Informationsempfehlungen durch Kollaboration und Textanalyse

Magdalena Eibensteiner

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im September 2013

© Copyright 2013 Magdalena Eibensteiner

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

Ich erkläre eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Hagenberg, am 30. September 2013

Magdalena Eibensteiner

Inhaltsverzeichnis

Erklärung **iii**

Kurzfassung **vi**

Abstract **vii**

1 Einleitung **1**
1.1 Zielsetzung 2
1.2 Inhaltlicher Aufbau 2

2 Grundlagen **4**
2.1 Ähnlichkeitstheorien 4
2.1.1 Klassifizierung der Ähnlichkeitsverfahren 4
2.1.2 Messen von Ähnlichkeit 5
2.1.3 Vektorbasierte Ähnlichkeitsmaße 9
2.2 Semantic APIs 13
2.2.1 OpenCalais API 13
2.2.2 Zemanta API 14
2.2.3 Alchemy API 15
2.3 Kollaborative Daten 17
2.3.1 Kollaboratives Filtern 17
2.3.2 Webanalyse Werkzeuge 18

3 Verwandte Arbeiten **20**

4 Konzeption **28**
4.1 Problem Definition 28
4.2 Einfluss der kollaborativen Daten 29
4.3 Auswahl der Semantic APIs 30
4.4 Bestimmen der Ähnlichkeit 33
4.5 Einbettung in ein Content Management System 35

5 Implementierung **36**
5.1 Drupal Modul Entwicklung 36

5.2 Einbindung der Webanalyse Metriken 37
5.3 Umsetzung der Textanalyse 39
5.4 Generierung der Informationsempfehlungen 40

6 Evaluierung **43**
6.1 Evaluierungsmethode 43
6.1.1 Evaluierungsmaße 44
6.2 Evaluierungsergebnisse 45
6.3 Diskussion 49

7 Zusammenfassung **52**

A Inhalt der CD-ROM **54**
A.1 Masterarbeit (PDF 54
A.2 Projekt 54
A.3 Online-Quellen (PDF) 54
A.4 Evaluierung 54

Quellenverzeichnis **55**
Literatur 55
Online-Quellen 56

Kurzfassung

Der Umfang von Informationen im World Wide Web ist in den letzten Jahren beträchtlich gestiegen und mittlerweile nicht mehr überschaubar. Daher gewinnen Empfehlungssysteme immer mehr an Bedeutung, da diese Benutzern helfen können die Informationen zu finden die sie brauchen. In dieser Arbeit wird ein Ansatz zur Generierung von Informationsempfehlungen vorgestellt der auf kollaborativen und inhaltsbasierten Daten aufbaut. Der Fokus liegt auf der Empfehlung von Beiträgen oder Artikel innerhalb einer Webseite.

Empfehlungssysteme betrachten als Grundlage häufig das Verhalten der Benutzer und versuchen aus den Informationen über die Gesamtheit der Benutzer oder Benutzergruppen, Interessen eines individuellen Benutzers abzuleiten. Solche Systeme können vor allem kurz nach Veröffentlichung zu Problemen führen, da erst wenige Informationen gesammelt wurden (Kaltstartproblem). Eine weitere Möglichkeit Empfehlungen zu generieren, ist die Bewertung der Ähnlichkeit der einzelnen Beiträge basierend auf den Inhalten. Dies ist nicht immer einfach, da Ähnlichkeiten meist auch von semantischen Merkmalen in den Texten abhängen. Diese können von Menschen sehr gut verstanden werden, die computergestützte Interpretation birgt jedoch viele Herausforderungen.

Im Zuge dieser Arbeit werden verschiedene Herangehensweisen an diese Problemstellung betrachtet. Es wird ein Ähnlichkeitsmaß entwickelt, das sowohl inhaltsbasierte Informationen als auch kollaborative Daten miteinbezieht. Dieses Maß wird im Rahmen einer News-Webseite verwendet um Informationsempfehlungen zu generieren. Die abschließende Evaluierung des Maßes zeigt, dass es im Stande ist qualitativ gute Empfehlungen zu geben.

Abstract

The amount of information in the World Wide Web has increased considerably in the last years and is no longer manageable. Therefore, recommendation systems are becoming increasingly important as they can help users to find the information they really need. This work presents an approach to produce information-recommendations based on collaborative and content-based data. It focuses on the recommendation of articles or contributions within one web page.

Recommendation systems often observe the user's behavior and try to derive the individual user's interests by using the information of all users or groups of users. Such systems can cause problems especially shortly after publication, since only few information was collected (so-called cold-start problem). Another way to generate recommendations, is to estimate the similarity of the individual entries based on the contents. This is not always easy, since similarities usually depend on semantic features in the texts. Semantic can be very well understood by people, but the interpretation by computers yields to many challenges.

In the course of this master thesis several approaches are considered to solve this problem. A similarity measure is developed that involves both content-based information, and collaborative data. This measure is used as part of a news website to generate recommendations. The final evaluation of the measure shows that it is able to provide good quality recommendations for users.

Kapitel 1

Einleitung

Das World Wide Web (WWW) bietet eine Fülle an Informationen die jederzeit und überall zugänglich sind. Die Herausforderung liegt schon lange nicht mehr im Finden von Informationen an sich, sondern „richtigen“ beziehungsweise „relevanten“ Informationen in einem bestimmten Kontext. Empfehlungen gewinnen daher immer mehr an Bedeutung, da sie beim effizienten Suchen von bestimmten Informationen helfen können. Nicht nur im gesamten WWW tritt diese Problematik auf sondern auch bereits bei Webauftritten mit sehr umfangreichen Textinhalten zu unterschiedlichen Themengebieten, beispielsweise Online-Zeitungen.

Empfehlungssysteme können Benutzer beim Konsumieren von Informationen unterstützen indem sie Verknüpfungen zwischen verwandten Dokumenten schaffen und diese anzeigen, beispielsweise als Hyperlinks. So können dem Benutzer alle Einträge einer Seite zu einem bestimmten Kontext vorgeschlagen werden, ohne dass dieser explizit danach suchen muss.

Die Verknüpfungen zwischen Dokumenten können durch das Betrachten des Benutzerverhaltens gefunden werden. Man erhebt welche Artikel oder Einträge sequentiell von verschiedenen Benutzern betrachtet werden und schließt dadurch auf Gemeinsamkeiten zwischen den Dokumenten. Wenn ein Benutzer einen bestimmten Artikel liest, können so weitere Einträge vorgeschlagen werden, die von anderen Benutzern gelesen wurden, die auch den aktuell angezeigten Artikel betrachtet haben.

Ein Empfehlungssystem, dass sich rein auf das Benutzerverhalten stützt, führt vor allem kurz nach Veröffentlichung der Webseite zu Schwierigkeiten (Kaltstartproblem), da noch keine oder zu wenige aussagekräftige Daten über gemeinsame Seitenaufrufe durch verschiedene Benutzer vorliegen. Um diesem Problem entgegenzuwirken können zusätzlich zum Benutzerverhalten die Inhalte der Seiten zum Erstellen der Empfehlungen herangezogen werden. Dies kann auch später die Qualität der Empfehlungen steigern und generell als Faktor zur Bestimmung verwandter Dokumente einbezogen werden. Um basierend auf dem Inhalt Ähnlichkeiten zwischen Dokumenten ab-

leiten zu können, müssen die einzelnen Dokumente zuerst analysiert werden. Aufgrund dieser Analyse können die Dokumente miteinander verglichen und ein Ähnlichkeitsmaß gefunden werden, das Informationen über den Grad der Zusammengehörigkeit bietet.

Die Erhebung von Ähnlichkeiten zwischen Texten ist ein wesentlicher Teil der Computer-Linguistik. Dementsprechend gibt es viele verschiedene Vorgehensweisen an dieses Problem heranzugehen. Immer mehr Stellenwert gewinnt die Berücksichtigung von semantischen Informationen in den Texten. Dies bringt jedoch auch viele Herausforderungen mit sich, da die Semantik verschiedener Begriffe und ihre Zusammenhänge zwar von Menschen sehr gut verstanden wird, eine entsprechende maschinelle Interpretation jedoch schwer nachzubilden ist. Anwendungen die sich mit der Semantik von Texten und Worten befassen sind beispielsweise Semantic APIs. Diese dienen zur Analyse von Texten und zur Identifikation semantischer Metadaten, beispielsweise benannte Entitäten, Beziehungen von Begriffen in Texten, der Identifikation von Schlüsselwörtern oder der Zuweisung einer Überkategorie zum Text. Die APIs arbeiten mit Algorithmen aus den Bereichen Natural Language Processing, Machine Learning oder Information Retrieval und greifen auf umfangreiche Informationen aus verschiedenen Datenbanken zurück. Es entstand die Idee diese APIs zu nutzen um Ähnlichkeiten zwischen Textdokumenten einer Webseite zu finden und den Grad der Zusammengehörigkeit zu berechnen.

1.1 Zielsetzung

Es soll ein System entwickelt werden, das basierend auf kollaborativen Daten und Informationen aus der Textanalyse Empfehlungen generiert. Durch die Betrachtung verschiedener Herangehensweisen an dieses Problem, soll ein Ähnlichkeitsmaß gefunden werden, das diese beiden Einflussgrößen kombiniert. Der Fokus liegt auf Webseiten mit umfangreichen Textinhalten die dem Benutzer präsentiert werden, insbesondere Online-Zeitungen. Ausgehend von dem Artikel, der aktuell vom Benutzer gelesen wird, sollen aus der Gesamtheit aller im System vorhandenen Artikel, jene für die Empfehlungen ausgewählt werden, die der Benutzer nach dem aktuellen Beitrag lesen möchte. Durch eine Evaluierung der Ergebnisse soll die Qualität der so entstehenden Empfehlungen, bzw. des Ähnlichkeitsmaßes erhoben werden.

1.2 Inhaltlicher Aufbau

Im nächsten Kapitel wird auf einige grundlegenden Themen näher eingegangen die für den Rest der Arbeit relevant sind, es wird erklärt welche Formen von Ähnlichkeiten zwischen Texten gemessen werden können und es werden einige Ähnlichkeitsmaße beschrieben die Einblick in die Berechnung von

Ähnlichkeiten geben bzw. als Basis für komplexere Vorgehensweisen dienen. Des Weiteren wird näher auf Semantic APIs eingegangen und beschrieben welche Daten von diesen abgerufen werden können. Auch verschiedene Webanalyse Metriken werden angeführt und erklärt. Andere Projekte und Methoden im Gebiet Ähnlichkeitsberechnung von Texten werden im Kapitel 3 Verwandte Arbeiten angeführt. Im darauf folgenden Kapitel 4 Konzeption wird näher auf die Grundidee des umgesetzten Systems und dessen Methoden eingegangen. Die eigentliche Umsetzung des Systems wird im Kapitel 5 Implementierung beschrieben. Das System wird anhand verschiedener Evaluationsmetriken getestet, eine Beschreibung der verwendeten Maße und die Ergebnisse werden im Kapitel 6 Evaluierung besprochen.

Kapitel 2

Grundlagen

In diesem Kapitel werden einige essenzielle Themen näher erläutert die für die folgende Arbeit wichtig sind. Zuerst wird näher auf das Thema Ähnlichkeiten zwischen Texten eingegangen, dazu wird eine Klassifizierung der Maße erläutert, die sich auf unterschiedliche Merkmale von Texten bezieht. Außerdem werden grundlegende Ansätze zur Ermittlung der Ähnlichkeit beschrieben. Es wird geklärt was Semantic APIs sind, einige Beispiele und Metadaten die sie liefern werden angeführt. Abschließend wird ein Überblick über Webanalysemetriken gegeben.

2.1 Ähnlichkeitstheorien

Das automatisierte Messen von Ähnlichkeiten ist ein wichtiger Bestandteil in vielen computergestützten Textverarbeitungsaufgaben. Beispielsweise im Bereich des Information Retrievals (IR) wo ein Anfragestring mit einer Vielzahl von zur Verfügung stehenden Dokumenten verglichen wird, ein Vorgehen das bei Suchmaschinen Anwendung findet. Auch beim Klassifizieren oder Klustern von Dokumenten oder beim automatisierten Erstellen von Zusammenfassungen oder Übersetzungen spielt die Ähnlichkeit zwischen Zeichenketten eine große Rolle. Um Dokumente als verwandt bezeichnen zu können muss ein Wert bestimmt werden der Aufschluss darüber gibt, welche Ähnlichkeiten bzw. Unterschiede zwischen zwei Texten bestehen. Dies passiert dadurch, dass man nach gleichen Eigenschaften und Merkmalen in Texten sucht.

2.1.1 Klassifizierung der Ähnlichkeitsverfahren

In der Linguistik unterscheidet man drei Hauptmerkmale von geschriebenen Texten, die auch für das computergestützte Messen von Ähnlichkeiten herangezogen werden können, syntaktische, lexikalische und semantische Merkmale [9]. Diese drei Charakteristiken werden auch für eine grobe Klassifizierung verschiedener Verfahren zur Messung von Ähnlichkeit verwendet.

Die Syntaktik bezieht sich auf den Aufbau und die Organisation von korrekten Sätzen. Hauptsächlich wird diese durch grammatikalische Regeln bestimmt aber auch durch die strukturellen Beziehungen zwischen Wörtern, beispielsweise wie ein Adjektiv ein folgendes Nomen beschreibt. Die syntaktische Ähnlichkeit kann hier durch die Länge einzelner Wörter oder Sätze oder auch durch das Vorhandensein oder Fehlen bestimmter Wörter erkannt werden.

Wortschatz, thematische Zusammenhänge zwischen Wörtern und eine hierarchische Beziehung zwischen Oberbegriffe und Unterbegriffe werden als lexikalische Merkmale bezeichnet. Treten in einem Dokument beispielsweise die Begriffe „fahren“, „Reifen“ und „Motor“ auf, kann man daraus auf den Oberbegriff Automobil schließen.

Die Semantik befasst sich mit der Bedeutung von Wörtern und Wortgruppen. Eine sehr große Rolle spielt hier die menschliche Interpretationsfähigkeit, daher sind semantische Merkmale für die computergestützte Verarbeitung aufwändiger und komplexer zu erfassen als syntaktische oder lexikalische Eigenschaften. Ein Beispiel um diese Problematik näher zu erklären:

Frage: „Leihst du mir einen Stift?“

Antwort: „Meine Bürotür ist offen.“

Ein Mensch kann diese Antwort als positiv, also als Erlaubnis sich den Stift zu leihen interpretieren. Die Antwort enthält jedoch kein Wort, das auch von einem Computer als explizit positiv interpretiert werden könnte.

Je nach Komplexität und Problemstellung einer automatisierte Textverarbeitungsaufgabe werden Eigenschaften aus einer oder mehreren der linguistischen Klassen herangezogen.

2.1.2 Messen von Ähnlichkeit

Es gibt unzählige Herangehensweisen an die Berechnung von Ähnlichkeit zwischen Texten, hier sollen einige elementare Verfahren erklärt werden die schon früh entwickelt wurden jedoch immer noch eingesetzt werden, vor allem aber als Basis für weitere Entwicklungen dienen.

Elementare Verfahren

Sehr grundlegende Maße betrachten die Texte meist als Mengen von Wörtern oder noch kleineren Einheiten von n-Grammen ($n > 1$). Hauptsächlich werden syntaktischen Merkmalen betrachtet.

Jaccard-Index: Bereits 1901 entwickelte der Botaniker Paul Jaccard den Jaccard-Index, er dient zur Berechnung der Ähnlichkeit von Mengen. Der Ähnlichkeitswert entspricht der Anzahl von gemeinsamen Elementen, geteilt durch die Größe der Vereinigungsmenge [14]. Verwendet wird dieses Maß bis

heute, bispielsweise in der Duplikatenerkennung. Der Jaccard-Index für zwei Mengen A und B wird wie folgt berechnet:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}. \tag{2.1}$$

Um diese Methode bei Texten anzuwenden werden diese zuerst in einzelne Worte oder n-Gramme zerlegt. Betrachtet man, beispielsweise folgende Texte als Menge von einzelnen Wörtern:

a: Anna spricht mit Karl.

b: Anna spricht nicht mit Max.

ergeben sich die Mengen $\{Anna, spricht, mit, Karl\}$ und $\{Anna, spricht, nicht, mit, Max\}$. Ihr Jaccard-Index wird mithilfe von Gl.2.1 folgendermaßen berechnet:

$$\frac{|\{Anna, spricht, mit\}|}{|\{Anna, spricht, mit, Karl, nicht, Max\}|} = \frac{3}{6} = 0,5.$$

Levenshtein-Distanz: Dieses Maß wird auch Editierdistanz genannt und ist ebenfalls eine sehr frühe Entwicklung im Bereich der Ähnlichkeitsberechnung. Hier werden die Zeichen gezählt die geändert werden müssen um eine Zeichenkette in eine andere umzuwandeln. Die Anzahl der Änderungen ergeben das Ähnlichkeitswert. Erlaubte Operationen sind Einfügen, Löschen und Ersetzen von Zeichen [16]. Dieses Maß wird beispielsweise für Autokorrektur Mechanismen verwendet. Ein sehr einfaches Beispiel zur Levenshtein-Distanz ist die Betrachtung der Begriffe „Tier“ und „Tor“. Der Wert beträgt hier 2, eine mögliche Vorgehensweise wäre:

1. Tier
2. Toer (ersetze i durch o)
3. Tor (lösche e).

Vektorbasierte Verfahren

Alternativ zur Auffassung und Verarbeitung der Texte als Mengen werden diese auch oft als Vektoren dargestellt. Werden die einzelnen Dokumente als reelle Vektoren aus demselben Vektorraum repräsentiert, können Methoden aus der Vektorraum Theorie angewendet werden um sie zu vergleichen [7].

