Empfehlungssysteme im Kontext von Semantic APIs

Sabine Mühlböck

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im September 2013

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 29. September 2013

Sabine Mühlböck

Inhaltsverzeichnis

Kurzfassung

In den letzten Jahren haben sich Benutzer im World Wide Web von passiven Konsumenten zu aktiven Teilnehmern und Erstellern von Inhalten entwickelt. Infolgedessen entstand eine Unmenge an Informationen, die Benutzern einen Überblick über das Angebot an Webseiten zu einem bestimmten Thema erschwert. Bei der Recherche in einer gewissen Materie ist jedoch nicht nur die Menge an Daten ein Problem: Auch die Verifizierung von Inhalten ist oft mühsam, da viele Inhalte im Web nicht wissenschaftlich belegt sind. Um den Benutzer bei Recherchen zu unterstützen, wurde im Zuge dieser Arbeit *ReadBeyond* entwickelt. Dieses Empfehlungssystem gibt weiterführende Verweise auf Webseiten, die zum Kontext der jeweiligen aufgerufenen Seite passen. Der „Forschende“ soll so von der weiterführenden Suche dispensiert werden und nach dem Aufrufen einer Seite nur mehr für die Auswahl einer vorgefertigen Empfehlung, nicht aber für die Suche an sich, verantwortlich sein. Das hybride Empfehlungssystem beinhaltet eine inhaltsbasierte Komponente, deren Kern Semantic APIs sind, und eine kollaborative Komponente, die Benutzerbewertungen miteinbezieht. Dafür wurde ein Algorithmus, der die verschiedenen Ansätze kombiniert, entwickelt und getestet.

Abstract

In recent years, users underwent a development from passive consumers to active participants and content creators. Hence an enormous amount of information arose, whereby users can hardly survey the range of websites on the internet. Whilst researching in a certain field, the verification of the content is also grinding, since a lot content is not scientifically proven. As a result, the recommender system *ReadBeyond* was developed in order to support the user's research. The system recommends links to carry on the research, which are set in the same context as the original website. Thus users are dispensed from the research and only have to choose recommendations without being responsible for the search itself. The hybrid recommender system consists of a content-based component, of which its core part are semantic APIs and the collaborative component, which is formed by the ratings of the users. Consequently, an algorithm that combines both approaches is developed and evaluated.

Kapitel 1

Einleitung

In den letzten Jahren war das World Wide Web gekennzeichnet durch die Partizipation von aktiven Benutzern. Im Vergleich dazu waren Benutzer zuvor eher passive Konsumenten von Informationen. Schon 1980 prophezeite Toffler [44] den Aufstieg der *prosumer*, eine Fusion aus *producers* (Produzenten) und *consumers* (Konsumenten). Unter anderem durch Blogs und Wikis war es nun im *Web 2.0* möglich, Inhalte auch als Laie der breiten Masse zur Verfügung zu stellen, ohne selbst Entwicklerkenntnisse zu besitzen. Infolgedessen entstand eine Unmenge an Informationen, die durch die Beteiligung vieler *prosumer* veröffentlicht wurde. Benutzern wiederum fällt es dadurch schwieriger einen Überblick über das Angebot an Webseiten zu einem bestimmten Thema zu bekommen. Bei der Recherche in einer gewissen Materie ist jedoch nicht nur die Menge an Daten ein Problem: Auch die Verifizierung von Inhalten ist oft mühsam, da viele Inhalte im Web (vor allem in Blogs etc.) nicht wissenschaftlich belegt sind. Ist man für Studienarbeiten und dergleichen auf der Suche nach hochwertigen Quellen, kann sich dies schnell als erschwerlich herausstellen.

Um den Benutzer bei Recherchen zu unterstützen, soll ein Empfehlungssystem entwickelt werden, das weiterführende Verweise auf Webseiten gibt, die zum Kontext der jeweiligen aufgerufenen Seite passen. Der „Forschende“ soll so von der weiterführenden Suche dispensiert werden und nach dem Aufrufen einer Seite nur mehr für die Auswahl einer vorgefertigen Empfehlung, nicht aber für die Suche an sich, verantworlich sein. Das System selbst soll hybride Eigenschaften aufweisen und sowohl die Stärken eines inhaltsbasierten als auch eines kollaborativen Empfehlungssystems in sich vereinen. Die inhaltsbasierte Komponente wird dabei von Semantic APIs mittels Empfehlungen und einem Wert, der die Relevanz gegenüber den Empfehlungen zur ursprünglich aufgerufenen Webseite darstellt, bereit gestellt. Um diese Kombination der Ansätze möglich zu machen, wird im Zuge dieser Arbeit ein Algorithmus entwickelt, der die kollaborativ gesammelten Bewertungen von Benutzern in den Relevanzwert miteinberechnet.

Der Aufbau dieser Arbeit ist in vier Hauptteile gegliedert. Kapitel 2 geht auf die nötigen Grundlagen ein, die für die folgende Arbeit wichtig sind. Kapitel 3 greift das vorhergehende Kapitel auf und hat verwandte Arbeiten zum Thema. Eingegangen wird dabei speziell auf beispielhafte Systeme, die dem eigenen entwickelten Ansatz, *ReadBeyond*, ähneln. Im dritten Teil, Kapitel 4, wird eben dieses eigens entwickelte Projekt, das dieser Arbeit zugrunde liegt, näher betrachtet. Kapitel 5 schließt die Hauptteile mit einer Evaluierung und Beschreibung der Ergebnisse ab. Im letzten Kapitel 6 wird ein Fazit gezogen und ein kurzer Ausblick in die Zukunft des entwickelten Systems gegeben.

Kapitel 2

Grundlagen

Der erste Abschnitt dieses Kapitels geht auf die Konzepte und Techniken des Semantic Web ein, auf denen wiederum die Auswahl der Semantic APIs für die Umsetzung basiert. Anschließend werden unterschiedliche Empfehlungssysteme beschrieben und dabei besonders der hybriden Ansatz fokussiert. Ferner ist für den späteren Verlauf der Arbeit auch eine Beschreibung von Benutzerskripten – mit einem besonderen Augenmerk auf ihre Eigenschaften, Herausforderungen und dem Unterschied zu Browser-Erweiterungen – wichtig.

2.1 Semantic Web

Das World Wide Web bietet durch Interaktivität eine immer größere Menge an Informationen.

Der Anteil an benutzer-generiertem Inhalt steigt rapide und eine Vielzahl an Unternehmen, die Dienstleistungen rund um das Internet anbieten, boomen. Benutzer konsumieren also Inhalt nicht mehr nur passiv, vielmehr sind sie aktiv an dessen Erstellung beteiligt. In erster Linie lassen sich drei Aktivitäten zusammenfassen, für die das Internet genützt wird: das Suchen und Integrieren von Informationen und Data-Mining; wobei die Suchfunktion die am meisten verwendete ist. Gegenwärtig können Maschinen die Dokumente im Web zwar präsentieren, aufgrund von fehlenden Informationen aber nicht verstehen [47].

Zu weiteren Problemen des Web, wie es heute bekannt ist, zählen Informationsintegration und Heterogenität. Informationen können großteils nur manuell und nicht automatisch integriert werden. Ferner begünstigt die dezentrale Struktur des Web heterogene Dateiformate, Kodierungstechniken, Sprachen und vieles mehr. Um diese Schwierigkeiten zu dezimieren, ist die Einführung von einheitlichen, offenen Standards unumgänglich. Durch diese können unterschiedliche Anwendungen Informationen interoperabel austau-

schen. Solche Standards werden vom World Wide Web Consortium[1] (W3C) definiert. Überdies muss es möglich sein aus vorhandenen Informationen automatisch logische Schlussfolgerungen zu ziehen und implizites Wissen verarbeiten zu können [18].

Der Begriff *Semantic Web* wurde von Tim Berners-Lee in seiner gleichnamigen Arbeit 2001 geprägt [9]. Während die Syntax die Satzlehre und die Struktur von Zeichen beschreibt, wird die Semantik durch die Bedeutung dahinter geprägt. Semantic Web kann also als *Web der Bedeutungen* bezeichnet werden [47]. Als Erweiterung des Webs soll es möglichst dezentralisiert sein. Maschinen müssen Zugriff auf standardisierte Informationen und Schlussregeln haben, um es Software-Agenten zu ermöglichen, komplexe Aufgaben für Benutzer auf unterschiedlichen Seiten durchzuführen [9].

2.1.1 Konzepte

Zwei Grundlegende Konzepte des Semantic Web sind Ontologien und Annotationen. Ontologien definieren die Beschreibung eines bestimmten Wissensbereichs. Sie sind die Grundlage bzw. bilden das Vokabular, mit dem Dokumente mit Metadaten ausgezeichnet oder annotiert werden.

Ontologie

Ursprünglich dem Bereich der Philosophie zuzuordnen, steht der Terminus dort für die Lehre des Seins [9].

In der Informationstechnik wird Wissen in Datenbanken, Diagrammen, Prozessbeschreibungen oder Ähnlichem beschrieben. Die Darstellung ist somit heterogen und fordert eine Konzeptualisierung, also eine Art Modell, welches von Menschen und Maschinen geteilt wird [12]. Ontologien im Themenfeld der Informatik sind solche Konzeptualisierungen und werden oft auch als Wissensbasis behandelt [18].

Ontologien beschreiben und repräsentieren einen bestimmten Wissensbereich, auch Domäne, und ermöglichen es, Informationen in diesem unter Benutzern, Applikationen sowie Datenbanken zu teilen. Definiert werden sowohl die grundlegenden Konzepte in dieser Domäne als auch die Beziehungen zwischen ihnen. Nicht vorgegeben ist die Strukturtiefe. So können beispielsweise Taxonomien, Metadatenformate und formale Systeme konzeptualisiert werden. Ontologien bestehen im Allgemeinen aus Begriffen (auch Klassen), Beziehungen zwischen diesen und Attributen, die diese Begriffe auszeichnen [17]. Superklassen und Subklassen ermöglichen eine hierarchische, feinere Struktur [47]. Die einzelnen Klassen können dann instanziert werden.

Liyang [47] charakterisiert die Vorteile von Ontologien wie folgt:

- Wissen in einer bestimmten Domäne kann wiederverwendet werden,

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

- die Annahmen in einer Domäne werden explizit,
- eine allgemein gültige Definition bzw. Verständnis für bestimmte Schlüsselbegriffe in einer Domäne werden bereitgestellt,
- mit formalen Beschreibungssprachen für Ontologien (wie RDFS und OWL siehe Abschnitt 2.1.2 und 2.1.2) kann Wissen kodiert und für Maschinen verständlich gemacht werden,
- automatische Maschinenverarbeitung wird ermöglicht.

Ferner kann zwischen flachen und tiefen Ontologien je nach Wissensdomäne unterschieden werden. Tiefe sind oft bei naturwissenschaftlichen Themen erforderlich. Flache wiederum werden verwendet, um Klassen, die sich für gewöhnlich nicht oder kaum verändern, zu definieren. So werden beispielsweise einfache Klassen wie Kunde, Bankkonto und Überziehungsrahmen im Kontext von Finanzen als flache Ontologie definiert [40].

Annotation

Das Konzept der Annotation beschreibt Dokumente, deren Inhalt mit Metadaten aufbereitet und so näher beschrieben werden [36]. Hitzler et al. [18] definieren Daten, die wiederum andere Daten näher beschreiben, als Metadaten. Durch die Anreicherung mit Annotationen können Maschinen die Bedeutung der Dokumente verstehen und so beispielsweise Informationen austauschen [36].

Es gibt drei verschiedene Möglichkeiten Metadaten in Dokumente einzufügen. Zum einen können Metadaten direkt in das Dokument eingefügt werden. Dies wird beispielsweise mit dem Resource Description Framework (siehe Abschnitt 2.1.2) umgesetzt. Der Vorteil dieser Methode gegenüber den externen Annotationen liegt darin, dass nur auf ein Dokument zugegriffen werden muss. Eine andere Methode ist die interne Verlinkung, bei der das annotierte Dokument nur einen Link zu einer externen Datei inklusive Metadaten beinhaltet. Ein Zugriff muss hier auf zwei Dokumente bestehen: auf das zu annotierende Dokument und die Annotationen. Hat man keine Möglichkeit auf eine Datei zuzugreifen, kann man auch von der externen Datei, welche die Metadaten enthält, auf die Datei verlinken [36].

2.1.2 Techniken

Die Prinzipien des Semantic Web werden im *Semanic Web Stack* oder *Semantic Web Layer Cake* visualisiert (siehe Abbildung 2.1). Die separaten Techniken und Standards sind von unten nach oben aufbauend arrangiert. Die zwei unteren Schichten – URI/IRI und Unicode – schaffen eine Grundlage für die Identifizierung der einzelnen Objekte im Semantic Web und der Verwendung einer internationalen Zeichenkodierung [21]. XML ist als zweite Ebene besonders bedeutsam: Einerseits ermöglicht sie dem Benutzer eine

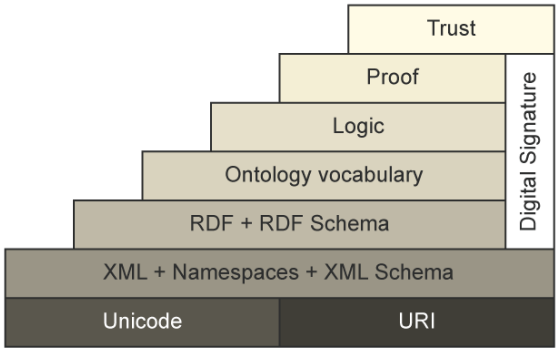


Abbildung 2.1: Semantic Web Stack (eigene Skizze nach [21]).

beliebige Struktur in Dokumente zu bringen; andererseits basieren die anderen Standards auf XML [41]. Mittels RDF and RDFS wird die eigentliche Bedeutung mithilfe von Relationen ausgedrückt. Aussagen können über bestimmte Objekte getroffen werden und ein Vokabular, auf das man mittels URIs verweisen kann, kann definiert werden. Durch Ontologien werden Vokabulare durch komplexere Relationen weiterentwickelt [21]. Sämtliche eben genannte Technologien sind bereits durch das W3C standardisiert.

Technologien, mit deren Verwendung Logik ausdrückbar wird, werden derzeit erforscht. Besonders die Simplizität von Schlussfolgerungsmechanismen soll so gegeben sei. Die Proof-Ebene ermöglicht das Verfassen von logischen Aussagen, mit denen ein Agent die Validität einer Aussage beweist. Wird diese Schicht mit der letzten, den digitalen Signaturen, kombiniert, kann der Benutzer den einzelnen Komponenten vertrauen [43].

RDF

Wie bereits beschrieben, werden Metadaten eingesetzt, um Dokumente im Web zu beschreiben und sie so für Maschinen verständlich und verarbeitbar zu gestalten. Es muss möglich sein Ressourcen zu beschreiben ohne dabei domänenspezifische Annahmen zu treffen. Das *Resource Description Framework* (RDF), als Baustein des Semantic Web, wird nicht nur für Metadaten in Webdokumenten verwendet, sondern auch um jegliche Ressource, die in der realen Welt existiert, zu beschreiben [47].

Anwendungen verarbeiten und manipulieren das Datenmodell von RDF problemlos. Wissen, welches mit RDF dargestellt wird, ist strukturiert und somit lesbar für Maschinen. Zielt man auf einen Vergleich ab, ist RDF für das Semantic Web wie HTML für das Web. Zum ersten Mal vom W3C[2] 1999 in einer Spezifikation erwähnt, zerteilt das abstrakte Modell Wissen in kleine, strukturierte Annahmen, womit jede denkbare Gegebenheit ausgedrückt

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

werden kann. Als formale Sprache ermöglicht RDF außerdem Schlussfolgerungen durch verlässliche Regeln [20, 47].

RDF kann Annahmen treffen, bei denen bestimmte Dinge gewisse Eigenschaften mit bestimmten Werten haben, zum Beispiel „Maria ist eine Frau“ [9]. Das Datenschema von RDF ist graph-basiert, da Informationen im Web oft dezentral verwaltet und gespeichert werden und sich so mehrere Ressourcen leicht kombinieren lassen. Der Graph ist gerichtet, das heißt seine Knoten sind mit gerichteten Pfeilen verbunden [18]. Die Knoten eines RDF-Graphs sind seine Objekte und Subjekte. Eine Annahme wird auch *Triple* genannt, da sie immer aus drei Teilen besteht: dem Subjekt, dem Objekt und dem Prädikat, wobei das Prädikat eine Beziehung zwischen beiden bezeichnet und als Pfeil dargestellt immer in Richtung des Objekts zeigt [20]. Sowohl Kanten als auch Knoten werden mit einem Bezeichner, einem *Uniform Resource Identifier* (URI) beschriftet. Diese URIs sind einzigartig, flexibel und beginnen immer mit einem Schema-Bezeichner. Subjekte und Objekte können mit URIs beschriftet werden. Besitzen sie keinen eindeutigen Bezeichner, werden sie *blank nodes* genannt. Prädikate müssen immer mit einem URI bezeichnet werden. Knoten und Kanten eines RDF-Graphen können mithilfe von URIs immer eindeutig beschrieben werden [47]. *Literale* stellen Datenwerte als Zeichenkette dar und besitzen einen Datentyp [18].

Um die Verarbeitung von Graphen durch Computer zu ermöglichen, werden RDF Datenmodelle oft mit XML (RDF/XML) repräsentiert. Andere formale Sprachen, die als RDF-Syntax verwendet werden, sind Notation-3 (N3), Turtel und N-Triples [20].

Eine Aussage in RDF kann also entweder als Graph, oder als ein oder mehrere Triples ausgedrückt und immer nur als binäre Beziehung zwischen zwei Ressourcen dargestellt werden. Reicht dies nicht aus, müssen vermittelnde Ressourcen dazwischen modelliert werden [47].

Im Gegensatz zu XML steht nicht die Darstellung von Dokumenten im Vordergrund, sondern vor allem die mögliche Weiterverarbeitung. Durch die Verwendung von URIs können gleiche Bezeichner, wie in XML, vermieden werden. Zahlreiche Werkzeuge und Bibliotheken für verschiedenste Programmiersprachen sind für den Einsatz bzw. die Erstellung von RDF verfügbar [18].

RDF-Schema

Mit RDF (siehe Abschnitt 2.1.2) können Aussagen über bestimmte Ressourcen, die zueinander in Beziehung stehen, getroffen werden. Diesen Ressourcen können sowohl Typen als auch Literale, also Datenwerte, zugewiesen werden. All diese Termini werden unter dem Begriff *Vokabular* zusammengefasst [18]. Liyang [47] bezeichnet das Vokabular, oder die gemeinsame Sprache, als Schlüssel zur Beschreibung von jeglichen Ressourcen. Damit ein System Schlüsse ziehen kann, ist es nötig auch Klassen einzuführen. Hitzler [18, S.

66] beschreibt:

> Beispielsweise ist ganz intuitiv klar, dass jede Universität eine Institution sein muss oder dass nur Personen bei einer Insitution beschäftigt sein können.

Diese Schlussfolgerung kann das System allein mit RDF nicht machen; die Zeichenketten haben weder Bedeutung noch logischen Zusammenhang.

RDF-Schema (RDFS) ist sozusagen eine Sprache, durch deren Vokabular, welches oft domänenspezifisch ist, terminologisches Wissen spezifiziert werden kann. Somit können semantische Beziehungen formuliert werden und die Bedeutung eines Dokuments wird erfassbar [18].

RDFS zählt zu den W3C Recommendations[3]. Die verwendeten Sprach-Konstrukte sind selbst *Klassen* und *Properties*, also die Relationen oder Prädikate im RDF-Triple. Diese standardisierten Konstrukte wiederum beschreiben Klassen und Relationen einer bestimmten Domäne. Auch das Erstellen von Sub-Klassen oder Sub-Properties ist möglich. Letztere werden von einer Klasse zu all ihren Sub-Klassen vererbt. Relationen werden separat erstellt und müssen ausdrücklich mit Klassen assoziiert werden – andernfalls sind sie unabhängig. So können sie potenziell jede Klasse beschreiben. Alle RDFS Termini sind durch URIs vordefiniert [47]. Hitzler [18, S. 67] ergänzt:

> Dabei ist RDFS zunächst nichts anderes als selbst ein spezielles RDF-Vokabular. Mithin ist jedes RDFS-Dokument ein syntaktisch korrektes RDF-Dokument. Dies hat den Vorteil, dass es von allen Programmen, die RDF unterstützen, gelesen und verarbeitet werden kann, wobei dann jedoch ein Teil der speziell für RDFS festgelegten Bedeutung (die RDFS-Semantik) verlorengeht.

Da terminologisches Wissen formuliert und repräsentiert wird, ist RDFS eine Ontologiesprache, die jedoch nur für sogenannte leichtgewichtige Ontologien eingesetzt wird. Schwergewichtige oder komplexere Ontologien können beispielsweise mit der Beschreibungssprache OWL (siehe Abschnitt 2.1.2) definiert werden [18].

