wort mit einem Großbuchstaben beginnt,
- *word3*: Letzten drei Zeichen des Wortes,
- *word2*: Letzten zwei Zeichen des Wortes,
- *pos*: Part of Speech Tag des Wortes
- *pos2*: Ersten beiden Zeichen des Part of Speech Tags,
- *bias*: Bezieht das Verhälnis der Anzahl aller Labels in den Traingsdaten mt ein,
- *upper*: Boolean Wert, der angibt, ob in ein Wort nur aus Großbuchstaben besteht und
- *digit*: Gibt an, ob es sich bei einem Wort um eine Zahl handelt

Abschließend kann noch die Zahl der maximalen Iterationen, sowie zwei Koeffizienten, die zur Regulierung benötigt werden, konfiguriert werden. Zum Training des CRFs wird der L-BFGS Algorithmus von Liu und Nocedal (1989) verwendet.

4.1.3. Trainieren des Modells

Ziel dieser Arbeit ist es Archivdatensätze zu annotieren, welche bereits in strukturierter Form vorliegen. Zum Generieren des Modells wurden 200 Datensätze einer Archivdatendatei annotiert, welche im CSV-Format vorliegt. Das Annotieren der Trainingsdaten umfasst nur das Markieren von Entitäten. Links zu den Wissensbasen werden an dieser Stelle noch nicht benötigt. Aus den annotierten Datensätzen kann ein Trainingsdatensatz zum Lernen eines Modells erstellt werden. Jeweils eine Zeile des Datensatzes wird hierbei als ein Dokument betrachtet. Die Entitäten wurden dazu in folgender Form annotiert:

<entity>gefundenen Entität</entity>

Auflistung 4.2 zeigt einen kleinen Ausschnitt aus der Datei mit Traingsdaten.

```
id,AnzAkte,AnzBilder,AnzVorstellungen,Auffuehrungssprache,Auffuehrungsstaette,Auffuehrungstitel...
54096,3,,6,D,<entity>Stadttheater</entity>,<entity>Der Freischütz. Romantische Oper in drei Auf...
3055,,,,,<entity>Teatro Kursaal</entity>,<entity>Luciano Sangiorgi. Unico concerto del pianista...
32773,,,,D,<entity>Stadttheater</entity>,<entity>Ware Lüge. Kabarettprogramm</entity>,,,,,STS L...
12179,3,,,D,,<entity>Die Fledermaus. Operette in drei Akten</entity>,,,,,STS Leiter Dokumentati...
53826,3,,,D,<entity>Stadttheater</entity>,<entity>Tannhäuser und der Sängerkrieg auf der Wartbu...
3065,,,,I,<entity>Teatro Kursaal</entity>,<entity>La locandiera</entity>,,,,,STS Leiter Dokumen...
52358,4,,1,D,<entity>Stadttheater</entity>,<entity>Die Braut von Messina oder Die feindliche Br...
57537,,,,F,<entity>Claque</entity>,<entity>Cérémonie pour un noir assassiné</entity>,,0,0,,STS ...
36382,,,,D,<entity>Sommertheater</entity>,<entity>Boeing-Boeing</entity>. Deutsch: <entity>Elis...
32224,,,,D,<entity>Stadttheater</entity>,GB,,,,,STS Leiter Dokumentation,04/09/2017,,* Hinweis ...
```

Auflistung 4.2: Trainingsdaten

Die annotierte CSV-Datei wird anschließend automatisch in das von Rasa NLU unterstütze JSON-Format übertragen. Da das Tool einen *Whitespacetokenizer* verwendet, führt die Struktur der CSV-Daten dazu, das Tokens nicht richtig erkannt werden, da insbesondere Spalten durch Kommas getrennt sind. Eine einfache Lösung hierfür ist das Ersetzen der Kommas durch Whitespaces. Hierbei ändert sich die Länge der Strings nicht, womit später auch keine Anpassung von Indizes notwendig ist. Das gleiche Verfahren kann auch später beim Anwenden des Modells verwendet werden.

Das JSON-Format besitzt folgende Struktur:

```
{
        "rasa_nlu_data": {
                "common_examples": [],
                "regex_features" : [],
                "entity_synonyms": []
        }
}
```

Auflistung 4.3: Rasa NLU JSON-Format für Trainingsdaten

Neben dem Array common_examples, aus denen das Modell berechnet wird, können Synonyme und reguläre Ausdrücke angegeben werden, mit denen Entitäten extrahiert werden können. Diese Arbeit verwendet aber ausschließlich ein aus Beispielen gelerntes Modell. Reguläre Ausdrücke und Synonyme werden nicht verwendet. Auflistung 4.4 zeigt beispielhaft die Struktur eines *common-example Elements*.

```
{
  "text": "show me chinese restaurants",
  "intent": "restaurant_search",
  "entities": [
    {
      "start": 8,
      "end": 15,
      "value": "chinese",
      "entity": "cuisine"
    }
  ]
}
```

Auflistung 4.4: Beispiel Auszeichnung von Entitäten Quelle: Rasa Technologies GmbH (2017)

Ein solches Element enthält zunächst den Text, auf den sich die Entitäten beziehen, und die Intention des Textes. Diese ist für das Erkennen von Entitäten nicht relevant und kann weggelassen werden. Danach folgt eine Menge von Entitäten, welche jeweils aus dem Start- und Endindex bestehen, an welcher die Entität im Text vorkommt. Zudem wird der Teilstring und die Art der Entität angegeben. Die Art der Entität wird

ausschließlich zur Klassifizierung verwendet und ist für die Erkennung von Entitäten nicht notwendig. Deshalb kann hier jeweils immer „entity“ eingetragen werden. Dies erleichtert die automatische Generierung des Trainingsdatensatzes.
Nach Umwandlung der Trainingsdaten in das zuvor beschriebene Format kann nun ein CRF-Modell trainiert werden. Das Modell kann anschließend im Dateisystem gespeichert werden, sodass es später beim Start der Anwendung nicht erneut trainiert werden muss.

4.2. Entity Linking

Für das Linking der im ersten Schritt gefundenen Entitäten wird eine Erweiterung des AGDISTIS-Ansatzes von Usbeck u. a. (2014) verwendet. AGDISTIS nutzt bislang ausschließlich DBpedia als Wissensbasis. Da in den betrachteten Archivdaten viele Entitäten existieren, die bisher nicht in DBpedia erfasst sind, muss AGDISTIS um weitere Wissensbasen erweitert werden. Dies erfordert folgende Schritte:

1. Die Generieren eines neuen Triple Indexes
2. Das Anpassen der Kandidatengenerierung, da AGDISTIS nur das Standard *rdfs:label* betrachtet, welches nicht von allen betrachteten Wissensbasen verwendet wird.
3. Die Verlinkung zu mehreren Wissensbasen, um die Vorteile verschiedener Wissensbasen zu kombinieren. Für das Linking werden die schon in Kapitel 2.2 genannten Wissensbasen Wikidata und die Gemeinsame Normdatei der deutschen Nationalbibliothek (GND) verwendet.

Im Folgenden wird zunächst AGDISTIS als System, sowie die Grundlagen des Ansatzes vorgestellt. Anschließend erfolgt die ausführliche Darstellungen aller notwendigen Änderungen und Erweiterungen für das Linking der betrachteten Archivdaten.

4.2.1. AGDISTIS

Als Eingabe verwendet das Verfahren ein Dokument D und eine Menge von vorab durch ein NER-Verfahren gefundenen Entitäten B. Wie die meisten Linkingansätze verwendet AGDISTIS zunächst ein Verfahren zur Generierung einer Kandidatenmenge C. Anschließend werden mit Hilfe eines graphenbasierten Verfahrens Kontextinformationen aus der zugrundeliegenden Wissensbasis extrahiert und zum Abschluss der *HITS*

Algorithmus (Kapitel 2.4) angewandt, um den besten Kandidaten k aus der Kandidatenmenge C auszuwählen. Als nächstes sollen diese drei Phasen im Detail vorgestellt werden.

Kandidatengenerierung

Die Aufgabe der Kandidatengenerierung ist es, eine geeignete Kandidatenmenge C in der zugrundeliegenden Wissensbasis K zu finden. Die Menge aller RDF-Triples T von K sind in einem Suchindex I enthalten, welcher auch die Labelmenge L aller Ressourcen umfasst. Zum Generieren der Kandidatenmenge C für eine Entität $b \in B$ wird aus dem String s von e eine Query $Q = (p, s)$ generiert mit p als festem *Prediacte*. p beschränkt sich hier auf das *rdfs:label*. Im allgemeinen können aber weitere Labels hinzugefügt werden, was bei der Verwendung unterschiedlichster Wissensbasen, wie in Abschnitt 4.2.3 beschrieben, von zentraler Bedeutung ist. Durch Anwendung der Query Q auf I wird nach einer initialen Kandidatenmenge C_0 gesucht. Da C_0 oftmals Kandidaten aus einer nicht untersuchten Domäne enthält oder Kandidaten deren Label $l \in L$ nicht exakt genug zum String s passen, werden Filter auf C_0 angewendet. Die übrigen Kandidaten bilden die Kandidatenmenge C.

Oftmals ist es nicht direkt möglich allein durch den String s direkt eine Kandidatenmenge C zu ermitteln. In diesen Fällen können alternative Schreibweisen (*surface forms*) von s betrachtet und Normalisierungs- und Expansionpolicies auf s angewendet werden, um die Suche nach Kandidaten zu verbessern. Normalisierung bezieht sich im westlichen auf die Entfernung von Prefixen und Affixen, sowie der Eliminierung von Plural und Genitiv Formen von s (Usbeck u. a., 2014). Die Expansion Policy beschäftigt sich mit der Auflösung von Koreferenzen zwischen einer Entität $b \in B$ und einer weiteren Entität $b' \in B$ innerhalb eines Dokuments D. Dies ist sowohl bei Webdokumenten als auch bei Archivdaten sehr wichtig, da hier Entitäten oft nur einmal mit ihrem vollen Namen genannt und später meist nur noch Kurzformen verwendet werden.

Berechnung der optimalen Ressource

Aus der zuvor generierten Kandidatenmenge C muss als nächstes der Disambiguierungsgraph G_d mit Tiefe d generiert werden. Das generierte Wissen wird dabei in einem *knowledge Graph* $G_K = (V, E)$ überführt, wobei die Knoten V die Ressourcen der Wissensbasis und die Kanten E die *Predicates* zwischen zwei Knoten sind. Dabei gilt für $x, y \in V, (x, y) \in E \Leftrightarrow \exists P : (x, p, y)$ ist ein RDF Triple in der Wissensbasis K. Im initialen Graphen $G_0 = (V_0, E_0)$ ist V_0 die Menge der Kandidaten C und E_0 die Menge der Kanten, welche zunächst leer ist. Die Kanten werden mit Hilfe der Breiten-

suche hinzugefügt, wobei die Expansion $p(G_i)$ des Graphen $G_i = (V_i, E_i)$ zum Graphen $G_{i+1} = (V_{i+1}, E_{i+1})$ in folgender Weise erfolgt:

$$V_{i+1} = V_i \cup \{y : \exists x \in V_i \wedge (x, y) \in E\}$$
$$E_{i+1} = \{(x, y) \in E : x, y \in V_{i+1}\}$$

Dieser Schritt wird exakt d mal ausgeführt, der resultierende Graph G_d hat somit die Tiefe d. Die Expansion des Graphen besteht somit darin in jedem Schritt alle Knoten hinzuzufügen, die durch eine Kante in der Wissensbasis mit einem bereits hinzugefügten Knoten im *Knowledge Graph* verbunden sind. Gleichzeitig werden auch Kanten zwischen bereits enthaltenen Knoten hinzugefügt. Im letzten Schritt erfolgt die Identifizierung des richtigen Kandidatenknotens durch den *HITS*-Algorithmus. Hierfür werden wie bereits in Kapitel 2.4.2 beschrieben *Authority-Values* x_a, y_a und *Hub-Values* x_h, y_h für alle $x, y \in V_d$ berechnet. Initial sind die Werte für die *Hub-* und *Authority-Values* wie folgt festgelegt:

$$\forall x \in V_d, x_a, y_a = \frac{1}{|V_d|}$$

Anschließend wird k mal folgende Berechnung ausgeführt:

$$x_a \longleftarrow \sum_{(y,x) \in E_d} y_h, \quad y_h \longleftarrow \sum_{(y,x) \in E_d} x_a$$

Nach Ausführung des HITS-Algorithmus wird aus der Menge der Kandidaten derjenige Knoten mit dem größten *Authority-Value* als Ergebnis des Linkings ausgewählt und zurückgegeben. Ein großer Vorteil dieses Verfahrens ist, dass nicht auf Data Mining Algorithmen mit nicht polynomieller Laufzeit zurückgegriffen wird. Die Zahl der Iterationen beim *HITS*-Algorithmus kann auf eine sehr geringe Zahl festgelegt werden, da der Algorithmus sehr schnell konvergiert. In der Praxis reichen hier meist 20 Iterationen (Usbeck u. a., 2014).

4.2.2. Indexgenerierug

Wie zuvor in Kapitel 4.2.1 beschrieben, werden die Wissensbasen in einen Suchindex überführt, welcher aus einer Menge aus RDF-Triples besteht. Für den Index wird das Suchframework Lucene[3] der Apache Software Foundation aus dem Jahr 2001 verwendet. Dieser Suchindex muss zunächst aus einer Datei erstellt werden, die im Turtle Format

3 https://lucene.apache.org/core/ (abgerufen am 23.8.2018)

vorliegen muss. Der entstehende Triple Index besteht aus folgenden Feldern:

- *subject*: URI der Ressource
- *predicate*: URL einer Eigenschaft
- *objectstring*/*objecturi*: Label oder URL zu einer Referenz

Der bisherige Ansatz verwendet einen Suchindex, der meist ausschließlich aus DBpedia Daten erstellt wurde.
Da DBpedia allerdings nur einen geringen Teil der in den Archivdaten vorhandenen Entitäten enthält, ist es notwendig weitere Wissensbasis zum Suchindex hinzuzufügen. Hier sind die Wissensbasis der deutschen Nationalbibliothek und Wikidata besser geeignet, da insbesondere die GND auf die betrachteten Daten spezialisiert ist. Auch Wikidata enthält einen großen Teil der betrachteten Entitäten, zusammen mit der GND kann somit ein großer Teil der betrachteten Entitäten abgedeckt werden. Diese Aufgabe ist allerdings nicht trivial, da Wissensbasen Referenzen untereinander aufweisen und es somit mehrere Knoten für eine Ressource gibt, die zu einem Knoten zusammengefasst werden müssen (siehe Abbildung 4.1).

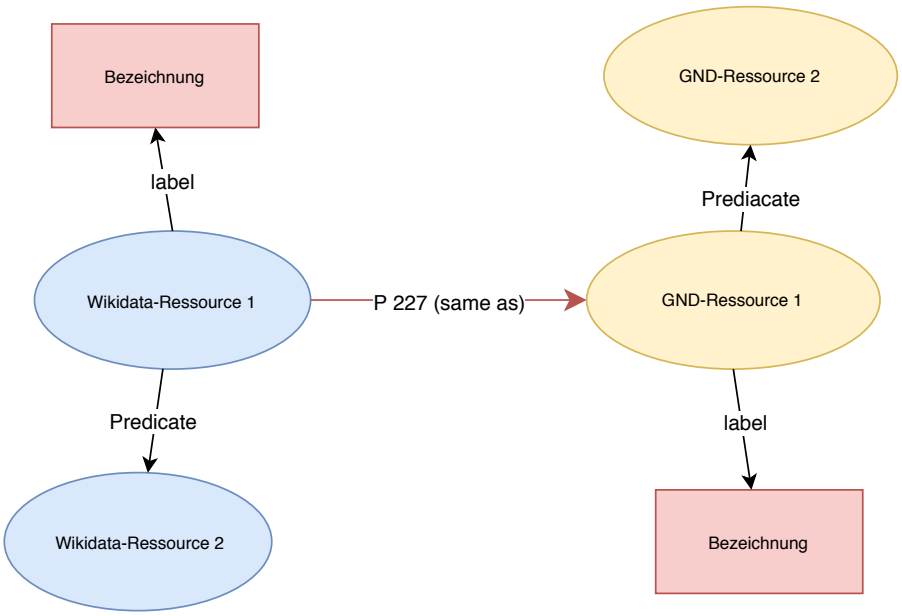

Abbildung 4.1.: *same as* Links zwischen Knoten aus unterschiedlichen Wissensbasen

Eine Lösung dieses Problems wird in Abschnitt 4.2.4 beschrieben.
In diesem Abschnitt sollen aber zunächst die notwendigen Anpassungen für die Indizierung der GND und Wikidata als einzelne Wissensbasen beschrieben werden.

