Analyse von Semantic Web-APIs und deren Methoden

Andrea Bauer

DIPLOMARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im Juni 2011

© Copyright 2011 Andrea Bauer

Diese Arbeit wird unter den Bedingungen der *Creative Commons Lizenz Namensnennung–NichtKommerziell–KeineBearbeitung Österreich* (CC BY-NC-ND) veröffentlicht – siehe http://creativecommons.org/licenses/by-nc-nd/3.0/at/.

Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die aus anderen Quellen entnommenen Stellen als solche gekennzeichnet habe.

Hagenberg, am 24. Juni 2011

Andrea Bauer

Inhaltsverzeichnis

Kurzfassung

Diese Arbeit beschäftigt sich mit Semantic APIs, welche unstrukturierte Texte automatisch mit verschiedenen Arten von Metadaten versehen. Dazu gehört beispielsweise das Extrahieren von Eigennamen, Kategorien, Gefühlen zu bestimmten Themen, Links zu diversen Online-Plattformen, oder das Anbieten von verwandten Artikeln und passenden Bildern zu einem Text. Durch die Vielfalt dieser Informationen ergibt sich eine Großzahl unterschiedlicher Anwendungsmöglichkeiten.

In der Arbeit wurde versucht, durch die Kombination mehrerer Semantic APIs einen Mehrwert für die Verschlagwortung von Texten zu schaffen, angesichts der Grundannahme, dass jeder Dienst eigene Stärken und Schwächen aufweisen müsste. Es wurde bestätigt, dass durch die Kombination von APIs Ergebnisse für Tags zwar vermehrt werden, gleichzeitig ergeben sich aber neue Probleme, wie zum Beispiel inkonsistente Vorschläge für Tags, welche durch unterschiedliche Schreibweisen von Namen entstehen.

Da von Semantic API-Anbietern wenig über Methoden preisgegeben wird, wie die Dienste eigentlich arbeiten, ist ein weiterer Schwerpunkt der Arbeit, das Thema der natürlichen Sprachverarbeitung zu beleuchten. In Augenschein genommen wurden dabei im Detail die Eigennamenerkennung und Arbeitsweisen mit Wikipedia, welche innerhalb der APIs stark zum Einsatz kommen.

Eine abschließende Evaluierung der Eigennamenerkennung durch Semantic APIs zeigte individuelle Stärken und Schwächen der Dienste auf. Fehler in den Ergebnissen zeigten, dass die Aufgabe der Eigennamenerkennung durchaus keine triviale Aufgabe ist und trotz der intensiven Forschung noch immer Schwierigkeiten aufweist.

Abstract

This thesis deals with semantic APIs, which take unstructured text content and return different kinds of metadata, such as named entities, categories, opinions, links to online platforms, or suggestions for related articles and pictures. Through the diversity of this information, a great variety of potential applications emerges.

The thesis aims to provide additional value for the tagging of texts by combining the results of different semantic APIs, since the combination should diminish the effect of individual weaknesses. It was indeed confirmed that the use of different services provides additional suggestions for tags, but also new problems may arise, such as inconsistent suggestions for tags which originate in variant notations of names.

As there is little information about methods semantic services work with, another focus of the thesis lies in Natural Language Processing. In particular, both Named Entity Recognition and the utilization of the existing data structures of Wikipedia were examined, which most services strongly deal with.

Finally, an evaluation of the services' capabilities in Named Entity Recognition revealed individual strengths and weaknesses. Mistakes in the results showed that Named Entity Recognition does not constitute a trivial task and still faces a number of difficulties despite intensive research.

Kapitel 1

Einleitung

Das World Wide Web birgt für den Menschen eine enorme, ständig wachsende Menge an Informationen. Um dieser Menge Herr zu werden, ist es empfehlenswert, diese in einer organisierten Form aufzubereiten. So formulierte bereits im Jahr 1999 Tim Berners-Lee seine Vision des Semantic Web, bei der es darum geht, Informationen so zu strukturieren, sodass auch Maschinen in der Lage sind, diese zu verarbeiten. Nun ist seit diesem Zeitpunkt das Semantic Web gewachsen, dennoch ist noch vielen Web-Entwicklern der konkrete Nutzen des Semantic Web schleierhaft bzw. die Erstellung von strukturierten Daten unklar oder zu aufwendig.

Eine mögliche Hilfestellung für strukturierte Daten sind Semantic APIs, welche in den letzten Jahren immer zahlreicher im Internet vorzufinden sind. Es handelt sich dabei um Textanalysedienste, welche Texte in natürlicher Sprache mit semantischen Metadaten versehen. Die Art dieser Metadaten ist vielfältig, sie reicht von Eigennamen, Kategorien, Fakten und Ereignissen, Gefühlen, Informationen über den Verfasser bis zu verwandten Bildern und Artikel eines Textes. Die Möglichkeiten, wie diese Metadaten für den Menschen nutzbringend verwertet werden können, sind groß. So ist es beispielsweise möglich, Webseiten in die Linking Open Data Cloud einzubinden, d.h. Informationen für das Semantic Web aufzubereiten und für die Öffentlichkeit zur Verfügung zu stellen. Außerdem besteht die Option, Webportale besser zu organisieren, Themenseiten zu erstellen, Informationen für semantische Suchmaschinen anzubieten, verwandte Artikel zu Themen vorzuschlagen, uvm.

1.1 Zielsetzung

Anwendungen werden besonders gerne für Content-Management-Systeme entwickelt, da diese eine wesentliche Technologie für die Aufbereitung und Verwaltung von digitalen Inhalten darstellen. So gibt es beispielsweise Anwendungen, welche Redakteuren ermöglichen, Texte im CMS mit Keywords

zu versehen. In der Regel liegt der Schwerpunkt solcher Anwendungen jedoch auf nur einer Semantic API. So besteht aber die Grundannahme, dass jeder Dienst eigene Stärken und Schwächen vorweisen müsste, da die computergesteuerte Analyse der natürlichen Sprache aufgrund ihrer Komplexität eine große Herausforderung darstellt. Beispielsweise muss für mehrdeutige Wörter wie „Washington“ im Text festgestellt werden, ob es sich hierbei um einen Ort oder eine Person handelt; weiters sind Textanalysedienste oft auf bestimmte Themenbereiche spezialisiert. Das Ziel der vorliegenden Arbeit ist es, durch eine Kombination mehrerer Semantic APIs einen Mehrwert zu erreichen und Stärken und Schwächen auszugleichen.

Während von Semantic API-Anbietern hauptsächlich die Funktionen und Anwendungsmöglichkeiten ihrer Textanalysen präsentiert werden, wird wenig preisgegeben, mit welchen Methoden die Dienste eigentlich arbeiten. Ein weiterer Schwerpunkt der Arbeit ist deshalb, das Thema des Natural Language Processing näher zu beleuchten. In Augenschein genommen wird dabei besonders die Erkennung von Eigennamen, welche die meisten Semantic APIs anbieten. Außerdem folgt ein Überblick über die Verschlagwortung von Texten mit Wikipedia[1], die als größte vorliegende Enzyklopädie in den letzten Jahren eine immer wichtigere Rolle im Bereich der Textverarbeitung eingenommen hat.

Mit dem Hintergrundwissen, wie Semantic APIs eigentlich arbeiten, wird zuletzt eine Evaluierung der APIs bezüglich ihrer Fähigkeiten und Unterschiede in der Eigennamenerkennung vorgenommen. Hier wird klar erkennbar, mit welchen Schwierigkeiten die Dienste konfrontiert sind.

1.2 Aufbau

Kapitel 2 beschreibt die Problematik der aktuellen Aufbereitung von Informationen im World Wide Web und präsentiert die Idee des Semantic Web von Tim Berners-Lee. Nach der Darlegung der wichtigsten Kerntechnologien zur Umsetzung dieser Idee, folgt die Erläuterung der Linking Open Data Cloud.

Kapitel 3 führt in das Thema von Semantic APIs ein. Hier wird beschrieben, welche Anbieter es in diesem Bereich gibt, welche Metadaten für Texte von diesen extrahiert werden und welche konkreten Einsatzgebiete sich damit ergeben.

Kapitel 4 enthält die Darstellung der selbst entwickelten Anwendung Enrich für das CMS Typo3. Zuerst werden Motivation und Aufbau der Anwendung präsentiert. Eine anschließende Evaluierung und Darlegung von Erweiterungsmöglichkeiten geben Aufschluss über den Mehrwert der Anwendung.

Kapitel 5 beleuchtet schließlich die Methoden von Semantic APIs. Nach einer allgemeinen Einführung in das Thema Natural Language Processing

[1] http://www.wikipedia.org/

wird insbesondere auf die Problematiken und Techniken der Eigennamenerkennung eingegangen. Weiters erfolgt ein Überblick über die Verschlagwortung von Texten mithilfe von Wikipedia.

In Kapitel 6 wird eine Evaluierung der APIs bezüglich ihrer Fähigkeiten, Stärken und Schwächen in der Eigennamenerkennung vorgenommen.

Zuletzt gibt Kapitel 7 einen Überblick über die in der Arbeit gewonnenen Erkenntnisse.

Kapitel 2

Semantic Web

Das World Wide Web birgt eine enorme, ständig wachsende Menge an Informationen. Der große Nutzen für den Menschen besteht in der Möglichkeit, aktuelle Informationen schnell und kostengünstig zu erhalten. Diese können zu jeder Zeit an beliebigen Orten abgerufen werden.

Allerdings sind die meisten Informationen im Web nur für den Menschen aufbereitet, was ein großes Hindernis darstellt. Ein Mensch kann die Bedeutung von Informationen im Web ohne Probleme verstehen, interpretieren und mit anderen Dingen in Beziehung setzen, eine Maschine aber nicht. So kann zum Beispiel eine Suchmaschine, welche rein auf dem Finden von Zeichenketten basiert, nicht unterscheiden, ob es sich bei dem Wort *Washington* um eine Person oder einen Ort handelt. Die Verschiedenartigkeit von Informationen stellt eine weitere Schwierigkeit dar. Webseiten verwenden unterschiedliche natürliche Sprachen, basieren auf unterschiedlicher Zeichenkodierung und sind auch vom Aufbau her ungleich. Das macht es schwer, Informationen zu einem Thema zu sammeln und weiterzuverarbeiten. Auch sind manche Informationen, nach denen ein Web-User sucht, nicht explizit im Web vorzufinden. Es handelt sich dabei um die Problematik des impliziten Wissens. Hier müssen vorhandene Fakten kombiniert werden, um zu neuem Wissen zu gelangen [25].

Basierend auf diesen Schwierigkeiten stellt Berners-Lee, der Direktor des W3C[1], seine Vision des Semantic Web dar [5]: Es geht darum, Daten auf Webseiten zu strukturieren und in einer Weise aufzubereiten, sodass auch Maschinen Informationen verarbeiten können. Mithilfe einer strukturierten Umgebung könnten zum Beispiel Software-Agenten erstellt werden, welche fortgeschrittene Aufgaben für Benutzer durchführen sollen. Der fiktive Software-Agent von Berners-Lee würde etwa nicht nur die Keywords einer Krankenhaus-Webseite identifizieren können wie „treatment, medicine, physical, therapy“. Er würde auch wissen, dass Dr. Hartman montags und dienstags in diesem Krankenhaus arbeitet und mögliche Tage und Uhrzeiten

[1] http://www.w3.org/

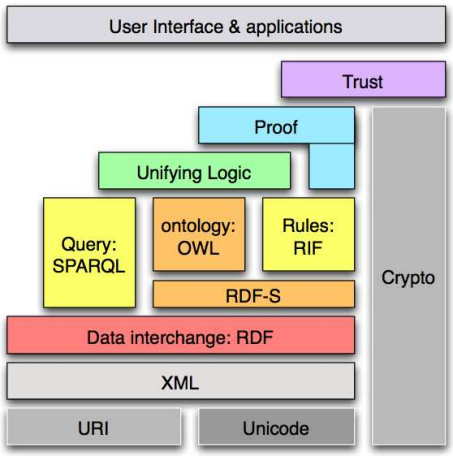

Abbildung 2.1: Semantic Web Stack.

für Termine zurückliefern.

Berners-Lee et al. beschreiben das Semantic Web folgendermaßen [5]:

> *The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation.*

Auch findet man immer wieder die Äußerung, das Ziel des Semantic Web ist es, Maschinen zu befähigen, Informationen zu verstehen. Diese Äußerung ist aber zu hochgesteckt, da sie die Beziehung zur künstlichen Intelligenz zu sehr hervorhebt. Eigentlich geht es beim Semantic Web vielmehr darum, Informationen so zu repräsentieren, dass Maschinen in der Lage sind, damit in einer Weise umzugehen, welche für den Menschen von Nutzen ist [25].

2.1 Technologien

Um die Idee des Semantic Web umsetzen zu können, wird natürlich eine Reihe von Technologien benötigt. Diese werden im Semantic Web Stack (Abb. 2.1)[2] von Berners-Lee dargestellt, wobei die unteren Schichten die Basis für die oberen bilden. Insgesamt präsentiert der Stack sowohl bereits standardisierte als auch noch in Entwicklung befindliche Technologien. Die wichtigsten Kerntechnologien werden nachfolgend näher erklärt.

2.1.1 URI

Ein Uniform Resource Identifier (URI) ist eine weltweit eindeutige Zeichenfolge, welche eine abstrakte oder physikalische Ressource kennzeichnet. Eine

[2] http://www.w3.org/DesignIssues/diagrams/sweb-stack/2006a.png

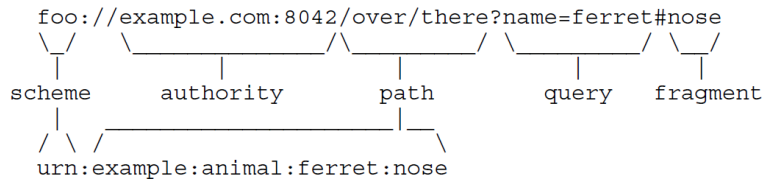

Abbildung 2.2: Aufbau von URIs.

Ressource kann also jedes Objekt sein, das klar identifiziert werden kann. Dazu gehören nicht nur Ressourcen, die im Web abrufbar sind, sondern auch Menschen, Orte, Unternehmen oder gebundene Bücher in einer Bibliothek. Eine URI besteht aus den fünf Teilen *scheme, authority, path, query* und *fragment*, wovon aber nur scheme und path in jeder URI vorhanden sein müssen (Abb. 2.2) [4].

2.1.2 XML

XML (eXtensible Markup Language) ist eine Recommendation des W3C zum Datenaustausch. Es ist eine Markupsprache, zeichnet demnach bestimmte Teile von Textdokumenten mit Informationen aus und legt deren logische Struktur fest. Diese Informationen werden auch Metadaten genannt, da sie Daten sind, welche andere Daten näher beschreiben. Dies geschieht mithilfe von XML-Tags, wie `<Person>Anna Maier</Person>`. Durch XML-Schemata, welche das Bilden spezieller Vokabulare und das Definieren von Regeln erlauben, können aber auch eigene Markup-Sprachen definiert werden. Dazu gehört beispielsweise XHTML, welches dem Web-Browser Anweisungen gibt, wie er bestimmte Teile einer Webseite darstellen bzw. interpretieren muss. Grundsätzlich bietet XML eine einfache Möglichkeit, Daten zu speichern, zu verarbeiten und zu verbreiten. Es ist eine Art „universelles Austauschformat“ [25].

Maschinen können die Bedeutung von Tags aber noch nicht interpretieren. Das bedeutet, sie können - im Gegensatz zum Menschen - zum Beispiel keinen Zusammenhang zwischen den Tags `<Apfel>` und `<Birne>` herstellen. In Bezug auf das Semantic Web bietet XML allerdings eine wichtige Grundlage zum Definieren der Sprachen RDF(S) und OWL [25].

XML-Namensräume

Beim Kombinieren von XML-Dokumenten aus verschiedenen Quellen entsteht durch die freie Wählbarkeit von XML-Namen die Möglichkeit von Namenskonflikten. Diese werden häufig durch mehrdeutige Wörter verursacht, die als Elementnamen verwendet werden. Zum Beispiel kann das Tag `<Titel>` sowohl als akademischer Titel als auch als Bezeichnung für Publikationen verwendet werden. Die Lösung zu diesem Problem sind Namensräume.

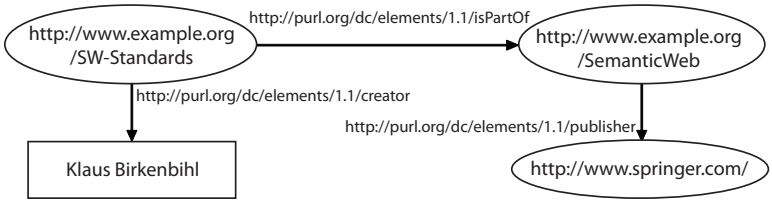

Abbildung 2.3: Beispiel für RDF-Graph (nach [8]).

URIs werden zum optionalen Bestandteil von Tagnamen und machen diese eindeutig. Wie im nachfolgenden Beispiel erkennbar, wird einem Elementnamen ein Attribut `xmlns` hinzugefügt, welches eine URI als Wert zugeordnet bekommt. Das Präfix ist optional und kann innerhalb des Elements als Abkürzung für die oft sehr langen URIs benutzt werden [25].

```
<Element-Name xmlns[:Präfix] = URI> ... </Element-Name>
```

2.1.3 RDF

Das Resource Description Framework (RDF) bietet die Möglichkeit, Daten nicht nur zu strukturieren, sondern mit Informationen über die Daten selbst anzureichern und mit anderen Informationen in Beziehung zu setzen. Beschreibungen können in der Form `Subjekt->Prädikat->Objekt` gemacht werden, wobei das Objekt selbst wiederum eine Ressource ist und zum Subjekt anderer Beschreibungen werden kann. Abbildung 2.3 stellt verknüpfte Statements in einem RDF-Graphen dar. Ovale beschreiben darauf die Ressourcen (Subjekte und Objekte), Pfeile die Prädikate und Rechtecke literale Werte. Prädikate und Ressourcen werden durch URIs eindeutig gekennzeichnet. Auf der Abbildung ist zu erkennen, dass das Kapitel *SW-Standards* von *Klaus Birkenbihl* geschrieben wurde und Teil des Buches *SemanticWeb* ist, welches vom *Springer-Verlag* veröffentlicht wurde. Die Attribute `isPartOf`, `creator` und `publisher` sind Eigenschaften des Dublin Core, eine Sammlung einfacher Metadatenstandards [8]. RDF basiert auf XML, die äquivalente XML-Struktur zum eben beschriebenen RDF-Graphen findet sich in Prog. 2.1.

Die Idee des Semantic Web ist, dass zwischen Ressourcen ein Geflecht semantischer Beziehungen im Web entsteht, aufgrund derer sich dann Anwendungen erstellen lassen [8]. Die zentrale Anfragesprache für RDF-Daten bildet SPARQL (SPARQL Protocol and RDF Query Language).

2.1.4 RDF-Schema

RDF-Schema (RDFS) bietet die Möglichkeit, mit Aussagen Taxonomien oder Ontologien aufzubauen. Es ist eine Erweiterung von RDF und erlaubt, Klas-