Das Typsystem von RDFS ähnelt dem von objektorientierten Programmiersprachen wie Java. Trotzdem unterscheiden sich die beiden in einigen Punkten. Wie bereits erwähnt, werden bei RDFS Relationen, welche separat beschrieben werden, Klassen zugewiesen. In objektorientierten Programmiersprachen (OOP) werden Klassen u.a. als Gruppe von Properties oder Attributen bezeichnet. Damit einhergehend sind die Relationen in RDFS unabhängig, im Gegensatz dazu ist der Sichtbarkeitsbereich bei OOP oft nur auf die jeweilige Klasse beschränkt. Oft sind RDFS-Beschreibungen nicht zwingend Bestimmungen, wobei Typdeklarationen von Attributen bei OOP

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

präskriptiv sind. Zusätzliche Informationen, die Ressourcen beschreiben sollen, werden zwar von RDFS angeboten, bieten aber Anwendungen keine Beschreibung wie sie interpretiert werden sollen. Zuletzt sind Aussagen in RDFS immer Beschreibungen [24].

OWL

Die *Web Ontology Language* (OWL) ist die momentan wohl bekannteste Repräsentationssprache, die – auf Logik basierend – Wissen in Form von Ontologien beschreibt. Auch implizites Wissen kann so durch Schlussfolgerungen dargestellt werden [18]. Wie schon RDFS (siehe Abschnitt 2.1.2) verfolgt OWL den Zweck, vor allem Web-Ressourcen für automatisierte Prozesse zugänglich und verständlich zu machen. Da diese Ressourcen weit über das Web distribuiert sind, verknüpft OWL Ontologien miteinander, indem Informationen einzelner Ontologien wiederum bei anderen importiert werden. Dies erhöht die Bedeutung von Ontologien: Um ihren Einfluss zu steigern müssen sie weitgehend verteilt und wiederverwendet werden. Dem zugrunde liegt eine sogenannte *Open World Assumption*. Die Beschreibungen der Ressourcen sind nicht auf einzelne Dokumente beschränkt und beispielsweise einzelne Klassen einer Ontologie können in anderen Ontologien erweitert werden. Durch diese Erweiterung sind neue Informationen zwar womöglich widersprüchlich, aber sie löschen oder widerrufen alte Fakten nicht [42]. Die *Open World Assumption* nimmt außerdem an, dass jederzeit neue Informationen aufkommen können und die zu einem bestimmten Zeitpunkt verfügbare Information nicht die einzig verfügbare ist. Man geht also davon aus, dass mehr existiert, als in der Ontologie beschrieben ist [3].

Liyang [47] definiert OWL wie folgt:

> OWL = RDF Schema + new constructs for better expressiveness.

OWL beschreibt wie RDFS (siehe Abschnitt2.1.2) Ontologien, die Klassen, Properties und ihre Relationen in einer bestimmten Domäne inkludieren. Im Gegensatz zu RDFS kann OWL allerdings komplexere Beziehungen beschreiben. Da OWL auf RDFS aufbaut, können Teile des RDFS Vokabulars auch in OWL-Dokumenten verwendet werden [47].

Ein Individuum, eine Instanz, wird in OWL oft Objekt, eine Klasse Kategorie und eine Property Rolle genannt. Indem man verschiedene Klassen und/oder Rollen kombiniert, werden diese in komplexe Relationen zueinander gesetzt und neue Klassen oder Rollen entstehen [47].

Man unterscheidet zwischen [42]:

- Konkreten Rollen (*object properties*): Zwei Instanzen einer Klasse werden miteinander in Beziehung gesetzt.
- Abstrakten Rollen (*datatype properties*): Eine Instanz und ein Datentyp werden miteinander in Beziehung gesetzt.

Um komplexere oder ausdrucksstärkere Ontologien zu beschreiben, werden logische Konstruktoren auf Klassen eingesetzt: Konjunktion, Disjunktion und Negation; komplexe Klassen entstehen. Mit bestimmten Einschränkungen auf Rollen können komplexe Aussagen formuliert werden: Symmetrie, Transitivität, Funktionalität und inverse Funktionalität. Ferner gibt es Wert- und Kardinalitäts-Einschränkungen [18]. So können zum Beispiel folgende zwei Aussagen mit OWL beschrieben werden:

- Die FH Oberösterreich hat genau einen Geschäftsführer.
- An der FH Oberösterreich gibt es mindestens 4000 Studienplätze.

OWL besteht aus drei Teilsprachen: OWL Full, OWL DL und OWL Lite. Je nach Komplexität oder gewünschter Skalierbarkeit wird entsprechend gewählt [18].

Trotzdem hatte OWL, wie es 2004 vom W3C standardisiert wurde, große Einschränkungen in seiner Ausdrucksfähigkeit. Daher wurde OWL in *OWL 1* umbenannt und im April 2008 *OWL 2* durch die W3C OWL Working Group bestimmt [47]. Einige Änderungen bieten neue Ausdrucksmöglichkeiten, andere sind sogenannter *syntaktischer Zucker*[4]. Folgende Erneuerungen wurden beispielsweise vorgenommen bzw. hinzugefügt [16, 47]:

- neue Eigenschaften von Rollen: asymmetrisch, reflexiv und disjunktiv,
- verbesserte Annotationsmöglichkeiten,
- Einschränkungen in der Kardinalität,
- Property Chains,
- eigene Datentypen, die der Benutzer definieren kann,
- IRIs anstatt URIs[5].

2.2 Semantic APIs

Eine API (application programming interface) dient als Schnittstelle, die Entwicklern den Zugriff auf bereits implementierte Funktionalitäten bietet. Durch die Wiederverwendung von Code und einem hohen Maß an Abstraktion werden Programmieraufgaben erleichtert [37]. Uddin et al. [46] vergleichen APIs mit Bibliotheken und Frameworks, die Software-Systemen die Wiederverwendung von Funktionalitäten erlauben. Das clientseitige Programm wiederum nützt vorhandene Funktionalitäten mittels APIs.

Besitzt ein Dokument keinerlei Annotationen wie in Abschnitt 2.1 beschrieben, kann trotzdem die Bedeutung einer Ressource mit Hilfe einer API

[4]Nach Van Roy [38] reduziert syntaktischer Zucker in der Informatik die Zeilen eines Programms und macht es so besser leserlich. Die Syntax wird erweitert und der Code vereinfacht

[5]Ein IRI (Internationalized Resource Identifier) erlaubt auch Unicode-Zeichen, wobei URIs auf den ASCII-Zeichensatz beschränkt sind [47].

entschlüsselt werden. *Semantic APIs* sind spezielle APIs, die unstrukturierten Text oder ganze Webseiten verarbeiten und die kontextuelle Struktur des Inhalts bereitstellen [11].

2.2.1 Anbieter

Je nach Anforderungen können unterschiedliche Anbieter in Betracht gezogen werden. Grundsätzlich ist der Einsatz von Semanic APIs erst sinnvoll, wenn der übermittelte Input eine gewisse Länge von z. B. mehreren Absätzen hat. APIs wie Zemanta oder AlchemyAPI (s. unten) können Texte ab einer Länge von 140 Zeichen analysieren. Zemanta, OpenCalais und AlchemyAPI wurden hier besonderes fokussiert, da diese auch im eigenen Ansatz verwendet wurden; für eine Begründung für diese Auswahl siehe Abschnitt 4.1.2.

Zemanta

Die API Zemanta[6] ist ein sogenannter Aggregator: also eine Software, die einen bestimmten Input wie unstrukturierten Text sammelt, verarbeitet und mit Medieninhalten wiederaufbereitet [31]. Dabei lässt sich Zemanta als Browser-Extension oder Plugin für diverse Plattformen wie WordPress[7], Blogger[8], TypePad[9], und Movable Type[10] installieren. Für Joomla[11], Wordpress und Movable Type ist außerdem ein serverseitiges Plugin verfügbar.

Hauptsächlich für Blogger entwickelt, liefert die API dem Benutzer dem Kontext verwandte Artikel, Bilder oder Links zu Webseiten passend zu seinem Blog-Eintrag [48]. Ergänzend wird die Bedeutung des Textes soweit erfasst, dass dem Benutzer Schlagworte und Kategorien – im Grunde genommen grobe Themen, denen der Input zuzuordnen ist wie */computer/software* – zur Beschreibung des Inputs vorgeschlagen werden. Diese Outputs können dem jeweiligen Blog-Eintrag angefügt werden, um ihn so mit zusätzlichen Informationen oder Medien anzureichern. Indem Zemanta mit anderen, eventuell eigenen, Blogs, dem eigenen Twitter-, Flickr- oder Instagram-Account assoziiert wird, werden eigene Inhalte zur Aufwertung der Artikel vorgeschlagen. Momentan unterstützt Zemanta nur die englische Sprache [48].

Mit 1.000 kostenfreien Aufrufen pro Tag gilt Zemanta als Einstieg in unterschiedliche vorindizierte Datenbanken, wobei der Input mit den Einträgen in der Datenbank verglichen wird und so der Inhalt kategorisiert werden kann. Dabei lernt das System bei jeder Anwendung. Zusätzlich können die Benutzer Pakete kaufen, wodurch ihnen mehr Aufrufe pro Tag zur Verfügung stehen. Hinter der API kommen semantische Algorithmen und *Natural*

[6] http://www.zemanta.com/
[7] http://de.wordpress.com/
[8] http://www.blogger.com/
[9] http://www.typepad.com/
[10] http://www.movabletype.com/
[11] http://www.joomla.at/

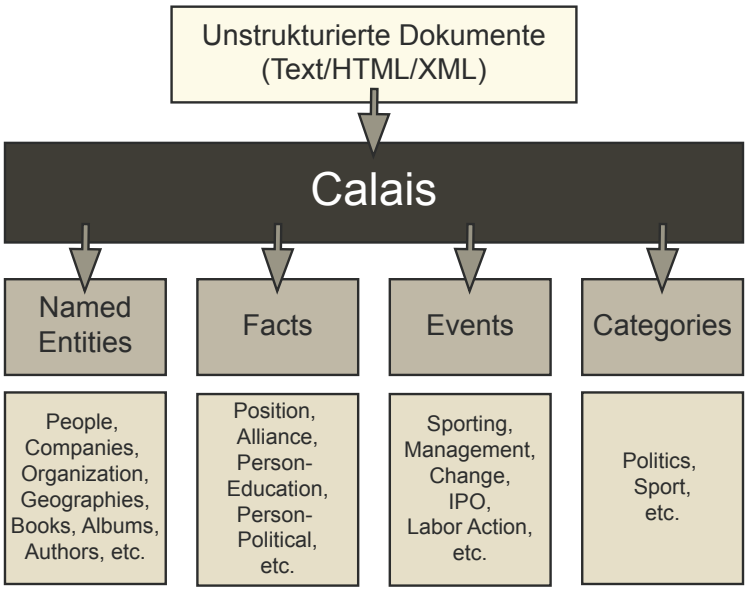


Abbildung 2.2: Input und Output von OpenCalais (eigene Skizze nach [49]).

Language Processing (NLP) zum Einsatz [11].

Natural Language Processing (NLP): Das Ziel des NLP ist die Bedeutung eines Textes zu verstehen. Dadurch lassen sich diese automatisch von Maschinen verarbeiten und vom Benutzer formulierte Anfragen verarbeiten. Dies ist vor allem auch durch Schlussfolgerungsmechanismen möglich. NLP ermöglicht auch das Erkennen von Entitäten in Texten (*Named Entity Recognition*), welches vor allem von anderen Semantic APIs angeboten wird [8].

OpenCalais

OpenCalais[12] verarbeitet unstrukturierten Text mittels NLP (siehe Abschnitt 2.2.1) und maschinellem Lernen und liefert dem Benutzer die darin enthaltenen Entitäten, Fakten und Ereignisse (siehe Abbildung 2.2) annotiert mit RDF(siehe Abschnitt 2.1.2). Der gefundene Output von Calais wird im Format RDF (siehe Abschnitt 2.1.2) beschrieben [49].

Bauer [7] erwähnt, dass Calais dem Benutzer fünf Arten von Metadaten als Output liefert:

- Named Entities: Dies sind Klassifizierungen wie u. a. *Anniversary*, *City*, *Company*, *Continent*, *Person*, die mit einer URI (siehe Abschnitt 2.1.2)

[12] http://www.opencalais.com/

versehen sind, die wiederum auf ein Repository zeigen, wo der Benutzer mehr Informationen über die Identität finden kann.
- Events und Facts: Für vordefinierte Ereignisse wie *MovieRelease* werden dazugehörige Fakten ausgelesen.
- Kategorien: Kategoriern sind grobe Themen, denen der eingegebene Text zuzuordnen ist.
- Social Tags: Sie ahmen nach, welche Stichwörter eine Person einem Text zuordnen würde.

Momentan kann OpenCalais nur Entitäten in englischen, spanischen oder französischen Texten finden. Die anderen angebotenen Metadaten werden nur in englischen Texten extrahiert [11]. Die API ist kostenfrei für kommerzielle und nicht kommerzielle Nutzung, wobei die Transaktionen pro Lizenz und Tag auf 50.000 und pro Sekunde auf 4 limitiert sind. Gegen Bezahlung lässt sich mit *ProfessionalCalais* eine Lizenz erwerben, die 100.000 Transaktionen pro Tag und 20 pro Sekunde ermöglicht [49].

AlchemyAPI

Im Gegensatz zu Zemanta und OpenCalais kann AlchemyAPI[13] – laut eigenen Angaben als einziger Anbieter – als Input sowohl unstrukturierten Text als auch nicht-textuellen Inhalt wie Bilder verarbeiten. So können beispielsweise Named Entities und Kategorien aus eingescannten Bildern, Fotografien oder Printmedien wie Zeitungen extrahiert werden. Durch komplizierte Algorithmen und NLP (siehe Abschnitt 2.2.1) wird dem Benutzer so die semantische Bedeutung seines Inputs aufgezeigt [50].

AlchemyAPI bietet folgende Funktionalitäten [50]:

- Named Entity Extraction (siehe Abschnitt 2.2.1),
- Sentiment Analysis: Identifizerung von positiven, neutralen oder negativen Stimmungen innerhalb des übermittelten Inputs auf verschiedenen Ebenen (Dokument, Benutzer, Entitäten, Keywords),
- Keyword Extraction: Extraktion von wichtigen Begriffen bzw. Schlagwörtern,
- Concept Tagging: der Text wird mit Schlagwörtern versehen, die auch eine Person aus dem Inhalt erschließen würde (vergleichbar mit den Social Tags von OpenCalais),
- Relations Extraction: Extraktion von Triples, also Subjekt-Objekt-Beziehungen,
- Author Extraction: automatische Extraktion von Informationen über den Autor des Textes,
- Web Page Cleaning: Extraktion des Textes des Inputs, z. B. einer Website, um ihn so von anderen Elementen wie der Navigation sozusagen

[13] http://www.alchemyapi.com/

zu reinigen,

- Language Detection: *Verarbeitung* des vom Benutzer übermittelten Inputs in acht Sprachen, darunter Englisch, Spanisch, Deutsch, Russisch, Italienisch, Portugiesisch, Französisch und Schwedisch und *Identifizierung* der Sprache des Textes unter 97 unterstützten Sprachen,
- Topic Categorization: Zuordnung von Texten zu verschiedenen Themen, um sie so leichter zu klassifizieren und Taxonomien erstellen zu können,
- Content Scraping: Extraktion von strukturierten Daten wie Preisangaben,
- RSS/ATOM Feed Detection: Erkennung von RSS-/Atom-Feeds,
- Microformats Parsing: Identifzierung von Mikroformaten.

Die API bietet Software Development Kits für die wichtigsten Progammiersprachen. Ein Output in den Formaten XML, JSON und RDF wird angeboten. Alchemy kann kostenfrei kommerziell und nicht kommerziell genützt werden. Je nach kostenpflichtigem Paket ist der Benutzer auf bis zu 200 Millionen Aufrufe pro Tag limitiert [50].

Andere Anbieter

Wie bereits erwähnt gibt es zahlreiche Anbieter für Semantic APIs, auf die jedoch nicht näher eingegangen wird. Der Vollständigkeit halber werden hier weitere Anbieter angeführt:

- OpenAmplify[14],
- Dapper[15],
- SemanticHacker[16],
- Semantic API[17],
- Ontos API[18],
- DBPedia[19].

2.2.2 Einsatzgebiete

Semantic APIs werden in verschiedenen Einsatzgebieten angewendet. Grundsätzlich werden sie genutzt, um unstrukturierten Input mit Metadaten anzureichern. Dabei können viele APIs in Content Management Systeme (CMS) durch Plugins integriert werden oder bieten Software Development Kits für die gängigen Programmiersprachen.

[14] http://www.openamplify.com/
[15] http://open.dapper.net/
[16] http://www.textwise.com/
[17] http://www.sensebot.net/semanticapi.aspx
[18] http://www.ontos.com/
[19] http://dbpedia.org/

Die Einbindung in unterschiedliche CMS und Blog-Systeme bilden die Voraussetzung für das Anreichern von Artikeln. Benutzer können so Einträge mit zusätzlichen Bildern oder weiterführenden Links versehen.

Funktionalitäten wie die *Topic Categorization* der AlchemyAPI erleichtern die Erstellung bzw. Erweiterung von Taxonomiesystemen.

Bauer [7] beschreibt außerdem den Einsatz von Semantic APIs bei der Aufbereitung von Inhalten für semantische Suchmaschinen, die Erstellung von Mashups, den Einsatz in Medienplattformen, um beispielsweise Themen zu kategorisieren oder die Suche zu optimieren, die Analyse von Beiträgen in sozialen Netzwerken und Brainstorming-Anwendungen.

2.3 Empfehlungssysteme

Personalisierung bezeichnet die Adaptierung eines Informationssystems, sodass die unterschiedlichen Bedürfnisse der Benutzer erfüllt werden. Das Ziel dieser ist zum einen den Benutzer in der immer größer werdenden Informationsflut zu entlasten und ihm so durch einfachen Zugriff auf die benötigten Informationen zur richtigen Zeit einen Mehrwert zu schaffen [32]. Mobasher [26] fügt hinzu, dass dies geschehen soll, ohne dass der Benutzer explizit um diese Informationen bittet. Zum anderen sollen bei kommerziellen Webseiten, die gegenüber ihrer steigenden Konkurrenz herausstechen wollen, loyale, vertrauenswürdige Kundenbeziehungen geformt werden [32].

Personalisierungssysteme erkennen für gewöhnlich die einzelnen Benutzer, sammeln Informationen über u. a. ihre Vorlieben oder Interessen und bilden diese in einem Benutzermodell ab. Um diese Informationen zu finden, wird Data-Mining eingesetzt [32].

Eine spezielle Art von Personalisierungssystemen im Web sind Empfehlungssysteme, die dem Benutzer einzelne Objekte wie Filme, Produkte oder Webseiten empfehlen [26].

2.3.1 Data Mining in Personalisierungssystemen

Wird Data Mining in Personalisierungssytemen online eingesetzt, spricht man von *Web Mining*. Das Ziel des Web Mining ist es, Daten von Webseiten zu extrahieren. Eine Unterscheidung ist nicht immer klar und kann teilweise auch verschwommen sein; Kosala und Blockeel [22] versuchen eine Unterteilung folgendermaßen:

- *Web content mining*: Daten aus dem Inhalt der Webseite werden extrahiert. Diese können aufgrund der unterschiedlichen Inhalte wie Texte, Bilder, Videos etc. verschiedenartige Datentypen haben und strukturiert, unstrukturiert oder semistrukturiert sein.
- *Web structure mining*: Es wird versucht ein Modell zu finden, dass hinter der Linkstruktur und dem Aufbau einer Webseite liegt.

- *Web usage mining*: Der Fokus liegt hier auf den Interaktionen des Benutzers. Diese Daten werden beispielsweise durch die Registrierung, Cookies, Sessions, Mausklicks, das Scrollverhalten oder Server Logs gesammelt.

Web Usage Mining erfolgt in vier Schritten. Zuerst werden die unterschiedlichen Daten aus den einzelnen Quellen gesammelt und inhaltlich und strukturell identifiziert. Im nächsten Schritt werden sie bereinigt und eventuelle Inkonsistenzen beseitigt. Anschließend werden darunterliegende Muster durch Techniken des maschinellen Lernens und der Statistik erkannt. Zuletzt wird das daraus entstandene Wissen evaluiert und präsentiert [32].

2.3.2 Filtertechniken

Mobasher [26] unterscheidet zwischen *customization* (Anpassung) und *automatic personalization* (automatische Personalisierung, Adaptierung). Bei der Anpassung ist der Benutzer selbst für die Erstellung seines Benutzermodells verantwortlich, indem er oft manuell bevorzugte Einstellungen oder Anforderungen an das System eingibt. Geschieht dies automatisiert (und somit implizit), ist das System für die Erstellung bzw. Aktualisierung des Modells maßgebend.