GND Indexierung

In der GND der deutschen Nationalbibliothek werden häufig RDF-Blank-Nodes zur Angabe weiterer Eigenschaften verwendet. Das Problem ist, dass diese beim Generieren des Indexes bisher nicht aufgelöst werden und somit die relevanten Informationen nicht direkt der URL zugeordnet werden können. Auflistung 4.5 zeigt einen kurzen Ausschnitt der Turtle-Repräsentation für den Datensatz von Johann Wolfgang von Goethe. Am Ende des ersten Blocks wird hier eine Blank-Node-Referenz festgelegt, welche später dann stellvertretend für die eigentliche URL verwendet wird. Diese Blank-Node-Referenzen müssen zur Generierung des GND Index aufgelöst werden, indem sie durch die URL der Ressource ersetzt werden. Die semantische Aussage bleibt dabei gleich.

```
<http://d-nb.info/gnd/118540238> a gndo:DifferentiatedPerson ;
        gndo:gndIdentifier "118540238" ;
        foaf:page <https://de.wikipedia.org/wiki/Johann_Wolfgang_von_Goethe> ;
        owl:sameAs <http://dbpedia.org/resource/Johann_Wolfgang_von_Goethe> , <http://viaf.org/viaf/24602065> ;
        gndo:oldAuthorityNumber "(DE-588)1131918517" ;
        owl:sameAs <http://d-nb.info/gnd/1131918517> ;
        dnbt:deprecatedUri "http://d-nb.info/gnd/1131918517" ;
        gndo:oldAuthorityNumber "(DE-588)1095607278" ;
        owl:sameAs <http://d-nb.info/gnd/1095607278> ;
        dnbt:deprecatedUri "http://d-nb.info/gnd/1095607278" ;
        gndo:oldAuthorityNumber "(DE-588)1022736213" ;
        owl:sameAs <http://d-nb.info/gnd/1022736213> ;
        dnbt:deprecatedUri "http://d-nb.info/gnd/1022736213" ;
        gndo:oldAuthorityNumber "(DE-588)1032060956" ;
        owl:sameAs <http://d-nb.info/gnd/1032060956> ;
        dnbt:deprecatedUri "http://d-nb.info/gnd/1032060956" ;
        gndo:oldAuthorityNumber "(DE-588c)4021455-2" , "(DE-588)1014927390" ;
        owl:sameAs <http://d-nb.info/gnd/1014927390> ;
        dnbt:deprecatedUri "http://d-nb.info/gnd/1014927390" ;
        gndo:oldAuthorityNumber "(DE-588a)1014927390" , "(DE-588)101488358X" ;
        owl:sameAs <http://d-nb.info/gnd/101488358X> ;
        dnbt:deprecatedUri "http://d-nb.info/gnd/101488358X" ;
        gndo:oldAuthorityNumber "(DE-588a)101488358X" , "(DE-588)185848826" ;
        owl:sameAs <http://d-nb.info/gnd/185848826> ;
        dnbt:deprecatedUri "http://d-nb.info/gnd/185848826" ;
        gndo:oldAuthorityNumber "(DE-588a)185848826" , "(DE-588)185808069" ;
        owl:sameAs <http://d-nb.info/gnd/185808069> ;
        dnbt:deprecatedUri "http://d-nb.info/gnd/185808069" ;
        gndo:oldAuthorityNumber "(DE-588a)185808069" , "(DE-588a)118540238" , "(DE-588a)1014123208" ;
        gndo:relatedWork <http://d-nb.info/gnd/1085150313> , <http://d-nb.info/gnd/1085154025> ;
        gndo:variantNameForThePerson "Goethe, Johann Wolfgang" ;
        gndo:variantNameEntityForThePerson _:node1ci79at7qx17938733 .

_:node1ci79at7qx17938733 gndo:forename "Johann Wolfgang" ;
        gndo:surname "Goethe" .

<http://d-nb.info/gnd/118540238> gndo:variantNameForThePerson "Goethe, Johan Wolfgang von" ;
        gndo:variantNameEntityForThePerson _:node1ci79at7qx17938734 .

_:node1ci79at7qx17938734 gndo:forename "Johan Wolfgang" ;
        gndo:prefix "von" ;
        gndo:surname "Goethe" .
```

Auflistung 4.5: GND Ausschnitt

Da Blank-Nodes immer zunächst in einem Triple definiert werden und somit das *Object* des Triples sind und sie später ausschließlich als *Subject* verwendet werden, können sie auf einfache Weise beim Generieren des Indexes durch folgende Vorverarbeitung in

URLs umgewandelt werden:

Algorithm 3: Blank Node Matching

```
blankNodeMatcher = (blankNode,URL);
for triple ∈ KB do
    if triple.object is blankNode then
        blankNodeMatcher.put(triple.object,triple.subject);
    end
    if triple.subject is blankNode then
        triple.subject = blankNodeMatcher.get(triple.subject);
    end
end
```

Wikidata Indexierung

Für den Suchindex wird wie zuvor beschrieben das Suchframework Lucene verwendet, welches Int32 Integer Variablen zur Indizierung von Dokumenten nutzt, womit die Zahl der Dokumente, die indiziert werden können, auf genau 2.147.483.647 Dokumente beschränkt ist. Da der Turtle RDF Dump von Wikidata aber deutlich mehr Triples enthält, müssen einige nicht benötigte Triples entfernt werden. Als erste Maßnahme ist hier die Beschränkung auf sogenannte *Truthy Statements* (Wikimedia Foundation, 2018) sinnvoll, welche direkte Links zwischen Entitäten verwenden. Sonst würden auch Triples aufgenommen werden, für welche über Blank-Nodes weitere Eigenschaften wie Wikipediaseiten verlinkt, die aber nicht benötigt werden. Diese Informationen werden weder für den HITS Algorithmus noch für die Kandidatengenerierung benötigt und verbessern nicht das Gesamtergebnis. Auflistung 4.6 zeigt ein Beispiel solcher nicht gewünschten Triples.

```
wds:Q42-F078E5B3-F9A8-480E-B7AC-D97778CBBEF9 a wikibase:Statement,
                wikibase:BestRank ;
        wikibase:rank wikibase:NormalRank ;
        ps:P31 wd:Q5 ;
        prov:wasDerivedFrom wdref:2b369d0a4f1d4b801e734fe84a0b217e13dd2930,
                wdref:55d23126bca9913374faf69ba8fbd21474a74421 .

wd:Q42 p:P21 wds:q42-39F4DE4F-C277-449C-9F99-512350971B5B .

wds:q42-39F4DE4F-C277-449C-9F99-512350971B5B a wikibase:Statement,
                wikibase:BestRank ;
        wikibase:rank wikibase:NormalRank ;
        ps:P21 wd:Q6581097 ;
        prov:wasDerivedFrom wdref:2b369d0a4f1d4b801e734fe84a0b217e13dd2930,
                wdref:a02f3a77ddd343e6b88be25696b055f5131c3d64,
                wdref:55d23126bca9913374faf69ba8fbd21474a74421 .

wd:Q42 p:P106 wds:Q42-e0f736bd-4711-c43b-9277-af1e9b2fb85f .
```

```
wds:Q42-e0f736bd-4711-c43b-9277-af1e9b2fb85f a wikibase:Statement,
                wikibase:BestRank ;
        wikibase:rank wikibase:NormalRank ;
        ps:P106 wd:Q214917 .

wd:Q42 p:P106 wds:q42-E13E619F-63EF-4B72-99D9-7A45C7C6AD34 .
```

Auflistung 4.6: Wikidata Ausschnitt

Dies reicht allerdings noch nicht aus, was eine weitere Reduzierung der Triples notwendig macht. Eine weite sinnvolle Maßnahme ist es deshalb zusätzlich nicht benötigte Sprachen zu entfernen. Die Sprachen sind dabei jeweils als Eigenschaft des *Objects* angegeben und können so leicht identifiziert werden.

```
wd:Q42 a wikibase:Item ;
        rdfs:label "Douglas Adams"@fr ;
        skos:prefLabel "Douglas Adams"@fr ;
        schema:name "Douglas Adams"@fr ;
        rdfs:label "Douglas Adams"@pl ;
        skos:prefLabel "Douglas Adams"@pl ;
        schema:name "Douglas Adams"@pl ;
        rdfs:label "Douglas Adams"@it ;
        skos:prefLabel "Douglas Adams"@it ;
        schema:name "Douglas Adams"@it ;
        rdfs:label "Douglas Adams"@en-gb ;
        skos:prefLabel "Douglas Adams"@en-gb ;
        schema:name "Douglas Adams"@en-gb ;
        rdfs:label "Douglas Adams"@nb ;
        skos:prefLabel "Douglas Adams"@nb ;
        schema:name "Douglas Adams"@nb ;
        rdfs:label "Douglas Adams"@es ;
        skos:prefLabel "Douglas Adams"@es ;
        schema:name "Douglas Adams"@es ;
        rdfs:label "Douglas Adams"@en-ca ;
        skos:prefLabel "Douglas Adams"@en-ca ;
        schema:name "Douglas Adams"@en-ca ;
        rdfs:label "Douglas Adams"@hr ;
        skos:prefLabel "Douglas Adams"@hr ;
        schema:name "Douglas Adams"@hr ;
        rdfs:label "Douglas Adams"@pt ;
        skos:prefLabel "Douglas Adams"@pt ;
```

Auflistung 4.7: Wikidata Sprachen

Der generierte Index beschränkt sich auf die Sprachen Deutsch, Englisch und Französisch, da dies auch die in den Archivdaten verwendetet Sprachen sind.

4.2.3. Generierung von Kandidaten

Für die Verlinkung der Archivdaten müssen vor allem bei der Kandidatengenerierung einige Anpassungen vorgenommen werden. Das liegt zum Einen daran, dass für die Suche nach GND Triplen weitere *Predicates* durchsucht werden müssen, siehe Kapitel 4.2.3 und zum Anderen daran, dass speziell die Namen von Aufführungsstätten und Aufführungstitel in den betrachteten Archivdaten oft in einer abstrakten Form vorliegen, welche sich stark von der in der Wissensbasis angegebenen Bezeichnungen unterscheidet und somit im Rahmen der Kandidatengenerierung viele gute Kandidaten

ausgefiltert werden. Die Kapitel 4.2.3, 4.2.3 und 4.2.3 beschreiben Lösungsansätze für diese Probleme.

Generierung von Kandidaten aus der GND

Die Suche nach möglichen Kandidaten in einem Triple Index erfolgt über eine Query, bei der die Felder *predicate* und *objectstring* gegeben sind. Das *Object* ist gegeben durch die *Entity Mention* im Text, für die ein Triple gesucht wird. Das *Predicate* ist das *rdfs:label*, welches der Standard für eine Bezeichnung von Ressourcen in den meisten Wissensbasen ist. Ziel ist es also die URIs aller zu der Query passender Ressourcen zu finden.
Ein Problem entsteht allerdings, wenn eine Wissensbasis nicht das Standard *rdfs:label* sondern eigene *Predicates* für die Bezeichnung von Items verwendet. In diesem Fall findet AGDISTIS keine Kandidaten.
Ein Beispiel für eine solche Wissensbasis ist die GND der deutschen Nationalbibliothek. Aus diesem Grund muss die Kandidatensuche um eine Liste von *Predicates* erweitert werden. Die GND verwendet unterschiedliche *Predicates* für verschiedene Arten von Entitäten. Folgende Entitäten für Personen, Organisationen und Werktitel sind relevant:

http://d-nb.info/standards/elementset/gnd#preferredNameForThePerson
http://d-nb.info/standards/elementset/gnd#preferredNameForTheCorporateBody
http://d-nb.info/standards/elementset/gnd#variantNameForTheWork

Das Vorgehen zum Finden von Kandidaten mit Hilfe der GND Eigenschaften ist dann analog zur Suche mit dem Standard *rdfs-Label*. Für alle verwendeten *Predicates* wird jeweils eine Query generiert, mit der im Triple Index nach Kandidaten gesucht wird. Die Liste mit den zu betrachtenden *Predicates* kann in der Konfigurationsdatei von AGDISTIS hinterlegt werden. Neben dieser Liste muss der GND-Präfix zu den erlaubten Knoten und Kantentypen hinzugefügt werden, da AGDISTIS diese Kandidatentriples sonst ausfiltern würde.

Domain-White-Listener

Der Domain-White-Listener dient in AGDISTIS dazu, Kandidaten-URIs auf eine abgeschlossene Menge von Ressourcentypen zu beschränken. Dazu wird das *rdf:type Predicate* verwendet, indem für jede Kandidaten URL der entsprechende Typ im Index gesucht wird. Erlaubte Typen sind dafür in einer White-List definiert. Gehört ein Kandidat nicht

zu einem in der White-List definierten Typen, wird der Kandidat ausgefiltert. Wikidata verwendet allerdings die eigene Eigenschaft *instance of* (*P31*) zum Beschreiben des Typs. Um den Domain-White-Listener für Wikidata zu nutzen, muss dieser angepasst werden, sodass auch anhand des *P31-Predicates* nach Typen gesucht wird. Folgende Typen werden für diese Arbeit verwendet:

- Aufführungsorte

  http://d-nb.info/standards/elementset/gnd#CorporateBody

  http://d-nb.info/standards/elementset/gnd#Country

  http://www.wikidata.org/entity/Q153562 (opera house)

  http://www.wikidata.org/entity/Q24354 (theater)

- Aufführungs- und Werktitel

  http://d-nb.info/standards/elementset/gnd#MusicalCorporateBody

  http://d-nb.info/standards/elementset/gnd#MusicalWork

  http://d-nb.info/standards/elementset/gnd#Work

  http://purl.org/ontology/bibo/Document

  http://www.wikidata.org/entity/Q25379 (play)

  http://www.wikidata.org/entity/Q1344 (opera)

  http://www.wikidata.org/entity/Q1132324 (gloss)

  http://www.wikidata.org/entity/Q7725634 (literary work)

- Personen

  http://d-nb.info/standards/elementset/gnd#DifferentiatedPerson

  http://www.wikidata.org/entity/Q5 (human)

Diese Liste von Entitäten umfasst im wesentlichen alle Typen, der in den Archivdaten vorkommenden Entitäten. Diese können darüber hinaus in beiden betrachteten Wissensbasen verwendet werden, um Entity-Spezifische Programmfeatures zu entwickeln (siehe Kapitel 4.2.3)

Suche durch Entity-Spezifische Features

Einige Entitäten wie Namen von Aufführungsstätten (*CH:Zürich:Stadttheater*) oder Aufführungstitel wie *Der Bockerer. Tragische Posse* enthalten zusätzliche Tokens wie erweiterte Beschreibungen oder Suffixe, durch die Entitäten im Datensatz näher definiert werden. Das Problem ist allerdings, dass dies häufig dazu führt, dass keine oder keine passenden Kandidaten gefunden werden können, da die Labels der gesuchten Ressourcen diese Informationen nicht enthalten und somit nicht ähnlich zu den gesuchten Strings sind. Die Erweiterungen sind je nach Art der Entität unterschiedlich. Deshalb ist es für diese Arbeit am sinnvollsten, je nach Art der Entität, ein spezielles Feature zu entwickeln. Zwar nutzt AGDISTIS nur die Strings der Entitäten als Eingabe, dennoch können mit Hilfe der zuvor im Domain-White-Listener verwendeten Methode, Kandidaten je nach Entitätstyp gefiltert werden.
Dazu wird zunächst das entsprechende Feature auf einen String angewendet und die Kandidatenmenge C_0 generiert. Auf C_0 wird anschließend ein Filter angewendet, der ebenfalls wie der Domain-White-Listener die *Predicates rdf:type* und *P31* verwendet, jedoch nur die Typen akzeptiert, die zum Entitätstyp, für welches das Feature entwickelt wurde passen.
Im Rahmen dieser Arbeit werden speziell für die Aufführungs- und Werktitel, sowie für Aufführungsorte neue Features benötigt. Diese werden nun im folgenden im Detail vorgestellt.

