Master Thesis

im Rahmen des
Universitätslehrganges „Geographical Information Science & Systems“
(UNIGIS MSc) am Interfakultären Fachbereich für GeoInformatik (Z_GIS)
der Paris Lodron-Universität Salzburg

zum Thema

„Semiautomatische Extraktion geographischer Information aus gering strukturierten Textdaten“

am Beispiel von Nachrichtentexten
des Mikroblogging-Portals Twitter

vorgelegt von

B.Sc. Jan Hölzer
103089, UNIGIS MSc Jahrgang 2013

Zur Erlangung des Grades
„Master of Science (Geographical Information Science & Systems) – MSc(GIS)”

Gutachter: Prof. Dr. Josef Strobl

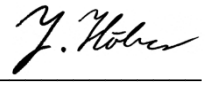
Wuppertal, den 29.02.2016

Danksagung

Mein besonderer Dank gilt Herrn Prof. Dr. Josef Strobl und Frau Ass. Prof. Dr. Gudrun Wallentin, die mich bei der Themenfindung dieser Arbeit unterstützt und während des Studiums aktiv beraten haben. Außerdem bedanke ich mich bei den übrigen ModulbetreuerInnen und dem gesamten UNIGIS-Team für eine unvergessliche Studienzeit.

Eigenständigkeitserklärung

Ich versichere, diese Master Thesis ohne fremde Hilfe und ohne Verwendung anderer als der angeführten Quellen angefertigt zu haben, und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat. Alle Ausführungen der Arbeit, die wörtlich oder sinngemäß übernommen wurden, sind entsprechend gekennzeichnet.

Wuppertal, den 29.02.2016 

Datum, Unterschrift

Zusammenfassung

Durch die steigenden Nutzerzahlen Sozialer Medien und den mit ihnen einhergehenden Veränderungen in der Erfassung, Nutzung und Verbreitung von Nachrichten eröffnen sich zahlreiche neue Chancen und Herausforderungen. Gleichzeitig ist ein weitreichender Zugang zu Informationen für eine Gesellschaft von herausragender Bedeutung. Entsprechend wichtig erscheint es, sowohl aus Sicht der journalistischen Arbeit als auch im Kontext zahlreicher anderer Anwendungsbereiche, belastbare Erkenntnisse darüber zu gewinnen, von wie vielen Personen bestimmte Medien genutzt werden. Der Georeferenzierung der innerhalb großer Datenbestände vorhandenen Benutzer-generierten Informationen kommt deshalb eine besonders hohe Relevanz zu.

Im Verlauf dieser Arbeit werden zunächst die innerhalb der Mikroblogging-Plattform Twitter eingesetzten Datenmodelle untersucht. Auf der Grundlage dieser Untersuchung wird eine Anwendung erstellt, die die Echtzeiterfassung von Tweet-, User- und Place-Objekten über die bei Twitter zur Verfügung stehenden Programmierschnittstellen und deren persistente Speicherung in einer Datenbank vornimmt.

Anschließend werden Methoden zusammengestellt, die es ermöglichen, die innerhalb von Tweet-Texten vorhandenen Toponyme und sonstigen Entitäten unter Verwendung frei verfügbarer Anwendungsbibliotheken zu erkennen, zu klassifizieren und anschließend mit den der erkannten Entitätsklasse entsprechenden Annotation zu versehen. Diese strukturierten Auszeichnungen können in weiteren Anwendungsschritten dazu genutzt werden, mithilfe der zuvor beschriebenen Anfragesprachen zusätzliche Informationen zu den erkannten Klassen aus öffentlich zugänglichen Datenbeständen abzurufen. Zur Georeferenzierung von Tweets bzw. der damit verknüpften User-Objekte wird auf die aus Wikipedia-Inhalten generierte DBPedia-Ontologie zurückgegriffen, um geographische Koordinaten zu den jeweils im Tweet-Text erkannten Toponymen abzufragen. Die auf diese Weise mit Koordinaten versehenen geographischen Objekte können anschließend zusammen mit weiteren aus den Texten extrahierten Informationen in ein strukturiertes Datenformat überführt werden, das von gängigen Geoinformationssystemen unterstützt wird. Im Verlauf dieser Arbeit wird dazu die Geography Markup Language eingesetzt. Da dieses Datenformat viele Gemeinsamkeiten mit den zur Modellierung von Ontologien gebräuchlichen Formaten aufweist, ist eine weitere Zusammenführung von Datenbeständen denkbar.

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Motivation und Relevanz der Arbeit

Die steigende Relevanz Sozialer Medien in vielen Lebensbereichen ist an mannigfaltigen Entwicklungen der letzten Jahre zu erkennen. Im Jahr 2015 besaßen die Internetnutzer weltweit im Durchschnitt 2,8 aktiv genutzte Social-Media-Accounts.[1] 80% aller deutschen Internetnutzer nutzten im Jahr 2015 soziale Medien.[2] 40 % der deutschen Social-Media-Nutzer haben sich im Jahr 2015 mindestens einmal täglich bei Twitter eingeloggt.[3] Aufgrund der rapide steigenden weltweiten Nutzerzahlen des Dienstes und seiner vergleichsweise einfach gehaltenen Datenstruktur empfiehlt sich die Mikroblogging-Plattform Twitter allerdings auch besonders für viele Anwendungen, die auf Analysen großer Mengen Nutzer-generierter Informationen angewiesen sind. Z. B. wird die durch Twitter generierte Datenbasis für Anwendungen auf den Gebieten der Demoskopie, der Mediennutzungsforschung, der Soziologie, des Krisenmanagements und des Marketing genutzt. Durch die große Anzahl aktiver Benutzer können wichtige Nachrichten bei Twitter in sehr kurzen Zeiträumen sehr viele Adressaten erreichen. Dieser Effekt wird seit 2013 verstärkt zur Katastrophenwarnung eingesetzt, indem z. B. Tweets von Hilfsorganisationen als Warnmeldungen gekennzeichnet werden.[4] Umgekehrt können die in Echtzeit abrufbaren Daten jedoch auch zur Katastrophenvorhersage genutzt werden.[5]

Auch im Bereich des Journalismus kommt Diensten wie Twitter eine wachsende Bedeutung zu. Nicht nur zur Umsatzsteigerung kommerzieller Medienunternehmen stellt die Reichweitenmessung ein wichtiges Instrument zur Verfügung. Auch um einen möglichst flächendeckenden und ausgewogenen Zugang zu Informationen im öffentlich-rechtlichen Bereich zu gewährleisten, müssen Reichweiten der unterschiedlichen Medienangebote regelmäßig untersucht werden. Der oben beschriebene Wandel des Nutzerverhaltens und die daraus abzuleitenden Neuausrichtungen medialer

[1] Statista 2015.
[2] Faktenkontor 2015.
[3] Tobesocial 2015.
[4] Pena, G. 2013.
[5] Uni Stuttgart 2015.

Kommunikation lassen tiefgreifende Veränderungen in der Erfassung, Nutzung und Verbreitung von Nachrichten erkennen. Möglichst detaillierte Erkenntnisse über die qualitative und quantitative Reichweite von Medien erscheinen deshalb für zahlreiche Anwendungsfelder relevant.

1.2 Zielsetzung

Innerhalb dieser Arbeit sollen Methoden erarbeitet werden, die es ermöglichen, die in kurzen Nachrichtentexten Sozialer Medien lose strukturiert vorhandenen Geoinformationen zu extrahieren und zusammen mit den automatisiert oder manuell erfassten Geodaten der explizit zur Geolokalisierung vorhandenen Attribute in eine gemeinsame Datenstruktur zu überführen. Diese Datenstruktur soll insbesondere zur Speicherung, Analyse und Visualisierung unter Verwendung eines Geoinformationssystems herangezogen werden können.

Zusätzlich soll eine Integration der auf diese Weise verarbeiteten Informationen in eine übergeordnete Datenbasis möglich sein, auf deren Grundlage domänenübergreifende, strukturierte Anfragen ausgeführt werden können. In einem beispielhaften Anwendungsfall soll untersucht werden, ob im Gegensatz zu der für Printmedien und lineare Medien eher grob aufgelösten quantitativen Reichweitenmessung im Kontext sozialer Medien eine genauere ggf. auf einzelne Themengebiete bezogene Aussage über den Ursprung und den Konsum von Nachrichteninhalten getroffen werden kann.

Am Beispiel der Mikroblogging-Plattform Twitter sollen die quantitativen Anteile von Nachrichten erfasst werden, die über unterschiedliche Vorgehensweisen eine Georeferenzierung ermöglichen. Zugleich soll eine qualitative Bewertung der unter Verwendung der zuvor untersuchten Methoden erfassten Kennzahlen vorgenommen werden.

1.3 Aufbau und Vorgehensweise

Nach einer kurzen allgemeinen Einführung in die Themengebiete Soziale Medien und Online-Nachrichten sowie deren Reichweitenbestimmung wird im Grundlagenteil zunächst die Mikroblogging-Plattform Twitter ausführlich beschrieben. Im Zuge dessen werden die der Plattform inhärenten Datenstrukturen dargestellt und dabei insbesondere die für die spätere Geolokalisierung relevanten Attribute der Tweet-, User- und Place-Objekte hervorgehoben. Neben den zur Anreicherung mit Geodaten vorgesehenen Attributen werden jedoch auch einige Objekteigenschaften angeführt, die ggf. später zur „indirekten“ Georeferenzierung von Tweets herangezogen werden können.

Anschließend werden Vorgehensweisen zur Akquise von Twitter-Daten aufgezeigt, indem die in der Twitter-Plattform implementierten Programmierschnittstellen (APIs) und deren Endpunkte mit ihren jeweiligen Vor- und Nachteilen untersucht werden. Nach einem kurzen Exkurs über die bei dem Verbindungsaufbau zu Twitters APIs zwingend einzusetzende Authentifizierungs- bzw. Autorisierungsmethode „OAuth“ werden die im Rahmen des nachfolgenden Anwendungsentwurfes und der Datenanalyse benötigten, formalen Repräsentationsformen, Datenformate und deren Anfragetechnologien beschrieben. Dabei sollen zusätzliche Möglichkeiten der Datenmodellierung und Datenmigration auf der Basis semantischer Technologien und Methoden zur semantischen Auszeichnung schwach strukturierter Text-Daten vorgestellt werden.

In einem nächsten Schritt werden im Hinblick auf die bestehenden Anforderungen der Informationsextraktion Verfahren der Rechner-gestützten Textanalyse bzw. der Analyse „natürlicher Sprache“ untersucht und konkrete Anwendungsbibliotheken ausgewählt, auf deren Grundlage die nachfolgend verwendete Anwendung implementiert werden soll. Nach Abschluss der Entwicklung bzw. Weiterentwicklung und Implementierung der dazu notwendigen Klassen werden unterschiedliche Datensätze als Stichproben unter Verwendung der Twitter Streaming API bezogen und zur weiterführenden Analyse und späteren Aufbereitung zunächst in einer Datenbank gespeichert. Ausgehend von diesem Datenbestand werden die innerhalb der Twitter-Plattform in strukturierter Form vorliegenden Geoinformationen quantitativ und qualitativ analysiert, um daran anschließend durch Einbeziehung der vorher ausgewählten Anwendungsbibliotheken die in schwach strukturierter Form vorhandenen Geoinformationen aus Tweet-Texten und anderen über die Streaming API abrufbaren Attributen zu extrahieren und in eine

Datenstruktur zu überführen, die in einem Geoinformationssystem verarbeitet werden kann.

Abschließend werden die unterschiedlichen Resultate im Hinblick auf ihre Eignung zum Einsatz im Kontext der Mediennutzungsforschung bewertet, bevor in einem zusammenfassenden Fazit die Ergebnisse der Arbeit diskutiert sowie mögliche Handlungsempfehlungen aufgezeigt werden.

2 Grundlagen und Methodik

2.1 Soziale Medien und Nachrichtenartikel

Der Begriff „Soziales Netzwerk“ beschreibt ein Netzwerk, dessen Knoten den Akteuren und dessen Kanten den sozialen Interaktionen bzw. den sozialen Relationen zwischen diesen Akteuren entsprechen.[6] Diese sozialen Relationen können entweder bidirektional - z. B. bei Freundschaftsbeziehungen - oder unidirektional, wie z. B. im Falle der Follower eines Twitter-Accounts, ausgeprägt sein. Unidirektionale Relationen können als direkte Relation - z. B. bei Anhängerschaft eines bestimmten Benutzers - oder als indirekte Relation durch die Auswahl bestimmter Diskussionsfäden (Threads), klassifiziert werden.[7]

Innerhalb von Mikroblogs mögliche soziale Interaktionen sind z. B. die Antwort auf einzelne Nachrichten oder die Bezugnahme auf einen bestimmten Benutzer. Da die unterschiedlichen Relationen für gewöhnlich als strukturierte Attribute innerhalb der Datenstrukturen von Mikroblogs gespeichert werden, lassen viele der auf diesen Relationen basierenden Interaktionen eine relativ genaue Analyse zu. Anhand der „Follower“ eines Twitter-Benutzers ist z. B. ein gewisser Anteil der Konsumenten eines Tweets mehr oder weniger genau zu bestimmen. Bei Nachrichtenartikeln existieren zwar zwischen dem Autor und den Lesern eines Artikels häufig Relationen, die den innerhalb eines Mikroblogs vorhandenen Strukturen ähneln; allerdings werden diese i. d. R. nicht in strukturierter Form gespeichert (ggf. in der Abonnementen-Liste des Verlags). Ein Nachrichtenartikel kann durch die folgenden vier Fragewörter beschrieben werden: wer, wann, wo und was.[8] Nachrichtenartikel unterliegen anders als Mikroblog-Texte meist einer redaktionellen Bearbeitung, die eine gewisse – jedoch nicht maschinenlesbare – Textform vorgibt. Ein weiteres, wichtiges Unterscheidungsmerkmal stellt die Textlänge dar. So ist für Mikroblog-Texte i. d. R. eine maximale Zeichenanzahl definiert – Tweets sind z. B. auf 140 Zeichen beschränkt. Für Nachrichtentexte gibt es hingegen keine allgemein gültige Zeichenbegrenzung.

[6] Vgl. Aggarwal, 2011 S. 2.
[7] Vgl. Liao et. al., 2012 S. 131 f.
[8] Vgl. Liu et. al., 2012 S. 104.

2.2 Mediale Reichweite von Online-Nachrichten

Mediale Reichweite bezeichnet eine quantitative Kennzahl, die den Anteil einer Gruppe von Personen wiederspiegelt, der ein bestimmtes Medium innerhalb eines bestimmten Zeitintervalls konsumiert.[9] So haben in Deutschland im Jahr 2015 z. B. durchschnittlich 28,86 Mio. Leser ab 14 Jahren pro Tag eine regionale Tageszeitung durch ein Abonnement bezogen.[10] 40,6 Mio. Personen, die mindestens 14 Jahre alt waren, haben täglich das Internet genutzt. Die durchschnittliche Nutzungsdauer betrug dabei pro Tag 108 Minuten.[11]

	Netto-Reichweite (%)	Netto-Reichweite (Mio. Unique User)	Brutto-Reichweite
FOCUS online	39,2	20,72	344,57
BILD	37,8	19,96	1628,38
SPIEGEL ONLINE	35,8	18,95	763,39
DIE WELT	30,2	15,97	162,84
STERN.de	20,5	10,83	135,35
ZEIT ONLINE	20,4	10,77	139,42
Süddeutsche.de	20,1	10,61	153,72
FAZ.NET	15,0	7,93	156,41
Handelsblatt	7,8	4,12	58,54

Tabelle 1: Brutto- und Nettoreichweite ausgewählter Online-Medien über einen Zeitraum von 3 Monaten[12]

Im Kontext der medialen Reichweite von Online-Angeboten unterscheidet man zwischen der Brutto-Reichweite und der Netto-Reichweite. Die Brutto-Reichweite gibt die absolute Anzahl der Benutzerkontakte in Mio. wieder. Die Netto-Reichweite gibt an, wie viele Benutzer welches Nachrichtenportal mindestens einmal während des untersuchten Zeitraums aufgerufen haben. Die Netto-Reichweite in % repräsentiert den

[9] Vgl. Hasebrink, Schröder, 2006 S. 291.
[10] Vgl. Institut für Demoskopie Allensbach, 2015.
[11] Vgl. Statista, 2015.
[12] AGOF digital facts, 2015.

Anteil der Benutzer, die ein bestimmtes Nachrichtenportal aufgerufen haben, relativ zu der um Mehrfachaufrufe bereinigten Gesamtzahl der Benutzer („Unique User“).

2.3 Die Mikroblogging Plattform Twitter

Die Mikroblogging Plattform "Twitter" wurde im März 2006 von Mitarbeitern des damals in San Francisco ansässigen Unternehmens "Odeo" zunächst unter dem Namen "Twttr" gegründet.[13] Im Jahr 2015 verzeichnete das Unternehmen Twitter 320 aktive Nutzer pro Monat und beschäftigt weltweit 4300 Mitarbeiter an 41 Standorten.[14]

2.3.1 Formale und inhaltliche Strukturen

Das innerhalb von Twitter eingesetzte Datenmodell basiert im Wesentlichen auf den vier Objektklassen Tweet, User, Entity und Place.[15] Jede dieser Objektklassen besitzt unterschiedliche Attribute, die eine unterschiedlich genaue, direkte oder indirekte Geolokalisierung ermöglichen.

User

Objekte der Klasse "User" repräsentieren die in Twitter verwendeten Benutzerkonten und bilden deren Eigenschaften und Relationen ab. Ein Benutzerkonto ist dabei nicht zwangsläufig einer realen Person zugeordnet. Es kann z. B. ebenfalls für eine Organisation oder eine Anwendung stehen. Im Folgenden wird der Begriff "Benutzer" synonym für all diejenigen Entitäten verwendet, die durch ein Benutzerkonto repräsentiert werden können. Folgende Attribute der Objektklasse "User" können ggf. zur Geolokalisierung verwendet werden:

[13] Vgl. O'Reilly et. al. 2013 S. 13.
[14] Vgl. Twitter, 2015.
[15] Vgl. Twitter, 2015 1.

- lang: Das Attribut "lang" enthält den BCP47-Code der innerhalb der (graphischen) Benutzerschnittstelle aktuell eingestellten Sprache. Zwar steht die Spracheinstellung nicht zwangsläufig im Zusammenhang mit dem Inhalt der verfassten Tweets; allerdings kann je nach der verwendeten Spracheinstellung mit unterschiedlicher Genauigkeit und mit unterschiedlicher Wahrscheinlichkeit auf die "geographische Herkunft" eines Tweets geschlossen werden.

- location: Das Attribut "location" enthält optional eine benutzerdefinierte Zeichenkette, die den Standort des jeweiligen Benutzers repräsentiert. Dabei sind jedoch keinerlei Vorgaben zur Größe der verwendeten geographischen Bezugsobjekte vorhanden. Der Wert "Europa" wäre deshalb ebenso denkbar wie der Wert "Universität Salzburg".

- time_zone: Das Attribut "time_zone" enthält die durch den Benutzer eingestellte Zeitzone. Genau wie bei dem Attribut "lang" besteht ein Zusammenhang zum Inhalt der von dem Benutzer verfassten Tweets nur zu einer gewissen Wahrscheinlichkeit.

- withheld_in_countries: Das Attribut "withheld_in_countries" kann einen oder mehrere zweistellige Länderkürzel der Länder enthalten, in denen das Benutzerkonto gesperrt wird. Dies bedeutet, dass Benutzerkonten aus den entsprechenden Ländern z. B. keine Tweets des gesperrten Benutzerkontos aufrufen können.

Tweet

Objekte der Klasse "Tweet" stellen die von Benutzerkonten verfassten Statusupdates ("Tweets") dar. Genau wie die Klasse "User" verfügt die Klasse "Tweet" über zahlreiche Attribute, die einen unterschiedlich ausgeprägten geographischen Bezug aufweisen und dadurch ggf. zur Geolokalisierung verwendet werden können:

- coordinates: Das Attribut "coordinates" umfasst ein in geoJSON formatiertes array, welches ein Pointfeature der bei der Erstellung des Tweets erfassten Koordinaten enthält. Nur wenn das Attribut "geo_enabled" des User-Objekts den Wert "true" aufweist, können neu verfasste Statusupdates mit zusätzlichen Geodaten versehen werden.[16]

- geo: Diese Relation ist lediglich aus Kompatibilitätsgründen vorhanden. Inzwischen wird stattdessen das Attribut "coordinates" verwendet.

- place: Mithilfe der Relation "place" kann ein Tweet mit einem oder mehreren Orten (place-Objekten) verknüpft werden. Dabei besteht nicht unbedingt ein Zusammenhang zu dem Standort, an dem der Tweet verfasst wurde. Es kann z.B. auch ein inhaltlicher Bezug des Tweet-Textes zu dem genannten Ort bestehen.