Inhaltsbasiertes Filtern

Inhaltsbasierte Empfehlungssysteme empfehlen dem Benutzer ein Objekt basierend auf dessen Beschreibung und dem Benutzermodell, das beispielsweise die Interessen des Benutzers beinhaltet. Solche Objekte sind u. a. Webseiten, Produkte, Filme oder Restaurants, die der Benutzer – in einer Liste dargestellt – selektieren kann, um so mehr Details zu erhalten. Entweder werden gewisse Teilmengen mit Objekten selektiert oder eine Reihenfolge, in der die Objekte angezeigt werden, je nach Interessen des Benutzers festgelegt. Die einzelnen Objekte werden meistens in Datenbanken gespeichert. In den jeweiligen Einträgen werden die Attribute festgehalten, wobei jedes Objekt die gleichen Attribute besitzt. Dies gilt jedoch nur für strukturierte Daten. Bei unstrukturierten Texten wie News-Artikeln sind gleiche Attribute mit wohlgeformten Werten nicht möglich. Auch Polyseme, also gleiche Wörter unterschiedlicher Bedeutung, und Synonyme stellen eine Schwierigkeit dar. Bei semistrukturierten Texten, die Attribute mit festen Werten und freiem Text darstellen, wird letzterer oft in eine strukturierte Form umgewandelt, bei dem beispielsweise jedes Wort ein Attribut und der Wert seine Häufigkeit darstellt. Oft wird *Stemming*, bei dem der Worstamm von mehreren Wörtern herausgearbeitet wird, eingesetzt. Wird nur die Häufigkeit der aufgetretenen Wörter gezählt, ist dies nicht sehr aussagekräftig. Oft ist der Zusammenhang der Wörter bzw. die Phrase aussagekräftiger. Zum Beispiel kann in der Beschreibung eines Steakhouses folgender Satz stehen: "Nichts auf der

Karte würde einem Vegetarier schmecken."Wird nur die Häufigkeit der aufgetretenen Wörter gezählt, würde dies vermuten dass das Restaurant auch vegetarische Küche führt. Dies zeigt, dass sich die eingesetzten Techniken bei strukturierten und unstrukturierten Texten differenzieren müssen [30].

Das Benutzermodell kann zwei Arten von Informationen enthalten: die Präferenzen des Benutzers, also Beschreibungen der Objekte, die ihn interessieren, und eine Aufzeichnung der früheren Interaktionen mit dem System. Zweitere sind nötig, um Objekte, die der Benutzer bereits gelesen oder gekauft hat, auszufiltern oder um Algorithmen des maschinellen Lernens, die Benutzerprofile erzeugen, mit Trainings-Daten zu versorgen [30].

Das System kann sowohl durch explizites oder implizites Feedback als auch durch Beobachtungen der Interaktionen lernen. Die Algorithmen, mit denen das System dazulernt, bilden die Schlüsselkomponente eines inhaltsbasierten Empfehlungssystems. Mithilfe der Algorithmen und des Benutzermodells können auch Vorhersagen getroffen werden [30].

Regelbasierte Empfehlungssysteme: Auf dem Verlauf der früheren Interaktionen des Benutzers basierend, hat dieses System bestimmte Regeln, um Objekte zu empfehlen. Beispielsweise werden Kunden, die ein Buch einer Trilogie gekauft haben, auch die anderen beiden empfohlen [30]. Die aufgestellten Regeln basieren oft auf demografischen, psychografischen oder anderen persönlichen Charakteristiken eines Benutzers und sind oft sehr domänenspezifisch. Die Daten sind häufig die subjektiven Einschätzungen des Benutzers selbst und somit anfällig für Verzerrungen. Da die Benutzermodelle oft statisch sind, veralten sie mit der Zeit, insofern sie nicht aktualisiert werden [26].

Fallbasierte Empfehlungssysteme: Techniken des fallbasierten Schließens sind der Ursprung dieser Systeme. In einer Datenbank werden frühere Problemlösungen in Spezifikation-Lösung-Paaren gespeichert. Bei neuen Problemen werden ähnliche Spezifikationen gesucht und ihre Lösungen adaptiert, um das aktuelle Problem zu lösen. Bei fallbasierten Empfehlungssystemen werden Objekte als Fälle bezeichnet und Empfehlungen, die am besten zu der Anfrage des Benutzers passen, gefunden. Verglichen werden – wie bei regulären inhaltsbasierten Systemen – die Beschreibungen der Objekte. Fallbasierte Systeme unterscheiden sich jedoch in zwei Dingen von regulären [30]:

- Oft werden inhaltsbasierte Empfehlungsssyteme bei unstrukturierten oder semistrukturierten Inhalten wie News-Artikeln eingesetzt. Fallbasierte wiederum verwenden oft strukturierten Inhalt anstatt freiem Text und werden beispielsweise im E-Commerce eingesetzt, wo detailierte Produktangaben für die einzelnen Produkte verfügbar sind. Innerhalb dieser strukturierten Beschreibungen können sowohl numeri-

sche als auch nominale Eigenschaften benutzt werden.

- Da fallbasierte Systeme auf strukturierte Texte angewiesen sind, verwenden sie nicht nur die Ähnlichkeitsberechnungen einzelner Schlüsselwörter, sondern komplexere Metriken. So können Empfehlungen gemacht werden, deren Anfrage ähnlich ist zu der des Benutzers, auch wenn keine exakten Treffer gefunden werden. Um die Ähnlichkeit nicht numerischer Eigenschaften zu berechnen, ist oft zusätzliches, domänenspezifisches Wissen notwendig. Werden bei einem System, das Urlaubsangebote empfiehlt, beispielsweise keine Treffer zu Wanderurlauben gefunden, muss evaluiert weren, ob Ski- oder Strandurlaube einem Wanderurlaub ähnlicher sind, um diese zu empfehlen. Um dies zu bewerkstelligen, ist domänenspezifisches Wissen aus einer Ontologie (siehe Abschnitt 2.1.1) über Urlaubstypen nötig.

Enthält das Benutzermodell Informationen über die früheren Interaktionen des Benutzers, tendiert es bei künftigen Anfragen zur Überspezialisierung. Benutzerstudien zeigen jedoch, dass Benutzer Empfehlungssysteme dann besonders nützlich empfinden, wenn sie unerwartete Objekte liefern [26]. Enthält der Inhalt nicht ausreichend Informationen, können keine Empfehlungen ausgesprochen werden. Dadurch ist es schwierig von Worthäufigkeiten auf die Interessen des Benutzers zu schließen. So kann ein Witz über Anwälte von einem Blondinenwitz differenziert werden, jedoch ist es komplex einen lustigen Witz von den anderen zu unterscheiden. In solchen Fällen sollte auf kollaborative Techniken (siehe Abschnitt 2.3.2) zurückgegriffen werden. Unterstützend können diese Techniken in hybriden Empfehlungssystemen zusätzlich angewendet werden [30]. Oft ist es schwierig Objekte durch extrahierte Teile der Texte zu beschreiben, da im Normalfall Daten im Web heterogen sind [26].

Der Inhalt von Webseiten kann durch Semantic APIs (siehe Abschnitt 2.2) analysiert werden. Diese liefern auch die Relevanzen einzelner Schlüsselwörter im Kontext, die bei den einzelnen Objekten abgelegt und für spätere Empfehlungen genutzt werden können. Die API Zemanta (siehe Abschnitt 2.2.1) bietet die Analyse des Inhalts einer Webseite und die Empfehlung von Objekten, die von unterschiedlichen, ausgewählten Webseiten stammen.

Kollaborative Empfehlungssysteme

Beim kollaborativen Filtern werden Objekte mithilfe der Bewertungen bzw. Meinungen von anderen Benutzern evaluiert. Werden bei reiner Mundpropaganda zumeist nur Individuen oder Hunderte von Personen erreicht, können durch den Einsatz von Computern die Meinungen von Tausenden eingeholt und in Echtzeit verarbeitet werden. Einerseits wird so bestimmt, welche Meinung eine große Community, also eine Gemeinschaft von Menschen im Netz, über ein Objekt hat. Andererseits kann eine personalisierte Ansicht, die nur aus Objekten besteht, deren Meinungen am besten zu einem Benutzer

oder einem Benutzersegment passen, erstellt werden [39]. Um die Meinungen der Benutzer einzuholen, werden Bewertungen eingesetzt. Diese sogenannten *Ratings* bestehen zumeist aus einem bestimmten Wert. Schafer et al. [39] unterscheiden drei Arten von Ratings, die sowohl explizit und/oder implizit gesammelt werden können:

- Skalenbasierte Bewertungen: zum Beispiel numerische Bewertungen wie 1-5 Sterne,
- Binäre Bewertungen: zum Beispiel gut/schlecht oder einverstanden/-nicht einverstanden,
- Unäre Bewertungen: zeigen, ob ein Benutzer ein bestimmtes Objekt bereits wahrgenommen oder sogar gekauft hat.

Damit Objekte beispielsweise nicht nur durch die Ähnlichkeit von Schlüsselwörtern empfohlen werden, ist für bessere Nachvollziehbarkeit die Qualität des Objektes bedeutend. Diese kann durch kollaborative Techniken ermittelt werden. Zu Beginn wurden diese Techniken eingesetzt, um dem Benutzer zusätzliche Informationen in Form von Bewertungen für ein Objekt zu geben. Später wurden diese genutzt, um den Inhalt einer Webseite zu adaptieren und so zum Beispiel nur bestimmte News-Artikel anzuzeigen [39].

Kollaboratives Filtern kann unterschiedliche Aufgaben erfüllen. Zum einen kann es einem Benutzer helfen, Objekte, die ihm oder seiner Gruppe gefallen, zu finden. Zum anderen kann es konkrete Informationen über bestimmte Objekte, die den Benutzer interessieren, geben. Da durch die Bewertungen Benutzermodelle erstellt werden können, können auch Benutzer, die man aufgrund ähnlicher Bewertungen vermutlich mag, gefunden werden. Kollaborative Systeme können nicht nur Objekte empfehlen, sondern auch Vorhersagen über ein bestimmtes Objekt treffen. Letzteres ist jedoch problematisch, wenn nur wenige oder noch kein Benutzer das jeweilige Objekt bewertet hat [39].

Ein kollaboratives System kann anhand bestimmter Kennzahlen evaluiert werden. Hauptsächlich wird dazu die Treffgenauigkeit der Vorhersagen eines Objekts betrachtet. Des Weiteren kann die Neuheit ins Auge gefasst werden, das heißt ob ein Objekt empfohlen wurde, welches dem Benutzer unbekannt war. Zusätzlich kann der Prozentsatz an Objekten, für die das System vorhersagen treffen kann, zur Evaluierung herangezogen werden. Auch die Lernrate des Systems, die besagt wie viele Empfehlungen ein Benutzer geben muss, damit die Vorhersagen treffend sind, ist aussagekräftig [39].

Bei kollaborativen Empfehungssystemen geschieht der Prozess der Modellierung meist online und in Echtzeit. Nehmen Benutzer und Objekte zu, kann es zu Verzögerungen kommen und mitunter einige Zeit dauern, bis Empfehlungen oder dynamischer Inhalt bereitgestellt werden kann [26]. Eine große Herausforderung vor allem für die Akzeptanz von kollaborativen Systemen stellen die Sicherheit und Privatsphäre von Benutzern dar. Ein kollaboratives Empfehlungssystem kann umso bessere Empfehlungen treffen, je

mehr Informationen über den Benutzer gesammelt werden. Dieser sorgt sich unter anderem darum wo die Informationen gespeichert und wie sie verwendet werden. Es ist unumgänglich, dass die Benutzer dem System vertrauen und sich sicher sein können, dass die gesammelten Daten nur für Vorhersagen und Empfehlungen genutzt und nicht weitergegeben werden. Die bereits abgegebenen Bewertungen müssen immer die Meinungen der Benutzer repräsentieren und zum Beispiel nicht durch maliziöse Benutzer manipuliert werden [39].

Hybride Empfehlungssysteme

Ein Unterschied zwischen kollaborativem und inhaltsbasiertem Filtern liegt bei der Annahme *was* ähnlich bewertet wird: Kollaboratives Filtern geht davon aus, dass Benutzer mit ähnlichem Geschmack Objekte ähnlich bewerten. Inhaltsbasiertes Filtern wiederum geht davon aus, dass Objekte mit ähnlichen Eigenschaften ähnlich gewertet werden. Ein Vorteil von inhaltsbasiertem Filtern ist die Vorhersage von Objekten auch ohne Bewertungen. Beim kollaborativen Filtern werden bestenfalls mehrere, jedoch zumindest eine Bewertung benötigt. Inhaltsbasierte Empfehlungssysteme sind auf ausreichenden Inhalt angewiesen, um ihn analysieren zu können. Auch bestimmte Inhaltstypen wie Video oder Audio erschweren die Extraktion des Inhalts. Wie bereits erwähnt, bevorzugen Benutzer unerwartete Empfehlungen. Diese Funktion kann kollaboratives Filtern bieten. Oft werden beide Ansätze miteinander kombiniert und ein (teilweise automatisierter) hybrider Ansatz angewendet [39].

Burke [10] unterscheidet nicht nur grob zwischen inhaltsbasierten und kollaborativen Techniken, sondern auch zwischen wissensbasierten und demografischen Techniken. Letztere bieten Empfehlungen basierend auf dem demografischen Profil der Benutzer, sodass beispielsweise Produkte nur auf einem bestimmten demografischen Nischenmarkt angeboten werden. Wissensbasierte empfehlen Objekte aufgrund der Bedürfnisse und Präferenzen des Benutzers. Alle Techniken besitzen Vor- und Nachteile und leiden an dem sogenannten *cold-start Problem*; für neue Benutzer, die noch kein Benutzermodell und keine Ratings abgegeben haben, bzw. für Benutzer mit nur wenigen Ratings ist es sehr schwierig Empfehlungen auszusprechen. Ein weiteres Problem sind Änderungen der Präferenzen. Wird ein Benutzer beispielsweise Vegetarier, wird es zahlreiche Ratings benötigen, um dem Benutzer keine Empfehlungen mehr für Steakhäuser anzubieten. Ein Vorteil von wissensbasierten Systemen ist der Umgang mit diesen Änderungen: sie müssen nicht zuerst gelernt werden, sondern reagieren auf die unmittelbaren Bedürfnisse des Benutzers [10].

Hybride Empfehlungssysteme versuchen die Vorteile der einzelnen Systeme durch eine Kombination zweier oder mehrerer Techniken hervorzuheben. Es ist auch möglich, dass zwei Empfehlungssysteme mit gleichen Techniken,

beispielsweise zwei inhaltsbasierte Systeme, kombiniert werden. Häufiger ist jedoch eine Kombination aus unterschiedlichen Methoden. In einer Taxonomie klassifiziert Burke [10] sieben hybride Empfehlungssysteme (die deutsche Übersetzung wurde von Franz [15] übernommen; siehe Abbildungen 2.3 und 2.4):

1. Gewichtet (*weighted*): Der gewichtete Ansatz ist wahrscheinlich der einfachste. Jedes beteiligte Empfehlungssystem bewertet ein bestimmtes Objekt; anschließend werden die beiden Wertungen linear zu einer Gesamtwertung kombiniert. Die Objekte werden dann nach ihrer Gewichtung gereiht. Die einzelnen Systeme können in unterschiedlichen Verhältnissen, zum Beispiel 60:40, gewichtet werden.
2. Wechselnd (*switching*): Je nach Situation wird *ein* bestimmtes System gewählt. So können für unterschiedliche Benutzermodelle verschiedene Empfehlungssysteme eingesetzt werden. Vorausgesetzt wird ein verlässliches Kriterium, nach dem die Systeme ausgewählt werden.
3. Gemischt (*mixed*): Jedes Empfehlungssystem produziert ihre eigene Liste mit Empfehlungen, die dem Benutzer zusammengefügt angezeigt wird. Oft werden die Ergebnisse nach dem prognostizierten Rating oder dem Vertrauen in das Empfehlungssystem sortiert.
4. Merkmal-Kombination (*feature combination*): Der Algorithmus eines Empfehlungssystems wird hauptsächlich verwendet, aber mit den Eigenschaften eines zweiten Empfehlungssystems angereichert. Beispielsweise kann so ein inhaltsbasiertes auf Eigenschaften eines kollaborativen Systems zurückgreifen.
5. Merkmal-Anreicherung (*feature augmentation*): Für jedes Objekt wird eine Eigenschaft durch ein System generiert, damit die eigentlichen Daten für das zweite Empfehlungssystem abgefangen und mit der neuen Eigenschaft ergänzt. Dieser Ansatz wird beispielsweise verwendet, wenn ein System bereits sehr gut entwickelt ist, aber eine zusätzliche Wissensquelle gewünscht wird. Im Gegensatz zum Merkmal-Kombinations Ansatz, der nicht auf beliebige Kombinationenen der Empfehlungssysteme anwendbar ist, ist dieser flexibler. Konträr zum Merkmal-Kombinations Ansatz, werden nicht die Inputs kombiniert und vorverarbeitet, so Jannich et al. ([19], S. 132), sondern mehrere komplexe Transformationsschritte durchgeführt. Die Implementierung des mitwirkenden Empfehlungssystems ist also sehr stark mit der des Haupt-Recommenders verwoben, beispielsweise aus Gründen der Funktionalität.
6. Kaskadiert (*cascade*): Der kaskadierte Ansatz verkettet zwei Empfehlungssysteme hierarchisch. Ein System kann so von einem anderen, schwächeren verfeinert, aber seine Entscheidungen können von diesem nicht überworfen werden. Das zweite System soll nur die Bewertung der Ergebnisse beeinflussen und Gleichstände aufheben.

7. Meta-Level: Das unterstützende System lernt nur das Benutzermodell und gibt dieses an das Haupt-Empfehlungssystem weiter. Im Gegensatz zum Merkmal-Anreicherungs Hybrid wird hier die Wissensquelle des Haupt-Systems komplett mit den Daten des unterstützenden Systems überschrieben.

Die sieben Ansätze für hybride Empfehlungssysteme lassen sich nach Franz [15] in drei Kategorien einteilen :

1. Kombination von unterschiedlichen Empfehlungssystemen,
2. Verbesserung eines Haupt-Empfehlungssystems durch ein anderes,
3. Vereinende Empfehlungssysteme, also solche, die die Vorteile eines anderen Systems nützen.

Zur ersten Kategorie, der Kombination aus unterschiedlichen Systemen, zählen der gewichtete, wechselnde, gemischte und kaskadierte Ansatz. Die Empfehlungen des Merkmal-Kombinations und des Merkmal-Anreicherungs-Ansatz werden durch ein zweites System verbessert und fallen somit in die zweite Kategorie. Der Meta-Level-Ansatz zählt zur dritten Kategorie der vereinenden Empfehlungssysteme. Um die Systeme u. a. auf Effizienz zu vergleichen, implementierte Burke [10] mit seinen Kollegen das System Entree (siehe Abschnitt 3.1).

2.4 Benutzerskripte

Im Zusammenhang mit Benutzerskripten werden häufig die Begriffe *Active Browsing* und *Greasemonkey* genannt.

Active Browsing: Für gewöhnlich konsumieren Benutzer Webseiten passiv, ohne Einfluss auf die vom Webserver stammenden Inhalte zu haben. *Active Browsing* bezeichnet genau das Gegenteil: Die User Experience auf bestimmten Seiten wird manipuliert, indem die Daten auf Webseiten verändert werden. Benutzer haben so selbst Kontrolle über das Aussehen und die Funktionalität von Seiten im Internet [5, 45]).

Greasemonkey: Mittels der Firefox-Erweiterung *Greasemonkey* können clientseitige Skripte, sogenannte Benutzerskripte, im Browser installiert werden. Diese verändern das Aussehen oder Verhalten einer Webseite und ermöglichen so *Active Browsing*. Diese Änderungen werden ohne dem Wissen oder der Zustimmung der Besitzer der Webseite vorgenommen. Ist ein Benutzerskript installiert und aktiviert, treten die Änderungen sofort nach dem Laden der jeweiligen Webseite in Kraft [5, 33] (siehe Abbildung 2.5). Obwohl Greasemonkey nur ein AddOn für Firefox darstellt, wird es oft synonym mit Benutzerskripten verwendet.