Feature für Aufführungs- und Werkstitel

Zum Finden einer Kandidatenmenge C_0 von Aufführungstiteln, werden N-Grams aus dem Suchstring s gebildet, nach denen nacheinander im Index gesucht wird. N-Grams sind Teilstrings, die genau n Elemente eine Strings s enthalten. Elemente können auf unterschiedliche Weise definiert werden. In diesem Fall entspricht ein Element einem Token des Strings s.
Für einen String s wird die Menge M aller möglichen Teilstrings bestehend aus n, $n-1, n-2, ...$ und 2 Tokens gebildet. Enthält ein String mehr als fünf Tokens werden auch alle einzelnen Tokens von s (Unigrams) zu M hinzugefügt. Dies ermöglicht das Suchen nach Aufführungstiteln, wie *Rigoletto. Oper in drei Akten*. Hier besteht der eigentliche Titel nur aus einem Token. Die restlichen Tokens enthalten lediglich weitere Informationen.
Nach der Generierung von M wird der Index so lange nach Teilstrings t aus M durchsucht, bis mindestens ein Kandidat im Index gefunden wurde. Dies hat den Grund, dass

die Kandidatenmenge möglichst klein gehalten werden soll, um später das Linking zu beschleunigen.
Die Reihenfolge, nach der im Index I nach den Teilstrings t gesucht wird hat somit eine zentrale Bedeutung. Eine Beobachtung im Datensatz ist es, dass vor allem bei Aufführungstiteln der eigentliche Titel am Anfang steht. Deshalb wird die Liste zunächst aufsteigend nach den Indizes des ersten enthaltenen Tokens $t_0 \in t$ geordnet. Das zweite Kriterium der Sortierung ist die Menge der enthaltenen Token $m = |t|$. Hier wird absteigend sortiert. Für den String *Der Bockerer. Tragische Posse* entsteht somit folgende Reihenfolge:

```
Der Bockerer. Tragische Posse
Der Bockerer. Tragische
Der Bockerer.
Der
Bockerer. Tragische Posse
Bockerer. Tragische
Bockerer.
Tragische Posse
Tragische
```

Im Rahmen der Kandidatengenerierung wird beim obigen Beispiel bereits beim dritten Versuch mit dem Teilstring *Der Bockerer* ein passendes Ergebnis gefunden.

Feature für Aufführungsorte

Aufführungsorte liegen oft in einer der Formen *CH: Zürich: Stadttheater* oder *Zürich: Stadttheater: Stadttheater* vor. Das Länderkürzel ist dabei häufig ein Problem, da dies nicht in der Bezeichnung einer Ressource verwendet wird. Des weiteren führen doppelte Tokens wie *Stadttheater* oft dazu, dass viele falsche Kandidaten gefunden werden, da sie beim neu definierten Distanzmaß (siehe Kapitel 4.2.3) doppelt gewichtet würden. Aus diesem Grund werden alle Doppelpunkte aus dem String entfernt, dies erleichtert das Finden gleicher Tokens. Anschließend werden aus dem resultierenden String alle Tokens mit weniger als zwei Zeichen und alle doppelten Tokens entfernt.
Der resultierende String ist in beiden Fällen *Zürich Stadttheater*. Dieses Schema lässt sich auf einen großen Teil aller in den Archivdaten vorhandenen Aufführungsstätten übertragen und führt so zu einer deutlichen Verbesserung des Evaluierungsergebnisses für Aufführungsorte (Kapitel 6.4.5).

Dynamisches Filtern der Kandidaten

AGDISTIS verwendet zum Filtern der Kandidaten einen festen Filterwert (*Thershold_trigram*). Da aber mit der Länge eines Strings die Wahrscheinlichkeit für Abweichungen zunimmt, ist es sinnvoll den Filterwert je nach Länge des Strings herabzusetzen. Die Länge des Strings ist dabei die Anzahl der enthaltenen Token des Strings. Ein einfaches Modell ist dabei die lineare Reduzierung: Sei n die Anzahl der Tokens im Text. Der Filterwert F wird dann wie folgt festgelegt:

$$F = 1 - 0.15 * n$$

Das lineare Modell gibt dabei nicht die tatsächliche Abnahme der Wahrscheinlichkeit einen String der Länge n im Index zu finden wieder. Ein besseres Modell kann beispielsweise aus Trainingsdaten gelernt werden. Für diese Arbeit soll das lineare Modell verwendet werden, da es in der Praxis bereits zu einer signifikanten Verbesserung der Ergebnisse beiträgt.

Distanzmaß

AGDISTIS nutzt die N-Gram-Distanz aus dem Lucene-Framework zum Vergleich des Labels einer Ressource und einer im Text ausgezeichneter Entität. Damit eine Ressource in die Menge der Kandidaten aufgenommen wird, muss eine Distanz größer als $0,81$ erreicht werden. Die N-Gram-Distanz D im Lucene-Framework basiert auf der N-Gram-Distanz von Kondrak (2005) D_n und es gilt: $D = 1 - D_n$. Im folgenden wird zunächst schrittweise die N-Gram-Distanz D_n eingeführt und der Algorithmus zur Berechnung der Distanz wird vorgestellt. Anschließend wird eine alternative Distanz vorgestellt, welche den Vergleich von Strings, die aus mehr als einem Wort bestehen ermöglicht. Kondrak (2005) definiert zunächst die Editierdistanz für Unigrams zwischen den Strings X und Y wie folgt:

$$d(x, \epsilon) = 1$$

$$d(\epsilon, y) = 1$$

$$d(x, y) = \begin{cases} 0 & \text{für } x = y \\ 1 & sonst \end{cases}$$

$$d(X, Y) = d(\tau_{k,l}) = min(d(\tau_{k-1,l}) + 1, d(\tau_{k,l-1}) + 1, d(\tau_{k-1,l-1}) + d(x_k, y_l))$$

Dabei ist $\tau_{k,l} = (x_1, ..., x_k, y_1, ..., y_k)$ ein Paar von Präfixen zwischen den zu vergleichenden Strings X und Y.
Die Editierdistanz für N-Grams ist wie folgt definert. Für Unigrams (d_1) entspricht die Definition der zuvor vorgestellten Editierdistanz für Unigrams.

$$d_n(X,Y) = d_n(\tau_{k,l}) = min(d_n(\tau_{k-1,l}) + 1, d_n(\tau_{k,l-1}) + 1, d_n(\tau_{k-1,l-1}) + d_n(\tau^n_{k-n,l-n}))$$

mit

$$d_n(\tau^n_{i,j}) = \frac{1}{n}\sum_{u=1}^{n} d_1(x_{i+u}, y_{j+u})$$

Da Symbole am Anfang eines Strings mit der definierten N-Gram Distanz weniger oft in N-Grams enthalten ist, wird ein Affix der Länge $n-1$ zu beiden zu vergleichenden Strings hinzugefügt, damit alle Symbole gleich oft betrachtet werden. Zudem wird eine Normalisierung durchgeführt, indem der Quotient aus der berechnete Distanz und der Länge des längeren der beiden Strings berechnet wird. Die Distanz kann durch dynamische Programmierung wie in Algorithmus 4 beschrieben berechnet werden.

Algorithm 4: N-DIST(X,Y)

```
K ⟵ length(X);
L ⟵ length(Y);
for u ⟵ 1 to N − 1 do
    X ⟵ x'_1 + X;
    Y ⟵ y'_1 + Y;
end
for i ⟵ 0 to K do
    D[i, 0] ⟵ i
end
for j ⟵ 0 to L do
    D[0, j] ⟵ j
end
for i ⟵ 1 to K do
    for j ⟵ 1 to L do
        D[i, j] ⟵ min(D[i−1, j]−1, D[i, j−1], D[i−1, j−1]+d_N(τ^n i − 1, j − 1));
    end
end
return D[K, L]
```

Im betrachteten Datensatz sind häufig Entitäten enthalten, die mehr als ein Wort umfassen. In diesem Zusammenhang ist es sinnvoll Vertauschungen von ganzen Wörtern in das Distanzmaß mit einzubeziehen. Beispielsweise ist für die beiden Strings

Basel Stadtheater und **Stadtheater Basel**

die $Distanz = 0,28$, obwohl eindeutig die gleiche Entität beschrieben wird. Aus diesem Grund ist für diese Anwendung eine Erweiterung der N-Gram-Distanz sinnvoll, die das Vertauschen von Wörtern erlaubt. Diese Erweiterung funktioniert auch alternativ mit allen anderen Distanzmaßen. Zur Berechnung der Distanz zwischen den beiden Strings X und Y werden die Strings zunächst jeweils in die Tokenarrays $X_T = x_1, ..., x_n$ und $Y_T = y_1, ..., y_m$ aufgeteilt. Hierfür können je nach Anwendung unterschiedliche Tokenizer verwendet werden. In dieser Anwendung wird ein *Whitespace Tokenizer* angewandt. Zur Berechnung der Distanz wird anschließend für jedes Token aus X_T die N-Gram-Distanz zu jedem Token aus Y_T berechnet und die maximalen Distanzen D aller Tokens aus X_T werden in SUM_X aufsummiert. Anschließend wird das gleiche Verfahren für das Tokenarray Y_T angewandt um die Summe SUM_Y zu berechnen. Die Distanz zwischen den beiden Strings wird dann durch folgende Formel berechnet:

$$d(x, y) = \frac{\frac{SUM_X}{length(X_T)} + \frac{SUM_Y}{length(Y_T)}}{2}$$

Algorithmus 5 beschreibt das Verfahren zur Berechnung der Distanz zwischen den Strings X und Y: Für die zuvor beschriebenen Strings ist hier die Distanz 1,0. Für Strings, die nur aus einem Wort bestehen ist die Distanz identisch zur zuvor beschriebenen N-Gram-Distanz.

4.2.4. Mehrere Wissensbasen

Im folgenden soll das Linking über mehrere Wissensbasen beschrieben werden. Die besondere Herausforderung ist hier, dass Ressourcen in Wissensbasen häufig Referenzen zu Ressourcen in anderen Wissensbasen enthalten. Diese Referenzen werden auch *same as-edges* genannt. Ressourcen zweier Wissensbasen, die durch eine solche *same as-edge* verbunden sind, können als identisch angesehen werden. Die in dieser Arbeit betrachtete *GND* enthält keine dieser *same as-edges* zur Wikidata. *Wikidata* hingegen besitzt *same as-edges* zur *GND*. Da diese Kanten zwischen den beiden Wissensbasen existieren, ist die Erstellung eines gemeinsamen Indexes nicht ohne weiteres möglich, da für

Algorithm 5: DISTANCE(X,Y)

```
X_T ⟵ tokenize(X);
Y_T ⟵ tokenize(Y);
SUM_X ⟵ 0;
SUM_Y ⟵ 0;
LX_T ⟵ length(X_T);
LY_T ⟵ length(Y_T);
for x ⟵ 0 to LX_T do
    maxdistance ⟵ 0;
    for y ⟵ 0 to LY_T do
        distance = N-Gram-Distance(X_T[x],Y_T[y]);
        if distance > maxdistance then
            maxdistance=distance;
        end
        SUM_X=SUM_X+maxdisatnce
    end
end
for y ⟵ 0 to LY_T do
    maxdistance ⟵ 0;
    for x ⟵ 0 to LX_T do
        distance = N-Gram-Distance(Y_T[x],X_T[y]);
        if distance > maxdistance then
            maxdistance=distance;
        end
        SUM_Y=SUM_X+maxdistance;
    end
end
return (SUM_X/LX_T + SUM_Y/LY_T)/2
```

gleiche Ressourcen jeweils zwei Knoten im Wissensgraphen angelegt würden. Um dies zu verhindern gibt es zwei verschiedene Möglichkeiten, die im folgenden vorgestellt werden. Die erste Möglichkeit ist die Verwendung von je einer AGDISTIS Instanz für jede betrachtete Wissensbasen. Die zweite ist die Erstellung eines gemeinsamen Suchindex für beide Wissensbasen.

Verwendung von zwei AGDISTIS Instanzen

Die erste Variante des Ansatzes verwendet je eine AGDISTIS Instanz für beide Wissensbasen. Abbildung 4.2 zeigt den grundlegenden Aufbau des Ansatzes.

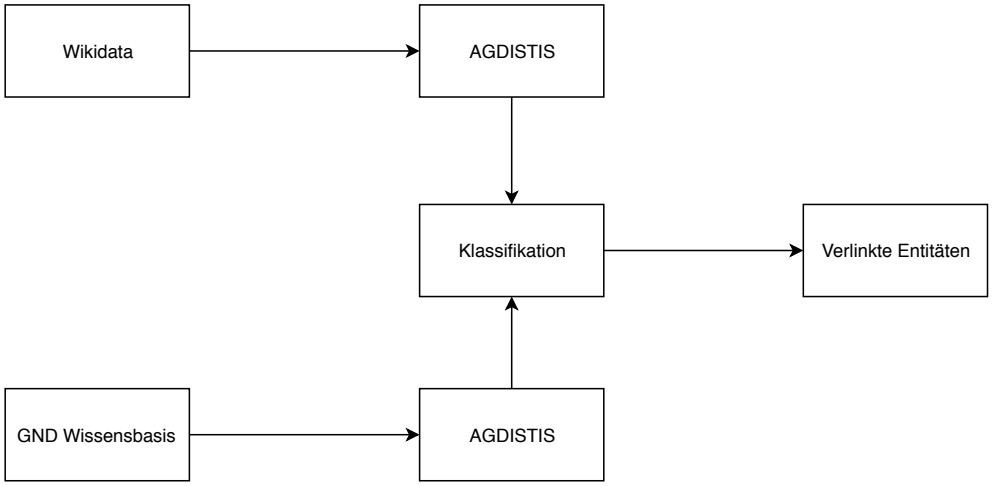

Abbildung 4.2.: Zwei AGDISTIS Instanzen

Dabei wird jede Entität gleichzeitig von beiden Instanzen klassifiziert. In diesem Fall sind vier Fälle möglich:

1. Die beiden Ressourcen sind identisch
2. Beide Instanzen liefern eine Ressource, aber die gefundene Ressource in Wikidata besitzt keinen *same as* Link
3. Nur eine der beiden Instanzen liefert eine Ressource
4. Die beiden Ressourcen sind unterschiedlich

Im ersten und zweiten Fall sind keine weiteren Schritte mehr notwendig. Im dritten Fall gibt es theoretisch die Möglichkeit, dass nur in der Wikidata Instanz eine Ressource gefunden wird. In diesem Fall ist es möglich, dass auch ein *same as-link* zur *GND* existiert. Falls dieser existiert, wird dieser ebenfalls hinzugefügt. Im vierten Fall muss entschieden werden, welches die richtige Ressource ist. Dieses Problem kann durch binäre Klassifikation gelöst werden. Algorithmus 6 beschreibt die Verarbeitung der Ergebnisse der beiden Instanzen:
Das Problem dieses Ansatzes ist, dass für die binäre Klassifikation eine große Menge an Trainingsdaten benötigt wird. Trainingsdaten können jedoch nur gesammelt werden, indem eine Große Menge von Entitäten durch beide Wissensbasen verlinkt werden, und Entitäten, bei denen die Instanzen unterschiedliche Ressourcen annotieren, gesammelt und manuell klassifiziert werden. Dieser Prozess ist sehr langwierig und kann im Rahmen dieser Arbeit nicht mehr umgesetzt werden.
Allerdings ist es möglich die maximale und minimale Performance des Ansatzes anhand

Algorithm 6: classify($X_{wikidata}$,Y_{gnd},$index_{wikidata}$

```
if X_wikidata = Y_gnd then
  | return X_wikdata, Y_gnd
end
else if not x_wikdata.has-same_as() then
  | return X_wikdata, Y_gnd
end
else if Y_gnd =  null and not X_wikidata =null then
  | if x_wikdata.has-same_as() then
  |   | return X_wikidata,index_wikidata.get-gnd(X_wikidata)
  | end
  | else
  |   | return X_wikdata,nulll
  | end
end
else if not Y_gnd = null and X_wikidata = null then
  | return null,Y_gnd
end
else if Y_gnd != X_wikidata then
  | label = classify(entity);
  | if label = Y_gnd then
  |   | return index_wikidata.get-wikidata(Y_gnd),Y_gnd
  | end
  | else
  |   | return X_wikidata,index.get-gnd(X,wikidata)
  | end
end
return null,null
```

der Performances beider AGDISTIS Instanzen zu berechnen. Dies wird in Kapitel 6.4.4 beschrieben.