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/">

  <rdf:Description rdf:about="http://www.example.org/SW-Standards">
    <dc:creator> Klaus Birkenbihl </dc:creator>
    <dc:isPartOf rdf:resource="http://www.example.org/SemanticWeb"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://www.example.org/SemanticWeb">
    <dc:publisher rdf:resource="http://www.springer.com/" />
  </rdf:Description>
</rdf:RDF>
```

Programm 2.1: RDF als XML-Struktur [8].

sen und Klassenhierarchien festzulegen und Eigenschaften Werte- und Gültigkeitsbereiche zuzuweisen [8]. Dies passiert mittels der Erstellung von

- Klassen (z.B. `rdfs:Resource, rdfs:Class, rdf:Property`)
- und Prädikaten
  (z.B. `rdfs:subClassOf, rdfs:domain, rdfs:range, rdf:type`)

2.1.5 OWL

RDF-Schema bietet zwar die Möglichkeit, bereits einfache Ontologien zu erstellen. Für komplexere Modellierungen ist jedoch eine ausdrucksstärkere Sprache notwendig [25]. Die Web Ontology Language (OWL) erweitert das Vokabular von RDF-Schema um neue Klassen und Prädikate. In OWL besteht etwa die Möglichkeit, Klassen miteinander zu verknüpfen, um neue zu erzeugen. Dies geschieht z.B. durch Durchschnitts-, Vereinigungs- oder Komplementäroperationen. Die Beschreibung von Eigenschaften wird insofern erweitert, dass diese nicht nur auf einzelne Domänen und Wertebereiche beschränkt werden können, sondern dass auch deren Verhalten beschrieben werden kann. Eine Eigenschaft p ist etwa *transitiv*, wenn gilt: `(a p b)(b p c) => (a p c)` [8]. Abbildung 2.4 stellt den Sachverhalt in einem Beispiel dar. Sowohl Peter als auch Matthew sind Nachfahren von William.

2.2 Linked Data

Beim Konzept *Linked Data* geht es darum, im Web strukturierte Daten zu veröffentlichen und diese mit externen Quellen zu verlinken. Daten sollen im Web so veröffentlicht werden, dass sie maschinenlesbar sind, ihre Bedeutung klar definiert ist und sie zu externen Datensätzen verlinkt werden bzw.

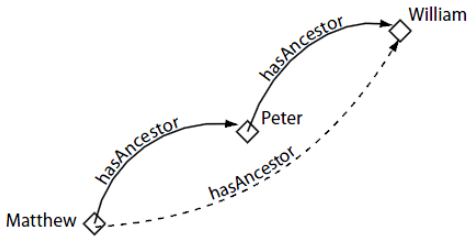

Abbildung 2.4: Beispiel für eine transitive Eigenschaft [26].

umgekehrt auf sie selbst verlinkt werden kann [10]. Berners-Lee definierte folgende Regeln, um diese Ziele zu erreichen [3]:

- *Use URIs as names for things*
- *Use HTTP URIs so that people can look up those names.*
- *When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL)*
- *Include links to other URIs. so that they can discover more things.*

Linking Open Data Cloud

Das am besten einsehbare Beispiel, wie Linked Data-Prinzipien aufgenommen und angewendet wurden, ist das Linking Open Data Project, ein W3C-Community-Projekt, welches 2007 gegründet wurde. Das Ziel ist es, lizenzfreie Daten im Web nach den Linked Data-Prinzipien in RDF zu konvertieren und zu veröffentlichen. Jeder kann an diesem Projekt teilnehmen, indem er selbst Datensätze in RDF-Form zu Verfügung stellt und mit bereits existierenden verlinkt. Anhand eines Diagramms (Abb. 2.5)[3] können der Umfang und die Vielfalt der vorhandenen Datensätze in der Linking Open Data Cloud eingesehen werden. Jeder Knoten stellt einen eigenen, in RDF-Format publizierten Datensatz dar. Pfeile bilden Links zwischen den Datensätzen ab, wobei dickere Pfeile eine höhere Anzahl von Verbindungen skizzieren [10]. Die Art der Daten ist vielfältig, die Bandbreite der Inhalte reicht von Medien, geographischen Daten, wissenschaftlichen Veröffentlichungen, benutzergenerierten Inhalten bis zu Biowissenschaften und Regierungsdaten. Auch Datensätze, welche Informationen aus verschiedenen Domänen verbinden, sind sehr wichtig, um das Abkapseln einzelner Daten in thematisch getrennte Inseln zu verhindern. DBpedia[4] beispielsweise veröffentlicht die Daten aus den Infoboxen in Wikipedia in RDF-Form und deckt damit eine große Spanne an Themen ab. Viele Anbieter haben begonnen, ihre Daten mit jenen auf

[3] Diagramm von Richard Cyganiak und Anja Jentzsch (http://lod-cloud.net/)
[4] http://dbpedia.org/

DBpedia zu verlinken, was es zu einem der zentralen Knoten in der Linking Open Data Cloud macht [9].

Applikationen

Verschiedene Anwendungen bestehen bereits, welche Linked Data sinnvoll und gewinnbringend einsetzen [10]:

Linked Data Browsers: Spezielle Browser erlauben Benutzern, anhand von RDF-Links zwischen verschiedenen Datenquellen zu navigieren. Beispielsweise können Informationen über eine Entität gesammelt und für den Endnutzer in einer Ansicht zusammengestellt werden.

Linked Data Search Engines: Es gibt eine Reihe von Suchmaschinen, welche Linked Data durchsuchen und für Endanwender anbieten. Dabei können zum Beispiel Suchanfragen und -ergebnisse für menschliche Endanwender aufbereitet werden.

Andere Dienste stellen spezielle APIs zur Entwicklung von Applikationen zur Verfügung, mithilfe derer RDF-Dokumente durch Angabe von URIs oder Keywords gefunden werden können. Die Idee dieser Art von Suchmaschinen ist, dass nicht jede Applikation eine eigene Infrastruktur zum Crawlen und Indizieren von RDF-Dokumenten erstellen braucht.

Domain-specific Applications: Applikationen können auch zu domänenspezifischen Zwecken erstellt werden. Dabei entstehen Mashups mit ausgewählten Datensätzen der Linking Open Data Cloud.

DBpedia Mobile ist zum Beispiel eine Smartphone-Anwendung für Touristen. Basierend auf der aktuellen GPS-Position des Smartphones werden auf dem Screen Informationen, Bilder und Kritiken zu nahe gelegenen, sehenswerten Plätzen zur Verfügung gestellt.

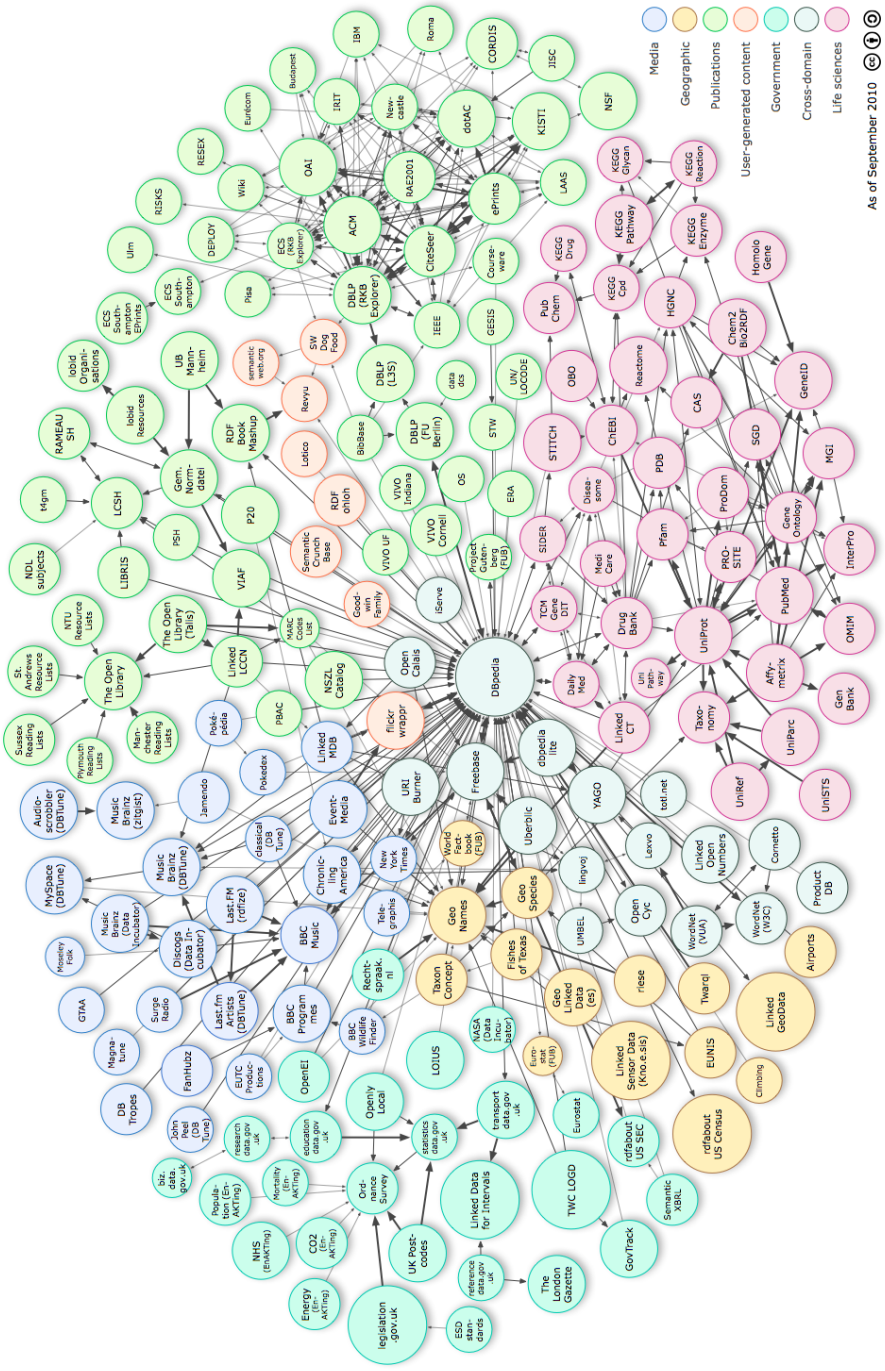

Abbildung 2.5: Linking Open Data Cloud-Diagramm.

Kapitel 3

Semantic APIs

Eine zunehmende Anzahl von Semantic APIs ist mittlerweile im World Wide Web vorzufinden. Mithilfe ihrer soll der Vision des Semantic Web ein Schritt näher getreten werden.

Dienste wie OpenCalais[1] oder Zemanta[2] sind im Grunde Textanalysedienste und arbeiten mit Methoden des Natural Language Processing. Sie werden gerne in Content-Management-Systemen integriert und erlauben es dort, Texte zu analysieren und diese mit semantischen Informationen auszustatten. So kann Content besser organisiert und die Qualität von Inhalten verbessert werden.

Dotsika beschreibt den Begriff „Semantic APIs“ folgendermaßen [17]:

> *APIs that take unstructured text (including web pages) as input and return the content's contextual framework are termed semantic APIs.*

Wie in Abbildung 3.1 ersichtlich, ermöglichen es diese Dienste also, unstrukturierte Texte bzw. Webinhalte mit semantischen Metadaten auszuzeichnen. Die Arten von Metadaten sind generell sehr unterschiedlich und werden im folgenden Kapitel präsentiert.

Für gewöhnlich generieren die APIs als Output RDF, XML, JSON oder reinen Text. Viele der Dienste sind zwar kommerziell, bieten aber auch eine kostenfreie Nutzung mit einer limitierten Anzahl von API-Aufrufen an.

3.1 Anbieter

Im Folgenden werden die bekanntesten Semantic APIs und ihre Eigenschaften vorgestellt. Jede API hat einen eigenen Schwerpunkt und bietet unterschiedliche Arten von Metadaten an, welche in Folge für Anwendungen

[1] http://www.opencalais.com/
[2] http://www.zemanta.com/

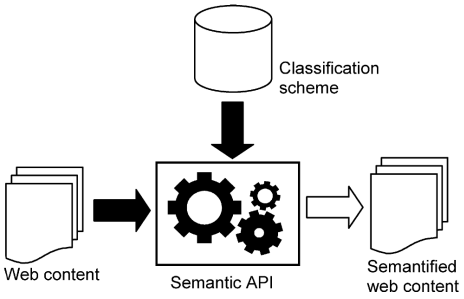

Abbildung 3.1: Semantic APIs [17].

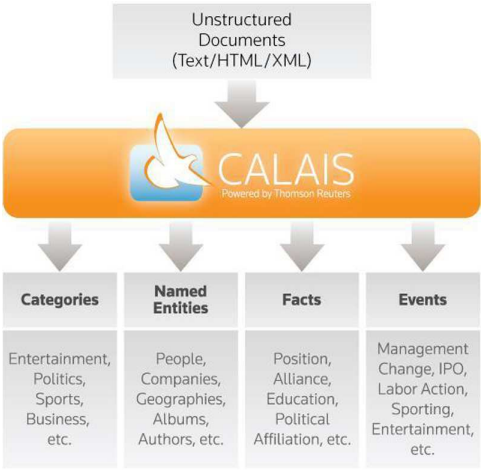

Abbildung 3.2: Aufgaben von OpenCalais.

genutzt werden können. Die Informationen über die Dienste wurden den entsprechenden Produktseiten entnommen.

3.1.1 OpenCalais

Eine der bekanntesten Semantic APIs stellt OpenCalais dar. OpenCalais bietet einen Webservice an, welcher automatisch semantische Metadaten zu Texten erzeugt. Mithilfe von Natural Language Processing, Machine Learning und anderen Methoden werden die Texte analysiert und fünf Arten von Metadaten geliefert (Abb. 3.2)[3]: *Kategorien, Social Tags, Named Entities, Facts* und *Events*. Unterstützt für alle fünf Arten von Metadaten wird von OpenCalais vorerst nur die Sprache Englisch; im Französischen und Spanischen werden gegenwärtig nur Entitäten erkannt.

[3] http://tinyurl.com/6crya5y

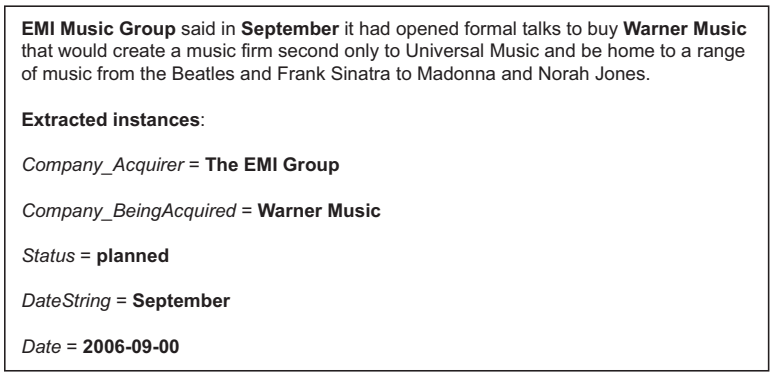

Abbildung 3.3: Beispiel für extrahiertes Event *Acquistion*.

Named Entities: Die Erkennung von Eigennamen in Texten wird von den meisten Semantic APIs angeboten und ist auch ein grundlegendes Feature von OpenCalais. Es werden eine Reihe von semantischen Klassifizierungen unterstützt, wie *Anniversary, City, Company, Continent, Country, Person, Organization*, etc. Beim Extrahieren einiger Eigennamen stellt OpenCalais auch URIs zur Verfügung. Diese zeigen in das Calais Repository der Linking Open Data Cloud. Es handelt sich dabei um eine XML-Seite, welche zusätzliche Informationen zur Entität wie auch Verbindungen zu verwandten URIs in DBpedia, Freebase[4] oder Reuters[5] enthält.

Events und Facts: Neben der Erkennung von Eigennamen bietet OpenCalais die Möglichkeit, *Events* und *Facts* aus Texten zu extrahieren. Für bestimmte Ereignisse (z.B. *Acquisition, Merger, MovieRelease*) werden dabei einzelne Daten und Fakten ausgelesen und angegeben. Dafür gibt es von OpenCalais eine Reihe von vordefinierten Templates (Abb. 3.3)[6].

Kategorien: Der Dienst bietet außerdem die Möglichkeit einer Dokumentenklassifizierung an. Es wird versucht, herauszufinden, um welches Thema es sich in Texten handelt, z.B. um *Sports, Environment* oder *Politics*. Dabei ist es auch möglich, dass ein Text in mehrere Kategorien eingeordnet wird.

Social Tags: Social Tags sind ein neues Feature von OpenCalais und stellen eigentlich Begriffe aus Wikipedia dar. Durch sie wird nachzuahmen versucht, mit welchen Stichwörtern eine Person einen Text versehen würde. Zum Beispiel würde einem Text, der von Barack Obama und einem neuen Gesetz handelt, auf jeden Fall das Social Tag *U.S. Legislation* zugewiesen. Es wird

[4] http://www.freebase.com/
[5] http://www.reuters.com/
[6] http://tinyurl.com/3gb5k34

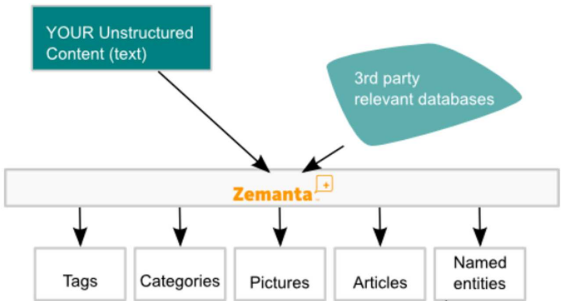

Abbildung 3.4: Aufgaben von Zemanta.

somit versucht, allgemeine Tags für spezifischen Content zur Verfügung zu stellen, welche dann genützt werden können, Texte zu organisieren.

3.1.2 Zemanta

Zemanta liefert zu unstrukturierten Texten folgende Metadaten: *Bilder, Artikel, Links, Keywords* und *Kategorien* (Abb. 3.4)[7]. Der Dienst bietet eine Einstiegsstelle zu vielen Inhalts-Datenbanken, um zu Kontextdaten zu gelangen. Mit statistischen Methoden wird das Framework eines Textes mit der internen Datenbank verglichen. Durch Machine Learning-Methoden und der Benutzereingabe von Endanwendern des Zemantawidgets kann das System weiter verbessert werden.

Bilder: Die Bilder, welche Zemanta zurückliefert, stammen aus Portalen wie Wikipedia, Flickr[8] und anderen Bildanbietern. Es wird versucht, jene Bilder, welche die Zustimmung des Autors erfordern würden, herauszufiltern.

Artikel: Zemanta bietet auch Links zu verwandten Artikeln im Internet an. Dazu gehören reguläre Newsportalseiten aber auch Blogartikel, wobei der Schwerpunkt hier auf „hochangesehenen“ Blogs liegt. Es wird dabei versucht, möglichst viele Weltansichten, Regionen, und Themen abzudecken. Im Jänner 2011 bestand die Anzahl der verfolgten Quellen aus 50000, wobei täglich neue dazukommen. Diese werden manuell untersucht, um Spam zu vermeiden.

Links: Die API stellt für einzelne Phrasen oder Begriffe in Texten Links zur Verfügung. Bei diesen Begriffen handelt es sich generell um Eigennamen, deren Typ, wie z.B. *Person* oder *Organization* wenn möglich ebenfalls angegeben wird. Die Links verweisen zu Objekten von bekannten Datenbanken.

[7] http://code.zemanta.com/bostjan/clipart/api/general.png
[8] http://www.flickr.com/

Dazu gehören u.a. Wikipedia, IMDB[9], MusicBrainz[10] oder Amazon[11].

Keywords: Zur Verschlagwortung von Texten stellt Zemanta jeweils acht Keywords zur Verfügung. Dabei handelt es sich einerseits um Begriffe, die auf den Worten und Phrasen des Textes basieren. Andererseits werden auch Vorschläge für Keywords gemacht, die im Text nicht explizit als Phrasen auftauchen. Es handelt sich dabei um verwandte Themen und Konzepte.

Kategorien: Wie OpenCalais extrahiert Zemanta auch Kategorien von Texten.

3.1.3 AlchemyAPI

Weiters zählt AlchemyAPI[12] zu den Anbietern von semantischen Textanalysen; hier werden folgende Funktionalitäten angeboten: *Named Entity Extraction, Keyword Extraction, Sentiment Analysis, Document Categorization, Concept Tagging, Language Detection* und *Structured Content Scraping.* Neben dem Englischen werden je nach Art der Aufgabe auch andere Sprachen unterstützt wie Deutsch, Französisch, Italienisch, Portugiesisch, Russisch, Spanisch und Schwedisch.

Named Entity Extraction: AlchemyAPI identifiziert in Texten Eigennamen und unterstützt eine große Menge an Klassifizierungstypen wie *Automobile, Anniversary, City* oder *Company.* Kann eine Entität eindeutig identifiziert werden, werden auch Verbindungen zur Linking Open Data Cloud zur Verfügung gestellt.

Keyword Extraction: Die Keyword Extraction nimmt die Identifizierung von wichtigen Wörtern und Phrasen in einem Text vor, mit welchen dieser werden kann.

Concept Tagging: Beim Concept Tagging geht es darum, Konzepte in einem Text zu finden bzw. Abstraktionen zu machen. Zum Beispiel würde einem Text, welcher von Hillary Clinton, Barbara Bush und Laura Bush handelt, das Konzept *First Ladies of the United States* zugewiesen.

Sentiment Analysis: Die Stimmungsanalyse identifiziert positive bzw. negative Stimmungen/Meinungen in Texten und versieht diese mit einem numerischen Wert. Die Analyse ist auf Ebene des Textes selbst verfügbar, auf Ebene der Entitäten und der Keywords.

[9] http://www.imdb.com/
[10] http://musicbrainz.org/
[11] http://www.amazon.com/
[12] http://www.alchemyapi.com/

Document Categorization: Bei der Dokumentenkategorisierung werden Texte in eine von zwölf vorhandenen Kategorien eingeordnet.

Language Detection: AlchemyAPI unterstützt weiters die Spracherkennung von Texten in 97 verschiedenen Sprachen.

Structured Content Scraping: Mithilfe von Structured Content Scraping können strukturierte Daten aus Webseiten extrahiert werden, wie Produktmerkmale, Beschreibungen, Kosten, etc.

3.1.4 OpenAmplify

OpenAmplify[13] nimmt unstrukturierten Text entgegen und liefert die enthaltenen *Topics* und *Actions* mit weiteren Informationen darüber. Mittels Extrahieren von *Demographics* und *Style* wird außerdem versucht, Daten über den Autor selbst und dessen Schreibstil herauszufinden. In einer Grafik (Abb. 3.5)[14] ist die Gesamtheit der zur Verfügung stehenden Metadaten ersichtlich.

Topics: Die Topics bestimmen, worum es im entsprechenden Text geht. Zusatzinformationen geben an, ob der Autor des Textes positiv oder negativ über die enthaltenen Dinge schreibt und ob er Auskunft über ein Thema gibt oder Hilfe wünscht. Bei Personen, Organisationen und Orten kann zusätzlich der Entitätstyp identifiziert werden. Weiters werden die enthaltenen Domänen (z.B. *Automobile*, *Medicine*) eines Textes bestimmt.

Actions: Die Aktionsanalyse gibt an, welche Aktionen im Text vorgekommen sind, gerade vorkommen oder vorkommen werden. Sie werden nicht nur als einfache Verben sondern in Kombination mit ihrem Objekt („buy a car“) angegeben. Es werden sowohl die Zeitformen eines Verbs als auch jegliche temporale Referenzen („this week“) aufgeführt. Der Wert *Decisiveness* bestimmt, mit welcher Wahrscheinlichkeit eine Aktion in der Zukunft passieren wird. *Guidance* gibt wiederum an, ob ein Autor zu einer Aktion Beratung gibt oder wünscht.

Demographics: OpenAmplify extrahiert weiters Informationen über den Autor des Textes selbst. Es handelt sich dabei um Informationen bezüglich Alter, Geschlecht und Bildungsgrad.

[13] http://www.openamplify.com/
[14] http://www.openamplify.com/sites/default/files/OpenAmplify_Integration_main.jpg

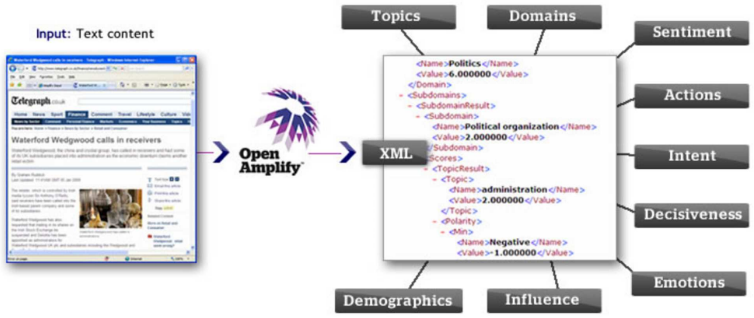

Abbildung 3.5: Aufgaben von OpenAmplify.

Style: Die Ergebnisse der Styleanalyse geben an, ob ein Text sehr viel Slang enthält und wie „flamboyant“, also lebhaft und farbenprächtig dieser ist.

3.2 Anwendungen

Semantic APIs sind eine mögliche Hilfestellung, unstrukturierte Inhalte mit semantischen Metadaten anzureichern. Sie ermöglichen eine beachtliche Reihe interessanter Anwendungen, von denen eine Auswahl in diesem Kapitel vorgestellt wird. Auffallend dabei ist, dass eine Großzahl dieser Anwendungen in Content-Management-Systemen integriert werden kann. Der Grund ist jener, dass diese Systeme eine wesentliche Technologie für die Aufbereitung und Verwaltung von digitalen Inhalten darstellen.

3.2.1 Anreicherung von Artikeln

Eine beliebte Anwendung für Semantic APIs stellt die Anreicherung von Web-Artikeln dar. Es geht darum, dass Texte bzw. Artikel von Redakteuren in Blog- und Content-Management-Systemen sehr schnell mit Zusatzinformationen ausgestattet werden.

Mit Hilfe des Zemanta-Widgets können beispielsweise Texte sehr einfach und schnell mit Bildern, Tags, In-Text-Links und verwandten Artikeln versehen werden (Abb. 3.6). Dabei werden die von den Benutzern verfassten Texte im Hintergrund analysiert und Vorschläge für die eben erwähnten Metadaten erstellt. Der Benutzer hat die Möglichkeit, mit wenigen Klicks seinen Artikel anzureichern. Zemanta unterstützt eine Reihe unterschiedlicher Blog- und Content-Management-Systeme. Das Widget bietet außerdem die Option für den Benutzer, dem System auch die eigene Blog-Adresse zur Verfügung zu stellen, sodass die eigenen Blogartikel auch als Artikel-Empfehlung für andere Benutzer auftauchen.

Basierend auf der Semantic API OpenAmplify existiert eine ähnliche Ap-

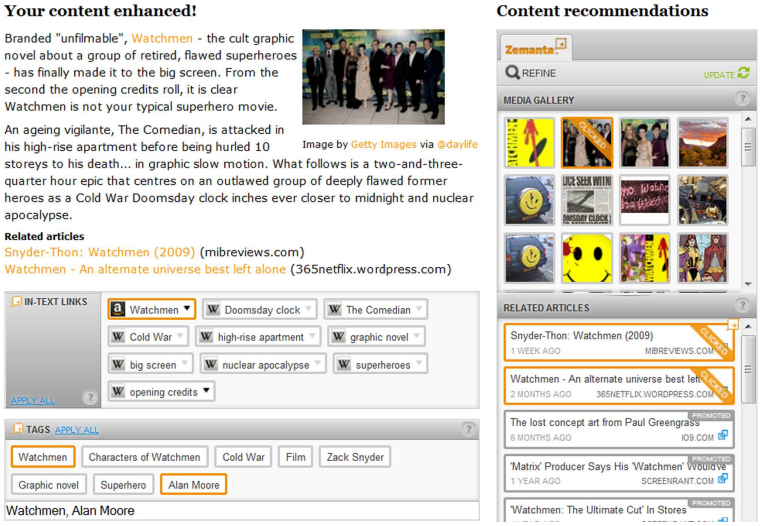

Abbildung 3.6: Demoanwendung von Zemanta.

plikation für das CMS Drupal[15]. Auch hier werden die einzelnen Text-Artikel des Systems analysiert, jedoch werden diese mit folgenden Zusatzinformationen ausgestattet:

- Mithilfe einer Tagcloud bzw. Mood Cloud werden die Themen eines Textes extrahiert und dargestellt. Jedes der Tags besitzt eine von drei Farben (grün, blau oder rot), welche die Einstellung des Verfassers zu den Begriffen anzeigen.
- Anhand einiger Webservices wird weiterer Content eingebracht: verwandte Artikel, passende Bilder und Produkte.
- Im Block „About this page“ werden Informationen über die Seite angezeigt: Eine Liste stellt Domänen, Themen und Orte der Seite dar. Zusätzlich werden die von OpenAmplify verfügbaren Daten über Autor und Schreibstil in Liniendiagrammen präsentiert.

3.2.2 Unterstützung für Taxonomiesysteme

Auch als Unterstützung für Taxonomiesysteme können Semantic APIs hilfreich sein. So gibt es im CMS Drupal die Option, die „Calais Collection“ zu integrieren. Diese enthält eine Reihe von nützlichen Anwendungen für das CMS, darunter die Tagging Integration. Hier besteht die Möglichkeit, die von OpenCalais gelieferten Metadaten zu Texten in Drupal-Knoten, den Inhaltselementen des CMS, zu integrieren. Dazu werden die Metadaten in das Tagging-Vokabular von Drupal aufgenommen. Zum Beispiel kann ein Artikel über Nintendo DS mit entsprechenden Eigennamen ausgestattet werden.