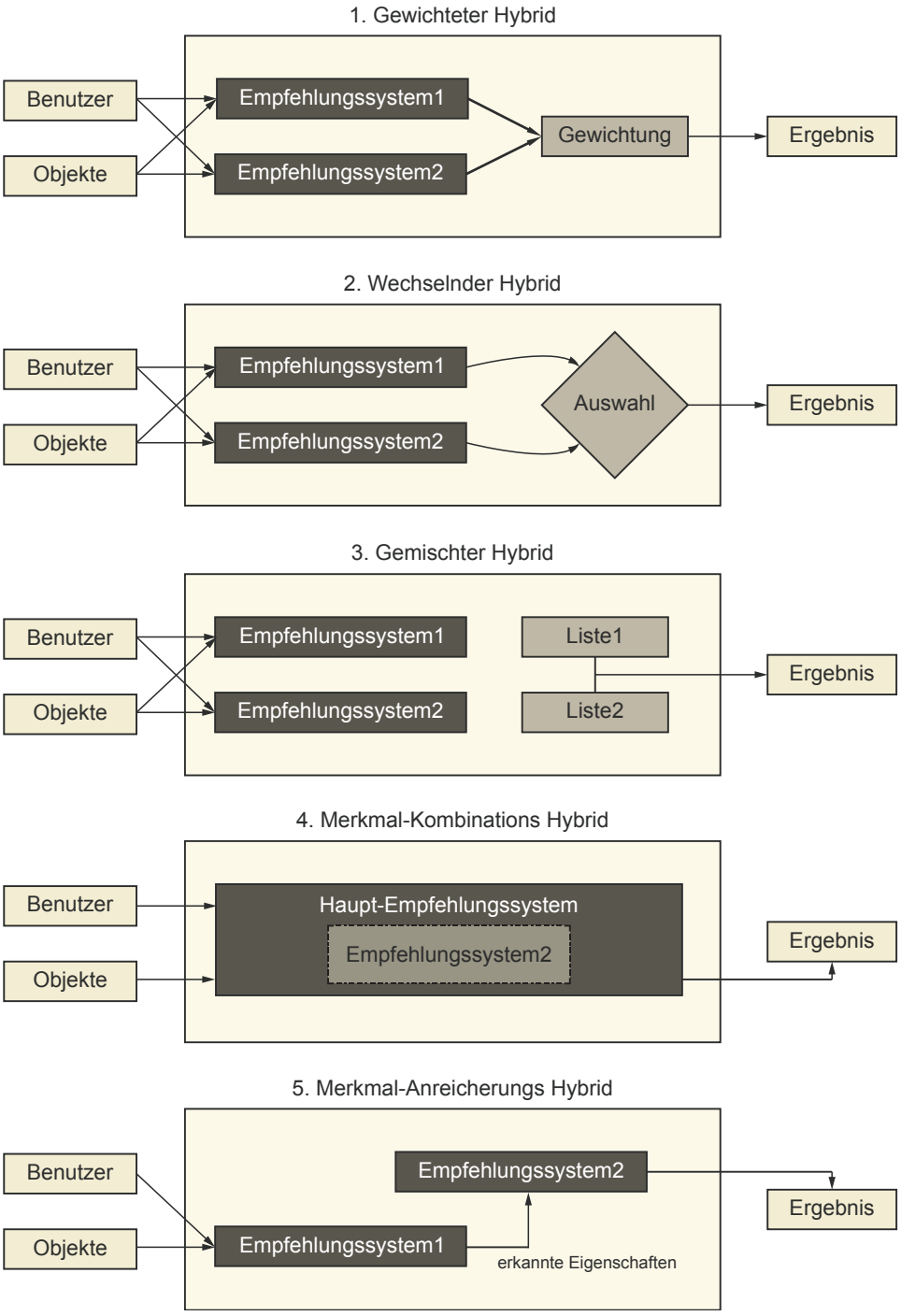


Abbildung 2.3: Teil 1 von 2 der sieben Ansätze für hybride Empfehlungssysteme nach Burke [10] (eigene Skizze nach [15]).

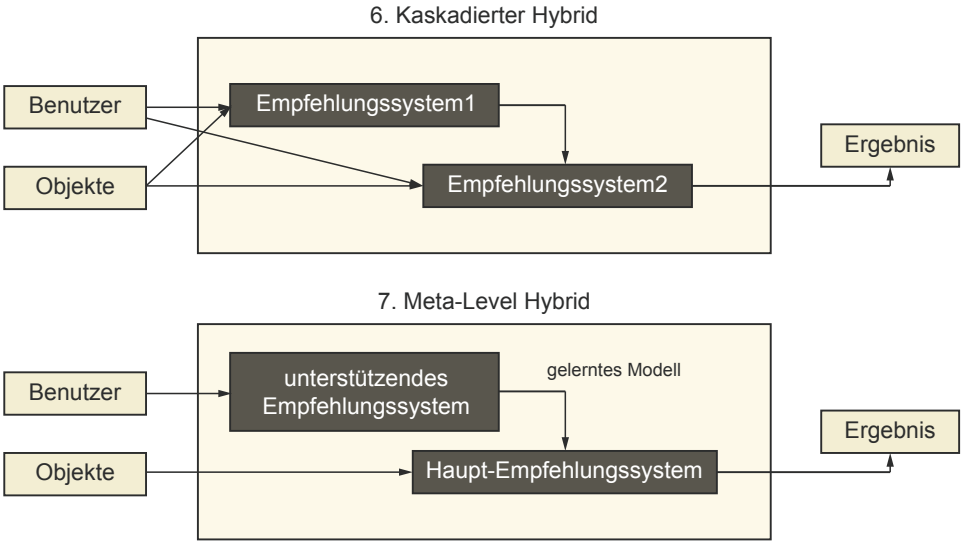


Abbildung 2.4: Teil 2 von 2 der sieben Ansätze für hybride Empfehlungssysteme nach Burke [10] (eigene Skizze nach [15]).

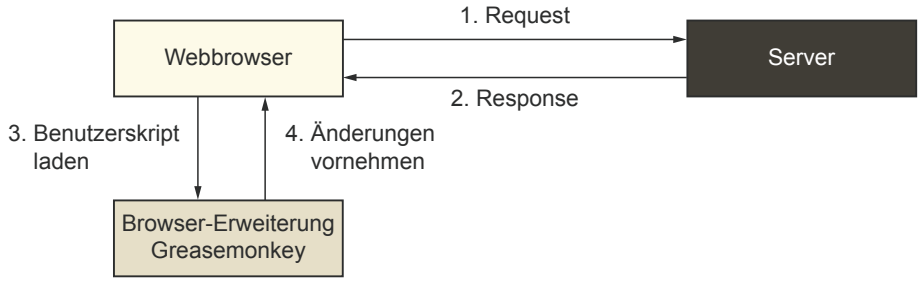


Abbildung 2.5: Ablauf eines Benutzerskripts

2.4.1 Aufbau

Benutzerskripte enden mit `*.user.js` und werden durch gewisse Metadaten ausgezeichnet:

```
1  // ==UserScript==
2  //
3  // @name Beispiel-Benutzerskript
4  // @description Dieses Benutzerskript gibt eine Nachricht als Alert aus.
5  // @namespace http://www.example-userscript.at/userscripts/first/
6  //
7  // Name Benutzerskript Benutzer-Id Link (optional)
8  // @author sabine (http://userscripts.org/users/sabine)
9  //
10 // @license GNU GPL v3 (http://www.gnu.org/copyleft/gpl.html)
11 //
12 // optionale Angabe, die fuer einen About-Link genutzt werden koennte:
```

```
13 // @homepage http://www.example-userscript.at/userscripts/first/
14 //
15 // Versionsnummer
16 // @version 1.0b
17 //
18 // Das Benutzerskript wird nur geladen auf:
19 // @include http://www.fh-ooe.at
20 //
21 // Das Benutzerskript wird nicht geladen auf:
22 // @exclude http://www.fh-ooe.at/footer/sitemap/fh-ooe/
23 //
24 // Zusaetzliche Bibliotheken, die geladen werden sollen, bevor das Skript laedt.
25 // @require https://ajax.googleapis.com/ajax/libs/jquery/1.6.0/jquery.min.js
26 //
27 // @history 1.0 first version
28 // @history 1.0b first beta version, who knew!!
29 //
30 // ==/UserScript==
31
32 jQuery(function(){
33   window.alert("Hallo, das ist ein Beispiel fuer ein Benutzerskript!");
34 });
```

Da Benutzerskripte nur aus simplem Javascript-Code bestehen, sind keine speziellen Werkzeuge zur Erstellung nötig. Skripte können entweder selbst geschrieben oder online heruntergeladen werden. Auf der Webseite Userscripts.org[20] finden Benutzer über tausende nicht kostenpflichtige Benutzerskripte.

2.4.2 Benutzerskripte vs. Plugins

In den Metadaten können Webseiten mit *@include* angegeben werden, bei denen das Benutzerskript geladen wird. Erst nachdem je nach Angabe eine oder mehrere Webseiten geladen wurden, wird das Benutzerskript abgearbeitet.

Plugins, auch Erweiterungen oder Add-ons, werden im Browser installiert und bereits bei dessen Start geladen. Sie sind zumeist bei allen Webseiten aktiv und nicht nur bei bestimmten. Möchte man also beispielsweise eine zusätzliche Funktionalität für eine bestimmte Seite, sollte man auf Benutzerskripte zurückgreifen. Dies wird durch laufende Prozesse im Browser verdeutlicht: Beim Starten des Browsers werden alle installierten Erweiterungen geladen und benötigen daher Arbeitsspeicher. Benutzerskripte haben daher einen geringeren Ressourcenverbrauch. Falls Benutzerskript und Plugin dieselbe Funktionalität bieten *können*, sollte also auf ersteres zurückgegriffen werden [51].

Erweiterungen werden grundsätzlich für bestimmte Browser geschrieben und können nicht übergreifend verwendet werden. Die APIs der einzelnen

[20] http://userscripts.org

Webbrowser und ihre Javascript-Unterstützung können zwar leicht variieren, mit gewissen Änderungen können aber Benutzerskripte erstellt werden, die browserübergreifend einsetzbar sind [52].

2.4.3 Herausforderungen

Das Firefox-AddOn Greasemonkey stellt für Benutzerskripte Methoden zur Verfügung, die großteils von anderen Browsern nicht genützt werden können. Beispielsweise können u. a. die Methoden `GM_registerMenuCommand`, mit der ein Event gefeuert wird, wenn ein Menüpunkt in der Firefox-Benutzeroberfläche geklickt wird, oder `GM_getResourceText`, die den Text einer externen Ressource lädt, nicht ausgeführt werden [52, 53].

Greasemonkey-Skripte werden in einer sogenannten Sandbox-Umgebung ausgeführt. Prevezanos [34] definiert *Sandbox* als eine Laufzeitumgebung, die dazu dient, Programme aus Sicherheitsgründen in einer eigenen Umgebung auszuführen. So wird das Benutzerskript vom eventuell maliziösen Code der Webseite isoliert und in einem eigenen Geltungsbereich, einem Scope, ausgeführt [54]. Google Chrome und Greasemonkey verhindern somit den Zugriff des Benutzerskripts auf den *Window-Scope* der Webseite. Im clientseitigen Javascript ist das Window-Objekt gleichzeitig das globale Objekt; das Benutzerskript hat also weder auf das Window-Objekt noch auf den globalen Geltungsbereich der Webseite Zugriff [14]. Durch diverse kreative Lösungen, die auch im eigenen Ansatz umgesetzt wurden, kann dies umgangen werden (siehe Abschnitt 4.2.2).

2.4.4 Browserunterstützung

Obwohl Benutzerskripte in einigen Versionen von Google Chrome direkt und einfach installierbar sind, bietet die Erweiterung *Tempermonkey*[21] zusätzliche Funktionalitäten wie Import- und Export-Funktionen und Unterstützung von allen Greasemonkey-Methoden.

Ab Version 8 bietet Opera native Unterstützung für Benutzerskripte und mit eventuell kleinen Änderungen können diese direkt installiert und aktiviert werden [52].

Für Safari bieten die Erweiterungen *GreaseKit*[22] und *NinjaKit*[23] die benötigten Voraussetzungen, um Benutzerskripte installieren und managen zu können.

[21] http://http://tampermonkey.net/
[22] http://http://8-p.info/greasekit/
[23] http://os0x.hatenablog.com/entry/20100612/1276330696

Kapitel 3

Verwandte Arbeiten

Um verschiedene hybride Empfehlungssysteme zu evaluieren, führte Burke eine Studie durch, deren Ergebnis für die Auswahl des Empfehlungssystems für *ReadBeyond* ausschlaggebend ist. Ferner werden Projekte, die denselben hybriden Ansatz wie *ReadBeyond* haben, oder von semantischen APIs unterstützt werden, vorgestellt.

3.1 Auswahl von hybriden Empfehlungssystemen

Um die Vor- und Nachteile der kategorisierten hybriden Empfehlungssysteme (siehe Abschnitt 2.3.2) zu präzisieren, zog Burke [10] das Restaurant-Empfehlungssystem Entree heran, da dieses Zugriff auf öffentlich zugängliche Benutzerprofile anbot. Mittels fallbasierter Techniken (siehe Abschnitt 2.3.2) wurden Restaurants selektiert und gereiht. Der Einstiegspunkt des Systems war beispielsweise ein Restaurant in einer beliebigen Stadt, welches dem Benutzer zusagte. Um in einer anderen Stadt ein ähnliches Lokal zu finden, konnte der Benutzer - ausgehend von einem konkreten, ähnlichen Vorschlag - mittels Kritiken wie „ruhiger“, „billiger“ etc. sein optimales Ziellokal, den Endpunkt der Anfrage, finden.

Ausschlaggebend für die Evaluierung des Systems war der ARC, der „average rank of the correct recommendation“. Er beschrieb den durchschnittlichen Rang eines positiv bewerteten Objekts, also die Position in der Liste von Empfehlungen von dem Objekt, welches der Benutzer positiv bewertete. Dieser Wert war durchschnittlich sehr hoch und für ein E-Commerce-System beispielsweise unpassend, da vom Kunden dort erwartet wird, nur die ersten zwölf Ergebnisse durchzusehen. Der Hauptgrund für diesen hohen ARC war die geringe Datenmenge, denn die Effizienz der kollaborativen und inhaltsbasierten Alogrithmen litt unter der spärlichen Anzahl von Bewertungen. Zusätzlich war die Studie limitiert durch implizite Bewertungen, kurze Benutzerprofile und die Domäne [10]. Getestet wurden inhaltsbasierte, wissensbasierte und zwei Arten von kollaborativen Filtertechniken sowie sechs

Typen von hybriden Empfehlungssystemen:

1. Gewichtet,
2. Wechselnd,
3. Merkmal-Kombination,
4. Merkmal-Anreicherung,
5. Kaskadiert,
6. Meta-Level.

Insgesamt wurden 41 Systeme evaluiert.

Zu den weniger effektiven Systemen zählten der gewichtete, wechselnde, Merkmal-Kombinations und Meta-Level Hybrid. Dennoch zeigten sie einen geringen Vorteil gegenüber den nicht hybridisierten Ansätzen. Der kaskadierte Hybrid zeigte eine deutliche Verbesserung gegenüber den anderen getesteten Kombinationen und den einfachen Empfehlungssystemen. Der Merkmal-Anreicherungs Hybrid erzielte die beste Leistung, vor allem in der Kombination von inhaltsbasierten und kollaborativen Techniken. Zwischen den Hybridisierungstechniken tauchten signifikante Unterschiede auf. Manche Systeme konnten z. B. das meiste aus einer stark-schwachen Kombination herausholen, andere nicht. Wählt man einen hybriden Ansatz aus, muss also genau überprüft werden, welche Ziele des Designs umgesetzt werden sollen (Gesamtgenauigkeit, cold-start Problem, etc.). Zusätzlich muss die Effizienz der einzelnen Systeme, vor allem zur Laufzeit, inspiziert werden. Da kollaborative Algorithmen mehrere Profile mit dem des Benutzers vergleichen müssen, sind jene die langsamsten. Der gewichtete Ansatz benötigt prinzipiell am meisten Zeit, da hier beide Recommender jede Anfrage verarbeiten müssen [10].

3.2 Verwandte Empfehlungssysteme

Aufgrund der guten Leistung des Merkmal-Anreicherungs Ansatzes wird auf diesen im kommenden Abschnitt näher eingegangen und Repräsentanten dieser Kategorie beschrieben. Da diese Arbeit Empfehlungssysteme im besonderen Kontext von Semantic APIs behandelt, werden außerdem Beispiele dieser Art vorgestellt.

3.2.1 Merkmal-Anreicherungs Hybride

Eine Art Merkmal-Anreicherungs Hybrid ist „content-boosted collaborative filtering“ (CBCF), also frei übersetzt inhalts-verstärktes kollaboratives Filtern. Mit diesem Ansatz wollten Melville und seine Kollegen [25] typische Nachteile des kollaborativen Filterns (siehe Abschnitt 2.3.2) wie Datenknappheit beseitigen. Für ihre Studien wurde eine Teilmenge schon bestehender, von Benutzern getätigter Film-Bewertungen eines Datensets gewählt. Die darin enthaltenen Filme wiederum waren mit der Internet Movie

Database (IMDb)[1] verknüpft, um Informationen wie Filmtitel, Regisseur, Schauspieler, Genre, Handlung, Keywords, Kommentare der Benutzer, externe Rezensionen, Rezensionen aus der Newsgroup und gewonnene Preise zu extrahieren und in einer Datenbank abzulegen. Da in dem Teildatensatz die meisten Filme kaum bewertet wurden, wurde ein inhaltsbasiertes System darauf trainiert, die Berwertungen mit prognostizierten Bewertungen zu erweitern. CBCF wurde im Vergleich zu einem reinen inhaltsbasierten, einem reinen kollaborativen System und einem einfachen hybriden Ansatz, der die durchschnittlichen Ergebnisse der ersten beiden empfiehlt, mit zwei unterschiedlichen Metriken getestet. Gegenüber den anderen erreichte der CBCF je nach Metrik und Recommender eine kleine Verbesserung von 4-9,7 %. Das heißt im Vergleich lieferte CBCF also nicht nur eher passendere, hochwertigere Empfehlungen, sondern verringerte auch die Wahrscheinlichkeit von schlechten Empfehlungen durch die Bewältigung der beim kollaborativen Filtern bekannten Probleme [25]. Wie *ReadBeyond* (siehe Kapitel 4) kombinerte der Prototyp von Melville et al.[25] ein kollaboratives mit einem inhaltsbasiertem System. Interessant hierbei ist allerdings der Fokus: Während ihr Haupt-Recommender ein kollaboratives System war, dessen Eigenschaften mithilfe eines inhaltsbasierten Systems ergänzt werden, ist dies bei *ReadBeyond* mit einem inhaltsbasiertem Haupt-Empfehlungssystem genau umgekehrt. Der Einsatz dieser zwei Ansätze wundert nicht: Bei der Studie von Burke (siehe oben) erzielte eine Kombination des inhaltsbasierten und kollaborativen Ansatzes bei Merkmal-Anreicherungshybriden eine der besten Leistungen.

LIBRA (Learning Intelligent Book Recommendation Agent) ist ein eigentlich inhaltsbasiertes Empfehlungssystem, welches dem Benutzer Bücher empfiehlt. Nötige Informationen stammen aus einer Datenbank, die aus Amazon[2] extrahierte Daten beinhaltet. Benutzer verwenden anschließend ein Traingingsset von Büchern, welche sie mit einer Skala von eins bis zehn bewerten, um so ein Benutzerprofil aufzubauen. Der inhaltsbasierte Recommender LIBRA nützt also die Informationen aus dem Web und Algorithmen, mit den aus Bewertungen des Benutzers ein Benutzerprofil erstellt wird, um die Nachteile von reinen inhaltsbasierten oder kollaborativen Empfehlungssystemen zu überwinden [27]. Genau wie *ReadBeyond* nützt LIBRA als Haupt-Empfehlungssystem ein inhaltsbasiertes, dessen Schwächen mit den Eigenschaften eines kollaborativen Systems als Merkmal-Anreicherungs Hybrid vermindert werden.

3.2.2 Empfehlungssysteme unterstützt durch Semantic APIs

Dayta.Me [2] ist ein Recommender, der aufgrund des persönlichen Online-Kalenders die bevorstehenden Aktivitäten eines Benutzers mit Informatio-

[1] http://www.imdb.com/
[2] http://www.amazon.com

nen anreichert. Die Empfehlungen basieren auf Personen, die der Benutzer treffen, Plätzen, die er besuchen und Aktivitäten, an denen er beteiligt sein wird. Unkonventionell sind dabei die empfohlenen Informationen: So werden nicht nur wie gewöhnlich News-Artikel oder Objekte aus sozialen Medien vorgeschlagen, sondern auch auf Echtzeit-Ressourcen wie Wetter und Staumeldungen und archivierte, statistische Daten wie aus dem öffentlichen Sektor oder der Regierung zugegriffen. Als beispielhaftes Szenario zur Nutzung schildern Al-Mahrubi et al. [2] den Interaktionsvorgang des Benutzers Mario. Mario nimmt als postgradualer Student an einer Konferenz in London teil. Das Event inklusive dessen Details vermerkt er in seinem Google Calendar[3]. Zwei Tage vor dem Ereignis, wird er u. a. an die abschließende Buchung eines Hotels erinnert. Zusätzlich werden ihm verschiedenste Informationen wie Tweets des Organisators der Konferenz oder passende News-Artikel oder Blog-Einträge empfohlen. Da Dayta.Me auch auf Kriminalitätsstatistiken zugreifen kann, wählt Mario aufgrund der hohen Kriminalitätsrate in Gebieten nahe der Konferenz ein Hotel in der Nähe des Veranstaltungsortes. Auch eine Karte mit Vorschlägen für diverse Sightseeing-Möglichkeiten, verknüpft mit Wikipedia-Artikeln[4], ist verfügbar. Wichtige Entitäten aus dem Eintrag im Kalender des Benutzers werden unter anderem mit der API OpenCalais (siehe Abschnitt 2.2.1) extrahiert. Insgesamt verwendet die Applikation 30 verschiedene Datenquellen und Services, die in drei Kategorien eingeteilt werden können: Datenquellen aus dem Kontext des Benutzers (z. B. das Event im Google Calendar), Datenquellen für Objekte, die dem Benutzer empfohlen werden (z. B. RSS Feeds oder Twitter) und Datenquellen, die während dem Vorgang der Empfehlung verwendet werden (wie OpenCalais). Der Haupt-Recommender ist in diesem Fall ein regelbasiertes Empfehlungssystem [2].