Generierung eines gemeinsamen Index

Der zweite Ansatz ist die Generierung eines gemeinsamen Index. Hierzu muss das Problem gelöst werden, dass für eine Ressource in einem kombinierten Index mehrere URLs vorhanden sein können und damit mehrere Knoten für die Ressource im Index möglich sind. Abbildung 4.4 zeigt eine graphische Darstellung des Problems. Die beiden Ressourcen beschreiben beide das Theater Basel und sind durch Wikidata durch die Eigenschaft *P: 227* verknüpft. Bei der Indexierung würden beide Knoten als separate Ressourcen indexiert, was dazu führt, dass sie bei der Linking als konkurrierende Kandidaten betrachtet werden.

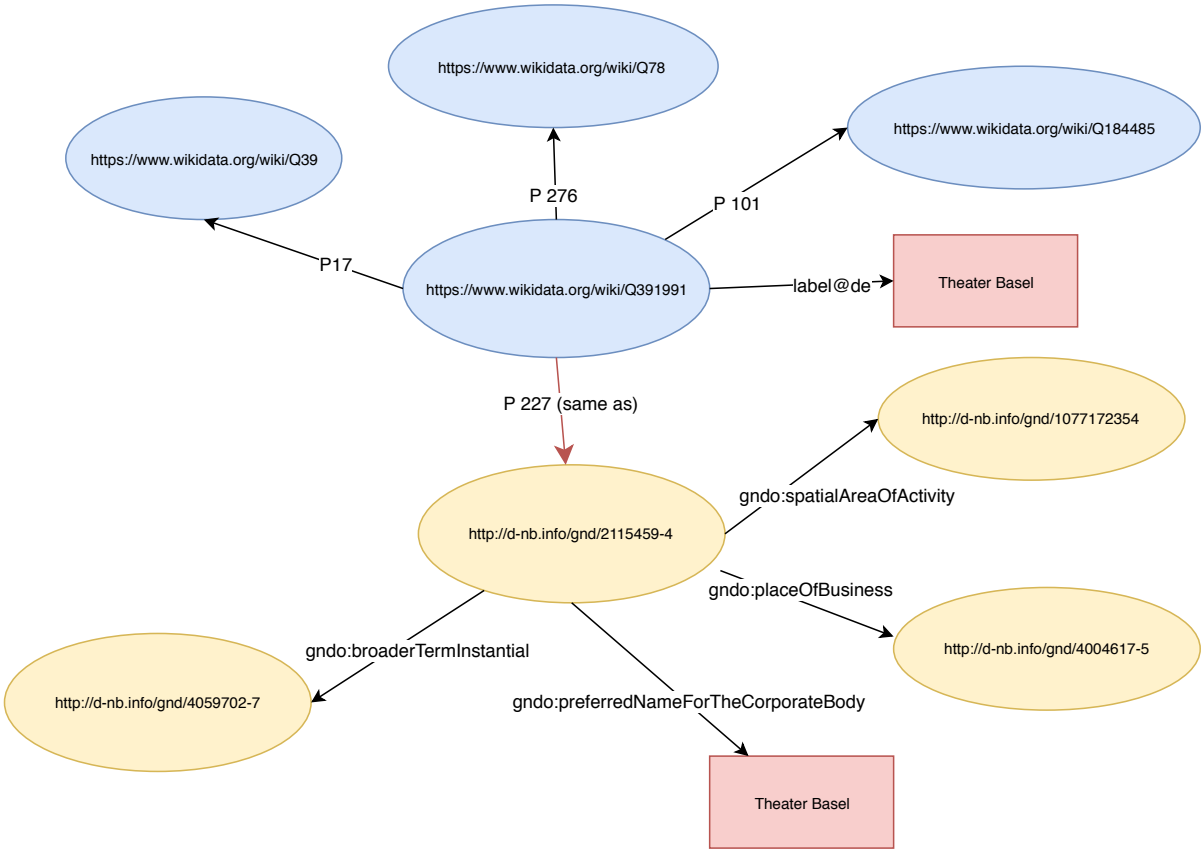

Abbildung 4.3.: Gemeinsamer Index mit zwei Knoten für eine Ressource

Das Ziel ist es nun die beiden Knoten zusammenzuführen, sodass alle aus- und eingehenden Kanten in einem Knoten zusammengefasst werden. Gleichzeitig soll aber auch die *same as-Edge* erhalten bleiben, damit später beide URLs ausgegeben werden können. Das Resultat für das Beispiel in Abbildung 4.3 ist in Abbildung 4.4 dargestellt.
Zur Generierung dieses Indexes wird zunächst der *Wikidata Index* indiziert und gleichzeitig werden die *same as-Edges* zwischen den *Wikidata* Ressourcen und den *GND* Ressourcen in einer Hashmap $H : GND \longrightarrow Wikidata$ gespeichert. Anschließend wird der GND Index indiziert. Für jede URL U wird dabei in H nach einer Ressource in Wikidata gesucht. Falls keine Ressource existiert, wird die *GND* URL beibehalten. Die zuvor beschriebenen Verfahren für das Blank-Node-Matching auf den Triples der GND und die Reduzierung der Menge der Triples für den Wikidata Index müssen auch hier angewendet werden Zur Ausgabe beider URLs kann wieder der Index genutzt werden, indem für eine Ressource R mit der Eigenschaft $P227$ die entsprechende *GND* URL gesucht wird. Algorithmus 7 versanschaulicht das Verfahren zur Generierung des kombinierten Indexes.

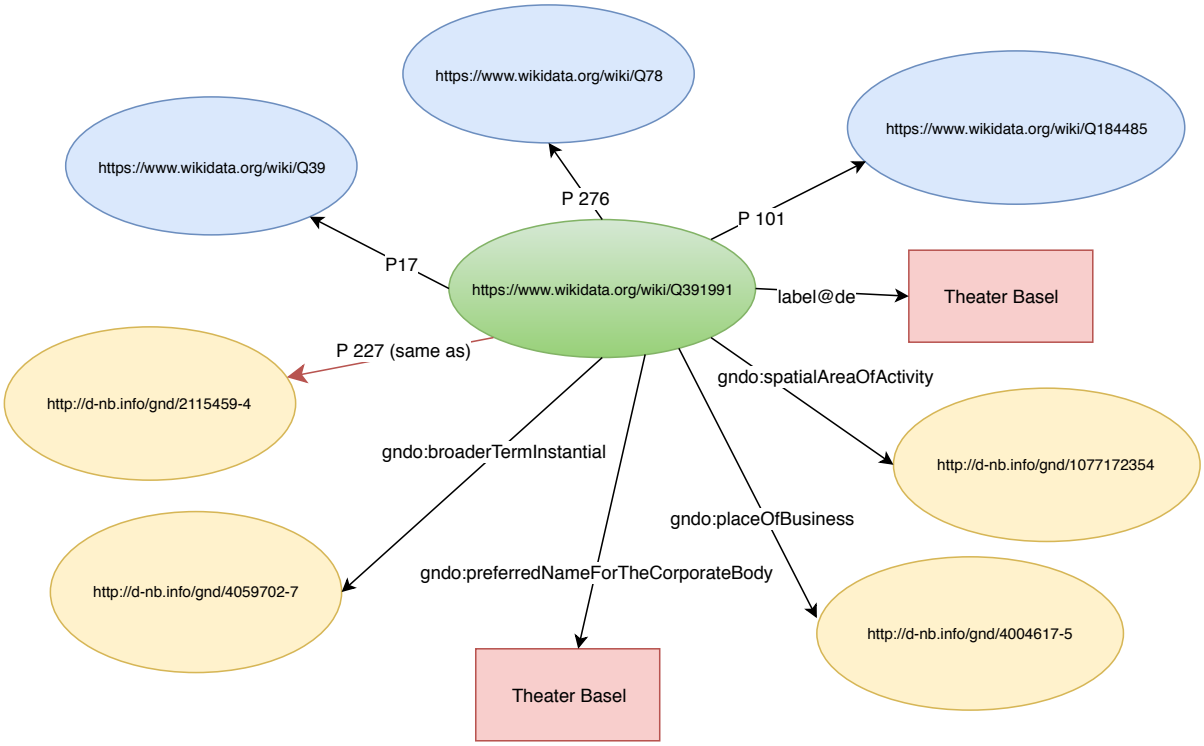

Abbildung 4.4.: Gemeinsamer Index mit zusammengefassten Knoten

Zur Kandidatengenerierung ist es abschließend noch notwendig, dass die Knoten und Kantentypen beider Wissensbasen zu den erlaubten Knotentypen in der Konfiguration hinzugefügt werden. Die Kandidatatengenerierung und das Linking durch den HITS-Algorithmus benötigen dagegen keine weiteren Anpassungen.

Algorithm 7: Combine Index ($KB_{Wikidata}$, KB_{GND})

```
index = new Index;
nodeMatcher = (URL_GND,URL_Wikidata);
for triple ∈ KB_Wikidata do
    if shouldIndex(triple) then
        if triple.predicate is P227 then
            nodeMatcher.put(triple.object,triple.subject);
        end
        index.add(triple);
    end
end
blankNodeMatcher = (blankNode,URL);
for triple ∈ KB_GND do
    if nodeMatcher.contains(triple.subject) then
        triple.subject=nodeMatcher.get(subject)
    end
    if triple.object is blankNode then
        blankNodeMatcher.put(triple.object,triple.subject);
    end
    if triple.subject is blankNode then
        triple.subject = blankNodeMatcher.get(triple.subject);
    end
    index.add(triple);
end
return index;
```

5. Systementwicklung

Das entwickelte System zur Annotierung von Archivdaten besteht im wesentlichen aus zwei Hauptkomponenten:

1. CRF-Modell
2. AGDISTIS

Das CRF-Modell erhält als Eingabe einen nicht annotierten String und identifiziert Entitäten in diesem String. Das grundlegende Konzept wurde bereits in Kapitel 4.1 beschrieben. AGDISTIS erhält als Eingabe die Ausgabe des CRFs und berechnet für jede gefundene Entität die am besten passende Ressource aus einer oder zwei Wissensbasen (siehe Kapitel 4.2). Der Ablauf zur Annotierung eines Datensatzes ist in Abbildung 5.1 dargestellt.

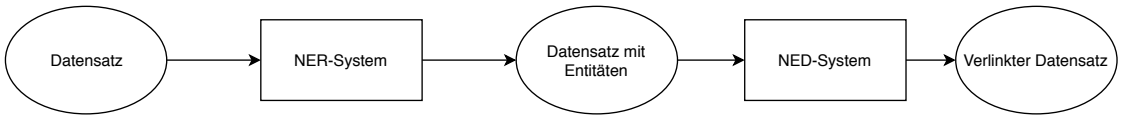

Abbildung 5.1.: Ablauf

Diese beiden Hauptmodule werden im wesentlichen für zwei Anwendungen benötigt. Zum einen wird ein Webservice implementiert, welcher einen Eingabestring erhält, für diesen Eingabestring Entitäten und Links ermittelt und die verlinkten Entitäten zurücksendet. Zum anderen soll es möglich sein ganze Dateien zu annotieren, dies wird durch ein einfaches Script umgesetzt. Das Ziel bei beiden Anwendungen ist die Generierung eines NIF-Dokuments (siehe Kapitel 2.1.2).

5.1. Repositories

Das entwickelte System teilt sich in zwei Projekte: Das erste Projekt ist AGDISTIS, für welches im Rahmen dieser Arbeit einige neue Features entwickelt wurden, um insbesondere die Verwendung weiterer Wissensbasen und die Annotierung der Archivdaten

zu ermöglichen. AGDISTIS ist in einem öffentlichen Repository verfügbar[1]. Die Implementierung ist bereits gut dokumentiert.
Das zweite Projekt ist *CRF-AGDISTIS*, welches die CRF-Implementierung von Rasa NLU mit AGDISTIS kombiniert. Auch für dieses Projekt ist ein öffentliches Repository verfügbar[2]. In diesem Kapitel soll auf einige Implementierungsdetails dieses Projekts eingegangen werden.

5.2. Webservice

Die Webservice-Komponente wurde mit dem Python Framework Flask[3] von Pallets Team (2010) implementiert, welches eine einfache API für das Design von Webservices bereitstellt. Der Webservice stellt für eine aufrufende Anwendung folgende Funktionen zur Verfügung:

1. *nifEntityRecognition*: Markieren von Entitäten
2. *nifa2kb*: Markieren und Linking von Entitäten

Es handelt sich hierbei um *HTTP POST* Methoden, welche als Eingabe ein NIF-Dokument erwarten. Zusätzlich zu diesen beiden Methoden wurde ein einfaches Webinterface implementiert, welches die Funktion der beiden Webservices demonstriert. Dieses Interface ist in Abbildung 5.2 dargestellt. Der Nutzer kann hier angeben, ob für einen Eingabetext nur Entity Recognition, ober auch das Linking ausgeführt werden soll. Als Ergebnis wird ein NIF-Dokument mit allen Entitäten ausgegeben.
Als nächstes soll auf die Interaktion zwischen dem Webservice und den beiden Hauptkomponenten eingegangen werden. Abbildung 5.3 zeigt die Schnittstellen zwischen dem Webservice und den beiden Hauptkomponenten und die beiden bereitgestellten Webservices. Das CRF-Model basiert auf dem Python Framework Rasa NLU. Da der Webservice ebenfalls in Python implementiert ist kann die Python API von Rasa NLU verwendet werden. Die Kommunikation mit AGDISTIS funktioniert dagegen über einen Webservice, welcher ebenfalls das NIF-Format verwendet.
Die Methode *nifEntityRecognition* dient dazu, dass ausschließlich das CRF-Modell verwendet wird ohne ein anschließendes Linking durchzuführen. Die Methode *nifa2kb* führt sowohl die Entity Recognition als auch das Linking durch. Abbildung 5.4 beschreibt den Ablauf eines Aufrufs für die Methode *nifa2kb*. Die Methode *nifa2kb* erhält dabei als Ein-

1 `https://github.com/dice-group/AGDISTIS` (abgerufen am 23.8.2018)
2 `https://github.com/vdanielupb/CRF-AGDISTIS` (abgerufen am 23.8.2018)
3 http://flask.pocoo.org/docs/1.0/license/ (abgerufen am 23.8.2018)