[15] http://drupal.org/

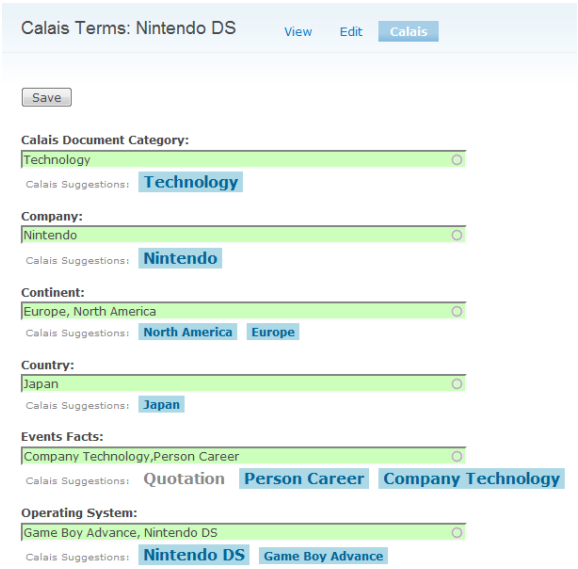

Abbildung 3.7: Tagvorschläge von OpenCalais für den englischen Text Nintendo DS aus Wikipedia.

Anhand einer Eingabemaske (Abb. 3.7) hat der Benutzer die Möglichkeit, die Vorschläge von OpenCalais zu verwalten und auch eigene Tags hinzuzufügen. Die Größe der Tags gibt jeweils an, in welchem Ausmaß das Tag von OpenCalais auch auf den Text zutrifft. Mit diesen Informationen können etwa sehr einfach und schnell Tagclouds erstellt werden.

3.2.3 Daten für semantische Suchmaschinen

Anwendungen wie Calais Marmoset oder AlchemySEO können die Inhalte von Webseiten für eine „intelligente Suche“ aufbereiten. Dies geschieht, indem sie Suchmaschinen beim Indizieren von Webseiten semantische Metadaten anbieten. Wird eine Seite von einem Browser aufgerufen, bleibt der Inhalt der Webseite unverändert. Steuert jedoch ein Webcrawler die Seite an, wird die entsprechende Semantic API der Anwendung angesteuert, um die Inhalte der Webseite mit semantischen Metadaten auszustatten. Mithilfe von Mikroformaten und auch Html-Meta-Keywords werden diese in die Seite eingebaut und der Suchmaschine zur Verfügung gestellt.

3.2.4 Mashups

Im Weiteren besteht für Anwendungen die Möglichkeit, die von Semantic APIs mitgelieferten Links in die Linking Open Data Cloud zu verwerten. Damit können Mashups erstellt werden, welche die vernetzten Daten aus mehreren Web-Datenquellen miteinander kombinieren. Ein Beispiel einer sol-

chen Anwendung ist das Calais Geo Modul für das CMS Drupal. Hier werden die OpenCalais-Tags Country, City und State/Province, welche aus von Redakteuren verfassten Texten extrahiert werden, in einer Google Maps Karte dargestellt. Benötigt werden dazu die Längen- und Breitengrade dieser Entitäten, die im OpenCalais RDF-Repository verfügbar sind, aber auch direkt vom OpenCalais Webservice übertragen werden.

3.2.5 Einsatz in Medienplattformen

Semantic APIs werden gerne in diversen Medienportalen eingesetzt. Sie dienen dabei beispielsweise der Kategorisierung von Themen, einer verbesserten Suchfunktionalität, der Suchmaschinenoptimierung, der Erstellung von Themenseiten und dem Vorschlagen von verwandten Artikeln. Auch erfolgt bereits die Einbindung von Plattformen in die Linking Open Data Cloud.

Am Beispiel ViewChange.org, einem interessanten Projekt im Bereich Semantic Web, wird dargestellt, wie so ein Einsatz aussehen kann. Es handelt sich dabei um eine semantische Videoplattform, die positive Veränderungen bezüglich Hunger, Armut und Krankheit in Entwicklungsländern vermitteln möchte. Semantic APIs kommen ins Spiel, indem sie Videotranskripte analysieren und semantische Entitäten zu Videos liefern. Dies birgt folgende Möglichkeiten:

- Die Videos können nach Themen kategorisiert werden, was das Suchen und Finden auf der eigenen Seite erleichtert.
- Die Themen werden semantisch mit passenden Freebase- und DBpedia-Entitäten verlinkt, wodurch eine aktive Teilnahme in der Linking Open Data Cloud stattfindet.
- Zusätzlich werden mithilfe der Themen verwandte Videos, Artikel und Wege, aktiv zu werden, aufgespürt, welche neben den Videos als weiterführende Informationen präsentiert werden.

DailyMe[16] setzt Semantic APIs auf eine etwas andere Weise ein. Es handelt sich dabei um ein personalisiertes Nachrichtenportal, welches Nachrichten aus vielen bedeutenden Quellen zusammenbringt und den Interessen des Benutzers entsprechend zusammenstellt. Die Funktionsweise ist folgende: Alle Beiträge, die ein Besucher auf der Webseite liest, werden von OpenCalais mit semantischen Metadaten versehen und getrackt. So kann DailyMe die Interessen des Benutzers herausfinden und dementsprechende Nachrichten anzeigen. Auf „My Newstogram“, einer Art Profilseite des Benutzers, ist einsehbar, für welche Kategorien, Themen, Leute, Unternehmen und Entitäten sich dieser interessiert. Die Interessen eines Benutzers zu kennen, ermöglicht in Folge auch die Schaltung zielgerichteter Anzeigen. Die dem Portal zugrunde liegende Technologie wird außerdem Drittanbietern zum Einsatz angeboten [57, 63].

[16] http://www.dailyme.com/

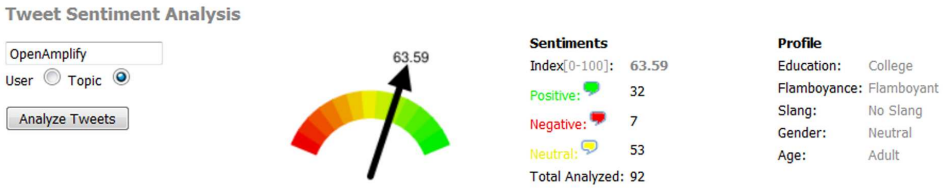

Abbildung 3.8: Analyse des Themas „OpenAmplify“ auf Tweetsentiments.com.

3.2.6 Analyse von Beiträgen in Sozialen Netzwerken

Das Analysieren von Posts in sozialen Netzwerken stellt ein weiteres Anwendungsgebiet für Semantic APIs dar.

Tweetsentiments.com beispielsweise nimmt Beiträge aus Twitter[17], und stellt mithilfe der OpenAmplify-API die aktuelle Stimmung in der Twittergemeinde fest. Außerdem können die Stimmung bzw. die Eigenschaften einzelner Benutzer oder Themen bestimmt werden (Abb. 3.8).

Twopular[18] ist eine Anwendung, welche sich den aktuellen Themen in Twitter widmet und Trends aufzeigt. Das aktuelle Projekt dieser Plattform ist es, die Trends zusätzlich mit Tags von OpenCalais auszustatten und somit Beziehungen zwischen diesen herzustellen.

Radian6[19] ist im Gegensatz zu den eben genannten Anwendungen ein professionelles, sehr umfangreiches Produkt und vertraut seit Kurzem auf die Unterstützung von Semantic APIs wie OpenCalais oder OpenAmplify. Es handelt sich dabei um eine Social Media Monitoring-Plattform, welche die Beiträge und Konversationen in zahlreichen Online-Communities, Blogs News-Portalen und Foren untersucht und Einblicke ins Social Web liefert. Mithilfe Radian6 können Unternehmen herausfinden, was sich im Social Web abspielt, wie Leute über bestimmte Themen, z.B. das eigene Unternehmen, denken und als Folge Maßnahmen treffen, deren Erfolge wiederum evaluiert werden können. Das unlängst erneuerte Angebot von Radian6 Insights stellt zusätzliche Informationen zu Beiträgen und Quellen zur Verfügung, welche die Analysemöglichkeiten für Unternehmen erweitern:

- Dazu gehören grundlegende geo- und demographische Daten.
- Daten der Klout-API[20] liefern Informationen über die Einflussnahme im Social Web, beispielsweise, welche Leute (Twitter, Facebook[21]) die Masse zu einem bestimmten Thema beeinflussen.
- Semantic APIs kommen via OpenCalais und OpenAmplify ins Spiel,

[17] http://twitter.com/
[18] http://twopular.com/
[19] http://radian6.com/
[20] http://klout.com/
[21] http://facebook.com/

welche Entitäten und Themen der Beiträge extrahieren bzw. Stimmungen zu Themen liefern.

3.2.7 Brainstorming

Eine alternative Applikation, in der Semantic APIs sinnvoll zum Einsatz kommen, ist Knowledge Storm[22], eine mobile Anwendung zum Brainstorming. Sie bringt verschiedene Informationen auf intelligente Art und Weise zusammen. Es werden dabei lokal gespeicherte SMS oder E-Mails mit Informationsquellen aus dem Internet, wie etwa der Reuters Knowledge API[23] kombiniert. Dies passiert durch die Erstellung von „Knowledge Storms". Möchte sich der Benutzer näher mit einem Thema beschäftigen, hat er die Möglichkeit, ausgehend von einer E-Mail, SMS oder kurzen Notiz, einen Knowledge Storm anzulegen. Nachdem die Texte von OpenCalais analysiert wurden, werden verwandte Quellen zum Thema vorgeschlagen. Das sind entweder weitere lokal gespeicherte Informationen auf dem Handy oder passende Informationen aus dem Internet. Es besteht nun die Möglichkeit, diese Vorschläge in den Knowledge Storm aufzunehmen oder sie auch wieder zu entfernen. Durch Ordnen der Informationsstücke werden diesen Gewichtungen zugewiesen.

[22] http://knowledgestorm.mobi/
[23] http://http://thomsonreuters.com/

Kapitel 4

Beispielanwendung Enrich

4.1 Motivation

Enrich ist eine Anwendung, welche auf Basis von Semantic APIs erstellt wurde und eine Extension für das freie Content-Management-System Typo3[1] darstellt. Wie die im vorausgehenden Kapitel beschriebene Anwendung Zemanta, ermöglicht Enrich, einen im CMS verfassten Artikel mit Zusatzinformationen zu versehen.

Im Wesentlichen dient Enrich also als Unterstützung für Redakteure. Verfasste Texte können von ihnen per Button-Klick an Semantic APIs gesendet werden, welche dann Keywords zur Verschlagwortung präsentieren. Dadurch fällt der Aufwand für den Benutzer weg, selbst über geeignete Keywords für den Text nachzudenken. Stattdessen hat er die Möglichkeit, mit nur wenigen Klicks aus den präsentierten Vorschlägen passende Keywords auszuwählen.

Weiters können mit Enrich Text-Artikel mit Medien angereichert werden. Auf Basis der ausgewählten Keywords wird nach passenden Bildern, Links zu verwandten Artikeln und Videos gesucht, die dem Redakteur zur Anreicherung des verfassten Textes angeboten werden. Hierbei werden diverse Content-APIs wie Flickr, Yahoo! BOSS[2] und YouTube[3] angesteuert. Mit einfachen Klicks können Artikel also schnell mit Medien versehen und somit ansprechender gestaltet werden. Außerdem besteht die Möglichkeit, einen Text mit Links zu verwandten Artikeln auszustatten. So können Lesern zu einem Thema noch mehr Informationen zur Verfügung gestellt werden.

Was diese Anwendung von bereits bestehenden abhebt, ist die Kombination verschiedener Semantic APIs bei der Suche nach geeigneten Keywords. Die Idee entstand aus der Grundannahme, dass jeder Dienst eigene Stärken und Schwächen vorweisen müsste. So sind Textanalysedienste meist auf bestimmte Domänen und deren Vokabular trainiert; die Qualität der Disam-

[1] http://typo3.org/
[2] http://developer.yahoo.com/search/boss/
[3] http://www.youtube.com/

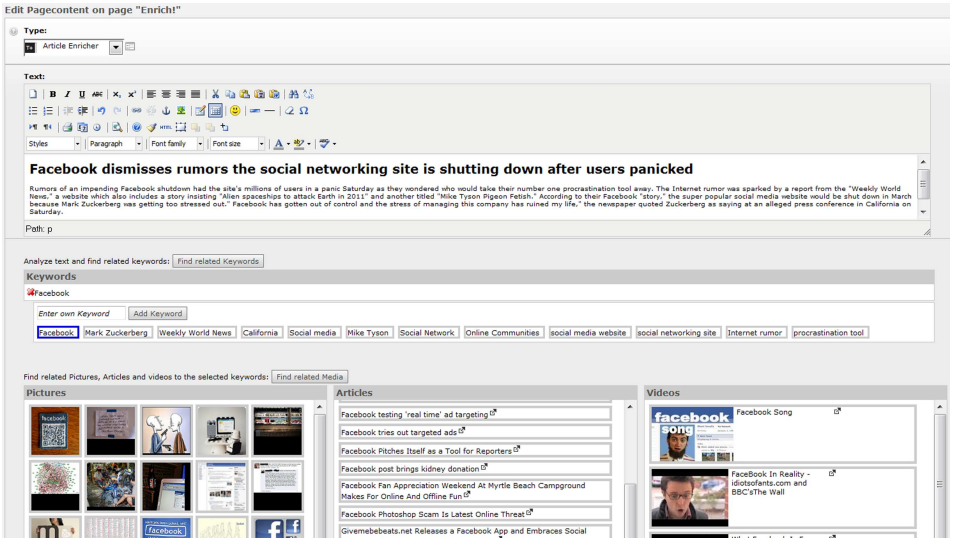

Abbildung 4.1: Inhaltselement „Article Enricher“ (Artikel aus http://tinyurl.com/45nwm3b).

biguierung von mehrdeutigen Wörtern ist unterschiedlich; außerdem sind die Dienste auf verschiedene Sprachen ausgelegt. Das Ziel ist, durch Kombinieren der einzelnen Stärken treffende und bessere Keywords zu ermitteln. Da für Typo3 noch keine ähnliche Anwendung existiert, wurde dieses CMS als Gastsystem ausgewählt.

4.2 Ergebnis

Um eine bessere Vorstellung von der Anwendung zu bekommen, sollen zunächst deren Erscheinungsbild und Funktionsweise präsentiert werden.

Abbildung 4.1 stellt das Inhaltselement „Article Enricher“ der Extension im Typo3-Backend dar. Hier hat der Redakteur die Möglichkeit, im Rich Text Editor einen Artikel zu verfassen und sich anschließend Vorschläge für Keywords von den Semantic APIs präsentieren zu lassen. Wurden die gewünschten Keywords ausgewählt bzw. eigene Keywords in das Textfeld eingegeben, kann basierend auf der Auswahl nach verwandten Medien gesucht werden. In drei Spalten erscheinen passende Bilder-, Artikel- und Videoeinträge, welche mit Klick dem verfassten Text im RTE beigefügt werden. Anschließend kann das Inhaltselement gespeichert und im Frontend angesehen werden.

Weiters gibt es im Extension-Manager von Typo3 einige Konfigurationsmöglichkeiten zur Anwendung. Das entsprechende Interface wird in Abbildung 4.2 dargestellt. Hier kann angegeben werden, welche der Semantic APIs OpenCalais, AlchemyAPI und Zemanta überhaupt angesteuert werden sollen. Diese sind dafür verantwortlich, den Textartikel zu analysieren und

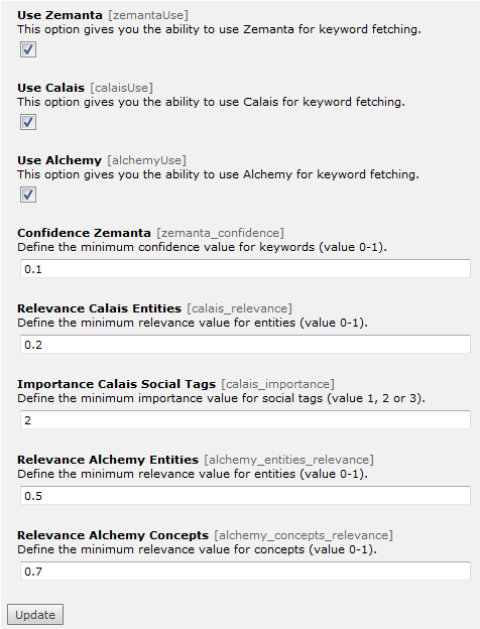

Abbildung 4.2: Konfigurationsmöglichkeiten der Anwendung.

liefern zu diesem Keywords mit verschiedenen Werten zurück. Die Werte skizzieren, in welchem Ausmaß bzw. mit welcher Sicherheit ein Keyword auch auf einen Text zutrifft. Hier besteht die Möglichkeit, je API einen Schwellwert festzulegen, welcher angibt, ab welchem Wert die verschiedenen Arten von Keywords in Verwendung treten und überhaupt für den Benutzer angezeigt werden.

4.3 Aufbau der Anwendung

Wie in Abbildung 4.3 ersichtlich, besteht die Anwendung Enrich aus zwei Hauptkomponenten: der Typo3-Extension selbst und einem Java-Servlet.

Als Extension für Typo3 wurde im CMS ein neues Inhaltselement namens „Article Enricher“ erstellt, das dem Redakteur beim Verfassen von angereicherten Artikeln zur Verfügung steht. Dabei entstand unter der Verwendung von HTML und JavaScript das bereits beschriebene, in Abbildung 4.1 ersichtliche Benutzer-Interface. Mit Klick auf „Find Related Keywords“ wird nun der vom Redakteur im Richtext-Editor verfasste Eintrag inklusive der Konfigurationsdaten (Abb. 4.2) an das Java-Servlet geschickt.

Das Java-Servlet enthält die wichtigste Funktionalität der Applikation: Es nimmt den Zugriff auf die APIs und die Verwaltung deren Antworten vor und liegt auf einem externen Server. Es wurde deswegen auf eine eigene Abstraktionsebene gehoben, um die Wiederverwendung für andere Anwendun-

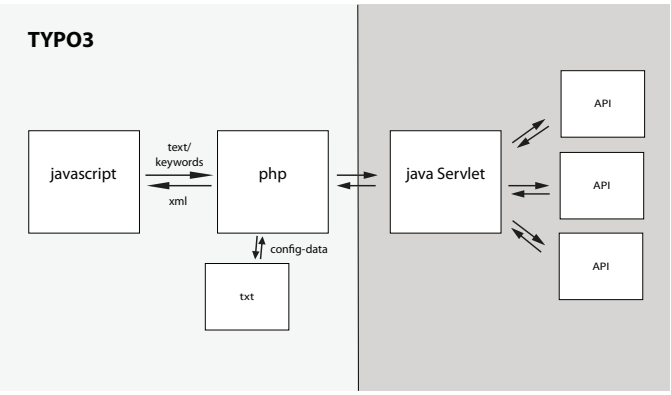

Abbildung 4.3: Aufbau der Anwendung Enrich.

gen zu ermöglichen. Zunächst interpretiert es die gelieferten Konfigurations-Daten und schickt den vom Redakteur verfassten Text an die gewünschten Semantic APIs weiter, welche passende Keywords zurückliefern. Diese werden vom Servlet in einem XML zusammengestellt und an die Typo3-Extension zurückgegeben. Die Extension ist dafür verantwortlich, die Keywords aus dem XML zu extrahieren und als Auswahloptionen für den Benutzer anzuzeigen.

Ähnlich ist der Ablauf beim Herbeiholen und Anzeigen von verwandten Medien: Mit Klick auf „Find related Media“ werden die vom Redakteur selektierten Keywords an das Java-Servlet weitergeleitet. Dieses übernimmt den Zugriff auf die Content-APIs und holt zu den Keywords passende Medien. Die Ergebnisse werden wiederum als XML zurückgeliefert und via JavaScript ins Typo3-Backend eingefügt.

Beim Zugriff mittels JavaScript auf einen anderen Server stößt man unwillkürlich auf das Problem der Same-Origin-Policy. Dies ist eine Sicherheitsmaßnahme und besagt, dass ein Skript nur auf die Eigenschaften und Methoden zugreifen kann, die vom gleichen Ursprung (same origin) stammen, wie das Skript selbst [50]. Als Lösung bot sich hier die Einrichtung eines php-Proxys ein, der die Aufrufe an den Webserver, sowie dessen Antworten weiterreicht. Weiters kümmert sich das php-Skript um die Einbindung der Konfigurationsdaten aus einer .txt-Datei.

4.3.1 Integration in Typo3

Typo3 ist ein freies Content-Management-System, das auf der Skriptsprache PHP basiert; als Datenbank wird meist MySQL eingesetzt. Mithilfe von Erweiterungen können zahlreiche zusätzliche Funktionen ins CMS eingefügt werden. Dazu gehören News- und Shopsysteme, Suchmaschinen, Galerien oder Diskussionsforen, welche auf typo3.org im Extension-Repository zur

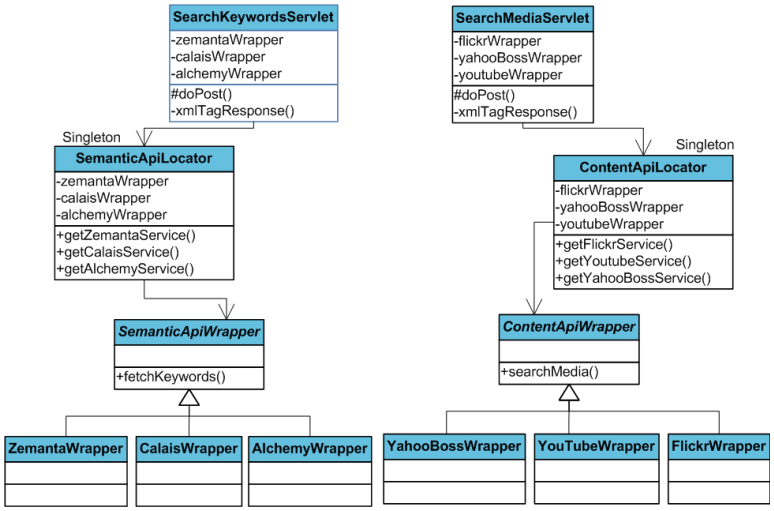

Abbildung 4.4: Klassendiagramm Java-Servlets.

Verfügung stehen. Typo3 bietet Entwicklern ein Framework an, um auch eigene Erweiterungen zu erstellen. Behilflich dabei ist die Extension Kickstarter, welche es erlaubt, sehr schnell die Datenbank mit eigenen Tabellen zu erweitern und ein auf PHP aufbauendes Basisgerüst für Extensions zu erstellen[4].

4.3.2 Aufbau der Java-Servlets

Wie bereits erwähnt, wurden der API-Zugriff und die Verwaltung der XML-Antworten auf eine eigene Abstraktionsebene gehoben, um die Wiederverwendung für andere Anwendungen zu ermöglichen. Zwei Java-Servlets sind dafür zuständig, Zugriffe auf die Semantic APIs bzw. Content-APIs vorzunehmen. Die gewünschten Ergebnisse der APIs werden in einem XML zusammengestellt und anschließend an den Client zurückgeliefert.

In Abbildung 4.4 ist der Aufbau der Servlet-Struktur erkennbar. Die Endpunkte bilden die zwei Servlets SearchKeywordsServlet und SearchMediaServlet. Die verschiedenen ServiceWrapper dienen dazu, die Zugriffe zu den einzelnen APIs herzustellen. Sie sind jeweils von der zugehörigen Basisklasse SemanticApiWrapper bzw. ContentApiWrapper abgeleitet.

Wie der Name bereits andeutet, ist das SearchKeywordsServlet dafür zuständig, nach relevanten Keywords zu suchen. Es schickt den vom Client gelieferten Text an die Semantic APIs Zemanta, OpenCalais und AlchemyAPI. Dazu erzeugt es das Singleton SemanticApiLocator, welches die einzelnen API-Wrapper als statische Variablen speichert und sich darum kümmert, dass von diesen jeweils nur eine Instanz erzeugt wird. Mittels der Methode

[4] http://typo3.com/Highlights.1629.0.html

fetchKeywords() wird der API-Aufruf gestartet. Die API-Antworten werden geparst und in einem XML zusammengestellt.

Analog dazu arbeitet das SearchMediaServlet, welches nach relevanten Medien sucht. Es schickt die vom Client erhaltenen Keywords an die Content-APIs Yahoo! BOSS, YouTube und Flickr und retourniert die entsprechenden Bilder, Artikel und Videos in einem XML.

4.4 Auswahl der Semantic APIs

Eine Hilfestellung für die Auswahl geeigneter APIs lieferte DiCiuccio in [16], einem Blogartikel, in dem verschiedene Entity Extraction APIs gegenübergestellt und bezüglich den Anforderungen in der ViewChange.org-Plattform (Kap. 3) evaluiert werden. Der Artikel präsentiert Zemanta als klaren Gewinner der durchgeführten Tests, weiters bieten OpenCalais und AlchemyAPI brauchbare Ergebnisse für Entitäten.

Auch für die Anwendung Enrich wurden schlussendlich diese drei APIs ausgewählt:

- Zemanta ist vor allem deswegen interessant, da sie auf das betreffende Anwendungsgebiet, dem Zuordnen von Keywords für Webseiten, spezialisiert ist. Für einen Text wird jeweils eine konstante Anzahl von Keywords zur Verfügung gestellt. Dabei handelt es sich auch um Keywords, die im Text selbst nicht auftauchen, aber eng mit dem Thema in Verbindung stehen.
- Auch basierend auf OpenCalais gibt es einige Anwendungen, die Texte mit Keywords versehen. Dabei werden die identifizierten Entitäten und Social Tags der API eingesetzt. OpenCalais wird sehr vielseitig in großen Medienplattformen eingesetzt und ist wohl die bekannteste Semantic API, was eine hohe Erwartung an deren Qualitäten mit sich bringt.
- AlchemyAPI bietet vor allem aufgrund der großen Sprachenunterstützung einen Anreiz. Während OpenCalais neben Englisch auch Französisch und Spanisch unterstützt, liefert AlchemyAPI zusätzlich Keywords zu deutschen Texten, deren Qualität in weiterer Folge untersucht wird.

4.5 Kombination der Keywords

Bei der Auswahl der Metadaten, welche die APIs zu Texten liefern, wurden folgende Arten von Keywords für die Anwendung in Betracht gezogen: Keywords und Links von Zemanta, Named Entities und Social Tags von OpenCalais, und Named Entities, Konzepte und Keywords von AlchemyAPI. In Abb. 4.5 werden alle Arten von Keywords in einer Übersicht für einen BBC-

Artikel[5] dargestellt. Fast alle wurden für die Verwendung in der Extension als geeignet befunden:

- **Zemanta-Keywords**: Die Keywords von Zemanta liefern ausgezeichnete Ergebnisse und wurden deshalb in die Auswahl mit aufgenommen. Je geringer der Confidence-Wert der Keywords, desto unsicherer ist ihre Verwendung.
- **Zemanta-Links**: Zemantas Markup ist eigentlich dazu da, den Lesern durch Links mehr Informationen zu einzelnen Konzepten und Themen im Text zu liefern. Dabei werden Themen des Textes identifiziert und markiert. Es entstand die Überlegung, diese Themen zusätzlich in die Liste der Keywords aufzunehmen. Dies wurde jedoch unterlassen, da Zemanta mit den eben beschriebenen Keywords bereits die besten Ergebnisse liefert.