Abbar et al. [1] versuchen in einem neuen Ansatz News-Artikel nicht wie üblich aufgrund ihrer Popularität (meist gelesen, meist kommentiert) oder Aktualität, sondern auch aufgrund ihrer Kommentare zu empfehlen. Dabei wird auf die im Artikel und den Kommentaren enthaltenen Entitäten und Sentimente Wert gelegt. Besonders vielseitig seien diese, so meinen die Autoren, bei den meist aufgerufenen Artikeln. Empfohlen werden also solche Artikel, die in erster Linie eine bestimmte Relevanz zum gelesenen Artikel aufweisen, jedoch möglichst unterschiedliche Sentimente und Entitäten sowohl im Artikel als auch in den Kommentaren beinhalten. So können zwei Artikel relevant sein, indem sie beispielsweise dasselbe Thema Politik haben, jedoch kann ein Artikel über Barack Obama und der zweite über Angela Merkel handeln und sich so unterscheiden. Die einzelnen Entitäten (Themen, Personen, Orte) werden mithilfe der Semantic API OpenCalais (siehe Abschnitt 2.2.1) extrahiert [1].

Wie auch *ReadBeyond* nützen diese zwei beispielhaften Arbeiten seman-

[3] https://www.google.com/calendar/?
[4] http://de.wikipedia.org

tische APIs, um Entitäten aus Texten auszulesen, die dann genützt werden, um passenden Inhalt zu empfehlen.

Kollaborative Verschlagwortungssysteme leiden für gewöhnlich an Schwächen wie vielfachen Schlagworten mit derselben Bedeutung, falsche Schreibweisen oder Mehrdeutigkeit. Apostolski und seine Kollegen [4] entwickelten ein Social Bookmarking System, das den Benutzer bei der Verschlagwortung der Links mittels Extraktion von Entitäten und semantischer Annotation unterstützt. Das System liefert dem Benutzer außerdem weitere Empfehlungen aus der Community [4]. u. a. drei Engines sind Teil der Umsetzung:

- Bookmarking Engine: Mit ihr interagieren die Benutzer, wenn sie Seiten zu ihren Lesezeichen hinzufügen wollen. Sie empfängt außerdem Schlagwörter, die von der Recommender Engine vorgeschlagen werden.
- Recommender Engine: Zu den drei Arten von Empfehlungen in diesem System zählen Schlagwort-, Lesezeichen- und Benutzerempfehlungen. Zemanta ist hier verantwortlich für die Extraktion der Entitäten und die semantische Annotation. Empfehlungen von Lesezeichen können sowohl intern, also innerhalb des Systems, als auch extern sein. Letztere leiten auf Webseiten, die noch nicht im System gespeichert sind, weiter. Dabei werden der Inhalt von Zemanta analysiert und dann Empfehlungen von der API geliefert. Zuletzt können auch Benutzer empfohlen werden, die ähnliche Interessen haben.
- Semantic Engine: Um die Daten (Schlagwörter und Lesezeichen) auch außerhalb des Systems verarbeiten zu können, wird eine Ontologie herangezogen.

Alles in allem haben die Benutzer das Verschlagwortungs-Empfehlungssystem gut angenommen. Durchschnittlich 83 % der Empfehlungen wurden von den Benutzern so akzeptiert [4].

Wie *ReadBeyond* verwendet auch das System der Kollegen Zemanta, um sich externe Links (Artikel usw.) vorschlagen zu lassen.

Kapitel 4

ReadBeyond

Nachdem in der Einleitung in Kapitel 1 die Ausgangssituation beschrieben wurde, wird das Fehlen eines Werkzeuges bewusst, das Benutzern bei Recherchetätigkeiten im Internet unterstützt. Um einen Ansatz entwickeln zu können, der dem Benutzer einen deutlichen Mehrwert bietet, müssen die Anforderungen an das System definiert werden. Aufgrund derer können gewisse Designentscheidungen getroffen werden, auf denen wiederum das Konzept von *ReadBeyond* basiert. Wird dieses inklusive eines Designs der Benutzeroberfläche fertiggestellt, kann mit der Umsetzung des System begonnen und spezielle Lösungsansätze können gefunden werden.

4.1 Design

Das Design von *ReadBeyond* ist ausschlaggebend für seine Umsetzung. Es werden einzelne Designentscheidungen erklärt, die wiederum auf einer Anforderungsanalyse basieren. Um die Vorgangsweise bei der Nutzung zu verdeutlichen, schildert ein Szenario das Beispiel eines Studenten, der das Empfehlungssystem zur unterstützenden Recherche nützt. Darauf stützt sich die Beschreibung des Ablaufs, der von einem grafischen Prototypen unterstützt wird.

4.1.1 Anforderungen

Nach einer Analyse der Ausgangsituation in Kapitel 1 können nun mehrere Anforderungen festgelegt werden, die das Empfehlungssystem erfüllen soll, um dem Benutzer einen Mehrwert durch dessen Anwendung zu ermöglichen.

Browserübergreifende Anwendungsmöglichkeiten

Die Präferenzen bei der Nützung eines bestimmten Browsers sind von Benutzer zu Benutzer unterschiedlich. Manche nützen einen bestimmten für sämtliche Zwecke, manche nützen je nach Aufgabe unterschiedliche Browser.

ReadBeyond soll eine Browserunterstützung von 50 % erreichen, soll also von den gängigsten Anbietern unterstützt werden und bei der Hälfte der Benutzer installierbar sein.

Dezenz

Während einer Recherche darf der Benutzer nicht vom eigentlichen Thema, dem Inhalt der Webseite, abgelenkt werden. *ReadBeyond* soll den Benutzer bei seiner Arbeit unterstützen ohne dabei im Vordergrund zu stehen. Daher ist es unumgänglich, dass das Interface dezent ist und sich leicht ein- und ausblenden lässt. Wird der Benutzer von zu vielen Empfehlungen überwältigt, muss er dennoch Zeit aufwenden, um sich in der Masse an Vorschlägen zurechtzufinden und eine Entscheidung zu treffen. Daher ist es wichtig, dem Benutzer nur eine Auswahl an Webseiten zu empfehlen.

Relevanz/Genauigkeit

Der Benutzer soll darauf Vertrauen können, dass die empfohlenen Webseiten relevant für sein gewähltes Thema sind. Wählt er einen Link, soll dieser neue Erkenntnisse liefern können und themenbezogen sein. Trifft dies nicht zu, wird mehr Zeit des Benutzers beansprucht, indem er gezwungen wird andere Quellen zu suchen, und sein Vertrauen in das Empfehlungssystem an sich geschwächt.

Qualität

Aus ähnlichen Gründen wie bei der Relevanz sollen nur qualitativ hochwertige Webseiten empfohlen werden. Persönliche, unqualifizierte Kommentare oder beispielsweise Werbeseiten dürfen nicht in den Empfehlungen aufgelistet werden. Wird der Benutzer auf eine maliziöse Webseite weitergeleitet, kann dies das Vertrauen in das System erheblich erschüttern und seine Sicherheit gefährden.

Konfiguration

Die Schritte, die nötig sind, um das Empfehlungssystem anzuwenden (beispielsweise eine Installation), sollen für den Benutzer möglichst wenige sein.

Deaktivierung

Da Benutzer auch im Internet surfen können sollen, ohne dass Empfehlungen abgegeben werden, muss die Möglichkeit bestehen, *ReadBeyond* deaktivieren zu können. Dies kann eintreten, wenn Benutzer nicht wollen, dass gewisse Seiten vom System verfolgt werden, oder der Ressourcenverbrauch möglichst gering gehalten werden soll. Eine eventuelle Deaktivierung muss nicht nur

grundsätzlich ausführbar sein, sondern auch leicht und unkompliziert durchzuführen sein.

4.1.2 Designentscheidungen

Mithilfe der folgenden Designentscheidungen soll *ReadBeyond* die an das System gestellten Anforderungen (siehe Abschnitt4.1.1) erfüllen.

Benutzerskript

Die *Österreichische Web-Analyse* (ÖWA), ist eine Organisation, die - wie der Name bereits sagt - objektive Daten über die Nutzung des Online-Marktes erhebt und analysiert [55]. Monatlich wird auch die Browserverteilung in Österreich unter Internetnutzern ermittelt. Benutzerskripte können mittels Erweiterungen einfach in den Browsern Mozilla Firefox, Google Chrome, Safari und Opera installiert werden. Wie in Abbildung 4.1 ersichtlich, haben so laut der Browserstatistik der ÖWA vom Juli 2013 60,4 % der Internetnutzer die Möglichkeit Benutzerskripte zu installieren. Vergleicht man dies mit einer internationalen Browserstatistik von W3C, erkennt man dort einen ähnlichen Trend, der allerdings bereits in den letzten Jahren stattgefunden hat. Zählt man die Nutzung der oben erwähnten Browser im Juli 2013 zusammen, kommt man laut dieser Statistik sogar auf eine Reichweite von 86,9 %, aufgrund der starken Verwendung von Google Chrome und des geringen Gebrauchs des Internet Explorers [56]. Nicht beachtet werden dabei jedoch die unterschiedlichen Browserversionen. So kann beispielsweise Opera unter Version 8 Benutzerskripte nativ nicht installieren (siehe Abschnitt 2.4.4). Daher muss die Browserstatistik relativiert betrachtet werden. Doch wird durch vergleichende Daten des letzten Jahres auch deutlich, dass die Anwendung der Browser Chrome und Safari, bei denen eine einfache Installation möglich ist, deutlich steigt (siehe Tabelle 4.1). Somit kann gesagt werden, dass *ReadBeyond* als Benutzerskript die Anforderung erfüllt, von mehr als 50 % der Benutzer genützt werden zu können. Im Vergleich dazu können Plugins nur für spezifische Browser erstellt werden.

Da Benutzer unter Umständen nicht immer *ReadBeyond* verwenden wollen, soll es auch die Möglichkeit einer Deaktivierung geben. Auch dies spricht für die Verwendung eines Benutzerskripts. Diese sind wesentlich ressourcenschonender, da sie nur auf den jeweiligen Seiten, für die sie gedacht sind, aktiviert werden. Browser-Plugins laden bereits beim Öffnen des Browsers und verbrauchen somit mehr Arbeitsspeicher (siehe Abschnitt2.4.2).

Aus Gründen der Browserunterstützung (und somit der Verbreitungsmöglichkeiten) und der ressourcenschonenderen Nutzung, wird *ReadBeyond* als *Benutzerskript* anstelle einer Browser-Erweiterung umgesetzt. Zumal Benutzerskripte weitgehend nicht die Bekanntheit von Plugins genießen, wird *ReadBeyond* eben mit diesen umgesetzt, um die Handhabung mit ihnen bzw.

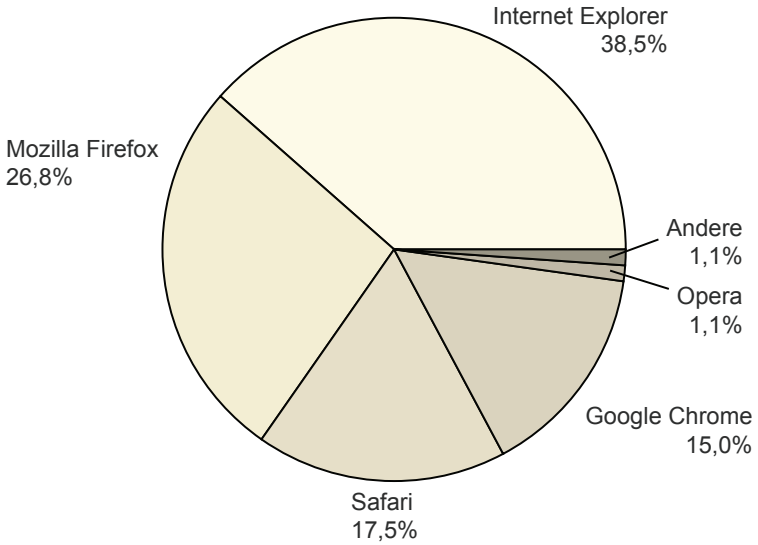


Abbildung 4.1: Browserstatistik Juli 2013 (eigene Skizze nach [55]).

	Februar 2012	*Juli 2013*
Internet Explorer	48,9 %	38,5 %
Firefox	30 %	26,8 %
Safari	9,4 %	17,5 %
Chrome	9,4 %	15,0 %
Opera	1,3 %	1,1 %
Andere	1,0 %	1,1 %

Tabelle 4.1: Gegenüberstellung der Browserstatistik in einem Zeitraum von 18 Monaten [55]

ihre Eigenschaften zu ergründen.

Semantic APIs

In ihrer Arbeit evaluiert Bauer [7], ob die Kombination mehrerer semantischer APIs einen Mehrwert gegenüber der effektivsten API, Zemanta[1], alleine bietet. Zemanta punktet durch eine konstante Leistung bei der Verschlagwortung sowohl von einfachen als auch von schwierigeren Texten. OpenCalais'[2] Vorteil im Vergleich dazu liegt besonders in der Identifizierung von Namen. Eine weitere Stärke sind die Social Tags; alles in allem liefert OpenCalais

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

jedoch auch falsche, unpassende Vorschläge. AlchemyAPI[3] gleicht OpenCalais mit kleinen Schwächen, bietet aber im Gegensatz zu den anderen beiden eine höhere Unterstützung unterschiedlichster Sprachen. Der Nutzen einer Kombination der drei APIs ist gering und „eine fehlerfreie Kombination aller APIs unmöglich“ [7], beim Ausfall einer API können jedoch die anderen einspringen [7]. Ursprünglich ausgewählt wurden diese drei von Bauer [7] aufgrund eines Blog-Artikels von DiCiuccio [57]. Er empfiehlt bei der Extraktion von Entitäten Zemanta als primäre API zu verwenden. Aufgrund der beiden Empfehlungen begrenzt sich die Auswahl der APIs für *ReadBeyond* auf diese drei. Im Prototypen wird primär AlchemyAPI zur Verschlagwortung eines Textes eingesetzt, um vor allem auch deutschsprachige Texte in die spätere Evaluierung miteinzubeziehen und testen zu können. Als einzige der drei bietet Zemanta Empfehlungen für weiterführende Webseiten an. Aus Ressourcengründen wird der Einsatz von semantischen APIs im Prototypen auf AlchemyAPI und Zemanta beschränkt, OpenCalais soll im fertig gestellten System zum Einsatz kommen.

Benutzerprofile

ReadBeyond nutzt das kollektive Wissen der Benutzer, die unterschiedliche Themenbereiche recherchieren. Entgegen den üblichen Empfehlungssystemen handelt es sich also nicht um bestimmte Thematiken wie Filme, Bücher oder News-Kategorien, die von den jeweiligen Systemen empfohlen werden. Vielmehr geht es darum, zu einem bestimmten Kontext Webseiten vorzuschlagen, die nicht nur von einer API empfohlen werden, sondern auch von anderen Benutzern als geeignet erachtet werden. Dadurch ist es nicht von Nutzen, wenn Benutzerprofile erstellt werden, die die Präferenzen von bzw. Daten über einzelne Benutzer festhalten, da individuellen Daten, die einem Profil zugeordnet werden können, keinen Wert für den Benutzer und somit auch für *ReadBeyond* haben.

4.1.3 Szenario

Paul ist ein 23-jähriger Germanistik-Student, der gerade mit seiner Seminararbeit mit dem Thema „Charakter-Parallelitäten bei J. W. von Goethe“ beginnt. Er möchte sich zuerst einen Überblick über den Autor verschaffen und tippt dessen Namen in eine Suchmaschine ein. Anschließend aktiviert er *ReadBeyond*, um Unterstützung bei seiner Recherche zu erhalten. Er wählt das erste Suchergebnis, den Wikipedia-Artikel von Goethe, aus, um sich in das Thema einzulesen. Da er seine Literatur nur aus Quellen wie Büchern oder wissenschaftlichen Arbeiten wählen darf, nutzt er in weiterer Folge die Fähigkeiten von *ReadBeyond*. In dessen Interface werden ihm fünf weiterführende Webseiten aufgelistet. Jede einzelne davon besitzt außerdem ein

[3] http://www.alchemyapi.com/

Rating, das Paul die Bewertung anderer Benutzer näher bringt. Er entscheidet sich für den obersten Link, da dessen Bewertungen am höchsten sind. Nach dem Laden der Webseite befindet sich anstelle der vorgeschlagenen, weiterführenden Links die Möglichkeit, selbst über diese Seite abzustimmen. Er stimmt ab, da er *ReadBeyond* zu mehr Genauigkeit und besseren Vorschlägen auch für andere Benutzer helfen will und setzt nachfolgend seine Recherche fort.

4.1.4 Interface und Ablauf

Um das Interface und den Ablauf der Nutzung von *ReadBeyond* zu veranschaulichen, wird an das Szenario aus dem vorigen Abschnitt angeknüpft.

Ist das Benutzerskript aktiviert und wird eine Seite geöffnet, erscheint das Interface von *ReadBeyond* (siehe Abbildung 4.2). Maximal fünf Empfehlungen werden dem Benutzer angezeigt, um ihm einen Überblick über die Vorschläge zu geben und ihn nicht zu überfordern. Die Benutzeroberfläche selbst ist dunkel und leicht transparent, um dahinterliegende Informationen nicht vollständig zu verbergen und ein Erahnen dieser möglich zu machen. Um die Möglichkeit zur Deaktivierung zu bieten, lässt sich die Oberfläche mittels Klick auf das nach unten zeigende Pfeilchen schließen (siehe Abbildung 4.3). Die Vorschläge sind nun nicht mehr sichtbar; somit kann der Benutzer vor allem sämtlichen Inhalt der Seite ungestört lesen. Will er *ReadBeyond* gerade nicht verwenden, wird eine Deaktivierung des Benutzerskripts empfohlen: Denn die aufgerufenen Seiten werden trotzdem aufgezeichnet und Empfehlungen generiert, nur nicht angezeigt.

Wie in Abbildung 4.4 ersichtlich, besteht eine Empfehlung visuell zuerst aus den Schlagwörtern, die die weiterführende Webseite beschreiben.

Sie sollen dem Benutzer die Möglichkeit geben, sich ein Bild über den Inhalt zu machen und ihm zu der Entscheidung zu verhelfen, der Empfehlung zu folgen, indem er auf den Link klickt, oder nicht. Rechts unten ist ein Sterne-Rating, welches die Bewertungen anderer Nutzer anzeigt. Bewertet wird hierbei aber nur die Relevanz der empfohlenen Webseite zum Kontext der gerade geöffneten. Die Sternchen-Bewertung ist skalenbasiert; Werte von ein bis fünf Sterne sind möglich. Ist die Relevanz groß, kann jedoch unter Umständen die empfohlene Webseite aufgrund ihrer Qualität nicht geeignet sein. Beispielsweise kann der Kontext passen, aber die Seite nicht wissenschaftlich sein, weil sie nur eine persönliche Meinung beinhaltet. Für diesen Fall gibt es eine zweite Bewertung, die mittels der Daumen links anzeigt, ob einem die Qualität der Seite zusagt oder nicht. Da es schwierig ist hier Abstufungen zu machen, wird bewusst nur eine binäre Wertung umgesetzt. Bewerten kann der Benutzer erst, wenn er die jeweilige empfohlene Seite besucht hat – somit sind alle Rating-Möglichkeiten deaktiviert und können nicht ausgewählt werden. Aus diesem Grund ist ein ganzer Empfehlungs-Block verlinkt, das heißt sowohl Schlagwörter als auch die zwei Bewertungsmöglichkeiten.