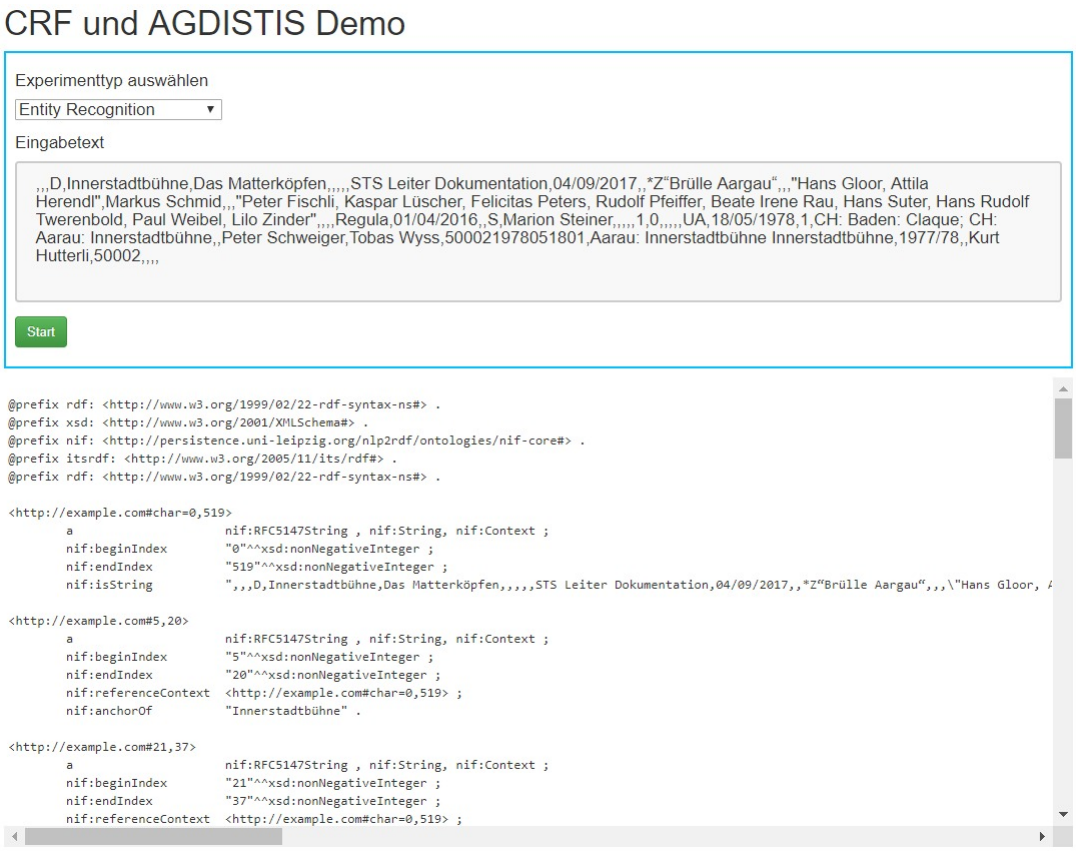

Abbildung 5.2.: System Demo

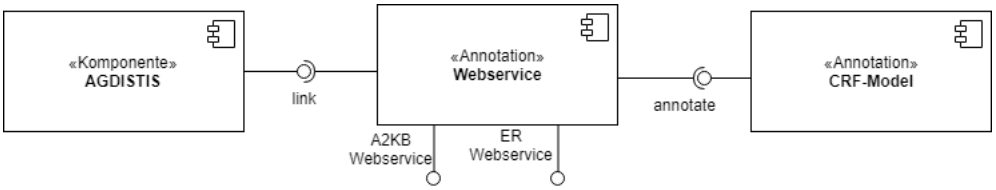

Abbildung 5.3.: Komponentendiagramm

gabeparameter einen String im NIF-Format, welcher den zu annotierenden Text enthält. Da das CRF-Modell nicht das NIF-Format unterstützt, muss der zu verarbeitende Text zunächst aus dem NIF-String extrahiert werden. Das CRF annotiert dann durch die Methode *parse* den Text und gibt die Annotationen zurück.
Diese werden anschließend wieder in ein NIF-Dokument umgewandelt.
Dieses NIF-Dokument wird anschließend per Webservice an AGDISTIS geschickt. Nach der Ausführung des Linkings schickt AGDISTIS einen NIF-String mit allen URLs zurück. Dieser String muss nicht weiter bearbeitet werden und kann vom Webservice direkt an den Client zurückgesendet werden. Die Methode *nifNamedEntityRecognition*

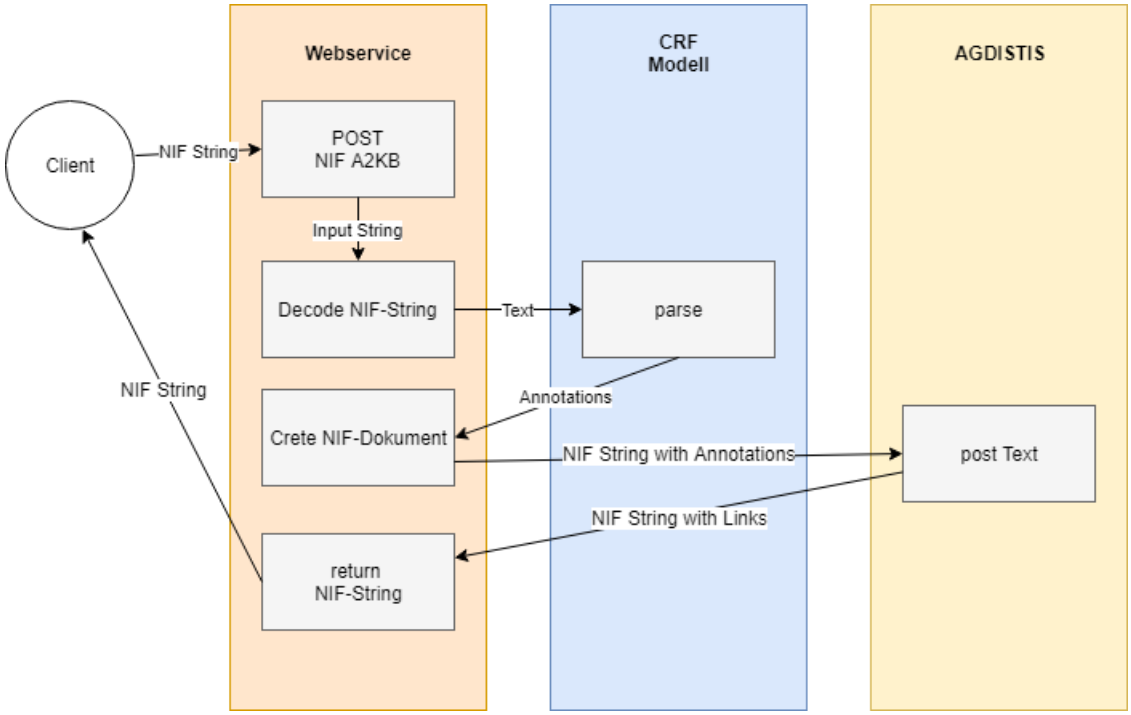

Abbildung 5.4.: NIF-Webservice *a2kb*

funktioniert analog, jedoch wird das Dokument nach dem Schritt *Create NIF-Document* direkt an den Client zurückgesendet, da kein Linking erforderlich ist.

5.3. File Annotator

Neben dem Webservice-Modul wurde ein Script zur Annotierung ganzer Dateien implementiert. Die Fallbeispieldatei liegt im CSV-Format vor. Das Ablaufdiagramm ist in Abbildung 5.5 abgebildet. Die CSV Datei wird dabei zeilenweise annotiert, da jede Zeile ein Dokument repräsentiert. Ziel ist auch hier ein NIF-Dokuemt zu generieren, dass die Annotationen aller Zeilen enthält. Für jede Zeile wird ein eigenes NIF-Dokument generiert, welches anschließend in das spätere Gesamtdokument integriert wird. Nach Bearbeitung aller Zeilen wird das generierte NIF-Dokument in einer Datei gespeichert.

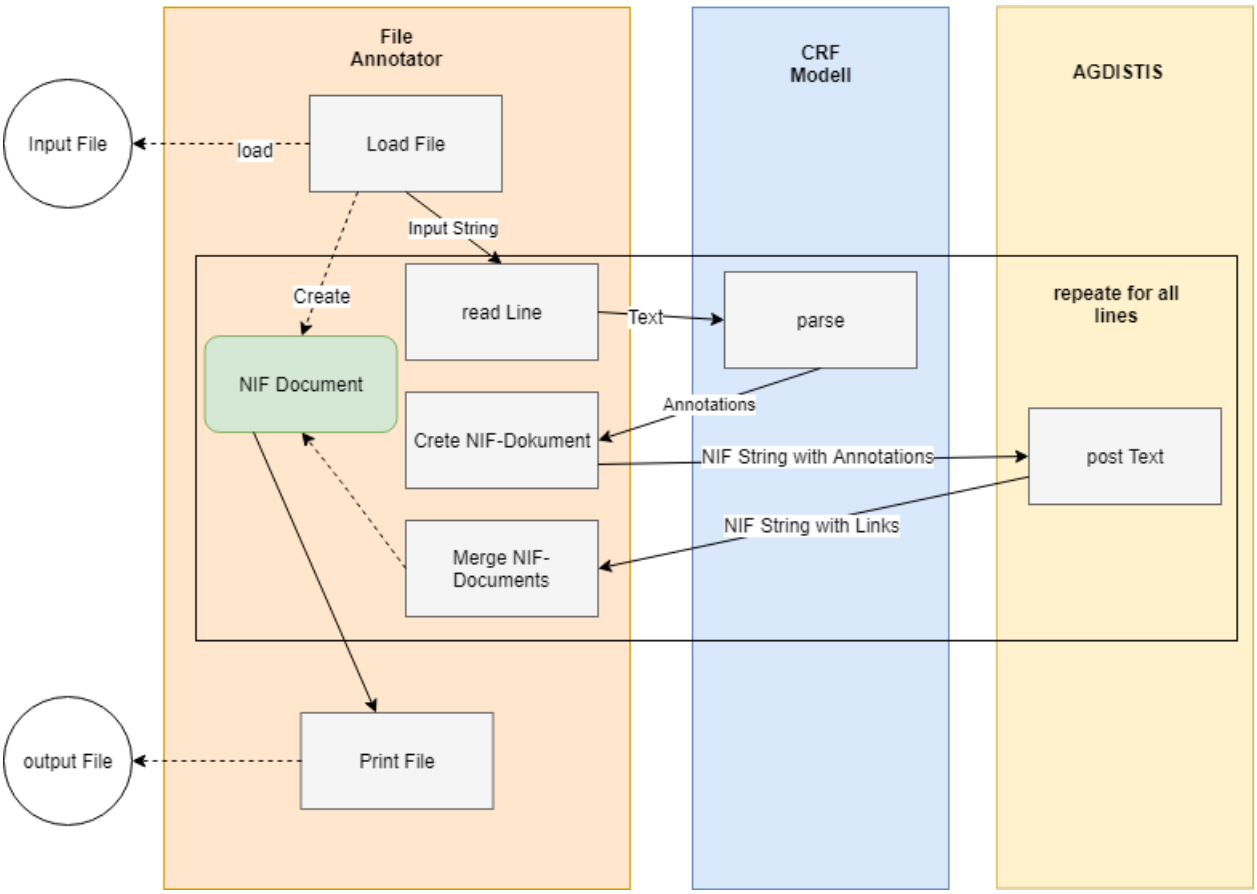

Abbildung 5.5.: Verarbeitung einer Datei durch den File Annotator

5.4. Trainieren eines Modells

Beide Anwendungen benötigen ein trainiertes CRF Modell. Zum Trainieren des Modells wird wieder ein eigenes Script verwendet. Der Ablauf wurde bereits in Kapitel 4.1.3 beschrieben. Dabei ist es möglich mehrere Modelle zu trainieren und je nach Anwendung kann eines der trainierten Modell ausgewählt werden. Dies erhöht die Wiederverwendbarkeit des Ansatzes. Für die Qualität der Annotierung ist ein gutes Vorhersagemodell essentiell.

5.5. Konfiguration

Das System bietet verschiedene Einstellmöglichkeiten, welche in einer Konfigurationsdatei hinterlegt werden. Die Struktur wird in Listing 5.1 dargestellt.

```
[rasa]
rasa_config_file = conf/conf.json
rasa_model_directory = model/
rasa_model = model/default/example_model/
training_file = data/example.csv
[flask]
host=127.0.0.1

[agdistis]
agdistis_url = http://localhost:8080/AGDISTIS

[files]
file_to_annotate = data/example.csv
file_to_write = out/nif_test.ttl
```

Auflistung 5.1: Konfiguration des Gesamtsystems

Der erste Block enthält Einstellungen, die für das CRF benötigt werden. Hier muss die Rasa-Konfiguration hinterlegt werden (siehe Kapitel 4.1.2). Des Weiteren kann ein Verzeichnis angegeben werden, in dem neu trainierte Modelle gespeichert werden. Die Einstellung *rasa_model_directory* gibt an, in welchem Verzeichnis das für den File-Annotator und den Webservice zu verwendende Modell gespeichert ist. Abschließend kann angegeben werden, welche Trainingsdatei verwendet werden soll, um ein neues Modell zu trainieren. Als nächstes wird im Block Flask hinterlegt, von welcher URL der Webservice erreichbar sein soll und im nächsten Block AGDISTIS wird die URL des AGDISTIS Webservice konfiguriert. Die Einstellungen im letzten Block werden für den File-Annotator benötigt, hier werden eine Eingabe- und Ausgabedatei benötigt.

6. Evaluation

In diesem Kapitel soll das entwickelte System anhand der weit verbreiteten Evaluationsmaße Precsion, Recall und F1-Measure evaluiert und soweit möglich mit anderen Ansätzen verglichen werden. Dazu wird zunächst das für die Evaluierung verwendete Framework GERBIL vorgestellt. Anschließend erfolgt die Beschreibung des für die Evaluation generierten Goldstandards. Abschließend werden die Ergebnisse der Evaluierung im Detail erläutert.

6.1. GERBIL

GERBIL ist eine Plattform für die Durchführung von Annotierungsexperimenten von Röder u. a. (2017). Der Nutzer hat durch Verwendeung der Software die Möglichkeit Experimente verschiedener Art zu konfigurieren und auszuführen. Neben bereits vorab integrierten Tools wie FOX oder Dbpedia Spotlight kann der Nutzer eigene Annotatoren einbinden, die über einen standardisierten Webservice mit GERBIL kommunizieren können. Des Weiteren können eigene Datensätze hochgeladen werden, welche im NIF Format vorliegen müssen. Im folgenden werden nun die Experimenttypen und Evaluationsmaße beschrieben, welche in GERBIL verwendet werden.

6.1.1. Experimenttypen

GERBIL stellt eine Vielzahl verschiedener Experimenttypen zur Verfügung(Röder u. a., 2017). Die für diese Arbeit relevanten Typen werden im folgenden kurz vorgestellt.

- A2KB: Beim A2KB-Experiment bekommt der Annotator einen Text ohne jegliche Annotationen und soll anschließend sowohl Entitäten identifizieren als auch ein Linking zu einer Wissensbasis durchführen. Gleichzeitg kann eine künstliche URL für Entitäten angegeben werden, die nicht in der Wissensbasis enthalten sind.
- D2KB: Der Annotator erhält einen Text mit bereits markierten Entitäten und fügt zu allen Entitäten Links zu einer oder mehreren Wissensbasen hinzu. Auch hier kann eine künstliche URL verwendet werden, um Entitäten zu markieren, die nicht in der Wissensbasis enthalten sind.

- Entity Recognition: Der Annotator erhält einen Text ohne Annotationen und markiert Entitäten im Text ohne Verlinkung zu einer Wissensbasis.

Neben diesen Experimenttypen, gibt es zudem weitere Experimente, bei denen die Beziehungen zwischen Entitäten zu finden oder Typen zu den Entitäten hinzuzufügen sind.
Für diese Arbeit werden ausschließlich die Experimenttypen Entity Recognition zur Evaluierung des CRF-Modells welches für die Entitäten Erkennung verwendet wird, D2KB zur Evaluierung des erweiterten AGDISTIS-Ansatzes und A2KB zur Evaluierung der Kombination der beiden Teilmodule verwendet.

6.1.2. Evaluationsmaße

Für die Evaluierung der Experimente nutzt GERBIL die bekannten Maße Recall Precision und F1-Measure (Röder u. a., 2017), welche ursprünglich aus dem Bereich Information Retrieval stammen (Van Rijsbergen, 1979). Nachfolgend werden diese drei Maße sowohl für NER als auch für das Linking eingeführt.