Bag-of-words Modell: Das Bag-of-words Modell [11] ist ein sehr elementarer vektorbasierter Ansatz. Es wird ein Verzeichnis aller Wörter die in den betrachteten Texten auftreten angelegt. Mithilfe dieses Verzeichnisses und den Worthäufigkeiten wird für jeden Text ein Vektor gebildet. Nachteil des Bag-of-words Modells, ist das Reihenfolge und Zusammenhänge der Wörter ignoriert werden. Das Vorgehen basiert auf der Annahme je häufiger ein

Wort in einem Dokument vorkommt, desto wichtiger ist es für die Beschreibung des Dokumenteninhalts und daher auch für die Ähnlichkeit zu anderen Dokumenten. Betrachtet man beispielsweise diese Texte:

a: Anna spricht mit Karl, aber nicht mit Max.

b: Max redet oft mit Karl.

würde sich ein entsprechendes Verzeichnis folgendermaßen gestalten:

1 : „Anna“,
2 : „spricht“,
3 : „mit“,
4 : „Karl“,
5 : „aber“,
6 : „nicht“,
7 : „Max“,
8 : „redet“,
9 : „oft“

Für jeden der beiden Texte wird ein Vektor erstellt in dem die Häufigkeiten der Wörter eingetragen werden. Die Reihenfolge der Wörter im Verzeichnis bestimmt auch die Reihenfolge im Vektor, jeder Vektor enthält so viele Einträge wie das Verzeichnis, hier 9:

$\vec{a} = (1, 1, 2, 1, 1, 1, 1, 0, 0)$
$\vec{b} = (0, 0, 1, 1, 0, 0, 1, 1, 1).$

Vektorraummodell: Das Vektorraummodell ist eine maßgebende Form der vektorbasierten Ansätze und wurde von Salton [18] eingeführt. Bei dieser Methode werden die Dokumente in einem mehrdimensionalen Vektorraum dargestellt. Die Position eines Dokuments in diesem Raum wird durch einen oder mehrerer Indexterme definiert. Für jedes Dokument entsteht ein t-dimensionaler Vektor, wenn t Terme indexiert werden (s. Abb. 2.1). Die Ähnlichkeit von zwei Dokumentenvektoren $s(D_i, D_j)$ kann durch die mathematische Distanz dieser Vektoren erhoben werden.

Salton sieht vor, dass die einzelnen Terme bei der Verwendung des Vektorraummodells gewichtet werden, also ein Wert miteinbezogen wird, der widerspiegelt wie relevant ein Term für die Beschreibung des Inhalts des Dokuments ist. Dieser Wert soll zwischen 0, nicht relevant und 1, sehr relevant liegen. Die Gewichtung kann auch ignoriert werden indem alle Terme mit dem Standardgewicht 0 bewertet werden [18].

Abb. 2.2 zeigt die schematische Darstellung eines Systems, dass auf dem Vektorraummodel aufbaut. Es handelt sich um ein System aus dem Bereich Information-Retrieval, hier werden hauptsächlich Suchanfragen eines Benutzers mit zur Verfügung stehenden Dokumenten abgeglichen. Die Vektoren

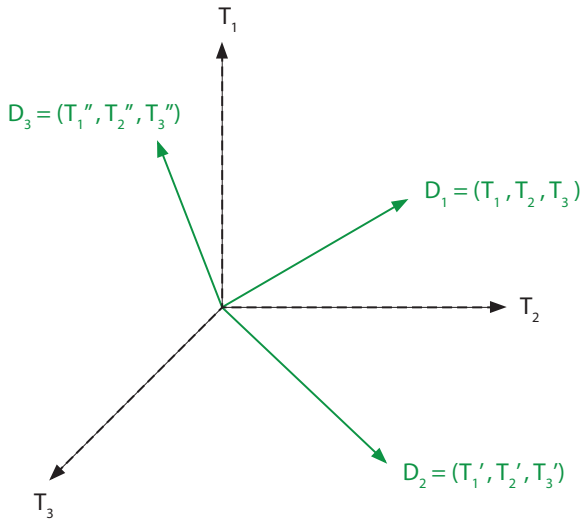


Abbildung 2.1: Darstellung eines dreidimensionalen Vektors im Dokumenten-Vektorraum [18].

beschreiben den Text durch reelwertige Einträge, jeder Eintrag im Vektor entspricht dem Gewicht des entsprechenden Terms im Dokument [7].

Korpusbasierte Verfahren

Vor allem zur Berechnung von semantischer Ähnlichkeit ist Wissen über den natürlichen Sprachgebrauch notwendig, beziehungsweise Informationen darüber wie die Bedeutung von Wörtern durch Menschen interpretiert wird. Deshalb werden Ansätze immer populärer die versuchen aus großen Textsammlungen, lexikalische aber auch semantische Attribute zu gewinnen und diese zur Berechnung der Ähnlichkeit heranzuziehen.

Einer repräsentativen Menge von Dokumenten werden sogenannte Kookurrenzdaten entnommen. Diese sagen aus wie oft verschiedene Terme in Dokumenten gemeinsam auftreten. Um Aufschluss darüber zu erhalten ob das gemeinsame Auftreten von Termen zufällig ist, muss die Häufigkeit der gemeinsamen Auftreten mit der gesamten Anzahl von Dokumenten in denen einer der Terme vorkommt, in Beziehung gesetzt werden. Zur Erhebung und Verarbeitung der Daten gibt wiederum verschiedene Ansätze, es werden beispielsweise Matrizen verwendet um die Beziehungen der Terme abzubilden. Auch verschiedene Analyseverfahren und Machine-Learning Ansätze und werden eingesetzt [7].

Den korpusbasierten Verfahren liegt die *Assoziationstheorie* zugrunde, diese besagt, dass zwischen Objekten die einmal gemeinsam vorkommen Assoziationen gebildet werden, es also sehr wahrscheinlich ist, dass beim Auftreten eines Objekts auch das andere auftritt. Bei den korpusbasierten Verfah-

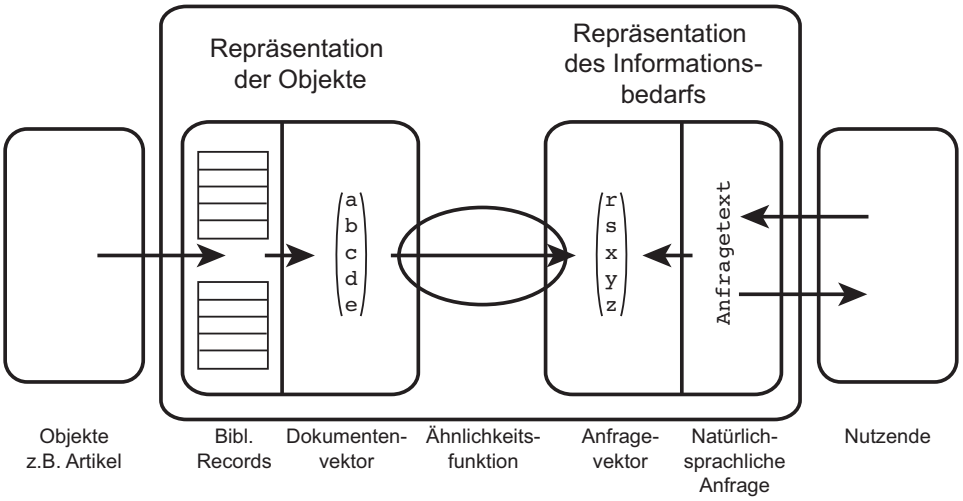


Abbildung 2.2: Schematische Darstellung eines Vektorraum-Text-Retrieval Systems [7].

ren versucht man also aus großen Textsammlungen Assoziationen zwischen einzelnen Termen zu berechnen [7].

Als solche Textsammlung kann beispielsweise *WordNet* dienen. Eine frei zugängliche Datenbank in der lexikalische und semantische Beziehungen der englischen Sprache gespeichert sind. *WordNet* enthält Informationen über Nomen, Verben, Adjektive und Adverben, deren hierarchische Beziehungen und Synonyme sowie semantische Zusammenhänge zwischen einzelnen Wörtern. Es gibt auch eine ähnliche deutschsprachige Datenbank *GermaNet* diese ist jedoch nicht frei verfügbar. Weniger umfangreiche Informationen werden von *OpenThesaurus* zur Verfügung gestellt, dieses Projekt ist in mehreren Sprachen verfügbar und frei nutzbar.

Eine weitere Dokumentensammlung die für das Messen von Ähnlichkeiten verwendet wird ist *Wikipedia*.[1] *Wikipedia* ist in den letzten Jahren zu einer der größten Online Wissenssammlungen gewachsen und wird laufend durch freiwillige Autoren erweitert und überarbeitet, wodurch die Einträge sehr aktuell sind. Ein großer Vorteil ist, dass *Wikipedia* in vielen Sprachen verfügbar ist und das enthaltene Wissen sehr breit gefächert ist. Außerdem enthält *Wikipedia* bereits Informationen über hierarchische Beziehungen zwischen Wörtern, Synonyme und Mehrdeutigkeiten.

2.1.3 Vektorbasierte Ähnlichkeitsmaße

Durch die Repräsentation der Dokumente als Vektoren kann ihre Ähnlichkeit durch Methoden aus der Vektorraumtheorie ermittelt werden. Einige elemen-

[1] http://de.wikipedia.org/ – abgerufen am 14. Mai 2013

tare Ähnlichkeitsmaße werden in [15] beschrieben und theoretisch verglichen.

Cosinus-Maß: Beim Cosinus-Maß hat die Länge der Vektoren keinen direkten Einfluss auf die Ähnlichkeit. Die Ähnlichkeit wird nur durch den eingeschlossenen Winkel der Vektoren definiert und es können Ähnlichkeitswerte zwischen -1 und 1 erzielt werden. Die Ähnlichkeit beruht hier auf der Beziehung der Vektoreinträge zueinander, respektive ist sie von der Richtung der Vektoren abhängig. Das Maß kann auch als Skalarprodukt der normierten Vektoren angesehen werden. Für zwei Vektoren $\vec{a}$ und $\vec{b}$ wird das Cosinus-Maß durch folgende Formel berechnet:

$$sim_{\cos}(\vec{a}, \vec{b}) = \frac{\sum_{i=1}^{N} a_i \cdot b_i}{\sqrt{\sum_{i=1}^{N} a_i^2} \cdot \sqrt{\sum_{i=1}^{N} b_i^2}}. \tag{2.2}$$

Pseudo-Cosinus-Maß: Beim Pseudo-Cosinus-Maß werden die Vektoren nicht durch die euklidische Länge normiert sondern durch die Summe der Vektoreinträge. Sind alle Einträge der Vektoren positiv, werden vor allem große Werte im Vektor weniger abgeschwächt. Die Ähnlichkeitswerte liegen zwischen 0 und 1. Die Formel zur Berechnung dieses Maßes ist wie folgt definiert,

$$sim_{pseudo}(\vec{a}, \vec{b}) = \frac{\sum_{i=1}^{N} a_i \cdot b_i}{\left(\sum_{i=1}^{N} a_i\right) \cdot \left(\sum_{i=1}^{N} b_i\right)}. \tag{2.3}$$

Dice-Maß: Das Dice-Maß lässt sich nicht mehr als normiertes Skalarprodukt beschreiben, hier wird durch die Summe aller Einträge beider Vektoren geteilt,

$$sim_{dice}(\vec{a}, \vec{b}) = \frac{2 \cdot \sum_{i=1}^{N} a_i \cdot b_i}{\left(\sum_{i=1}^{N} a_i\right) + \left(\sum_{i=1}^{N} b_i\right)}. \tag{2.4}$$

Jaccard-Maß: Dieses Maß erinnert an den Jaccard-Index (s. Abschn.2.1.2) und die dort betrachtete Schnittmenge und Vereinigungsmenge der Texte und wird in Form von

$$sim_{jaccard}(\vec{a}, \vec{b}) = \frac{\sum_{i=1}^{N} a_i \cdot b_i}{\left(\sum_{i=1}^{N} a_i\right) + \left(\sum_{i=1}^{N} b_i\right) - \left(\sum_{i=1}^{N} a_i \cdot b_i\right)} \tag{2.5}$$

berechnet. In [7] wird beschrieben, dass dieses Maß nur für Vektoren mit Werten von 0 oder 1 zuverlässige Ergebnisse bringt, beipsielsweise kann so die Anzahl einzelner Terme in einem Dokument wiedergegeben werden. Beinhalten die Vektoren beliebige reelle Zahlen, kann eine kleine Änderung des Vektors eine beliebig große Auswirkung auf die Ähnlichkeit haben.

Overlap-Maß: Durch das Overlap-Maß wird die minimale Übereinstimmung von zwei Vektoren ermittelt. Die Division durch das Minimum der Summen führt dazu, das die Ähnlichkeiten nicht mehr durch Geraden, sondern durch komplexe Flächen repräsentiert werden, diese können durch Fallunterscheidungen bestimmt werden. Die Formel zur Berechnung dieses Maß ist folgendermaßen definiert,

$$sim_{overlap}(\vec{a}, \vec{b}) = \frac{\sum_{i=1}^{N} \min(a_i, b_i)}{\min\left(\sum_{i=1}^{N} a_i, \sum_{i=1}^{N} b_i\right)}. \tag{2.6}$$

In [15] wird die Ausprägung der Ähnlichkeit der verschiedenen Maße im zweidimensionalen Raum dargestellt (s. Abb. 2.3), dadurch werden die Auswirkungen besser vorstellbar.

Gewichtung von Termen

Wie bereits beim Vektorraummodell erwähnt können Terme gewichtet werden um ihre Relevanz für ein Dokument auszudrücken. In vielen Verfahren zur Berechnung der Ähnlichkeit von Texten, vor allem vektorbasierte Ansätze werden Gewichtungen verwendet.

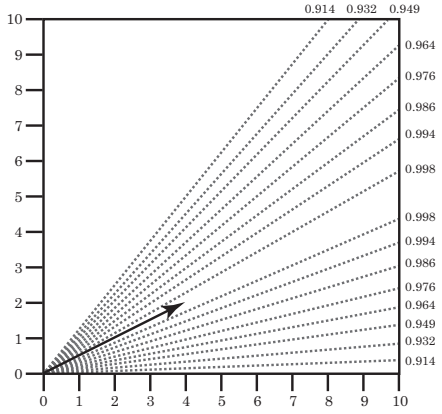
Eine sehr reduzierte Methode der Gewichtung ist die Verwendung von Stoppwortlisten. Dabei werden Wörter die im Sprachgebrauch sehr häufig verwendet werden, beispielsweise bestimmte oder unbestimmte Artikel, Präpositionen, Konjunktionen usw. mit 0 gewichtet. Durch Stoppwortlisten kann jedoch nur die „Unwichtigkeit“ eines Terms definiert werden, es können keine Listen über generell wichtige Terme angelegt werden. Stoppwortlisten werden häufiger verwendet um diese Terme von vornherein aus den Texten zu entfernt, bevor eine weitere Verarbeitung stattfindet, als zur Gewichtung [7].

Sehr häufig wird das Term Frequency - Inverse Document Frequency (TF-IDF) Maß zur Gewichtung von Termen eingesetzt. Diese Gewichtung berücksichtigt sowohl kontextabhängige oder lokale Einflussfaktoren durch die Termhäufigkeit (TF), als auch kontextunabhängige oder globale Einflussfaktoren durch die inverse Dokumentenhäufigkeit (IDF) und kombiniert beide [7].

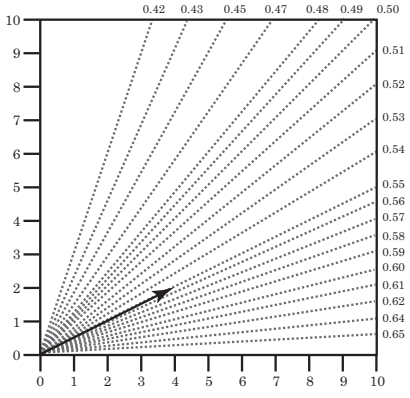
Die Termhäufigkeit $TF_{i,j}$ gibt an, wie oft das Wort j in einem bestimmten Dokument i enthalten ist. Besonders in langen Dokumenten ist die Termhäufigkeit allgemein höher, dies kann ausgeglichen werden indem man die Termhäufigkeit eines Wortes zu der Häufigkeit des im Text am öftesten vorkommenden Terms in Beziehung setzt, dies passiert in der Form

$$TF_{i,j} = \frac{h(i,j)}{\max_{k \in \{1,\dots n\}} h(i,k)}. \tag{2.7}$$

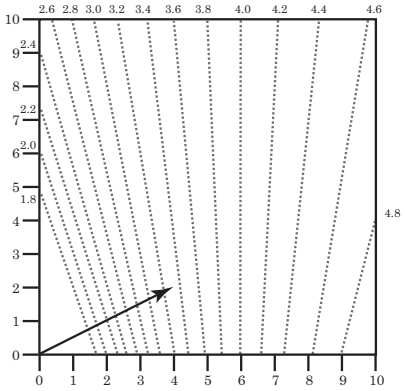
Die inverse Dokumentenhäufigkeit IDF_j erfasst die Bedeutung des Wortes für alle betrachteten Dokumenten $|D|$, wobei $d(j)$ die Anzahl der Dokumente



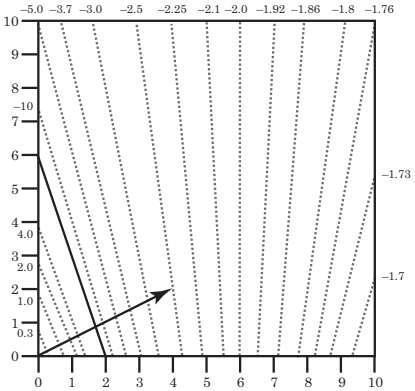
(a) Cosinus-Maß



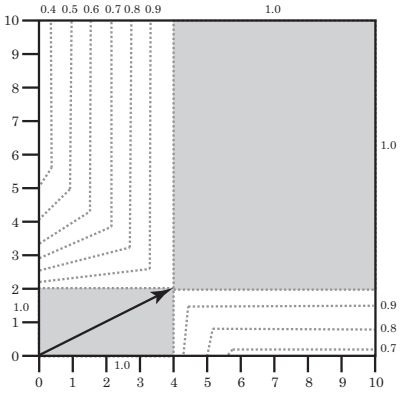
(b) Pseudo-Cosinus-Maß



(c) Dice-Maß



(d) Jaccard-Maß



(e) Overlap-Maß

Abbildung 2.3: Darstellung der einzelnen Ähnlichkeitsmaße im zweidimensionalen Raum für einen Beispielvektor $(4, 2)$. Die gestrichelten Linien bzw. Flächen zeigen die Ausprägung der Ähnlichkeit [15].

in der das Wort j vorkommt bezeichnet. Durch die Anwendung des Logarithmus werden die Gewichtungen seltener Terme abgeschwächt,

$$IDF_j = \log \frac{|D|}{d(j)}. \tag{2.8}$$

Der TF-IDF Wert ist hoch wenn ein Term häufig, in wenigen Dokumenten enthalten ist, die Kombination der Werter erfolgt durch die Formel

$$TFIDF_{i,j} = TF_{i,j} \cdot IDF_i. \tag{2.9}$$

2.2 Semantic APIs

In [6] wird ein Semantic Applications Programmers Interface (API) wie folgt definiert:

> "APIs that take unstructured text (including web pages) as input and return the content's contextual framework are termed semantic APIs." [6, S. 337]

Diese Dienste basieren auf der Idee des Semantic Web, die 2001 von Tim Berners-Lee ins Leben gerufen wurde. Dabei werden Webinhalte so aufbereitet, dass die semantische Bedeutung der Texte nicht nur von Menschen erkannt werden kann, sondern auch von Maschinen interpretierbar ist [2]. Semantic APIs bieten die Möglichkeit unstrukturierten Text mit entsprechender semantischer Bedeutung anzureichern und verschiedene Merkmale in Texten zu identifizieren (s. Abb. 2.4).