- **OpenCalais-Named Entities**: OpenCalais extrahiert Entitäten, welche eigentlich auch den Inhalt eines Textes beschreiben und deshalb in die Liste der Keywords aufgenommen wurden. Ein Relevance-Wert der Keywords gibt an, wie wichtig ein Begriff zur Charakterisierung des Textes ist. Die Entitätstypen „Currency“ und „Position“ wurden aufgrund fehlender Bedeutung aus der Liste ausgeschlossen.
- **OpenCalais-Social Tags**: Mit den Social Tags von OpenCalais wurde versucht, nachzuahmen, mit welchen Stichwörtern eine Person einen Text versehen würde. Die Tags stellen Begriffe aus Wikipedia dar und wurden ebenfalls in die Auswahl der Keywords aufgenommen. Leider gibt es für den Stellenwert der Resultate nur eine grobe Abstufung (1 oder 2).
- **AlchemyAPI-Named Entities** und **AlchemyAPI-Concepts**: AlchemyAPIs Entitäten und Konzepte sind ähnlich den soeben beschriebenen Metadaten von OpenCalais. Beide liefern Relevance-Werte zwischen 0 und 1.
- **AlchemyAPI-Keywords**: AlchemyAPI liefert zu Texten eine beachtliche Reihe von Keywords. Diese sind jedoch aufgrund der schlechten Qualität nicht für eine sinnvolle Verschlagwortung von Artikeln geeignet.

Bei dieser Auswahl handelte es sich um eine bunte Mischung von Keywords unterschiedlicher Qualitäten, deren Kombination sich nicht als einfach erwies. Bei der Analyse der Texte ging eindeutig Zemanta als beste API hervor, mit guter Trefferquote und wenigen Fehlern. Ihr folgten OpenCalais und AlchemyAPI, welche vermehrt auch unpassende Vorschläge für Keywords lieferten.

Es entstand der Gedanke, solche Fehltreffer mit restriktiveren Vorgehensweisen zu beseitigen. Anbieten würde sich beispielsweise die Methode, dass

[5] http://www.bbc.co.uk/news/entertainment-arts-13088111

ein Keyword nur dann angezeigt wird, wenn dieses von mehreren APIs gefunden wird. Dieser Weg funktioniert zwar, führt aber auch dazu, dass gute Treffer, die nur von einer API aufgedeckt werden können, dann leider verschwinden.

Auch das Verhindern der Fehltreffer durch das Regulieren der numerischen Werte ist schwierig. Durch Setzen dieser Werte kann nie verhindert werden, dass ungünstige Resultate sichtbar bleiben, wenn diese mit sehr großer Wahrscheinlichkeit identifiziert wurden. Genauso werden auch Resultate, die eigentlich sehr gut sind, aber eben auch unsicher, wegfallen.

Letztendlich ermöglichen die Parameter dem Benutzer dennoch, anzugeben, welche Art von Keyword-Ergebnissen er bevorzugt. Höhere Werte liefern restriktivere aber tendenziell treffendere Keywords, niedrigere Werte ermöglichen eine größere Anzahl an Vorschlägen. Dies bedeutet aber auch eine größere Anstrengung, die Entitäten durchzuarbeiten. Zusätzlich können bestimmte Arten von Metadaten ganz außen vor gelassen werden. Ist der Benutzer beispielsweise nicht an Entitäten von OpenCalais und AlchemyAPI interessiert, setzt er die entsprechenden Relevance-Werte auf 1 und verhindert somit deren Erscheinen.

Außerdem wurde die Funktionalität geschaffen, dass bei auffallend sicherer Identifikation eines Keywords, dieses ohne Zutun des Benutzers von Vornherein markiert und in die Liste der bereits selektierten Keywords mitaufgenommen wird. Gearbeitet wurde dabei mit den Relevance/Confidence-Werten der Keywords der APIs, welche jeweils zwischen 0 und 1 liegen. Das Problem dabei ist, dass die Werte je API nicht gleich berechnet werden, geschweige denn die gleiche Bedeutung haben und auf unterschiedlichen Skalen liegen. Während Zemanta tendenziell niedrige Werte liefert, liefert OpenCalais mittlere und AlchemyAPI sehr hohe. Da eine Normalisierung der Werte zu aufwendig ist, wurde zur Berechnung der Sicherheit eines gefundenen Keywords der Median der vorhandenen Werte gewählt. Hätte beispielsweise OpenCalais ein Keyword mit dem Wert 0,3 identifiziert, Zemanta mit 0,15 und AlchemyAPI gar nicht (also mit dem Wert 0), wäre der resultierende Median 0,15. Die Anwendung Enrich markiert aber nur Keywords ab einem sehr hohen Wert von 0,6.

Finden gleich zwei von drei APIs ein Keyword nicht, stürzt der Median auf 0. Dies wäre eine Möglichkeit, die oben erwähnte Überlegung, einzelne Fehltreffer zu eliminieren, umzusetzen, indem diese einfach nicht angezeigt würden. Dieser Weg wurde jedoch nicht gewählt, da durch diese Methode auch gute Treffer verschwinden würden.

4.6 Evaluierung

Bei der entscheidenden Frage, wie groß nun wirklich der Mehrwert durch das Kombinieren von Semantic APIs ist, wurden schließlich Texte aus ver-

Zemanta-Keywords		OpenCalais-Entities	
Multi-sport event	0.18	Patrick Hickey	0.45
Deloitte	0.12	Swimming	0.35
Asian Games	0.03	athletics	0.35
Government	0.03	Europe	0.35
Multilateral	0.03	European Championships	0.36
European Athletic Association	0.02	gymnastics	0.24
Games	0.02		
International Olympic Committee	0.02		

Tabelle 4.1: Keywords für einen Artikel aus BBC: Das Unternehmen Deloitte führt eine Studie durch, um herauszufinden, ob sich ein Multi-Sport-Event in Europa auszahlen würde.

schiedenen Domänen und unterschiedlichen Sprachen getestet. Das Hauptaugenmerk wurde darauf gelegt, ob die Kombination der APIs bessere Ergebnisse liefert als Zemanta alleine, jene API, die explizit auf das Zuweisen von Stichwörtern spezialisiert ist und die besten Resultate liefert. Die für die Evaluierung verwendeten englischen Texte stammen aus verschiedenen Themenbereichen der BBC-Website[6].

Zemanta: Zemanta liefert in allen Kategorien gute Ergebnisse. Schwierigkeiten tauchen auf, wenn ein Text wenige ausdrucksstarke Bezeichnungen von Dingen enthält; aber auch dann ist das Ergebnis passabel. Da Zemanta immer eine konstante Anzahl von Keywords liefert, landet eine Großzahl der identifizierten Keywords unter 0.05, einer sehr unsicheren Zuordnung. Diese Keywords könnten also, wenn gewünscht, herausgefiltert werden. Die Ergebnisse eines schwierigen Textes finden sich in Tabelle 4.1.

OpenCalais: OpenCalais liefert mit seinen Entitäten zu eben erwähntem Text völlig andere Ergebnisse (Tabelle 4.1). Neben der Person „Patrick Hickey“, der für die Einrichtung eines Multi-Sport-Events plädiert, tauchen hier Entitäten der Typen „Continent“, „SportsGame“ und „SportsEvent“ auf.

Die Stärke von OpenCalais in Vergleich zu Zemanta ist das Identifizieren von Namen (hier zum Beispiel „Patrick Hickey“), welche nicht öffentlich bekannt sind, also zum Beispiel keinen Wikipedia-Eintrag haben.

Nicht so treffend für das Taggen dieses Textes sind jedoch die genannten Sportarten, die im Text zwar auftreten, aber für die Verschlagwortung dieses Textes nicht sehr aussagekräftig sind. Solche Begriffe treten in sehr vielen von OpenCalais analysierten Texten auf. Problematisch sind vor allem auch längere oder sehr namensreiche Texte, aus denen eine sehr lange Liste von

[6] http://www.bbc.co.uk/

Zemanta-Keywords		OpenCalais-Social Tags	
French Polynesia	0.26	Baleen whales	2
University of Queensland	0.23	Zoomusicology	2
Humpback whale	0.20	Animal communication	2
Current Biology	0.18	Animal intelligence	2
Australia	0.18	Humpback Whale	2
Whale song	0.15	Whale song	2
Whale	0.12	Whale	2
Eastern states of Australia	0.11	Environment	2
		Zoology	1
		Cetaceans	1
		Biology	1
		Entertainment_Culture	1

Tabelle 4.2: Keywords für einen Artikel über den Gesang von Buckelwalen.

identifizierten Entitäten hervorgeht. Da dies für diesen Anwendungsbereich nicht sinnvoll ist, sollte der Relevance-Wert der Entitäten auf mindestens 0,2 gesetzt werden. Dennoch können mit dieser Regulierung unpassende Begriffe nicht gänzlich herausgefiltert werden.

Ein weiteres Problem ist die Fehleranfälligkeit von OpenCalais. Insgesamt liefert die API zwar gute Resultate, trotzdem kommt es immer wieder zu falsch identifizierten Entitäten, die als Ergebnis angeboten werden (Näheres dazu in Kap. 6).

Als Fazit erweitern die Entitäten von OpenCalais zwar jene von Zemanta, es ist jedoch auch mit vielen unpassenden Vorschlägen zu rechnen.

Im Gegensatz zu den Entitäten sind die Social Tags von OpenCalais nützlicher. Durch sie wird es möglich, Texten auch allgemeine Überbegriffe hinzuzufügen, welche von Zemanta oft nicht präsentiert werden. Ein Beispiel für ein sehr gutes Ergebnis findet sich in Tabelle 4.2. Als Mindestwert für den Importance-Faktor empfiehlt sich der Wert 2.

AlchemyAPI: AlchemyAPIs Entitäten sind jenen von OpenCalais sehr ähnlich, wenn auch weniger umfangreich. Störend wirkt hier die oft fehlende Erkennung von Koreferenzen von Personen (siehe Kap. 6). Die Konzepte AlchemyAPIs sind vom Prinzip her den Social Tags von OpenCalais ähnlich, fügen dem Ergebnis aber wenig neue hilfreiche Begriffe hinzu. Beim Einsatz der Konzepte sollte der Mindest-Relevance Wert auf 0.75 festgelegt werden.

Andere Sprachen: Obwohl Zemanta nur Englisch unterstützt, kann die API auch bei der Analyse anderer Sprachen eingesetzt werden, denn bei Texten mit vielen bekannten Bezeichnungen (z.B. Namen und Teams aus der Formel 1) liefert Zemanta passable (aber immer englisch-sprachige) Er-

gebnisse. Fehlen solche Bezeichnungen in Texten, wird es aber so gut wie unmöglich für die API, sinnvolle Keywords zu finden.

OpenCalais unterstützt außer englischen auch französische und spanische Texte, liefert zu jenen aber nur eine begrenzte Anzahl von Entitäten und keine Social Tags. AlchemyAPI unterstützt noch einige Sprachen mehr, darunter auch Deutsch. Die beiden APIs erkennen hauptsächlich Namen von Personen, Orten und vereinzelt auch Unternehmen. Das Analysieren von anderssprachigen Texten ist aufgrund ihrer Komplexität sehr schwierig. Die Zuordnung von geeigneten Keywords funktioniert deshalb nur mangelhaft.

Domänen: Bezüglich trainierter Domänen wurden zwischen den APIs keine auffallenden Unterschiede festgestellt. Da alle APIs mitunter mit Themen der umfangreichen Enzyklopädie Wikipedia arbeiten, ein Portal das Themen aus allen Bereichen enthält, wurden auch Begriffe aus allen möglichen Sparten angeboten.

Zusammenfassend lässt sich sagen, dass der Mehrwert für automatische Taggingvorschläge durch das Kombinieren von APIs gering ist. Durch ein Kombinieren werden zwar zusätzliche Keywords für den Benutzer vorgeschlagen. OpenCalais' Stärke ist es, Eigennamen, welche nicht durch Plattformen wie Wikipedia oder ähnliches bekannt sind, identifizieren zu können. So kann beispielsweise der Name „Upper Austria University of Applied Sciences“ von OpenCalais und AlchemyAPI erkannt werden, von Zemanta jedoch nicht. Das Problem ist aber, dass die APIs OpenCalais und AlchemyAPI vermehrt auch unpassende Keywords hinzufügen. Das geschieht erstens, da diese APIs fehleranfälliger sind bzw. für den Text wenig-relevante Stichwörter liefern, welche auch mithilfe der Konfigurations-Parameter nicht herausgefiltert werden können. Zweitens kommt es durch die Kombination der APIs dazu, dass eigentlich gleiche Begriffe in unterschiedlicher Form auftauchen. Das betrifft beispielsweise die Schreibweise von Namen (Jurgen Melzer vs. Jürgen Melzer) oder auch die Bezeichnung von Unternehmen (Apple vs. Apple Inc.).

Aufgrund dessen ist eine fehlerfreie Kombination aller APIs unmöglich. Die beste Kombination ist noch erreichbar, wenn die Keywords von Zemanta mit den Social Tags von OpenCalais verbunden werden. Die SocialTags von OpenCalais stellen eigentlich Begriffe aus Wikipedia dar, sind darum oft allgemeiner und können den Ergebnissen von Zemanta durchaus sinnvolle Vorschläge hinzufügen. Inkonsistenzen der Tags werden zwar auftauchen, sie sind aber weniger zahlreich. Zusätzlich besteht die Möglichkeit, dem Ergebnis die Entitäten von OpenCalais und AlchemyAPI hinzuzufügen, um Begriffe wie „Upper Austria University of Applied Sciences“ identifizieren zu können. Hier wird aber empfohlen, die Mindest-Relevance-Werte sehr hoch anzusetzen, um Inkonsistenzen weitgehend zu vermeiden.

Die Analyse anderer Sprachen als Englisch ist noch nicht ausgereift und

funktioniert nur mangelhaft. Eine Kombination der APIs kann die Vorschläge für Tags zwar vermehren, führt jedoch erneut stark zu Inkonsistenzen.

Der größte Vorteil der Kombination mehrerer APIs ist die Gegebenheit, dass beim Ausfall einer API nicht auf Ergebnisse verzichtet werden muss, sondern auf eine der anderen zurückgegriffen werden kann.

4.7 Erweiterungsmöglichkeiten

Da die Anwendung Enrich aktuell einen Prototyp darstellt, der im Kernbereich auf eine sinnvolle Kombination von Keywords abzielt, wurden andere Bereiche der Applikation noch außen vorgelassen. Nachfolgend sollen darum einige Erweiterungsmöglichkeiten der Anwendung dargestellt werden.

Zu allererst soll hier die Ausbaufähigkeit der Content-APIs angesprochen werden. Die bisher verwendeten APIs Yahoo BOSS!, Flickr und YouTube bilden aktuell eine Grundlage und stellen nur eine kleine Auswahl an Diensten dar, welche verwandte Medien zu Texten zur Verfügung stellen. Eine Reihe von weiteren Diensten existieren im Wide World Web, welche angewendet werden können, um Redakteuren verwandte Medien zur Verfügung zu stellen. Einige Content-APIs für Artikel und Videos evaluiert beispielsweise [16].

Ein noch nicht beachteter Punkt sind weiters die Lizenzrechte der zur Verfügung gestellten Bilder. Im Moment werden zwar die Quellen von Bildern jeweils unterhalb des im Text eingefügten Bildes angegeben, das rechtfertigt aber noch nicht deren einwandfreie Nutzung. Redakteure sind demnach zum aktuellen Zeitpunkt selbst für die Nutzung von Bildern verantwortlich. In der Programmierung müsste man in weiterer Folge darauf achten, den Benutzern ausschließlich lizenzfreies Material zur Verfügung zu stellen.

Die Speicherung von Keywords zu einem Artikel in Typo3 werden derzeit der Einfachheit halber in der Datenbank durch einen durch Kommata getrennten String vorgenommen. Nebenbei bemerkt werden in Typo3 auch die „hauseigenen“ Keywords zur jeweiligen Seite auf die gleiche Art und Weise gespeichert. Als zusätzliche Funktionalität für Enrich wäre es beispielsweise vorstellbar, bei der Suche nach verwandten Artikeln darüber hinaus die bereits verfassten Artikel der eigenen Website zu berücksichtigen. Dabei wäre eine Speicherung der Keywords in einer eigenen Tabelle vorteilhaft. Weiters wäre es auch einfacher, weitere Extensions für Typo3, welche die zu den Content-Elementen gespeicherten Keywords ausnützen, zu erstellen. Möglichkeiten dazu sind beispielsweise TagClouds, keywordbasierte Indexseiten oder Anwendungen zur Suchmaschinenoptimierung. Natürlich besteht auch die Option, bereits im Typo3-Extension Repository[7] existierende Erweiterungen zu verwenden, anzupassen oder zu erweitern. Dafür anbieten würden sich u. a. A-Z Keyword List, SEO dynamic tag oder Tagclouds for all tables (TIMTAB).

[7] http://typo3.org/extensions/repository/

Da die Logik des Kombinierens der Keywords auf eine eigene Abstraktionsebene gehoben wurde, um auch anderen die Nutzung zu ermöglichen, ist auch der Ausbau der XML-Antworten vorstellbar. Anhand von Blogs und Artikeln, welche sich mit der Auswahl von geeigneten Semantic APIs beschäftigen, ist festzustellen, dass eine Verlinkung in die Linking Open Data Cloud ein wichtiges Kriterium darstellt. Aufgrund dessen besteht die Option, die vorhandenen Links von Entitäten und Konzepten ebenfalls zu sammeln und für die Nutzung zur Verfügung zu stellen.

Abbildung 4.5: Artikel über Terry Pratchett mit möglichen Keywords aus den APIs OpenCalais, Zemanta und AlchemyAPI (mehr Informationen liefern die RDF-Antworten).

Kapitel 5

Natural Language Processing

Von Semantic API-Anbietern werden sehr ausführlich Funktionen, Einsatzweisen und Vorteile ihrer Textanalysen präsentiert. Wie die Dienste im Hintergrund aber arbeiten und welche Methoden sie verwenden, ist schwer herauszufinden. Um genauer zu verstehen, was hinter dieser „Black Box“ eigentlich steckt, wird im folgenden Kapitel das Gebiet des Natural Language Processing (NLP) näher beleuchtet.

Nach einer Definition des Begriffs werden Teilbereiche und Ansätze erläutert. Genauer eingegangen wird im Anschluss auf die Eigennamenerkennung, welche einen wichtigen Vorverarbeitungsschritt für viele NLP-Anwendungen darstellt und von den meisten Semantic APIs angeboten wird. Abschließend folgt ein Überblick über Einsatzweisen von Wikipedia.

5.1 Definition und Felder

Der Begriff NLP wird verwendet, um Funktionen von Software- oder Hardwarekomponenten in Computersystemen zu beschreiben, welche gesprochene oder geschriebene Sprache analysieren bzw. synthetisieren. Das Beiwort „natural“ soll dabei die menschliche Sprache von formellen Sprachen wie mathematische Notationen oder Programmiersprachen unterscheiden [28].

Abgedeckt werden von NLP viele Anwendungsgebiete, deren Themenbereiche und Methoden sich stark überlappen. Einige der Gebiete werden hier kurz vorgestellt, um einen ersten Einblick in die Tätigkeitsbereiche von NLP zu vermitteln [28]:

- Beim **Document Retrieval**, der wohl wichtigsten NLP-Anwendung im World Wide Web, geht es darum, Dokumente aufgrund von Benutzeranfragen zu finden. Der allgemeinere Ausdruck **Information Retrieval** inkludiert zusätzlich auch die Abfrage von Bildern oder Musik.
- **Document Classification** ist eine Aufgabe, bei der Dokumente aufgrund deren Inhalts bestimmten Klassen zugewiesen werden.
- Das Ziel der **Informationsextraktion** ist es, nicht Dokumente zu fin-

den, sondern bestimmte Informationen in diesen. Beispielsweise könnte man aus einer Sammlung von Nachrichtenartikeln über Firmenübernahmen extrahieren, wer welche Firma gekauft hat.

- Die so gelieferten Informationen liefern den Ausgangspunkt für die Aufgabe der **Textzusammenfassung**.
- **Text Mining** ist ein Gebiet, das mehrere Teilaufgaben beinhaltet. Eine Applikation könnte beispielweise Dokumente eines Newsfeeds aufgrund deren Inhalts auswählen, diese in Kategorien gliedern und aus den gewünschten Dokumenten sachdienliche Informationsstrukturen extrahieren.
- Auch **Sentimentanalyse** ist ein Teilbereich von NLP. Es geht dabei um das Extrahieren von Meinungen, Gefühlen und Emotionen aus Texten [32].

5.2 Ansätze

Generell gibt es zwei Arten von Herangehensweisen, wie NLP-Anwendungen arbeiten. Im Verlauf des weiteren Kapitels werden diese immer wieder auftreten: der Knowledge Engineering Approach und der Automatic Training Approach [1]:

Den Knowledge Engineering Approach kennzeichnet die Entwicklung von handgeschriebenen Regeln, welche die gesuchten Informationen, wie z.B. Eigennamen, aus dem Text extrahieren. Dazu werden Experten benötigt, die sich in der entsprechenden Domäne auskennen und auch über die entsprechenden Formalismen, wie Regeln verfasst werden, Bescheid wissen. Die Fähigkeit bzw. Begabung der Personen spielt eine große Rolle, wie performant die Ergebnisse ausfallen; die Erstellung der Regeln beansprucht aber auch sehr viel Zeit. Ein großer Nachteil dieses Ansatzes ist, dass die erstellten Regeln meist nur für Texte einer bestimmten Domäne gelten. Beim Wechseln dieser Domäne muss das gesamte Regelsystem angepasst bzw. neu erstellt werden [1].