Named Entity Recognition

Bei der Ausführung eines NER-Experiments können für einen Teilstring beliebiger Länge eines Dokuments folgende für die Evaluierung relevante Ereignisse auftreten:

- Der String wurde korrekt als Entität markiert (*true positive*)
- Der String wurde als Entität markiert ist aber keine Entität (*false positive*)
- Der String wurde nicht als Entität markiert, ist aber eine Entität (*false negative*)

Des Weiteren ist es möglich, dass nur ein Teil einer Entität markiert wurde, oder dass ein annotierter Teilstring neben Tokens, die zu einer Entität gehören, weitere Zeichen umfasst. GERBIL stellt hierür die beiden Matching-Varianten *Strong Matching* und *Weak Matching* zur Verfügung, welche ebenfalls in diesem Abschnitt vorgestellt werden. Basierend auf diesen Möglichkeiten können folgende Maße zwischen der Menge aller Annotierungen, die durch einen Annotator vorgenommen wurden und der Menge aller in einem Goldstandard enthaltenen Entitäten, ermittelt werden:

$$\text{Recall} = \frac{\textit{true positives}}{\textit{true positives} + \textit{false negatives}}$$

Dies entspricht dem Verhältnis der von einem Annotator korrekt gefundenen Entitäten zu allen relevanten Entitäten. Die Precision dagegen ist wie folgt definiert:

$$\text{Precision} = \frac{\textit{true positives}}{\textit{true positives} + \textit{false positives}}$$

Dies entspricht dem Verhältnis der von einem Annotator korrekt gefundenen Entitäten im Verhältnis zu allen vom Annotator gefundenen Entitäten. Recall und Precision sind nur zusammen aussagekräftig. Beispielsweise erreicht ein Annotator, welcher nur eine Entität korrekt markiert und sonst keine weiteren Entitäten markiert hat, eine Präzision von eins, erreicht jedoch einen sehr schwachen Recallwert. Dagegen erreicht ein Annotator, welcher alle möglichen Strings auswählt einen Recall von eins, da alle Entitäten gefunden werden. Die Precision ist aber sehr gering, da sehr viele *false positives* generiert werden. Das Erreichen eines hohen Recalls und einer einer hohen Präzision sind also konkurrierende Ziele, es muss somit abgewogen werden, in welche Richtung ein Annotator optimiert werden soll, da es für Anwendung sinnvoll sein kann, dass die Abdeckung sehr hoch ist und dafür Fehler akzeptiert werden. Auf der anderen Seite kann es auch sinnvoll sein, dass möglichst wenige Fehler gemacht werden, dafür aber auf eine hohe Abdeckung verzichtet werden kann.
Ein Maß, welches die beiden Werte verbindet ist das F1-Measure, welches wie folgt definiert ist und das harmonische Mittel der beiden Werte bildet.

$$\text{F1-Measure} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

Micro und Macro

Ein Datensatz besteht meist aus mehreren Dokumenten. Aus diesem Grund werden für F1-Measure, Recall und Precision jeweils Macro und Micro Versionen berechnet. Die Micro Versionen, werden anhand aller im Datensatz vorhandenen Annotationen berechnet. Die Macro Version ist der durchschnittlich erreichte Wert über alle Dokumente. (Röder u. a., 2017)

Matching

Durch die Matching-Einstellung kann festgelegt werden, ab wann ein Ergebnis als korrekt gewertet wird. GERBIL stellt hier zwei Varianten zur Verfügung. Zum einen kann das *Strong Matching* konfiguriert werden, bei dem eine Annotierung nur als richtig gewertet wird, wenn sie exakt zu einer Entität im Goldstandard passt. Zum anderen kann das *Weak Matching* verwendet werden, welches Abweichungen von der im Goldstandard

enthaltenen Entität erlaubt. (Röder u. a., 2017)
Das Weak Matching ist für eine Annotation $a = (s, e, d, u)$ des Annotators und einer Annotation $a' = (s', e', d', u')$ eines Goldstandards wie folgt definiert (Röder u. a., 2017):

$$M_w(m, G) = \begin{cases} 1 \text{ wenn } u = u' \\ \wedge((s \leq s' \wedge e \leq e' \wedge s' < e) \\ \vee(s \geq s' \wedge e \geq e' \wedge s < e') \\ \vee(s \leq s' \wedge e \geq e') \\ \vee(s \geq s' \wedge e \leq e')) \\ \\ 0 \text{ sonst} \end{cases}$$

s und s' sind dabei die Startpositionen, e und e' die Endpositionen und u und u' die URLs der beiden Annotationen. Die URL ist nur für ein Linkingexperiment relevant.
Weak Annotation Match ist vor allem für Experimente sinnvoll, in denen Entity Recognition Teil des Experiments ist, da zum Beispiel für die gesuchte Entität *President Barack Obama* auch *Barack Obama* als korrekt anerkannt werden soll (Röder u. a., 2017). Aus diesem Grund werden alle Entity Recognition und A2KB Experimente in dieser Arbeit mit dem *Weak Matching* durchgeführt, während D2KB Experimente mit dem *Strong Matching* durchgeführt werden.

Evaluationsmaße für das Linking

Da ein Annotator einer Entität mehrere URIs aus verschiedenen Wissensbasen zuordnen kann und gleichzeitig auch ein Goldstandard für eine Entität oft mehrere URIs zu unterschiedlichen Wissensbasen enthält, werden für den Vergleich zweier Annotationen die URI-Sets S_1 und S_2 definiert.
Des Weiteren unterscheidet GERBIL zwei Klassen von URI-Sets: Die erste Klasse C_{KB} umfasst alle URI-Sets, die zumindest eine URI enthalten, die einer bekannten Wissensbasis zugeordnet werden kann. Die zweite Klasse C_{EE} enthält alle URI-Sets, für die dies nicht erfüllt ist. Die URI Sets S_1 und S_2 stimmen überein, wenn

$$((S_1 \in C_K B) \wedge (S_2 \in C_K B) \wedge (S1 \cap S2) \neq \emptyset) \vee ((S_1 \in C_E E) \wedge (S_2 \in C_E E))$$

Basierend auf diesem Matching werden ebenfalls die Maße Precision, Recall und F1-Measure berechnet. Für eine detaillierte Analyse stellt GERBIL zudem weitere Evalua-

tionsmaße zur Verfügung, die nachfolgend vorgestellt werden.

InKB, EE und GSInKB

Bei diesen Maßen werden nur bestimmte Teilmengen aller Entitäten betrachtet. InKB berechnet das Matching ausschließlich durch Entitäten, bei denen zumindest eines der beiden URI-Sets S_1 und S_2, des Goldstandards und des zu evaluierenden Annotators, zur Klasse C_{KB} gehört. Gegenteilig betrachtet EE ausschließlich Entitäten, bei denen S_1 und/oder S_2 der Klasse C_{EE} angehören. GSInKB erfasst dagegen nur die Entitäten, bei denen das URI-Set des Goldstandards (S_1) der Klasse C_{KB} zugeordnet wird. GSInKB wird ausschließlich für D2KB Experimente berechnet. Für alle diese Werte werden jeweils auch Micro und Macro Versionen angegeben. (Röder u. a., 2017)

6.2. Goldstandard

Zur Evaluierung des entwickelten Systems wird ein Goldstandard für Archivdaten benötigt, welcher sowohl Markierungen für Entitäten als auch Links zu beiden betrachteten Wissensbasen enthält.
Als Grundlage wurden dazu zufällig 100 Zeilen aus einer CSV-Datei mit Archivdaten ausgewählt. Jede Zeile repräsentiert dabei ein Dokument. Diese 100 Zeilen wurden unabhängig von zwei Personen annotiert. Dabei wurden Entitäten anhand der Thesen von Jha u. a. (2017) annotiert:

- Ein Satz muss keine lineare Struktur aufweisen. Zum Beispiel im String *Barack und Michelle Obama* werden *Barack* als erste Entität und *Michelle Obama* als zweite Entität markiert.
- Eine Entität besteht auf so vielen aufeinander folgenden Tokens wie möglich z. B. *legendary cryptanalyst Alan Turing*.
- Eine Entität wird zu einer exakt passenden Ressource verlinkt.
- Indirekt genannte Entitäten werden nicht markiert.
- Es werden nur Entitäten einer gegebenen Menge von Entitätstypen markiert. In diesem Fall Aufführungs- oder Veranstaltungstitel, Personen, und Aufführungsstätten, beziehungsweise Orte.

Die beiden annotieren Datensätze wurden anschließend manuell auf Differenzen untersucht und zu einem Goldstandard zusammengefügt.

Zum Abschätzen der Qualität eines Goldstandards wird standardmäßig das Maß Cohens Kappa von Cohen (1960) verwendet, welches folgendermaßen definiert ist:

$$\kappa = \frac{p_0 - p_c}{1 - p_c}$$

p_0 ist dabei der gemessene Übereinstimmungswert und p_c der Wert der zufälligen Übereinstimmung. Zur Berechnung des Übereinstimmungswerts und der zufälligen Übereinstimmung muss die Anzahl der eingeschätzten Objekte und die Menge der Kategorien bekannt sein.
Für NER können die Kategorien: *ist eine Entität* und *ist keine Entität festgelegt werden.* Allerdings ist die Menge der Objekte unklar, da die Annotatoren unabhängig voneinander entscheiden, welche Teilstrings markiert werden und welche nicht. Eine einfache Lösung diese Problems ist, dass die einzelnen Tokens eines Dokuments als zu klassifizierende Objekte definiert werden. Allerdings werden die beiden Klassen so als gleichwertig angesehen, was nicht korrekt ist, da es mehr Entity-Tokens gibt als Nicht-Entity-Tokens.
Für das Linking ergibt sich das Problem, dass die Annotatoren das Markieren der Entitäten und das Zuordnen von URLs in einem Schritt durchführen. Die Menge der Objekte, die verlinkt werden ist somit in beiden Datensätzen nicht identisch. Hinzu kommt, dass die Annotatoren frei aus der Menge aller Ressourcen aus zwei Wissensbasen auswählen, was zu einer großen Menge von Kategorien führt. Die Berechnung von Cohens Kappa ist deshalb für NER und das Linking nicht zielführend.
Stattdessen können zum Vergleich beider annotierten Datensätze die zuvor in Kapitel 6.1.2 beschriebenen Evaluierungsmaße F1-Measure, Recall und Precision verwendet werden (Hripcsak und Rothschild, 2005). Diese werden mit Hilfe des Evaluationsframeworks GERBIL (siehe Kapitel 6.1) berechnet, indem einer der beiden Datensätze als Referenzdatensatz verwendet wird und ein Annotator jeweils das passende annotierte Dokument aus dem zweiten Datensatz auswählt und zurücksendet.
Insgesamt wird das in Tabelle 6.1 dargestellte Annotator Agreement[1] erreicht.
Die Ergebnisse zeigen, dass das Annotator Agreement für Entity Recognition sehr hoch ist. Für das Linking ist es dagegen deutlich geringer. Es gibt aber dennoch eine ausreichend hohe Übereinstimmung. Der geringere Wert für das Linking liegt vor allem daran, dass es zum Beispiel für Namen oft sehr viele mögliche Kandidaten gibt und es somit oft schwierig ist den richtigen Kandidaten zu finden, insbesondere dann, wenn nur ein Nachname vorhanden ist. Zudem unterscheidet sich der String einer Entität häufig

1 http://gerbil.aksw.org/gerbil/experiment?id=201808110022
http://gerbil.aksw.org/gerbil/experiment?id=201808230001
(abgerufen am 23.8.2018)

	Gesamt	Entity Recognition	Linking
F1-Measure (Strong Matching)	0,7527	0,9424	0,7526
Precision (Strong Matching)	0,7826	0,9798	0,8071
Recall (Strong Matching)	0,725	0,9077	0,705
F1-Measure (Weak Matching)	0,7319	0,9068	0,7526
Precision (Weak Matching)	0,761	0,9429	0,8071
Recall (Weak Matching)	0,705	0,8735	0,705
Cohens Kappa (Tokenbasis)		0.9301	

Tabelle 6.1.: Inter Annotator Agreement

sehr stark von der passenden Entität in der Wissensbasis, wie bereits in Kapitel 3.2.1 beschrieben. Dies macht das Finden von Entitäten auch für menschliche Annotatoren oft sehr schwierig.

6.3. Evaluation Named Entitity Recognition

Im Folgenden sollen die Evaluationsergebnisse des NER-Moduls im Detail beschrieben und mit einigen weiteren NER-Ansätzen auf dem erstellten Goldstandard getestet werden. Anschließend erfolgt die Evaluation des CRF-Modells. GERBIL stellt bereits Schnittstellen zu einigen bekannten NER-Modulen zur Verfügung. Im Rahmen dieser Arbeit wird das entwickelte CRF-Verfahren mit folgenden drei Ansätze mit Hilfe des GERBIL-Frameworks verglichen:

- FOX[2] (siehe Kapitel 3.1.3)
- DBpedia Spotlight[3] (siehe Kapitel 3.1.3)
- FREME[4]

Tabelle 6.2 stellt die Ergebnisse[5] der vier Ansätze dar. Die Ergebnisse zeigen, dass zumindest FOX, ohne weitere Anpassungen bereits ein hohes F1-Measure erreicht, obwohl keine der von FOX genutzten Ansätze auf Archivdaten gelernt wurde. Allerdings benötigt FOX sehr viel Zeit für die Annotierung, das Ergebnis durch mehrere NER Tools

2 http://gerbil.aksw.org/gerbil/experiment?id=201808110018 (abgerufen am 23.8.2018)
3 http://gerbil.aksw.org/gerbil/experiment?id=201808110023 (abgerufen am 23.8.2018)
4 https://github.com/freme-project/freme-ner (abgerufen am 23.8.2018)
5 http://gerbil.aksw.org/gerbil/experiment?id=201808110018 http://gerbil.aksw.org/gerbil/experiment?id=201808110023 (abgerufen am 23.8.2018)

	FOX	DBpedia Spotlight	FREME	CRF
Micro F1-Measure	0,8247	0,6569	0,6143	**0,9327**
Micro Precision	0,8643	0,7197	0,6438	**0,9224**
Micro Recall	0,7885	0,6041	0,5874	**0,9433**
Macro F1-Measure	0,7697	0,6647	0,4076	**0,9247**
Macro Precision	0,8302	0,7217	0,3334	**0,9187**
Macro Recall	0,7404	0,6369	0,541	**0,9371**
Zeit in Millisek. pro Dokument	3085	553	**133**	276,21

Tabelle 6.2.: NER Ergebnisse

berechnet wird und jeweils auf alle Annotatoren gewartet werden muss.
Dies ist in diesem Zusammenhang ein Nachteil, da es das Ziel ist sehr große Archivdateien zu annotieren, was bei einer Menge von 60.000 Datensätzen bei einer Berechnungszeit von drei Sekunden pro Dokument bereits eine Berechnungszeit von 50 Stunden allein für das Finden von Entitäten zur Folge hat. Hinzu kommt noch die Zeit für das Linking der Entitäten. Die anderen beiden Annotatoren erreichen dagegen ein deutlich schlechteres Ergebnis und eignen sich nicht für das Annotieren von Archivdaten.
Das Ergebnis des CRF Ansatzes ist dagegen mehr als 10 % besser als für alle anderen getesteten Ansätze. Dies liegt vor allem daran, dass ausschließlich auf einem Archivdatencorpus gelernt wurde. Der Ansatz erreicht das Ergebnis bereits mit der Standardeinstellung von Rasa NLU. Die Zeit ist mit 276,21 Millisekunden ebenfalls deutlich schneller als das beste getestete Referenzsystem FOX.
Alles in allem deuten die sehr hohen Werte im F1-Measure darauf hin, dass es sich in diesem Fall um ein einfaches NER-Problem handelt.
Die einheitlichen Struktur aller Strings in dem betrachteten Datensatz erleichtert dabei das Generieren eines guten CRF-Modells, da die Felder, welche Entitäten enthalten immer an der gleichen Stelle stehen und immer die gleiche Reihenfolge haben. Dennoch wird NER für diese Daten benötigt, da in den Archivdaten viele Freitextfelder mit unstrukturiertem Text enthalten. Zudem werden in einigen Feldern oft Namen aufgelistet, die voneinander getrennt werden müssen. Die Daten sind zudem an vielen Stellen nicht konsistent.

6.4. Evaluation Named Entity Linking

In diesem Abschnitt soll die Evaluierung des AGDISTIS-Ansatzes auf den betrachteten Archivdaten, sowie aller neu entwickelten Features beschreiben werden. Zusätzlich wird untersucht, wie sich die Betrachtung mehrere Wissensbasen auf die Performance von

AGDISTIS auswirkt. Für den Linking Ansatz gibt es leider kein Referenzsystem, da derzeit kein Verfahren verfügbar ist, der eine der relevanten Wissensbasen nutzt. Anhand der Evaluierung des Ansatzes können aber die Stärken und Schwächen von AGDISTIS für das Linking von Archivdaten abgeschätzt werden und zusätzlich die Kombination zweier Wissensbasen untersucht werden, was ein bislang neuer Ansatz ist.