- user: Die Relation "user" bildet die Beziehung jedes Objektes der Klasse "Tweet" mit dem Objekt der Klasse "user" ab, welches den Urheber des Tweets repräsentiert. Eine indirekte Geolokalisierung von Tweets ist demnach unter Verwendung der entsprechenden Attribute des dem Urheber zugeordneten user-Objektes möglich.

- withheld_in_countries: Dieses Attribut kann ein String-Array aus zweistelligen Ländercodes enthalten. Sobald ein Ländercode innerhalb des Arrays gespeichert ist, ist der Inhalt des Tweets aus dem entsprechenden Land nicht mehr abrufbar.

[16] Vgl. Twitter, 2015 2.

Place

Objekte der Klasse "Place" repräsentieren benannte Orte unterschiedlichen Typs und deren Koordinaten. Auf der Basis der einzigartigen place_id, die jedes Place-Objekt besitzt, kann eine Relation zu einem oder mehreren Tweet-Objekten bestehen.

Places können in Abhängigkeit des Kontextes, in dem sie genutzt werden, unterschiedliche Felder enthalten. Es existiert z. B. ein Feld des Typs String mit der Feldbezeichnung "country", das die Bezeichnung des Landes enthält, in dem sich der benannte Ort befindet. Das Feld kann jedoch möglicherweise auch den Wert NULL oder keinen Wert enthalten bzw. gar nicht vorhanden sein.[17] Nachfolgend werden einige der zur Verfügung stehenden Felder ausführlicher beschrieben.

- attributes: Das Feld „attributes“ kann Metadaten im JSON-Format (siehe folgender Abschnitt) als unicode-formatierte Schlüssel-Wert-Paare enthalten. Die verwendbaren Attribute sind nicht genau vorgegeben. Neben einigen häufig eingesetzten Attributen, wie z. B. „locality“ oder „postal_code“, können eigene Attribute definiert werden, die nur für einen bestimmten Anwendungsfall vorgesehen sind.

```
"attributes": {
    "street_address": "Hellbrunner Straße 34",
    "locality": "Salzburg",
    "iso3": "AT"
}
```

- place_type: Typische Werte dieses Feldes sind z. B. „city”, „country“ oder „neighborhood“.

- name und full_name: Der dem Place-Objekt zugewiesene Name und eine um zusätzliche Informationen ergänzte längere Variante (Z. B. „Salzburg“ und „Salzburg, AT“).

[17] Vgl. Twitter, 2015 3.

JSON und GeoJSON

Die unter Verwendung der Twitter APIs abrufbaren Daten liegen in der JavaScript Object Notation (JSON) vor.[18] Die als Teil der unterschiedlichen Objekte vorhandenen Geodaten entsprechen i.d.R. der GeoJSON Format Spezifikation. Allerdings wird innerhalb der zurückgegebenen Twitter-Daten zuerst die geographische Breite und dann die geographische Länge angegeben.[19] Die GeoJSON Format Spezifikation sieht die umgekehrte Reihenfolge vor.[20]

JSON bezeichnet ein plattform- und programmiersprachenunabhängiges Datenaustauschformat. Es basiert auf dem Konzept der JavaScript Objekt Literale.[21]

Im Rahmen seiner Anwendung werden häufig Objektzustände in String-Literale umgewandelt. Dieser Vorgang wird als Objektserialisierung bezeichnet. Da die JSON-Syntax lediglich eine Untermenge der Javascript-Syntax darstellt, können jedoch nicht alle Javascript-Werte zu JSON-String-Literalen serialisiert werden.[22] Funktionen zur Serialisierung bzw. Deserialisierung JSON-formatierter Objekte sind in den Spezifikationen zahlreicher Programmiersprachen enthalten. Objektbezeichner und Attributbezeichner werden in doppelten Anführungszeichen dargestellt.

```
{
   "auto": [
   {
      "marke": "bmw",
      "farbe": "schwarz",
      "ps"   : "109",
      "zugelassen": false
   }
   ]
}
```

[18] Vgl. Twitter, 2015 4.
[19] Vgl. Twitter, 2015 5.
[20] Vgl. Butler et. al 2008.
[21] Vgl. Basset, 2015 S. 5 ff.
[22] Vgl. Flanagan, 2012 S. 148.

2.3.2 Vergleich der APIs zur Datenakquise

Firehose

Der Daten-Stream, der die Gesamtmenge aller öffentlichen Tweets umfasst, wird als „Twitter Firehose“ bezeichnet. Mit „Datasift“ und „NTT Data“ existieren aktuell lediglich zwei unabhängige Dataprovider, die Echtzeitzugriff auf die gesamte Menge öffentlicher Twitter-Daten haben.[23]

Search API

Als Teil von Twitters v1.1 REST API ermöglicht die Twitter Search API den Zugriff auf eine - i.d.R. durch die Anfrageparameter nicht genau definierbare - Untermenge der innerhalb der letzten Tage generierten Tweet-Daten mithilfe von REST-Anfragen. Unter Verwendung von „OAuth 2.0“ besteht eine Limitierung von 180 Anfragen pro Benutzerprofil innerhalb von 15 Minuten. Alternativ besteht die Möglichkeit einer benutzerunabhängigen Authentifikation der zuvor im Twitter-Account aktivierten Anwendung ("application-only-authentication"). Im Rahmen dieser Authentifikationsmethode sind bis zu 450 Anfragen in 15 Minuten möglich.[24]

[23] Vgl. Hern, 2014.
[24] Vgl. Twitter, 2015 6.

Streaming API

Im Gegensatz zur Twitter Search API besteht bei der Verwendung der Twitter Streaming API eine persistente HTTP-Verbindung. Anders als mithilfe einer REST-Anfrage können gerade generierte Tweet-Daten in Echtzeit bezogen werden. Die abgerufene Datenmenge ist dabei auf ca. 1 % der gesamten Datenmenge ("Firehose") beschränkt. Zusätzlich besteht eine Limitierung durch die zur Verfügung stehenden Anfrageparameter. So können z. B. max. 400 Wortphrasen und / oder 5000 UserIDs zur Definition der abzurufenden Datenmenge verwendet werden.[25]

Die als Teil der Streaming API zur Verfügung stehenden Endpunkte sind in drei Kategorien unterteilt. Die erste Kategorie („Public streams“) wird zum Bezug der innerhalb der Twitter-Plattform öffentlich zugänglichen Daten vorgeschlagen.[26] Alternativ können unter Verwendung der „User streams“ Daten empfangen werden, die mit dem gerade authentifizierten Benutzerkonto in Verbindung stehen. Die dritte Kategorie („Site Streams“) stellt Zugriffspunkte für umfangreiche Anwendungen bereit, die Statusänderungen in Echtzeit für eine große Anzahl angemeldeter Benutzer verarbeiten müssen. Sie befindet sich in einer geschlossenen Beta-Phase. D. h., sie steht aktuell (Juni 2015) nur einem eingeschränkten Benutzerkreis zur Verfügung.[27] Im Rahmen dieser Arbeit wird auf die Kategorie der „Public streams“ zurückgegriffen. Innerhalb dieser Kategorie wird der Zugriffspunkt „POST statuses/filter“ verwendet.

[25] Vgl. Twitter, 2015 7.
[26] Vgl. Twitter, 2015 7.
[27] Vgl. Twitter, 2015 8.

2.3.3 Authentifizierung und Autorisierung mithilfe von OAuth

Das Sicherheitsprotokoll OAuth ermöglicht die Authentifizierung einer Drittanwendung gegenüber einer Webanwendung - wie z. B. Twitter - und die Zugriffsautorisierung dieser Drittanwendung durch einzelne Benutzer der Webanwendung auf die innerhalb ihres Benutzerkontos vorhandenen Daten. Die Weitergabe der eigenen Zugangsdaten wird dabei vermieden. Die Entwicklung von OAuth wurde im Jahr 2006 zunächst von Blaine Cook und Chris Messina initiiert. Eine erste Spezifikation des Protokolls ("OAuth Core 1.0") wurde am 03. Oktober 2007 veröffentlicht.[28] Inzwischen liegt die Spezifikation von OAuth 2.0 als RFC-Standard 6749 vor.[29]

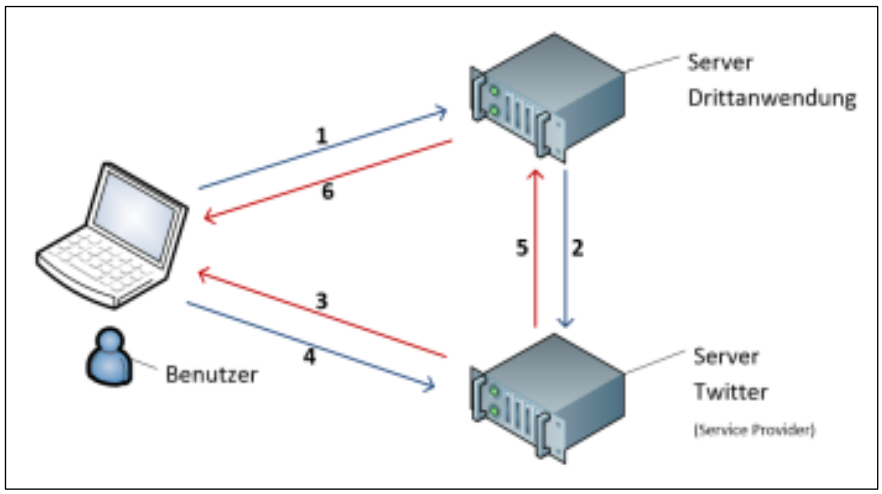


Abbildung 1: Ablaufskizze zur Authentifizierung mithilfe von OAuth

1. Ein Benutzer (der Resourcenbesitzer) sendet eine Anfrage über seine beim Service Provider (in diesem Fall "Twitter") gespeicherten Daten an einen Server, auf dem eine Drittanwendung ausgeführt wird.

2. Die Drittanwendung sendet daraufhin eine Anfrage zur Authorisierung an den Service Provider.

3. Der Service Provider sendet eine Authentifizierungsanfrage an den Benutzer.

[28] Vgl. Hammer, 2007.
[29] Vgl. IETF, 2012.

4. Der Benutzer authentifiziert sich gegenüber dem Service Provider und authorisiert damit die Drittanwendung.

5. Der Service Provider sendet ein Token zur Drittanwendung, wodurch diese authorisiert wird, auf die angefragten Ressourcen zuzugreifen.

6. Die Drittanwendung bestätigt den Zugriff gegenüber dem Benutzer.

2.4 Ontologien

Ursprünglich wurde der Ontologiebegriff im Kontext der theoretischen Philosophie als die Lehre des Seins definiert. Im Bereich der Informatik und der daran angrenzenden wissenschaftlichen Disziplinen, insbesondere auf dem Gebiet der künstlichen Intelligenz, ist der Ontologiebegriff seit Beginn der 90er Jahre gebräuchlich. In diesem Zusammenhang wird eine Ontologie als oberste Abstraktionshierarchie verstanden, die die grundlegenden Regeln zur Modellierung einer Wissensdomäne umfasst.[30] Nach einer etwas allgemeineren Definition können Ontologien als formale, explizite Spezifikationen der Konzeptualisierung einer Wissensdomäne verstanden werden.[31][32]

2.4.1 Formale Repräsentationsformen

Die nachfolgend beschriebenen Repräsentationsformen basieren auf der eXtensible Markup Language (XML). Ursprünglich aus der Standard Generalized Markup Language (SGML) hervorgegangen, wurde ihre erste Spezifikation im Jahr 1998 durch das World Wide Web Consortium (W3C) vorgelegt.[33] Die aktuelle XML-Spezifikation („XML 1.0 Fifth Edition“) wurde im Jahr 2008 durch das W3C veröffentlicht und zuletzt am 07.02.2013 modifiziert.[34]

Die Metasprache XML dient der Beschreibung bzw. Strukturierung von Daten mithilfe von Auszeichnungselementen („tags“), für die bestimmte Regeln definiert sind. Sie stellt die syntaktische Grundlage vieler weiterer, durch das W3C definierter Beschreibungssprachen - wie z. B. RDF und OWL -, dar.[35] I. d. R. wird in der ersten Zeile eines XML-Dokumentes innerhalb der XML-Deklaration die verwendete XML-Version und der zu verwendende Zeichensatz angegeben. Unterhalb der XML-Deklaration enthält ein XML-Dokument Elemente, die aus einem öffnenden und einem schließenden tag bestehen und ineinander verschachtelt werden können. Das öffnende

[30] Vgl. Neches et. al., 1991 S. 40.
[31] Vgl. Gruber, 1993 S.1.
[32] Vgl. Gruninger, Uschold, 1996 S. 5.
[33] Vgl. W3C, 1998.
[34] Vgl. W3C, 2013.
[35] Vgl. Hitzler, et. al., 2008 S. 17.

und schließende tag enthalten den Bezeichner des jeweiligen Elementes. Zusätzlich kann jedes XML-Element innerhalb des öffnenden tags mit Attributen ausgezeichnet werden.

```
<?xml version="1.0" encoding="UTF-8" ?>
   <buecherregal>
        <buch1 auflage="1">
        </buch1>
        <buch2 auflage="3">
        </buch2>
   </buecherregal>
```

Ein XML-Dokument gilt genau dann als validierbar bzw. wohlgeformt, wenn die folgenden Bedingungen erfüllt sind:

- Jedes Element setzt sich aus einem öffnenden Start-tag und einem schließenden End-tag zusammen.
- Start- und End-Tags befinden sich auf derselben Schachtelungsebene.
- Es ist genau ein Wurzelelement vorhanden, in dem alle übrigen Elemente verschachtelt sind.
- Die Bezeichner für Elemente und Attribute enthalten keine Leerzeichen.

Unter Verwendung einer oder mehrerer Dokumenttypdeklarationen (DTD), die nach der XML-Deklaration eingefügt werden, kann dem Dokument ein Vokabular - also eine vorher definierte Menge erlaubter Element- bzw. Attributbezeichner sowie deren hierarchischer Aufbau - zugewiesen werden. Mithilfe eines XML-Parsers kann das Dokument anschließend gegen das angegebene Vokabular validiert werden.[36]

```
<? xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE locations SYSTEM "locations.dtd">
```

[36] Vgl. Geisler, 2009 S. 19.

XML Schema Definitionen (XSDs) bezeichnen eine weitere Methode zur Beschreibung einer XML-Struktur. In einer XSD können folgende Definitionen enthalten sein:

- Elemente und Attribute, die in einem XML-Dokument vorkommen dürfen.
- Die Anzahl und die Reihenfolge von Kind-Elementen (den einem Element hierarchisch untergeordneten Elementen).
- Datentypen der erlaubten Elemente und Attribute.
- Standardwerte bzw. Konstanten der erlaubten Elemente und Attribute.

```
<? xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

I. d. R. werden XML-Schemata hauptsächlich für Dokumente verwendet, die eine eher homogene Struktur aufweisen (z. B. Dokumente mit Adressdaten oder Bücherlisten). Für heterogen strukturierte Dokumente - wie z. B. SVG-, GML- oder HTML-Dokumente - werden hingegen oftmals DTDs eingesetzt.[37]

Um Namenskonflikte bei Element- bzw. Attributbezeichnern zu vermeiden, die ggf. in Abhängigkeit des Kontextes unterschiedliche Bedeutungen haben können, wird innerhalb von XML-Dokumenten häufig zusätzlich auf bestimmte Namensräume verwiesen. Dabei werden mithilfe eines oder mehrerer xmlns-Attribute ein oder mehrere Präfixes angegeben. Diese Zuweisungen können sowohl innerhalb des Wurzelelementes für das gesamte XML-Dokument als auch in jedem untergeordneten Element vorgenommen werden. Im zweiten Fall ist der Namensraum wiederum für jedes Kind-Element gültig.[38]

```
<?xml version="1.0" encoding="UTF-8" ?>
  <root xmlns:a="http://www.beispieldomain.xyz/namensraum_a">
    <child xmlns:b="http://www.beispieldomain.xyz/namensraum_b">
    </child>
  </root>
```

[37] Vgl. Lubkowitz, 2005 S. 871.
[38] Vgl. w3schools.com.

RDF

Das Resource Description Framework (RDF) stellt eine im Gegensatz zu der hierarchischen Struktur von XML auf gerichteten Graphen basierende Form der Wissensrepräsentation dar.[39] Die Beschreibung des RDF-Modells und seine Syntaxspezifikation wurden am 22.02.1999 durch das W3C veröffentlicht.[40] Die aktuelle Spezifikation des RDF-Datenmodells (RDF 1.1) besteht seit dem 25.02.2014.[41] Ein (gerichteter) RDF-Graph besteht aus einem oder mehreren Tripeln der Form „Subjekt, Prädikat, Objekt“.

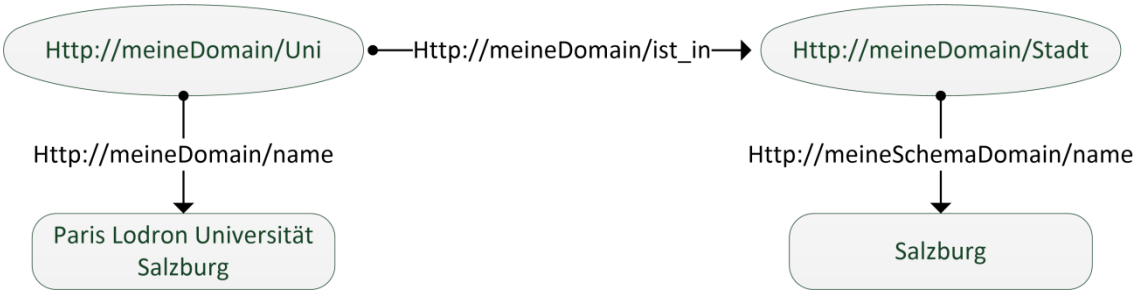


Abbildung 2: Graphendarstellung RDF

Die Knoten des Graphen bilden dabei benannte Ressourcen, die entweder durch „Unified Resource Identifiers“ (URIs) bzw. „Internationalized Resource Identifiers“ (IRIs) dargestellt werden und konkrete Objekte repräsentieren oder in Form von Literalen bestimmte Datenwerte wiederspiegeln (rechteckig abgebildet). Seine Kanten bilden die (binären) Relationen zwischen den durch die Knoten repräsentierten Ressourcen-Objekten ab.

Abbildung 3: Unified Resource Identifier

[39] Vgl. Roth-Berghofer, 2012 S. 109.
[40] Vgl. Lassila, Swick, 1999.
[41] Vgl. Cyganiak, 2014.

Um einen Austausch von RDF-Graphen - z. B. über das World Wide Web - zu ermöglichen, muss die Graphen-Darstellung in eine formale Datenstruktur aus linearen Zeichenketten serialisiert werden.[42] Jedes Tripel wird dabei innerhalb eines RDF-Dokumentes in ein sogenanntes RDF-Statement überführt. Zur Formalisierung von RDF-Graphen existieren mehrere unterschiedliche Syntax-Formen. Da innerhalb der Infrastruktur des World Wide Web die meisten Anwendungen und Systeme mit XML-basierten Parsern ausgestattet sind, wird besonders häufig die am 25.02.2014 durch einen XML-Namensraum spezifizierte RDF-Grammatik „RDF/XML“ eingesetzt.[43] [44] [45]

Innerhalb von RDF/XML-Dokumenten erhält das Wurzelelement immer den Bezeichner rdf:RDF. Mithilfe des Attributes xmlns wird dem Präfix rdf der entsprechende URI des RDF/XML-Namensraumes zugewiesen. Die einzelnen RDF-Statements werden entsprechend der in XML vorgegebenen hierarchischen Struktur ineinander verschachtelt. Dabei werden Relationen (Prädikatelemente) zwischen dem öffnenden und schließenden tag eines Subjektelementes ebenfalls als Elemente formalisiert, die jeweils ein oder mehrere Objektelemente enthalten können. Auf diese Weise können innerhalb eines Subjektelementes eine oder mehrere Relationen zu einem oder mehreren Objektelementen abgebildet werden.[46] Außerdem können Ressourcen durch URI-Referenzen auf die in der RDF/XML-Spezifikation angegebenen Datentypen typisiert werden. Dazu werden Ressourcen unter Verwendung der URI rdf:type als Instanzen einer bestimmten Klasse gekennzeichnet.[47]

```
<?xml version="1.0" encoding="utf-8"?>
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:ex="http://example.com/myNamespace">
  </rdf:RDF>
```

[42] Vgl. Hitzler et. al., 2008 S. 40.
[43] Vgl. Gandon, Schreiber, 2014.
[44] Vgl. Segaran et. al., 2009 S. 69.
[45] Vgl. Allemang, Hendler, 2008 S. 53.
[46] Vgl. Hitzler et. al., 2008 S. 42-45.
[47] Vgl. Hitzler et. al., 2008 S. 68.

Die Terse RDF Triple Language (Turtle) ist eine weitere weit verbreitete, formale Sprache zur Notation von RDF-Statements. Sie stellt eine Teilmenge der von Tim Berners Lee entworfenen Notation 3 (N3) dar.[48] Turtle wurde am 25.02.2014 durch das W3C als Empfehlung verabschiedet und damit als ein Standard etabliert.[49] In der einfachsten Form wird ein RDF-Tripel damit als eine durch Leerzeichen getrennte Abfolge von URIs bzw. IRIs oder Literalen und einen abschließenden Punkt dargestellt. Indem ein Statement anstelle eines Punktes mit einem Semikolon abgeschlossen wird, können einem Subjekt in Form einer Prädikatliste mehrere Prädikat-Objekt-Paare zugewiesen werden. Ebenfalls können einem Prädikat mithilfe von Kommata mehrere Objekte in Form einer Objektliste zugewiesen werden.

Um die Lesbarkeit des Quellcodes zu erhöhen, können Teile des URIs zu Beginn eines Dokumentes unter Angabe von @prefix einem Präfix zugeordnet werden.