Die APIs verwenden zur Textanalyse Algorithmen, beispielsweise aus den Bereichen Natural Language Processing, Machine Learning oder Information Retrieval. Die verschiedenen APIs bieten Webservices zur Erfassung unterschiedliche Arten von Metadaten an, können jedoch derzeit oft nur in englischer Sprache uneingeschränkt genutzt werden [6].

Es folgt eine Beschreibung von drei renommierten Semantic APIs, diese bieten Methoden zur Verschlagwortung und Kategorisierung von Texten an. Die angeführten Informationen entstammen den Webseiten der Anbieter.

2.2.1 OpenCalais API

Eine sehr bekannte Semantic API ist *OpenCalais*[2] von Thomson Reuters mit der Intention, Inhalte besser zugänglich, kompatibel und dadurch vielseitiger nutzbar zu machen. Der *OpenCalais* Webservice bietet Nutzern die Möglichkeit Inhalte automatisch mit semantischen Metadaten zu annotieren. Derzeit kann der Webservice mit englischen, französischen und spanischen Texten genutzt werden. Die Metadaten können vier verschiedenen Hauptkategorien zugeordnet werden [24]:

[2] http://www.opencalais.com/ – abgerufen am 14. Mai 2013

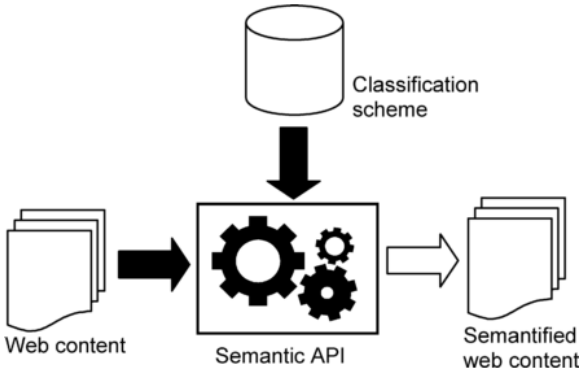


Abbildung 2.4: Schematische Darstellung von Semantic APIs [6].

Topic: Der Text wird einer oder mehreren Kategorien zugewiesen, die beschreiben welches Thema der Text behandelt. Dazu verwendet *OpenCalais* eine breitgefächerte Kategorien-Taxonomie, mit Begriffen wie *Business-Finance, Education, Entertainment-Culture, Law-Crime, etc.*

Social Tags: Durch die Social Tags versucht *OpenCalais* Schlagwörter zu finden die den analysierten Text beschreiben. Hierbei werden nicht nur Wörter aus dem Text extrahiert sondern auch Zusammenhänge und Überbegriffe betrachtet und so nachgeahmt welche Wörter ein Mensch für die Verschlagwortung des Inhalts auswählen würde. Beispielsweise würde ein Beitrag über die Vorteile von BMWs, Ferraris und Porsches mit den Social Tags „Sportwagen“ und „Motorsport“ versehen werden.

Entities: *OpenCalais* identifiziert Entitäten, respektive Eigennamen die im Text vorkommen, dabei werden verschiedene semantischen Metadaten unterstützt, beispielsweise *Anniversary, City, Country, Person, Organization, etc.*

Events & Facts: Auch bestimmte Ereignisse und Fakten werden von *OpenCalais* erkannt. Wie bei den Entitäten werden auch diese durch semantische Metadaten klassifiziert, beispielsweise *Acquisition, Bankruptcy, BusinessRelation, CompanyFounded, etc.*

2.2.2 Zemanta API

Mithilfe der *Zemanta* API[3] können ebenfalls Merkmale im Text, wie beispielsweise Schlagwörter oder Kategorien gefunden werden. Zusätzlich wird

[3] http://developer.zemanta.com/docs/ – abgerufen am 14. Mai 2013

der gesendete Text mit Inhalten aus verschiedenen Quellen im WWW abgeglichen und Verlinkungen zu verwandten Inhalten zurückgeliefert. Derzeit liefert ein API Aufruf vier verschiedene Hauptergebnisse, zu jedem wird auch ein Konfidenz Wert angegeben der die Relevanz des Ergebnisses für den gesendeten Text angibt. Optional kann die Antwort eines API Aufrufs auch Kategorien beinhalten die den Text charakterisieren [25].

Artikel: Ein Teil der API Antwort sind Links zu Artikeln, die zum gesendeten Text verwandten sind. Dabei handelt es sich nicht ausschließlich um Artikel im herkömmlichen Sinne sondern beispielsweise auch um Tweets, Fotoalben oder bekannte Zitate. Diese Artikel können einerseits direkt auf einer Seite verlinkt werden um dem Leser einen tieferen Einblick in ein Thema zu ermöglichen, sie werden aber auch von Autoren genutzt um weitere Fakten in die eigenen Texte einfließen zu lassen. Die vorgeschlagenen Inhalte stammen aus verschiedenen Quellen im Internet, beispielsweise bekannte Nachrichtenportale oder renommierte Blogs. Im Jahr 2011 wurden von *Zemanta* 50000 verschiedene Informationsquellen berücksichtigt, die laufend manuell geprüft werden um Spam zu vermeiden.

Keywords: Dem gesendeten Text werden auch Keywords zugeordnet, die den Inhalt beschreiben, dadurch müssen diese nicht mehr manuell vom Autor gefunden werden. Es werden nicht nur Wörter vorgeschlagen die explizit im Text vorkommen sondern auch generelle Themen oder Konzepte die den gesamten Inhalt des Textes beschreiben.

Bilder: Die *Zemanta* API liefert Referenzen zu Bildern aus verschiedenen Quellen zurück. Hauptsächlich werden diese aus *Wikipedia* und *Flickr* entnommen, aber auch von verschiedenen Stock Image Anbietern. *Zemanta* ist dabei bemüht nur Bilder vorzuschlagen die frei, ohne die Zustimmung des Besitzers, veröffentlicht werden dürfen.

Links: Für im Text vorkommende Konzepte und Begriffe stellt die API auch Links zur Verfügung die diese näher beschreiben, dabei wird der Kontext des Wortes berücksichtigt. Dazu verwendet *Zemanta* Wissensdatenbanken, wie beispielsweise *Wikipedia*, *IMDB*, *Amazon* oder *Youtube*.

2.2.3 Alchemy API

Die *Alchemy* API[4] stellt Benutzern verschiedene Webservice Funktionen bereit, die besonders umfangreiche Möglichkeiten bietet semantische Metadaten abzurufen. Ein großer Vorteil der *Alchemy* API gegenüber anderen APIs

[4] http://www.alchemyapi.com/ – abgerufen am 14. Mai 2013

ist, dass Texte in 8 verschiedenen Sprachen analysiert werden können, auch in Deutsch. Die *Alchemy* API bietet folgende Funktionalität [22]

Named Entity Extraction: Ähnlich wie bei *OpenCalais* werden auch hier benannte Entitäten und Eigennamen in Texten identifiziert, wie Personen, Unternehmen, Organizationen oder Städte und andere geographische Merkmale.

Sentiment Analysis: Die *Alchemy* API ermöglicht es die Stimmung oder Meinung eines Textes zu identifizieren, also ob der Text ein Thema positiv, negativ oder neutral beschreibt.

Keyword/Term Extraction: Wie auch bei den zuvor genannten APIs können auch hier Schlagwörter extrahiert werden, die für die Beschreibung des Inhaltes relevant sind.

Concept Tagging: Die Konzepte die die *Alchemy* API identifiziert, sind Begriffe die den Text abstrakter beschreiben als Schlagwörter ähnlich wie die Social Tags der *OpenCalais* API. Es wird versucht Zusammenhänge zwischen einzelnen Wörtern im Text zu finden und diese durch einen gemeinsamen Überbegriff zusammenzufassen.

Relation Extraction: Diese Funktionalität ermöglicht es herauszufinden welche Beziehungen zwischen Subjekten, Aktionen und Objekten in einem Text bestehen, wodurch noch spezifischere semantische Informationen gefiltert werden können. Im folgenden Beispiel einer Analyse des Satzes: *"A determined Bill Clinten traveld to Chicago"* wird die Realation Extraction veranschaulicht. Die API identifiziert *Bill Clinton* als Person, *Chicago* als Stadt und die Beziehung der beiden im Text durch die Aktion *traveled to*:

```
<subject>
  <text>A determined Bill Clinton</text>
  <entity>
    <type>Person</type>
    <text>Bill Clinton</text>
  </entity>
</subject>
<action>
  <text>traveled to</text>
</action>
<object>
  <text>Chicago</text>
  <entity>
    <type>City</type>
    <text>Chicago</text>
  </entity>
</object>
```

Topic Categorization/Text Classification: Wie auch bei den bisher beschriebenen APIs kann auch durch die *Alchemy* API der gesamte Text einer bestimmten Kategorie zugewiesen werden.

Author Extraction: Die API kann den Autor eines Beitrags auf einer Website identifizieren ohne, dass dessen Name explizit durch HTML gekennzeichnet ist.

Language Identification: Die *Alchemy* API erkennt die Sprache in der ein bestimmter Text geschrieben ist. Auch wenn die weitere Analyse in nur 8 Sprachen erfolgen kann, können mehr als 95 Sprachen identifiziert werden.

Text Extraction: Entfernt alle HTML Tags einer Website, wie Navigationslinks, Werbung und ähnliches und liefert den eigentlichen Kerninhalt der Seite, beispielsweise einen Artikel zurück.

Content Scraping: Diese Funktionalität ermöglicht es strukturierte Daten aus einem Text zu filtern, beispielsweise Produktmerkmale, Beschreibungen, Kosten etc.

Microformats Parsing: Identifiziert alle Microformats die innerhalb einer Website vorkommen und liefert diese zurück.

RSS/ATOM Feed Detection: Liefert alle RSS/ATOM Feed Links die in einer Website eingebettet sind.

2.3 Kollaborative Daten

Durch Betrachten des Benutzerverhaltens soll, ersichtlich werden welche Artikel von einem Benutzer gelesen werden und so zwischen einzelnen Dokumenten Beziehungen hergestellt werden, um diese zum Generieren von Empfehlungen nutzen zu können. Die bekannteste Methode die in vielen Empfehlungssystemen verwendet wird um kollaborative Daten zu erheben ist das Kollaborative Filtern.

2.3.1 Kollaboratives Filtern

Seit jeher diskutieren Menschen über Filme die sie gesehen haben oder Bücher die sie gelesen haben, beeinflussen durch ihre Meinung die Meinungen anderer oder folgen Empfehlungen von Personen die ähnliche Vorlieben haben wie sie selbst. Dieses Verhalten dient als Vorbild für das kollaborative Filtern.

Das Ziel von Systemen die kollaboratives Filtern anwenden ist, automatisiert vorherzusagen welche Objekte einen bestimmten Benutzer interessieren und Empfehlungen zu geben. Dabei kann es sich um verschiedene Objekte handeln wie beispielsweise Bücher, CDs, Artikel oder Urlaubsziele.

Bei dieser Methode werden die Benutzer aktiv mit einbezogen, durch Bewertungen können sie ihre Meinung zu bestimmten Objekten abgeben. Diese Bewertungen werden herangezogen um Gemeinsamkeiten zwischen Benutzern zu berechnen und Personen mit ähnlichen Interessen zu identifizieren, beim kollaborativen Filtern werden diese Benutzer als Nachbarn bezeichnet. Vorhersagen und Empfehlungen für einen individuellen Benutzer werden aus den Bewertungen seiner Nachbarn generiert [19].

2.3.2 Webanalyse Werkzeuge

Eine andere Herangehensweise an das Erheben kollaborativer Daten ist die Verwendung von Werkzeugen aus der Webanalyse. Diese werden für gewöhnlich verwendet um die Effizienz einer Webseite, und deren Popularität zu prüfen. Dabei wird ebenfalls das Verhalten des Benutzers betrachtet und mit den Ergebnissen versucht Webseiten noch benutzerfreundlicher und intuitiver zu gestalten. Einige dieser Werkzeuge können auch herangezogen werden um zu erheben welche Seiten ein Benutzer aufruft und welche Artikel gelesen werden.

In [12] werden die einzelnen Webanalyse Metriken näher erklärt, hier eine Liste der Metriken die im hier beschrieben Zusammenhang wichtig sind:

1. **Besuch:** sequentielle Anzeige einer oder mehrerer Seiten von einem Besucher, dieser wird durch die Internet Protocol (IP)-Adresse identifiziert. Setzt der Benutzer für einen gewissen Zeitraum, beispielsweise 30 Minuten, keine Interaktion mehr gilt der Besuch als beendet.
2. **Seitenaufruf:** Jede von einem Benutzer während eines Besuchs nachgefragte Seite wird als Seitenaufruf bezeichnet.
3. **Klickpfad:** Gibt an wie der Benutzer durch die Seiten navigiert, beispielsweise welchen Verlinkungen er folgt.
4. **Verweildauer:** Dauer des gesamten Besuchs der Webseite oder auch die Dauer die der Besucher auf einer bestimmten Seite verbringt.

Die Erhebung der Daten kann serverseitig oder auch clientseitig stattfinden. Ruft ein Benutzer eine Webseite im Browser auf wird eine Anfrage an einen Webserver gesendet. Vom Server werden die entsprechenden Dateien an den Browser zurück geliefert, gleichzeitig werden Informationen über die Anfrage und die Antwort vom Server in Logfiles protokolliert. Zu jedem Aufruf können beispielsweise Datum und Uhrzeit, URL der aufgerufenen Datei, IP-Adresse des Aufrufenden, Herkunft des Aufrufs, Cookies, usw. gespeichert werden. Alternativ dazu können die Daten auch clientseitig erfasst werden, die Informationen die so erhoben werden können sind wesentlich umfangrei-

cher als es durch Logfiles möglich ist. Serverseitig ist nur die Erhebung von Daten über die Seitenaufrufe möglich, Interaktionen des Benuzters zwischen zwei Seitenaufrufen oder Informationen zu weiteren Einstellungen des Benutzers sind in Logfiles nicht ersichtlich. Die clientseitige Datenerhebung kann mithilfe von JavaScript erfolgen, so können Informationen über Mausklicks, Tastatureingaben, Browsereinstellungen, Titel einer Seite, usw. erfasst werden [12].

Als Unterstützung für Webseiten Betreiber gibt es vorgefertigte Webanalyse Systeme die die Erhebung und großteils auch die Auswertung der Daten übernehmen. Das wohl am häufigsten verwendete Tool ist *Google Analytics*,[5] das einen besonders hohen Funktionsumfang und eine einfache Benutzeroberfläche bietet. Es gibt jedoch auch Alternativen die immer mehr an Bedeutung gewinnen, beispielsweise die Open-Source Lösung *Piwik*.[6]

[5] http://www.google.at/intl/de/analytics/ – abgerufen am 20. August 2013
[6] http://piwik.org/ – abgerufen am 20. August 2013

Kapitel 3

Verwandte Arbeiten

Die hier betrachteten verwandten Projekte befassen sich vorwiegend mit der semantischen Ähnlichkeit zwischen Texten, dieser wird in den letzten Jahren immer mehr Bedeutung zugemessen und daher wird in diesem Bereich intensiv geforscht.

Eine sehr effektive und populäre Methode zur Berechnung von semantischer Ähnlichkeit und semantischen Beziehungen zwischen Wörtern, ist die latente semantische Analyse (LSA) [5]. Als Ausgangsbasis dient eine große Sammlung natürlichsprachlicher Texte, aus diesen Dokumenten werden Informationen über die Häufigkeit und die Beziehungen zwischen Wörtern gewonnen und zur Weiterverarbeitung in einer Termfrequenz-Matrix abgebildet (s. Tab. 3.1). Überflüssige und für den Inhalt wenig ausschlaggebende Wörter werden ignoriert, indem beispielsweise ein Schwellwert für die Termhäufigkeit festgelegt wird oder Stoppwörter gefiltert werden. Auch Gewichtungsmethoden können in der LSA einfließen. Ein wichtiger Bestandteil des Prozesses ist, eine Singulärwertszerlegung der Termfrequenz-Matrix, wodurch drei Teilmatrizen entstehen, die durch Multiplikation wieder die Originalmatrix ergeben. Durch die so entstehende Reduktion der Anzahl von Dimensionen in den Teilmatrizen gehen überflüssige Informationen der Originalmatrix verloren und es verbleiben die wesentlichen Daten. Die LSA führt dazu, dass ein semantischer Raum entsteht, in dem Terme und Dokumente durch ihre Vektoren repräsentiert werden. Die Position eines Terms hängt von der gemeinsamen Verwendung mit anderen Termen ab, daher liegen nicht nur Wörter nahe beieinander die häufig gemeinsam auftreten sondern auch Wörter die im gleichen Kontext verwendet werden, beispielsweise Synonyme aber auch Substantive oder Flexionen von Verben.

LSA wird in vielen Applikationen eingesetzt, in der Literatur wird jedoch die hohe Komplexität und der dadurch entstehende hohe Rechenaufwand als Nachteil angeführt [4, 13].

Ein Verfahren, das sowohl semantische als auch syntaktische Merkmale der Texte in die Ähnlichkeitsberechnung mit einfließen lässt wird in [13]

Tabelle 3.1: Beispieltextsammlung und daraus entstehende Termfrequenz-Matrix[5]. In die Matrix wird die Häufigkeit eines Terms in einem Dokument eingetragen. Bei den Beispieltexten handelt es sich um Titel von Memos und es werden nur Terme aus den Texten in die Matrix übernommen die in mehr als einem Dokument vorkommen.

Dokumente	
d1:	*Human* machine *interface* for Lab ABC *computer* applications
d2:	A *survey* of *user* opinion of *computer system response time*
d3:	The *EPS user interface* management *system*
d4:	*System* and *human system* engineering testing of *EPS*
d5:	Relation of *user*-perceived *response time* to error measurement
d6:	The generation of random, binary, unordered *trees*
d7:	The intersection *graph* of paths in *trees*
d8:	*Graph minors* IV: Widths of *trees* and well-quasi-ordering
d9:	*Graph minors*: A *survey*

Terme	**Dokumente**								
	d1	d2	d3	d4	d5	d1	d2	d3	d5
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

beschrieben. Hier werden drei Maße betrachtet, die Ähnlichkeit der Zeichenketten, die semantische Ähnlichkeit der Wörter und als syntaktische Einflussgröße die gemeinsame Wortfolge, diese werden kombiniert und normiert um ein generelles Ähnlichkeitsmaß zu erhalten. Ziel des Verfahren ist es, einen normierten Wert zwischen 0 und 1 zu errechnen, der aussagt wie hoch die Ähnlichkeit zwischen zwei Texten ist, dazu soll für jedes Wort des ersten Textes, das am meisten ähnliche Wort aus dem zweiten Text gefunden werden. Beide Eingabetexte werden zur weiteren Verarbeitung in einzelne Wörter zerlegt, wobei Stoppwörter von vornherein gefiltert werden, außerdem werden alle Wörter in ihre Grundform zurückgeführt.