6.4.1. GND

Die GND ist eine Wissensbasis, welche zum Teil weit von der ursprünglich von AGDISTIS vorausgesetzten Struktur abweicht. So mussten, da die GND nicht das *rdfs:label* verwendet, weitere *Predicates* für die Generierung von Kandidaten betrachtet werden. Des Weiteren mussten bei der Indexierung Blank Nodes aufgelöst werden. Tabelle 6.3 zeigt das Ergebnis[6] für den Goldstandard, wenn nur die GND von AGDISTIS als Wissensbasis verwendet wird und der Goldstandard ausschließlich URIs aus der GND enthält. Gleichzeitig stellt sie die Performance für die Entity-Spezifischen Features (Kapitel 4.2.3), sowie das entwickelte Distanzmaß (Kapitel 4.2.3) zum Vergleich von Strings mit mehreren Token dar.
Die Ergebnisse zeigen, dass vor allem das geänderte Distanzmaß die Ergebnisse deutlich verbessert. Dies liegt daran, dass insbesondere Personennamen, welche die größte Menge aller Entitätstypen innerhalb des Goldstandards ausmachen im Goldstandard immer nach dem Schema *Vorname - Nachname* beschrieben werden. In der Wissensbasis jedoch liegen sie im Schema *Nachname, - Vorname* vor, was dazu führt, dass sie durch die ursprünglich verwendete N-Gram Distanz ausgefiltert werden. Die anderen entwickelten Features für Aufführungstitel und Orte führen allerdings zu kaum einer Verbesserung.
Zudem fällt auf, dass bei der Ursprünglichen Version von AGDISTIS der EE Recall-Werte sehr hoch, die EE Precision-Werte jedoch sehr niedrig sind. EE-Entitäten sind in diesem Fall Entitäten, welche nicht in der Wissensbasis enthalten sind. Der hohe Recall-Wert und der gleichzeitig niedrige Precision-Wert deutet darauf hin, dass der Annotator für viele Entitäten sehr häufig keine Kandidaten in der Wissensbasis findet, obwohl Entitäten vorhanden sind. Durch die Hinzunahme der Distanz und der Entitätstyp spezifischen Features nähern sich die Werte für Recall und Precision an, da mehr Kandidaten gefunden werden.

6 http://gerbil.aksw.org/gerbil/experiment?id=201808110017
http://gerbil.aksw.org/gerbil/experiment?id=201808120004
http://gerbil.aksw.org/gerbil/experiment?id=201808120001
(abgerufen am 23.8.2018)

	AGDSISTIS alle Features	mit Distanzmaß	ohne Features und ohne Distanzmaß
Micro F1-Measure	0,5197	0,5139	0,3376
Micro Precision	0,5204	0,5145	0,3381
Micro Recall	0,519	0,5132	0,3372
Macro F1-Measure	0,4512	0,4528	0,3031
Macro Precision	0,4526	0,4555	0,3057
Macro Recall	0,4507	0,4517	0,3019
InKB Macro F1-Measure	0,3877	0,3714	0,0205
InKB Macro Precision	0,4156	0,5045	0,0933
InKB Macro Recall	0,3745	0,3073	0,0118
InKB Micro F1-Measure	0,4551	0,4603	0,026
InKB Micro Precision	0,4782	0,6003	0,3415
InKB Micro Recall	0,4342	0,3733	0,0135
EE Macro F1-Measure	0,4698	0,4597	0,4271
EE Macro Precision	0,4616	0,3757	0,3054
EE Macro Recall	0,5249	0,6466	0,8487
EE Micro F1-Measure	0,6329	0,5771	0,5032
EE Micro Precision	0,5855	0,4535	0,338
EE Micro Recall	0,6886	0,793	0,9845
Zeit in Millisek. pro Dokument	1.889,15	1.136,65	1.274,47

Tabelle 6.3.: Ergebnisse des GND Index

6.4.2. Wikidata

Da Wikidata das Standard *rdfs:label* verwendet, mussten im Rahmen der Kandidatensuche initial keinerlei Anpassungen gemacht werden. Die Werte sind wie in Tabelle 6.4 zu entnehmen ist insgesamt deutlich höher als für den GND-Index[7] und erreichen ohne weitere Features bereits ein Micro F1-Measure von 66,62 %. Mit den vorgenommenen Änderungen erreicht AGDISTIS am Ende ein Ergebnis von 75,21 %. Verwendet wurde hier der Wikidata Goldstandard. Auch hier zeigen die InKB und EE Werte eine schrittweise Annäherung durch die Hinzunahme weiterer Features, was auf eine Verbesserung der Kandidatengenerierung hindeutet.
Die Verbesserung durch die Hinzunahme der Entitätstyp spezifischen Features ist im Vergleich zur GND deutlich höher. Dies deutet darauf hin, dass speziell diese Features einen höheren Effekt bei Wikidata erzielen als bei der GND.
Insgesamt gesehen funktioniert AGDISTIS mit der Wikidata als Wissensbasis deutlich besser als mit der GND der deutschen Nationalbibliothek, was zum Einen an daran liegt, dass die Kandidatengenerierung besser funktioniert als bei der GND. Zum Anderen kann der HITS-Algorithmus Ambiguität zwischen Entitäten oft schwerer auflösen, da viele Entitäten oft kaum ein- und ausgehende Kanten haben. Die GND enthält zudem sehr viele feingranulare Ressourcen. Beispielsweise sind bei Organisationen wie Theaterinstitutionen oft auch Ressourcen für Vorgängerorganisationen enthalten, während Wikidata diese in einer Ressource zusammenfasst. Diese tragen zusätzlich zu einer höheren Ambiguität bei.

7 http://gerbil.aksw.org/gerbil/experiment?id=201808110014
http://gerbil.aksw.org/gerbil/experiment?id=201808120003
http://gerbil.aksw.org/gerbil/experiment?id=201808120002
(abgerufen am 23.8.2018)

	AGDSISTIS alle Features	mit Distanzmaß	ohne Features und ohne Distanzmaß
Micro F1-Measure	0,7521	0,6959	0,6662
Micro Precision	0,7531	0,6968	0,6671
Micro Recall	0,7511	0,695	0,6654
Macro F1-Measure	0,7216	0,6583	0,6153
Macro Precision	0,7256	0,6623	0,6193
Macro Recall	0,7199	0,6566	0,6136
InKB Macro F1-Measure	0,618	0,5236	0,4567
InKB Macro Precision	0,6599	0,7239	0,6703
InKB Macro Recall	0,612	0,4429	0,3718
InKB Micro F1-Measure	0,6698	0,5888	0,5319
InKB Micro Precision	0,6835	0,7255	0,7081
InKB Micro Recall	0,6566	0,4954	0,426
EE Macro F1-Measure	0,7264	0,6782	0,6558
EE Macro Precision	0,7279	0,6007	0,5671
EE Macro Recall	0,7581	0,8456	0,8543
EE Micro F1-Measure	0,829	0,7718	0,7537
EE Micro Precision	0,8157	0,6823	0,6498
EE Micro Recall	0,8426	0,8883	0,8972
Zeit in Millisekunden pro Dokument	1.889,15	1.136,65	1.274,47

Tabelle 6.4.: Ergebnisse des Wikidata Index

6.4.3. Kombinerte Wissensbasen

Die Tabelle 6.6 zeigt das Ergebnis[8] für die Evaluation, wenn der kombinierte Suchindex verwendet wird und im Goldstandard gleichzeitig GND- und Wikidata-URIs enthalten sind. Das F1-Measure ist insgesamt gesehen um circa 8 % besser als wenn nur GND Entitäten Index verwendet wird, jedoch 15 % geringer im Vergleich zum Linking nur für Wikidata URLs. Zur besseren Interpretation der Ergebnisse[9] sind in Tabelle 6.5 die Ergebnisse dargestellt, wenn jeweils mit dem Wikidata und mit dem GND-Index alle Links im Goldstandard vorhergesagt werden.

	Wikidata	GND
Micro F1-Measure	0,5894	0,5029
Micro Precision	0,5902	0,5036
Micro Recall	0,5887	0,5023
Macro F1-Measure	0,5783	0,4359
Macro Precision	0,5809	0,4373
Macro Recall	0,5771	0,4353

Tabelle 6.5.: Gesamter Goldstandard Ergebnisse für den Wikidata- und den GND-Index

Die Ergebnisse zeigen, dass die Kombination beider Indizes das Ergebnis im Vergleich zum Linking nur mit dem Wikidata-Index nur geringfügig verbessert, obwohl der einzelne Wikidata-Index keine Links zu Entitäten finden kann, die nur eine GND-URI besitzen.
Die Kombination der beiden Wissensbasen für das Linking zu mehreren Wissensbasen verbessert also nicht signifikant die Performance von AGDISTIS. Ein Grund dafür ist, dass Wikidata-Ressourcen generell mehr Links zu anderen Ressourcen haben und GND-Ressourcen somit nicht durch den HITS-Algorithmus als *Authorities* identifiziert werden. Wikidata enthält zudem deutlich mehr Triple (über 14 Milliarden) im Vergleich zur GND (circa 165 Millionen). Es liegt die Vermutung nah, dass speziell bei Ressourcen, welche nur in der GND enthalten sind, sich dennoch häufig Wikidata-Ressourcen durchsetzen, da zwischen ihnen mehr Links existieren.

8 http://gerbil.aksw.org/gerbil/experiment?id=201808110007
http://gerbil.aksw.org/gerbil/experiment?id=201808110003
http://gerbil.aksw.org/gerbil/experiment?id=201808100008
(abgerufen am 23.8.2018)

9 http://gerbil.aksw.org/gerbil/experiment?id=201808110013
http://gerbil.aksw.org/gerbil/experiment?id=201808110016
(abgerufen am 23.8.2018)

	AGDSISTIS alle Features	mit Distanzmaß	ohne Features und ohne Distanzmaß
Micro F1-Measure	0,5946	0,5681	0,5029
Micro Precision	0,5953	0,5688	0,5036
Micro Recall	0,5938	0,5674	0,5023
Macro F1-Measure	0,5442	0,5187	0,4651
Macro Precision	0,5456	0,5214	0,4678
Macro Recall	0,5437	0,5176	0,464
InKB Macro F1-Measure	0,5072	0,4952	0,3889
InKB Macro Precision	0,5132	0,6328	0,6839
InKB Macro Recall	0,5086	0,4263	0,2849
InKB Micro F1-Measure	0,5614	0,5601	0,4434
InKB Micro Precision	0,5599	0,6658	0,7248
InKB Micro Recall	0,563	0,4833	0,3194
EE Macro F1-Measure	0,5163	0,4521	0,4643
EE Macro Precision	0,5458	0,3818	0,3477
EE Macro Recall	0,5181	0,6063	0,7948
EE Micro F1-Measure	0,6717	0,5802	0,5629
EE Micro Precision	0,679	0,4052	0,338
EE Micro Recall	0,6645	0,7601	0,9214
Zeit in Millisek. pro Dokument	2.939,29	1.664,60	1.802,24

Tabelle 6.6.: Kombinierter Index Ergebnisse

6.4.4. Alternativer Ansatz

Nachfolgend soll ein alternative Ansatz für das Linking über zwei Wissensbasen evaluiert werden, welcher in Kapitel 4.2.4 beschrieben wurde. Da der Ansatz nicht implementiert wurde, kann ein Evaluationsergebnis nicht direkt durch GERBIL berechnet werden. Allerdings ist es möglich die schlechtest mögliche und die bestmögliche Performance anhand der einzelnen Ergebnisse der beiden AGDISTIS-Instanzen für die GND und für Wikidata für ein D2KB-Experiment zu berechnen. Ziel ist es das Micro F1-Measure, die Micro Precision und den Micro Recall zu berechnen. Alle anderen GERBIL-Werte können ebenfalls nach dem gleichen Verfahren ermittelt werden. Für eine Abschätzung der Möglichkeiten des Ansatzes reichen aber die zuvor genannten Werte. Dazu müssen die minimal und maximal möglichen *true positives*, *false positives* und *false negative* berechnet werden. Diese können ermittelt werden, indem für jede Entität in jedem Dokument das Ergebnis für beide Annotatoren verglichen wird. Beim Vergleich der Ergebnisse zweier Entitäten können folgende Fälle eintreten:

1. Beide Instanzen liefern ein *true positive*: Die maximalen und minimalen *true positives* erhöhen sich um eins.

2. Beide Instanzen liefern ein *false negative*: Die minimalen und maximalen *false negatives* erhöhen sich um eins.

3. Beide Instanzen liefern ein false positive: Die minimalen und maximalen *false positives* erhöhen sich um eins.

4. Eine der beiden Instanzen liefert ein *true positive*, eine ein *false negative*: Die maximalen *true positives* und die maximalen *false negatives* erhöhen sich um eins.

5. Eine der beiden Instanzen liefert ein *true positive*, eine ein *false positive*: Die maximalen *true positives* und die maximalen *false positives* erhöhen sich um eins.

Aus den so berechneten Ergebnissen können anschließend minimale und maximale Micro Werte berechnet werden.

	maximal	minimal
Micro F1-Measure	0,7234	0,361
Micro Precision	0,7234	0,361
Micro Recall	0,7234	0,361

Tabelle 6.7.: maximale und minimale Performace für zwei Instanzen

Die in Tabelle 6.7 dargestellten Werte zeigen, dass im optimalen Fall das Ergebnis um

circa 13 % besser ist als wenn zwei Wissensbasen verwendet werden. Allerdings beträgt das minimale Ergebnis nur circa 36 %. Das Potential des Ansatzes hängt also davon ab, ob es möglich ist ein gutes Klassifikationsmodell zu finden, welches aus den Ergebnissen der beiden Instanzen eine URI auswählt. Die Schwierigkeit liegt hier in der Erzeugung neuer Trainingsdaten, da zunächst Ergebnisse aus beiden Instanzen gesammelt werden müssen (siehe Kapitel 6.4.4). Der Ansatz ist zudem nur schlecht auf die Hinzunahme weiterer Wissensbasen erweiterbar, da jeweils immer neue Trainingsdaten erzeugt werden müssen. Das gleiche gilt für die Kombination anderer Wissensbasen.

6.4.5. Verschiedene Entitätstypen

Abschließend zur Evaluation des Linking Systems soll die Performace des kombinierten Indexes auf verschiedenen Entitätstypen untersucht werden. Dazu werden aus dem Goldstandard jeweils Datensätze extrahiert welche nur Entitäten eines bestimmten Typs enthält. Die betrachteten Typen sind Personen, Aufführungs- und Werkstitel, sowie Aufführungsorte. Tabelle 6.8 zeigt das Ergebnis[10] für alle Entitätstypen und alle implementierten Features.
Auffallend ist, dass das Ergebnis für Personen deutlich höher ist als für die beiden anderen betrachteten Entitätstypen. Das Ergebnis für Personen konnte durch das geänderte Distanzmaß um ungefähr 6 % gesteigert werden. Darüber hinaus ist die Performance für das Linking der Aufführungsorte bei der AGDISTIS-Variante ohne Erweiterungen besonders schlecht. Die Verbesserung durch die entwickelten Features ist hier am größten (ungefähr 24 %). Dennoch ist die Verlinkung der Ortsentitäten im betrachteten Datensatz offensichtlich am schwierigsten. Ein Grund dafür ist, dass in den Archivdaten teils sehr alte Daten enthalten sind und manche Aufführungsorte mittlerweile einen neuen Namen tragen. Beispielsweise ist für das *Theater Basel* im Datensatz häufig noch alte Name *Basel Stadttheater* enthalten. Zudem enthält speziell die GND häufig mehrere Datensätze für eine Unterorganisation, die oft den gleichen Namen oder einen ähnlichen Namen tragen, wie die Organisation selbst, was das Linking zusätzlich erschwert.