```
@prefix ex:<http://example.org/>.
    <http://example.com/#student1> ex/name "Alice".
    <http://example.com/#student2> ex/name "Bob".
                                   <http://example.com/title> "B.Sc.", "M.Sc.".
```

[48] Vgl. Berners-Lee, T., 2011.
[49] Vgl. Beckett et. al., 2014.

RDFS

RDF Schema (RDFS) bildet eine semantische Erweiterung von RDF, die ein Klassen- und Eigenschaftensystem in Anlehnung an Konzepte aus der objektorientierten Programmierung (OOP) definiert. Anders als in der OOP werden jedoch nicht Klassen durch die Eigenschaften definiert, die ihre Instanzen aufweisen können, sondern Eigenschaften (rdf:Property) durch die Ressourcenklassen beschrieben, für die sie gelten.[50] Dazu dienen in rdfs die Properties (Instanzen von rdf:Property) „rdfs:domain“ und „rdfs:range“, mit denen der Definitions- bzw. Wertebereich von Relationen eingeschränkt werden kann. Z. B. impliziert eine Aussage der Form "P rdfs:domain C", dass alle in Tripeln mit Prädikat P als Subjekt enthaltenen Ressourcen Instanzen der Klasse C darstellen. Ein Statement der Form „P rdfs:range C“ bedeutet analog dazu, dass alle Objektressourcen innerhalb eines Tripels mit Prädikat P Instanzen der Klasse C sind.[51]

Innerhalb von RDFS werden alle Ressourcen von der Klasse rdfs:Resource abgeleitet. Unter Verwendung von rdf:type kann eine Ressource als Instanz einer bestimmten Klasse beschrieben werden. Die rdfs:domain von rdf:type ist demnach rdfs:Resource. Die rdfs:range von rdf:type ist rdfs:class.[52] Außerdem beschreibt die RDFS-Spezifikation die Properties rdfs:subPropertyOf und rdfs:subClassOf, mit denen Teilrelationen und Vererbungen abgebildet werden können.[53] Eine vollständige Liste der innerhalb von RDFS vorhandenen RDF-Klassen und RDF-Properties findet sich in der RDFS-Spezifikation.[54]

[50] Vgl. Brickley, Guha, 2014.
[51] Vgl. Brickley, Guha, 2014.
[52] Vgl. Brickley, Guha, 2014.
[53] Vgl. Stuckenschmidt, 2011 S. 106.
[54] Vgl. Brickley, Guha, 2014.

OWL

Die aktuelle Version der Web Ontology Language (OWL bzw. OWL2) wurde am 11.12.2012 als Handlungsempfehlung durch die OWL Working Group des W3C veröffentlicht.[55] Zunächst kann eine OWL2-Ontologie unabhängig von der zum Datenaustausch verwendeten, konkreten Syntax als formale Beschreibung einer Wissensdomäne betrachtet werden, die jedoch immer die drei folgenden syntaktischen Kategorien enthält:[56]

- Durch IRIs eindeutig identifizierbare Entitäten (z. B. Klassen, Eigenschaften, Individuen).

- Ausdrücke, die bestimmte Gegebenheiten innerhalb der beschriebenen Domäne repräsentieren (z. B. Einschränkungen, die für eine bestimmte Klasse von Individuen Gültigkeit haben).

- Axiome, die innerhalb der beschriebenen Wissensdomäne als wahr aufgefasst werden (z. B. kann angenommen werden, dass die Klasse „a:Student“ eine Unterklasse der Klasse „a:Person“ ist).

Diese drei syntaktischen Kategorien geben die logische Struktur einer OWL2 Ontologie vor, die automatisierte Schlussfolgerungen auf semantischer Ebene (Inferenzierung) ermöglicht. Z. B. kann aus der Tatsache, dass „a:Tuffi“ eine Instanz der Klasse „a:Elefant“ und dass „a:Elefant“ eine Unterklasse von „a:Säugetier“ ist, geschlossen werden, dass „a:Tuffi“ ebenfalls eine Instanz der Klasse „a:Säugetier“ sein muss.

Zusätzlich sind in OWL2 Annotationen vorgesehen, die lediglich der besseren Lesbarkeit durch menschliche Benutzer dienen und keinerlei Auswirkungen auf die logische Struktur einer OWL2 Ontologie haben. OWL stellt zwar eine Untermenge der Prädikatenlogik erster Stufe (PL1) dar; anders als in prädikatenlogischen Aussagen sind

[55] Vgl. W3C OWL Working Group, 2012.
[56] Vgl. Motik et. al., 2012.

in OWL jedoch ausschließlich ein- und zweistellige Prädikate möglich. Dadurch sind die getroffenen Aussagen grundsätzlich entscheidbar.[57] [58]

Für die Formalisierung der an dieser Stelle beschriebenen, grundlegenden logischen Struktur einer OWL Ontologie sind unterschiedliche Syntaxen bzw. Notationsformen spezifiziert. Die Standardsyntax, die von allen gängigen OWL2-konformen Werkzeugen zum Austausch von OWL2-Daten unterstützt wird, ist RDF/XML.[59] Jede OWL2-Ontologie lässt sich demnach auf RDF-Graphen abbilden.[60]

Um die Anforderungen unterschiedlicher Anwendungsfälle zu erfüllen, werden folgende OWL2-Profile als Untermengen der vollständigen OWL2-Syntax definiert:[61]

- OWL 2 EL: Als das umfangreichste OWL2-Profil ermöglicht OWL 2 EL den Einsatz von in Polynomialzeit lösbaren Algorithmen für alle zur Deduktion bzw. Induktion gebräuchlichen Anwendungsfälle. Aufgrund der Ausdrucksstärke dieses Profils erscheint es jedoch innerhalb mancher Anwendung weniger performant als die beiden nachfolgend beschriebenen Profile. Es eignet sich insbesondere für sehr große Ontologien.

- OWL 2 QL: OWL 2 QL ermöglicht in logarithmischer Platzkomplexität zu beantwortende, konjunktive Anfragen auf der Grundlage einer in relationalen Datenbank-Management-Systemen (RDBMS) einsetzbaren Anfragesprache (z.B. SQL). Dieses Profil wird für Ontologien überschaubarer Komplexität, die jedoch eine große Anzahl von Individuen enthalten, empfohlen.

- OWL 2 RL: Genau wie OWL 2 QL eignet sich das Profil OWL 2 RL für eher leichtgewichtige Ontologien. Allerdings ist es vor allem für die Implementierung in Polynomialzeit lösbarer Algorithmen auf der Basis von direkt auf RDF-Tripeln ausführbaren Anfragen vorgesehen (z. B. SPARQL).

[57] Vgl. Kreuzer, Kühling, 2006 S. 64 ff.
[58] Vgl. Roth-Berghofer, 2012 S. 127 f.
[59] Vgl. W3C OWL Working Group, 2012.
[60] Vgl. Patel-Schneider et. al., 2012.
[61] Vgl. W3C OWL Working Group, 2012.

2.4.2 Repräsentation von GML-Daten in OWL

Die Geography Markup Language (GML) bezeichnet eine in XML-Schema formalisierte XML-Grammatik zur strukturierten Beschreibung geographischer Information. Genau wie andere XML-Grammatiken besteht die GML-Grammatik aus einem Schema-Dokument, das das zu verwendende Vokabular beschreibt, und einem oder mehreren Instanz-Dokumenten mit den konkreten geographischen Daten.[62] GML 1.0, die erste Version der GML-Spezifikation, wurde im Mai 2000 durch das Open Geospatial Consortium (OGC) veröffentlicht. Im Gegensatz zu den ab Version 2.0 verwendeten XML Schemata wurde die GML Grammatik in dieser Version noch mithilfe eines RDF Schemas und Dokumenttypdefinitionen (DTDs) beschrieben.[63]

In GML 1.0 können einfache geographische Features abgebildet werden, deren Attributwerte einfachen Geometrien entsprechen.[64] Das der GML 1.0 Spezifikation zugrundeliegende Geometriemodell enthält Definitionen einfacher geometrischer Formen - wie z. B. Punkt, Linie oder Polygon - sowie ein Koordinatenelement und ein Box-Element, mit dem die geographische Ausdehnung geographischer Objekte definiert werden kann. Zusätzlich ist es möglich, anwendungsspezifische Erweiterungen mit zusätzlichen semantischen Regeln zu definieren (z. B. distinkte Objektklassen für Flüsse und Straßen).[65]

[62] Vgl. OGC, 2012.
[63] Vgl. OGC, 2000.
[64] Vgl. Pospech, 2008 S. 5.
[65] Vgl. Van den Brink et. al., 2013 S. 251.

```
<?xml version "1.0" encoding "UTF-8" ?>
  <gml:FeatureCollection
  xmlns:xsi "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation "http://meineDomain/lmeinSchema.xsd"
  xmlns:gml "http://www.opengis.net/gml">
    <gml:featureMember>
      <location fid "123456">
        <geometryProperty>
          <gml:Point srsName "EPSG:4326">
            <gml:coordinates>-0.1275 51.5072</gml:coordinates>
          </gml:Point>
        </geometryProperty>
      </location>
    </gml:featureMember>
  <gml:FeatureCollection>
```

Da GML ursprünglich auf RDF bzw. auf RDFS basierte, und auch in aktuellen Versionen zahlreiche RDF-Elemente aufgreift, ist die Repräsentation von GML-Schemata und GML-Features in OWL relativ einfach möglich.[66] Die Objektstruktur, in der Objekte Attribute enthalten, deren Werte entweder einfache Literale oder wiederum andere Objekte darstellen können, ähnelt der für RDF-Daten charakteristischen Tripelstruktur. Diese Struktur ist auch in aktuellen Versionen der GML Spezifikation erhalten geblieben.[67]

[66] Vgl. Lieberman, 2006 S. 40 ff.
[67] Vgl. Van den Brink et. al., 2013 S. 252.

2.4.3 Strukturierte Anfragen mit SPARQL und GeoSPARQL

Die SPARQL Protocol And RDF Query Language (SPARQL) wird seit dem 15.01.2015 durch das W3C als Standardsprache für Anfragen auf RDF-Graphen empfohlen.[6869] Die syntaktische Struktur einer Anfrage ähnelt dabei in vielen Bereichen dem Aufbau einer in der Structured Query Language (SQL) formulierten Anfrage. Die meisten SPARQL-Anfragen enthalten ein mithilfe eines SELECT-Befehls und einer dazugehörigen WHERE-Klausel strukturiertes Tripel-Muster, das dem Aufbau eines RDF-Tripels (Subjekt, Prädikat, Objekt) entspricht. Jedes Element dieses Tripel-Musters kann durch eine mit „?“ gekennzeichnete Variable ersetzt werden, deren für die innerhalb der WHERE-Klausel formulierte Bedingung gültiger Wert im SELECT-Teil angefragt wird.[70] [71] Die innerhalb der WHERE-Klausel verwendete Notation ähnelt der Turtle-Syntax. Zusätzlich verwendet sie jedoch die in N3 definierte Klammersyntax, um Teilgraphen abzugrenzen. Auf diese Weise können auch komplexe Anfragebedingungen formuliert werden.

```
PREFIX ont: <http://example.org/myOntology/>

PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?lat ?long WHERE {

   ?s foaf:name "Salzburg” .

   ?s a ont:city .

}
```

[68] Vgl. Sintek, 2012 S. 161.
[69] Vgl. Prud’hommeaux, Seaborne, 2008.
[70] Vgl. Allemang, Hendler, 2008 S. 67.
[71] Vgl. Segaran et. al., 2009 S. 86.

Am 10.09.2012 wurde GeoSPARQL als Standard durch das OGC veröffentlicht, der SPARQL um zahlreiche Funktionen zum Umgang mit geographischen Informationen erweitert.[72] GeoSPARQL definiert dazu ein eigenes Vokabular zur Repräsentation geographischer Daten in RDF und zusätzlich eine Erweiterung von SPARQL, mit der Anfragen auf diesen Daten ausgeführt werden können. Der GeoSPARQL-Standard umfasst dabei folgende Komponenten bzw. Erweiterungen:[73]

- Eine Kernkomponente, die übergeordnete RDFS/OWL-Klassen für räumliche Objekte definiert.
- Eine Topologie-Vokabular-Erweiterung, die die Zuweisung und Anfrage topologischer Relationen zwischen räumlichen Objekten durch RDF Properties definiert. Für jede als Teil dieser Erweiterung definierte Relation - wie z. B. "geo:sfIntersects" - gilt der Definitions- bzw. Wertebereich ("Domain/Range") "geo:SpatialObject". Bei den Features, deren Relation abgebildet wird, handelt es sich also immer um Instanzen der Klasse "geo:SpatialObject". Dies gilt für Relationen der "Simple Features"-, „Egenhofer“- und „RCC8“-Familie.[74]
- Eine Geometriekomponente mit RDFS-Datentypen zur Serialisierung geometrischer Daten, geometriebezogener RDF Properties und nicht-topologischer, räumlicher Anfragefunktionen für Geometrieobjekte.
- Eine Geometrie-Topologie-Komponente, die topologische Anfragefunktionen definiert.
- Eine RDFS-Komponente zur semantischen Implikation von RDF Tripeln, die aus bestehenden RDF- bzw. RDFS-Semantiken abgeleitet werden.
- Eine Komponente zur Transformation einfacher, zum Testen topologischer Relationen zwischen zwei Features genutzter Tripelmuster in äquivalente Anfragen an konkrete Geometrien und topologische Anfragefunktionen.

[72] Vgl. Herring, Perry, 2012 S. 1.
[73] Vgl. Herring, Perry, 2012 S. 16.
[74] Vgl. Herring, Perry, 2012 S. 26 - 28.

In Abhängigkeit der funktionalen Anforderungen müssen nicht alle dieser Komponenten in ein System zur räumlichen Inferenzierung implementiert sein. Ein System, das ausschließlich der qualitativen räumlichen Schlussfolgerung dient, müsste z. B. lediglich die Kernkomponente sowie das topologische Vokabular umfassen.[75]