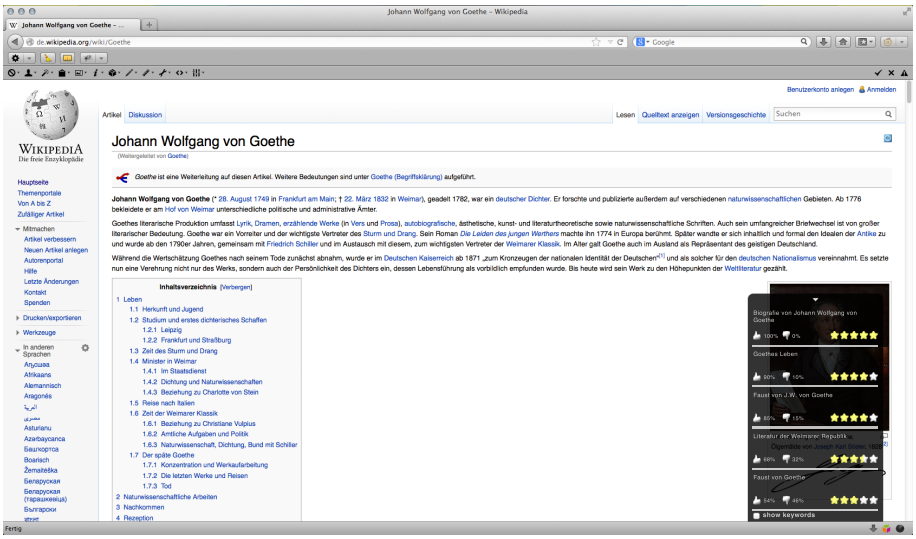


Abbildung 4.2: Benutzeroberfläche von *ReadBeyond* mit Keywords im Browser

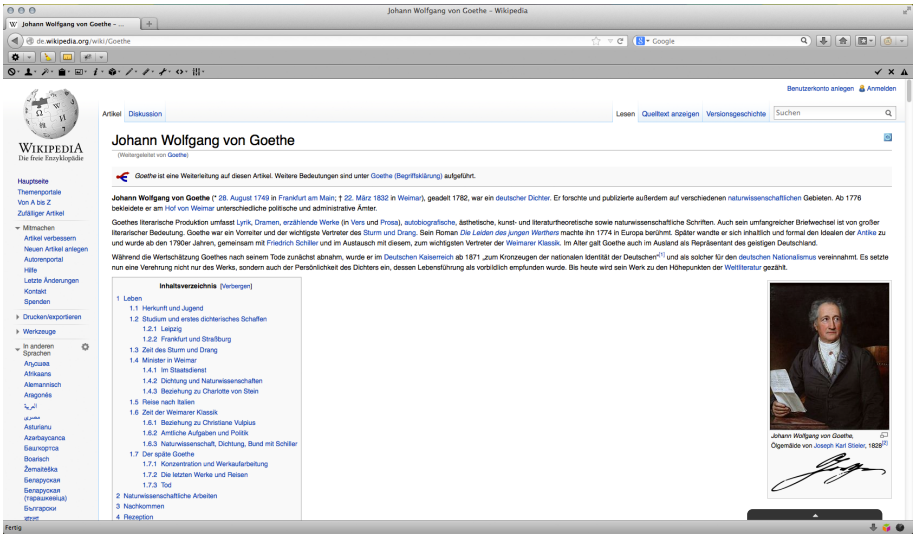


Abbildung 4.3: Geschlossene Benutzeroberfläche von *ReadBeyond* im Browser

Dies erleichtert dem Benutzer durch großzügige Klickflächen die Nutzung von *ReadBeyond* während einer Recherche.

Zusätzlich hat der Benutzer die Möglichkeit, sich anstatt der Schlüsselwörter die Titel der Webseite anzeigen zu lassen. Es kann durchaus sein, dass zwei Seiten im Groben eine Biografie Goethes beinhalten und somit

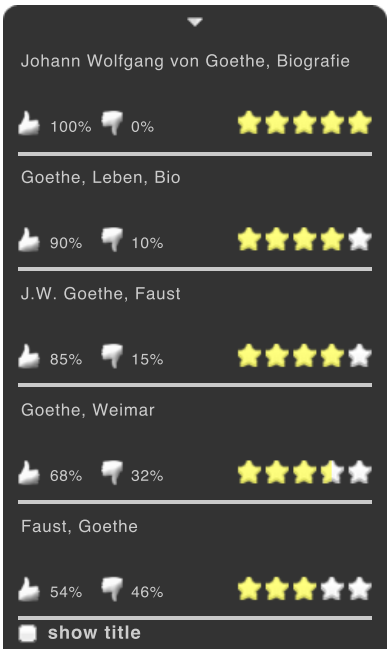


Abbildung 4.4: Vergrößerung des Designs von *ReadBeyond*

beispielsweise denselben Seitentitel „Biografie Goethe“. Ist der Text jedoch nicht derselbe, können die zugewiesenen Schlagwörter durchaus unterschiedlich sein und sind daher aussagekräftiger als der Seitentitel alleine. Daher werden in der Benutzeroberfläche von *ReadBeyond* standardmäßig die Keywords angezeigt und nicht die Seitentitel. Nichtsdestotrotz kann der Benutzer hier umschalten.

Wählt der Benutzer durch einen Klick einen weiterführenden Link aus, wird er auf die entsprechende Seite weitergeleitet. Anstelle der Empfehlungen befindet sich nun in der Oberfläche von *ReadBeyond* die Möglichkeit, selbst eine Bewertung abzugeben. Da der Mehrwert dieses Empfehlungssystems in der Kooperation mit den Benutzern liegt, sollen diese zu einer Kritik verleitet werden. Solange der Benutzer keine Bewertung abgibt, oder über einen zusätzlichen Klick zur Empfehlungsoberfläche zurückkehrt, werden keine Empfehlungen angezeigt. Der Benutzer wird vorrangig gebeten sowohl eine Bewertung für die Relevanz, als auch für die Qualität der empfohlenen Webseite abzugeben (siehe Abbildung 4.5). Möchte er dies nicht, hat er die Möglichkeit dies zu überspringen. Durch diesen Extra-Klick soll erreicht werden, dass der Benutzer eher eine Evaluierung abgibt, da dies nur einen geringen Mehraufwand mit sich bringt. Es wird jedoch davon ausgegangen, dass die Zielgruppe von *ReadBeyond* dem System gegenüber aufgeschlossen ist und den Mehrwert einer Bewertung erkennt, also grundsätzlich gerne bereit ist, diese abzugeben. Um den Benutzer zu einer Bewertung zu „ver-

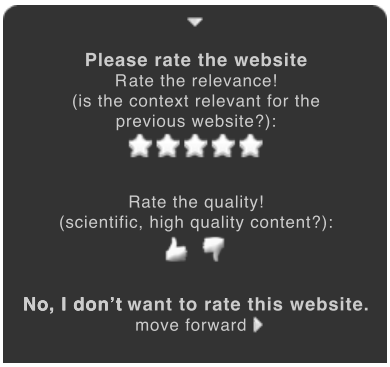


Abbildung 4.5: Vergrößerung der Bewertungsoberfläche von *ReadBeyond*

führen“, könnte die Evaluierungoberfläche auffallender auf der Seite platziert werden. Dabei darf man jedoch nicht vergessen, dass ein Benutzer ja den Inhalt einer Seite erfassen können muss, um darüber urteilen zu können. Daher wurde diese Oberfläche anstelle der Empfehlungen platziert, um so einen ungestörten Blick auf die Webseite bieten zu können.

Aus Ressourcengründen wird im Prototyp nur die skalenbasierte Bewertung angezeigt und auch der Benutzer kann selbst nur skalenbasiert bewerten. Dies steht dann gebündelt für die Relevanz und die Qualität der empfohlenen Webseite. Im Prototyp wird außerdem nur das Auswählen von ganzen Sternen bereit gestellt. Nach gründlichen Benutzerstudien kann angedacht werden, auch halbe Sterne zu vergeben.

4.2 Umsetzung des Prototyps

Nachdem die Anforderungen analysiert und ein Konzept entwickelt wurde, wurde *ReadBeyond* umgesetzt. Im Zuge dieser Arbeit sollen ein Überblick über die Architektur hinter dem Empfehlungssystem gegeben und interessante Lösungsansätze betrachtet werden. Der Prototyp dient als Grundlage für eine spätere Evaluierung.

4.2.1 Systemarchitektur

Der Ablauf zwischen den einzelnen beteiligten Komponenten wird in diesem Abschnitt näher beschrieben. Dabei wird erläutert, wie der Benutzer mit dem Browser interagiert, dieser Informationen an den Webserver weiterleitet und anschließend mit den verwendeten APIs und der Datenbank kommuniziert. Es wird weiters ein Einblick in die Zusammenhänge zwischen den verschiedenen Dateien am Server gegeben.

Verwendete Technologien

Clientseitig verwendet *ReadBeyond* im Benutzerskript die freie Javascript-Bibliothek *jQuery*[4], die beispielsweise Navigieren im DOM erleichert. Das *Document Object Model (DOM)* ist eine plattform- und sprachunabhängige API und erlaubt Sprachen wie Javascript den Zugriff bzw. die Bearbeitung des Inhalts, der Struktur und dem Aussehen von HTML- und wohlgeformten XML-Dokumenten [23]. Der Aufbau eines Benutzerskriptes wird in Abschnitt 2.4.1 erläutert.

Serverseitig wird *PHP*[5] eingesetzt, eine freie Skriptsprache, die auch in HTML eingebaut werden kann. Mittels PHP wird z. B. die Verbindung zur Datenbank und den verwendeten APIs hergestellt. Die Daten werden in MySQL, einem Datenbank-Management-System, gespeichert.

Um zwischen Client und Server zu kommunizieren, wird *Ajax* mithilfe einer *jQuery*-Funktion eingesetzt. *Ajax* (Asynchronous Javascript and XML) ermöglicht asynchrone Kommunikation mit dem Server, die es möglich macht, Teile einer Seite ohne komplettes Neuladen nachzuladen.

Ablauf im Client-Server-Modell

Die folgenden Schritte beschreiben die Interaktion zwischen Benutzer, Browser, Webserver, Datenbank und den beiden im Prototyp verwendeten APIs. In einem Sequenzdiagramm in Abbildung 4.6 sind diese visualisiert.

1. Der Benutzer öffnet eine beliebige Seite im Browser durch die Eingabe einer URL.
2. Das Laden der neuen Webseite löst einen Ajax-Call aus und die URL der Seite wird an ein PHP Skript am Webserver übermittelt.
3. Eine neue Datenbankverbindung wird eröffnet. Gleichzeitig wird auch mit AlchemyAPI eine Verbindung hergestellt.
4. Es wird in der Datenbank abgefragt, ob bereits eine Webseite diesen Ursprungs gespeichert ist.
5. Die Webseite wird gegebenenfalls zurückgeliefert.
   (a) Ist die Webseite noch nicht vorhanden, wird sie angelegt.
   (b) Ist die Webseite vorhanden, wird zum nächsten Schritt übergegangen.
6. Der Text der Webseite wird mittels der PHP-Methode `file_get_contents()` abgefragt.
7. Der Browser liefert den Text dann an den Webserver.
8. Anschließend werden für den übermittelten Text von der API Zemanta weiterführende Artikel und Markups angefordert.

[4] http://jquery.com/
[5] http://php.net/

9. Diese liefert Zemanta zurück an den Webserver.
10. Am Webserver werden diese Empfehlungen verarbeitet: Jene mit einem Relevanzwert über einer bestimmten Grenze werden gefiltert.
11. Anschließend werden alle Empfehlungen zur Ursprungsseite aus der Datenbank angefordert.
12. Diese liefert als Antwort die angeforderten Empfehlungen.
13. Sind die gefilterten Empfehlungen der API Zemanta bereits in den Ergebnissen der Datenbank inkludiert?
    (a) Falls nicht, speichere die Empfehlung in der Datenbank ab.
    (b) Falls die Empfehlungen bereits inkludiert sind, wird zum nächsten Schritt übergegangen.
14. Von AlchemyAPI werden Keywords zum Inhalt der empfohlenen Seite angefordert.
15. Die API liefert die Keywords.
16. Anfrage an die Datenbank, ob das jeweilige Schlagwort bereits gespeichert ist.
17. Die Datenbank liefert ein leeres oder gefülltes Array mit dem enthaltenen Schlagwort, falls es gespeichert ist.
    (a) Ist das entsprechende Schlagwort noch nicht gespeichert, wird es in der Datenbank abgelegt.
    (b) Ist das entsprechende Schlagwort gespeichert, wird zum nächsten Schritt übergegangen.
18. Die Empfehlungen für die Ursprungswebseite inklusive Schlagwörtern werden an das Benutzerskript übergeben.
19. Die Benutzeroberfläche im Frontend wird befüllt.
20. Der Benutzer wählt einen Link aus.
21. Dadurch wird ein weiterer Ajax-Call gesendet und parallel wird eine Variable im Local Storage gesetzt, die festlegt, dass der Benutzer auf der Evaluierungsseite landen wird. Die neue Seite wird geladen.
22. Der Browser zeigt dem Benutzer nun das Evaluierungsfenster an.
23. Die Aufrufe der ausgewählten Empfehlung werden in der Datenbank erhöht.
24. Der Benutzer evaluiert die Seite und bewertet sie somit mit einer gewissen Punktezahl.
25. Ein Ajax-Call mit der ID der Empfehlung und deren Wertung werden an den Webserver übermittelt.
26. Eine neue Evaluierung wird in der Datenbank angelegt.
27. Am Webserver werden die Punkte des Benutzers in die bisherige Bewertung miteinberechnet.

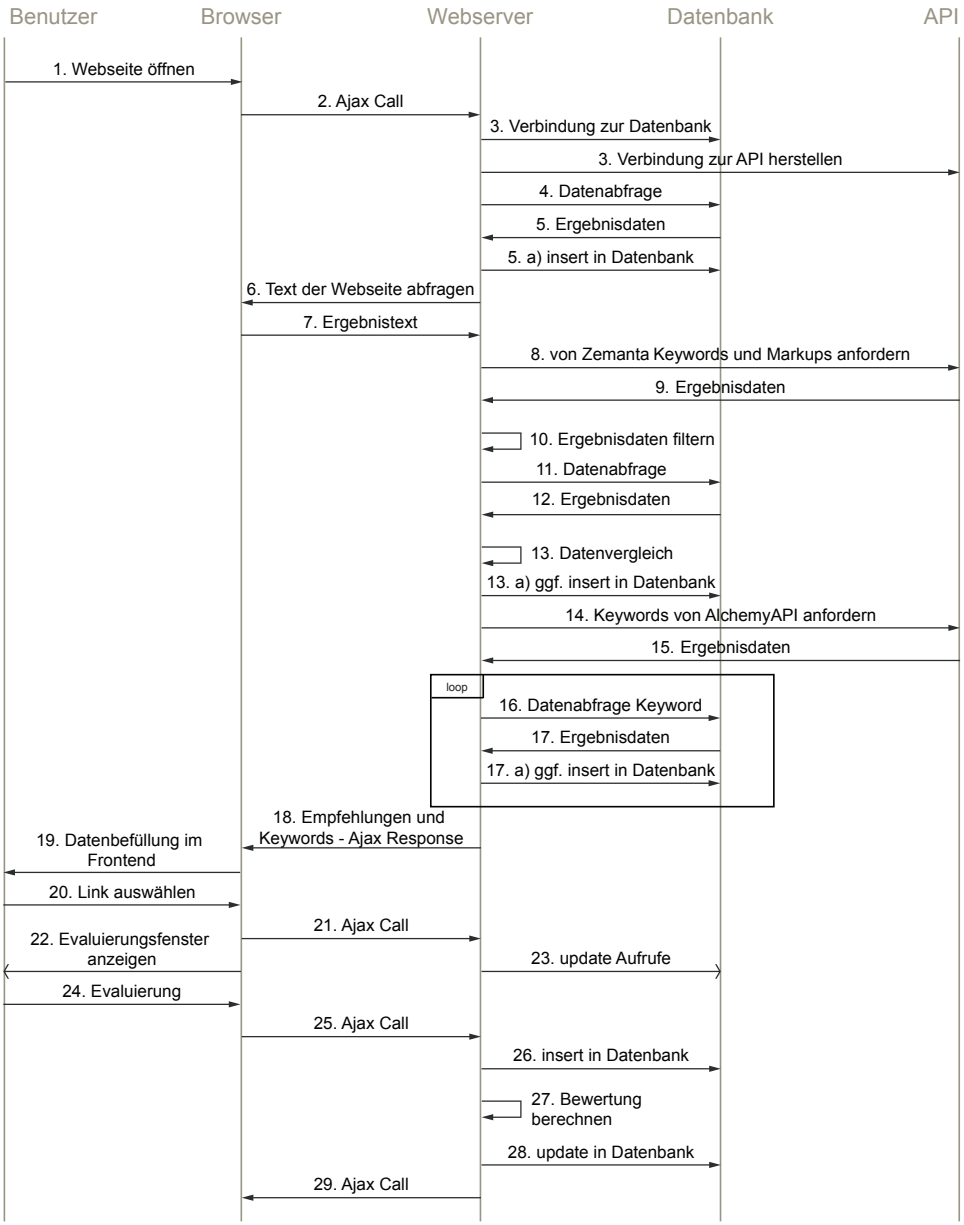


Abbildung 4.6: Sequenzdiagramm des Client-Server-Modells von *ReadBeyond*

28. Der neue Wert wird in der Datenbank aktualisiert.
29. Ajax sendet eine Antwort an das Benutzerskript.
30. Woraufhin wieder Empfehlungen für die neue Seite angezeigt werden, das heißt Schritt 2 wiederholt sich und ein Ajax-Call wird durchgeführt.

Clientseitige Architektur

Auf der Seite des Clients befindet sich lediglich das Benutzerskript. Dieses ist hauptsächlich der Einstiegspunkt und für die Annahme der aufbereiteten Daten maßgebend. Der Aufbau eines Benutzerskriptes allgemein wurde bereits in Abschnitt 2.4.1 erklärt.

ReadBeyond wird auf allen Seiten inkludiert, außer der Suchmaschine Google. Dies ist die erste Anlaufstelle für die meisten „Forschenden“ und liefert an sich bereits Vorschläge (in Form von Suchergebnissen) zu einer bestimmten Suchanfrage.

Die gesamte Logik des Systems liegt auf dem Webserver.

Serverseitige Architektur

Der erste Ajax-Call des Benutzerskripts wird von der Datei `processData.php` verarbeitet. Darin wird zuerst die Klasse `Database` aus `dbConnection.php` instanziert. Weiters wird darin mit der API Zemanta kommuniziert, woraufhin empfohlene Artikel und Markups von der API empfangen werden. Diese werden an die Instanz der Datenbank-Klasse `Database` weitergeleitet und verarbeitet: Gegebenenfalls werden Einträge in der Datenbank hinzugefügt und die Schlagwörter zu den neuen Empfehlungen gesucht. Die gefilterten Ergebnisse werden dann von `processData.php` mittels einer Ajax-Response ans Benutzerskript geliefert und im Frontend ausgegeben.

Alle Interaktionen mit der Datenbank gehen nur von `Database` aus. Die Klasse ist zuständig für das Auslesen der Daten aus der Datenbank, um diese mit Daten der APIs zu vergleichen und gegebenenfalls zu ergänzen.

Neben `processData.php` werden auch noch `processRecommendation.php` und `processEvaluation.php` in diesem Fall als verarbeitende php-Files bezeichnet. Sie empfangen die Ajax-Anfragen und etwaige Daten aus dem Benutzerskript und leiten letztere an eine Instanz von `Database` weiter. Die bearbeiteten oder gefilterten Daten werden von den verarbeitenenden Dateien zurück ans Benutzerskript gesendet.

AlchemyAPI stellt zwei Klassen zur Verfügung, die – in die eigene Webanwendung eingebunden – die Kommunikation mit der API erleichtern.

Die serverseitige Architektur wird auf Abbildung 4.7 verdeutlicht.

4.2.2 Implementierung

ReadBeyond beinhaltet zwei interessante Problemlösungsansätze. Zum einen ist dies die Einbindung von jQuery bzw. des restlichen Skripts in die vom Benutzer geöffnete Webseite. Zum anderen wird ein Algorithmus entwickelt, der den hybriden Ansatz möglich machen soll, indem die kollaborativen Bewertungen der Benutzer in die Daten der API miteinfließen.

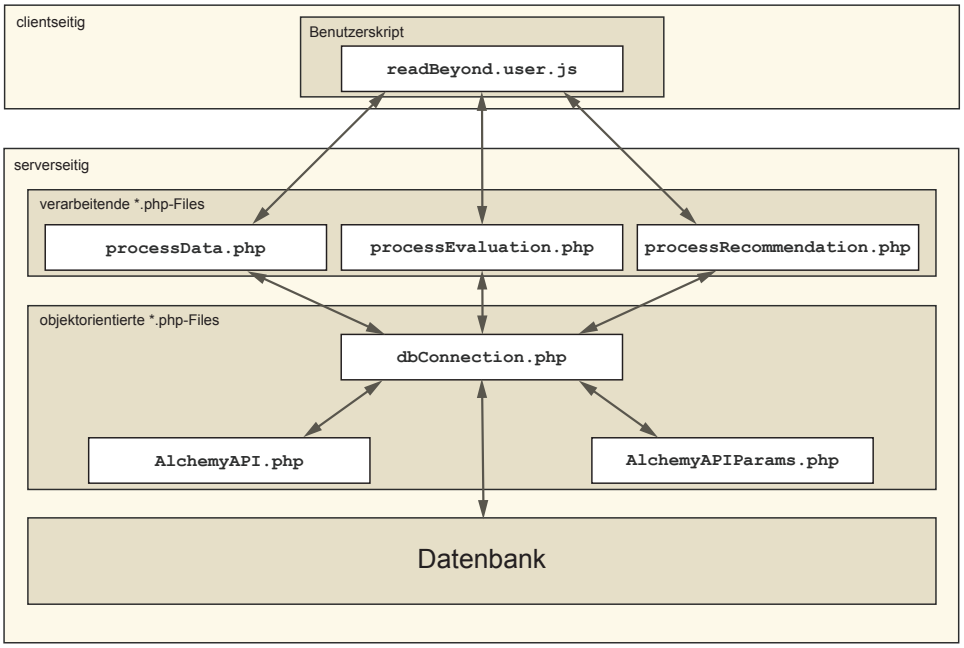


Abbildung 4.7: Serverseitige Architektur von *ReadBeyond*

Einbindung jQuery

Einige Greasemonkey-Funktionen wie `@require` werden von Browsern wie Google Chrome ignoriert. Ist jQuery bereits in eine Webseite integriert, muss die Javascript Bibliothek für das Benutzerskript aufgrund des Sandbox-Modes (siehe Abschnitt 2.4.3) explizit eingebunden werden. Um `@require` z. B. in anderen Browsern und in Google Chrome auch ohne der Extension Tampermonkey nutzen zu können, jQuery innerhalb der Seite laden zu können und das Benutzerskript unter dem globalen *Windows-Scope* laufen zu lassen, muss eine kreative Lösung gefunden werden.