Daher steht die Entwicklung des moderneren Automatic Training Approach im Fokus der aktuellen Forschung. Hier werden mithilfe von annotierten Trainingsdokumenten Regeln automatisch abgeleitet. Dadurch ist zwar die manuelle Erstellung von Regeln nicht mehr erforderlich, jedoch wird üblicherweise eine große Menge an Trainingsdaten der entsprechenden Domäne benötigt. Diese Daten stehen jedoch nicht immer zur Verfügung und deren Erstellung ist sehr aufwendig [1].

Die beschriebenen Ansätze werden außerdem miteinander kombiniert; neuere Methoden versuchen zusätzlich, auch Regeln ohne annotiertes Trainingsmaterial abzuleiten [28].

5.3 Linguistische Methoden

Für die Verarbeitung natürlicher Sprache existiert eine Reihe von verschiedenen Aufgaben. So müssen Texte in einzelne Sätze zerlegt werden und Sätze in Wörter; anschließend werden Wortarten bestimmt und Sätze geparst, d.h. deren Phrasen- und Satzstruktur bestimmt [28].

Von NLP-Anwendungen werden aber nicht immer alle diese Aufgaben durchgeführt. So führen zum Beispiel alle Suchmaschinen den Schritt der Tokenscannung durch, aber nur einige die Bestimmung der Wortart [28].

5.3.1 Satztrennung

Der erste Schritt der Verarbeitung ist die Satztrennung. Die Trennung eines Textes in einzelne Sätze ist nicht so einfach wie man denken könnte, denn Punkte im Text stellen nicht unbedingt das Ende eines Satzes dar. So tauchen Punkte auch in Abkürzungen oder Zahlen auf. Weiters werden Sätze mit großgeschriebenen Buchstaben begonnen, aber nicht jedes dieser großgeschriebenen Wörter stellt den Anfang eines neuen Satzes dar. Aufgrund dessen arbeiten Satztrenner meist mit Regular Expressions und Ausnahmeregeln. Ein Beispiel so einer Ausnahme wäre:

> Punkte, denen ein Leerzeichen und dann ein Großbuchstabe folgt, denen aber ein Titel (Mr., Mrs., Dr., etc.) vorhergeht, kennzeichnen kein Satzende.

Andere Satztrenner verwenden empirische Methoden und werden mit händisch segmentierten Textdaten trainiert [28].

5.3.2 Tokenscannung

Der nächste Schritt ist die Tokenscannung. Sie teilt einen Datenstrom von Textzeichen in aussagekräftige Einheiten, die Tokens genannt werden. Token bestehen dabei aus einer Sequenz von Textzeichen und werden meistens durch Leerzeichen getrennt. Bei der Durchführung der Tokenscannung müssen jedoch auch Satzzeichen wie Punkt, Komma und Bindestrich berücksichtigt werden. Es wird also festgelegt, ob es sich bei Zeichen wie „data-base“ oder „$1,500“ um jeweils eines oder zwei Tokens handelt. Weitere Schwierigkeiten bilden zusammengesetzte Wörter, wie „Lebensversicherungsgesellschaft“, welche häufig im Deutschen vorkommen oder durch Leerzeichen getrennte Wörter, welche eigentlich zusammen gehören, wie „pomme de terre“ (frz. für Kartoffel). Tokenscanner arbeiten meist mit Regeldefinitionen, endlichen Automaten, statistischen Modellen und Lexikons [28].

5.3.3 Stemming und Lemmatisierung

Um Dinge auszudrücken, werden in Texten üblicherweise verschiedene Formen von Wörtern benutzt. So kann das Wort „organize“ aus grammatikalischen Gründen auch als „organizes“ oder „organizing“ auftauchen. Weiters gibt es Wörter mit gleichen Stammfamilien (z.B. „democracy“, „democratic“, „democratization“). Die Aufgabe von Stemming und Lemmatisierung ist es, solche Wörter auf eine Stammform zurückzuführen [33].

Stemming bezeichnet generell einen heuristischen Prozess, welcher grob Wörter abschneidet und mögliche Affixe ('un-', 'dis-','-ing', '-ness', etc.) von Wörtern entfernt. Der resultierende Term ist also nicht unbedingt ein korrektes Wort. Lemmatisierung hingegen führt Wörter auf deren korrekte Grundform zurück, so wie sie auch im Wörterbuch zu finden ist [33].

Der bekannteste Algorithmus für ein Stemming der englischen Sprache ist der Porter Stemmer [51], welcher auf sequentiell durchgeführten Phasen von Wortreduktionen basiert. Jede Phase besteht aus einer Regelgruppe, wobei jene Regel, welche das längste Suffix entfernt, auf das Wort angewendet wird. Folgende Konvention wird in der ersten Regelgruppe angewendet:

Regel		**Beispiel**	
SSES	$\rightarrow$ SS	caresses	$\rightarrow$ caress
IES	$\rightarrow$ I	ponies	$\rightarrow$ poni
SS	$\rightarrow$ SS	caress	$\rightarrow$ caress
S	$\rightarrow$	cats	$\rightarrow$ cat

Auch wird beim Porter Stemmer das Ausmaß m eines Wortes berücksichtigt. Dabei wird grob die Anzahl der Silben eines Wortes bestimmt, um festzulegen, ob ein Wort zum Entfernen eines Suffixes lang genug ist. So würde die Regel

$$(m > 1) \text{ EMENT} \rightarrow$$

das Wort „replacement“ zu „replac“ verkürzen, aber nicht „cement“ zu „c“ [51].

5.3.4 Bestimmung der Wortart

Die Bestimmung von Wortarten wird im Englischen als part-of-speech tagging oder POS-Tagging bezeichnet. Jedes im Text vorkommende Wort wird dabei mit einem Bezeichner versehen, der bestimmt, ob es sich um ein Nomen, Verb, Adjektiv oder etwas anderes handelt.

Auch die Wortartbestimmung arbeitet mit unterschiedlichen Ansätzen [28]: Die regelbasierten Techniken konzentrieren sich auf das manuelle Verfassen von Regeln, die den Kontext des Wortes oder morphologische Aspekte berücksichtigen. Zwei Beispielregeln für die englische Sprache sind:

Steht ein unbekannter Begriff zwischen einem Artikel und einem Nomen, soll er als Adjektiv bezeichnet werden.

Endet ein unbekanntes Wort mit „-ing“ und folgt es einem Verb, soll es als Verb bezeichnet werden.

Die stochastischen Ansätze arbeiten mit manuell annotierten Trainingsdaten. Aufgrund von Informationen oder Wahrscheinlichkeiten über Häufigkeiten werden Wörtern entsprechende Tags zugeordnet [28].

5.3.5 Eigennamen- und Nominalphrasenerkennung

Bei der Eigennamenerkennung geht es um das Identifizieren von Namen von Personen, Unternehmen, Orten, etc. in Texten. In Kapitel 5.4 wird auf dieses Thema näher eingegangen.

Bei der Erkennung von Nominalphrasen (NP) hingegen geht es um das Identifizieren von Gruppen zusammengehöriger Wörter, deren Kopf ein Nomen ist (Abb. 5.1).

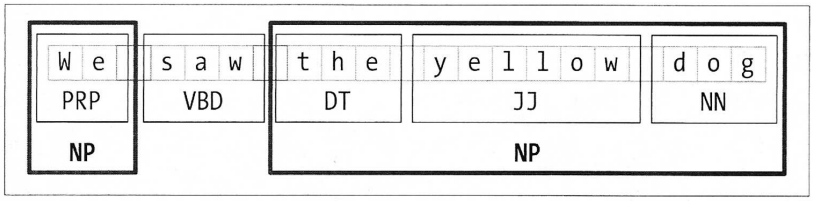

Abbildung 5.1: Basierend auf den Wortartnotierungen können Nominalphrasen identifiziert werden [7].

Wiederum gibt es zur Extraktion sowohl symbolische als auch statistische Ansätze. Dabei wird beim symbolischen Ansatz mit Regeln gearbeitet. Beispielsweise starten viele Nominalphrasen mit einem Artikel und enden vor einem Verb [28].

5.3.6 Parsing

Parsing geschieht mithilfe von Grammatiken, also Regeln, welche besagen, welche Wortart-Kombinationen korrekte Phrasen und Satzstrukturen bilden. Meist werden aber keine vollständigen syntaktischen Analysen durchgeführt, sondern flache, fragmentarische Analysen. Das bedeutet, komplizierte Strukturen in Sätzen wie Präpositionalphrasen, werden außen vor gelassen [28].

Die Erstellung von Grammatikregeln ist sehr arbeitsintensiv. Obwohl bereits viele generelle Grammatikregeln verfasst wurden, wird man nie alle Konstrukte, die in englischen Texten vorkommen, abdecken können. Genauso kann kein Lexikon alle in Texten auftauchenden Wörter enthalten. Statis-

tische Methoden können beim Lösen solcher Probleme helfen. So kann das gültige Wort „res judicata“, als ein Bigramm (eine Wortefolge von zwei Wörtern) identifiziert werden, obwohl es nicht im Lexikon zu finden ist. Denn sehr oft taucht die Wortfolge in bestimmten Dokumenten, etwa in Sammlungen von Gerichtsfällen, auf [28].

Weiters kann mit annotierten Textcorpora gearbeitet werden, mithilfe derer grobe syntaktische Strukturen von Sätzen abgeleitet werden können [28].

Die Annotation von syntaktische Strukturen erfolgt durch Klammern; die Strukturen können auch in Bäumen dargestellt werden (Abb. 5.2).

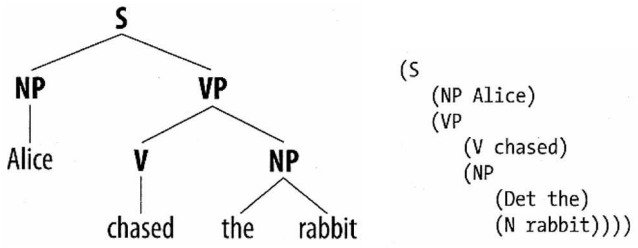

Abbildung 5.2: Syntaktische Strukturen als Baum und in Textform [7].

5.4 Eigennamenerkennung

An dieser Stelle wird auf das Thema der Eigennamenerkennung bzw. Named Entity Recognition (NER) etwas genauer eingegangen. Alle vorgestellten Semantic APIs bieten die Extraktion unterschiedlicher Eigennamen aus Texten an; weiters ist die Eigennamenerkennung eine wichtige Basis für viele weitere NLP-Anwendungen.

NER ist eine Teilaufgabe aus dem Gebiet der Informationsextraktion. Deren Ziel ist es, Eigennamen (Named Entities, NE) aus Texten zu identifzieren, wie etwa Personen, Orte, Unternehmen, Organisationen, usw. Dies kann auf zwei Teilaufgaben heruntergebrochen werden: die Begrenzung einer Entität zu finden und deren Typ zu bestimmen [7]. Es gibt eine sehr große Anzahl an Publikationen zum Thema NER, was daran liegt, dass NER auch einen wichtigen Vorverarbeitungsschritt für viele andere Anwendungen wie Frage-Antwort-Systeme, Textzusammenfassungen oder Maschinenübersetzungen darstellt. Weiters wurden wettbewerbsmäßig viele Evaluationskampagnen durchgeführt, welche die Entwicklung von NER-Systemen vorantrieben [54]. Eine davon war die sechste Message Understanding Conference (MUC-6) [23], bei der erstmalig eine eigenständige NER-Evaluation durchgeführt wurde.

Gängig bei der Evaluierung von NER-Systemen sind die Maße Präzisi-

on und Vollständigkeit (engl. precision and recall), welche auch in anderen Bereichen, wie z.B. Information Retrieval, angewendet werden [47]:

- Die Präzision P gibt den Anteil der korrekt gewonnenen Entitäten im Vergleich zu den insgesamt gefundenen Entitäten an. Eine hohe Präzision bedeutet also, dass fast alle gefundenen Entitäten relevant sind.
- Die Vollständigkeit V hingegen bezeichnet den Anteil der korrekt gewonnenen Entitäten im Vergleich zu den insgesamt gewinnbaren Entitäten. Eine hohe Vollständigkeit bedeutet somit, dass fast alle relevanten Entitäten gefunden werden konnten.

Als zusammenfassendes Maß wurde das harmonische Mittel der beiden Werte, das F-Maß,

$$F = \frac{(\beta^2 + 1) \cdot P \cdot V}{\beta^2 P + V}, \tag{5.1}$$

festgelegt. In der Regel wird dabei β=1 gesetzt, was eine gleiche Gewichtung der Werte P und V bewirkt [47].

5.4.1 Schwierigkeiten in NER

Eigennamen beginnen im Englischen meistens mit Großbuchstaben. So deuten die Wörter „John Smith“, „Thomson Corporation“, oder „Los Angeles“ auf Eigennamen hin. Leider garantieren die Großbuchstaben jedoch nicht unbedingt das Vorhandensein eines Namens, denn es werden z.B. auch Wörter am Satzanfang groß geschrieben. Man könnte meinen, die Benutzung von Namenslisten könnte helfen, dieses Problem zu lösen. Tatsächlich birgt die Verwendung von Listen aber neue Schwierigkeiten, denn keine existierende Liste ist vollständig, da von Tag zu Tag neue Namen für Personen und Unternehmen entstehen. Eine weiteres Problem bildet die Tatsache, dass viele Namen mehrdeutig sind. So kann eine Liste nicht unbedingt helfen, zu entscheiden, ob „Christian Dior“ ein Name oder eine Firma darstellt. Genauso könnte „The White house“ je nach Kontext, in dem es benutzt wird, einen Ort aber auch eine Organisation darstellen. Bei dem Wort „June“ muss unterschieden werden, wann es sich um einen Vornamen und wann um einen Monatsnamen handelt [7, 28].

Je nach Domäne sind meist auch andere Arten von Entitäten von Interesse wie Datumsangaben, Geldbeträge, Prozentzahlen, etc. Eine Herausforderung sind außerdem Namen, die aus mehreren Wörtern bestehen wie „Stanford University“, oder Namen, die andere Namen beinhalten, wie „Cecil H. Green Library“. Andere Namen wiederum sind eigentlich Phrasen und beinhalten auch Kleinbuchstaben („The Phantom of the Opera“). Es müssen also jeweils Anfang und Ende all dieser Eigennamen identifiziert werden [7, 28].

Im Folgenden wird nun auf die Techniken, wie Eigennamen extrahiert werden, eingegangen.

5.4.2 Regelbasierte Ansätze

Die regelbasierten Ansätze basieren auf dem Verfassen von Regeln, welche helfen, Eigennamen aus Texten zu identifizieren. So können mithilfe von grammatischen, syntaktischen und orthographischen Mustern Regeln abgeleitet werden. Ein Wortmuster wie „<Word> <Word>, Inc.“ würde beispielsweise darauf hindeuten, dass es sich beim entsprechenden Wort um ein Unternehmen handelt. Bei den Message Understanding Conferences MUC-6 und MUC-7 war eine Teilaufgabe eben diese Identifikation von Eigennamen, bei der auch viele der Teilnehmer mit regelbasierten Systemen arbeiteten.

Verbreitet unter diesen ist vor allem der Verzicht auf eine ausführliche Analyse der syntaktischen Satzstruktur. Anstelle einer tiefen und breiten Analyse mit kontextfreien Grammatiken (CFG), ist vielmehr eine flache Analyse lokaler Phänomene, meist mit endlichen Automaten, gängig [54]. Drei Nachteile von CFGs stellt [22] dar: Die Systeme sind zu langsam, um sie in passabler Zeit zu entwickeln und zu testen. Weiters führen globale Parsingergebnisse zu Fehlern auf lokaler Ebene, welche in einer lokalen Analyse nicht vorgekommen wären. Außerdem ist die Erweiterung einer komplexen Grammatik an eine neue Domäne schwierig. Im Vergleich dazu ist die Bearbeitung lokaler Gegebenheiten weniger schwierig, sie setzt aber eine Vorgehensweise voraus, die berücksichtigt, dass unterschiedliche Reihenfolgen von Regelanwendungen zu unterschiedlichen Resultaten führen können [54].

Das beste NER-System der MUC-7 basiert auf einem (hauptsächlich) regelbasierten Ansatz und wird hier vorgestellt. Es stützt sich auf Methoden wie Listen, Regeln und Wahrscheinlichkeitstechniken, die in einer bestimmten Abfolge von Schritten durchgeführt werden [42]:

1. Der erste Schritt der Applikation besteht aus sehr sicheren Regeln, die sich stark auf den Kontext im Text beziehen. So würden Bezeichner wie „Mr.“, „Dr.“, „Sen.“ auf Personen hindeuten und Bezeichner wie „Ltd.“ oder „Inc.“ auf Unternehmen. Das System benützt außerdem Namenslisten, kontrolliert jedoch zu diesem Zeitpunkt bei der Klassifizierung, ob der Kontext eines potentiellen Eigennamens die Vorschläge der Liste unterstützt. So würde der Ort „Washington“ nur als Ort markiert werden, wenn er auch in passendem Kontext auftaucht, wie z.B „in the Washington Area“.
2. Als Nächstes werden alle identifizierten Eigennamen im Dokument gesammelt und Halbordnungen der zusammengesetzten Wörter erstellt. Wurde zum Beispiel im ersten Schritt die Organisation „Lockheed Martin Production“ identifiziert, so können nun alle Instanzen von „Lockheed Martin Production“, „Lockheed Martin“, „Lockheed Production“, „Martin Production“, „Lockheed“ und „Martin“ als mögliche Organi-

sationen gekennzeichnet werden. Mithilfe eines statistischen Modells (Maximum Entropy, siehe Kap. 5.4.3) wird anschließend versucht, einige dieser Vorschläge zu bestätigen.

3. Nachdem diese Schritte erledigt wurden, erfolgt erneut das Anwenden von Regeln, diesmal aber ohne kontextuelle Bedingungen. So würden Ortsnamen aus Listen, die im Text gefunden werden, gekennzeichnet, auch wenn kein entsprechender Kontext vorhanden ist.

4. Als Nächstes werden erneut jene Wörter, welche Teile von bereits identifizierten Eigennamen darstellen, behandelt (siehe 2.).

5. Zum Schluss erfolgt die Kennzeichnung von Eigennamen im Dokumententitel, welche in Großbuchstaben vorliegen. So kann im Titel „MURDOCH SATELLITE EXPLODES ON TAKE-OFF“ „Murdoch“ als Person identifiziert werden, da dieser als „Rupert Murdoch“ bereits im Text gefunden wurde.

Mikheev et al. greifen in ihrem Ansatz stark auf die Prinzipien der internen und externen Evidenz [36] zurück. Während die interne Evidenz die Struktur des Wortes bzw. die Namensbestandteile berücksichtigt (z.B. „Ltd.“, „G.m.b.H.“), wird bei der externen Evidenz der Kontext eines Eigennamens einbezogen. So können mehrdeutige Einträge in Listen wie „Washington“, ein Wort, das sowohl einen Ort als auch eine Person darstellen kann, mithilfe des Kontextes (z.B. „in the Washington area“) disambiguiert, also gelöst werden.

Außerdem befassen sich die Entwickler im Besonderen mit dem Problem von großgeschriebenen Wörtern an mehrdeutigen Positionen. So werden Wörter am Satzbeginn, nach einem Doppelpunkt oder Anführungszeichen sowie Wörter nach Anführungszeichen großgeschrieben, deuten aber nicht unbedingt auf das Vorhandensein eines Eigennamens hin. Es entstand die Sequence Strategy, eine Strategie, die zuerst Wörter in eindeutigen Positionen identifiziert, und erst anschließend ähnliche Namen in mehrdeutigen Positionen. Wird beispielsweise die Phrase „Rocket Systems Development Co.“ in der Mitte eines Satzes eindeutig erkannt, so kann in einem nächsten Schritt jede Teilphrase des Begriffs (z.B. „Rocket Co.“, „Rocket Systems“) auch in mehrdeutigen Positionen markiert werden. Eine Spanne von großgeschriebenen Wörtern darf auch bis zu drei kleingeschriebene Wörter enthalten, um Phrasen wie „The Phantom of the Opera“ identifizieren zu können. Generierte Teilphrasen müssen jedoch immer mit großgeschriebenen Wörtern beginnen und enden [41].

Im Rahmen der MUC-7 lieferte das System ein F-Maß von 93% und erreichte somit den höchsten Wert der teilnehmenden NER-Systeme. Im Vergleich zu anderen erreichte es besonders gute Leistungen bei der Erkennung von Organisationen [42].

5.4.3 Ansätze des Maschinellen Lernens

Während sich frühe Studien hauptsächlich mit manuell erstellten Regeln auseinandersetzten, liegt der Schwerpunkt der aktuellen Forschung bei Ansätzen des Maschinellen Lernens. Ein alternativer Ansatz zu NER sind also Programme, welche mit Algorithmen des Maschinenlernens und statistischen Modellen selbst lernen, wie Eigennamen identifizert werden können. Dazu gibt es zwei Methoden: überwachtes Lernen und halb- bzw. unüberwachtes Lernen [46].

Beim überwachten Lernen werden mithilfe von annotierten Trainingsdokumenten Merkmale von positiven und negativen Eigennamen-Beispielen identifiziert und so Regeln abgeleitet. Einen großen Nachteil hierbei stellt jedoch die große Menge an benötigten annotierten Textcorpora dar. Das Nichtvorhandensein solcher Ressourcen und der große Aufwand, sie zu erstellen, führte zu Methoden des halb- und unüberwachten Lernens [46].

Folgend wird zuerst auf Methoden des überwachten Lernens eingegangen, anschließend auf solche des unüberwachten.

Hidden Markov Modelle

Ein beliebter Ansatz sind Hidden Markov Modelle (HMM), die mit dem Nymble-System [6] ihren Beginn im NER-Bereich nahmen:

Eine Art, sich NER vorzustellen, ist, dass in einem Text ursprünglich alle Eigennamen mit Tags gekennzeichnet worden waren, dieser dann aber durch einen gestörten Kommunikationskanal gelangte und somit alle Tags gelöscht wurden. Die Aufgabe ist es nun, ein Programm zu erstellen, welches den Originalprozess, der die mit Namensklassen annotierten Wörter generierte, modelliert. Es muss also für jedes Wort im Text entschieden werden, ob es sich dabei um einen Namen handelt oder nicht, bzw. um welche Art von Namen [6].

Behilflich dabei ist das HMM, ein stochastisches Modell, welches benutzt werden kann, um eine Sequenz von versteckten Zuständen anhand von Beobachtungen zu bestimmen. Die Beobachtungen sind dabei die Wörter des vorliegenden Textes, die Zustände stellen die Namensklassen dar. Durch die Beobachtungen kann nun die wahrscheinlichste Abfolge der Namensklassen generiert werden [54].