```
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
SELECT ?s WHERE {
    ?s geo:hasGeometry ?geometry .
    FILTER(geof:sfWithin(?geometry,
                "POINT(-77.089005 38.913574)
               geo:wktLiteral))
}
```

[75] Vgl. Herring, Perry, 2012 S. 17.

2.5 Natural Language Processing und Information Extraction

Der Begriff Natural Language Processing (NLP) wird innerhalb der englischsprachigen Literatur häufig synonym zu dem Begriff Computational Linguistics (CL) verwendet.[76] Teilweise wird NLP jedoch auch als eine von mehreren Ausprägungen bzw. Teildisziplinen der Computerlinguistik beschrieben.[77] NLP kann allgemein als die maschinelle Verarbeitung natürlicher Sprache verstanden werden. Natürliche Sprachen, die sich über einen längeren Zeitraum entwickelt haben (z. B. Englisch oder Deutsch), werden dabei i. d. R. von "konstruierten", "künstlichen" bzw. eher formal strukturierten Sprachen - wie z. B. Programmiersprachen - abgegrenzt.[78] [79] Ein Schwerpunkt innerhalb des NLP liegt heute im Bereich des maschinellen Lernens. Dabei werden Algorithmen entwickelt, die es ermöglichen, unter Verwendung impliziten Wissens linguistische Annotationen zu generieren.[80]

Den meisten NLP-Anwendungen liegt eine Architektur zugrunde, deren einzelne Prozesse unter Einbeziehung zahlreicher Komponenten sequentiell durchgeführt werden. Man spricht deshalb häufig von einer NLP-Pipeline.[81]

Die den einzelnen Verarbeitungsschritten zugrundeliegenden Algorithmen lassen sich in überwachte Lernalgorithmen und unüberwachte Lernalgorithmen unterteilen. Bei der Verwendung überwachter Algorithmen sind die möglichen Attribute, anhand derer eine Auszeichnung vorgenommen wird, vor dem ersten Durchlauf bekannt. Gewöhnlich liegt dabei eine unter Verwendung dieser Attribute ausgezeichnete Datenmenge - ein sog. Trainingskorpus - vor.[82] Im Folgenden werden einige NLP-Komponenten beschrieben, die im Kontext dieser Arbeit besonders relevant erscheinen.

[76] Vgl. Kumar, 2011 S. 1.
[77] Vgl. Carstensen et. al., 2010 S. 2 f.
[78] Vgl. Kapetanios et. al., 2014 S. 6.
[79] Vgl. Jackson, Moulinier, 2002 S. 13 f.
[80] Vgl. Biemann, 2012 S. 1.
[81] Vgl. Shi et. al., 2014 S. 53.
[82] Vgl. Zafarani et. al., 2014 S. 113 f.

2.5.1 Tokenisierung

Der Begriff Tokenisierung bezeichnet die Zerlegung eines Textes in linguistische Einheiten. Innerhalb segmentierter Schriftsysteme, denen z. B. das lateinische, kyrillische oder griechische Alphabet zugrundeliegt, kann ein Wort als eine Einheit alphanumerischer Zeichen charaktierisiert werden, die auf beiden Seiten durch Leerzeichen oder Interpunktion begrenzt wird.[83] In Abhängigkeit seiner weiteren Verwendung innerhalb einer NLP-Pipeline muss ein Token jedoch nicht zwangsläufig einem einzelnen Wort entsprechen. Innerhalb einer Anwendung zur maschinellen Übersetzung kann es z. B. sinnvoll erscheinen, aus mehreren Wörtern zusammengesetzte Ausdrücke zu einem Token zusammenzufassen.[84] Token können demnach auch Sätzen, Phrasen, Wörtern oder Morphemen entsprechen.[85]

Die Tokenisierung von Microblog-Texten - wie z. B. Tweets - wird durch die begrenzte Zeichenmenge und die oftmals sehr spezifische Schreibweise erheblich erschwert. Prozesse der Tokenisierung können entweder auf symbolischen oder auf statistischen Verfahren basieren. Symbolischen Verfahren liegen Heuristiken zugrunde, die als Liste regulärer Ausdrücke formalisiert sind. Innerhalb statistischer Verfahren werden die morphosyntaktischen Merkmale eines Wortes analysiert und dessen Kontext in n-gramm-Modellen beschrieben. Z. B. wird in einem Bigramm-Modell die Wahrscheinlichkeit dafür erfasst, dass ein Wort in Abhängigkeit zu einem voranstehenden Wort auftritt.[86] Entsprechende Modelle werden i. d. R. anhand größerer Textmengen erstellt, die dem zu verarbeitenden Textabschnitt ähneln (Trainingskorpora).

Die Genauigkeit statistischer Verfahren ist vergleichbar mit der Genauigkeit symbolischer Verfahren. Allerdings sind statistische Verfahren flexibler einsetzbar, da ihnen kein statisches Regelwerk zugrunde liegen muss. Für die Anwendung auf Microblog-Texte eignen sich deshalb insbesondere statistische Verfahren der Tokenisierung.

[83] Vgl. Hagenbruch, 2010 S. 264 f.
[84] Vgl. Laboreiro et. al., 2010 S. 4.
[85] Vgl. Sproat, 2008 S. 459.
[86] Vgl. Hagenbruch, 2010 S.268 ff.

2.5.2 Gazetteering

In manchen Kontexten wird der Begriff Gazetteer in erster Linie synonym zu den Begriffen Ortslexikon oder Ortsverzeichnis verwendet und bezeichnet in diesem Zusammenhang eine Liste geographischer Identifikatoren (Toponyme), die Referenzen auf die räumlichen Koordinaten oder andere Informationen einer entsprechenden geographischen Bezugseinheit bilden. Ein Gazetteer kann demnach als ein Verzeichnis von Instanzen einer oder mehrerer Feature-Klassen definiert werden, das Informationen bezüglich der Position der einzelnen räumlichen Objekte enthält.[87] Häufig wird Gazetteering auch als eine vergleichsweise einfache Methode der Eigennamenerkennung (NER) charakterisiert. Ein Gazetteer könnte demzufolge auch allgemeiner als eine Liste von Eigennamen benannter Entitäten verstanden werden, die Referenzen zu bestimmten Merkmalsausprägungen dieser Entitäten enthält.[88]

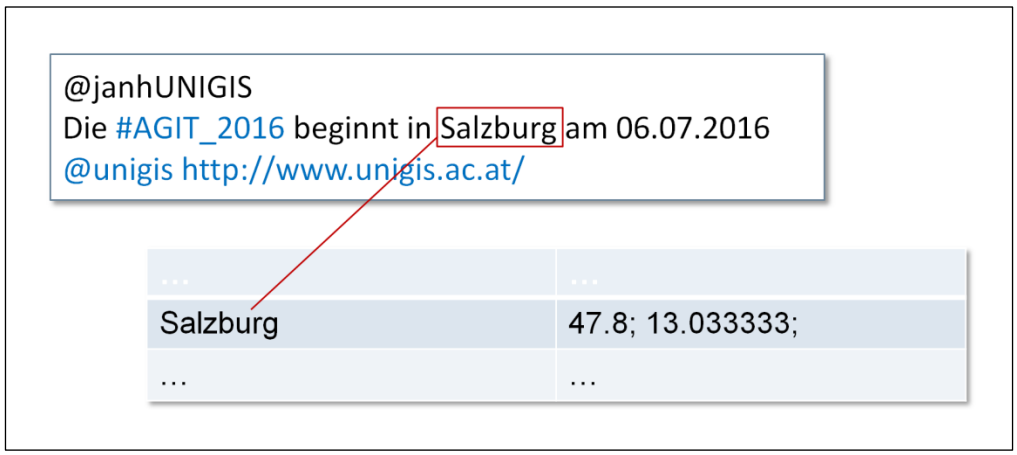


Abbildung 4: Gazetteering in Tweet-Texten

Gazetteering wird zwar häufig als ein (erster) Teilprozess der NER verwendet; ohne zusätzliche Methoden erscheint das Verfahren jedoch für die meisten Anwendungsfälle nicht ausreichend genau. So bestehen z. B. sehr hohe Anforderungen an die Vollständigkeit eines Gazetteers im Kontext der jeweiligen Anwendungsdomäne. Jedoch können Redundanzen das Ergebnis wiederum negativ beeinflussen. Ein weiteres

[87] Vgl. ISO, 2003 S. 2.
[88] Vgl. Ziegler, 2012 S. 27 f.

Problem stellen ggf. mehrfach vorhandene Eigennamen (Ambiguitäten) dar.[89] Der Eigenname "Washington" bezeichnet z. B. sowohl einen Bundesstaat sowie diverse andere geographische Entitäten innerhalb und außerhalb der USA, als auch einen ehemaligen US-Präsidenten. Laut unterschiedlicher Schätzungen sind ca. 1/3 aller Toponyme nicht eindeutig einer geographischen Bezugseinheit zuzuordnen.

Kontinent	**Orte mit mehreren Ortsnamen**	**Ortsnamen, die mehrere Orte beschreiben**
Nord- & Zentralamerika	11,5 %	57,1 %
Ozeanien	6,9 %	29,2 %
Südamerika	11,6 %	25,0 %
Asien	32,8 %	20,3 %
Afrika	27,0 %	18,2 %
Europa	18,2 %	16,6 %

Tabelle 2: Ambiguitäten innerhalb von Toponymen[90]

Besonderheiten des Gazetteerings innerhalb von Tweettexten

Ein zur NER in Tweets verwendetes Gazetteer weist aufgrund des geringen Textumfangs des Eingangskorpus (140 Zeichen) besondere Anforderungen auf, da Tweets häufig eine weniger formale Struktur aufweisen als andere Texte. Toponyme werden z. B. innerhalb von Tweets weitaus häufiger abgekürzt als innerhalb von Zeitungsartikeln.[91]

[89] Vgl. Neumann, 2010 S. 596.
[90] Smith, Crane, 2001 S. 131.
[91] Vgl. Beisecker, 2015.

2.5.3 Normalisierung von Wortformen

Da zumeist nicht jede auftretende Form eines Wortes mithilfe von NLP-Methoden untersucht werden kann - z. B. sind innerhalb eines Lexikons i. d. R. nur einzelne Wortformen gespeichert - können unterschiedliche Wortformen auf ihre Stammform reduziert werden. Dieser als Normalisierung bezeichnete Vorgang wird häufig unter der Verwendung sog. Stemming-Algorithmen durchgeführt.

Stemming-Verfahren werden verwendet, um eine lexikonfreie Suffixanalyse durchzuführen. Dabei werden jedoch auch semantisch unterschiedliche Wortformen mit zum Teil unterschiedlichen Suffixen auf denselben Stamm reduziert. Z. B. würden die Worte "Buche", "Buchen", "Buch" und "Bucht" auf den String "Buch" abgebildet.[92] Die Normalisierung bzw. die Stammformbildung der in einem Text vorhandenen Worte impliziert also zumeist zusätzliche Ungenauigkeiten und Ambiguitäten.

[92] Vgl. Neumann, 2010 S. 589.

2.5.4 Part of Speech Tagging

Im Rahmen des Part of Speech Tagging ("POS Tagging") werden die in den vorangegangenen Verarbeitungschritten ermittelten Worte anhand ihrer Wortart klassifiziert und entsprechend ausgezeichnet ("Tagging"). Dabei kann sowohl die Form eines Wortes (Morphologie) als auch seine (grammatikalische) Funktion innerhalb eines Satzes oder einer Phrase (Syntax) identifiziert werden.[93] Man spricht deshalb auch von morphosyntaktischer Klassifizierung.[94]

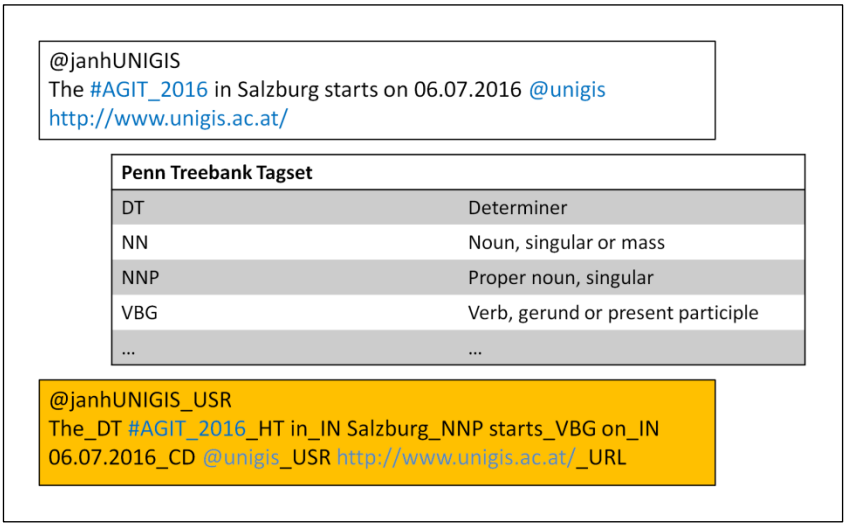


Abbildung 5: Part of Speech Tagging unter Verwendung des Penn Treebank Tagsets

[93] Vgl. de la Higuera, 2010 S. 28.
[94] Vgl. Paroubek, 2008 S. 99 f.

2.5.5 Named Entity Recognition (NER)

Der Erkennung benannter Entitäten kommt im Rahmen des NLP eine herausragende Bedeutung zu, da praktisch jede Anwendung auf dem Gebiet der semantischen Analyse bzw. der Modellierung semantisch angereicherter Datenstrukturen korrekt annotierte Entitäten voraussetzt. Die Bedeutung eines Tripels aus Subjekt, Prädikat und Objekt kann z. B. nur dann automatisch erfasst werden, wenn Subjekt und Objekt als Instanzen bestimmter Klassen erkannt wurden, da nur auf diese Weise die zwischen ihnen möglichen Relationen festzustellen sind.

Grundlegende Methoden zur Erkennung benannter Entitäten weisen für gewöhnlich bei der Anwendung auf längere Texte eine Genauigkeit von 85-90% auf. Wendet man die gleichen Methoden auf Tweets an, so beträgt die Genauigkeit lediglich 30-50%.[9596] Durch die Verwendung Ontologie-basierter – also Kontext-gestützter – NER bzw. Clustering-Verfahren kann die Genauigkeit jedoch gesteigert werden. Clustering-Algorithmen teilen Objekte basierend auf ihrer Ähnlichkeit bezüglich der analysierten Attribute in Klassen (clusters) ein.[97]

[95] Vgl. Ritter et. al., 2011 S. 1528 ff.
[96] Vgl. Liu et. al., 2012 S. 533.
[97] Vgl. Han, Kamber, 2006 S. 444.

3 Akquise von Twitter-Daten unter Verwendung der Streaming API

Um die benötigten Daten lokal zu speichern, wird eine Anwendung erstellt, die eine persistente Verbindung zur Twitter Streaming API aufbaut bzw. aufrechterhält und die mithilfe der jeweiligen Parameter angefragten Ergebnisdaten in Echtzeit in einer Datenbank speichert. Aufgrund der gegebenen Anforderungen und der freien Verfügbarkeit wird MySQL als Datenbank Management System (DBMS) und PHP als Programmiersprache ausgewählt. Ab Version 1.1 der Twitter API können Daten grundsätzlich nur unter Verwendung von OAuth über die zur Verfügung stehenden Schnittstellen angefragt werden. Die im Rahmen dieser Arbeit durchgeführten Anfragen an die Twitter API setzen lediglich die Authentifizierung der Anwendung voraus. Die benötigten Token müssen jedoch unter Verwendung eines Benutzerkontos generiert und die Anwendung dadurch mit diesem Konto assoziiert werden. Dazu meldet man sich in der Anwendungsverwaltung ("apps.twitter.com") mit einem bestehenden Twitter-Account an und erstellt eine neue Anwendung ("APP").

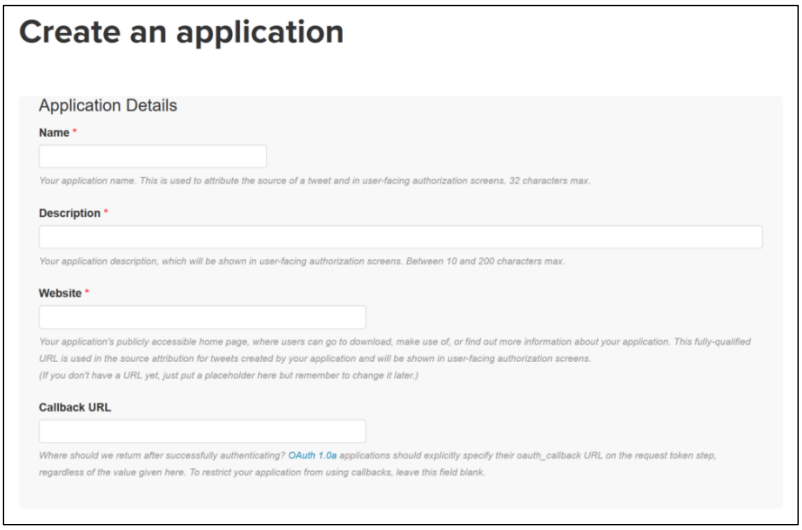


Abbildung 6: Erstellung einer Twitter App

Während des Erstellvorgangs müssen ein Name, eine Beschreibung und der Fully Qualified Domain Name (FQDN) der Webanwendung angegeben werden. Falls die Anwendung nicht unter einer öffentlichen Domain erreichbar sein soll, kann an dieser

Stelle ein Platzhalter eingetragen werden. Optional kann man in das Feld „Callback URL“ eine Adresse eintragen, die aufgerufen wird, sobald die Anwendung erfolgreich authentifiziert und autorisiert wurde.

Nach Abschluss der Konfiguration werden die vier für die Authentifizierung bzw. Autorisierung mit OAuth benötigten alphanumerischen Zeichenketten nach dem Muster der folgenden Beispielcodes generiert:

- **Consumer Key:** j6iMsA8iGXGeC3VpXR78JBZTC
- **Consumer Secret:** vxc6hx8Dz2p7BNkEnqyWeMeMjRu2xPmI30l5gEGXyjkoZU19QD
- **Access Token:** 2808721200-ewiXF6I6TELqs1F6R7D3TscbrBQpnVlvyEQowA2
- **Access Token Secret:** lxrYhfT0jMEjp2sYYOBBTfBWPQoHFhQ6zfiLT1U2FVBRd

Zum Aufbau einer Verbindung mit der Streaming API müssen die beiden letzten Zeichenketten innerhalb der Anwendung als Verbindungsparameter übergeben werden.

3.1 Datenbank Schema

Innerhalb der Datenbank werden zunächst lediglich vier Tabellen benötigt. Die erste Tabelle umfasst ein Textfeld, in dem die JSON-codierten Tweet-Objekte inkl. der zugehörigen User-Objekte und evtl. vorhandener Place-Objekte, die von der Streaming API gesendet werden, als base64-String gespeichert werden. Als Primärschlüssel wird ein Feld des Typs BIGINT(20) angelegt, in dem die tweet_id des jeweiligen Tweet-Objektes gespeichert wird. Die Felder der anderen drei Tabellen entsprechen den für diesen Anwendungsfall benötigten Attributen der drei Objektklassen. Die Primärschlüssel der Tabellen bilden die Felder „tweet_id“, „user_id“ und „place_id“. Die Felder „place_id“ und „user_id“ sind jeweils in der Tabelle „tweet“ als Fremdschlüssel vorhanden.

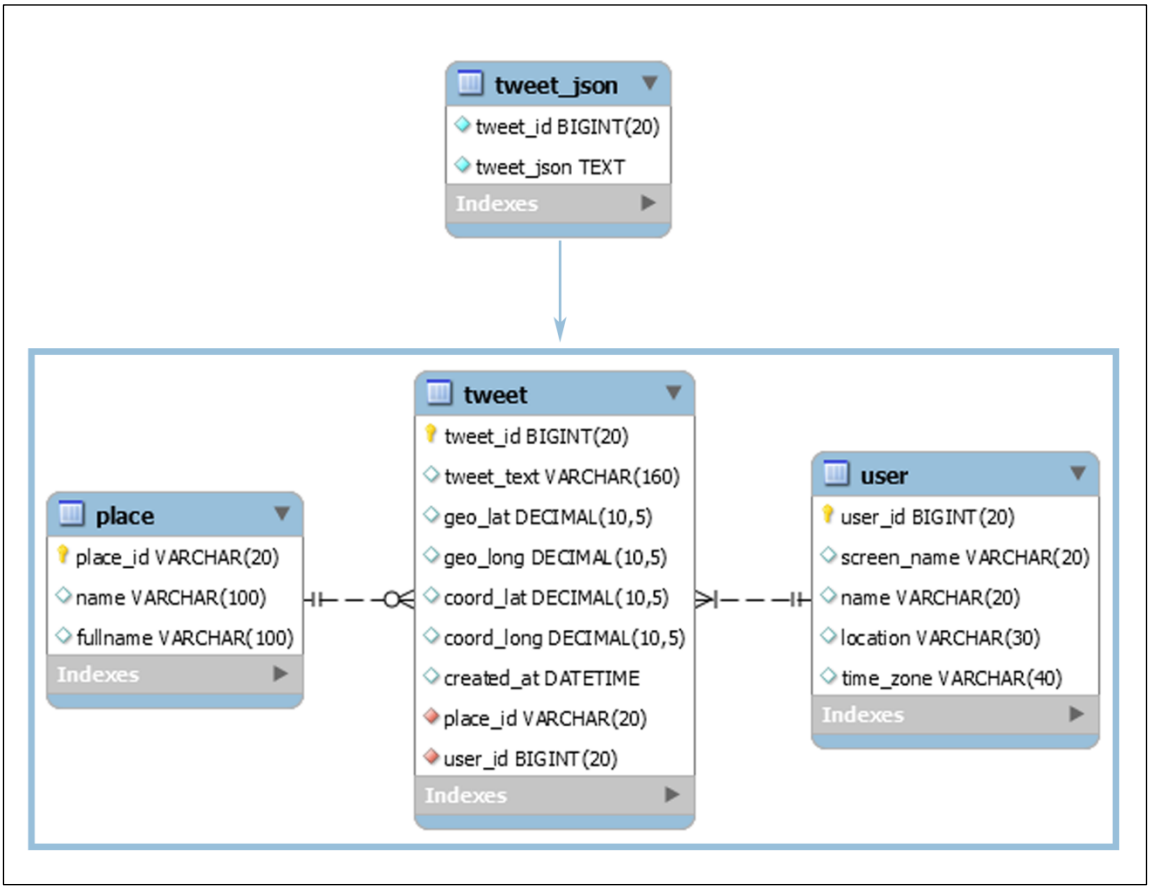


Abbildung 7: Datenbank Schema zur persistenten Speicherung serialsierter Tweet-, User- und Place-Objekte

Das in den nachfolgenden Kapiteln vorgestellte NLP-Framework Twitie kann Eingangsdaten im JSON-Format verarbeiten. Unter Verwendung des GATE-Plugins "Format_Twitter" können die Text-Attribute der einzelnen Tweet-Objekte in ein zur weiteren Verarbeitung benötigtes Textkorpus umgewandelt werden. Jeder Tweet-Text wird dabei mit einer Liste annotiert, die die übrigen innerhalb des JSON-Objektes vorhandenen Schlüssel-Wert-Paare enthält.[98] Um vorab eine Analyse der innerhalb von Tweets vorhandenen Geodaten durchführen zu können, werden die mit einem Tweet assoziierten Objekte und deren für die Analyse maßgeblichen Attribute zusätzlich in Textform ("serialisiert") gespeichert.

[98] Vgl. Bontcheva et. al., 2013 S. 84.

3.2 Anwendungsarchitektur

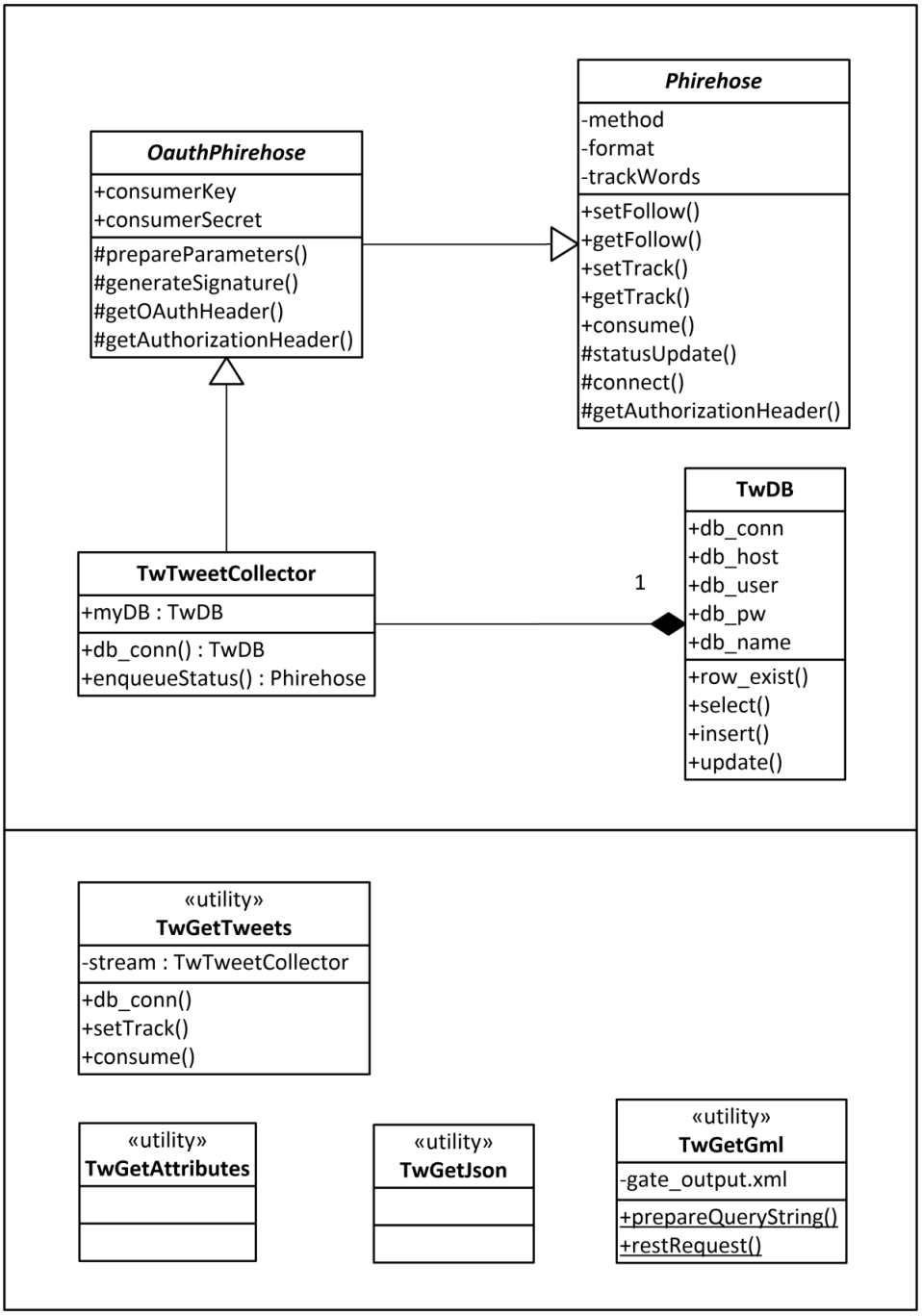


Abbildung 8: UML Klassendiagramm der im Rahmen der Arbeit entwickelten Anwendung

Die PHP-Anwendung wird auf der Basis der frei verfügbaren, abstrakten Klasse "Phirehose" und der abstrakten Klasse "OauthPhirehose" implementiert, die die Klasse „Phirehose“ erweitert.[99] Die Klasse „Phirehose“ stellt unter Anderem unterschiedliche set-Methoden zur Verfügung, mit denen z. B. die Filter-Parameter für die Anfragen an die Streaming API definiert werden können. Die Klasse OauthPhirehose dient der Authentifizierung der Anwendung gegenüber der Streaming API unter Verwendung von OAuth.

Zunächst wird die Datenhaltungsschicht in der Klasse "TwDB" als Transaction-Script entworfen.[100] Neben der Datenbankverbindung stellt sie gängige Methoden für den Datenbankzugriff - wie z. B. insert und update - bereit. Anschließend wird die Klasse "TwTweetCollector" erstellt, die wiederum die Klasse "OauthPhirehose" erweitert und damit zugleich die Klasse "Phirehose" generalisiert. In dieser Klasse werden lediglich zwei Methoden realisiert. Unter Verwendung der Methode "db_conn" kann der Konstruktor der Klasse "TwDB" aufgerufen und diese dadurch instantiiert werden. Die in der Klasse "Phirehose" vorhandene, abstrakte Methode "enqueueStatus", die automatisch aufgerufen wird, sobald ein Tweet-Objekt von der Streaming API empfangen wird, muss in der Klasse "TwTweetCollector" überschrieben werden. Innerhalb dieser Methode wird jedes im JSON-Format empfangene Tweet-Objekt decodiert, serialisiert und anschließend neben seiner tweet_id als base64-String gespeichert. Zusätzlich werden zwei Skripte erstellt, die einzeln oder gemeinsam als Hintergrundprozess ausgeführt werden können.

1. Das erste Skript "TwGetTweets.php" erstellt eine Instanz der Klasse "TwTweetCollector" und übergibt als Parameter die zur OAuth-Authentifizierung benötigten Zeichenketten (OAUTH_TOKEN und OAUTH_SECRET) sowie eine Konstante, die die zu verwendende Streaming-Methode (in diesem Fall "filter") definiert.[101] Anschließend stellt es eine Datenbankverbindung über die Objektmethode „db_conn“ her und ruft die Setter-Methoden mit den jeweils gewünschten Filtern auf. Mit dem Aufruf der in

[99] Vgl. Bailey, 2011.
[100] Vgl. Rau, 2007 S. 184 f.
[101] Vgl. Green, 2013.

der Klasse "Phirehose" definierten Methode "consume()" wird der Stream gestartet.

2. Das zweite Skript "TwGetAttributes" ruft die mithilfe des ersten Skriptes empfangenen Tweet-Objekte aus der Datenbanktabelle "tweet_json" ab, deserialisiert diese und speichert die Attributwerte in den dafür vorgesehenen Feldern der Tabellen "tweet", "user" und "place".[102]

Die Klassen „TwDB“, „TwTweetCollector“ und die beiden Skripte „TwGetTweets“ und „TwGetAttributes“ befinden sich im Anhang.

[102] Vgl. Green, 2013.

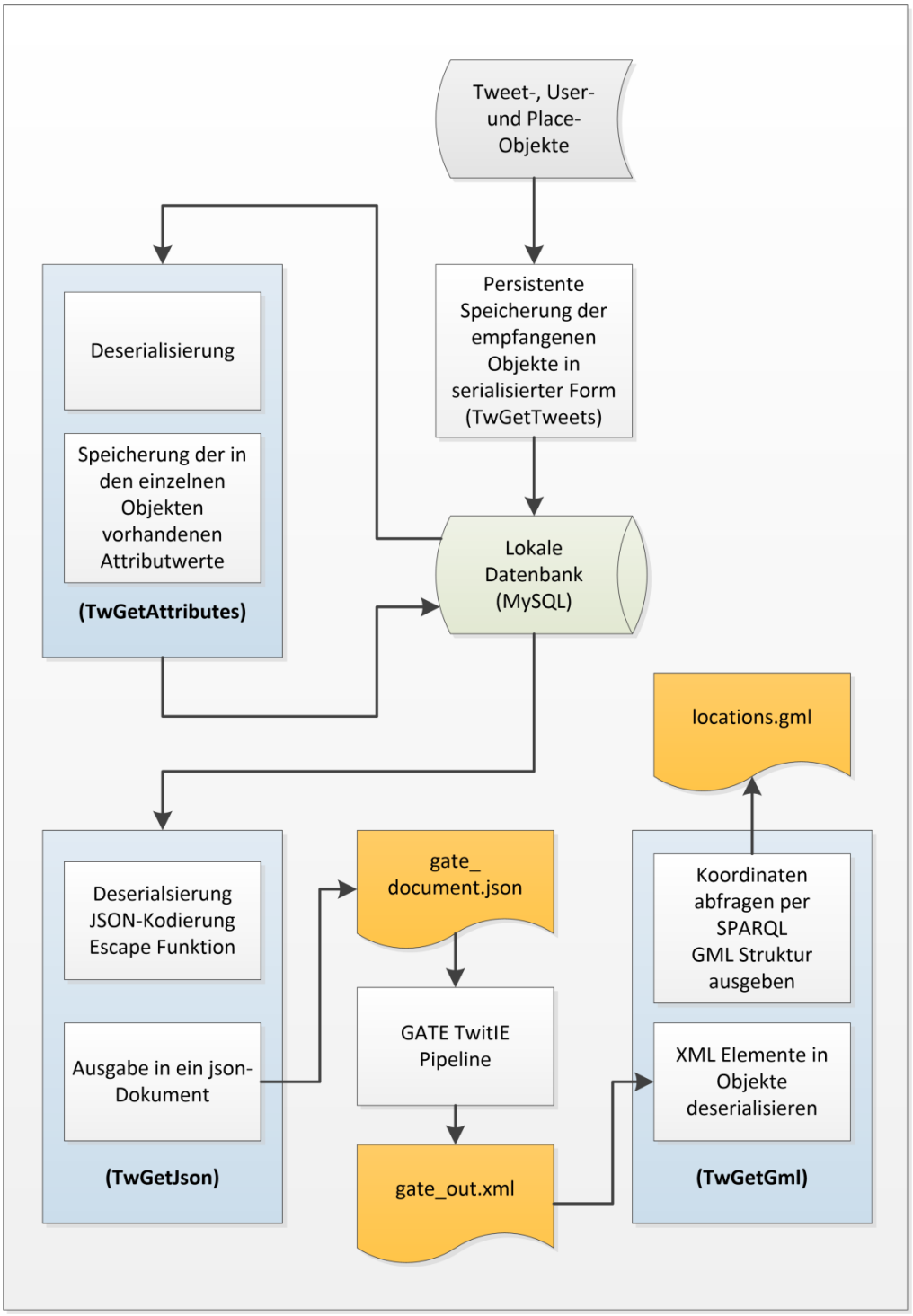


Abbildung 9: Ablaufskizze zu Abruf und Aufbereitung der Twitterdaten

4 Quantitative Analyse in Tweets vorhandener, strukturierter Geodaten

Um die bei der Geolokalisierung von Tweets gegebenen Anforderungen genauer erfassen zu können, soll ermittelt werden, wie häufig Tweets die dazu relevanten Attribute aufweisen. Dazu wird die im vorherigen Abschnitt beschriebene Anwendung auf einem Debian-Server mit PHP 5.6 und MySQL 5.7 implementiert. Innerhalb der in TwGetTweets vorhandenen Methode "setTrack()" wird zunächst ein array aus den fünfzig in der englischen Sprache am häufigsten verwendeten Nomen als Schlüsselwortfilter angegeben.[103] Ausdrücke, die eine geographische Entität beschreiben - wie z. B. das Wort "world", werden anschließend aus dem array entfernt. Zusätzlich wird die Ergebnismenge unter Verwendung der Methode "setLang()" auf englischsprachige Tweets beschränkt.

In einem zweiten Durchlauf wird das Schlüsselwort-array durch ein array mit fünfzig zufällig ausgewählten Begriffen ersetzt, die eine konkrete geographische Entität oder eine geographische Bezugseinheit addressieren (z. B. "country", "Austria", "city", "London" usw.). In jedem der beiden Durchläufe werden 100000 Tweet-Objekte und die mit ihnen assoziierten user- und place-Objekte über die Streaming API abgerufen, deserialisiert und die ausgewählten Attributwerte in der Datenbank gespeichert. Anschließend werden per SQL ("Structured Query Language") die Häufigkeiten der mit Werten befüllten Tabellenfelder über das kartesische Produkt der tweet-, user- und place-Tabelle abgefragt.