Die Ähnlichkeit der Zeichenketten wird durch die längste gemeinsame Teilsequenz ermittelt. Die semantische Ähnlichkeit wird durch ein korpusba-

siertes Verfahren ausgewertet, Informationen über Kontext und Häufigkeiten von Wörtern wird aus dem British National Corpus[1] entnommen. Die Berechnung der Ähnlichkeit zwischen zwei Wörtern basiert auf der Höhe der statistischen Wahrscheinlichkeit, dass sie zusammen auftreten, diese Art von Information nennt man Pointwise Mutual Information (PMI). Das Vorgehen basiert auf dem PMI-IR Algorithmus, der in [20] beschrieben wird. Hier werden Kookkurrenz-Daten verwendet, die durch Information Retrieval Methoden gewonnen werden. Ursprünglich wurde der Algorithmus entwickelt um für ein Wort passende Synonyme zu finden, dazu soll die höchste Ähnlichkeit zwischen dem Suchwort und verschiedenen Kandidaten ermittelt werden. Der PMI-IR Algorithmus errechnet für jeden möglichen Kandidaten einen Ähnlichkeitswert und wählt dann den Kandidaten mit dem höchsten Wert aus. Die Pointwise Mutual Information für zwei Terme w_1 und w_2 wird grundsätzlich wie folgt berechnet:

$$PMI(w_1, w_2) = \log_2 \left(\frac{p(w_1 \wedge w_2)}{(p(w_1) \cdot p(w_2))} \right), \tag{3.1}$$

$p(w_1 \wedge w_2)$ bezeichnet die Wahrscheinlichkeit, dass w_1 und w_2 gemeinsam auftreten. Sind die beiden Wörter statistisch unabhängig, wird dies durch $p(w_1) \cdot p(w_2)$ definiert. Sind sie nicht statistisch unabhängig ist der Wert von $p(w_1 AND w_2)$ größer als $p(w_1) \cdot p(w_2)$ und der Grad der Ähnlichkeit wird durch ihr Verhältnis festgelegt.

Diese Berechnung wird in [20] angepasst, da nach dem maximalen Ähnlichkeitswert aller Kandidaten gesucht wird, verzichtet man auf die Verwendung des Logarithmus, weil dieser monoton ansteigt und auf den Einfluss von $p(w_1)$, da dieser Wert für jeden Kandidaten gleich ist. Für die Wahrscheinlichkeitsberechnung wird die Anzahl von Treffern herangezogen, die bei einer entsprechenden Suche mit der Suchmaschine *AltaVista* gewonnen werden. Die angepasste Formel des PMI (s. Gl. 3.1) die von [20] verwendet wird, ist

$$PMI(w_1, w_2) = \frac{hits(w_1 \wedge w_2)}{hits(w_2)}. \tag{3.2}$$

Es ist möglich den Rahmen bzw. Abstand in dem die Wörter gemeinsam auftreten müssen, noch näher zu spezifiziert werden, dies kann vom gemeinsamen Vorkommen in einem Dokument bis zu kleineren Einheiten wie beispielsweise, dass beide Terme innerhalb von 10 Wörtern in einem Dokument stehen müssen reichen.

In [13] wird eine modifizierte Version des Algorithmus verwendet, in dem ebenfalls Pointwise Mutual Information eine Rolle spielt. Hier wird für jedes der beiden Wörter, deren Ähnlichkeit ermittelt werden soll, eine gewichtete Liste der Wörter erstellt mit denen es häufig auftritt (Nachbarwörter) und nach Übereinstimmungen in beiden Listen gesucht.

[1] http://www.natcorp.ox.ac.uk/ – abgerufen am 3. September 2013

Als syntaktisches Maß wird die Anzahl der Terme betrachtet, die in gleicher Reihenfolge in beiden Texten vorkommen, es wird empfohlen diese Einflussgröße sehr niedrig zu gewichten, beziehungsweise nur optional zu verwenden, da die Aussagekraft nicht besonders hoch ist.

In [10] wird ein Ansatz beschrieben, bei dem ein semantisches Netzwerk zur Ermittlung der semantischen Ähnlichkeit zwischen Begriffen verwendet wird. Dieses Verfahren ist analog zum menschlichen Vorgehen Assoziationen zwischen Begriffen zu erkennen, aus Informationsfragmenten wird neues Wissen gewonnen und in das bereits bestehende Wissen integriert, wodurch ein großes Netzwerk aus Informationen entsteht. Ausgehend von einem bestehenden Textkorpus, der in diesem Projekt von verschiedenen Webseiten entnommen wurde, wird automatisch ein semantisches Netzwerk erstellt, indem Konzepte oder Entitäten als Knoten und Beziehungen zwischen diesen durch gerichtete Kanten dargestellt werden (s. Abb. 3.1). Dies geschieht mithilfe eines Parsers der auf verschiedene Natural Language Processing Methoden zurückgreift um syntaktische und semantische Beziehungen aus dem Text zu extrahieren. Die semantische Ähnlichkeit zwischen den einzelnen Knoten im Netzwerk wird durch die Aktivierungsausbreitung ermittelt. Bei diesem Verfahren wird jeder Knoten im Netzwerk mit zwei Attributen ausgestattet, zum einen einen aktuellen Grad an Aktivierung und zum anderen einen Schwellwert, erreicht der aktuelle Aktivierungsgrad den Schwellwert aktiviert der Knoten seine Nachbarknoten, wiederum mit einem bestimmten Aktivierungsgrad, so breitet sich die Aktivierung im Netzwerk aus. Die Höhe des Aktivierungsgrad der zwischen den Knoten weitergegeben wird, wird mit der Distanz verringert, wodurch es möglich wird die Stärke der semantischen Beziehung zwischen zwei Begriffen im Netzwerk zu definieren. Will man beispielsweise die Ähnlichkeit zwischen Knoten x und Knoten y ermitteln, wird zuerst der Knoten x mit einem bestimmten Grad an Aktivierung belegt und dadurch die Aktivierungsausbreitung im Netzwerk gestartet. Danach wird der vom Knoten y erhaltene Aktivierungsgrad als *act(x,y)* gespeichert. Dieser Vorgang wird mit y als Initial Knoten wiederholt wodurch man *act(y,x)* erhält. Die Summe dieser beiden Werte wird als Maß der Ähnlichkeit zwischen den beiden Knoten herangezogen.

Immer bekannter werden Ansätze zur Berechnung von semantischer Ähnlichkeit die auf der Online-Enzyklopädie *Wikipedia* basieren, *Wikipedia* bietet viele Vorteile in diesem Bereich (s. Abschn. 2.1.2). Die explizite semantische Analyse (ESA) [8] ist eine Methode, wodurch die Semantik natürlichsprachlicher Konzepte die aus *Wikipedia* Artikeln entnommen werden, beispielsweise *computer science* oder *language*, sehr detailliert repräsentiert werden.

Dazu wird mittels Machine Learning Techniken ein semantischer Interpreter konstruiert, der natürlichsprachliche Textsegmente entgegennimmt und für den Text relevante *Wikipedia* Konzepte zuordnet. Für jedes *Wikipedia* Konzept wird ein Attribut-Vektor erstellt aus allen Wörtern die in den entsprechenden Artikeln vorkommen, die Relevanz der Wörter für ein Kon-

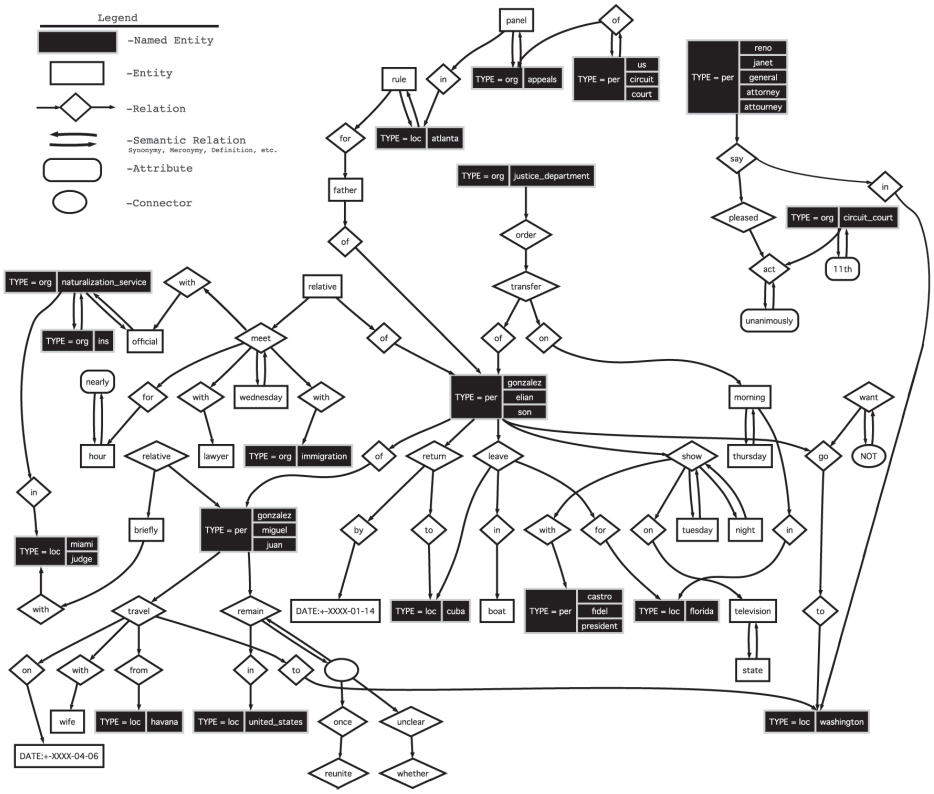


Abbildung 3.1: Beispiel eines semantischen Netzwerkes aus [10]

zept wird mithilfe des TF-IDF Maßes (s. Abschn. 2.1.2) bewertet. Um die Interpretation zu beschleunigen wird ein invertierter Index angelegt in dem jedes Wort einer Liste von Konzepten zugeordnet wird in dem es auftritt. Ist die Assoziation zwischen einem Wort und einem Konzept sehr schwach wird diese Assoziation nicht in den Index aufgenommen.

Der semantische Interpreter iteriert zuerst über die Wörter des Eingabetexts und ermittelt die dazu passenden Einträge im invertierten Index, diese werden je nach Relevanz gereiht und ebenfalls durch einen Vektor repräsentiert. Die Ähnlichkeit von zwei Eingabetexten wird durch das Cosinus-Maß ihrer Vektoren ermittelt. Aufbau und Verwendung des semantischen Interpreters wird in Abb. 3.2 dargestellt.

Mithilfe der Suchmaschine *Google* und den von *Google* indizierten Webseiten des WWW wird die Normalized Google Distance (NGD) berechnet [4]. Ein Maß das Auskunft über den semantischen Zusammenhang von zwei Termen gibt und auf der Annahme basiert, dass man objektive Informationen zu semantische Beziehungen zwischen Begriffen aus ihrem Vorkommen im WWW schließen kann. Die Berechnung basiert auf der Anzahl von Treffern die eine Suchanfrage mit der Suchmaschine liefert. Berücksichtigt wird

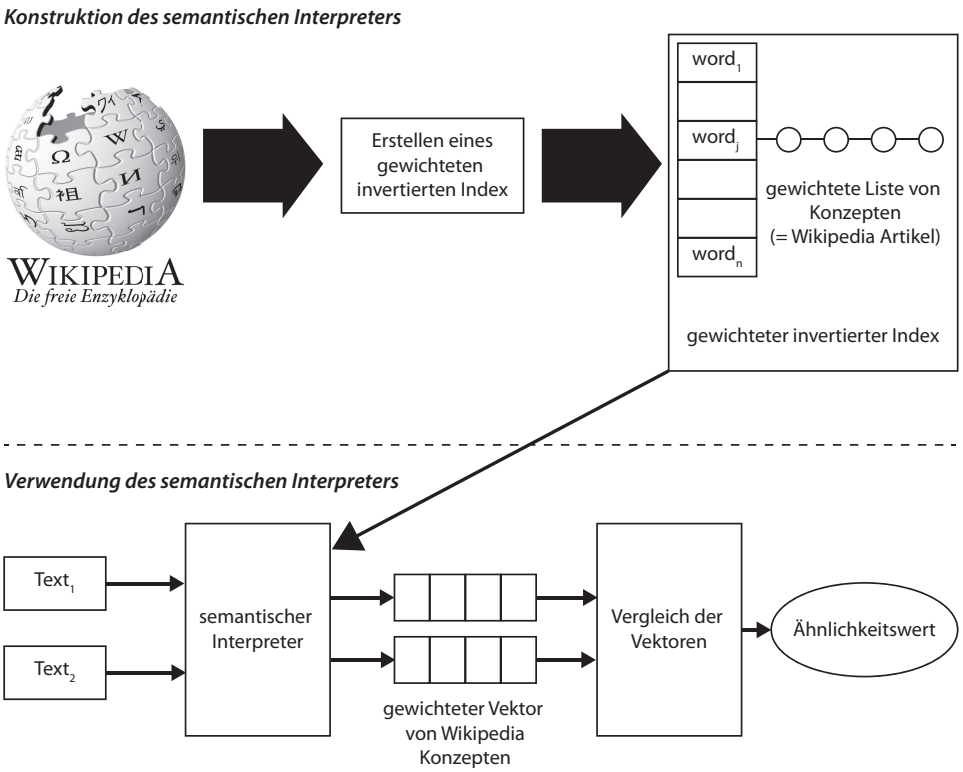


Abbildung 3.2: Darstellung von Aufbau und Verwendung des semantischen Interpreters nach [8].

zum einen die Höhe der Suchergebnisse für jeweils einen der Terme einzeln zum anderen die Anzahl gemeinsamer Auftreten.

Zur Berechnung der *NGD* für zwei Begriffe w_1 und w_2 wird folgende Formel definiert

$$NGD(w_1, w_2) = \frac{\max\{\log f(w_1), \log f(w_2)\} - \log f(w_1, w_2)}{\log N - \min\{\log f(w_1), \log f(w_2)\}}, \tag{3.3}$$

wobei $f(w_1)$ die Anzahl der Webseiten in denen w_1 auftritt bezeichnet und $f(w_1, w_2)$ die Seiten in denen beide Terme gemeinsam vorkommen. Als Normierungsfaktor N kann jeder Wert verwendet werden der größer ist als $f(w_1)$ und kleiner als die Gesamtanzahl der von *Google* indizierten Seiten.

In [4] wird folgendes Beispiel gegeben[2], eine Suche nach dem Begriff „horse“ liefert 46.700.000 Treffer. Eine Suche nach dem Wort „rider“ ergibt 12.200.000 Treffer und eine Suche nach beiden Begriffen gemeinsam führt zu 2.630.000 Suchergebnissen. Insgesamt waren von *Google* zu diesem Zeitpunkt 8.058.044.651 Seiten indiziert, dieser Wert wird in diesem Beispiel als

[2]Die Anzahl der Treffer entsprechen einem Stand von 2007

Normierungsfaktor verwendet. Setzt man diese Zahlen in die Gleichung 3.3 ein, ergibt dies eine $NGD(horse, rider) \approx 0,443$. Die *NGD* ist ein normiertes Maß das zwischen 0 und 1 liegt, wobei ein Wert von 0 bedeutet, dass die beiden Terme ident sind.

In einem zweiten Versuch indem angenommen wird, dass *Google* nur die Hälfte der Seiten (4.285.299.774) indiziert hat und dadurch auch die Anzahl der Treffer für die einzelnen Begriffe „horse“ (23.700.000) und „rider“ (6.270.000) und einer gemeinsame Suche „horse, rider“ (1.180.000) variieren, wird eine $NGD(horse, rider) \approx 0,460$ berechnet. Dies zeigt, dass eine steigende Anzahl von indizierten Seiten keinen großen Einfluss auf das Maß nimmt und sich der Wert stabilisiert.

Das *Ceryx* Framework [3] wurde speziell entwickelt um im Online News Bereich Empfehlungen, basierend auf einem Benutzerprofil und der semantischen Ähnlichkeit der Artikelinhalte zu generieren. Alle Artikel die ein Benutzer bereits gelesen hat, werden in einem Profil gespeichert, um die Vorlieben und Interessen des Benutzers zu erfahren. Für die Empfehlungen wird nach der semantischen Ähnlichkeit zwischen Daten aus diesem Profil und noch nicht gelesenen Artikeln ermittelt. Für die Berechnung der Ähnlichkeit werden zwei verschiedene Maße verwendet, die beide auf *WordNet* (s. Abschn. 2.1.2) Daten basieren.

WordNet liefert zu jedem Wort ein Synset, eine Reihe von möglichen Bedeutungen des Wortes. Für den Begriff „turkey“ beispielsweise werden die Erklärungen „Meleagris gallopavo“ (das Tier) und „Republic of Turkey“ zurückgegeben.

Diese Synsets werden in [3] einerseits in einer Modifizierung des TF-IDF Maßes verwendet dem Synset Frequency - Inverse Document Frequency (SF-IDF) Maß. Hier wird nicht, wie bei der Berechnung des TF-IDF Wertes, die Häufigkeit eines Terms ermittelt sondern eines Synsets, wodurch mehrere Wörter zusammengefasst werden. Es werden sowohl für die Artikel im Benutzerprofil, als auch für ungelesene Artikel die SF-IDF Werte berechnet und jeweils in einem Vektor gespeichert. Die Ähnlichkeit dieser Vektoren wird durch das Cosinus-Maß bestimmt.

Das Zweite in [3] vorgestellte Ähnlichkeitsmaß misst den semantischen Zusammenhang zwischen den Artikeln. Auch hier werden die *WordNet* Synsets aus den bereits vom Benutzer gelesenen Artikeln mit denen der noch nicht gelesenen Artikeln verglichen. Dazu wird ein Vektor mit jeder möglichen Kombination aus Synsets, aus dem Benutzerprofil und den noch nicht gelesenen Artikeln erstellt. Jedes Synset wird als Knoten in einer Taxonomie dargestellt, welche durch „is-a“ Beziehungen miteinander verknüpft werden. Für jede Kombinationen, bzw. jedes Knotenpaar im Vektor wird ein Ähnlichkeitswert berechnet, dazu werden fünf verschiedene Ähnlichkeitsmaße getestet. Das in der der Evaluation am erfolgreichsten ist das Maß von Wu und Palmer [21], das jeweils die Distanz zwischen den zwei Knoten in der Taxonomie betrachtet. Dazu wird der lowest common subsumer (LCS) von zwei

Knoten betrachtet, dies ist der in der Taxonomie am tiefsten liegende Knoten der beide Knoten, die gerade betrachtet werden, dominiert. Die Tiefe des LCS und Distanz der beiden Knoten ergibt den Ähnlichkeitswert, für zwei Knoten n_1 und n_2 würde die Berechnung wie folgt passieren:

$$sim(n_1, n_2) = \frac{2 \cdot depth(LCS(n_1, n_2))}{length(n_1, n_2) + 2 \cdot depth(LCS(n_1, n_2))}. \tag{3.4}$$

In [3] werden für den endgültigen Ähnlichkeitswert eines ungelesenen Artikels, die Werte aller möglichen Synset Kombinationen zwischen dem Artikel und dem Benutzerprofil aufsummiert und durch die Anzahl der Kombinationen geteilt.