Die Entwickler des Nymble-Systems beziehen auch Oberflächenmerkmale von Wörtern mit ein, um zu bestimmen, um welche Art von Wörtern es sich handelt. Es handelt sich dabei um 14 Kategorien, die in Abb. 5.3 dargestellt sind. Jeweils eine diese Kategorien wird einem Wort zugeteilt. In Verbindung mit Informationen über Wortpositionen und Wortnachbarn wird die wahrscheinlichste Namensklasse berechnet [6].

Die Generierung der Wörter und der Namensklassen erfolgt in drei Schritten [6]:

Word feature	Example text	Explanation
twoDigitNum	90	Two-digit year
fourDigitNum	1990	Four-digit year
containsDigitAndAlpha	A8-67	Product code
containsDigitAndDash	09-96	Date
containsDigitAndSlash	11/9/98	Date
containsDigitAndComma	1,000	Amount
containsDigitAndPeriod	1.00	Amount
otherNum	12345	Any other number
allCaps	BBN	Organization
capPeriod	P.	Personal name initial
firstWord	The	Capitalized word that is the first word in a sentence
initCap	Sally	Capitalized word in midsentence
lowercase	tree	Uncapitalized word
other	.net	Punctuation, or any other word not covered above

Abbildung 5.3: Oberflächenmerkmale des Systems Nymble [28].

- Als Erstes wird eine Namensklasse NC_0 ausgewählt, welche abhängig vom vorhergehenden Wort NC_{-1} und der vorherigen Wortklasse w_{-1} ist:
$$P(NC_0|NC_{-1}, w_{-1}). \tag{5.2}$$
- Dann wird das erste Wort in dieser Namensklasse generiert, welches von der aktuellen und der vorhergehenden Namensklasse abhängt. $\langle w_0, f_0\rangle$ steht hier für das aktuelle Wort-Merkmal-Paar:
$$P(NC_0|NC_{-1}, w_{-1}) \cdot P(\langle w_0, f_0\rangle \,|NC_0, NC_{-1}). \tag{5.3}$$
- Zuletzt werden alle folgenden Wörter in der aktuellen Namensklasse generiert, wobei jedes Wort vom vorhergehenden Wort und der aktuellen Namensklasse abhängt:
$$P(\langle w_0, f_0\rangle \,|\, \langle w_{-1}, f_{-1}\rangle \,, NC_0). \tag{5.4}$$

Die Berechnung der benötigten Wahrscheinlichkeiten erfolgt durch Zählen im annotierten Trainingscorpus. Die drei Schritte werden so lange wiederholt, bis alle beobachteten Wortsequenzen generiert wurden. Mithilfe des Viterbi Algorithmus ist es möglich, den gesamten Raum der möglichen Namensklassenzuweisungen zu durchsuchen und die wahrscheinlichste Folge aufzudecken [6].

Nun ist es aber so, dass in den Trainingsmodellen meist nicht alle Wörter, die in späteren Texten benötigt werden, auftreten. Beim Auftauchen eines unbekannten Wortes oder eines unbekannten Bigramms würde dann aufgrund der multiplikationsbasierten Berechnung die Wahrscheinlichkeit einer Sequenz auf 0 fallen. Um dies zu verhindern, kommen in solchen Fällen schwächere Modelle, nämlich Back-Off Modelle zur Anwendung. Diese beziehen weniger Informationen in die Berechnung mit ein, beispielsweise nur die Oberflächenmerkmale des Wortes [6].

Markov Modelle gehören zum generativen Ansatz, bei dem die Wahrscheinlichkeiten des gemeinsamen Auftretens von Beobachtung und Zustand, $P(x, y)$, berechnet werden. Schwierigkeiten entstehen jedoch, wenn die Beobachtung x kein atomares Merkmal darstellt, sondern einen komplexen Merkmalsvektor. Dadurch wird das Generieren des gemeinsamen Auftretens von $P(x, y)$ sehr schwierig, da die gemeinsame Wahrscheinlichkeit des Auftretens von y für alle möglichen Wertekombinationen des Merkmalvektors $\vec{x}$ bekannt sein muss. Die annotierten Trainingsdaten weisen meist aber eine begrenzte Anzahl von Wertekombinationen in ausreichender Zahl auf. So muss bei der Auswahl von Merkmalen Acht gegeben werden, dass nur Werte inkludiert werden, deren Wahrscheinlichkeit des gemeinsamen Auftretens mit y sich durch den annotierten Textcorpus vorhersagen lässt [54].

Im Ansatz von Zhou und Su [64] werden HMMs in einem gewissen Rahmen erweitert, sodass auch weitere Merkmale von Beobachtungen verarbeitet werden können. Die Entwickler arbeiten einerseits mit Trigrammen statt nur mit Bigrammen. Außerdem werden zusätzliche Oberflächenmerkmale der Wörter berücksichtigt, nämlich semantische Klassifikationen, das Vorkommen in Namenslisten und ob gleiche Wörter an anderer Stelle im Text bereits als Eigennamen identifiziert wurden. Das System zeigt auf den Daten von MUC-6 und MUC-7 bessere Leistungen als alle bisherigen Systeme.

Maximum Entropy

Die weiter oben beschriebenen Schwierigkeiten des generativen Ansatzes führten zu alternativen, diskriminativen Ansätzen. Diese modellieren anstatt der gemeinsamen Verteilung $P(x, y)$ die bedingte Wahrscheinlichkeit des Auftretens einer Namensklasse y, gegeben ein Wortvorkommen x; sie begrenzen sich also auf $P(y|x)$. Da dann nicht mehr die Wahrscheinlichkeit der Beobachtung x benötigt wird, können viel mehr Merkmale in die Berechnung mit einbezogen werden [54].

Maximum Entropy (ME) gehört zu den diskriminativen Lernverfahren und wurde im System MENE [11] im Rahmen der MUC-7 erfolgreich eingesetzt. Ein ME-Klassifizierer berechnet $P(y|x)$, das heißt, für ein Wort x die entsprechende Klassenzugehörigkeit y. ME ermöglicht dabei die Beachtung einer Vielzahl von binären Merkmalen und berechnet für jedes dieser Merkmale ein Gewicht. Bei der Ermittlung dieser Gewichte wird nach dem Prinzip der maximalen Entropie, also der größtmöglichen Ungewissheit, vorgegangen. Es soll jenes Modell erstellt werden, welches nur Vorhersagen trifft, die auch in den Trainingsdaten beobachtet werden konnten. Aus den berechneten Wahrscheinlichkeiten der Klassenzugehörigkeit wird meist wieder mittels Viterbi-Algorithmus die wahrscheinlichste Sequenz von Klassen ermittelt [54].

Ausführlichere Diskussionen über ME inklusive einer Beschreibung des Berechnungsvorgangs finden sich in [52]. Wie viele Autoren festgestellt ha-

ben, ist der Kernpunkt der ME, dass sie Modellierern erlaubt, sich auf das Finden von Merkmalen zu konzentrieren, während sich die gleichbleibende ME-Berechnung um die Zuweisung der Gewichte kümmert [11].

Beim System MENE wurden deterministische Oberflächenmerkmale wie die Großschreibung von Wörtern benutzt; lexikalische Merkmale geben an, ob ein Wort bereits in den Trainingsdaten gesehen worden ist. Andere Merkmale legen fest, um welchen Abschnitt im Text (Überschrift, Hauptteil) es sich handelt. Außerdem wird mit einer Reihe von Listen mit nützlichen Personen-, Orts- und Organisationsnamen gearbeitet, weiters mit Aussagen von externen Systemen, welche ebenfalls auf Merkmale abgebildet werden. Mithilfe geeigneter Trainingsdaten kann MENE den Voraussagen von regelbasierten Systemen widersprechen und deren Schwächen korrigieren [11]. In späteren Versionen von MENE wird auch mit der Auflösung von Referenzen gearbeitet, das heißt, ob ein Wort an anderer Stelle im selben Text bereits als Eigenname identifiziert wurde [12].

Conditional Random Fields

Ebenfalls einen generativen Ansatz skizzieren Conditional Random Fields (CRFs) [30], welche eine Erweiterung von HMMs und Maximum Entropy Markov Modellen (MEMMs) sind. Im Gegensatz zu MEMMs hängt bei CRFs ein Element nicht nur vom aktuellen und vom Vorgängerzustand ab, sondern von der ganzen Beobachtungsfolge.

MEMMs sind potentielle Opfer des „Label Bias“-Problems: Übergänge, die einen Status verlassen, konkurrieren nur gegeneinander anstatt alle anderen Übergänge im Modell zu berücksichtigen. Das führt dazu, dass Zustände mit wenigen Nachfolgern bevorzugt werden. Im Extremfall, wenn ein Status nur einen einzigen möglichen Nachfolgestatus hat, wird die Beobachtung ganz ignoriert. CRFs lösen eben diese Schwierigkeit, indem sie die ganze Beobachtungsfolge miteinbeziehen [30].

Während MEMMs die Wahrscheinlichkeiten einer Klassenzugehörigkeit eines Wortes zu einem bestimmten Zeitpunkt berechnen, modellieren CRFs die gesamte Sequenz von Klassenzugehörigkeiten, gegeben eine Sequenz von Beobachtungen [30].

Auf NER wurden CRFs zum ersten Mal von [35] angewandt, jedoch mit mittelmäßigen Ergebnissen. Nichtsdestotrotz liegt die Attraktivität von CRFs darin, dass sie mit einer großen Menge an Merkmalen umgehen können. Sie werden zum Beispiel in NER-Systemen der biomedizinischen Domäne erfolgreich eingesetzt, wie Evaluationskampagnen (BioNLP/NLPBA 2004, BioCreAtIvE I 2004 and BioCreAtIvE II 2006) [56, 62] beweisen.

Support Vektor Maschinen

Auch Support Vektor Maschinen (SVM) stellen einen diskriminativen Lernalgorithmus dar, welcher eine größere Menge an Merkmalen handhaben kann und widerstandsfähig gegen Overfitting ist. Andere diskriminative Verfahren konzentrieren sich auf die Modellierung der bedingten Wahrscheinlichkeit der Klassenzugehörigkeit einer Beobachtung $P(y|x)$. SVMs jedoch begrenzen sich auf die Suche nach dem besten Klassifizierer eines Merkmalvektors $h(\vec{x})$. Gearbeitet wird dabei immer mit einer binären Klassifikation, d.h. es wird bestimmt, ob es sich bei einem Wort z.B. um einen Eigennamen vom Typ Person handelt oder nicht [54].

SVMs beziehen als Input eine Menge von Trainingsdaten. Ein Trainingsbeispiel besteht dabei aus einem mehrdimensionalen Merkmalsvektor $\vec{x}_i$ mit einer zugehörigen Klassenbezeichnung y_i. Das Ziel ist nun, ein Modell zu finden, welches neue Beispiele für $\vec{x}$ möglichst korrekt klassifiziert. Die Ausgabe -1 oder +1 zeigt an, ob eine Klasse einem Merkmalsvektor zugeordnet wurde oder nicht. SVM sucht hier nach einer Hyperebene, welche die positiven von den negativen Beispielen mit größtmöglichem Abstand separiert (Abb. 5.4). Jene Instanzen, die der Hyperebene am nächsten liegen, werden Supportvektoren bezeichnet [27].

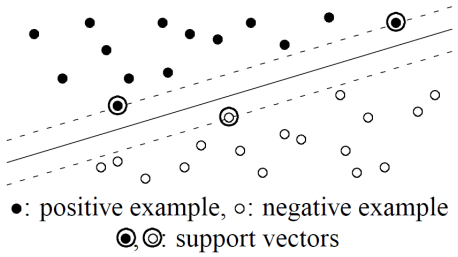

Abbildung 5.4: Aufgabe einer binären Klassifikation: Finden der Trennebene mit größtem Abstand [27].

Wenn nicht alle Daten linear separiert werden können, muss eine „weiche Trennung“ eingesetzt werden und SVM akzeptiert, dass manche Trainingsdaten fehlklassifiziert werden. Ziel ist es, den Fehler ϵ_i zu minimieren, welcher bestimmt, wie weit ein Beispiel jenseits der Hyperebene liegt. Eine andere Möglichkeit sind nicht-lineare SVMs. Sie bilden mithilfe des Kernel-Tricks den Merkmalsraum auf einem höherdimensionalem Raum ab und führen oft zu besseren Ergebnissen der Klassifikation, bedeuten aber auch einen höheren Trainingsaufwand [54].

SVMs werden meist dann eingesetzt, wenn mit einer Vielzahl von Merkmalen umgegangen werden muss. Beispiele sind japanische Texte [2], biomedizinische Texte [58] oder sprachenunabhängige NER-Systeme [34].

Halb- und unüberwachtes Lernen

Bei halb- und unüberwachtem Lernen werden im Gegensatz zu überwachtem Lernen keine vorbereiteten annotierten Trainingsdaten benötigt, welche oft schwer zu beschaffen sind.

Eine wichtige Technik in diesem Bereich ist „Bootstrapping“, welches von Riloff et al. angewandt wurde, um automatisch Lexika von Wörtern einer semantischen Klasse zu erstellen [53]: Der präsentierte Ansatz benötigt als Input lediglich eine „Saat“ von Wörtern einer semantischen Klasse und nicht-annotierte Trainingstexte. Die Autoren beginnen mit einer Reihe von Wörtern einer bestimmten semantischen Kategorie, beispielsweise mit einigen Orten wie „Guatemala“, „Honduras“ und Bolivia“. Anhand dieser Wörter kann in den Trainingsdaten nach Mustern gesucht werden, welche auf weitere Wörter der gesuchten semantischen Klasse hinweisen (z.B. „offices in x“, „facilities in x“, „operations in“). Diese Muster werden nach Verlässlichkeit gereiht und nur die am besten bewerteten werden dazu benutzt, anschließend neue Beispiele abzuleiten. Dieser Vorgang wiederholt sich etwa 50mal, sodass viele Einträge in das semantische Lexikon hinzugefügt werden. Die Autoren merken an, dass der Algorithmus sich rapide verschlechtert, sobald fehlerhafte Einträge in das Lexikon gelangen. Obwohl Precision und Recall im beschriebenen Verfahren relativ niedrig ausfällt, bedeutet ihre Arbeit einen großen Einfluss auf die weitere Forschung [46].

Ähnlich ist der Ansatz von Collins und Singer [14], welcher etwa zeitgleich aber unabhängig von Riloff et al. entwickelt wurde, um Eigennamen zu identifizieren. Die Autoren suchen mit einem Syntax-Parser nach spezifischen Eigennamen-Mustern. Ein Muster ist beispielsweise ein Eigenname, dem eine Nominalphrase folgt („Maury Cooper, a vice president at S&P“). Diese Muster werden in Paaren (spelling, context) festgehalten, wobei „spelling“ den Eigennamen bzw. dessen Oberflächenmerkmale darstellt und „context“ auf die Nominalphrase im Kontext hinweist. Weiters gibt es ein kleines Set von Startregeln (z.B. „New York“ und „California“ sind Ortsangaben; auf „Mr.“ folgt üblicherweise ein Personenname, etc.). Alle Kandidaten aus dem Text, welche nun eine dieser Startregeln erfüllen, werden dementsprechend klassifiziert und ihre Kontexte gesammelt. Die am häufigsten gefundenen Kontextregeln werden zu den Regeln hinzugefügt. Somit können wieder neue Eigennamen abgeleitet werden. Das Verfahren wird schrittweise wiederholt.

Spätere Vorgehen arbeiten mit extrem großen Textcorpora, welche es erlauben, präzise, aber seltene Merkmale zu extrahieren [18]. Paşca et al. [49] benutzen dabei Techniken, die dem Bootstrapping ähnlich sind. Mithilfe eines Wortähnlichkeitmaßes von Lin [31] fassen sie außerdem Wörter derselben semantischen Klasse zusammen, um generelle Muster abzubilden. So wird zum Beispiel im Muster „X was born in September“, September als Synonym für {March, October, April, Mar., ...} betrachtet. Paşca et al. haben gezeigt, dass mit ihrer Technik sehr große Textcorpora (100 Mio. Webdokumente)

genutzt werden können, um Fakten (z.B. Person-Geburtsjahr) abzuleiten. Bereits mit einer Saat von lediglich 10 Beispielfakten, konnten eine Million Fakten mit einer Präzision von ca. 88% abgeleitet werden.

Etzioni et al. [19] arbeiten ebenfalls mit Webdokumenten und extrahieren daraus Listen von Eigennamen. Sie stützen sich dabei auf die Arbeit von Hearst [24], welcher mit Suchmaschinenanfragen Hypernymbeziehungen (Unter-/Überbegriffe) aufzudecken versuchte. So würde eine Anfrage wie „such as X“ darauf hindeuten, dass ein Wort vor dieser Anfrage ein Hypernym von X darstellt. Das System von Etzioni extrahiert also mithilfe eines geeigneten Patterns für „City“ für den Satz „We provide tours to cities such as Paris, London, and Berlin.“ drei Listenkandidaten für die Klasse City. Anhand des PMI-IR Algorithmus von Turney [60] werden die Listenkandidaten beurteilt. PMI-IR misst die Abhängigkeit zweier Ausdrücke wieder mittels Suchmaschinenanfragen, wobei ein hoher PMI-IR Wert bedeutet, dass Ausdrücke oft gemeinsam auftreten.

Auch Toral und Muñez [59] versuchen, automatisch Listen von Eigennamen zu erstellen. Sie analysieren dabei jeweils den ersten Satz von Wikipedia-Einträgen, der meist eine Definition darstellt.

In den letzten Jahren versuchen viele Autoren die Vorteile von Wikipedia für verschiedene Zwecke auszuschöpfen. Da es sich bei Wikipedia um eine Enzyklopädie handelt, stellen die meisten Artikel Eigennamen dar. Die Einträge sind außerdem sehr zahlreich, werden täglich aktualisiert und sind auch in einer gewissen Weise strukturiert, wodurch sie besser zu bearbeiten sind als roher Text [29]. So präsentierten Bunescu und Paşca [13] eine Methode, wie Eigennamen mithilfe von internen Links in Wikipedia disambiguiert werden können. Weiters wurde Wikipedia benutzt, um überwachte Lernsysteme zu verbessern: Kazama und Torisawa [29] extrahierten Kategorienamen aus dem jeweils ersten Satz von Wikipediaeinträgen und verwendeten diesen als zusätzliches Merkmal für CRFs. Nothman et al. [48] und andere wiederum benutzen Wikipedia, um einen großen, mit Eigennamen annotierten Trainingskorpus für statistische Modelle zu erstellen.

Cucerzan und Yarowsky [15] stellen ein NER-System vor, welches aufgrund des rein zeichenorientiertes Ansatzes prinzipiell auf alle Sprachen anwendbar ist. Der Ansatz ist ebenfalls ein Bootstrapping-Algorithmus und basiert auf der wiederholten Bewertung von kontextuellen und morphologischen Mustern. Diese Muster werden in rein zeichenbasierten Strukturen dargestellt. Texte werden also nicht von Vornherein aufgrund von Trennungszeichen wie Leer- und Satzzeichen in Token eingeteilt, sondern jedes einzelne Zeichen kann den Anfang oder das Ende einer gesuchten Entität darstellen. Diese Vorgehensweise ermöglicht auch die Verarbeitung von Sprachen wie Chinesisch, welches üblicherweise keine Trennungszeichen zwischen Wörtern enthält.

Elsner et al. [18] präsentieren ein NER-System, das völlig unüberwacht ist, und lediglich mittels Clustering Eigennamen extrahiert, sowie in ihnen

enthaltene Wortklassen. Wortklassen sind Gruppen von Wörtern, welche die gleiche Rolle darstellen, wie Titel, Vornamen oder Arten von Organisationen. Mit diesen Strukturen können Konsistenzen zwischen Entitäten festgestellt werden, dass beispielsweise eine Person „Mr. Mark Johnson“ konsistent mit „Johnson“ ist. Das Modell verwendet Merkmale des Eigennamens selbst und dessen syntaktische Struktur, aber auch Informationen der Koreferenz. Auf den Daten von MUC-7 erreichte das System immerhin ein F-Maß von 86%, das bisher beste Ergebnis für ein unüberwachtes Modell.

5.5 Verschlagwortung mit Wikipedia

Erkennbar ist, dass viele Semantic APIs beim Zuweisen von Schlagwörtern stark mit Inhalten aus Wikipedia arbeiten. So merkt man, dass sowohl Links als auch Keywords von Zemanta zu einem Großteil Themen aus Wikipedia darstellen. Auch sind die Social Tags von OpenCalais und die Konzepte von AlchemyAPI eigentlich Begriffe aus Wikipedia. In diesem Kapitel werden deshalb Ansätze vorgestellt, welche sich mit ähnlichen Techniken befassen und deren Möglichkeiten bzw. Schwierigkeiten aufzeigen: Begriffe in Texten werden auf entsprechende Wikipediathemen verlinkt; Wikipedia-Artikel oder Wikipedia-Kategorien werden als kontrolliertes Vokabular für eine Verschlagwortung von Dokumenten verwendet; Wikipedia dient als Unterstützung für automatische Taggingsysteme.

Wikipedia wurde in den letzten Jahren für die NLP-Forschung entdeckt und ausgebeutet. Als größte Online-Enzyklopädie enthält Wikipedia Millionen von Artikeln in vielen verschiedenen Domänen und Sprachen. Diese können z.B. als großes kontrolliertes Vokabular für die Kategorisierung von Dokumenten verwendet werden. Durch die interne Beschaffenheit der Enzyklopädie können sogar thesaurus-ähnliche Strukturen für dieses Vokabular abgeleitet werden [43]. Folgende Strukturen helfen, viele nützliche Statistiken über Terme und Beziehungen zu extrahieren [40]:

- Die Artikel in Wikipedia stellen Eigennamen dar; sie können mit einer eindeutigen Zeichenfolge identifiziert werden.
- Von großem Wert sind die Vielzahl an Links, die in Artikeln eingebettet sind und wichtige Begriffe zu anderen Seiten verlinken.
- Auch werden Artikel bestimmten Kategorien zugewiesen, welche ihrerseits wieder breiteren Kategorien unterstellt sind.
- Mithilfe von Weiterleitungen (redirect pages) werden verwandte Begriffe oder Synonyme auf eine gemeinsame Beschreibung verwiesen.
- Außerdem gibt es im Fall von mehrdeutigen Begriffen eigens erstellte Seiten für Begriffsklärungen (disambiguation pages, Abb. 5.5). Diese verweisen mit Links zu jenen Artikeln, welche die entsprechende Beschreibung einer Bedeutung enthalten.