```
SELECT Count(*)
  FROM   user
    JOIN tweet
      ON user.user_id = tweet.user_id
      JOIN place
        ON tweet.place_id = place.place_id
  WHERE  tweet.geo IS NOT NULL
    AND tweet.geo != '';
```

[103] Vgl. linguasorb, 2015.

In den folgenden Tabellen werden die absoluten Häufigkeiten der Tweets abgebildet, die das jeweils untersuchte Attribut enthalten. Zusätzlich werden die absoluten Häufigkeiten der Tweets dargestellt, die die jeweils untersuchte Kombination aus zwei Attributwerten aufweisen.

AND	t.geo	t.coor-dinates	t.place _id	p.place_ name	p.place_ fullname	u.loca-tion	u.time _zone
t.geo	394	394	383	383	383	301	265
t.coordi-nates		394	383	383	383	301	265
t.place _id			1915	1915	1915	1552	1305
p.place_ name				1915	1915	1552	1305
p.place_ fullname					1915	1552	1305
u.loca-tion						65812	43888
u.time _zone							58698

Tabelle 3: Absolute Häufigkeiten von Geodaten – Probe 1, 100000 Tweets

In den Ergebnisdaten des ersten Durchlaufs stimmen die Werte des geo-Objektes exakt mit denen des Feldes "coordinates" überein. Das geo-Objekt ist als veraltet gekennzeichnet und sollte nicht mehr verwendet werden.[104] Die als GeoJSON formatierten Koordinatenpaare werden den Tweetobjekten i. d. R. durch eine Clientanwendung zugewiesen. Sie spiegeln den Punkt wieder, an dem der Tweet verfasst wurde. Mit 0,394 % ist allerdings nur ein sehr geringer Anteil der Tweets auf diese Weise lokalisierbar. 1,915 % der Tweets sind über das Attribut "place_id" mit einem place-Objekt verknüpft. Die Zuweisung eines place-Objektes erfolgt i. d. R. durch die

[104] Vgl. Twitter, 2015 9.

BenutzerInnen. Dabei muss ein Place nicht zwangsläufig den Ort bezeichnen, an dem der Tweet verfasst wurde. Es kann auch ein anderer Kontextbezug zu der entsprechenden geographischen Entität gegeben sein. Innerhalb der ersten Ergebnismenge sind die Attribute „name“ und „fullname“ aller abgerufenen „place“-Objekte mit Werten befüllt. Das Attribut „fullname“ gleicht i. d. R. dem „name“-Attribut, enthält jedoch gewöhnlich eine weitere Präzisierung (Der Wert "London" des „name“-Attributes könnte z. B. zusammen mit dem Wert "London, England" im „fullname“-Attribut verwendet werden). Ein „place“-Objekt kann unterschiedliche, ebenfalls als GeoJSON formatierte Features enthalten. Zusätzlich können jedem place-Objekt mithilfe des "Attributes"-Feldes mehr oder weniger frei definierbare Metadaten zugewiesen werden. Ein Place kann also ebensogut eine Adresse repräsentieren, an der der zugehörige Tweet erstellt wurde, wie eine Stadt oder eine Insel, auf die innerhalb des Tweettextes Bezug genommen wird. Die beiden übrigen untersuchten Attribute "location" und "time_zone" sind zwar bei 65,812 % bzw. bei 58,698 % der betrachteten Tweets mit Werten versehen; sie lassen aber ebenfalls nur eine sehr ungenaue Lokalsierung von Tweets zu. Es ist nicht definiert, welchen inhaltlichen Bezug ein Benutzer zu dem innerhalb des location-Feldes adressierten Ort haben muss. Außerdem kann das Feld jederzeit durch den Benutzer verändert und mit einer belibigen Zeichenkette befüllt werden. Es finden sich deshalb in diesem Feld nicht selten Zeichenketten wie "wherever I am" oder "planet earth".

AND	t.geo	t.coor-dinates	t.place_id	p.place_name	p.place_fullname	u.loca-tion	u.time_zone
t.geo	874	347	872	346	346	783	740
t.coor-dinates		347	346	346	346	309	295
t.place_id			2259	922	922	1935	1683
p.place_name				922	922	786	686
p.place_fullname					922	786	686
u.loca-tion						65636	45051
u.time_zone							59428

Tabelle 4: Absolute Häufigkeiten von Geodaten - Probe 2, 100000 Tweets

In der zweiten Stichprobe fällt zunächst auf, dass Tweets mit 0,874 % mehr als doppelt so häufig Werte im geo-Attribut aufweisen als im coordinates-Attribut. Die bei 0,347 % der Probe vorhandenen Werte im coordinates-Feld stimmen mit den dem geo-Objekt zugewiesenen Koordinaten überein. Mit 2,259 % sind etwas mehr Tweets mit einem place-Objekt verknüpft als in der vorherigen Untersuchung; von diesen place-Objekten sind jedoch nur ca. 41 % mit Namen versehen. Das location- und das time_zone-Feld sind ungefähr genauso häufig mit Werten versehen wie in den Tweet-Daten des ersten Durchlaufs. Offenbar werden Tweets, in deren Text eine Ortsbezeichnung enthalten ist, allgemein seltener unter Verwendung der stärker strukturierten Attribute mit benannten geographischen Entitäten verknüpft.

5 Informationsextraktion unter Verwendung der GATE Pipeline Twitie

GATE ("general architecture for text engineering") bezeichnet ein umfangreiches Framework zur Textanalyse. Basierend auf offenen Standards - wie z. B. Java, XML, RTF, HTML und SGML - umfasst es vielzählige Komponenten zur maschinellen Verarbeitung natürlicher Sprache, die je nach Anforderung erweitert bzw. ausgetauscht werden können.[105] Zu diesem Zweck verfügt es über eine graphische Entwicklungsumgebung und eine vollständig dokumentierte Programmierschnittstelle (API).

Unter Verwendung der GATE Klassenbibliothek oder eigenständig entwickelter Module wird i. d. R. ein sequenzieller Ablauf modelliert, der anhand eines Eingangsdokumentes die gewünschte Bearbeitung vornimmt und als Ergebnis ein Ausgabedokument erzeugt. Eine solche Sequenz wird auch als GATE-Pipeline bezeichnet. Die GATE-Pipeline ANNIE ("A Nearly New Information Extraction System") ist im Standardpaket als Plugin enthalten.[106] Die innerhalb von GATE eingesetzten Ressourcen werden in zwei Bereiche unterteilt:[107]

- Processing resources (PR): Die Programme bzw. Algorithmen, die innerhalb der GATE-Pipeline durchlaufen werden.
- Language resources (LR): Die Dokumente, Korpora, Ontologien usw., die innerhalb der GATE-Pipeline Verwendung finden - im Wesentlichen die zu verarbeitenden Daten.

Die einzelnen Ressourcen kommunizieren untereinander mithilfe der GATE Annotation API.[108]

[105] Vgl. Cunningham et. al., 2002 S. 168 f.
[106] Vgl. Cunningham et. al., 2014.
[107] Vgl. Cunningham et. al., 2014.
[108] Vgl. Bontcheva et. al., 2013 S. 84.

Anwendung einer GATE Pipeline:[109]

1. Auswahl eines Korpus: Als linguistisches Textkorpus oder Korpus wird eine Sammlung digital erfasster, für eine computerlinguistische Aufgabe aufbereiteter schriftlicher Äußerungen bezeichnet.[110]

2. Auswahl einer strukturierten Beschreibung der zu untersuchenden Wissensdomäne (z. B. Ontologie, Taxonomie usw.)

3. Mithilfe von „GATE Teamware“ wird eine Teilmenge des Korpus entsprechend der zuvor gewählten strukturierten Beschreibung manuell annotiert und so in die Struktur der Ontologie überführt. Die als Ergebnis dieses Arbeitsschrittes entstandene Datenstruktur wird als Gold Standard bezeichnet.

4. Unter Verwendung von GATE Developer wird eine Annotations-Pipeline erstellt, mit deren Hilfe die übrigen Teile des Korpus automatisiert annotiert werden können. Anschließend wird die Datenausgabe der automatisierten Annotation mit dem Gold Standard verglichen, um die Genauigkeit der eingesetzten Methodik zu messen.

Das Eingangsdokument bleibt innerhalb einer GATE-Pipeline in seiner ursprünglichen Form erhalten. Die Annotationen werden separat gespeichert. Auf diese Weise kann das GATE Dokument jederzeit in seinen Originalzustand zurückversetzt werden. Annotationen müssen nach dem Laden eines neuen Dokumentes zunächst aus dem Speicher gelöscht werden. Hierzu kann zu Beginn jeder GATE-Pipeline die Ressource "Document Reset PR" durchlaufen werden.[111]

[109] Vgl. Cunningham et. al., 2014.
[110] Vgl. Carstensen, 2010 S. 482.
[111] Vgl. Cunningham et. al., 2014.

5.1 GATE Twitter plugin

Neben zahlreichen Werkzeugen, die eine effiziente Verarbeitung anderer "Social Media"-Daten ermöglichen, umfasst das GATE Framework ein Twitter-plugin, das auf dem Stanford_CoreNLP plugin basiert.[112] Mithilfe dieses Plugins können die unter Verwendung der Twitter Search API bzw. der Twitter Streaming API im JSON Format bezogenen Daten direkt in die Anwendung importiert werden. Zusätzlich umfasst das Plugin einen Tokeniser, einen für die Verarbeitung von Tweets optimierten POS-Tagger, ein Werkzeug zur Zerlegung aus mehreren Wörtern zusammengesetzter Hashtags und Methoden zur Named Entity Recognition auf der Basis von TwitIE.[113]

[112] Vgl. Cunningham et. al., 2014.
[113] Bontcheva et. al., 2013 S. 87.

5.2 Informationsextraktion mit TwitIE

TwitIE bezeichnet eine speziell an die Anforderungen von Social-Media-Inhalten - insbesondere an die von Microblog-Texten - angepasste Variante der ANNIE-pipeline. Innerhalb der TwitIE-pipeline werden die als Teil der ANNIE-pipeline zur Verfügung stehenden Komponenten zur Satzerkennung (sentence splitter) und das zur Namenserkennung dienende Gazetteer wiederverwendet. Die übrigen Ressourcen unterscheiden sich von denen der ANNIE-pipeline.[114]

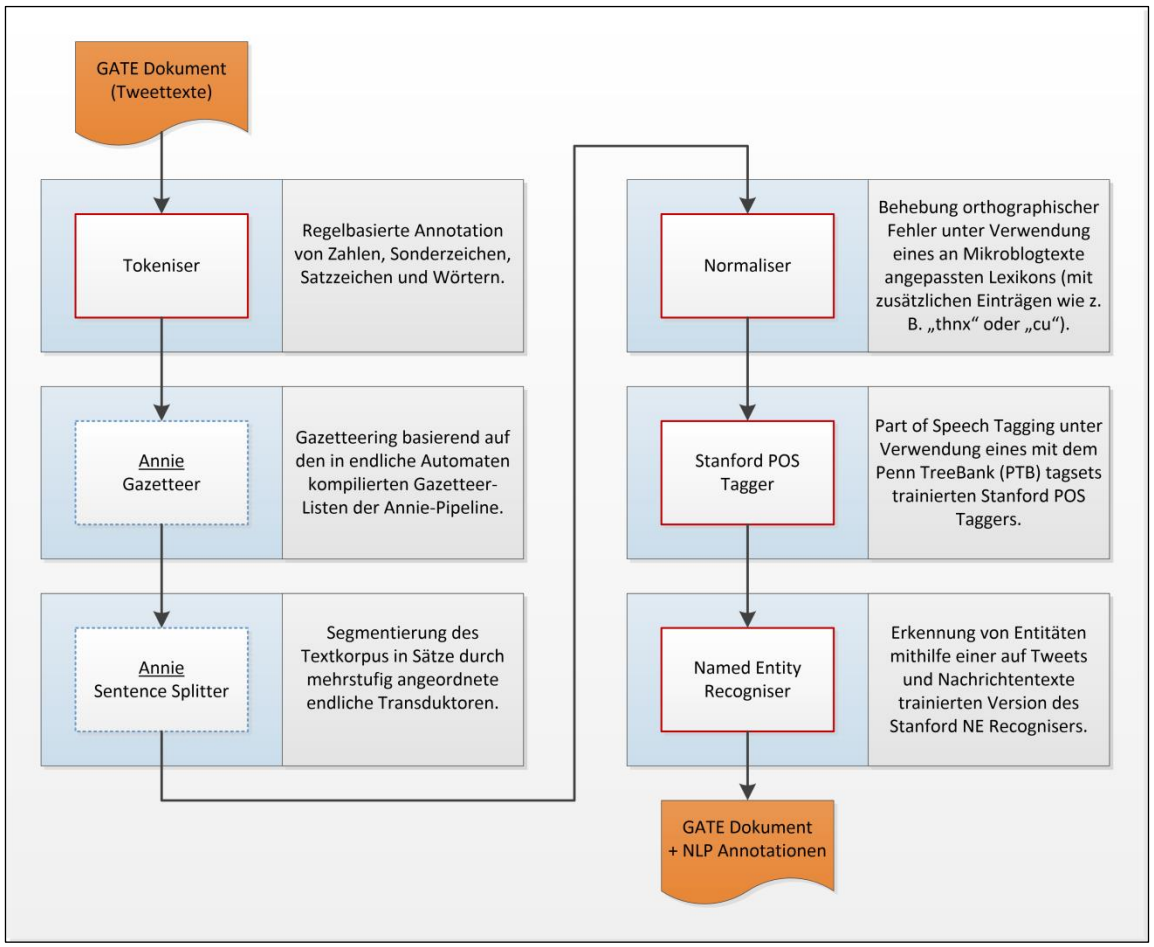


Abbildung 10: Ablauf einer TwitIE-Pipeline[115]

[114] Vgl. Bontcheva et. al. 2013 S. 85.
[115] Vgl. Bontcheva et. al. 2013 S. 85.

Um die TwitIE-Pipeline verwenden zu können, muss zunächst ein Gate-Dokument geladen werden, das die zu analysierenden Textdaten enthält. Um die weitere Verarbeitung durch das GATE-Twitter-Plugin bzw. das "Twitter JSON format"-Plugin zu ermöglichen, können die Tweets im JSON-Format übergeben werden. Die JSON-Repräsentation der Tweets kann dabei entweder in einzelnen Dokumenten oder aneinandergereiht in einem einzelnen Dokument vorliegen.[116]

Um ein Gate Dokument aus den empfangenen Twitter-Daten zu generieren, wird das zuvor verwendete Skript "TwGetAttributes.php" durch das Skript "TwGetJson.php" ersetzt. Anders als das erste Skript speichert dieses Skript nicht einzelne Attributwerte in der Datenbank, sondern überführt die deserialisierten Tweet-Objekte zurück ins JSON-Format, speichert diese in einer Textdatei ab und löscht anschließend die serialisierten, base64-codierten Tweet-Objekte aus der Datenbanktabelle "tweet_json".

Damit der mimeType der Daten ("text/x-json-twitter") während des Importvorgangs automatisch erkannt wird, wird die Textdatei mit der Dateiendung ".json" versehen. Nachdem das GATE-Dokument als neue LR in die Anwendung "GATE-Developer" importiert wurde, sind die aus den einzelnen Attributen generierten Annotationen verfügbar und können im entsprechenden Editor aktiviert oder deaktiviert werden.

Bevor nun die TwitIE-pipeline in die Anwendung geladen wird, wird aus dem GATE-Dokument ein sogenanntes GATE-Corpus erzeugt, welches der TwitIE-Applikation als Eingangsparameter übergeben wird. Neben den LR müssen noch die als Teil der Pipeline vorgesehenen Plugins importiert werden. Einzelne Plugins können in das Verzeichnis "plugins" unterhalb des Wurzelverzeichnisses kopiert und anschließend mithilfe des "Creole Plugin Managers" in der Anwendung angezeigt werden.

[116] Vgl. Cunningham et. al., 2014.

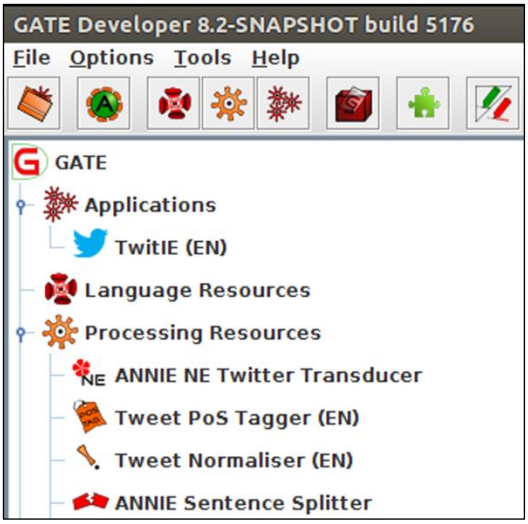


Abbildung 11: Ladevorgang einer Processing Resource in der GATE Developer Anwendung

Indem man eine vollständige Pipeline als Applikation hinzufügt, werden die als Teil der Pipeline benötigten PR - sofern sie im Plugin-Verzeichnis vorhanden und im Creole Plugin Manager aktiviert sind - automatisch importiert. Durch einen Doppelklick auf den Namen der Applikation können dieser weitere PR hinzugefügt oder nicht benötigte entfernt werden.

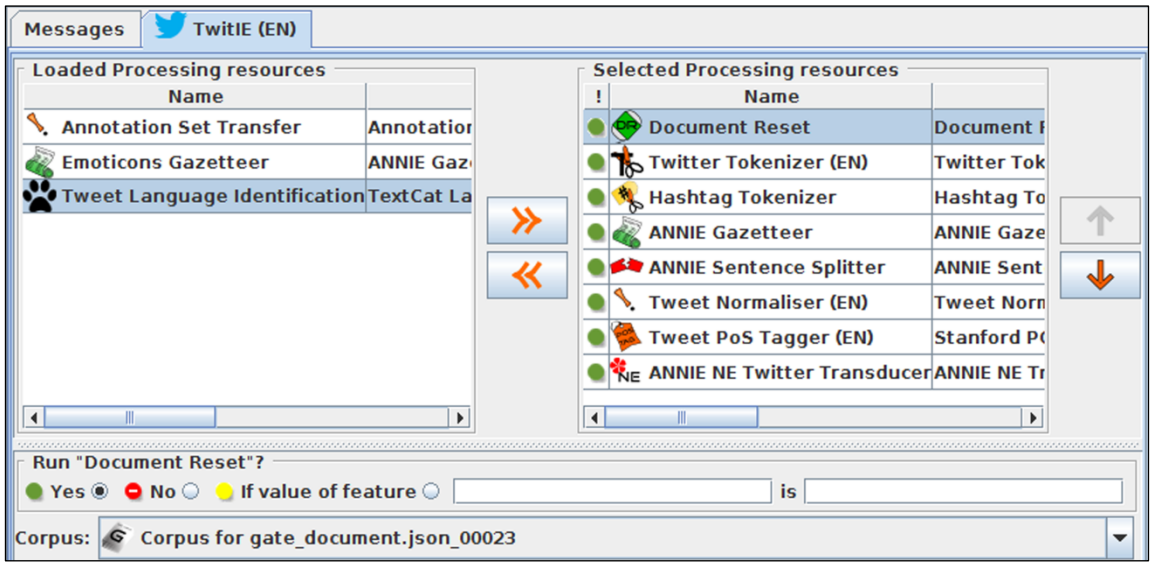


Abbildung 12: Zusammenstellung der Processing Resources einer GATE Pipeline

Da das im Rahmen dieser Arbeit verwendete GATE-Dokument lediglich englischsprachige Tweets enthält, kann z. B. auf die PR "Tweet Language Identification" verzichtet werden. Nachdem die Auswahl der PR abgeschlossen ist, wird das zuvor generierte GATE-Corpus ausgewählt und die Applikation gestartet. Wenn die TwitIE-pipeline ohne Fehlermeldungen durchlaufen wurde, kann man durch einen Doppelklick auf den Bezeichner des GATE Dokumentes unter "Annotation Sets" die im Eingangsdokument enthaltenen Tweet-Texte zusammen mit den durch das Framework erkannten Annotationen anzeigen lassen.

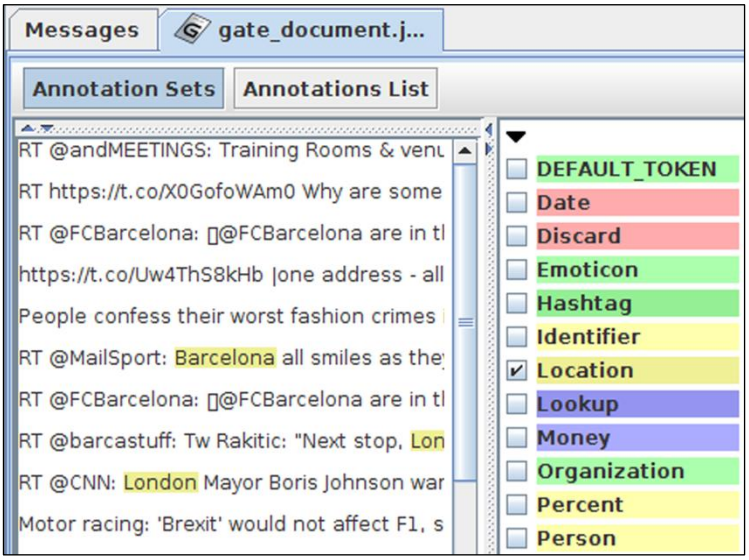


Abbildung 13: Anzeige und Auswahl von Annotation Sets

Oberhalb der bereits innerhalb des GATE Dokumentes vorhandenen Annotationen können einzelne oder mehrere Annotationsklassen ausgewählt und die damit ausgezeichneten Tokens innerhalb der Tweet-Texte farblich markiert werden. Die Ergebnisdaten können im XML-Format exportiert werden. Durch den Export als "GATE XML" werden sämtliche Annotationen (auch die bereits vorher vorhandenen) ohne die analysierten Texte in eine XML-Datei exportiert. Jedem XML-Element, das eine Annotation repräsentiert, sind dabei zwei Attribute zugewiesen, deren Werte den Anfang und das Ende der annotierten Zeichenkette wiederspiegeln.

<tweets>
@janhUNIGIS Die #AGIT_2016 beginnt in <Location gateId "123"
ruleFinal "LocFinal"
locType "city" kind "locName">Salzburg</Location> 06.07.2016 @unigis
http://www.unigis.ac.at/ pic.twitter.com/xyz123
</tweets>

Alternativ kann eine "inline XML"-Datei erzeugt werden, die die durch Leerzeilen voneinander getrennten Tweet-Texte und die Annotationen in Form darin eingebetteter tags enthält. Unter Verwendung dieser Exportfunktion können zudem einzelne Annotationsklassen entfernt oder hinzugefügt werden.

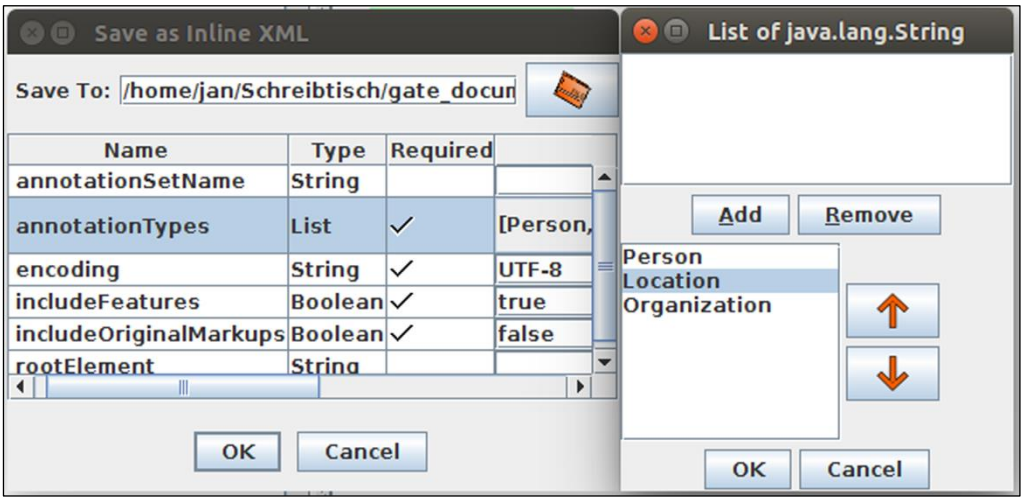


Abbildung 14: Auswahl der Exportparameter für ein mit TwitIE annotiertes Dokument

So ist es z. B. möglich, ausschließlich die tags von erkannten Personen, Organisationen oder benannten Orten in das Ergebnisdokument einzubetten. Außerdem können zusätzliche Attribute der einzelnen Tags deaktiviert werden, so dass nur noch die Bezeichner vorhanden sind. Auf diese Weise ist es besonders einfach die Ergebnisdaten unter Verwendung eines XML-Parsers weiterzuverarbeiten.

Abbildung 15: Beispieldarstellung mit TwitIE annotierter Tweets

6 Georeferenzierung mit SPARQL

SPARQL-Anfragen an eine Ontologie, die entsprechende Geodaten verknüpft, ermöglichen eine Georeferenzierung bei gleichzeitiger Disambiguierung der angefragten Ortsnamen. Frei verfügbare Webservices, die auf RDF basieren und somit SPARQL-Anfragen verarbeiten können, stehen z. B. mit "GeoNames" oder "DBPedia" zur Verfügung. Im Folgenden wird die Georeferenzierung unter Verwendung von SPARQL-Anfragen an die DBPedia-REST-API verdeutlicht. Aufgrund des begrenzten Umfangs dieser Arbeit wird auf eine weiterführende Disambiguierung an dieser Stelle verzichtet; bei Vorliegen entsprechender Kontextdaten wäre es allerdings möglich, die SPARQL-Anfrage um diese Anforderung zu erweitern.

Es wird zunächst ein PHP-Skript erstellt, das die innerhalb der Datei "gate_output.xml" gespeicherten, mit Location-tags versehenen Ortsbezeichner einliest und einer Liste hinzufügt. Dabei werden nur diejenigen Toponyme einbezogen, die noch nicht in der Liste vorhanden sind. Anschließend soll das Skript diese Liste durchlaufen, um die zu jedem Ortsbezeichner gespeicherten geographischen Koordinaten mithilfe einer SPARQL-Anfrage abzurufen und im GML-Format auszugeben.