10 http://gerbil.aksw.org/gerbil/experiment?id=201808110008
http://gerbil.aksw.org/gerbil/experiment?id=201808110009
http://gerbil.aksw.org/gerbil/experiment?id=201808110010
http://gerbil.aksw.org/gerbil/experiment?id=201808110004
http://gerbil.aksw.org/gerbil/experiment?id=201808110005
http://gerbil.aksw.org/gerbil/experiment?id=201808110006
http://gerbil.aksw.org/gerbil/experiment?id=201808110000
http://gerbil.aksw.org/gerbil/experiment?id=201808110001
http://gerbil.aksw.org/gerbil/experiment?id=201808110002
(abgerufen am 23.8.2018)

	Mic F1-Measure	Mic Precision	Mic Recal l	Mac F1-Measure	Mac Precision	Mac Precision
Person	0,6863	0,6863	0,6863	0,6632	0,6632	0,6632
Titel	0,494	0,494	0,494	0,4851	0,4851	0,4851
Ort	0,3425	0,3425	0,3425	0,3943	0,3943	0,3943
Person-Distanz	0,6967	0,6967	0,6967	0,6689	0,6689	0,6689
Titel-Distanz	0,4524	0,4524	0,4524	0,3972	0,3972	0,3972
Ort-Distanz	0,2049	0,2049	0,2049	0,1977	0,1977	0,1977
Person-keine Features	0,6379	0,6379	0,6379	0,6525	0,6525	0,6525
Titel-keine Features	0,4464	0,4464	0,4464	0,3939	0,3939	0,3939
Ort-keine Features	0,0948	0,0948	0,0948	0,0823	0,0823	0,0823

Tabelle 6.8.: Ergebnisse der verschiedenen Entitätstypen

6.4.6. Gesamtes System

Tabelle 6.9 zeigt das Evaluationsergebnis[11] für das A2KB Experiment also der Kombination aus dem NER und Linking Ansatz. Insgesamt erreicht das System ein F1-Measure von ungefähr 55,8 %. Da bislang kein Referenzsystem verfügbar ist, kann die Qualität des Ansatzes nur schwer eingeordnet werden. Das größte Verbesserungspotential liegt hier beim Linking-System durch eine besseren Abstimmung der beiden Wissensbasen. Das NER-Problem ist für die betrachteten Daten dagegen deutlich leichter zu lösen. Dies liegt vor allem an der einheitlichen Struktur der Archivdaten, was das Trainieren eines CRF-Modells sehr einfach macht.
Insgesamt eignet sich der Ansatz sehr gut für die Annotierung von Archivdaten, wobei die Nutzung einer einzelnen Wissensbasis deutlich einfacher ist.
Die Nutzung mehrerer Wissensbasen muss dagegen weiter untersucht werden.

	Gesamt	Entity Recognition	D2KB
Micro F1-Measure	0,5577	0,9324	0,5819
Micro Precision	0,5514	0,9219	0,6211
Micro Recall	0,5642	0,9433	0,5474
Macro F1-Measure	0,5015	0,9237	0,5189
Macro Precision	0,4972	0,9173	0,562
Macro Recall	0,5091	0,9366	0,4898

Tabelle 6.9.: Gesamtergebnis

11 http://gerbil.aksw.org/gerbil/experiment?id=201808110028
(abgerufen am 23.8.2018)

7. Fazit und Ausblick

In dieser Arbeit wurde ein Ansatz entwickelt, welcher dazu geeignet ist in einer großen Menge von Archivdaten Entitäten zu erkennen und URIs in einer oder mehreren Wissensbasen zuzuordnen. Die beiden betrachteten Wissensbasen sind die gemeinsame Normdatei der deutschen Nationalbibliothek und Wikidata.
Während das NER Problem relativ leicht zu lösen war und ein F1-Measure von über 90 % erreicht wurde, war das Linking Problem insgesamt deutlich schwieriger zu lösen. Dabei gibt es bereits sehr große Unterschiede in der Qualität des Linkings je nach dem welche Wissensbasis verwendet wird. Während das Linking allein auf Wikidata Entitäten ein F1-Measure von 75 % erreicht, liegt das Ergebnis beim Linking von GND Entitäten nur bei 51 %.
Die Kombination zweier Wissensbasen ist durch das Ungleichgewicht der Menge an Einträgen der beiden Wissensbasen ein schwierig zu lösendes Problem. Insbesondere die geringere Menge an Links zwischen Ressourcen in der GND erschwerte in dieser Arbeit die Kombination der beiden Wissensbasen. Zur Verbesserung des Linkings wurden im AGDISTIS-Ansatz verschiedene neue Features eingeführt, die maßgeblich zur Verbesserung der Qualität des Linkings beitragen konnten. Insbesondere das neue Distanzmaß hat zu einer erheblichen Verbesserung geführt und sollte auf weiteren Datensätzen getestet werden. Alles in allem konnte der betrachtete Ansatz das Problem lösen. Potential für die Weiterentwicklung liegt insbesondere darin, die Disambiguierung über GND Entitäten weiter zu verbessern, zum Beispiel indem ältere Versionen von Ressourcen durch Vorgänger- Nachfolger Relationen ausgefiltert werden.
Ein weiteres Ziel sollte es sein, weitere Ansätze auf dem Goldstandard zu testen, damit die Ergebnisse dieses Ansatzes besser eingeschätzt werden können. Des Weiteren sollten für das Problem des Linkings durch zwei Wissensbasen weitere Kombinationen von Wissensbasen getestet werden.

Literaturverzeichnis

[Brin und Page 1998] Brin, S. ; Page, L.: The Anatomy of a Large-Scale Hypertextual Web Search Engine. In: *Seventh International World-Wide Web Conference (WWW 1998)*, URL http://ilpubs.stanford.edu:8090/361/, 1998

[Cohen 1960] Cohen, Jacob: A Coefficient of Agreement for Nominal Scales. (1960)

[Dai u. a. 2012] Dai, Hongjie ; Wu, Chi-Yang ; Tzong, Richard ; Tsai, Richard Tzong-Han ; Hsu, Wen-Lian: From Entity Recognition to Entity Linking: A Survey of Advanced Entity Linking Techniques. (2012)

[Daiber u. a. 2013] Daiber, Joachim ; Jakob, Max ; Hokamp, Chris ; Mendes, Pablo N.: Improving Efficiency and Accuracy in Multilingual Entity Extraction. In: *Proceedings of the 9th International Conference on Semantic Systems*. New York, NY, USA : ACM, 2013 (I-SEMANTICS '13), S. 121–124. – URL http://doi.acm.org/10.1145/2506182.2506198. – ISBN 978-1-4503-1972-0

[Deutsche National Bibliothek 2012] Deutsche National Bibliothek: *Gemeinsame Normdatei (GND)*. 2012. – URL http://www.dnb.de/DE/Standardisierung/GND/gnd_node.html. – Zugriffsdatum: 2018-08-23

[Faruqui und Padó 2010] Faruqui, Manaal ; Padó, Sebastian: Training and Evaluating a German Named Entity Recognizer with Semantic Generalization. (2010)

[Ferragina und Scaiella 2012] Ferragina, Paolo ; Scaiella, Ugo: Fast and Accurate Annotation of Short Texts with Wikipedia Pages. In: *IEEE Softw.* 29 (2012), Januar, Nr. 1, S. 70–75. – URL http://dx.doi.org/10.1109/MS.2011.122. – ISSN 0740-7459

[Florian 2002] Florian, Radu: Named Entity Recognition As a House of Cards: Classifier Stacking. In: *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*. Stroudsburg, PA, USA : Association for Computational Linguistics, 2002 (COLING-02), S. 1–4. – URL https://doi.org/10.3115/1118853.1118863

[Hellmann u. a. 2013] Hellmann, Sebastian ; Lehmann, Jens ; Auer, Sören ; Brümmer, Martin: Integrating NLP Using Linked Data. In: *Proceedings of the 12th International Semantic Web Conference - Part II*. New York, NY, USA : Springer-Verlag New York, Inc., 2013 (ISWC '13), S. 98–113. – URL `http://dx.doi.org/10.1007/978-3-642-41338-4_7`. – ISBN 978-3-642-41337-7

[Hoffart u. a. 2011] Hoffart, Johannes ; Yosef, Mohamed A. ; Bordino, Ilaria ; Fürstenau, Hagen ; Pinkal, Manfred ; Spaniol, Marc ; Taneva, Bilyana ; Thater, Stefan ; Weikum, Gerhard: Robust Disambiguation of Named Entities in Text. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Stroudsburg, PA, USA : Association for Computational Linguistics, 2011 (EMNLP '11), S. 782–792. – URL `http://dl.acm.org/citation.cfm?id=2145432.2145521`. – ISBN 978-1-937284-11-4

[Hripcsak und Rothschild 2005] Hripcsak, George ; Rothschild, Adam S.: Agreement, the F-Measure, and Reliability in Information Retrieval. (2005)

[Jha u. a. 2017] Jha, Kunal ; Röder, Michael ; Ngonga Ngomo, Axel-Cyrille: All That Glitters is not Gold – Rule-Based Curation of Reference Datasets for Named Entity Recognition and Entity Linking. In: *The Semantic Web. Latest Advances and New Domains: 14th International Conference, ESWC 2017, Proceedings*, Springer International Publishing, 2017. – URL `https://svn.aksw.org/papers/2017/ESWC_EAGLET_2017/public.pdf`

[Kleinberg 1999] Kleinberg, Jon M.: Authoritative Sources in a Hyperlinked Environment. In: *J. ACM* 46 (1999), September, Nr. 5, S. 604–632. – URL `http://doi.acm.org/10.1145/324133.324140`. – ISSN 0004-5411

[Kondrak 2005] Kondrak, Grzegorz: N-Gram Similarity and Distance. In: Consens, Mariano (Hrsg.) ; Navarro, Gonzalo (Hrsg.): *String Processing and Information Retrieval*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2005, S. 115–126. – ISBN 978-3-540-32241-2

[Konkol 2015] Konkol, Michal: *Named Entity Recognition*, University of West Bohemia, Dissertation, 2015

[Lafferty u. a. 2001] Lafferty, John D. ; McCallum, Andrew ; Pereira, Fernando C. N.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: *Proceedings of the Eighteenth International Conference on Machine Learning*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2001

(ICML '01), S. 282–289. – URL http://dl.acm.org/citation.cfm?id=645530.655813. – ISBN 1-55860-778-1

[Liu und Nocedal 1989] Liu, Dong C. ; Nocedal, Jorge: On the limited memory BFGS method for large scale optimization. In: *Mathematical Programming* 45 (1989), Nr. 1, S. 503–528. – URL https://doi.org/10.1007/BF01589116. – ISSN 1436-4646

[Manning u. a. 2014] Manning, Christopher ; Surdeanu, Mihai ; Bauer, John ; Finkel, Jenny ; Bethard, Steven ; McClosky, David: The Stanford CoreNLP Natural Language Processing Toolkit. In: *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Association for Computational Linguistics, 2014, S. 55–60. – URL http://www.aclweb.org/anthology/P14-5010

[Mansouri u. a. 2008] Mansouri, Alireza ; Suriani Affendey, Lilly ; Mamat, Ali: Named Entity Recognition Approaches. In: *IJCSNS International Journal of Computer Science and Network Security* 8 (2008), S. 339–344

[McCallum u. a. 2000] McCallum, Andrew ; Freitag, Dayne ; Pereira, Fernando C. N.: Maximum Entropy Markov Models for Information Extraction and Segmentation. In: *Proceedings of the Seventeenth International Conference on Machine Learning*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2000 (ICML '00), S. 591–598. – URL http://dl.acm.org/citation.cfm?id=645529.658277. – ISBN 1-55860-707-2

[Mccallum und Li 2003] Mccallum, Andrew ; Li, Wei: Early Results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-Enhanced Lexicons. (2003)

[Mohit 2014] Mohit, Behrang: Named Entity Recognition. In: *Natural Language Processing of Semitic Languages* Bd. 1. Springer, 2014, S. 221–245

[N. Darroch und Ratcliff 1972] N. Darroch, J ; Ratcliff, D: Generalized Iterative Scaling for Log-Linear Models. 43 (1972), 10

[Ng und Jordan 2001] Ng, Andrew Y. ; Jordan, Michael I.: On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes. In: *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*. Cambridge, MA, USA : MIT Press, 2001 (NIPS'01), S. 841–848. – URL http://dl.acm.org/citation.cfm?id=2980539.2980648

[Pallets Team 2010] Pallets Team: *Flask (A Python Microframework).* 2010. – URL `http://flask.pocoo.org/`. – Zugriffsdatum: 2018-08-23

[Rabiner und Juang 1986] Rabiner, L. R. ; Juang, B. H.: An introduction to hidden Markov models. In: *IEEE ASSp Magazine* (1986)

[Rasa Technologies GmbH 2017] Rasa Technologies GmbH: *Rasa: Open source conversational AI.* 2017. – URL `https://rasa.com/`. – Zugriffsdatum: 2018-08-23

[Ratinov und Roth 2009] Ratinov, Lev ; Roth, Dan: Design Challenges and Misconceptions in Named Entity Recognition. In: *Proceedings of the Thirteenth Conference on Computational Natural Language Learning.* Stroudsburg, PA, USA : Association for Computational Linguistics, 2009 (CoNLL '09), S. 147–155. – URL `http://dl.acm.org/citation.cfm?id=1596374.1596399`. – ISBN 978-1-932432-29-9

[Röder u. a. 2017] Röder, Michael ; Usbeck, Ricardo ; Ngonga Ngomo, Axel-Cyrille: GERBIL – Benchmarking Named Entity Recognition and Linking consistently. (2017)

[Shen u. a. 2015] Shen, Wei ; Wang, Jianyong ; Han, Jiawei: Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions. (2015)

[Speck und Ngonga Ngomo 2017] Speck, René ; Ngonga Ngomo, Axel-Cyrille: Ensemble Learning of Named Entity Recognition Algorithms Using Multilayer Perceptron for the Multilingual Web of Data. In: *Proceedings of the Knowledge Capture Conference.* New York, NY, USA : ACM, 2017 (K-CAP 2017), S. 26:1–26:4. – URL `http://doi.acm.org/10.1145/3148011.3154471`. – ISBN 978-1-4503-5553-7

[Sundheim 1995] Sundheim, Beth M.: Overview of Results of the MUC-6 Evaluation. In: *Proceedings of the 6th Conference on Message Understanding.* Stroudsburg, PA, USA : Association for Computational Linguistics, 1995 (MUC6 '95), S. 13–31. – URL `https://doi.org/10.3115/1072399.1072402`. – ISBN 1-55860-402-2

[Tjong Kim Sang und De Meulder 2003] Tjong Kim Sang, Erik F. ; De Meulder, Fien: Introduction to the CoNLL-2003 Shared Task: Language-independent Named Entity Recognition. In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4.* Stroudsburg, PA, USA : Association for Computational Linguistics, 2003 (CONLL '03), S. 142–147. – URL `https://doi.org/10.3115/1119176.1119195`

[Usbeck u. a. 2014] Usbeck, Ricardo ; Ngonga Ngomo, Axel-Cyrille ; Röder, Michael ; Gerber, Daniel ; Coelho, Sandro A. ; Auer, Sören ; Both, Andreas: AGDISTIS - Graph-Based Disambiguation of Named Entities Using Linked Data. In: Mika, Peter (Hrsg.) ; Tudorache, Tania (Hrsg.) ; Bernstein, Abraham (Hrsg.) ; Welty, Chris (Hrsg.) ; Knoblock, Craig (Hrsg.) ; Vrandečić, Denny (Hrsg.) ; Groth, Paul (Hrsg.) ; Noy, Natasha (Hrsg.) ; Janowicz, Krzysztof (Hrsg.) ; Goble, Carole (Hrsg.): *The Semantic Web – ISWC 2014*. Cham : Springer International Publishing, 2014, S. 457–471. – ISBN 978-3-319-11964-9

[Van Rijsbergen 1979] Van Rijsbergen, C. J.: *Information Retrieval*. 2nd. Newton, MA, USA : Butterworth-Heinemann, 1979. – ISBN 0408709294

[W3C 2004a] W3C: *RDF Primer*. 2004. – URL `https://www.w3.org/TR/rdf-primer/`. – Zugriffsdatum: 2018-08-23

[W3C 2004b] W3C: *Resource Description Framework (RDF): Concepts and Abstract Syntax*. 2004. – URL `https://www.w3.org/TR/rdf-concepts/`. – Zugriffsdatum: 2018-08-23

[W3C 2014] W3C: *Terse RDF Triple Language*. 2014. – URL `https://www.w3.org/TR/turtle/`. – Zugriffsdatum: 2018-08-23

[Wikimedia Foundation 2012] Wikimedia Foundation: *Wikidata*. 2012. – URL `https://www.wikidata.org`. – Zugriffsdatum: 2018-08-23

[Wikimedia Foundation 2018] Wikimedia Foundation: *Wikibase/Indexing/RDF Dump Format)*. 2018. – URL `https://www.mediawiki.org/wiki/Wikibase/Indexing/RDF_Dump_Format#Truthy_statements`. – Zugriffsdatum: 2018-08-23

Anhang A.

Inhalt der CD

Thesis.pdf: Dieses Dokument im PDF-Format

Evaluation.xsls: Evaluationsergebnisse

docs/: Enthält zitierte Forschungsarbeiten, soweit elektronisch vorhanden

nif-dateien/: Zur Evaluation verwendete NIF-Dateien

csv-dateien: Annotierte Dateien im CSV-Format