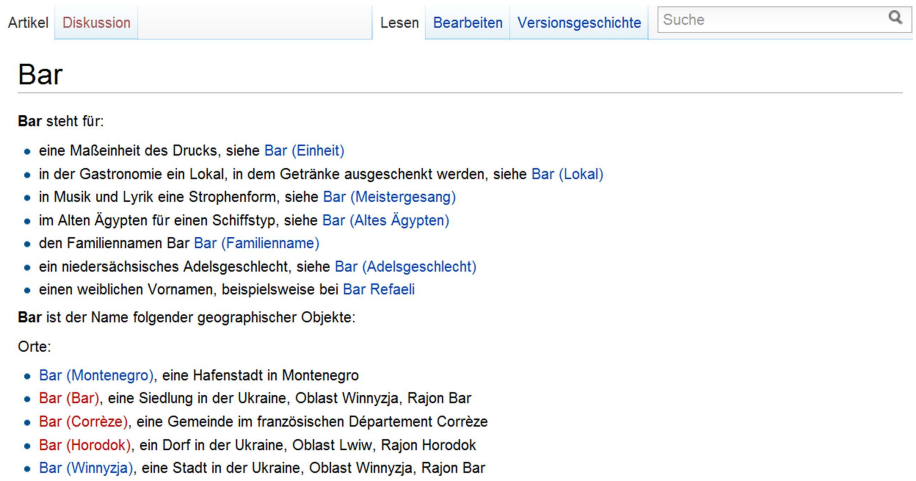

Abbildung 5.5: Disambiguation Page aus Wikipedia für den Begriff „Bar“.

Viele Strukturen und Eigenschaften in Wikipedia helfen also, die Enzyklopädie gewinnbringend für NLP-Anwendungen einzusetzen. Mit einem Problem namens „Wikification“ beschäftigen sich Mihalcea und Csomai [40]. Es geht dabei darum, in einem Dokument wichtige Konzepte zu identifizieren und diese dann automatisch zu passenden Wikipedia-Artikeln zu verlinken. Ihr System „Wikify“ besteht aus zwei Phasen. In der ersten Phase werden alle wichtigen Themen und Phrasen in einem Text identifiziert. Der am effektivsten getestete Ansatz für die Auswahl von Phrasen ist die Berechnung deren „Keyphraseness“. Sie wird definiert durch die Anzahl der Wikipedia-Artikel, die einen Begriff als Anker benutzen, geteilt durch die gesamte Anzahl der Dokumente, in denen dieser Begriff auftaucht. In anderen Worten bedeutet das, je öfter ein Begriff innerhalb von Wikipedia verlinkt wurde, desto größer ist die Wahrscheinlichkeit, dass er auch erneut zur Verlinkung ausgewählt wird. In der nächsten Phase müssen alle identifizierten Themen zum richtigen Artikel in Wikipedia verlinkt werden, denn oft deutet eine Phrase auf mehrere Einträge (Abb. 5.5). Der beste Ansatz des Systems ist hier, die Eigenschaften (POS-Tags und Wörter selbst) der Phrase und der umliegenden Wörter im Dokument zu verwenden und mit den Trainingsdaten aus Wikipedia zu vergleichen. Mit ihrem System haben die Entwickler zum ersten Mal gezeigt, dass Wikipedia eine hilfreiche Ressource sowohl für die Keyword Extraction als auch für die Disambiguierung von Wörtern darstellt.

Medelyan et al. [39] widmen sich einer ähnlichen Aufgabe. Sie konzentrieren sich auf die Problemstellung, wie man Texte mit Artikeln aus Wikipedia effektiv verschlagworten kann. Das Problem ähnelt dem der Wikification, da auch hier Kandidaten für Themen identifiziert werden müssen und Mappings zu den entsprechenden Wikipedia-Artikeln vorgenommen werden. Zusätzlich folgt am Ende jedoch ein Schritt der Gewichtung der Themen.

Beim Identifizieren des richtigen Wikipedia-Artikels gehen die Entwickler etwas anders vor als im System Wikify. Bei der Disambiguierung von mehrdeutigen Kandidaten berücksichtigen sie Beziehung und Häufigkeit („Commonness und Relatedness“) eines Kandidaten: Sie verwenden als Kontext bereits eindeutig identifizierte Kandidaten und bestimmen die durchschnittliche semantische Verwandtschaft [61] zu diesen. Diese berechnen sie, indem die Links in den entsprechenden Wikipedia-Artikeln miteinander verglichen werden und festgestellt wird, wie sehr sich diese Links überschneiden (Abb. 5.6). Zusätzlich wird zur Disambiguierung die Häufigkeit eines Wikipedia-Artikels miteinbezogen. Beispielsweise verlinkt das Wort Jaguar häufiger zum Wikipedia-Eintrag „Jaguar cars“ als zum Eintrag des Tieres [39].

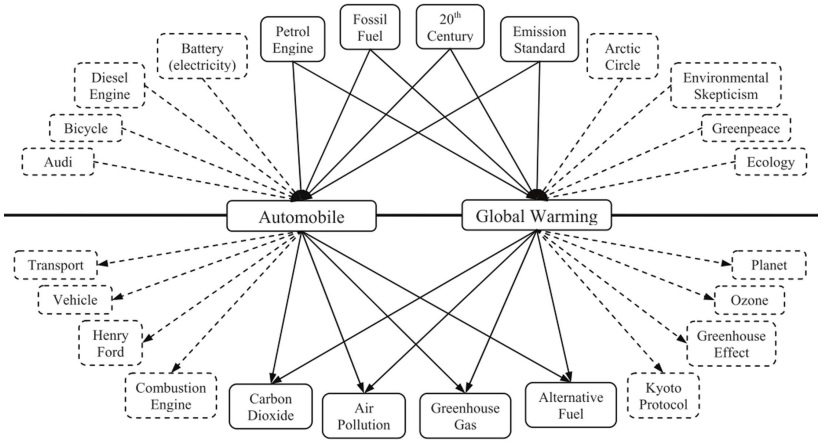

Abbildung 5.6: Durch die Links in Wikipedia kann eine semantische Verwandtschaft zwischen den Begriffen „Automobile“ und „Global Warning“ berechnet werden [61].

Wurden alle Kandidaten endgültig einem Wikipedia-Artikel zugewiesen, folgt am Ende ein Schritt der Filterung, welcher jeden Kandidaten mithilfe von statistischen und semantischen Eigenschaften charakterisiert und aufgrund dessen eine Gewichtung des Terms im Text vornimmt. Als Eigenschaften wurden folgende gewählt [39]:

- **TFxIDF** ist ein wichtiges Maß aus dem Information Retrieval zur Gewichtung von Termen in einem Dokument. Es vergleicht die Häufigkeit eines Terms in einem bestimmten Dokument mit der Häufigkeit, in der der Term überhaupt auftritt. Dieser Wert ist hoch für eigentlich seltene Phrasen, die aber oft in einem Dokument auftauchen.
- **Position des ersten Vorkommens**: Dieser Wert ist die relative Distanz des ersten Auftretens einer Phrase vom Beginn des Dokuments. Kandidaten mit besonders hohen oder niedrigen Werten sind mit höherer Wahrscheinlichkeit geeignete Indexterme, weil diese besonders

häufig am Anfang und am Ende eines Dokuments auftauchen.

- **Länge**: Experimente haben gezeigt, dass von menschlichen Verschlagwortern längere Prasen bevorzugt werden.
- **Node degree**: Hier geht es um die Zahl der Links im entsprechenden Wikipedia-Artikel, die zu Artikeln führen, die ebenfalls als Kandidaten für Terme identifiziert wurden. Ein Dokument, das ein gewisses Thema beschreibt und viele verwandte Konzepte beinhaltet, ist mit höherer Wahrscheinlichkeit signifikant.
- **Document-specific keyphraseness**: Für jeden Kandidaten wird der oben erläuterte Keyphraseness-Wert berechnet und dieser mit der Anzahl des Vorkommens des Terms im Dokument multipliziert.

Erweitert wird dieser Ansatz von Medelyan et al. in [37], die einen Algorithmus für automatisches Tagging entwickelten. Sie sehen das automatische Vorschlagen von Tags als eine potentielle Lösung für ein wesentliches Problem in Tagging-Plattformen: die Inkonsistenz von vergebenen Tags. Diese entsteht u.a. aufgrund der Polysemie und Synonymie[1] der menschlichen Sprache sowie aus unterschiedlichen Spezifitätsgraden, die Menschen zum Taggen verwenden [21].

Auch dieses System basiert auf zwei Schritten [37]: Zuerst werden passende Phrasen aus dem Text, die als Tags in Frage kommen, ausgewählt. Anschließend werden diese anhand eines Klassifikationsmodells gefiltert. Neben den oben beschriebenen Eigenschaften zur Bewertung der Kandidaten, werden hier noch einige weitere berücksichtigt:

- **Keyphraseness**: Das Modell integriert zwei Arten von Keyphraseness. Die herkömmliche Keyphraseness bestimmt, wie oft eine Phrase als Tag bereits in einem Trainingskorpus vorkommt. Viele automatische Tagging-Systeme [45] benutzen diesen Ansatz und schlagen Tags vor, welche in ähnlichen Dokumenten bereits zugewiesen wurden.
- **Wikipedia-based keyphraseness**: Diese Keyphraseness wurde weiter oben schon beschrieben und stellt dar, mit welcher Wahrscheinlichkeit eine Phrase in Wikipedia auch ein Link ist.
- **Spannweite**: Die Spannweite einer Phrase stellt die Distanz zwischen erstem und letztem Vorkommen im Text dar. Hohe Werte legen fest, dass Phrasen sowohl am Beginn als auch am Ende eines Dokuments verwendet wurden.
- **Semantische Verwandtschaft**: Mithilfe der semantischen Verwandtschaft kann die Beziehung einer gewissen Phrase zu allen anderen Kandidaten festgelegt werden. Je höher dieser Wert ist, desto wahrscheinlicher ist die Phrase auch ein geeigneter Tag.

[1] Während die Polysemie ein Wort bezeichnet, welches mehrere Bedeutungen hat, geht es bei der Synonymie um verschiedene Wörter, welche die gleiche Bedeutung haben.

- **Inverse Wikipedia linkage**: Dieses Merkmal berechnet die Anzahl von externen Wikipedia-Artikeln, die auf einen Kandidaten verlinken im Vergleich zur Anzahl aller Links in Wikipedia. So werden jene Konzepte gut bewertet, welche benutzt werden, um andere Konzepte zu beschreiben.

Der Algorithmus erreicht im Vergleich zu menschlichen Taggern konkurrenzfähige Ergebnisse. Bei der Einzelbewertung der verwendeten Eigenschaften schneiden Keyphraseness, Wikipedia-based keypraseness, TFxIDF und Spannweite am besten ab [37].

Milne und Witten [44] beschäftigen sich erneut mit dem Problem der Wikification und beschreiben einen Algorithmus, wie man bestimmte Phrasen in einem Text auf Wikipediathemen verlinken kann. Was dieses System von den bisher genannten unterscheidet, ist erstens, dass die Wikipedia-Artikel nicht nur als Informationsquelle verwendet werden, sondern auch als Trainingsdaten. Zweitens führen die Entwickler im System zuerst den Schritt der Disambiguierung durch und entscheiden dann erst, welche Phrasen im Text wichtig genug sind, um verlinkt zu werden. Bei der Disambiguierung von Phrasen nehmen sie genauso wie [39] Rücksicht auf Commonness und Relatedness der potentiellen Wikipedia-Artikel. Sie beziehen demnach ebenfalls eindeutige Phrasen im Text als Kontext mit ein, um mehrdeutige disambiguieren zu können. Dabei beachten sie aber zusätzlich, dass nicht alle dieser Kontextphrasen gleich nützlich sind, sondern sie bewerten diese. Das Wort „the“ beispielsweise ist zwar eindeutig, da es fast immer auf das grammatische Konzept des Artikels verlinkt, trägt aber nicht viel bei, um andere Phrasen zu disambiguieren, denn das Wort taucht in sehr vielen Texten auf, ohne überhaupt verlinkt zu werden. Außerdem sind manche der Kontextphrasen Ausreißer und haben mit dem zentralen Thema des Textes wenig zu tun. Aufgrund dessen werden die Kontextphrasen in einem Dokument bewertet, indem deren Linkwahrscheinlichkeit in Wikipedia und semantische Verwandtschaft zu ihren Kontextphrasen berechnet wird. Bei der Kombination der beiden Werte wird außerdem Rücksicht genommen auf die Homogenität des Textes selbst. Ist der Kontext verworren und nicht eindeutig, wird auf die Häufigkeit eines Wortes mehr Wert gelegt. Um zu entscheiden, welche der Phrasen im Text schlussendlich verlinkt werden, werden einem Klassifizierer verschiedene Merkmale der Phrasen bzw. der zugehörigen Wikipedia-Artikel mitgeteilt.

Im System von Gabrilovich und Markowitch [20] werden Texten Konzepte aus Wikipedia zugewiesen, um anschließend deren semantische Verwandtschaft zu bestimmen. Jeder Wikipedia-Artikel stellt ein mögliches Konzept dar. Er wird durch einen Vektor repräsentiert, der die entsprechenden Wörter eines Artikels enthält, welche mithilfe des TFxIDF-Schemas gewichtet werden. Anhand eines invertierten Indexes wird nun jedem einzelnen Wort eine Liste von gewichteten Konzepten zugewiesen. In Folge kann durch

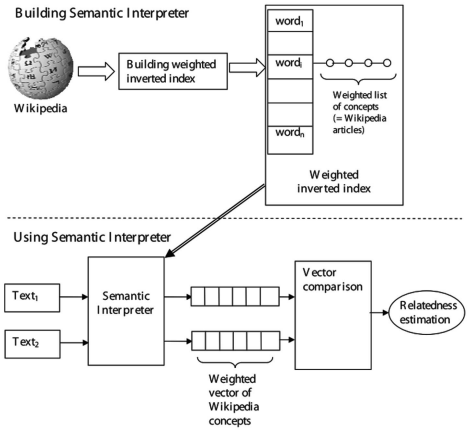

Abbildung 5.7: Die Berechnung einer semantischen Verwandtschaft von Wörtern oder Texten mithilfe Wikipedia [20].

einen centroid-basierten Klassifizierer auch ganzen Texten eine Liste von gewichteten Wikipedia-Konzepten zugeteilt werden. Anschließend wird mittels Cosinus-Maß deren Verwandtschaftsgrad bestimmt. Abb. 5.7 stellt die Vorgehensweise schematisch dar. Der Algorithmus erweist sich in der Berechnung von semantischen Verwandtschaften einzelner Wörter oder ganzer Texte sehr erfolgreich und zeigt außerdem eine Möglichkeit auf, wie Texte mit einzelnen Konzepten aus Wikipedia verschlagwortet werden können.

Schönhofen [55] beschreibt den Inhalt von Dokumenten mit den Kategorien aus Wikipedia. Dabei werden rein Titel und Kategorien von Wikipedia-Artikeln, aber nicht deren Texte in den Algorithmus miteinbezogen. Indem die Wörter eines Dokuments mit den Titeln von Wikipedia-Artikel abgeglichen werden, können Kategorien für ein Dokument identifiziert werden. Einige Faktoren spielen bei der Bewertung eine Rolle:

- gemeinsam benutzte Wörter in Dokument und Titel des Wikipedia-Artikels,
- die Stärke der Verbindung zwischen Dokument und Wikipedia-Artikel,
- die Wikipedia-Artikel selbst (ob es beispielsweise sehr viele Artikel mit ähnlichem Namen gibt)
- und das Ausmaß, indem Wörter, welche die aktuelle Kategorie unterstützen, auch andere Kategorien unterstützen, die noch mehr zum Dokument passen.

So können auch mithilfe der Kategorien in Wikipedia Texte verschlagwortet werden. Eine Erweiterung für den aktuellen Ansatz wäre es, zusätzlich Artikel-Texte, Links und die hierarchische Struktur der Kategorien in den Algorithmus mit einzubeziehen [55].

Wie an den vorgestellten Ansätzen deutlich wurde, hat Wikipedia ein

großes Potenzial, Informationen in einer geeigneten Weise zu beschreiben und zu organisieren. Im Vergleich zu traditionellen Ansätzen bietet es eine Art Mittelweg zwischen Qualität und Quantität. Bislang wurde sehr viel mit professionell erstellten Ressourcen wie Thesauri und Ontologien gearbeitet. Diese bieten zwar eine hohe Qualität, sie sind im Umfang und in der Abdeckung von Domänen jedoch immer eingeschränkt. Eine hohe Quantität, aber geringere Qualität, liefern wiederum Ansätze, welche mit sehr großen, jedoch unstrukturierten Textmengen versuchen, Informationen abzuleiten. Die Daten aus Wikipedia bieten einen Kompromiss zwischen diesen Vorgehensweisen. Mit einer riesigen Menge von Artikeln, Tausenden von Mitwirkenden, dem Potenzial noch weiter zu wachsen und einer Fülle von nützlichen Strukturen stellt es jede andere manuell erstellte Ressource in den Schatten. Dennoch muss gesagt werden, dass Wikipedia nicht das gleiche Ausmaß an Vertrauen bietet, wie eine professionell erstellte Ressource und immer weniger umfangreich und repräsentativ für die menschliche Sprache bleiben wird als das gesamte World Wide Web [38].

Kapitel 6

Evaluierung von Semantic APIs

Nach der Darstellung von Methoden, wie Semantic APIs arbeiten, folgt in diesem Kapitel eine Evaluierung der APIs OpenCalais, AlchemyAPI und Zemanta bezüglich der Eigennamenerkennung. In die Analyse eingeschlossen werden dabei die Fähigkeiten der Disambiguierung und das Angebot von Links in die Linking Open Data Cloud. Dargelegt werden vor allem Schwierigkeiten in der Eigennamenerkennung, mit denen die Dienste konfrontiert sind. Bei der Evaluierung zurückgegriffen wird dabei auf die Testergebnisse der Anwendung Enrich aus Abschnitt 4.6. Insbesondere Personen-, Orts- und Organisationsnamen werden in Augenschein genommen, welche die gängigste Form von Entitätentypen darstellen. In Fallbeispielen werden einzelne Sätze aus den Texten präsentiert, in denen Fehler passiert sind, um vorhandene Schwierigkeiten zu verdeutlichen.

OpenCalais

OpenCalais bietet derzeit 39 unterschiedliche Entitätstypen an und stellt generell eine gute Qualität von Entitäten zur Verfügung. Eine Disambiguierung wird vorerst nur für Unternehmen, Orte und elektronische Produkte angeboten. Im Vergleich zu anderen Anbietern werden auch wenig Links zu Datenbanken wie DBpedia angeboten. Wie auf der Webseite von OpenCalais angemerkt[1], arbeitet das System stark mit Lexika und manuell erstellten Regeln. Die Entwickler versuchen, nachzuahmen, wie ein Mensch beim Lesen eines Textes eine Entität identifiziert und achten auf Hinweise in der Entität selbst, im nahen Kontext und im Kontext des gesamten Textes.

Bei den Ergebnissen von OpenCalais ist durchaus mit Fehlern zu rechnen, wie folgende Beispielen zeigen:

- **On 21 June 1940, early in the second year of World War Two, the French president, Marshall Philippe Pétain, sued for peace with Adolf Hitler's Third Reich.**

[1] http://www.opencalais.com/how-does-calais-learn

Hier wird „War Two“ irrtümlich als französischer Präsident und Marschall erkannt, da das Kontextpattern „XX, the French president, Marshall“ darauf hindeutet, dass die zwei Wörter vor dem Komma eine Person darstellen.

- **The playing of anthems at victory ceremonies is one of the most emotive parts of any Games and it was an incredible moment for me at the Moscow and Los Angeles Games.**
  Hier wird „Los“ fälschlicherweise als Stadt gekennzeichnet und Los Angeles übersehen. Das Kontextpattern „Moscow and ..“ lässt das System darauf schließen, dass eine weitere Stadt folgen muss. Da jedoch an dieser Stelle das Event „Los Angeles Games“ identifiziert wurde, scheinen sich die entsprechenden Regeln gegenseitig zu behindern.
- **Man Utd boss Ferguson backs Torres to shine for Chelsea.**
  Hier wird „Utd“ als Person klassifiziert. „Man Utd“ wird nicht als Abkürzung für „Manchester United“ erkannt, obwohl dieses im weiteren Verlauf des Textes erwähnt wird. Die Verbindung von „Man“ und einem nicht im Lexikon vorhandenen Eintrag lässt das System schließen, dass „Utd“ eine Person ist.
- **Educated at military school, he was rapidly promoted and in 1796, was made commander of the French army in Italy, where he forced Austria and its allies to make peace.**
  Die gesamte Phrase „French army in Italy“ wird hier als Organisation erkannt. Das Problem ist hier, dass das System nicht weiß, ob „in Italy“ Teil dieser Phrase ist, da generell auch Eigennamen, welche Wörter mit Kleinbuchstaben (z.B. „Bank of Scotland“) beinhalten, erkannt werden.
- Weiters tendiert OpenCalais dazu, dass Unternehmen manchmal übersehen werden, insbesondere wenn diese eher unbekannt sind und nicht mit einer entsprechenden Kontextphrase eingeleitet werden.

Der Grund für die entstandenen Fehler ist die Komplexität und Vielfalt der natürlichen Sprache. Die erstellten Kontextregeln treffen zwar in der Regel zu, jedoch kommt es durch Unregelmäßigkeiten in der Sprache, wie beispielsweise komplexe Satzstellungen, immer wieder zu Ausnahmefällen, in denen eine Regel falsch klassifiziert.

AlchemyAPI

AlchemyAPI unterstützt laut eigenen Angaben Hunderte von Entitätentypen, welche auf deren Webseite einzusehen sind. Die API bietet im Vergleich zu OpenCalais bessere Disambiguierungs-Fähigkeiten für eine Reihe von Entitäten und in Folge dazu sehr viele Links zu Datenbanken wie DBPedia oder Freebase. Über die Funktionsweise von AlchemyAPI ist nur so viel bekannt, dass die Anbieter stark mit Maschinenlernen und statistischen Algorithmen arbeiten. Interessant ist, dass keiner der beschriebenen Fehler von OpenCa-

lais hier auftaucht, jedoch kommt es zu anderen:

- Kommt in einem Text der ganze Name einer Person vor (z.B. „Fernando Torres“) und an anderer Stelle nur der Vorname („Fernando“), so kann das System nicht erkennen, dass es sich hier um ein- und dieselbe Person handelt. Dieses Problem tritt nicht nur bei Personen auf, sondern auch bei Organisationen. So kann zwischen „Manchester United“ und „United“ keine Verbindung hergestellt werden.
- Wie oben beschrieben, versucht das System, im Text extrahierte Eigennamen zu disambiguieren. So kann für eine Person, welche im Text nur als „Lord Coe“ bezeichnet wird, eindeutig herausgefunden werden, dass es sich bei diesem um „Sebastian Coe“ handelt. Verbindet man nun diese Fähigkeit mit dem eben beschriebenen Problem der Vornamenzuordnung, kommt es dazu, dass „Sir Terry“, nicht als der im Text erwähnte „Sir Terry Pratchett“, sondern als „Sir Terry Wogan“ erkannt wird.
- **Chelsea was defeated by Manchester.**
  Weiters hat AlchemyAPI die Tendenz, unbekannte Begriffe als Personen zu klassifizieren. So wurde in diesem Satz „Chelsea“ als Person identifiziert und „Manchester“ als Stadt.

Aufgrund der eben beschriebenen Schwierigkeiten, sollte bei AlchemyAPI deshalb vor allem die Fähigkeit verbessert werden, Teilphrasen richtig zuzuweisen. Weiters sollte an der Fertigkeit gearbeitet werden, unbekannte Begriffe zu klassifizieren.

Zemanta

Zemanta führt keine Eigennamenerkennung im klassischen Sinn durch, da wie in Abschnitt 4.6 bereits deutlich wurde, Entitäten, die nicht öffentlich bekannt sind, auch nicht identifiziert werden können. Stattdessen werden Phrasen im Text mit Begriffen aus der Datenbank von Zemanta abgeglichen und somit Entitäten extrahiert bzw. Links zu vielen verschiedenen Portalen erstellt. Diese Vorgehensweise ähnelt etwas dem im vorherigen Kapitel beschriebenen Problem der Wikification. Zusätzlich werden Entitätentypen angeboten, sofern diese identifiziert werden können, wobei hier stark mit Schemata aus Freebase gearbeitet wird.