```
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
  SELECT ?long ?lat WHERE {
     ?l foaf:name "London"@en .
     ?l a dbo:Place .
     ?l geo:long ?long .
     ?l geo:lat ?lat .
  }
  LIMIT 1
```

Zu Beginn der Anfrage werden drei Namensräume eingebunden. Das Präfix "geo" wird einem RDF-Vokabular zugewiesen, das einen Namensraum für die Repräsentation

geographischer Koordinaten im Referenzsystem "WGS84" bereitstellt. Zusätzlich wird ein Präfix für die DBPedia-Ontologie und ein Präfix für die Friend-of-a-Friend-Ontologie (FOAF) definiert. Innerhalb des SELECT-Blocks werden ausschließlich die Koordinaten als Rückgabewerte angefordert. In der WHERE-Klausel werden vier Tripel als Abfragekriterien angegeben. Durch diese Abfrage werden die geographischen Koordinaten ("longitude, latitude") einer Resource im Referenzsystem „EPSG:4326“ angefragt, die im FOAF-Namensraum durch den Namen "London" repräsentiert wird und die innerhalb der DBPedia-Ontologie der Klasse "Place" angehört.

Um die Anfrage auszuführen wird innerhalb eines PHP-Skriptes (TwGetGml.php) mithilfe der Curl URL Request Library (curl) eine HTTP-GET-Anfrage an den DBPedia SPARQL Endpunkt (http://dbpedia.org/sparql) gesendet, die zwei Parameter übergibt. Der erste Parameter ("query") enthält die URL-codierte SPARQL-Anfrage. Der zweite Parameter ("format") gibt JSON als das gewünschte Rückgabeformat an. Nachdem der zurückgegebene JSON-String deserialisiert wurde, können die Koordinaten zwei Variablen zugewiesen und zusammen mit der Ortsbezeichnung als GML-Punkt-Feature ausgegeben werden.

```
<gml:featureMember>
  <tw:location fid="location.1">
    <tw:geometryProperty>
      <gml:Point srsName="EPSG:4326">
        <gml:coordinates>-0.1275 51.5072</gml:coordinates>
      </gml:Point>
    </tw:geometryProperty>
  </tw:location>
</gml:featureMember>
```

Die vollständige GML-Datei (locations.gml) sowie die zugehörige Schema-Datei (locations.xsd) finden sich im Anhang.

7 Georeferenzierung von Tweets durch Toponymextraktion

Um zu untersuchen, wie häufig mehrere Toponyme innerhalb eines Tweet-Textes vorkommen, werden 1000 Tweet-Objekte unter Verwendung des Schlüsselwortes „London“ über die Streaming API abgerufen. Anschließend werden mithilfe der TwitIE-pipeline Toponyme extrahiert bzw. annotiert und eine GML-Datei mit den von der DBPedia-Ontologie abgefragten Koordinaten generiert. Diese GML-Datei kann daraufhin in ein Geoinformationssystem importiert werden.

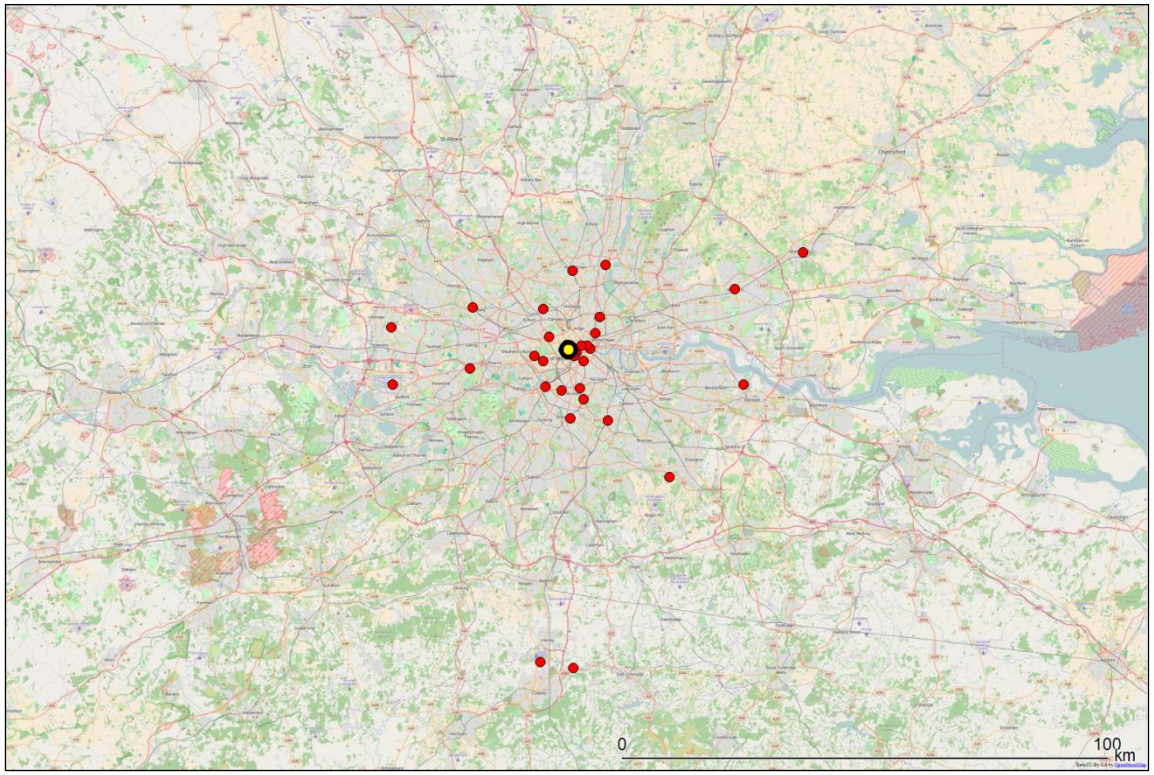


Abbildung 16: Kartendarstellung – durch Toponymextraktion geolokalisierte Tweets - mehrere Toponyme[117]

Die innerhalb der GML-Datei gespeicherte FeatureCollection enthält 106 Punktgeometrien. Es waren demnach 105 zusätzliche Toponyme (abweichend vom Schlüsselwort „London“) in den Tweet-Texten vorhanden, für die bei Ausführung der SPARQL-Anfrage geographische Koordinaten zurückgegeben wurden. 33 der durch die PointFeatures repräsentierten Standorte befinden sich innerhalb eines 50 KM – Radius

[117] Quelle Hintergrund-Layer: Open Street Map

um das Londoner Stadtzentrum. Die übrigen Standorte sind teilweise sehr viel weiter von London entfernt.

Um geolokalisierte Tweets gemeinsam mit den aus ihren Tweet-Texten extrahierten Toponymen in einer Karte darstellen zu können, werden zunächst noch einmal 100000 Tweets unter Verwendung der vorher erstellten Skripte gespeichert. Anschließend wird jedoch nicht wie im vorangegangenen Versuch ein Gate-Dokument aus JSON-Objekten generiert; stattdessen wird mithilfe von SQL ein XML-Dokument erstellt, das die Elemente „tweet“ und die dazugehörigen Kindelemente „tweet_id“, „coordinates“ und „tweet_text“ enthält. Das Element „coordinates“ wird dabei mit der innerhalb der „geo“- bzw. „coord“-Felder gespeicherten geographischen Länge und der geographischen Breite getrennt von einem Leerzeichen angegeben. Die Ergebnismenge wird zudem auf die Tweet-Objekte beschränkt, deren „geo“- oder „coordinates“-Attribut einen Wert enthält.

In dieser Probe werden auf diese Weise 385 Tweet-Elemente in das XML-Dokument integriert. Damit die einzelnen XML-Elemente zu Beginn der TwitIE-pipeline nicht als vorhandene Annotationen erkannt werden, wird die Datei mit der Endung „.txt“ gespeichert und anschließend unter Verwendung des GATE Developers ein passendes GATE Corpus generiert. Nach Ausführung der TwitIE-pipeline und dem Export als Inline-XML weist das Ergebnisdokument folgende Struktur auf:

```
<?xml version="1.0" encoding="utf-8" ?>
  <tweets>
    <tweet>
      <tweet_id>
      </tweet_id>
      <coordinates>
      </coordinates>
      <text>
        <Location></Location>
      </text>
    </tweet>
  </tweets>
```

Um sowohl die tweet_ids als auch die Namen der einzelnen Locations in eine GML-Struktur einzubeziehen, wird das Skript „TwGetGml.php“ entsprechend angepasst und die innerhalb des Ausgabedokumentes referenzierte Schema-Definition (locations.xsd) bzw. die darin für „locations“ enthaltene Typdefinition um ein Attribut „name“ erweitert. Zusätzlich werden drei Attribute erstellt, die innerhalb eines durch vorhandene Tweet-Koordinaten generierten PointFeatures auf die PointFeatures verweisen, die aus den extrahierten Toponymen des jeweiligen Tweets referenzieren. Die Struktur des Ausgabedokumentes entspricht demnach folgendem Muster:

```
<gml:featureMember>
  <tw:location fid "location.3">
    <tw:foreign1>Lexington</tw:foreign1>
    <tw:foreign2>KY</tw:foreign2>
    <name>700617123456789012</name>
    <tw:geometryProperty>
      <gml:Point srsName "EPSG:4326">
        <gml:coordinates>-84.49514 38.03171</gml:coordinates>
      </gml:Point>
    </tw:geometryProperty>
  </tw:location>
</gml:featureMember>
```

Die auf diese Weise erstellte GML-Datei enthält 455 Punktgeometrien. Abzüglich der 385 georeferenzierten Tweets sind demnach 70 aus den Texten extrahierte Features in der Datei enthalten. Dabei ist zu beachten, dass mehrfach vorhandene Toponyme zu einem Element zusammengefasst und Toponyme, deren Koordinaten nicht über die DBPedia-Ontologie abgerufen werden konnten, aus der Liste entfernt wurden. Ursprünglich wurden 136 Toponyme während der Ausführung der TwitIE-pipeline erkannt. In 35 Tweet-Texten traten 2 Toponyme auf. In 2 Tweet-Texten wurden sogar 3 Toponyme genannt. Es wurde demnach in 97 der 385 untersuchten Tweets mindestens ein Toponym erkannt.

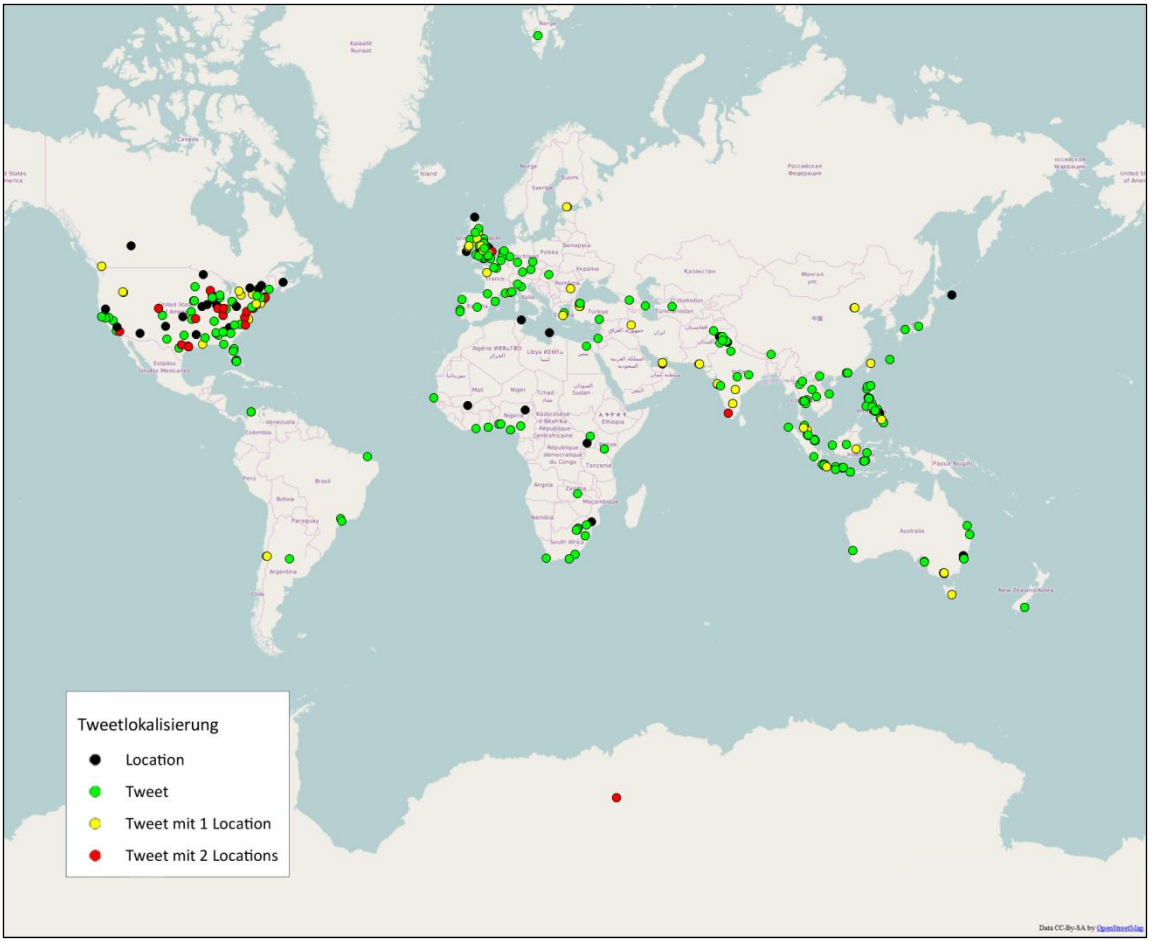


Abbildung 17: Kartendarstellung - Geolokalisierung durch Toponymextraktion und coordinates-Feld

8 Toponymextraktion aus Werten des location-Feldes

Zu einer Probe von 100000 Tweet-Objekten werden aus der Datenbank die User-Objekte abgerufen, die die Verfasser der Tweets repräsentieren. Zwischen der User- und der Tweet-Tabelle besteht eine 1:n Relation. Die 100000 Tweets wurden von 77836 Benutzern verfasst. Innerhalb von 53128 Benutzerdatensätzen enthält das locations-Feld einen Wert. Diese Datensätze werden per SQL-Befehl abgefragt und zusammen mit der jeweiligen user_id in einer Textdatei zusammengefasst. Jede Zeile dieser Textdatei enthält dabei eine user_id und den zugehörigen Wert des location-Feldes.

Anschließend wird aus der Textdatei ein GATE-Corpus generiert und damit die innerhalb des GATE Frameworks enthaltene ANNIE-Pipeline ausgeführt. Nachdem das GATE-Ergebnisdokument mit dem Parameter „include Features“ im XML-Format exportiert wurde, wird es mithilfe des abermals angepassten PHP-Skriptes eingelesen und die enthaltenen location-Elemente sowie die ihnen zugewiesenen locType-Attribute ausgewertet. Mithilfe des locType-Attributes kann ein location-Element durch die ANNIE-Pipeline einer Typklasse zugeordnet werden.

```
<tweets>
  <tweet>
    <id>700617630123456789</id>
    <coordinates>-1.14430 52.68725</coordinates>
    <text>@janhUNIGIS Die #AGIT_2016 beginnt in
      <Location gateId "123" ruleFinal "LocFinal" locType "city"
        kind "locName">Salzburg</Location>am 06.07.2016 @unigis
          http://www.unigis.ac.at/ pic.twitter.com/xyz123
    </text>
  </tweet>
</tweets>
```

		zugewiesene locType-Werte							
erkannte locations	erkannte locations inkl. locType	**city**	**coun-try**	**cro-vince**	**ce-gion**	**pre**	**post**	**airport**	**unknown**
42091	40319	19093	11273	5886	3387	521	98	34	27

Tabelle 5: Autrittshäufigkeiten bei der Toponymextraktion erkannter LocationTypes

9 Fazit und Ausblick

Innerhalb der erhobenen Datenbasis ist im arithmetischen Mittel über die verschiedenen Stichproben bei 33,45 % der untersuchten Tweets das Location-Feld des zugehörigen Benutzers nicht ausgefüllt. Bei 95,79 % der übrigen Datensätze wurde der innerhalb des Location-Feldes gespeicherte Wert mithilfe der GATE-Pipeline als Toponym erkannt und entsprechend annotiert. Allerdings konnten mithilfe des verwendeten Ansatzes zu 39,73 % dieser Toponyme keine geographischen Koordinaten aus der DBPedia-Ontologie abgefragt werden. Demnach konnten 38,42 % der Gesamtmenge aller untersuchten Tweets unter Verwendung des Location-Feldes mit Koordinaten ausgezeichnet werden. In Kombination mit den Daten aus dem „geo“- bzw. „coordinates“-Feld können 38,38 % der Tweets auf einer staatlichen oder einer einem Staat untergeordneten Bezugseinheit verortet werden.

Reichweitendaten linearer Medien werden häufig innerhalb relativ großer geographischer Bezugseinheiten erhoben. Z. B. werden die Einschaltquoten für das Fernsehen landesweit unter Einbeziehung einer vergleichsweise kleinen Stichprobe erfasst. Die dabei untersuchte Gruppe umfasst üblicherweise 2000-6000 Personen. Wenn man die Gesamtzahl der Nutzer von Twitter und die Einschaltquote von Rundfunk oder Fernsehen in Relation zu der jeweils auswertbaren Stichprobe betrachtet, ist davon auszugehen, dass bei Twitter eine sehr viel genauere Erhebung möglich ist.

Der Einsatz der im Rahmen dieser Arbeit vorgestellten Methoden zur Informationsextraktion kann die Anzahl der Tweets, die zumindest innerhalb größerer Bezugseinheiten verortet werden können, maßgeblich erhöhen. Um die zum Abruf der Tweet-Objekte über die Streaming API verwendeten Parameter - insbesondere die Liste der angefragten Schlüsselwörter - jedoch noch besser an bestimmte Kontexte anzupassen, könnte ein Textkorpus aus Nachrichtenartikeln mithilfe des GATE-Frameworks verarbeitet werden. Dazu müsste eine Pipeline modelliert werden, die die erkannten Entitäten und deren Relationen auf der Grundlage einer vorher entworfenen Ontologie extrahiert, und idealerweise in einer Graph-Datenbank oder einem Triple-Store speichert. Auf dieser Datenbasis könnten anschließend unter Verwendung von SPARQL oder GeoSPARQL strukturierte, kontextbasierte Suchanfragen ausgeführt werden. Es wäre zudem denkbar, eine als Teil der dabei eingesetzten Pipeline verwendbare Processing Ressource zu entwickeln, die die aus der Ontologie abgefragte

Triple-Struktur bestmöglich auf die 500 für Streaming-Anfragen nutzbaren Schlüsselwörter abbildet. Da der Streaming-Endpunkt innerhalb der einzelnen Elemente des TrackWords-Arrays Leerzeichen akzeptiert, müssten die einzelnen Tripel-Sequenzen dazu lediglich in kurze Phrasen aus Subjekt, Prädikat und Objekt, ohne die innerhalb der Ontologie verwendete Syntax, umstrukturiert werden. Da alle innerhalb des GATE-Frameworks vorhandenen Klassen ausführlich dokumentiert sind und der Quellcode frei verfügbar ist, könnte der gesamte Programmablauf zudem in Java umgesetzt und ggf. vollständig automatisiert werden.

Viele der mit der Informationsextraktion aus gering strukturierten Textdaten und der Synthese semantisch angereicherter Austauschformate in Verbindung stehenden Technologien bergen innerhalb zahlreicher Anwendungsgebiete ein erhebliches Innovationspotential. Zukünftige Entwicklungen im Kontext des Semantic Web und damit die höhere Verfügbarkeit in strukturierter Form vorliegender, öffentlich zugänglicher Informationen werden die möglichen Einsatzfelder vieler der in dieser Arbeit verfolgten Ansätze womöglich ebenfalls erweitern.

Literaturverzeichnis

Aggarwal, C. 2011. Social Network Data Analytics. 1. Auflage New York: Springer.

Allemang, D., Hendler, J., 2011. Semantic Web for the working ontologist – Effective modeling in RDFS and OWL. 2. Auflage Waltham: Morgan Kaufmann.

Basset, L., 2015. Introduction to JavaScript Object Notation. 1. Auflage Sebastopol: O'Reilly.

Biemann, C., 2012. Structure Discovery in Natural Language – Theory and Applications of Natural Language Processing Monographs. 1. Auflage Heidelberg et. al.: Springer.

Bontcheva, K., et. al., 2013. TwitIE: An Open-Source Information Extraction Pipeline for Microblog Text. In: Proceedings of the Inernational Conference on Recent Advances in Natural Language Processing. 1. Auflage Shoumen: INCOMA, S. 83-90.

Carstensen, K., et. al., 2010. Computerlinguistik – Was ist das? In: Carstensen, K., et. al. (Hrsg.): Computerlinguistik und Sprachtechnologie. 3. Auflage Heidelberg: Spektrum.

Carstensen, K., 2010. Ressourcen. In: Carstensen, K., et. al. (Hrsg.): Computerlinguistik und Sprachtechnologie. 3. Auflage Heidelberg: Spektrum, S. 481-491.

Cunningham, H., et. al., 2002. GATE: an Architecture for Development of Robust HLT Applications. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics. 1. Auflage Philadelphia: ACL, S. 168-175.

De la Higuera, C., 2010. Grammatical Inference – Learning Automata and Grammars. 1. Auflage New York: Cambridge University Press.

Flanagan, D., 2012. JavaScript – Das umfassende Referenzwerk. 6. Auflage Köln: O'Reilly.

Gruber, T., 1993. A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 5(2), S. 199-220.

Gruninger, M., Uschold, M., 1996. Ontologies: Principles, Methods and Applications. Knowledge Engineering Review, 11(2), S. 1-63.

Geisler, M., 2009. Semantic web. 1. Auflage Darmstadt: entwickler.press.

Hagenbruch, A., 2010. Flache Satzverarbeitung. In: Carstensen, K., et. al. (Hrsg.): Computerlinguistik und Sprachtechnologie. 3. Auflage Heidelberg: Spektrum.

Han, J., Kamber, M. 2006. Data Mining: Concepts and Techniques. 2. Auflage San Francisco: Morgan Kaufmann.

Hasebrink, U., Schröder, H. 2006. Medien von A bis Z. 1. Auflage. Wiesbaden: Verlag für Sozialwissenschaften.

Hitzler, P. et. al., 2008. Semantic Web. 1. Auflage Berlin / Heidelberg: Springer.

Jackson, P., Moulinier, I., 2002. Natural Language Processing for Online Applications – Text Retrieval, Extraction and Categorization. 1. Auflage Amsterdam / Philadelphia: John Benjamins Publishing Co.

Kapetanios, E., et. al., 2014. Natural Language Processing – Semantic Aspects. 1. Auflage Boca Raton: CRC Press.

Kreuzer, M., Kühling, S., 2006. Logik für Informatiker. 1. Auflage München: Pearson.

Kumar, E., 2011. Natural Language Processing. 1. Auflage New Delhi: I.K. International Publishing House.

Liao, Y., et. al., 2012. Mining Micro-blogs: Opportunities and Challenges. In: Abraham, A. (Hrsg.): Computational Social Networks. 1. Auflage London: Springer, S. 129-159.

Liu, J., et. al., 2012. Correlation Mining for Web News Information Retrieval. In: Abraham, A. (Hrsg.): Computational Social Networks. 1. Auflage London: Springer, S. 103-128.

Liu, X., et. al., 2012. Joint Inference of Named Entity Recognition and Normalization for Tweets. In: Association for Computational Linguistics (Hrsg.): ACL '12 Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1. 1. Auflage 2012 Stroudsburg: ACL, S. 526-535.

Lubkowitz, M., 2005. Webseiten programmieren und gestalten. 2. Auflage Bonn: Galileo Press.

Neches, R. et. al., 1991. Enabling Technology for Knowledge Sharing. AI Magazine, 12(3), S. 36-56.

Neumann, G., 2010. Text-basiertes Informationsmanagement. In: Carstensen, K., et. al. (Hrsg.): Computerlinguistik und Sprachtechnologie. 3. Auflage Heidelberg: Spektrum, S. 576-615.

Paroubek, P., 2008. Evaluating Part-Of-Speech-Tagging and Parsing – On the Evaluating of Automatic Parsing of Natural Language. In: Dybkjær, L., et. al. (Hrsg.): Evaluation of Text and Speech Systems. 1. Auflage Berlin: Springer, S. 99-124.

Pospech, T., 2008. GML – Geography Markup Language. 1. Auflage Norderstedt: GRIN Verlag.

Rau, K., 2007. Objektorientierte Systementwicklung – Vom Geschäftsprozess zum Java-Programm. 1. Auflage Wiesbaden: Vieweg.

Ritter, A., et. al., 2011. Named Entity Recognition in Tweets: An Experimental Study. In: Association for Computational Linguistics (Hrsg.): EMNLP '11 Proceedings of the Conference on Empirical Methods in Natural Language Processing. 1. Auflage Stroudsburg: EMNLP, S. 1524-1534.

Roth-Berghofer, T., 2012. Das Resource Description Framework. In: Dengel, A. (Hrsg.): Semantische Technologien. 1. Auflage Heidelberg: Spektrum, S. 109-129.

Sintek, M., 2012. Anfragesprachen und Reasoning. In: Dengel, A. (Hrsg.): Semantische Technologien. 1. Auflage Heidelberg: Spektrum, S. 161-177.

Segaran, T. et. al., 2009. Programming the Semantic Web. 1. Auflage Sebastopol: O'Reilly.

Shi, C., et. al., 2014. A Conceptual Framework of Online Natural Language Processing Pipeline Application. Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT. S. 53-59.

Smith, D., Crane, G., 2001. Disambiguating Geographic Names in a Historical Digital Library. In: Constantopoulos, P., Sølvberg, I. (Hrsg.): Research and Advanced Technology for Digital Libraries. 1. Auflage Berlin / Heidelberg / New York: Springer, S. 127-136.

Sproat, R., 2008. Linguistic Processing for Speech Synthesis. In: Benesty, J., et. al. (Hrsg.): Handbook of Speech Processing. 1. Auflage Berlin / Heidelberg: Springer, S. 457-470.

Stuckenschmidt, H., 2011. Ontologien – Konzepte, Technologien und Anwendungen. 2. Auflage Heidelberg: Springer.

Van den Brink, L., et. al., 2013. From Geo-data to Linked Data: Automated Transformation from GML to RDF. Geonovum, S. 249-261.

Zafarani, R., et. al., 2014. Social Media Mining. 1. Auflage New York: Cambridge University Press.

Ziegler, C., 2012. Mining for Strategic Competitive Intelligence – Foundations and Applications. 1. Auflage Berlin / Heidelberg: Springer.

Elektronische Ressourcen

AGOF digital facts, 2015. Ranking Gesamtangebote digital Verfügbar unter: <www.agof.de/service-downloads/downloadcenter/download-digital-facts/> Abgerufen 21. Februar 2016.

Bailey, F., 2011. Introduction to the Phirehose class. Verfügbar unter: <github.com/fennb/phirehose/wiki/Introduction/> Abgerufen 19. März 2015.

Beisecker, M., 2015. Twitter: Spezielle Abkürzungen und ihre Bedeutung. Verfügbar unter: <www.experto.de/computer/gadgets/twitter-spezielle-abkuerzungen-und-ihre-bedeutung.html/> Abgerufen 08. November 2015.

Beckett, D. et. al., 2014. RDF 1.1 Turtle Terse RDF Triple Language. Verfügbar unter: <www.w3.org/TR/turtle/> Abgerufen 20. Dezember 2015.

Berners-Lee, T., 2011. Notation 3 Logic. Verfügbar unter: <www.w3.org/DesignIssues/Notation3.html/> Abgerufen 20. Dezember 2015.

Brickley, D., Guha, R., 2014. RDF Schema 1.1. Verfügbar unter: <www.w3.org/TR/2014/REC-rdf-schema-20140225/> Abgerufen 01. Dezember 2015.

Butler, H. et. al. 2008. The GeoJSON Format Specification. Verfügbar unter: <geojson.org/geojson-spec.html/> Abgerufen 16. November 2015.

Cunningham, H., et. al., 2014. Developing Language Processing Components with GATE Version 8. Verfügbar unter: <gate.ac.uk/sale/tao/> Abgerufen 24. Januar 2016.

Cyganiak, R. et. al., 2014. RDF 1.1 Concepts and Abstract Syntax. Verfügbar unter: <www.w3.org/TR/2014/REC-rdf11-concepts-20140225/> Abgerufen 01. August 2015.

Faktenkontor, 2015. Social Media-Atlas 2015 / 2016: Social Media Nutzung in Deutschland. Verfügbar unter: <social-media-atlas.faktenkontor.de/2015/> Abgerufen 20. Februar 2016.

Gandon, F., Schreiber, G., 2014. RDF 1.1 XML Syntax. Verfügbar unter: <www.w3.org/TR/2014/REC-rdf-syntax-grammar-20140225/> Abgerufen 01.Dezember 2015.

Green, A., 2013. Twitter Database Server: Code Architecture. Verfügbar unter: <http://140dev.com/free-twitter-api-source-code-library/twitter-database-server/code-architecture/> Abgerufen 22. März 2015.

Hammer, E., 2007. Explaining OAuth. Verfügbar unter: <hueniverse.com/2007/09/05/explaining-oauth/> Abgerufen 22. März 2015.

Hern, A., 2014. Twitter buys Gnip, one of only four companies with 'firehose' access. Verfügbar unter: <www.theguardian.com/technology/2014/apr/16/twitter-buys-gnip-firehose-analytics-apple-topsy/> Abgerufen 19. März 2015.

Herring, J., Perry, M., 2012. OGC GeoSPARQL – A Geographic Query Language for RDF Data. Verfügbar unter: <portal.opengeospatial.org/files/?artifact_id=47664/> Abgerufen 24. Oktober 2015.

IETF, 2012. The OAuth 2.0 Authorization Framework. Verfügbar unter: <tools.ietf.org/html/rfc6749/> Abgerufen 22. März 2015.

Institut für Demoskopie Allensbach, 2015. Allensbacher Markt und Werbeträger Analyse. Verfügbar unter: <www.ifd-allensbach.de/awa/medien/printmedien.html#c1112/> Abgerufen 20 Februar 2016.

ISO, 2003. ISO 19112:2003 Geographic information – Spatial referencing by geographic identifiers. Verfügbar unter: <www.iso.org/iso/catalogue_detail.htm?csnumber=26017/> Abgerufen 28. Juni 2015.

Lassila, O., Swick, R., 1999. Resource Description Framework (RDF) Model and Syntax Specification. Verfügbar unter: <www.w3.org/TR/1999/REC-rdf-syntax-19990222/> Abgerufen 01. August 2015.

Laboreiro, G., et. al., 2010. Tokenizing Micro-Blogging Messages using a Text Classification Approach. Verfügbar unter: <www.researchgate.net/publication/220774822_Tokenizing_micro-blogging_messages_using_a_text_classification_approach/> Abgerufen 28. Juni 2015.

Lieberman, J., 2006. Geospatial Semantic Web Interoperability Experiment Report. Verfügbar unter: <https://portal.opengeospatial.org/modules/admin/license_agreement.php?suppressHeaders=0&access_license_id=3&target=http://portal.opengeospatial.org/files/?artifact_id=15198/> Abgerufen 17. Oktober 2015.

linguasorb, 2015. 100 Most Common English Nouns. Verfügbar unter: <www.linguasorb.com/english/most-common-nouns/> Abgerufen 24. Oktober 2015.

Motik, B. et. al., 2012. OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition). Verfügbar unter: <www.w3.org/TR/2012/REC-owl2-syntax-20121211/> Abgerufen 05. Dezember 2015.

OGC, 2012. Geography Markup Language. Verfügbar unter: <www.opengeospatial.org/standards/gml/> Abgerufen 17. Oktober 2015.

OGC, 2000. Geography Markup Language (GML) RDF Schema Definitions. Verfügbar unter: <schemas.opengis.net/gml/1.0.0/gml.rdfs> Abgerufen 17. Oktober 2015.

O'Reilly et. al. 2013. Das Twitter-Buch. 3. Auflage Köln: O'Reilly.

Patel-Schneider, P. et. al., 2012. OWL 2 Web Ontology Language Mapping to RDF Graphs (Second Edition). Verfügbar unter: <www.w3.org/TR/2012/REC-owl2-mapping-to-rdf-20121211/> Abgerufen 05. Dezember 2015.

Pena, G. 2013. Twitter Alerts: Critical information when you need it most. Verfügbar unter: <blog.twitter.com/2013/twitter-alerts-critical-information-when-you-need-it-most/> Abgerufen 12. Juli 2015.

Prud'hommeaux, E., Seaborne, A., 2008. SPARQL Query Language for RDF. Verfügbar unter: <www.w3.org/TR/rdf-sparql-query/> Abgerufen 24. Oktober 2015.

Statista, 2014. Statistiken zur Verbreitung und Nutzung des Internets weltweit. Verfügbar unter: <de.statista.com/themen/42/internet/> Abgerufen 05. Juli 2015.

Statista, 2015. Entwicklung der durchschnittlichen täglichen Nutzungsdauer des Internets in Deutschland in den Jahren 2000 bis 2015 (in Minuten). Verfügbar unter: <de.statista.com/ statistik/daten/studie/1388/umfrage/taegliche-nutzung-des-internets-in-minuten//> Abgerufen 30. Januar 2016.

Tobesocial, 2015. Studie zur Social Media Nutzung in 2015 – Wo und wie oft loggt sich Deutschland ein? Verfügbar unter: <tobesocial.de/blog/studie-social-media-nutzung-deutschland-2015-login-facebook-instagram-twitter/> Abgerufen 05. Juli 2015.

Twitter, 2015. Twitter Nutzung / Fakten zum Unternehmen. Verfügbar unter: <about.twitter.com/de/company/> Abgerufen 05. Juli 2015.

Twitter, 2015 1. API Overview. Verfügbar unter: <dev.twitter.com/overview/api/> Abgerufen 27. September 2015.

Twitter, 2015 2. Users Field Guide. Verfügbar unter: <dev.twitter.com/overview/api/users/> Abgerufen 27. September 2015.

Twitter, 2015 3. Places Field Guide. Verfügbar unter: <dev.twitter.com/overview/api/places#field_guide/> Abgerufen 27. September 2015.

Twitter, 2015 4. Processing Streaming Data. Verfügbar unter: <dev.twitter.com/streaming/overview/processing/> Abgerufen 16. November 2015.

Twitter, 2015 5. POST statuses / update. Verfügbar unter: <dev.twitter.com/rest/reference/post/statuses/update/> Abgerufen 16. November 2015.

Twitter, 2015 6. REST APIs. Verfügbar unter: <dev.twitter.com/rest/public/> Abgerufen 15. März 2015.

Twitter, 2015 7. The Streaming APIs. Verfügbar unter: <dev.twitter.com/streaming//> Abgerufen 27. September 2015.

Twitter, 2015 8. Site Streams. Verfügbar unter: <dev.twitter.com/streaming/sitestreams/> Abgerufen 27 September 2015.

Twitter, 2015 9. Tweets Field Guide. Verfügbar unter: <dev.twitter.com/overview/api/tweets/> Abgerufen 24. Oktober 2015.

Uni Stuttgart, 2015. Katastrophenvorhersage mit Twitter. Verfügbar unter: <www.uni-stuttgart.de/hkom/presseservice/pressemitteilungen/2015/18_CeBIT.html?__locale=de/> Abgerufen 12. Juli 2015.

W3C, 1998. Extensible Markup Language (XML) 1.0. Verfügbar unter: <www.w3.org/TR/1998/REC-xml-19980210/> Abgerufen 06. Juni 2015.

W3C, 2013. Extensible Markup Language (XML) 1.0 (Fifth Edition). <www.w3.org/TR/2008/REC-xml-20081126/> Abgerufen 06. Juni 2015.

W3C OWL Working Group, 2012. OWL 2 Web Ontology Language Document Overview (Second Edition). Verfügbar unter: <www.w3.org/TR/owl2-overview/> Abgerufen 05. Dezember 2015.

w3schools.com. XML Namespaces. Verfügbar unter: <www.w3schools.com/xml/xml_namespaces.asp/> Abgerufen 01.Dezember 2015.

Anhang

Anhang A: Quelltexte