Eine mögliche Lösung schlägt Banks [6] auf Stack Overflow[6], einer Frage- und Antwortplattform, vor. Der gekürzte Code wird im Skript nach den üblichen Metadaten eines Benutzerscripts eingefügt. Im Detail besteht der Code aus drei Funktionen, die im folgenden Teil näher beschrieben werden: `load()`, `execute()` und `loadAndExecute()` (siehe Programm 4.2.2 nach [6]).

Die Funktion `load()` lädt eine bestimmte `url` in das jeweilige Web-Dokument. Optional können Callbacks (Rückruffunktionen) als Parameter angegeben werden, die bei erfolgreichem oder fehlgeschlagenem Laden in Kraft treten. Zuerst wird mit Javascript ein `script`-Element generiert, dessen URL der übergebene Parameter wird. Da jQuery ja erst mit diesem Workaround geladen wird, kann die Bibliothek für diesen Schritt noch nicht

[6] http://stackoverflow.com/

genutzt werden. Sind die Parameter `onLoad` oder `onError` gesetzt, wird ein Event Listener auf das Skript gesetzt, der bei Eintreten der beiden Ereignisse das zugehörige Callback-Event ausführt. Zuletzt wird das Skript an den HTML Body angehängt.

Entweder eine Funktion oder Code wird `execute()` als Parameter übergeben. Es wird geprüft, ob der Typ des Parameters eine Funktion ist. Falls ja, werden um den Funktionsaufruf runde Klammern eingefügt, sodass man folgendes Konstrukt erhält: `(functionOrCode)();`. Dies bedeutet, dass die Funktion nicht nur deklariert, sondern auch gleich aufgerufen wird [14]. Ist der Parameter reiner Code ohne Funktionsdeklaration, wird mit diesem Code-Stück ohne Veränderung weitergearbeitet. Wieder wird ein `script`-Element generiert und ihm der eventuell veränderte Parameter als Text zugewiesen. Anschließend wird das Skript wieder an den Body angehängt. Die Funktion läuft jetzt unter dem globalen *Windows-Scope*.

`loadAndExecute()` verbindet schließlich diese zwei Funktionen und macht es möglich, zuerst ein Skript von einer bestimmten `url` zu laden, dann `functionAndCode` einzufügen und im Anschluss sofort auszuführen. Alle drei Funktionen sind im folgenden Programmabschnitt anschaulich umgesetzt [6]:

```
1 function load(url, onLoad, onError) {
2     e = document.createElement("script");
3     e.setAttribute("src", url);
4 
5     if (onLoad != null) { e.addEventListener("load", onLoad); }
6     if (onError != null) { e.addEventListener("error", onError); }
7 
8     document.body.appendChild(e);
9 
10    return e;
11 }
12 
13 function execute(functionOrCode) {
14    if (typeof functionOrCode === "function") {
15        code = "(" + functionOrCode + ")();";
16    } else {
17        code = functionOrCode;
18    }
19 
20    e = document.createElement("script");
21    e.textContent = code;
22 
23    document.body.appendChild(e);
24 
25    return e;
26 }
27 
28 function loadAndExecute(url, functionOrCode) {
29    load(url, function() { execute(functionOrCode); });
30 }
```

ReadBeyond setzt diesen Workaround von Banks [6] ein, um jQuery brow-

serübergreifend zu laden und anschließend den eigentlichen Code des Benutzerskriptes im *Windows-Scope* auszuführen.

Kombination Relevanzwert und kollaborative Bewertung

Ein Auswahlkriterium für die Empfehlungen von *ReadBeyond* ist der Relevanzwert, der bereits von Zemanta mitgeliefert wird. Dieser ist eine Gleitkommazahl und liegt zwischen 0 und 1. Der Wert beschreibt die Relevanz der empfohlenen Entität zum ganzen Text. Da *ReadBeyond* in die Kategorie der hybriden Empfehlungssysteme fällt, muss neben dem inhaltsbasierten auch der kollaborative Teil, die Bewertungen der Benutzer, miteinfließen. Auch auf das Alter muss Acht gegeben weden: Ältere Empfehlungen mit ausgezeichneten Bewertungen können nahezu unschlagbar und immer wieder empfohlen werden. Dadurch besteht die Gefahr, dass neue Empfehlungen mit ähnlicher Relevanz möglicherweise im Vergleich zu obsoleten, aber hoch bewerteten untergehen. Folglich muss es eine Möglichkeit geben den berechneten Relevanzwert einer Empfehlung proportional zu ihrem Alter zu verringern. Zusätzlich soll auch die Anzahl an Aufrufen in den Wert einfließen: Wird eine Empfehlung kaum aufgerufen, wird angenommen, dass die Benutzer mit den Schlagwörtern oder dem Titel nicht viel anfangen können und der Relevanzwert dadurch gemindert.

Bewertet der Benutzer eine Empfehlung, wird in weiterer Folge die Funktion `handleEvaluations()` am Server aufgerufen. Ihr werden zum einen die ID der Empfehlung, zu der die Evaluierung gehört, und zum anderen die skalenbasierte Bewertung selbst, also eine Ziffer zwischen 0 und 5, mitgegeben. Anschließend werden nötige Daten über die Empfehlung mit der mitgebenen ID aus der Datenbank ausgelesen:

- `relevance_calculated`: die bereits kalkulierte Relevanz; falls die Empfehlung neu ist und es dadurch weder Bewertungen noch eine altersbasierte Minderung der Relevanz gibt, ist dieser Wert 0.
- `relevance_original`: der von der API gelieferte Relevanzwert
- `timestamp`: beschreibt den Zeitpunkt des Eintrags der Empfehlung in der Datenbank
- `evaluation_mean`: die durchschnittliche Bewertung aller Benutzer zwischen 0 und 5; wurden noch keine Bewertungen getätigt, gilt: $evaluation\_mean = 0$.

Um einen Überblick über den Aufbau der Datenbank zu bekommen, ist das Modell in Abbildung 4.8 visualisiert.

Zuerst wird die Periode berechnet, seit der die Empfehlung bereits in der Datenbank gespeichert ist. Nachdem ein Zeitstempel des gegenwärtigen Zeitpunkts angelegt wurde, wird diese von dem eingetragenen abgezogen: Man erhält die Periode, seit der die Empfehlung in der Datenbank gelistet ist. Anschließend werden alle skalenbasierten Bewertungen `evaluation_score` aller

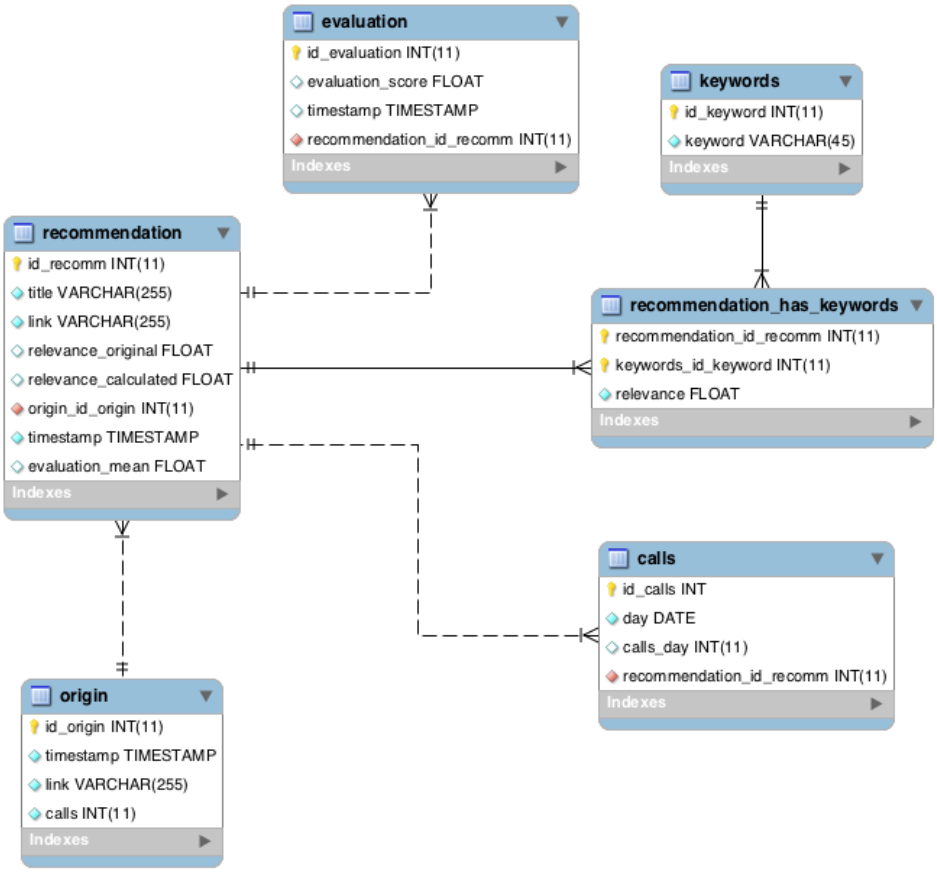


Abbildung 4.8: Datenbankmodell von *ReadBeyond*

Evaluierungen, die zu dieser Empfehlung bereits in der Datenbank eingetragen sind, ausgelesen. Mit den Daten wird ein Array befüllt, um festzustellen wie viele 1- bis 5-Stern Bewertungen es gibt. So bedeutet beispielsweise ein Array `z = (50,0,12,5,4)`, dass 50 Personen die Empfehlung mit einem Stern, null Personen mit zwei, zwölf Personen mit drei, fünf Personen mit vier und vier Personen mit fünf Sternen bewertet haben.

Zuerst wird das arithmetische Mittel m (siehe Gleichung 4.1) der Benutzerbewertungen berechnet. Jede Anzahl an Bewertungen einer bestimmten Sternezahl b, also der jeweilige Wert im Bewertungs-Array, wird mit seiner jeweiligen Position i im Array multipliziert. i beginnt bei 1 und endet bei 5; das heißt die durch die Multiplikation entstandenen Werte werden fünf Mal summiert. Dividiert man dieses Ergebnis durch die Summe aller abgegebenen Bewertungen, erhält man den arithmetischen Mittelwert.

$$m = \frac{\sum_{i=1}^{5} b_i \cdot i}{\sum_{i=1}^{5} b_i}. \tag{4.1}$$

Farmer und Glass [13] stoßen nun auf folgendes Problem: Arithmetische Mittelwerte können nicht miteinander verglichen werden, wenn die Bewertungen von einer signifikant unterschiedlichen Anzahl an Benutzern durchgeführt wurden, wie folgendes Beispiel zeigt:

1. Eine Entität erhält durch die Bewertung von *3* Personen einen Mittelwert von 4,667. Da die Zahl aufgerundet wird, ergibt dies ein Sterne-Rating von 5 vollen Sternen, obwohl nicht alle Personen die volle Punktezahl bewertet haben.
2. Dieselbe Entität erhält durch die Bewertung von *50* Personen einen Mittelwert von 4,432, was nach dem Runden vier volle Sterne bedeuten würde.

Das zweite Beispiel repräsentiert die Meinung der Benutzer besser und ist daher aussagekräftiger. Dadurch können Entitäten, die wie im Beispiel von drei Benutzern unterschiedlich bewertet werden, an die Spitze von Bewertungslisten kommen und über solchen, die öfter und eventuell höher bewertet werden, stehen. Die beiden entwickelten daher einen Algorithmus, der auf der *Anzahl* der Ratings basiert (siehe Gleichungen 4.2 und eq:rankMean). Durch die Gleichungen entsteht eine Kurve, die oben und unten begrenzt ist (siehe Abbildung4.9). Dies bedeutet, dass das arithmetische Mittel unter und über einer bestimmen Anzahl an Bewertungen einen gewissen Wert nicht unter- bzw. überschreiten darf.

$$l = \min(\max((n - f)/c, 0), 1) \cdot 2, \tag{4.2}$$

$$g = m - a + l \cdot a. \tag{4.3}$$

Zuerst wird die Gewichtung der Anzahl l (liquidity weight) berechnet. Dafür benötigt werden die Gesamtanzahl an Bewertungen n, die Untergrenze f (liquidity floor) und Obergrenze c (liquidity ceiling) an Bewertungen. Die letzten beiden sind Konstanten, deren Wert von Farmer und Glass [13] vorgeschlagen wird. f ist dabei die Mindestanzahl an Benutzereingaben, die für einen positiven Effekt auf die Bewertung nötig sind. Dieser Wert wird hier mit 10 belegt. Alle Bewertungen über c, dessen empfohlener Wert bei 60 liegt, bekommen keinen gewichteten Bonus. Somit wird einer Anzahl von 60 Personen zugetraut, einen repräsentativen Mittelwert zu generieren. Die dritte Konstante, die zum Einsatz kommt, ist der Anpassungsfaktor a (adjustment factor). Dieser Wert wird vom arithmetischen Mittel abgezogen oder hinzugefügt – je nach Anzahl an Bewertungen. Der Wert muss in dem

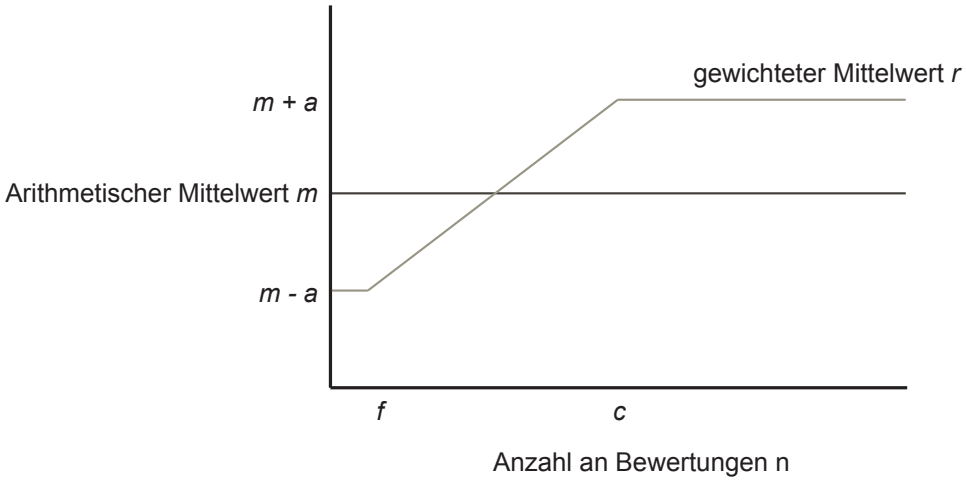


Abbildung 4.9: Kompensation von skalenbasierten Bewertungen durch die Gewichtung der Anzahl an abgegebenen Bewertungen (Eigene Skizze nach [13])

Bereich eines möglichen Rundungsfehlers liegen und beträgt im Beispiel der Autoren 0,1. Ist der Wert zu hoch, werden zu viele 4-Stern Bewertungen höher eingestuft. Ist der Wert zu niedrig, hat er den gegenteiligen Effekt [13].

Der gewichtete Mittelwert von Farmer und Glass [13] wird auch auf Seiten von Yahoo![7] eingesetzt. Da dort täglich viele neue Entitäten publiziert werden, bekommen einzelne oft nur wenige Benutzerbewertungen. Auch bei *ReadBeyond* kommt dieser Algorithmus zum Einsatz, um die Anzahl an Bewertungen zu gewichten.

Nun muss eine Lösung für das Problem der Aktualität der Bewertungen gefunden werden.

Zuerst werden alle Aufrufe von einer Empfehlung, die von Benutzern angeklickt wurden, aus der Datenbank selektiert. Dabei hat ein Datensatz `calls` folgende, benötigte Felder:

- `calls_day` (x): zeigt an, wie oft eine bestimmte Empfehlung an einem Tag aufgerufen wurde.
- `day`: beschreibt den Zeitraum (das Datum), in dem `calls_day` gemacht worden sind.

Ein Korrekturfaktor h wird benötigt, der für die Anzahl an Aufrufen und das Altern der Bewertungen bedeutend ist. In dem Moment, in dem eine neue Empfehlung in der Datenbank abgelegt und zum ersten Mal evaluiert wird, gibt es noch keine älteren Aufrufe. Insofern muss das Altern der Empfehlung nicht miteinbezogen werden und der Korrekturfaktor h, der dafür maßgebend

[7] http://www.yahoo.com/

ist, ist 1 und verändert später den Relevanzwert somit nicht. Solange es keine Evaluierungen gibt, gilt $h_0 = 1$. Am Tag nachdem die Empfehlung zum ersten Mal aufgerufen wurde, werden die Aufrufe x diesen Tages durch die durchschnittliche Anzahl der Aufrufe y an einem Tag dividert. y ist die Summe aller Aufrufe x_i, dividert durch die Zeitperiode aller Aufrufe n,:

$$y = \frac{\sum_{i=1}^{n} x_i}{n}, \tag{4.4}$$

$$h_1 = \frac{x}{y}, . \tag{4.5}$$

Wurde h ein Mal definiert, wird die Gleichung verändert und es gilt für alle folgenden Male:

$$h_{n+1} = \frac{h_n \cdot t + y}{2}. \tag{4.6}$$

t ist eine Konstante mit dem Wert 0,99, durch die garantiert wird, dass h von Tag zu Tag exponentiell abnimmt. Insgesamt erhält man jetzt mit h einen Wert, der etwas über die Anzahl der Aufrufe, vor allem im Vergleich zum Durchschnitt, und das Alter der Empfehlung aussagt. Ist dieser Wert kleiner als 1, hat er eine negative Auswirkung auf den Relevanzwert. Ist dieser Wert größer als 1, gibt es mehr Aufrufe als durchschnittlich – dies hat jedoch keine Auswirkungen und es gilt trotzdem $h = 1$.

Da in diesem Ansatz ein direkter Zusammenhang zwischen Relevanzwert, Aufrufen und Alter hergestellt wird, vermindert sich die Relevanz um diesen Wert h. Das Altern der Empfehlungen soll keine großen Auswirkungen haben und fließt daher nur zu einem bestimmten Prozentsatz ein. Um ihn ein wenig geringer zu gewichten, wird er nur zu 80 % miteinberechnet. Es gilt für die Konstante $s = 0,8$; folgende Gleichung entsteht:

$$h = \frac{h}{s}. \tag{4.7}$$

Ist h dadurch wieder größer als 1, verändert sich dadurch nichts und bleibt 1. Da der Relevanzwert der API r um h verringert wird, wenn er kleiner als 1 ist, muss r zuvor von 1 subtrahiert werden. Zusätzlich wird die Bewertung der Benutzer miteingerechnet. Um dies auch auf den Relevanzwert einfließen lassen zu können, wird der gewichtete Mittelwert durch fünf dividiert und man erhält die gewichtete Bewertung eines Sterns. Da auch die Bewertungen der Benutzer nicht voll gewichtet werden sollen, wird der Wert auch hier durch 20 % entkräftet:

$$g_s = \frac{\frac{g}{5}}{s} \tag{4.8}$$

$$r_c = r * (1 - h) * g_s \tag{4.9}$$

Die neue Relevanz r_c wird nun in der Datebank gespeichert und bei einer neuen Evaluierung gegebenenfalls verändert. Die in diesem Kapitel entwickelten Formeln wurden in *ReadBeyond* als Algorithmus umgesetzt und anschließend evaluiert. Die Werte der beiden Konstanten s und t wurden nach der Evaluierung (siehe Abschnitt 5.1) als optimal empfunden und können gegebenenfalls verändert werden.

Kapitel 5

Evaluierung

Bevor der Prototyp als Endversion *ReadBeyond* verbreitet wird, ist noch eine längere Testphase von mehreren Wochen nötig. Diese sollte mit möglichen Endbenutzern durchgeführt werden, die zu möglichst unterschiedlichen Themengebieten recherchieren. Wünschenswert sind außerdem Benutzer mit verschiedenen Vorkenntnissen: Für Personen, die weniger internetaffin sind und beispielsweise nur schwierig brauchbare Suchanfragen formulieren können, kann *ReadBeyond* eine Unterstützung bei Recherchen sein. Umgekehrt kann das System aber auch für Benutzer wie *Digital Natives*, also Personen, die bereits mit Technologien wie dem Internet aufgewachsen sind, eine Erleichterung der Arbeit darstellen. Um sowohl den Algorithmus als auch die Handhabung mit dem System und dessen Design zu testen, soll also eine möglichst heterogene Testgruppe gefunden werden.