Die Entitäten bzw. Links von Zemanta stellen durchwegs gute und relevante Ergebnisse für Texte dar. So konnte das System beispielsweise aus der Phrase „Scot Paul di Resta“ als einziges richtig schließen, dass „Paul di Resta“ einen Personennamen darstellt, das Wort „Scot“, welches im Englischen entweder einen Vornamen oder eine Nationalität darstellt, aber nicht Teil davon ist. Das System bietet weiters ausgezeichnete Fähigkeiten in der Disambiguierung und es kommt selten zu Fehlern. Trotzdem konnten einzelne Fehler identifiziert werden:

- **But more than 100 Britons have gone to Dignitas and no family members or friends have yet been prosecuted.**
  Hier werden die Worte „100 Britons“ auf „100 Greatest Britons“ in Wikipedia verlinkt, welche jedoch nicht gemeint sind.
- **United got the breakthrough two minutes before half-time when Javier Hernandez scored from Ryan Giggs's cross.**
  Der Fußballspieler „Javier Hernandez“ wird hier mit einem amerikanischen Comicbuch-Autor verwechselt.
- **An out-of-sorts Roger Federer crashed out of the Monte Carlo Masters at the quarter-final stage as Jurgen Melzer saw off the number two seed 6-4 6-4.**
  „4 6-4“ wird hier aus dem Zusammenhang gerissen und auf einen Fachbegriff im Lokomotiv-Bau verlinkt.

Durch die Analyse wird deutlich, dass die Dienste trotz der intensiven Forschung in der Eigennamenerkennung durchaus noch mit Schwierigkeiten konfrontiert sind.

Die Evaluierung der Texte legt die Vermutung nahe, dass rein regelbasierte Ansätze für die Erkennung von Eigennamen nicht ausreichen. Durch das Erweitern von Regeln besteht zwar die Möglichkeit, Fehler auszumerzen und die Qualität der Ergebnisse zu verbessern. Jedoch entwickelt sich eine Sprache ständig weiter und es kommt durch die vielen Unregelmäßigkeiten der natürlichen Sprache immer zu Konstellationen, in denen manuell erstellte Regeln falsch klassifizieren.

Da ein wichtiger Schwerpunkt der APIs aber darauf liegt, Eigennamen in Texten auf „reale“ Entitäten in der Linking Open Data Cloud zu verlinken, scheint die Ableitung statistischer Werte mithilfe von Trainingstexten von großem Nutzen zu sein.

Kapitel 7

Résumé

Semantic APIs helfen, Texte durch eine automatische Analyse der natürlichen Sprache zu strukturieren bzw. mit Metadaten zu versehen. So ergibt sich aus diesen Informationen eine Vielzahl von Anwendungsmöglichkeiten.

In dieser Arbeit wurde versucht, die Verschlagwortung von Texten in Content-Management-Systemen durch das Kombinieren mehrerer Semantic APIs zu verbessern. Es konnte bestätigt werden, dass jede API ihre eigenen Stärken und Schwächen aufweist und dass das Angebot von Tags für Texte deutlich erhöht werden kann. Jedoch ergeben sich durch die Kombination von Semantic APIs für die Verschlagwortung neue Probleme. So ist es erstens schwierig, trotz Relevanz- bzw. Confidence-Werte, fehlerhafte Ergebnisse von APIs herauszufiltern. Außerdem entstehen durch die Kombination der Ergebnisse Inkonsistenzen, zum Beispiel durch unterschiedliche Schreibweise von Namen (Apple vs. Apple Inc.) oder dem Vorschlagen von ähnlichen Begriffen. Der größte Vorteil der Kombination mehrerer APIs ist die Gegebenheit, dass beim Ausfall einer API nicht gänzlich auf Ergebnisse verzichtet werden muss, sondern auf eine der anderen zurückgegriffen werden kann.

Da über Funktionen und Anwendungsmöglichkeiten von Semantic API-Anbietern viel bekannt ist, aber wenig über Methoden, wie diese Dienste eigentlich arbeiten, wurde in der Arbeit ein Überblick über das Gebiet des Natural Language Processing gegeben. Insbesondere wurde dabei auf Probleme und Herangehensweisen der Eigennamenerkennung eingegangen, welches ein wichtiges Vorverarbeitungsgebiet für viele NLP-Anwendungen darstellt und von den meisten Semantic API-Diensten angeboten wird. Zusammenfassend kann gesagt werden, dass das Thema ein viel bearbeitetes Forschungsgebiet ist, nicht zuletzt aufgrund der vielen Evaluationskampagnen, welche die Entwicklung in dem Bereich maßgeblich vorantrieben. In weiterer Folge wurde dargelegt, dass auch Wikipedia mehr und mehr im Bereich der natürlichen Sprachverarbeitung zum Einsatz kommt und in Folge auch von Semantic API-Anwendern ausgebeutet wird. Mit einer großen

Menge an Texten in unterschiedlichen Themenbereichen, die zusätzlich in einer gewissen Weise strukturiert sind, hat Wikipedia das Potenzial, nützliche Beiträge zur Sprachverarbeitung zu leisten. Weiters bietet die Verarbeitung von Wikipedia-Artikeln eine unmittelbare Möglichkeit zur Verfügungstellung von Links in die Linking Open Data Cloud.

Bei der Evaluation der APIs bezüglich der Eigennamenerkennung wurde festgestellt, dass die Dienste durchgängig gute und qualitative Ergebnisse liefern. Dennoch zeigen Fehler in den Ergebnissen, dass die Aufgabe der Eigennamenerkennung durchaus keine triviale Aufgabe ist und trotz der intensiven Forschung noch immer Schwierigkeiten aufweist.

Anhang A

Inhalt der CD

A.1 Diplomarbeit

Pfad: /

diplomarbeit.pdf	Diplomarbeit im PDF-Format
API-Evaluierung/	Dateien der API-Evaluierung im XLS-Format

A.2 Applikation

Pfad: /Enrich

TextAnalysis.zip	Java-Servlets als Eclipse-Projekt
enrich.zip	Sourcecode der Typo3-Extension
enrich.t3x	Extension-File für Typo3
readme.txt	Installationshinweise

A.3 Sonstiges

Pfad: /

Online-Quellen/	Internetreferenzen im PDF-Format
Grafiken/	Verwendete Grafiken im EPS-Format

Literaturverzeichnis

[1] Appelt, D. E. und D. J. Israel: *Introduction to information extraction technology.* A tutorial prepared for IJCAI-99, 1999. http://www.ai.sri.com/~appelt/ie-tutorial/IJCAI99.pdf, PDF auf CD (Datei Appelt1999.pdf).

[2] Asahara, M. und Y. Matsumoto: *Japanese named entity extraction with redundant morphological analysis.* In: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, S. 8–15, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[3] Berners-Lee, T.: *Linked data.* Website, 2006. http://www.w3.org/DesignIssues/LinkedData.html, Kopie auf CD (Datei Berners-Lee2006.pdf).

[4] Berners-Lee, T., R. Fielding und L. Masinter: *Uniform Resource Identifier (URI): Generic Syntax.* RFC, 2005. http://tools.ietf.org/html/rfc3986, PDF auf CD (Datei Berners-Lee1998.pdf).

[5] Berners-Lee, T., J. Hendler und O. Lassila: *The Semantic Web.* Scientific American, 284(5):34–43, Mai 2001.

[6] Bikel, D., S. Miller, R. Schwartz und R. Weischedel: *Nymble: a High-Performance Learning Name-finder.* In: *Proceedings of the fifth conference on Applied natural language processing*, S. 194–201, Stroudsburg, PA, USA, März 1997. Association for Computational Linguistics.

[7] Bird, S., E. Klein und E. Loper: *Natural Language Processing with Python.* O'Reilly Media, Sebastopol, California, 2009.

[8] Birkenbihl, K.: *Standards für das Semantic Web.* In: Pellegrini, T. und A. Blumauer (Hrsg.): *Semantic Web*, X.media.press, S. 73–88. Springer-Verlag, Heidelberg, 2006.

[9] Bizer, C.: *The emerging web of linked data.* IEEE Intelligent Systems, 24:87–92, Sep. 2009.

[10] Bizer, C., T. Heath und T. Berners-Lee: *Linked Data - The Story So Far*. International Journal on Semantic Web and Information Systems, 5(3):1–22, März 2009.

[11] Borthwick, A., J. Sterling, E. Agichtein und R. Grishman: *NYU: Description of the MENE named entity system as used in MUC-7*. In: *Message Understanding Conference Proceedings MUC-7*, S. 1–6, Fairfax, Virginia, 1998. http://www-nlpir.nist.gov/related_projects/muc/proceedings/muc_7_proceedings/nyu_english_named_entity.pdf, PDF auf CD (Datei Borthwick1998.pdf).

[12] Borthwick, A. E.: *A maximum entropy approach to named entity recognition*. Dissertation, New York University, Computer Science Department, New York, USA, Sep. 1999.

[13] Bunescu, R. und M. Pasca: *Using encyclopedic knowledge for named entity disambiguation*. In: *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, S. 9–16, Trento, Italy, Apr. 2006. The Association for Computer Linguistics.

[14] Collins, M. und Y. Singer: *Unsupervised models for named entity classification*. In: *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, S. 100–110, 1999. http://acl.ldc.upenn.edu/W/W99/W99-0613.pdf, PDF auf CD (Datei Collins1999.pdf).

[15] Cucerzan, S. und D. Yarowsky: *Language independent named entity recognition combining morphological and contextual evidence*. In: *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, S. 90–99, Washington, D.C., 1999. http://www.aclweb.org/anthology-new/W/W99/W99-0612.pdf, PDF auf CD (Datei Cucerzan1999.pdf).

[16] DiCiuccio, R.: *Entity extraction & content api evaluation*. Website, 2010. http://blog.viewchange.org/2010/05/entity-extraction-content-api-evaluation/, Kopie auf CD (Datei DiCiuccio2010.pdf).

[17] Dotsika, F.: *Semantic APIs: Scaling up towards the Semantic Web*. International Journal of Information Management, 30(4):335–342, Aug. 2010.

[18] Elsner, M., E. Charniak und M. Johnson: *Structured generative models for unsupervised named-entity clustering*. In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, S.

164–172, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[19] Etzioni, O., M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. Weld und A. Yates: *Unsupervised named-entity extraction from the web: An experimental study.* Artificial Intelligence, 165(1):91–134, Juni 2005.

[20] Gabrilovich, E. und S. Markovitch: *Computing semantic relatedness using wikipedia-based explicit semantic analysis.* In: *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, S. 6–12, San Francisco, CA, USA, Jan. 2007. Morgan Kaufmann Publishers Inc.

[21] Golder, S. A. und B. A. Huberman: *Usage patterns of collaborative tagging systems.* Journal of Information Science, 32(2):198–208, Apr. 2006.

[22] Grishman, R.: *The NYU system for MUC-6 or where's the syntax?* In: *Proceedings of the 6th conference on Message understanding*, S. 167–175, Stroudsburg, PA, USA, 1995. Association for Computational Linguistics.

[23] Grishman, R. und B. Sundheim: *Message Understanding Conference-6: a brief history*, S. 466–471. Association for Computational Linguistics, 1996.

[24] Hearst, M.: *Automatic acquisition of hyponyms from large text corpora.* In: *Proceedings of the 14th conference on Computational linguistics*, S. 539–545, Stroudsburg, PA, USA, Juli 1992. Association for Computational Linguistics.

[25] Hitzler, P., M. Krötzsch, S. Rudolph und Y. Sure: *Semantic Web: Grundlagen (eXamen.press).* Springer-Verlag, Heidelberg, Okt. 2007.

[26] Horridge, M., H. Knublauch, A. Rector, R. Stevens und C. Wroe: *A Practical Guide To Building OWL Ontologies Using Protege 4 and CO-ODE Tools Edition 1.1*, Okt. 2007. http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_1.pdf, PDF auf CD (Datei Horridge2007.pdf).

[27] Isozaki, H. und H. Kazawa: *Efficient support vector classifiers for named entity recognition.* In: *Proceedings of the 19th international conference on Computational linguistics*, S. 1–7, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[28] Jackson, P. und I. Moulinier: *Natural language processing for online applications: Text retrieval, extraction and categorization.* John Benjamins Pub Co, Amsterdam, 2007.

[29] Kazama, J. und K. Torisawa: *Exploiting Wikipedia as External Knowledge for Named Entity Recognition.* In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, S. 698–707, Prague, Juni 2007. Association for Computational Linguistics.

[30] Lafferty, J., A. McCallum und F. Pereira: *Conditional random fields: Probabilistic models for segmenting and labeling sequence data.* In: *Proceedings of the 18th International Conference on Machine Learning*, S. 282–289, San Francisco, 2001. Morgan Kaufmann.

[31] Lin, D.: *Automatic retrieval and clustering of similar words.* In: *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, S. 768–774, Stroudsburg, PA, USA, 1998. Association for Computational Linguistics.

[32] Liu, B.: *Sentiment analysis and subjectivity.* In: Indurkhya, N. und F. J. Damerau (Hrsg.): *Handbook of Natural Language Processing, Second Edition.* CRC Press, Taylor and Francis Group, Boca Raton, FL, 2010.

[33] Manning, C. D., P. Raghavan und H. Schütze: *Introduction to Information Retrieval.* Cambridge University Press, Juli 2008.

[34] Mayfield, J., P. McNamee und C. Piatko: *Named entity recognition using hundreds of thousands of features.* In: *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, S. 184–187, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[35] McCallum, A. und W. Li: *Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons.* In: *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, S. 188–191, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[36] McDonald, D. D.: *Internal and external evidence in the identification and semantic categorization of proper names*, S. 21–39. MIT Press, Cambridge, MA, USA, 1996.

[37] Medelyan, O., E. Frank und I. H. Witten: *Human-competitive tagging using automatic keyphrase extraction.* In: *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, S. 1318–1327, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[38] Medelyan, O., D. Milne, C. Legg und I. H. Witten: *Mining meaning from Wikipedia*. International Journal of Human-Computer Studies, 67:716–754, Sep. 2009.

[39] Medelyan, O., I. Witten und D. Milne: *Topic indexing with Wikipedia*. In: *Proceedings of AAAI Workshop on Wikipedia and Artificial Intelligence*, S. 19–24, Chicago, USA, Juli 2008. AAAI Press.

[40] Mihalcea, R. und A. Csomai: *Wikify!: linking documents to encyclopedic knowledge*. In: *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, S. 233–242, New York, NY, USA, 2007. ACM.

[41] Mikheev, A.: *A knowledge-free method for capitalized word disambiguation*. In: *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, S. 159–166, Morristown, NJ, USA, 1999. Association for Computational Linguistics.

[42] Mikheev, A., C. Grover und M. Moens: *Description of the LTG system used for MUC-7*. In: *In Proceedings of 7th Message Understanding Conference*, S. 1–12, 1998. http://www-nlpir.nist.gov/related_projects/muc/proceedings/muc_7_proceedings/ltg_muc7.pdf, PDF auf CD (Datei Mikheev1998.pdf).

[43] Milne, D., O. Medelyan und I. H. Witten: *Mining Domain-Specific Thesauri from Wikipedia: A Case Study*. In: *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, S. 442–448, Washington, DC, USA, 2006. IEEE Computer Society.

[44] Milne, D. und I. H. Witten: *Learning to link with wikipedia*. In: *Proceeding of the 17th ACM conference on Information and knowledge management*, S. 509–518, New York, NY, USA, 2008. ACM.

[45] Mishne, G.: *AutoTag: a collaborative approach to automated tag assignment for weblog posts*. In: *Proceedings of the 15th international conference on World Wide Web*, S. 953–954, New York, NY, USA, 2006. ACM.

[46] Nadeau, D. und S. Sekine: *A survey of named entity recognition and classification*. Linguisticae Investigationes, 30:3–26, Jan. 2007.

[47] Neumann, G.: *Informationsextraktion*. In: Carstensen, K. U., C. Ebert, E. Endriss, S. Jekat, R. Klabunde und H. Langer (Hrsg.): *Computerlinguistik und Sprachtechnologie*, S. 448–456. Spektrum, Heidelberg, Berlin, 2001.

[48] Nothman, J., J. Curran und T. Murphy: *Transforming Wikipedia into named entity training data.* In: *Proceedings of the Australasian Language Technology Association Workshop*, S. 124–132, 2008.

[49] Paşca, M., D. Lin, J. Bigham, A. Lifchits und A. Jain: *Organizing and Searching the World Wide Web of Facts - Step One: The One-Million Fact Extraction Challenge.* In: *Proceedings of the National Conference on Artificial Intelligence*, S. 1400–1405. AAAI Press, 2006.

[50] Paola, S. D. und G. Fedon: *Subverting Ajax.* In: *23rd Chaos Computer Club Conference*, Dez. 2006.

[51] Porter, M. F.: *An algorithm for suffix stripping.* Program, 14(3):130–137, Juli 1980.

[52] Ratnaparkhi, A.: *A simple introduction to maximum entropy models for natural language processing.* Techn. Ber., Institute for Research in Cognitive Science, University of Pennsylvania, 1997.

[53] Riloff, E. und R. Jones: *Learning dictionaries for information extraction by multi-level bootstrapping.* In: *Proceedings of the National Conference on Artificial Intelligence*, S. 474–479, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.

[54] Rössler, M.: *Korpus-adaptive Eigennamenerkennung.* Dissertation, Universität Duisburg-Essen, Winterthur, Dez. 2007.

[55] Schonhofen, P.: *Identifying document topics using the wikipedia category network.* In: *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, S. 456–462, Washington, DC, USA, 2006. IEEE Computer Society.

[56] Smith, L., L. Tanabe, R. Ando, C. J. Kuo, F. I. Chung, C. N. Hsu, Y. S. Lin, R. Klinger, C. Friedrich, K. Ganchev, M. Torii, H. Liu, B. Haddow, C. Struble, R. Povinelli, A. Vlachos, W. Baumgartner, L. Hunter, B. Carpenter, R. Tsai, H. J. Dai, F. Liu, Y. Chen, C. Sun, S. Katrenko, P. Adriaans, C. Blaschke, R. Torres, M. Neves, P. Nakov, A. Divoli, M. M. Lopez, J. Mata und J. W. Wilbur: *Overview of BioCreative II gene mention recognition.* Genome Biology, 9(Suppl 2):S2, 2008.

[57] Smith, S.: *Dailyme: Seeing how you are being seen.* Website, 2009. http://www.mediapost.com/publications/?fa=Articles.showArticle&art_aid=111695, Kopie auf CD (Datei Smith2009.pdf).

[58] Takeuchi, K. und N. Collier: *Use of support vector machines in extended named entity recognition.* In: *Proceedings of the 6th conference on Natural language learning*, S. 1–7, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[59] Toral, A. und R. Munoz: *A proposal to automatically build and maintain gazetteers for Named Entity Recognition by using Wikipedia.* In: *Workshop On New Text Wikis And Blogs And Other Dynamic Text Sources, 11th Conference of the European Chapter of the Association for Computational Linguistics*, S. 56–61, Trento (Italy), Apr. 2006.

[60] Turney, P. D.: *Mining the web for synonyms: Pmi-ir versus lsa on toefl.* In: *Proceedings of the 12th European Conference on Machine Learning*, S. 491–502, London, UK, 2001. Springer-Verlag.

[61] Witten, I. und D. Milne: *An effective, low-cost measure of semantic relatedness obtained from Wikipedia links.* In: *Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy*, S. 25–30, Chicago, USA, Juli 2008. AAAI Press.

[62] Yeh, A., A. Morgan, M. Colosimo und L. Hirschman: *Biocreative task 1a: gene mention finding evaluation.* BMC Bioinformatics, 6(Suppl 1):S2, 2005.

[63] Zaino, J.: *Dailyme boosts personalization with opencalais.* Website, 2009. http://semanticweb.com/dailyme-boosts-personalization-with-opencalais_b424, Kopie auf CD (Datei Zaino2009.pdf).

[64] Zhou, G. und J. Su: *Named entity recognition using an HMM-based chunk tagger.* In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, S. 473–480, Morristown, NJ, USA, 2002. Association for Computational Linguistics.