Kapitel 4

Konzeption

In diesem Kapitel wird das Grundkonzept des implementierten Systems erläutert. Die Ermittlung der Ähnlichkeit zwischen einzelnen Textdokumenten wird auf einer Analyse der Inhalte und dadurch gewonnene Keywords der Texte basieren. Zur Identifikation der Keywords werden Semantic APIs herangezogen. Eine weitere Einflussgröße auf die Ähnlichkeit der Dokumente sind kollaborative Daten, diese werden mithilfe von Webanalyse Werkzeugen gewonnen. Beide Faktoren sollen zu einem Wert kombiniert werden der Auskunft über den Grad der Ähnlichkeit gibt. Eine vereinfachte Darstellung des Ablaufs und Aufbaus des Systems wird in Abb. 4.1 dargestellt.

4.1 Problem Definition

Ausgehend von dem Artikel der aktuell vom Benutzer gelesen wird, sollen aus der Gesamtheit aller im System vorhandenen Artikel diejenigen für die Empfehlungen ausgewählt werden, die der Benutzer hinterher lesen möchte. Die zu berücksichtigenden Faktoren können wie folgt definiert werden: (1) ein Artikel a aus der Gesamtheit aller Artikel D der vom Benutzer gelesen wird; (2) Kandidaten k sind die Artikel die dem Benutzer empfohlen werden können nachdem a aufgerufen wurde, sie sind ebenfalls Teil von D und (3) eine Ähnlichkeitsfunktion f die den Zusammenhang zwischen einem Kandidaten k und dem aktuell betrachteten Artikel a misst, also $f(a, k)$. Von f sollen sowohl inhaltsbasierte Daten, als auch kollaborative Daten berücksichtigt werden. Der in der Funktion erhobene Wert soll so ausgelegt sein, dass wenn $f(a, k_1) > f(a, k_2)$, ist k_1 dem Artikel a ähnlicher als k_2, außerdem soll der Wert im Bereich zwischen 0 und 1 liegen, wobei 1 die maximale Ähnlichkeit definiert und 0 keine Ähnlichkeit.

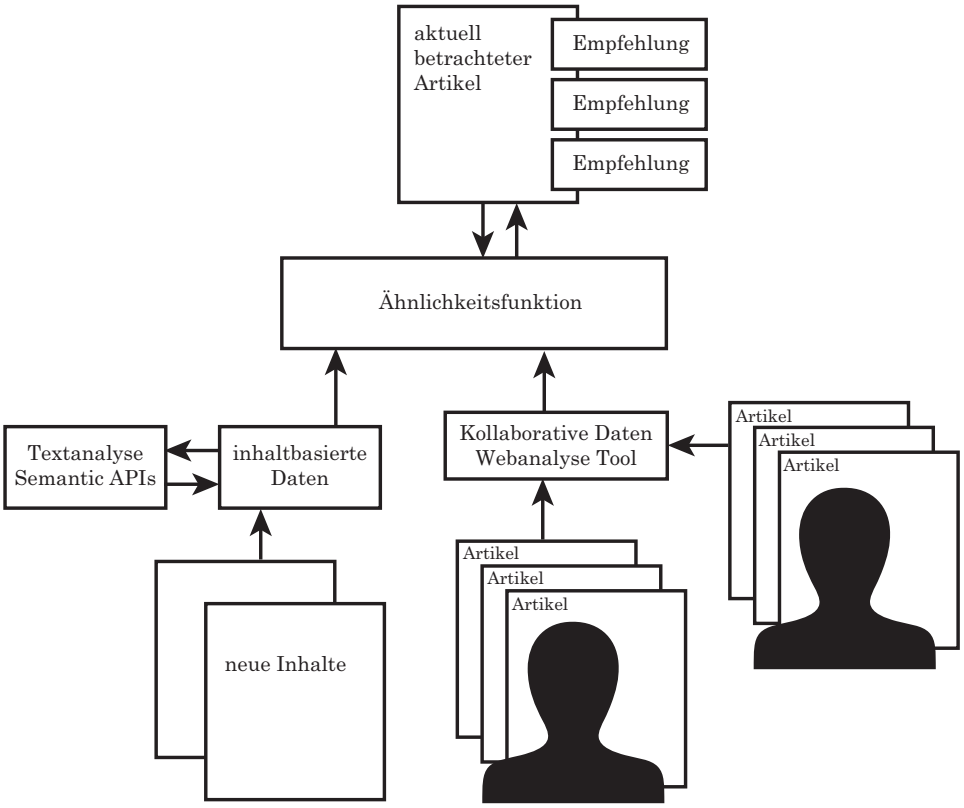


Abbildung 4.1: Schematische Darstellung vom Aufbau des Systems und Ablauf der Datenerhebung und Ausgabe.

4.2 Einfluss der kollaborativen Daten

Für die Erhebung der Ähnlichkeit sollen auch kollaborative Daten herangezogen werden. Diese werden durch Beobachten des Benutzerverhaltens gewonnen. Es wird davon ausgegangen, dass Artikel die nacheinander von einem Benutzer gelesen werden auch thematisch zusammenhängen. Liest ein Benutzer einen Artikel, sollen demnach weitere Artikel vorgeschlagen werden die andere Benutzer ebenfalls betrachtet haben, bevor oder nachdem sie den aktuell angezeigten Artikel gelesen haben. Beim hier umgesetzten System wird dafür laufend betrachtet welche Seiten von einzelnen Benutzern gelesen werden. Es wird jedoch nur die Sequenz der Seiten und wie viel Zeit ein Benutzer auf den einzelnen Seiten verbringt gespeichert. Diese Daten sind absolut anonym und werden so abgelegt, dass keine weiteren Informationen über den Benutzer nachvollziehbar sind, es also auch keinen Konflikt mit Privatsphäre oder Datenschutz der Benutzer gibt.

Zur Ermittlung der Daten sollen Webanalyse Werkzeuge eingesetzt werden. Diese ermöglichen es zu betrachten welche Seite ein Benutzer während

eines Besuchs öffnet und wie lange er dort verweilt. Damit können alle Daten gesammelt werden, die in diesem Zusammenhang Informationen über die Ähnlichkeit von Artikeln liefern. Ein Risikofaktor ist, dass man nur messen kann wie lange eine Seite im Browser geöffnet ist, ob der Benutzer den Text wirklich liest ist nicht festzustellen. Um hier absolute Sicherheit zu erlangen müsste jedoch eine direkte Benutzerinteraktion gefordert werden.

Dennoch kann die Zeitspanne herangezogen werden um die Ergebnisse aufzuwerten. Entspräche der angezeigte Artikel, beispielsweise überhaupt nicht dem Interesse des Benutzers, würde dieser die Seite rasch wieder verlassen. Es wird daher ein Schwellwert von 15 Sekunden festgelegt, so lange muss die Seite mindestens geöffnet sein um in die Berechnungen mit einzufließen.

4.3 Auswahl der Semantic APIs

Für die Analyse der Inhalte werden Semantic APIs herangezogen. Diese identifizieren Wörter in Texten, die für den Inhalt sehr relevant sind und diesen charakterisieren. In [23] findet man eine Gegenüberstellung und Evaluierung verschiedener Semantic APIs. Für die Extraktion von Keywords und Entitäten erbringt *Zemanta* die besten Testergebnisse. Auch *OpenCalais* liefert sehr hochwertige Ergebnisse und ist vor allem durch den hohen Bekanntheitsgrad interessant, da dieser auf eine stetige Weiterentwicklung und Qualitätssteigerung schließen lässt. Die *Alchemy* API schneidet im Vergleich zu *Zemanta* und *OpenCalais* in den Tests schlechter ab, bringt jedoch gegenüber anderen APIs auch sehr gute Ergebnisse. Aufgrund dieser Informationen sollen, für das hier beschriebene System, folgende Metadaten dieser drei APIs herangezogen werden, eine detaillierte Beschreibung der Metadaten wird in Abschn. 2.2 gegeben:

Zemanta:

- Keywords

OpenCalais:

- Topics
- Social Tags
- Entitäten

Alchemy:

- Konzepte
- Entitäten
- Keywords

Jede der APIs gibt zu allen genannten Metadaten einen Relevanz Wert an, der bewertet wie aussagekräftig der Term für den Inhalt des Textes ist. Dieser liegt, mit Ausnahme der Social Tags von *OpenCalais* immer zwischen 0, keine

Tabelle 4.1: Ergebnisse der *Zemanta* API Analyse eines Artikels mit dem Titel „Syria crisis: France, US and UK want 'strong' UN resolution".

Zemanta API	
Keywords	
Syria	0.266
United States	0.244
John Kerry	0.135
Bashar al-Assad	0.102
Sergey Lavrov	0.099
Chemical Weapons Convention	0.072
Russia	0.068
United Nations Security Council	0.067

Tabelle 4.2: Ergebnisse der *OpenCalais* API Analyse eines Artikels mit dem Titel „Syria crisis: France, US and UK want 'strong' UN resolution".

OpenCalais API			
Entitäten		Social Tags	
Syria	0.748	Asia	1
Bashar al-Assad	0.648	Middle East	1
Foreign relations of Syria	0.542	Assad family	1
chemical arms arsenal	0.307	Syrian uprising	2
UN Security Council	0.212	Levant	2

Relevanz und 1, sehr relevant. Die Social Tags werden mit den Werten 1 oder 2 gewichtet. In den Tabellen 4.1,4.2 und 4.3 werden Auszüge aus den Ergebnissen einer Analyse, eines Artikels der BBC-Webseite zum Thema Syrien-Krise,[1] durch die verschiedenen APIs aufgelistet.

Im Hinblick auf die Berechnung der Ähnlichkeit zwischen den Dokumenten ist es von Vorteil, Informationen über die Relevanz möglichst vieler Terme eines Textes zu haben. Daher soll eine Kombination der identifizierten Metadaten aller APIs vorgenommen werden. Ein Problem dabei ist die Zusammenführung der Relevanz Bewertungen. Diese werden je nach API unterschiedlich berechnet, die Social Tags von *OpenCalais* werden mit 1 oder 2 gewichtet, *Zemanta* liefert außerdem tendenziell niedrigere Bewertungen als die anderen APIs. In [1] wird die Kombination verschiedener Semantic APIs beschrieben und evaluiert. Dort wird angeführt, dass zur Vereinigung der Relevanz Werte der Median der verschiedenen Werte verwendet werden kann. Liefert beispielsweise *Zemanta* für einen Term den Wert 0.1, *OpenCalais* den

[1] http://www.bbc.co.uk/news/world-middle-east-24105381 – abgerufen am 16. September 2013

Tabelle 4.3: Ergebnisse der *Alchemy* API Analyse eines Artikels mit dem Titel „Syria crisis: France, US and UK want 'strong' UN resolution".

Alchemy API			
Konzepte			
Bashar al-Assad		0.949	
United Nations		0.693	
Iraq		0.495	
Entitäten		Keywords	
Syria	0.948	resolution	0.569
chemical weapons	0.752	Security Council	0.562
UN	0.804	precise timetable	0.418
UN Security Council	0.679	Laurent Fabius	0.468
United States	0.423	Mr Kerry	0.623

Wert 0.5 und *Alchemy* API identifiziert den Wert nicht, gleichbedeutend mit einer Bewertung mit 0, wäre der Median 0.1. Wird bei dieser Methode allerdings ein Keyword von zwei APIs nicht identifiziert wäre der Median 0, wodurch das Wort nicht weiter betrachtet wird.

Auch beim hier umgesetzten System wird der Median der Werte verwendet. Liefert nur eine der APIs ein Ergebnis zu einem Term und liegt die Relevanz über dem definierten Schwellwert von 0.7, wird hier jedoch dieser Wert für den Term behalten.

Durch die unterschiedlichen Bewertungen ergeben sich Schwierigkeiten für die inhaltsbasierte Ähnlichkeitsberechnung da diese auf den Relevanzwerten der APIs aufbaut. Eine Herausforderung ist die Bewertung der Social Tags, in der Dokumentation von *OpenCalais* [24] wird angeführt, dass Topics die einen höheren Relevanz Wert als 0.6 aufweisen in die Social Tag Ergebnisse aufgenommen werden. Liegt der Wert zwischen 0.6 und 0.8, wird ihre Relevanz als Social Tag auf 2 gesetzt, liegt er über 0.8 wird ihr Social Tag Wert mit 1 definiert. Diese Umlegung der Werte wird im hier beschriebenen System entsprechend umgekehrt durchgeführt, Social Tags mit einem Relevanz Wert von 1 werden im System mit 0.8 bewertet, respektive ein Relevanz Wert von 2 wird durch 0.6 ersetzt.

Die Beurteilung der *Zemanta* API unterscheiden sich stark von denen der anderen APIs. Dies erschwert bzw. verzerrt den Vergleich der Werte. An den Beispieldaten (s. Tab. 4.1,4.2 und 4.3) ist ersichtlich, dass der Begriff „Syria" von allen drei APIs als wichtig eingestuft wird. Die *Zemanta* API vergibt jedoch dennoch nur den Wert 0.266 im Gegensatz zum Wert 0.748 der von *OpenCalais* geliefert wird ist dies sehr niedrig. Da auch aus der Dokumentation der einzelnen APIs nicht klar ersichtlich ist wie die Werte zustande kommen oder auf welcher Skala sie liegen, wurde versucht die tendenziell niedrigen Werte der *Zemanta* API anzugleichen. Für jeden Relevanz Wert

w der von *Zemanta* geliefert wird, wird eine nichtlineare Skalierung in Form von $w = \sqrt{w}$ vorgenommen. Damit soll erreicht werden, dass kleine Werte stärker angehoben werden als hohe Werte. Voraussetzung ist, dass der Wertebereich zwischen 0 und 1 bestehen bleibt. Die Annäherung der Werte wird in Abb. 4.2 grafisch dargestellt. In Abb. 4.2(a) ist ersichtlich, dass durch diese Berechnung kleine Werte mehr verstärkt werden als hohe Werte. Die Darstellung in Abb. 4.2(b) zeigt den Einfluss auf tatsächlich von *Zemanta* vergebene Werte (s. Tab. 4.1).

Ob durch die Kombination der Ergebnisse der verschiedenen Semantic APIs und den Einfluss der Skalierung tatsächlich ein Mehrwert für die Berechnung der Ähnlichkeit entsteht, soll in der Evaluierung (s. Kapitel 6) geklärt werden.

4.4 Bestimmen der Ähnlichkeit

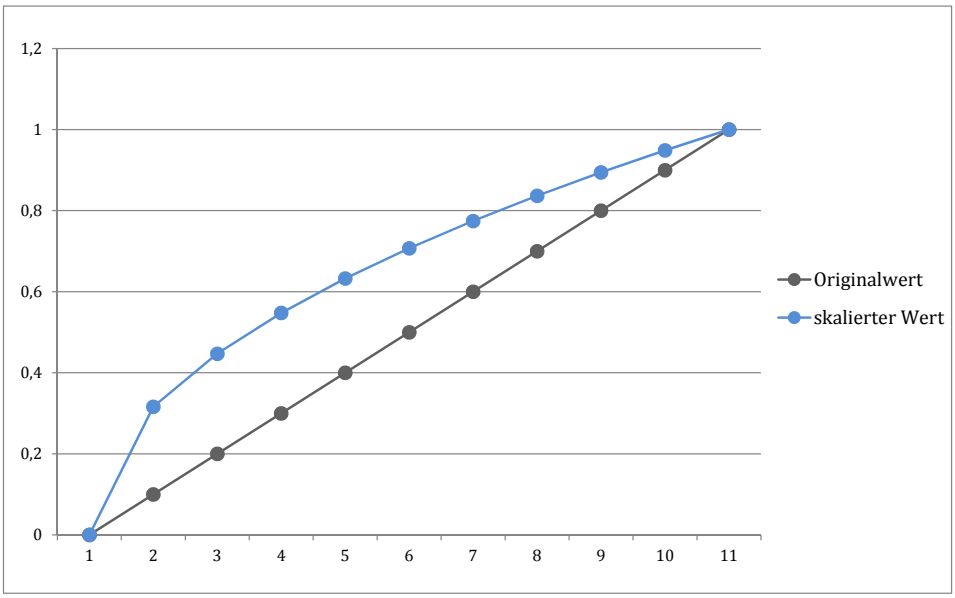
Für die Ermittlung der Ähnlichkeit zwischen zwei Dokumenten stehen nun kollaborative und inhaltsbasierte Daten zur Verfügung, diese sollen als Ausgangspunkt für eine Ähnlichkeitsfunktion dienen, die beide Einflussgrößen kombiniert. Diese Funktion soll verwendet werden um einem Benutzer Inhalte empfehlen zu können, die zu dem Artikel passen den er derzeit betrachtet.

Diese Funktion teilt sich in zwei Berechnungen. Zuerst soll der Zusammenhang der einzelnen Beiträge durch Kollaboration gemessen werden. Für diese Berechnung wird eine Adaption der Normalized Google Distance gewählt (s. Kapitel 3, Gl. 3.3). Diese Formel zur Berechnung der Ähnlichkeit zwischen zwei Termen basiert auf der relativen Häufigkeit des Auftretens eines Terms, respektive beider Terme gemeinsam. Die Werte werden durch die Anzahl der Treffer einer entsprechenden Suche mit der Suchmaschine *Google* erhoben.