```php
<?php
/**
* TwTweetCollector.php
* Extends the OauthPhirehose class to collect
* tweet objects from the Twitter streaming API
* @author Jan Hoelzer
* @version 1.0
*/
require_once('OauthPhirehose.php');
require_once('Phirehose.php');

class TwTweetCollector extends OauthPhirehose
{

  public $myDB;
  public function db_conn() {
    $this->myDB = new TwDB;
  }

  // called by the Phirehose class whenever a Tweet is received.
  public function enqueueStatus($status) {
    $tweet = json_decode($status);

    $tweet_id = $tweet->id_str;

    $tweet_json = base64_encode(serialize($tweet));

    $values = 'tweet_json = "'.$tweet_json.'", tweet_id = '.$tweet_id;
    $this->myDB->insert('tweet_json', $values);
  }
}
?>
```

```php
<?php
/**
* TwDB.php
* Database layer for TwGetTweets,
* TwTweetCollector and TwGetAttributes
* @author Jan Hoelzer
* @version 1.0
*/
class TwDB
{
 public $db_conn;
 public $db_host = 'localhost';
 public $db_user = 'TwUser';
 public $db_pw   = '********';
 public $db_name = 'TwDB';

 // Constructor of class TwDB
 function __construct() {

  try {
     $this->db_conn = mysqli_connect($this->db_host, $this->db_user, $this->db_pw, $this->db_name);
      } catch (Exception $e) {
        print 'Exception: '.$e->getMessage()."\n";
      }
  }
 public function row_exist($table,$where) {
  $row_exist = 'SELECT * FROM ' . $table .
   ' WHERE ' . $where;
  $response = mysqli_query($this->db_conn, $row_exist) or die(mysqli_error($this->db_conn));
  return mysqli_num_rows($response) > 0;
 }

 public function esc($str) {
  return mysqli_real_escape_string($this->db_conn, $str);
 }

 public function select($select) {
  $response = mysqli_query($this->db_conn, $select) or die(mysqli_error($this->db_conn));
  return $response;
 }

 public function insert($table, $values) {
  $insert = 'INSERT INTO '.$table.' SET '.$values;
  mysqli_query($this->db_conn, $insert) or die(mysqli_error($this->db_conn));
 }

 public function update($table, $values, $where) {
  $update = 'UPDATE '.$table.' SET '.$values.' WHERE '.$where;
  mysqli_query($this->db_conn, $update) or die(mysqli_error($this->db_conn));
 }
}
?>
```

```php
<?php
/**
* TwGetTweets.php
* Start as background process to collect
* tweet objects from the Twitter streaming API
* @author Jan Hoelzer
* @version 1.0
*/
require_once('TwTweetCollector.php');
require_once('TwDB.php');

const OAUTH_TOKEN  = '2708721500-ewiXF6I6TELqr1F6R7D3TscbrBQpnVlvyEQowA2';
const OAUTH_SECRET = 'lxrYhfT0jMEjp2sYYOBBTfBWTQoHCyQ6zfiLT1U2FVBRd';
const METHOD = 'filter';

// open a persistent connection to the Twitter streaming API
$stream = new TwTweetCollector(OAUTH_TOKEN, OAUTH_SECRET, METHOD);
$stream->db_conn();
// filters for tweet collection
$stream->setTrack(array('London'));
$stream->setLang('en');
$stream->consume();
?>
```

```php
<?php
/**
* TwGetAttributes.php
* Start as background process to unserialize
* tweet objects and insert the attributes into TwDB
* @author Jan Hoelzer
* @version 1.0
*/
require_once('TwDB.php');
$myDB = new TwDB;

while (true) {

  $select = 'SELECT tweet_json FROM tweet_json';
  $response = $myDB->select($select);

  while($row = mysqli_fetch_assoc($response)) {

    $tweet = unserialize(base64_decode($row['tweet_json']));

    $tweet_id = $tweet->id_str;
    if ($myDB->row_exist('tweet', 'tweet_id= '.$tweet_id)) {continue;}

    $tweet_text = $myDB->esc($tweet->text);
    if (isset($tweet->geo)) {
      $geo_lat = $tweet->geo->coordinates[0];
      $geo_long = $tweet->geo->coordinates[1];
    } else {
      $geo_lat = $geo_long = 0;
    }
    if (isset($tweet->coordinates)) {
      $coord_long = $tweet->coordinates->coordinates[0];
      $coord_lat = $tweet->coordinates->coordinates[1];
    } else {
      $coord_lat = $coord_long = 0;
    }
    $created_at = date('Y-m-d H:i:s', strtotime($tweet->created_at));
    if (isset($tweet->place->id)) {
       $place_id = $tweet->place->id;
       $place_name = $tweet->place->name;
       $place_fullname = $tweet->place->full_name;
    } else {
      $place_id = 0;
    }
    $user = $tweet->user;
    $user_id = $user->id_str;
    $screen_name = $myDB->esc($user->screen_name);
    $name = $myDB->esc($user->name);
    $location = $myDB->esc($user->location);
    $time_zone = $user->time_zone;
```

```
    // Insert (new) user
    $values = 'user_id = '.$user_id.', '.
      'screen_name = "'.$screen_name.'", '.
      'name = "'.$name.'", '.
      'location = "'.$location.'", '.
      'time_zone = "'.$time_zone.'"';

    if ($myDB->row_exist('user','user_id="'.$user_id.'")) {
      $myDB->update('user', $values, 'user_id = "'.$user_id.'");
    } else {
      $myDB->insert('user', $values);
    }

    // Insert tweet
    $values = 'tweet_id = '.$tweet_id.', '.
        'tweet_text = "'.$tweet_text.'", '.
        'geo_lat = '.$geo_lat.', '.
        'geo_long = '.$geo_long.', '.
        'coord_lat = '.$coord_lat.', '.
        'coord_long = '.$coord_long.', '.
        'created_at = "'.$created_at.'", '.
            'place_id = "'.$place_id.'", '.
        'user_id = '.$user_id;

    $myDB->insert('tweet', $values);

    // Insert (new) place
    if(isset($tweet->place->id)) {
      $values = 'place_id = "'.$place_id.'", '.
              'name = "'.$place_name.'", '.
              'fullname = "'.$place_fullname.'"';

      if ($myDB->row_exist('place', 'place_id="'.$place_id.'")) {
        $myDB->update('place', $values, 'place_id = "'.$place_id.'");
      } else {
        $myDB->insert('place', $values);
      }
    }
  }
  sleep(1);
}
?>
```

```php
<?php
/**
* TwGetJson.php
* Start as background process to unserialize
* tweet objects create gate document
* @author Jan Hoelzer
* @version 1.0
*/
require_once('TwDB.php');
$myDB = new TwDB;

while (true) {

  $select = 'SELECT tweet_json FROM tweet_json';
  $response = $myDB->select($select);

  while($row = mysqli_fetch_assoc($response)) {

    $tweet = unserialize(base64_decode($row['tweet_json']));
    $tweet_j = json_encode($tweet);
    $tweet_json = $myDB->esc($tweet_j);

    $values = 'tweet_id = '.$tweet_id.', '.
        'json = "'.$tweet_json.'"';
    $myDB->insert('json', $values);

    $gate_doc = fopen("gate_document.json", "a");
    fwrite($gate_doc, $tweet_j);
    fclose($gate_doc);

  }
  sleep(1);
}
?>
```

```php
<?php
/**
* TwGetGml.php
* Input: gate_output.xml
* Output: locations.gml (Browser)
* @author Jan Hoelzer
* @version 1.0
*/
$index = 0;

$eol = PHP_EOL;

function getRestQueryString($location)
{
  $query =
  'PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
   PREFIX dbo: <http://dbpedia.org/ontology/>
   PREFIX foaf: <http://xmlns.com/foaf/0.1/>
   SELECT ?lat ?long WHERE {
   ?s foaf:name "'.$location.'"@en .
   ?s a dbo:Place .
   ?s geo:lat ?lat .
   ?s geo:long ?long .
   }
   LIMIT 1';

  $requestUrl = 'http://dbpedia.org/sparql?'
    .'query='.urlencode($query)
    .'&format=json';

  return $requestUrl;
}

function restRequest($requestURL)
{
  $curl_handle = curl_init();

  curl_setopt($curl_handle,
    CURLOPT_URL,
    $requestURL);

  curl_setopt($curl_handle,
    CURLOPT_RETURNTRANSFER,
    true);

  $response = curl_exec($curl_handle);

  curl_close($curl_handle);

  return $response;

}



$xml = simplexml_load_file("gate_output.xml");

$location_array = (array)$xml->Location;
```

```
foreach ($location_array as $loc) {
   if(array_key_exists($loc, $locations))
   {
      continue;
   }
   else
   {
   $json = restRequest(getRestQueryString($loc));
   $response_array = json_decode($json, true);
   $long = $response_array[results][bindings][0][long][value];
   $lat = $response_array[results][bindings][0][lat][value];
   if (isset($long)) {
   $locations[$loc][long] = $long;
   $locations[$loc][lat]  = $lat;
   }
   }
}

echo '<?xml version="1.0" encoding="UTF-8" ?>'.$eol;
echo '<tw:FeatureCollection'.$eol;
echo 'xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"'.$eol;
echo 'xsi:schemaLocation="http://tw.mydomain.org/ locations.xsd"'.$eol;
echo 'xmlns:tw="http://tw.mydomain.org"'.$eol;
echo 'xmlns:gml="http://www.opengis.net/gml">'.$eol;

foreach ($locations as $location => $coordinates) {
   $index++;
   echo '<gml:featureMember>'.$eol;
   echo '<tw:location fid="location.'.$index.'">'.$eol;
   echo '<tw:geometryProperty>'.$eol;
   echo '<gml:Point srsName="EPSG:4326">'.$eol;
   echo '<gml:coordinates>'.$coordinates[long].' '.$coordinates[lat].'</gml:coordinates>'.$eol;
   echo '</gml:Point></tw:geometryProperty></tw:location></gml:featureMember>'.$eol;
}

echo '</tw:FeatureCollection>';
?>
```

Anhang B: Schema xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://example.org/" xmlns:tw="http://tw.example.org/"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
elementFormDefault="qualified" version="1.0">
	<xs:import namespace="http://www.opengis.net/gml"
schemaLocation="http://schemas.opengis.net/gml/2.1.2/feature.xsd"/>
	<xs:element name="FeatureCollection" type="tw:FeatureCollectionType"
substitutionGroup="gml:_FeatureCollection"/>
	<xs:complexType name="FeatureCollectionType">
		<xs:complexContent>
			<xs:extension base="gml:AbstractFeatureCollectionType">
				<xs:attribute name="lockId" type="xs:string" use="optional"/>
				<xs:attribute name="scope" type="xs:string" use="optional"/>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
	<xs:element name="location" type="tw:location_Type" substitutionGroup="gml:_Feature"/>
	<xs:complexType name="location_Type">
		<xs:complexContent>
			<xs:extension base="gml:AbstractFeatureType">
				<xs:sequence>
					<xs:element name="geometryProperty"
type="gml:PointPropertyType" nillable="true" minOccurs="0" maxOccurs="1"/>
					<xs:element name="name" nillable="true" minOccurs="0"
maxOccurs="1">
						<xs:simpleType>
							<xs:restriction base="xs:string">
							</xs:restriction>
						</xs:simpleType>
					</xs:element>
					<xs:element name="foreign1" nillable="true" minOccurs="0"
maxOccurs="1">
						<xs:simpleType>
							<xs:restriction base="xs:string">
							</xs:restriction>
						</xs:simpleType>
					</xs:element>
					<xs:element name="foreign2" nillable="true" minOccurs="0"
maxOccurs="1">
						<xs:simpleType>
							<xs:restriction base="xs:string">
							</xs:restriction>
						</xs:simpleType>
					</xs:element>
					<xs:element name="foreign3" nillable="true" minOccurs="0"
maxOccurs="1">
						<xs:simpleType>
							<xs:restriction base="xs:string">
							</xs:restriction>
						</xs:simpleType>
					</xs:element>
				</xs:sequence>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
</xs:schema>
```