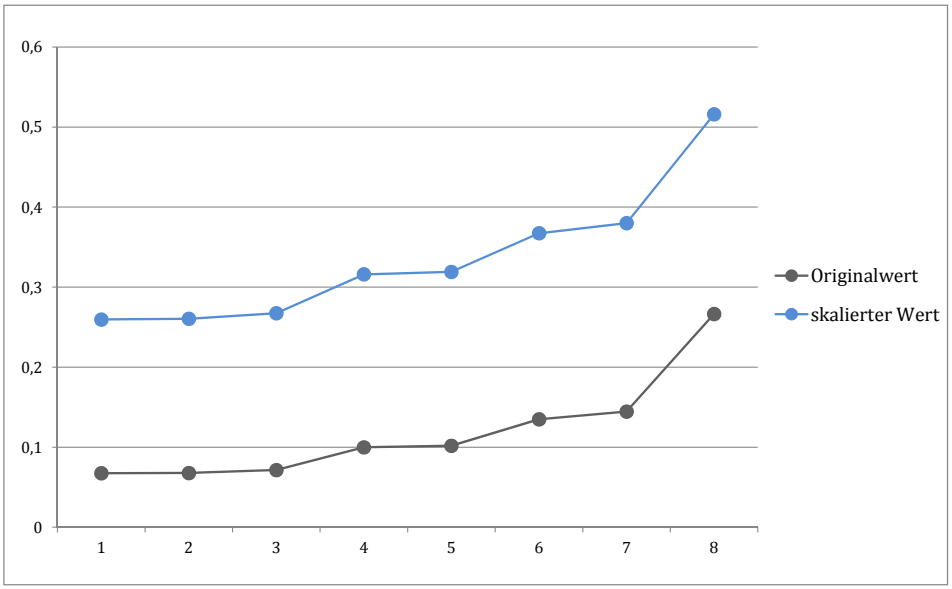
Hier wird sie umgelegt, sodass der Zusammenhang zwischen zwei Artikeln aufgrund der Häufigkeit, wie oft ein Besucher beide Seiten gemeinsam betrachtet, berechnet wird. Für jeden Besuch der Webseite wird, wie in Abschn. 4.2 beschrieben, ermittelt welche einzelnen Beiträge aufgerufen werden. Dadurch kann auch die Häufigkeit bestimmt werden, wie oft einzelne Seiten gemeinsam betrachtet werden. Ausgehend vom Artikel a der momentan vom Benutzer gelesen wird, kann dadurch der Zusammenhang für jeden Kandidaten k_i, in Form von

$$sim_{nd}(a, k_i) = 1 - \frac{\max\{\log f(a), \log f(k_i)\} - \log(a, k_i)}{\log N - \min\{\log f(a), \log f(k_i)\}}, \qquad (4.1)$$

berechnet werden. Die Anzahl der Besuche, in der a gelesen wurde wird von $f(a)$ definiert, sowie von $f(k_i)$ die Häufigkeit wie oft der Kandidat aufgerufen wurde. Die Häufigkeit der gemeinsamen Aufrufe beider Artikel wird durch $f(a, k_i)$ ausgedrückt. In [4] wird bezüglich der *NGD* angeführt, dass für N jeder Wert der größer ist als die Häufigkeit des ersten Terms und der gesamten



(a) Darstellung der nichtlinearen Eigenschaft der Skalierung.



(b) Darstellung der Werte von *Zemanta* API aus 4.1 und ihrer Anpassung.

Abbildung 4.2: Visualisierung der Werteannäherung durch eine nichtlineare Skalierung.

Anzahl der von *Google* indizierten Seiten gewählt werden kann. Daher wird hier als Normierungsfaktor N die Anzahl aller Seitenaufrufe herangezogen.

Um Ähnlichkeiten von Texten messen zu können, werden diese oft als Vektoren repräsentiert, dieser Ansatz soll auch hier zum Einsatz kommen. Sei $D = \{d_1, ..., d_n\}$ die Menge der Dokumente im System und $T = \{t_1, ..., t_m\}$ die Menge der Terme die durch die Semantic APIs gefunden wurden. Für jedes Dokument $d_i \in D$ kann zu jedem Term $t_k \in T$ durch die Bewertung der APIs ein Gewicht w_i, k zugeordnet werden. So lassen sich die Gewichte eines Dokuments zu einem Vektor zusammenfassen, $\vec{w_i} = (w_1, ..., w_m)$.

Die Ähnlichkeit der Vektoren wird durch das Cosinus-Maß ermittelt. Der Vektor des aktuell betrachteten Artikels $\vec{a}$ enthält die Relevanz Werte der Terme die im Text identifiziert wurden und wird mit den einzelnen Dokumentenvektoren der Kandidaten $\vec{k}$ verglichen. Der Wert für $sim_{\cos}(\vec{a}, \vec{k})$ wird nach der in Abschn. 2.1.3 angeführten Formel (s. Gl. 2.2) berechnet.

Die beiden Ähnlichkeitsmaße sollen nun zusammengeführt werden. Wiederrum für den Artikel a und einen Kandidaten k_i wird die gemeinsame Ähnlichkeit aus den kollaborativen Daten und der Textanalyse durch die gewichtete Summe der beiden einzelnen Ähnlichkeitsmaße berechnet:

$$sim(a, k_i) = sim_{nd}(a, k_i) \cdot w_1 + sim_{\cos}(a, k_i) \cdot w_2 \tag{4.2}$$

Durch die Gewichte w_1 und w_2 kann die Einflussnahme der einzelnen Maße geregelt werden. Es gilt $w_1 + w_2 = 1$. Werden beide Werte auf 0.5 gesetzt, werden auch beide Ähnlichkeiten gleich stark berücksichtigt. Setzt man den Wert w_1 auf 0, wird der kollaborativen Ähnlichkeit keine Bedeutung zugemessen, dies gilt analog auch für den Wert w_2 und die inhaltsbasierte Ähnlichkeit.

4.5 Einbettung in ein Content Management System

Als Rahmen für die Entwicklung wurde ein Content Management System (CMS) gewählt. Da der Fokus des umgesetzten Systems auf Webseiten mit großem textuellem Umfang liegt, wie es beispielsweise bei Online-Zeitungen der Fall ist, bietet ein CMS gute Unterstützung, da es die Verwaltung und Wartung der Inhalte erleichtern. *Drupal*[2] ist ein frei verfügbares Open Source CMS das der GNU General Public License (GPL) unterliegt. Die Grundfunktionalität des Systems kann durch Module erweitert werden. Laut einer Statistik über CMS Nutzung[3] liegt *Drupal* weltweit an dritter Stelle. Bereits eine Basisinstallation von *Drupal* ermöglicht es Textinhalte zu erstellen und zu veröffentlichen und bietet daher einen passenden Rahmen für die Umsetzung eines Empfehlungssystems.

[2] http://drupal.org/ – abgerufen am 14. Mai 2013

[3] W3Techs – Web Technology Surveys http://w3techs.com/technologies/overview/content_management/all – abgerufen am 13. September 2013

Kapitel 5

Implementierung

Dieses Kapitel beschreibt die tatsächliche Umsetzung des Empfehlungssystems basierend auf dem im vorhergehenden Kapitel beschriebenem Konzept. Zuerst wird ein kurzer Überblick über die Implementierung für das CMS *Drupal* gegeben. Danach wird die Erhebung der kollaborativen Daten durch Webanalyse Werkzeuge erklärt. Außerdem wird beschrieben wie die Textanalyse durch die Semantic APIs eingebunden wird und wie schließlich die Empfehlungen generiert werden.

5.1 Drupal Modul Entwicklung

Drupal ist in der Skriptsprache PHP geschrieben und verwendet als empfohlenes Datenbanksystem MySQL. Das System besteht aus einem Kern der die Grundfunktionalität bietet und durch verschiedene Module erweitert werden kann wodurch das System sehr flexibel und individuell adaptierbar ist. Zur Zeit der Entwicklung des hier beschriebenen Moduls ist die aktuelle Version 7.19. Es wurde das Modul *Semantic Similarity (semsim)* entwickelt, dass die Funktionalität eines Empfehlungssystems, basierend auf kollaborativen und inhaltsbasierten Daten bietet.

Ein *Drupal* Modul besteht aus mindestens zwei Dateien, diese müssen jeweils mit dem maschinenlesbaren Modulnamen (hier `semsim`) benannt sein. Die Datei mit der Endung `.info` enthält allgemeine Metadaten zum Modul, wie den Namen, eine Beschreibung, die *Drupal* Version auf der das Modul basiert oder Abhängigkeiten zu anderen Modulen. Diese Angaben werden im Backend des CMS auf der Modulübersichtsseite angezeigt, die zur Aktivierung undDeaktivierung der einzelnen Module dient. Es wird im Format von Initialisierungsdateien geschrieben, die Angaben werden dementsprechend als Wertepaare definiert. In der Datei mit der Endung `.module` liegt der eigentliche Modul Code. Optional kann von einem Modul auch eine Datei mit der Endung `.install` bereitgestellt werden. Diese wird bei der erstmaligen Aktivierung eines Moduls in einer *Drupal* Instanz ausgeführt und

wird daher verwendet um in den Set-up Vorgang einzugreifen. Hier können beispielsweise Modul spezifische Datenbanktabellen und -felder angelegt werden. `.module` und `.install` Dateien verlangen keine eigene Syntax, sondern sind herkömmliche PHP Dateien mit adaptierten Dateiendungen.

Drupal ermöglicht Modulen, den Aufruf von Funktionen und Zugriff auf alle Variablen des Cores. Die Interaktion zwischen Modul und Core passiert durch Hooks. Während der Abarbeitung der einzelnen Aufgaben des *Drupal* Cores werden an verschiedenen Punkten die Dateien aller aktivierten Module nach entsprechenden Hooks durchsucht und diese ausgeführt. Jeder Hook muss durch den eigenen Modulnamen ausgewiesen werden, wird beispielsweise von einem Benutzer die „Hilfe“ Seite im Backend aufgerufen, werden alle Module nach der Implementierung der `modulename_hook($path, $arg)` Funktion durchsucht und diese entsprechend abgearbeitet, so kann für jedes Modul eine eigene Hilfestellung für den Benutzer angezeigt werden. Inhaltselemente, wie beispielsweise Artikel, werden in der *Drupal* Umgebung generell als Nodes bezeichnet.

5.2 Einbindung der Webanalyse Metriken

Für die Erhebung der kollaborativen Daten wird die Open Source Webanalyse Plattform *Piwik*[1] verwendet. *Piwik* bietet weniger Funktionalität als die populärere Plattform *Google Analytics*, bringt jedoch den großen Vorteil der dezentralen Datenspeicherung mit sich. Die *Piwik* Installation und die Speicherung der Logdaten erfolgt am eigenen Server, wodurch die Gefahr eines Missbrauchs sensibler Daten minimiert wird. Der Funktionsumfang ist für die hier beschriebene Anwendung durchaus ausreichend.

Die Anforderungen für die Installation von *Piwik* sind sehr grundlegend, ein Webserver beispielsweise Apache, PHP Version 5.1.3 oder neuer und MySQL Version 4.1 oder neuer. Eine aktuelle Version von *Piwik* kann von der Webseite herunter geladen werden und wird in einem beliebigen Verzeichnis am Server abgelegt. Wenn die URL des Verzeichnisses erstmal in einem Browser aufgerufen wird, beginnt der Installationsvorgang von *Piwik* durch den der Benutzer geführt wird. Zur Administration der eigenen *Piwik* Installation wird ein Backend zur Verfügung gestellt, hier können Webseiten, die am gleichen Server liegen wie die *Piwik* Installation, angegeben werden die in die Webanalyse miteinbezogen werden sollen. Im Backend werden die gesammelten Daten für den Benutzer grafisch ausgegeben.

Piwik ermöglicht auch die Daten in eigenen Applikationen zu verwenden dazu dient die *Piwik Analytics* API, die Interaktion ist durch einen Representational State Transfer (REST) Service möglich. Für die Integration von *Piwik* in das CMS *Drupal* steht ein Modul zur Verfügung, indem die nötigen Konfigurationen vorgenommen werden können. *Piwik* vergibt für jede zu

[1] http://piwik.org/ – abgerufen am 20. August 2013

analysierende Webseite eine eigene interne ID, diese muss beispielsweise im Konfigurationsbereich von *Drupal* hinterlegt werden, außerdem die URL des Verzeichnisses in dem die *Piwik* Dateien liegen. Diese Angaben sind auch für einen REST Aufruf der Analytics API erforderlich, um Redundanz zu vermeiden baut das Modul *semsim* auf diesem Modul auf. Für die Nutzung von *semsim* muss daher das Modul *Piwik*[2] installiert und aktiviert sein.

Piwik stellt eine Vielzahl von REST Methoden zur Verfügung wodurch unterschiedliche Daten ausgelesen werden können. Statistiken über Besucher und Seitenaufrufe werden in Echtzeit verfolgt und gespeichert, diese können mithilfe der Methode `Live.getLastVisitsDetails` abgerufen werden, und stellen die wichtigste Basis für die Erhebung der kollaborativen Daten im Modul *semsim* dar. *Drupal* ermöglicht die Durchführung von sich regelmäßig wiederholenden Aufgaben durch die Methode `hook_cron()`. Dieser wird bei jedem Cron Lauf abgearbeitet, Standardeinstellung für den Zeitabstand in dem der Cron von *Drupal* ausgeführt wird, sind 3 Stunden. Dies kann jedoch vom Administrator der Webseite konfiguriert werden. Der Zeitpunkt der letzten Cron Ausführung kann durch Abruf einer *Drupal* internen persistenten Variable festgestellt werden.

Der Aufruf der REST Methode `Live.getLastVisitsDetails` der *Piwik Analytics* API passiert im Modul *semsim* in der Implementierung des Hooks `semsim_cron()`. Die URL setzt sich aus folgenden Parametern zusammen:

```
URL = (URL des Verzeichnisses der Piwik Installation)
idSite = (Piwik interne ID der Webseite)
period = range
date = (Zeitpunkt der letzten Cron Ausführung), today
module = API
method = Live.getLastVisitsDetails
format = JSON
```

Durch die Angabe des Parameters `period=range` werden alle Daten abgerufen die zwischen den beiden Zeitpunkten im Parameter `date` liegen, hier findet die Abfrage immer zwischen dem Zeitpunkt des letzen Cron Laufs und der aktuellen Zeit statt. Als Antwort Format könnte auch XML oder CSV gewählt werden. Es folgt ein beispielhafter Auszug aus den zurückgelieferten Daten dieses REST Aufrufs, diese beziehen sich auf einen Besuch der Seite:

```
"idSite" : "1",
"idVisit" : "4",
"visitorType" : "returning",
...
"actions" : "2",
"actionDetails" : [
  {"type":"action",
    "url":"http:\/\/newsexample\/",
    "pageTitle":"News",
    "pageIdAction":"2",
```

[2] https://drupal.org/project/piwik – abgerufen am 13. September 2013

```
        "pageId":"67",
        "serverTimePretty":"Tue 13 Aug 13:07:30",
        "timeSpent":"41",
        "timeSpentPretty":"41s",
        "icon":null},
        {"type":"action",
        "url":"http:\/\/newsexample\/node\/11",
        "pageTitle":"Syria crisis: France, US and UK want 'strong' UN
        resolution | News",
        "pageIdAction":"38",
        "pageId":"68",
        "serverTimePretty":"Tue 13 Aug 13:08:11",
        "timeSpent":"69",
        "timeSpentPretty":"69s",
        "icon":null}
    ],
    ...
```

Die Ergebnisse werden einzeln abgearbeitet und die für die Ähnlichkeitsberechnung relevanten Informationen in der Datenbank abgelegt. Die einzelnen Besuche werden durch die `idVisit` unterschieden, diese wird bei jedem Besuch, auch wenn der Benutzer die Webseite schon einmal aufgerufen hat, neu vergeben. Es werden nur Besuche von einzelnen Artikeln betrachtet, nicht wenn beispielsweise die Homeseite geöffnet wird. Zu jeder Aktion die der Benutzer gesetzt hat werden die *Durpal* interne ID des Artikels und die Zeitspanne die der Benutzer mit der Betrachtung des Artikels verbracht hat in der Datenbank abgelegt. Außerdem werden die einzelnen Aktionen nummeriert um nachvollziehen zu können in welcher Reihenfolge die Artikel betrachtet wurden.

5.3 Umsetzung der Textanalyse

Wird ein neuer Artikel gespeichert erfolgt automatisch eine Analyse des Textes durch die Semantic APIs *Zemanta*, *OpenCalais* und *Alchemy*. Dies passiert durch die Implementierung der Funktion `hook_node_presave($node)` im *semsim* Modul. Für Inhalte die bereits vor der Installation des Moduls erstellt wurden, kann die Analyse durch den Administrator initialisiert werden.

Die Klasse `SemsimAnalyzer.php` implementiert die Funktionalität für die Analyse durch die Semantic APIs. Die Abwicklung der Interaktion mit den einzelnen Webservices der APIs wurde separat in eigene Klassen ausgelagert. Für die Analyse werden, der Titel und der Bodytext eines Artikels herangezogen und an die jeweiligen Webservice Methoden der APIs weitergegeben. Es folgen Informationen zu den einzelnen Methoden Aufrufen der APIs:

Zemanta API: Die Methode `zemanta.suggest` liefert als Antwort alle Metadaten die von der API angeboten werden, also Artikel, Keywords, Bilder, Links und optional Kategorien [25].

OpenCalais API: Auch der Webservice von *OpenCalais* bietet die Möglichkeit alle Metadaten durch eine Anfrage zu erhalten. Standardmäßig werden keine Social Tags geliefert, dies kann jedoch durch setzen des Parameters `enableMetadataType=SocialTags` beeinflusst werden [24].

Alchemy API: Um Konzepte, Entitäten und Keywords der *Alchemy* API zu erhalten gibt es keine kombinierte Methode. Hier sind drei getrennte Aufrufe notwendig [22].

Für die Nutzung der Webservices ist jeweils ein API Key erforderlich, den man auf den einzelnen Webseiten anfordern kann. Die API Keys können im *Drupal* Backend im Konfigurationsbereich des Moduls angegeben werden. Jede der APIs bietet, neben anderen Formaten, auch die Rückgabe der Ergebnisse im JSON Format an, diese Option wurde für das Modul *semsim* gewählt.

Nach dem Ausführen der Webservice Aufrufe werden die Ergebnisse in der Datenbank abgelegt. Es werden Name, Relevanz Wert und die Quelle des Terms gespeichert, die Unterscheidung der verschiedenen Typen, beispielsweise Keyword oder Entität wird nicht länger berücksichtigt.

5.4 Generierung der Informationsempfehlungen

Basierend auf den erhobenen Daten durch die Textanalyse und der Beobachtung des Benutzerverhaltens, können nun Empfehlungen erstellt werden. Ruft ein Benutzer einen Artikel der Webseite auf, wird die Ermittlung der Ähnlichkeit zu anderen Artikeln initialisiert, um Empfehlungen anzeigen zu können. Die Berechnung der Werte, wird in der Klasse `SemsimSimilarity Calculator.php` vorgenommen. Wie in Abschn. 4.4 beschrieben, passiert dies in zwei Schritten. Zuerst werden die entsprechenden Daten aus der Datenbank ausgelesen. Sind für den aktuell angezeigten Artikel keine Daten gespeichert, beispielsweise weil noch keine Textanalyse durchgeführt wurde oder der Artikel sehr neu ist und daher noch keine kollaborativen Daten zur Verfügung stehen, kann auch keine Ähnlichkeit berechnet werden.

Sind Daten vorhanden, wird zuerst die kollaborative Ähnlichkeit nach Gl. 4.1 ermittelt. Aus der Datenbank werden sortiert nach den individuellen IDs der Besuche, die Häufigkeiten der Seitenaufrufe der einzelnen Artikel aus der gesamten Dokumentensammlung ausgelesen. Daraus kann die Häufigkeit der Aufrufe des aktuell betrachteten Artikels $f(a)$, jedem Kandidaten $f(k_i)$ und die gemeinsame Häufigkeit, respektive wie häufig ein Benutzer innerhalb eines Besuchs beide Seiten aufgerufen hat $f(a, k_i)$, ermittelt werden. Diese Werte werden in Gl. 4.1 eingesetzt und so der Grad der Ähnlichkeit $sim_{nd}(a, k_i)$ für jeden Kandidaten bestimmt.

Im zweiten Schritt wird die inhaltsbasierte Ähnlichkeit erhoben. Es wird sowohl für den aktuell betrachteten Artikel a, als auch für alle weiteren Dokumente k_i, ein Dokumentenvektor erstellt. Die Einträge in den Vekto-

ren entsprechen den Relevanz Werten die durch die Textanalyse erhoben wurden. In Alg. 5.1 wird der Vorgang Erstellung der Dokumentenvektoren dargestellt. Anschließen kann die Ähnlichkeit $sim_{\cos}(\vec{a}, \vec{k_i})$ zwischen dem aktuell betrachteten Artikel und jedem Kandidaten, wie in Gl. 2.2 definiert, berechnet werden.

Algorithmus 5.1: Bilden der Dokumentenvektoren.

```
1:  CreateVector(terms_a, terms_k)
    Bildet die Vektoren der Dokumente aufgrund der Terme die in den Texten identifiziert und bewertet wurden.
2:      vector_a : float                    ▷ für Relevanz Werte aus terms_a
3:      vector_k : float                    ▷ für Relevanz Werte aus terms_k
4:      terms ← combine(terms_a, terms_k)   ▷ Zusammenführung der Termlisten, sodass jeder Term aus terms_a und terms_k einmal vorkommt
5:      for i = 0 to sizeOf(terms) do
6:          if (terms_a contains getWord(terms[i])) then
7:              vector_a[i] ← getValue(terms[i])
8:          else
9:              vector_a[i] ← 0
10:         end if
11:         if (terms_k contains getWord(terms[i])) then
12:             vector_k[i] ← getValue(terms[i])
13:         else
14:             vector_k[i] ← 0
15:         end if
16:     end for
17:     return vector_a, vector_k.
18: end
```

Schließlich erfolgt die Kombination der beiden Ähnlichkeitsmaße durch die gewichtete Summe der Werte nach Gl. 4.2. Bezogen auf den aktuell betrachteten Artikel, steht jeweils eine Liste der Werte beider Ähnlichkeitsmaße zur Verfügung. Der Vergleich basiert auf den darin gespeicherten IDs der einzelnen Artikel, wurden für einen Artikel Werte für beide Maße gefunden und sollen beide Werte einen gleich hohen Einfluss in das kombinierte Maß haben, erfolgt die Berechnung durch:

$$sim(a, k_1) = sim_{nd}(a, k_1) \cdot 0.5 + sim_{\cos}(a, k_1) \cdot 0.5. \tag{5.1}$$

Wird beispielsweise für die kollaborative Ähnlichkeit kein Wert gefunden, weil der Artikel sehr neu ist, wird dieser Wert auf 0 gesetzt und folglich eine Berechung von $sim(a, k_1) = 0 + sim_{\cos}(a, k_1)0.5$ stattfinden, analog gilt das Gleiche, wenn keine inhaltsbasierte Ähnlichkeit erhoben wurde. Die Ergebnisse werden absteigend nach den erzielten Ähnlichkeitswerten gereiht.

Abbildung 5.1: Beispielhafte Ausgabe der Liste von Empfehlungen auf der Webseite. Der Block *"Similar Articles"* wird vom Modul *semsim* bereitgestellt.

Es ist möglich einen Schwellwert für die Ähnlichkeit zu setzen, in diesem Fall muss ein Artikel mindestens diesen Wert der Ähnlichkeit erreichen um in die Ergebnisliste einbezogen zu werden.

Die Ausgabe der Ergebnisse passiert über das *Drupal* Kern Modul Block. Dieses ermöglicht es in einem Modul einen Block mit Inhalten zu definieren, der vom Administrator an einer beliebigen Stelle auf der Seite ausgegeben werden kann, beispielsweise in einer Sitebar (s. Abb. 5.1).

Kapitel 6

Evaluierung

In diesem Kapitel werden die Evaluierungsergebnisse des entwickelten Ähnlichkeitsmaßes beschrieben. Die Evaluierung soll klären welche Semantic API, bzw. welche Kombination die beste Grundlage für das Verfahren bietet und wie gut die Qualität der generierten Empfehlungen ist. Es wurde eine Methode aus dem Information Retrieval (IR) Kontext gewählt. Klassische IR Systeme werden für Suchvorgänge in Dokumentensammlungen eingesetzt. Die Qualität eines solchen Systems wird bewertet, indem gemessen wird, wie viele relevante Dokumente aus der Sammlung für eine gegebene Suchanfrage gefunden werden. Diese Problemstellung ist der Aufgabe eines Empfehlungssystems, wie hier beschrieben, ähnlich. Ausgehend von einem Artikel, den der Benutzer zur Zeit betrachtet, sollen weitere Artikel aus der gesamten Dokumentensammlung gefunden werden, die diesem ähnlich sind. Die Qualität des Empfehlungssystems hängt nicht nur, von Genauigkeit und Größe der berechneten Ähnlichkeit zwischen den einzelnen Texten ab, sondern vor allem von der Ergebnismenge die es liefert.

6.1 Evaluierungsmethode

In [7] werden empirische Methoden zur Ermittlung der Effektivität eines IR Systems beschrieben. Eine Herausforderung bei der Bewertung der Ergebnisse eines Systems ist, dass die richtige Antwort bekannt sein muss. In einer bestimmten Dokumentensammlung und zu einer gegebenen Anfrage, muss die Menge der relevanten Dokumente definiert sein, dies wird auch als Goldstandard bezeichnet. In der Praxis wird die Relevanz der einzelnen Dokumente durch Personen eingeschätzt und diese durch verschiedene Evaluierungsmaße mit den Ergebnissen des Systems verglichen.

6.1.1 Evaluierungsmaße

Zwei häufig verwendete Maße zur Evaluierung von IR Systemen sind *Recall* (r) und *Precision* (p) [7]. Grundlage für die Maße sind die Mengen einer Suchanfrage, für eine Anfrage über die Dokumentensammlung D wird vom System eine Ergebnismenge A geliefert. Diese wird mithilfe der bekannten relevanten Dokumente, dem Goldstandard G bewertet. Daraus ergibt sich in den meisten Fällen eine Situation wie sie in Abbildung 6.1 dargestellt wird.

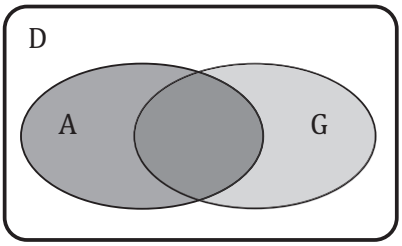


Abbildung 6.1: Darstellung der Mengen einer Suchanfrage, D bezeichnet die gesamte Dokumentensammlung, Ergebnismenge A wird vom System geliefert, der Goldstandard G muss definiert werden.

Durch Recall wird die Vollständigkeit der gelieferten Ergebnisse ausgedrückt, es wird der Anteil der relevanten Dokumente angegeben die gefunden wurden. Die Formel zur Berechnung ergibt sich aus der Schnittmenge von A und G und die Mengenmächtigkeiten,

$$r = \frac{|A \cap G|}{|G|}. \tag{6.1}$$

Precision beschreibt die Genauigkeit oder Reinheit der Ergebnismenge. Es wird geprüft wie viele Dokumente aus der Ergebnismenge tatsächlich relevant sind. Auch dieser Wert lässt sich durch die Mengenmächtigkeiten definieren,

$$p = \frac{|A \cap G|}{|A|}. \tag{6.2}$$

Recall und Precision können Werte zwischen 0 und 1 annehmen, bei beiden Maßen sind höhere Werte besser. Ein Recall Wert von 1 bedeutet, dass alle relevanten Dokumente gefunden wurden, respektive ist der Wert 0 wenn kein relevantes Dokument geliefert wurde. Ein Precision Wert von 1 wird erreicht wenn die Ergebnismenge nur relevante Dokumente enthält.

Die beiden Maße betrachten die Ergebnismenge aus zwei verschiedenen Blickwinkeln und sind in gewisser Weise gegenläufig. Wenn ein System alle Dokumente D für eine Anfrage zurückliefert, erreicht es dadurch einen Recall Wert von 1, die Precision wird jedoch sehr niedrig ausfallen, außer es sind wirklich alle Dokumente auch relevant. Wird nur ein einziges relevantes Dokument zurückgeliefert ergibt sich für Precision der Wert 1, wenn es jedoch

mehr relevante Dokumente gibt, ist der Wert für Recall sehr niedrig. Werden Recall und Precision verwendet um Systeme miteinander zu vergleichen, kann nur dann definitiv von einer Überlegenheit eines Systems gesprochen werden, wenn es für beide Maße bessere Werte erreicht.

Zur Kombination von Recall und Precision wird häufig das *F_β-Maß* eingesetzt, dieses basiert auf dem in [17] vorgestellten Effektivitätsmaß. Der positive Parameter β bezeichnet eine Gewichtung, durch hohe Werte würde der Einfluss von Recall größer werden, niedrige Werte stärken die Precision. In der allgemeinen Beurteilung wird meist eine Gleichgewichtung der beiden Maße angewendet, $\beta = 1$, diese Variante wird als *F_1-Maß* (F_1) bezeichnet und wird durch folgende Formel ausgedrückt,

$$F_1 = \frac{2 \cdot r \cdot p}{p + r}. \tag{6.3}$$

Um aussagekräftige Evaluierungsergebnisse zu erhalten, müssen mehrere Suchanfragen getestet werden und die Werte gemittelt werden. Dazu stehen zwei unterschiedliche Arten der Mittelwert Berechnung zur Verfügung [7]. Die *Makrobewertung*, bildet das arithmetische Mittel über alle Werte von Precision und Recall. Es wird der Mittelwert über alle Suchanfragen gebildet und jeder Wert geht gleich in die Bewertung ein, daher wird dieser Wert auch als Nutzer-orientiert bezeichnet. Die Berechnung für Recall erfolgt durch die Formel,

$$r_{makro} = \frac{1}{N} \cdot \sum_{i=1}^{N} \frac{|A_i \cap G_i|}{|G_i|}. \tag{6.4}$$

Bei der *Mikrobewertung* werden über alle Suchanfragen hinweg die Anzahl der relevanten und nicht relevanten Dokumente summiert und über diese Werte Recall und Precision berechnet. Da die Bewertung auf den relevanten Dokumenten über alle Anfragen hinweg basiert, gilt diese Mittelwertbildung als System-orientiert. Die Mikrobewertung des Recall Werts wird in Form von

$$r_{mikro} = \frac{\sum_{i=1}^{N} |A_i \cap G_i|}{\sum_{i=1}^{N} |G_i|} \tag{6.5}$$

berechnet. Die Formeln können analog auch für die Berechnung der Mittelwerte von Precision und F_1-Maß adaptiert werden.

6.2 Evaluierungsergebnisse

Zur Evaluierung von IR Systemen stehen Testkollektionen aus vordefinierten Anfragen und Dokumentensammlungen mit bereits annotierter Relevanz zu den einzelnen Anfragen zur Verfügung. Für das hier beschriebene System ist jedoch eine Evaluierung aufgrund von Suchanfragen ungeeignet. Daher wird eine eigene Testkollektion definiert. Es werden 50 Artikel aus verschiedenen

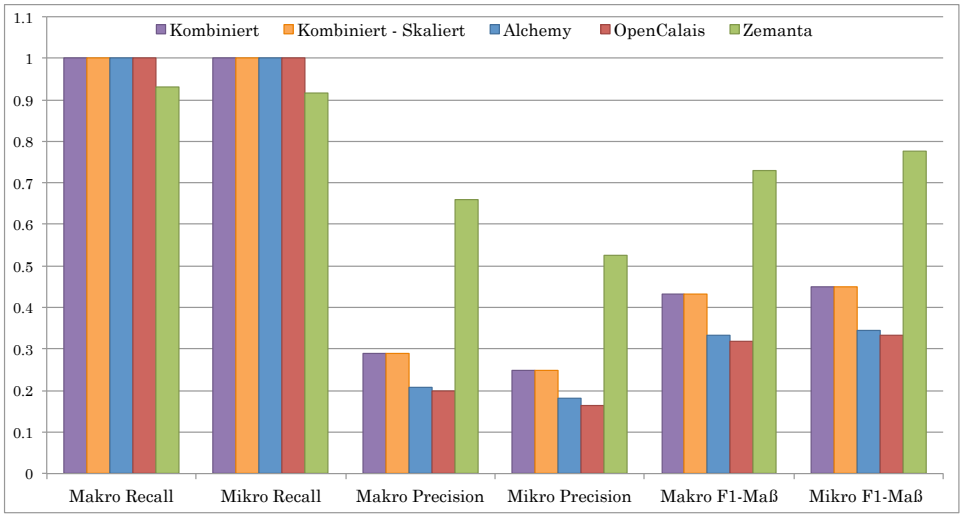


Abbildung 6.2: Mittelwerte der Evaluierungsmaße für Vollständigkeit und Genauigkeit des Systems ohne Einschränkung der Ergebnismenge, für die einzelnen APIs und Kombinationen.

Themenbereichen der BBC-Webseite[1] verwendet. Zu jedem Artikel wird der Goldstandard von relevanten Dokumenten aus der Sammlung händisch annotiert. Um durch die Evaluierung festzustellen welche der Semantic APIs oder Kombination am besten geeignet ist, wird die kollaborative Ähnlichkeit nicht berücksichtigt, sondern nur die inhaltsbasierte Ähnlichkeit der Dokumente betrachtet, andernfalls könnten keine klaren Ergebnisse bezüglich dem Einfluss der APIs erzielt werden.

Im ersten Testdurchlauf wird die Ergebnismenge nicht eingeschränkt, es werden alle Kandidaten gewertet für die ein positiver Ähnlichkeitswert errechnet wird, die Mittelwerte der Ergebnisse werden in Abb. 6.2 dargestellt. Die hohen Recall Werte lassen erkennen, dass das System aufgrund jeder API oder Kombination, in fast allen Fällen fähig ist alle relevanten Dokumente zu identifizieren. Ausnahme ist die Ermittlung basierend auf *Zemanta*, aber auch hier wird ein Recall Wert über 0.9 erzielt. Die Werte für Precision zeigen jedoch, dass die Ergebnismenge zu groß ist, respektive mehr Artikel geliefert werden als im Goldstandard definiert. Bereits in diesem ersten Durchlauf wird auch ersichtlich, dass durch Verwendung der *Zemanta* API bedeutend höhere Precision Werte erzielt werden, als durch die anderen APIs, respektive Kombinationen. Durch setzen eines Schwellwerts für den Ähnlichkeitswert, soll ein ausgeglicheneres Ergebnis erzielt werden.

Es werden Testläufe mit verschiedenen Schwellwerten über die gesamte Dokumentensammlung durchgeführt, Abb. 6.3 zeigt die Entwicklung von

[1] http://www.bbc.co.uk – abgerufen am 16. September 2013

Makro- und Mikromittelwert des F_1-Maßes über die Schwellwerte hinweg. Man sieht, dass die Werte aller APIs und Kombinationen anfänglich sehr mittelmäßig sind und danach stark ansteigen. Grund dafür ist, dass bei einem Schwellwert von 0 die Recall Werte sehr hoch sind, aber niedrige Werte für Precision entstehen. Die besten Werte werden für alle APIs, bei Schwellwerten zwischen 0.1 und 0.4 erreicht. Danach sinken die Werte des F_1-Maßes bei allen APIs und Kombinationen ab. Im Folgenden werden die Ergebnisse der einzelnen APIs und Kombinationen detaillierter betrachtet.

Betrachtet man die Werte der *Alchemy* API, kann man erkennen, dass sie bei einem Schwellwert von 0.1 das höchste Ergebnis über allen APIs und Kombinationen erreicht. In Tab. 6.1 werden die detaillierten Ergebnisse der *Alchemy* API bei einen Schwellwert der Ähnlichkeit von 0.1 dargestellt. Bei höheren Schwellwerten fallen die Ergebnisse jedoch stark ab und liegen deutlich tiefer als die anderen betrachteten F_1-Werte. Daraus kann man ableiten, dass die errechnete Ähnlichkeit zwischen den einzelnen Artikeln basierend auf den Daten der *Alchemy* API generell niedrig ausfällt.

Tabelle 6.1: Ergebnisse der *Alchemy API* bei einem Ähnlichkeitswert von mindestens 0.1.

Alchemy API

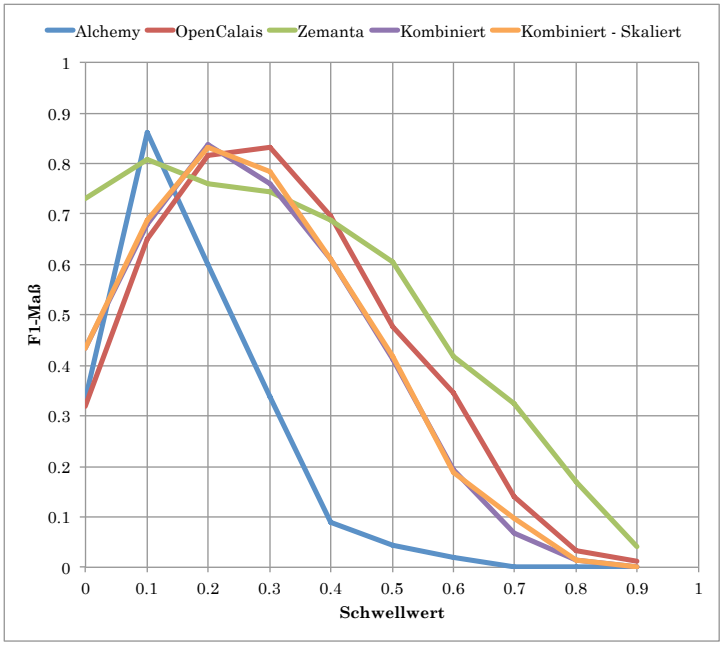
Schwellwert	Maß	Makrobewertung	Mikrobewertung
0.1	Recall	0.835	0.85
	Precision	0.918	1
	F_1-Maß	0.863	0.952

Die *OpenCalais* API liefert bei einem Schwellwert der Ähnlichkeit von 0.3 das höchste Ergebnis für das F_1-Maß (s. Abb. 6.3), auch bei 0.4 sind die Ergebnisse noch sehr gut. Ab einem höheren Schwellwert sinken jedoch auch hier die Werte sehr stark ab. In Tab. 6.2 werden die Ergebnisse für diese beiden Schwellwerte aufgelistet.

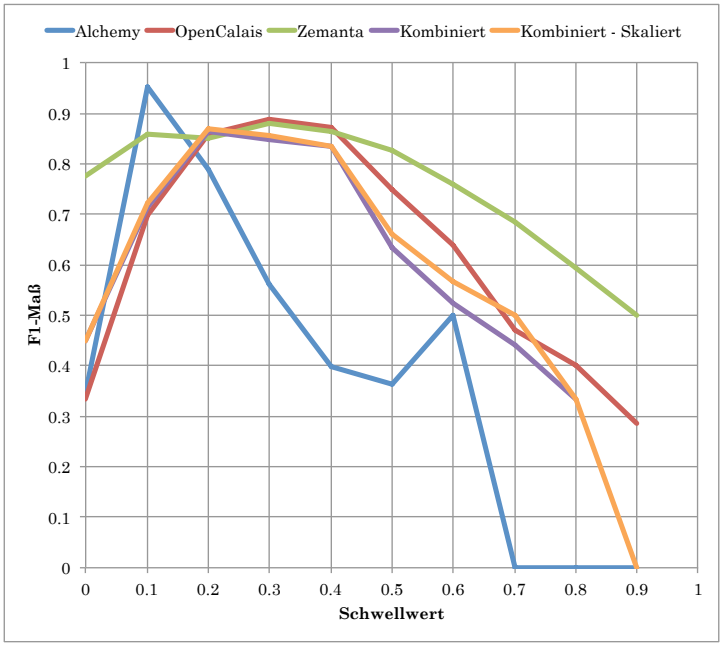
Tabelle 6.2: Ergebnisse der *OpenCalais* API bei Schwellwerten der Ähnlichkeit von 0.3 und 0.4.

OpenCalais API

Schwellwert	Maß	Makrobewertung	Mikrobewertung
0.3	Recall	0.795	0.822
	Precision	0.925	0.926
	F_1-Maß	0.833	0.889
0.4	Recall	0.632	0.673
	Precision	0.816	1
	F_1-Maß	0.695	0.872



(a) Makrobewertung



(b) Mikrobewertung

Abbildung 6.3: Mittelwerte des F_1-Maßes über verschiedene Schwellwerte hinweg.

Betrachtet man die Kurve der F_1-Werte von *Zemanta* (s. Abb. 6.3), kann man erkennen, dass hier die Werte stetiger sind, als bei den anderen APIs und Kombinationen. Auch durch Verwendung dieser API können sehr gut Ergebnisse erzielt werden und diese übertreffen vor allem bei höheren Schwellwerten die weiteren Ergebnisse. Daraus kann man schließen, dass die errechnete Ähnlichkeit der Inhalte basierend auf der *Zemanta* API tendenziell höher ist, als aufgrund der anderen APIs und Kombinationen. Die besten Ergebnisse werden zwischen einem Schwellwert für die Ähnlichkeit zwischen 0.2 und 0.5 erzielt, diese Werte werden in Tab.6.3 aufgelistet.

Tabelle 6.3: Ergebnisse der *Zemanta* API bei Schwellwerten der Ähnlichkeit von 0.2 bis 0.5.

Zemanta API

Schwellwert	Maß	Makrobewertung	Mikrobewertung
0.2	Recall	0.762	0.748
	Precision	0.836	0.86
	F_1-Maß	0.759	0.851
0.3	Recall	0.698	0.673
	Precision	0.868	0.986
	F_1-Maß	0.745	0.879
0.4	Recall	0.637	0.589
	Precision	0.837	1
	F_1-Maß	0.688	0.864
0.5	Recall	0.546	0.505
	Precision	0.776	1
	F_1-Maß	0.605	0.826

Die Ergebnisse der beiden Kombinationen der verschiedenen Semantic APIs sind sehr ähnlich (s. Abb. 6.3). Lediglich bei einem Schwellwert zwischen 0.5 und 0.8 sind die Ergebnisse der Kombinationsvariante nach Skalierung der Relevanz Werte der *Zemanta* API leicht höher als ohne Skalierung. Wobei die Abweichung der F_1-Werte der Makrobewertung noch geringer ausfallen als die der Mikrobewertung. Der Vergleich der Ergebnisse der Kombinationen mit den Werten die basierend auf den einzelnen APIs erzielt werden zeigt, dass Erstere ab einem Schwellwert von 0.2 zwar über der *Alchemy* API liegen, *OpenCalais* und *Zemanta* jedoch durchgängig bessere Ergebnisse liefern.

6.3 Diskussion

Die Evaluierung der Empfehlungen die das System liefert zeigt, dass durch das eingesetzte Ähnlichkeitsmaß, unabhängig von der verwendeten Semantic

API oder Kombination als Grundlage, alle relevanten Dokumente aus der Sammlung identifiziert werden können. Dies ist jedoch stark vom Schwellwert abhängig der die Ergebnismenge begrenzt, indem die berechnete Ähnlichkeit zwischen zwei Artikeln, diesen Wert mindestens erreichen muss. Es zeigt sich, dass die Qualität der Empfehlungen bei einem niedrigen Schwellwert besser ist als bei einem hohen. Man könnte davon ausgehen, dass durch eine niedrige berechnete Ähnlichkeit, viele Dokumente empfohlen werden die nicht relevant sind, dies wird jedoch durch die guten Werte für Precison widerlegt, da dieser Wert Aussage über die Genauigkeit des Systems gibt.

Die beste Qualität der Empfehlungen kann erreicht werden, indem die Metadaten der *Alchemy* API herangezogen werden und ein Schwellwert von 0.1 für die Ähnlichkeit definiert wird. Dies zeigt sich durch die Mittelwerte des F_1-Maßes, von 0.863 (Makrobewertung) bzw. 0.952 (Mikrobewertung) die durch diese Parameter erreicht werden. Das F_1-Maß kombiniert Recall und Precision, wobei beide Maße gleich stark gewichtet werden. Ein hoher Wert zeigt, dass die Ergebnismenge und der entsprechende Goldstandard durchschnittlich sehr stark übereinstimmen, das bedeutet, dass die Ergebnismenge weder viele irrelevante Dokumente, noch zu wenige relevante Dokumente liefert.

Auch durch Verwendung der Metadaten von *OpenCalais* können gute Empfehlungen erreicht werden. Hier liegt der Schwellwert für die Ähnlichkeit zwischen den Artikeln, der zu den besten Ergebnissen führt bei 0.3. Dadurch werden die Mittelwerte von 0.833 (Makrobewertung) und 0.872 (Mikrobewertung) für das F_1-Maß erreicht.

Die *Zemanta* API liefert vor allem bei höheren Schwellwerten bessere Ergebnisse, als die anderen APIs und Kombinationen. Der F_1-Wert fällt ebenfalls bei höheren Schwellwerten, ist jedoch beständiger als die weiteren betrachteten Ergebnisse. Bei einem Schwellwert von 0.5 werden Mittelwerte von 0.605 (Makrobewertung) und 0.826 (Mikrobewertung) für das F_1-Maß erzielt, dies kann immer noch als überdurchschnittlich bezeichnet werden. Betrachtet man jedoch wie diese F_1-Werte zustande kommen, erkennt man, dass die Mittelwerte für Recall nur bei rund 0.5 liegen, also in etwa nur die Hälfte der relevanten Dokumente gefunden werden.

Durch die Evaluierung konnte auch geklärt werden, dass eine Kombination der Metadaten der verschiedenen Semantic APIs, im hier beschriebenen Zusammenhang, keinen Mehrwert bringt. Vergleicht man die beiden Kombinationsvarianten miteinander zeigt sich, dass die Ergebnisse kaum Unterschiede aufweisen. Da *Zemanta* tendenziell die Relevanz der identifizierten Metadaten niedrig bewertet, wurde bei einer Kombinationsvariante versucht, die Relevanz Werte durch eine nichtlineare Skalierung anzunähern, wodurch minimal bessere Evaluationsergebnisse erzielt werden konnten.

Um die Genauigkeit und Zuverlässigkeit des kollaborativen Ähnlichkeitsmaßes zu erheben, sollte eine Studie über einen längeren Zeitraum erfolgen, in dem Benutzer mit dem System arbeiten. Da hier die Anzahl der Benutzer

und Häufigkeiten der einzelnen Seitenaufrufe eine große Rolle spielen, wäre es auf diese Weise möglich repräsentative Testdaten zu erheben und die Qualität des Ähnlichkeitsmaßes zu bewerten.

Kapitel 7

Zusammenfassung

Im Zuge dieser Arbeit wurden Herangehensweisen an die Messung von Ähnlichkeiten betrachtet. Im Bereich der Dokumentenähnlichkeit geht die Tendenz zur Erhebung dieser Werte immer mehr in Richtung semantischer Ähnlichkeit. Syntaktische Ähnlichkeitsmaße werden für die Betrachtung der Ähnlichkeit zwischen einzelnen Worten eingesetzt, für die Betrachtung von ganzen Texten jedoch kaum. Für die Bestimmung von semantischer Ähnlichkeit ist es notwendig Wissen über die natürliche Sprache zu haben. Dieses kann aus großen Textsammlungen gewonnen werden, wie beispielsweise *WordNet* oder anderen lexikalischen Datenbanken. Immer populärer werden auch Ansätze die *Wikipedia* verwenden um Merkmale in Texten zu finden, dieses Vorgehen ist sehr vielversprechend, da *Wikipedia* zur größten Online-Enzyklopädie herangewachsen ist, ständig überarbeitet und erweitert wird und Informationen zu vielen Wissensbereichen enthält.

Um Daten aus einer Textsammlung zu gewinnen sind verschiedene Vorarbeiten aus dem Bereich Natural Language Prossecing (NLP) notwendig, das führt auch zu einem hohen Rechenaufwand. Semantic APIs wenden ebenfalls Verfahren aus NLP an, um Terme zu identifizieren die einen Text charakterisieren und diese Terme im Bezug auf das Dokument zu bewerten. Daraus entstand die Idee diese APIs, als Basis für die Berechnung von Ähnlichkeiten in Texten heranzuziehen.

Unabhängig von der Quelle der Ähnlichkeitsmerkmale bringen vektorbasierte Verfahren zur Ermittlung der Ähnlichkeit zwischen zwei Dokumenten sehr gute Ergebnisse. Durch die Repräsentation der Dokumente als Vektoren kann ihre Ähnlichkeit durch verschiedene Funktionen aus der Vektortheorie gemessen werden. Auch im hier beschriebenen System wurde ein vektorbasierter Ansatz in Verbindung mit den Relevanz Bewertungen der einzelnen Terme durch die Semantic APIs gewählt. Der Wert für die inhaltsbasierte Ähnlichkeit wird durch das Cosinus-Maß bestimmt, dabei wird die Ähnlichkeit von zwei Vektoren durch den Cosinus ihres eingeschlossenen Winkel berechnet.

Als zweite Größe für die Ähnlichkeit zwischen Dokumenten wurden kollaborative Daten betrachtet. Im WWW werden häufig Informationen über das Benutzerverhalten erhoben um Gemeinsamkeiten zwischen einzelnen Benutzern zu definieren und daraus individuelle Vorlieben abzuleiten. So kann das Angebot besser an einen individuellen Benutzer angepasst werden. Die meist verbreitete Herangehensweise ist hier das Kollaborative Filtern, vor allem in E-Commerce Anwendungen. In dieser Arbeit wurden Webanalysetools eingesetzt um die kollaborativen Daten zu erheben, dies ist keine konkurrenzfähige Alternative zum kollaborativen Filtern, da weniger umfangreiche Daten erhoben werden können. Für die Verwendung zur Empfehlung von Beiträgen, beispielsweise in einer Online-Zeitung reichen diese Metriken jedoch aus.

Basierend auf den so erhobenen kollaborativen Daten, im Besonderen der Häufigkeiten von Seitenaufrufen durch individuelle Benutzer und einem vektorbasierten Textähnlichkeitsmaß wurde im Zuge dieser Arbeit ein Empfehlungssystem für Beiträge einer Webseite entwickelt. Ruft ein Benutzer eine Seite auf, wird der Zusammenhang mit anderen Seiten im gesamten System erhoben. Werden Dokumente gefunden die zum Inhalt der Seite ähnlich sind gibt das System Empfehlungen in Form von Verlinkungen aus, in der Hoffnung dem Interesse des Benutzers zu entsprechen und ihn durch dieses Angebot zu unterstützen.

Die durchgeführte Evaluierung hat gezeigt, dass durch das gewählte Ähnlichkeitsmaß Empfehlungen von guter Qualität ermittelt werden können. Durch setzen eines Schwellwerts für die Ähnlichkeit kann eine Ergebnismenge erzielt werden die alle relevanten und wenige oder keine nicht relevanten Dokumente enthält. Im Zuge der Evaluierung wurde jedoch vorerst nur die inhaltsbasierte Ähnlichkeit betrachtet, die Bewertung des kollaborativen Ähnlichkeitsmaßes ist noch ausständig und sollte mittels einer Benutzerstudie durchgeführt werden.

Anhang A

Inhalt der CD-ROM

A.1 Masterarbeit (PDF

Pfad: /

masterarbeit.pdf	Informationsempfehlungen durch Kollaboration und Textanalyse

A.2 Projekt

Pfad: /Projekt

semsim.zip	Sourcecode des Drupal Moduls

A.3 Online-Quellen (PDF)

Pfad: /OnlineQuellen

APIEvaluation.pdf . . .	Entity Extraction and Content API Evaluation – Rob DiCiuccio
Alchemy.pdf	Dokumentation Alchemy API
OpenCalais.pdf	Dokumentation OpenCalais API
Zemanta.pdf	Dokumentation Zemanta API

A.4 Evaluierung

Pfad: /Evaluierung

Evaluierungsdaten.xlsx .	Evaluierungs Ergebnisse
Testkollektion/	Artikel der Testkollektion im PDF-Format

Quellenverzeichnis

Literatur

[1] Andrea Bauer. „Analyse von Semantic Web-APIs und deren Methoden“. Fachhochschule Oberösterreich, Campus Hagenberg, 2011.

[2] Tim Berners-Lee, James Hendler und Ora Lassila. „The Semantic Web“. In: *Scientific American* 284.5 (Mai 2001), S. 34–43.

[3] Michel Capelle u. a. „Semantics-based news recommendation“. In: *Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics*. WIMS '12. Craiova, Romania, 2012, 27:1–27:9.

[4] Rudi L. Cilibrasi und Paul M. B. Vitanyi. „The Google Similarity Distance“. In: *IEEE Transactions on Knowlwledge and Data Engineering* 19.3 (März 2007), S. 370–383.

[5] Scott Deerwester u. a. „Indexing by Latent Semantic Analysis“. In: *Journal of the American Society for Information Science* 41 (1990), S. 391–407.

[6] Fefie Dotsika. „Semantic APIs: Scaling up towards the Semantic Web“. In: *International Journal of Information Management* (Aug. 2010), S. 335–342.

[7] Reginald Ferber. *Information Retrieval. Suchmodelle und Data-Mining-Verfahren für Textsammlungen und das Web*. Dpunkt Verlag, 2003.

[8] Evgeniy Gabrilovich und Shaul Markovitch. „Computing semantic relatedness using Wikipedia-based explicit semantic analysis“. In: *Proceedings of the 20th international joint conference on Artifical intelligence*. IJCAI'07. Hyderabad, India, 2007, S. 1606–1611.

[9] Edward Grefenstette. „Analysing Document Similarity Measures“. University of Oxford, Computing Laboratory, 2009.

[10] Brian Harrington. „A semantic network approach to measuring relatedness“. In: *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. COLING '10. Beijing, China, 2010, S. 356–364.

[11] Zellig S. Harris. „Distributional structure“. In: *Word, Journal of the linguisitc circle of New York* 10.23 (1954), S. 146–162.

[12] Marco Hassler. *Web Analytics*. mitp Business. mitp/bhv, 2012.

[13] Aminul Islam und Diana Inkpen. „Semantic text similarity using corpus-based word similarity and string similarity“. In: *ACM Transactions on Knowledge Discovery from Data* 2.2 (Juli 2008), 10:1–10:25.

[14] Paul Jaccard. „The Distribution of the Flora in the Alpine Zone“. In: *New Phytologist* 11 (1912), S. 37 –50.

[15] William P. Jones und George W. Furnas. „Pictures of relevance: A geometric analysis of similarity measures“. In: *Journal of the American Society for Information Science* 38.6 (1987), S. 420–442.

[16] Vladimir I. Levenshtein. „Binary Codes Capable of Correcting Deletions, Insertions and Reversals“. In: *Soviet Physics Doklady* 10 (1966), S. 707 –710.

[17] C. J. Van Rijsbergen. *Information Retrieval*. Newton, MA, USA: Butterworth-Heinemann, 1979.

[18] Gerard M. Salton, Andrew K. C. Wong und Chung-Shu Yang. „A vector space model for automatic indexing“. In: *Communications of the ACM* 18.11 (Nov. 1975), S. 613–620.

[19] J. Ben Schafer u. a. „Collaborative Filtering Recommender Systems“. In: *The Adaptive Web*. Hrsg. von Peter Brusilovsky, Alfred Kobsa und Wolfgang Nejdl. Berlin, Heidelberg: Springer-Verlag, 2007, S. 291–324.

[20] Peter D. Turney. „Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL“. In: *Proceedings of the 12th European Conference on Machine Learning*. EMCL '01. 2001, S. 491–502.

[21] Zhibiao Wu und Martha Palmer. „Verbs semantics and lexical selection“. In: *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*. ACL '94. Las Cruces, New Mexico, 1994, S. 133–138.

Online-Quellen

[22] *AlchemyAPI Documentation*. URL: http://www.alchemyapi.com/api/ (besucht am 14.05.2013).

[23] Rob DiCiuccio. Mai 2010. URL: http://blog.viewchange.org/2010/05/entity-extraction-content-api-evaluation/ (besucht am 16.08.2013).

[24] *OpenCalais – Web Service API*. URL: http://www.opencalais.com/documentation/opencalais-web-service-api/ (besucht am 14.05.2013).

[25] *Zemanta API Documentation*. URL: http://developer.zemanta.com/docs/ (besucht am 14.05.2013).