5.1 Evaluierung des Algorithmus

Um den Algorithmus aus Abschnitt 4.2.2 zu evaluieren, wurde ein Testprogramm entwickelt, mit dem vor allem eventuelle Schwächen oder Probleme aufgedeckt werden sollen. Anschließend werden die Ergebnisse präsentiert und der Algorithmus dementsprechend verbessert. Erst dann soll eine längere Testphase mit Endnutzern durchgeführt werden. Für den „Pretest“ wurden vier Testfälle definiert, die unterschiedliche Gegebenheiten abdecken sollen. Grundsätzlich sollen alle Faktoren, die in den Algorithmus einfließen, getestet werden:

- Anzahl der Bewertungen: Gibt es viele oder wenige Bewertungen? Gibt es beispielsweise mehr als 60 Bewertungen, sodass die Obergrenze beim gewichteten Mittelwert (siehe Abschnitt 4.2.2) überschritten wird?
- Bewertungslevel: Haben die Benutzer die Empfehlung hoch oder niedrig bewertet? Wie wirkt es sich aus, wenn mit dem Alter der Empfehlungen die Bewertungen steigen oder sinken – heben sich die Berechnungen eher auf, oder verändert sich der Relevanzwert trotzdem

stärker?

- Alter der Empfehlungen: Ist die Empfehlung neu im System, oder bereits seit Wochen gespeichert?
- Auffälligkeiten/Ausreißer: Gibt es Ausreißer? Wie verhält sich beispielsweise der Algorithmus, wenn eine Empfehlung nach ein paar Aufrufen eine lange Zeitspanne nicht mehr gewählt und evaluiert wird und anschließend wieder viele Evaluierungen bekommt? Wie verhält sich der Algorithmus, wenn es zwar viele Aufrufe, aber nur wenige getätigte Bewertungen gibt? Was passiert, wenn sich nur das Alter ändert, weil von Tag zu Tag genau die gleiche Anzahl an Bewertungen und Bewertungsgrößen abgegeben wird?

Vier Testfälle wurden entwickelt. Alle genauen Daten der Testfälle und die zugehörigen Ergebnisse können in Anhang A eingesehen werden. Der Algorithmus kann mit neuen Tests ständig verbessert werden.

5.1.1 Testfall 1

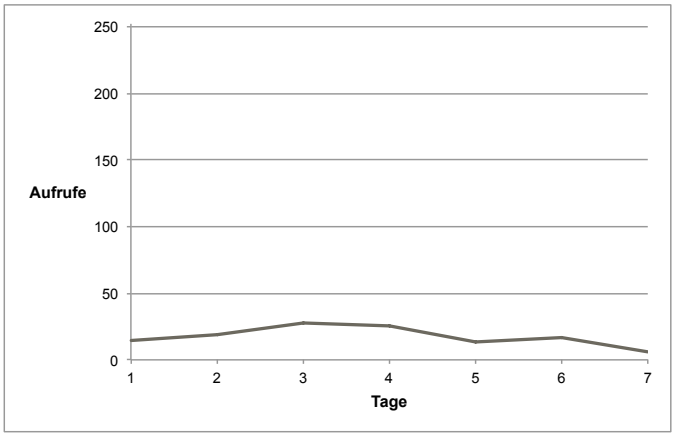
Im ersten Testfall wird die Evaluierung über einen relativ kurzen Zeitraum von sieben Tagen beobachtet und ist daher sehr jung. Die Bewertungsanzahl ist durchschnittlich – sie liegt an sechs der sieben Tage in dem Bereich zwischen 10 und 60, in denen der arithmetische Mittelwert stärker verändert wird (siehe Abbildung 5.1 (a)). Bei allen ausgewählten Empfehlungen wurde evaluiert, das heißt je Aufruf gibt es auch eine Bewertung. Diese wiederum sind eher hoch.

Da die Bewertungen der Benutzer nach den ersten beiden Tagen mittelmäßig bis gut sind, flacht der Relevanzwert der API von 0,7 nur ein wenig auf 0,67 ab (siehe Abbildung 5.1 (b)). Ab dem dritten Tag gibt es sehr viele 5-Sterne-Bewertungen, wodurch der Relevanzwert einen Sprung nach oben nimmt und bis zu 0,74 steigt. Da am letzten Tag die Zahl der Aufrufe im Vergleich sehr niedrig ist und auch die Benutzerbewertung nur durchschnittlich ist, fällt die Relevanz wieder auf 0,57 ab.

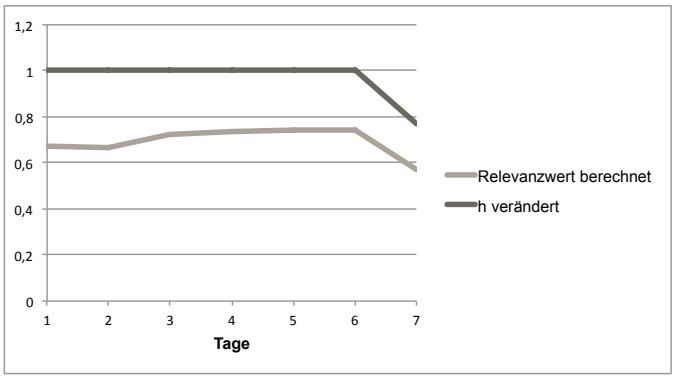
5.1.2 Testfall 2

Hier wird die Evaluierung für 22 Tage getestet und hat somit ein mittleres Alter. Allgemein wurde die Empfehlung nur sehr wenige Male aufgerufen und mittel bis gut bewertet (siehe Abbildung 5.2 (a)). Allerdings ist die Verteilung der Aufrufe hier auffällig; nur an den ersten beiden und den letzten drei Tagen gibt es Aufrufe und Bewertungen. An den 17 Tagen dazwischen wurde getestet, wie sich eine lange Phase ohne Benutzerinteraktion auf den Relevanzwert auswirkt.

Da es ab dem dritten Tag keine Aufrufe mehr gibt, sieht man deutlich, dass der Faktor des Alterns zu stark in den Relevanzwert miteinfließt – vor allem da die Anzahl der Aufrufe an den ersten zwei Tagen auch sehr niedrig



(a)



(b)

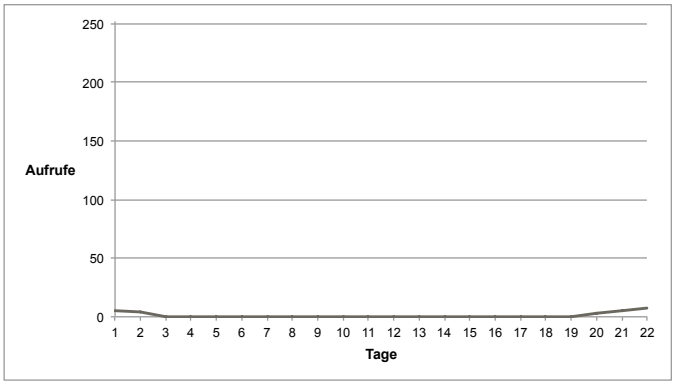
Abbildung 5.1: Auszug aus den Ein- (a) und Ausgabedaten (b) des Testfalls 1.

war (siehe Abbildung 5.2 (b)). An Tag 14 fällt etwas auf: Der Relevanzwert würde hier anscheinend ins Negative fallen und springt unerwartet auf 8,54 an. Ursprung für diesen Wert ist der Korrekturfaktor h, der auf 9,522 steigt. Hier liegt eindeutig ein Fehler in der Berechnung vor und der Algorithmus muss überarbeitet werden. Nach ein paar Tagen pendeln sich die Werte wieder ein.

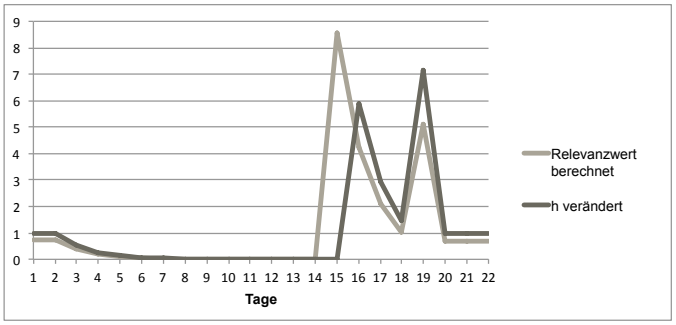
5.1.3 Testfall 3

Um eine lange Zeitdauer zu evaluieren, wird für eine Zeitspanne von 100 Tagen täglich eine Zufallszahl an Bewertungen mit geniertem Wert zwischen 0 und 65 erstellt (siehe Abbildung 5.3 (a)).

Durch die Erzeugung von zufälligen Werten liegt die Bewertung der Benutzer sowohl ungewichtet als auch gewichtet immer im Mittelmaß zwischen



(a)



(b)

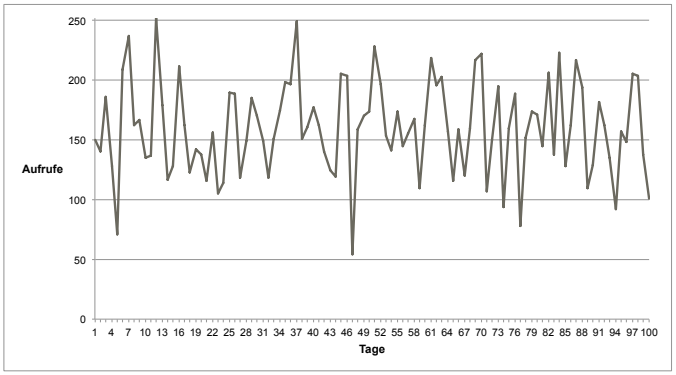
Abbildung 5.2: Auszug aus den Ein- (a) und Ausgabedaten (b) des Testfalls 2.

3 und 4. Bis auf Tag 47 gibt es immer mehr als 60 Bewertungen pro Tag, also mehr als die Obergrenze bei der Berechung des gewichteten Algorithmus. Ab dieser Stufe wird der arithmetische Mittelwert nur mit einem Faktor von 0,1 verändert (siehe Gleichung 4.3 in Abschnitt 4.2.2). Dies ist auch in der Auswertung ersichtlich: Der gewichtete Mittelwert ist stets um 0,1 größer als der arithmetische.

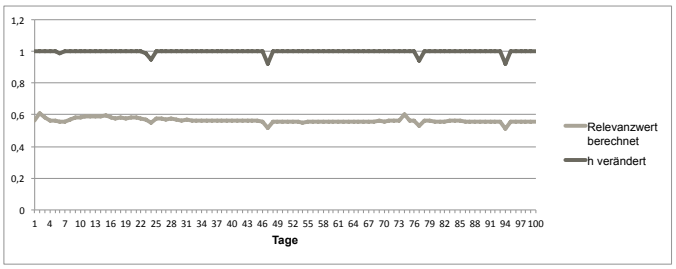
Da die Mittelwerte und h stets ungefähr gleich bleiben, kann beobachtet werden, dass sich die Relevanz nicht sonderlich stark verändert. Da es täglich sehr viele Benutzerbewertungen gibt, fällt das Alter der Empfehlung nicht sonderlich ins Gewicht. Verändert sich also h stärker, hängt dies mit der Anzahl der Aufrufe zusammen (siehe Abbildung 5.3 (b)).

5.1.4 Testfall 4

Ähnlich wie bei Fall 3 werden auch hier zufällige Werte generiert (siehe Abbildung 5.4 (a)). Hingegen wird ein besonders kurzes Zeitintervall von nur zwei Tagen evaluiert.



(a)



(b)

Abbildung 5.3: Auszug aus den Ein- (a) und Ausgabedaten (b) des Testfalls 3.

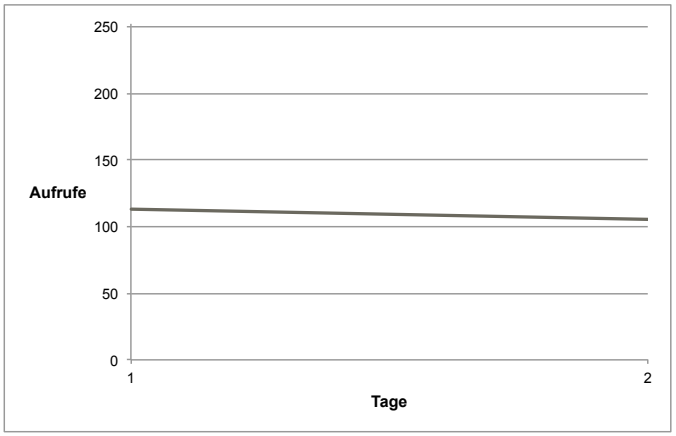
Am ersten Tag sind die Bewertungen mit einem gewichteten Mittelwert von 1,88 sehr niedrig. Dies wirkt sich sehr stark auf den Relevanzwert aus, indem dieser von den anfänglichen 0,7 auf 0,33 abnimmt (siehe Abbildung 5.4 (b)). Die Anzahl an Bewertungen ist am ersten Tag mit 113 schon sehr hoch, weswegen sich auch das Bewertungsniveau so deutlich auswirkt. Durch die vielen Aufrufe und die minimal besseren Bewertungen am zweiten Tag kann sich der Relevanzwert nochmals verbessern. Die Ergebnisse verhalten sich wie erwartet.

5.2 Evaluierung der Anforderungen

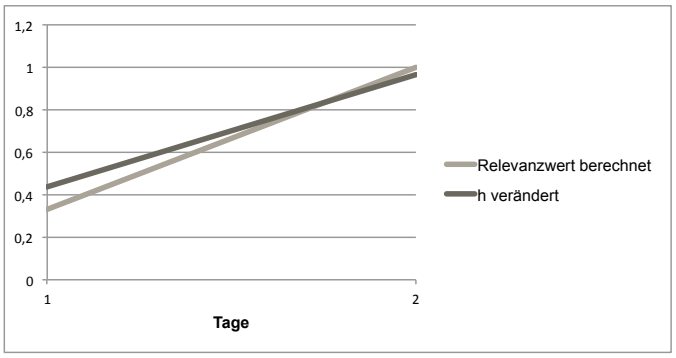
Eine weitere Möglichkeit *ReadBeyond* zu evaluieren, ist das Endergebnis mit den definierten Anforderungen aus Abschnitt 4.1.1 zu vergleichen.

5.2.1 Browserübergreifende Anwendungsmöglichkeiten

Ein Hauptgrund für die Implementierung eines Benutzerskripts anstelle eines Browserplugins war die Möglichkeit, so browserübergreifend eingesetzt werden zu können. Tatsächlich lässt sich das Skript ohne Änderungen, teilweise



(a)



(b)

Abbildung 5.4: Auszug aus den Ein- (a) und Ausgabedaten (b) des Testfalls 4.

nur mit minimalem Mehraufwand in unterschiedlichen Browsern installieren. In den Browsern Chrome[1] und Opera[2] ist für die Installation ein zusätzliche Erweiterung nicht nötig. Die Installation der Erweiterung GreaseKit war in Safari[3] aufgrund eines Fehlers nicht möglich. Stattdessen wurde auf Ninja-Kit zurückgegriffen, mit dem die Installation des Benutzerskriptes einwandfrei funktionierte. Mit dem wohl bekanntesten AddOn für Benutzerskripte ließ sich das Benutzerskript mit Greasemonkey für Firefox[4] problemlos installieren. Geht man davon aus, dass das Benutzerskript in diesen Browsern (und vermutlich auch mit anderen Versionen) läuft, erreicht man laut der Studie von ÖWA [55] potenziell 60,4 % der österreichischen Benutzer (siehe Abschnitt 4.1.2). Die Anforderung wurde somit erfüllt.

[1] getestet mit Version 18
[2] getestet mit Version 11.61
[3] getestet mit Version 6.0.5
[4] getestet mit Version 23.0.1

5.2.2 Dezenz

Das Interface von *ReadBeyond* ist an eine höhere Auflösung des Monitors von 1680×1050 Bildpunkten angepasst. Ist diese geringer, beispielsweise 800×600, ist das Interface sehr auffällig und unter Umständen auch störend. Dies muss durch Benutzer mit unterschiedlichen Endgeräten getestet und gegebenenfalls angepasst werden. Ob die Benutzeroberfläche auch bei einer hohen Auflösung dezent und unaufdringlich ist, muss ebenfalls in einer Benutzerstudie getestet werden.

5.2.3 Relevanz/Genauigkeit

Durch den hybriden Ansatz soll mit der kollaborativen Bewertungsmöglichkeit die Grundlage für einen exakteren Relevanzwert geschaffen werden. Dem Benutzer wird so vermittelt, dass er sich nicht nur auf externe Anbieter verlassen muss, sondern auch selbst etwas aktiv beisteuern und Kritik üben kann. Nur höchstens fünf Empfehlungen werden dem Benutzer vorgeschlagen; schafft der Relevanzwert eine bestimmte Grenze nicht, wird er zusätzlich ausgefiltert. Schaffen keine fünf Empfehlungen diese Hürde, werden weniger angezeigt. Dies zeigt dem Benutzer, dass nicht mutwillig irrelevante Artikel gezeigt werden, damit die Benutzeroberfläche befüllt ist. Ob die Berechnungen zur Einbindung der Benutzerbewertungen in den Relevanzwert nachvollziehbar und logisch sind, wird in dem Testprogramm in Abschnitt 5.1 geprüft. Zusätzlich sollte *ReadBeyond* trotzdem über einen längeren Zeitraum von mehreren Wochen getestet werden, um möglichst viele Auswirkungen auf den Relevanzwert zu testen.

5.2.4 Qualität

Eine hohe Qualität der Einträge ist eine Anforderung, die leider nicht erfüllt werden konnte. Zemanta verwendet grundsätzlich unterschiedliche Quellen für seine Empfehlungen. Hauptsächlich Artikel von geschätzten Blogs und Nachrichten-Seiten werden inkludiert. Im Falle der Markups werden verschiedenste Datenbanken wie Wikipedia, IMDB[5], MusicBrainz[6] (Musik Enzyklopädie), Amazon usw. als Quellen eingesetzt. *ReadBeyond* soll gerade bei Recherchen auch für wissenschaftliche Arbeiten eine Hilfe sein. Für diesen Zweck sind die angebotenen Artikel unpassend oder oft nicht ausreichend. Den wissenschaftlichen Anspruch kann Wikipedia zwar grundsätzlich erfüllen, doch eignet sich die Webseite nicht als wissenschaftliche Quelle. Ob die Artikel dort von Interessierten oder Experten geschrieben sind, ist oft nicht nachvollziehbar und eine Zusammenfassung anderer Quellen; Sekundärzitate sollen jedoch vermieden werden. Das System würde definitiv von einer weiteren Quelle, wie Datenbanken mit wissenschaftlichen Arbeiten, profitieren.

[5] http://www.imdb.com/
[6] http://musicbrainz.org/

Da im Prototyp nur eine Evaluierungsmöglichkeit umgesetzt wurde, kann infolgedessen die Qualität der Empfehlungen nicht separat bewertet werden.

5.2.5 Konfiguration

Für den Benutzer stellt nur die Installation einen Aufwand dar; Konfigurationen für die Nutzung von *ReadBeyond* sind nicht notwendig. Mithilfe einer prägnanten schriftlichen Installationsanweisung für die gängigsten Browser sollte die Installation auch ungeübten, nicht computeraffinen Benutzern keine Probleme bereiten.

5.2.6 Deaktivierung

Die letzte Anforderung an *ReadBeyond* ist die schnelle Deaktivierungsmöglichkeit. Dies ist vor allem in Firefox durch Greasemonkey gewährleistet, da sich die Erweiterung inklusive aller Benutzerskripte mit einem Klick deaktivieren lässt. Auch in den Browsern Chrome und Safari ist dies mit wenigen Schritten durchführbar. Ohne zusätzliche Erweiterung ist dies in Chrome etwas umständlicher – das Benutzerskript muss gelöscht und später neu hinzugefügt werden, insofern nicht Javascript als Ganzes oder alle Benutzerskripte deaktiviert werden sollen. Alles in allem ist eine Deaktivierung in den getesteten Browsern umzusetzen. Somit kann das Skript ressourcenschonend eingesetzt werden.

5.3 Akzeptanz durch den Benutzer

Wie bereits in Abschnitt 2.3.2 erwähnt, stellt zum Beispiel die Akzeptanz von kollaborativen Systemen eine Herausforderung dar. Je mehr Informationen über den Benutzer gesammelt werden können, umso exaktere Empfehlungen können getroffen werden [39]. Um die Akzeptanz des Benutzers gegenüber dem System zu sichern, ist es ratsam, sich bei dem Design und der Implementierung eines Empfehlungssystems an diverse Richtlinien zu halten.

5.3.1 Benutzerakzeptanz-Anforderungen

Paramythis et al. [29] entwickelten ein Framework, mit dessen Hilfe adaptive Systeme evaluiert werden können. Für fünf bestimmte Ebenen im Adaptionsprozess wurden Kriterien zur Evaluierung entwickelt. Die einzelnen Ebenen werden folgendermaßen gegliedert:

1. Sammeln von Eingabedaten,
2. Interpretation der gesammelten Daten,
3. Modellierung des aktuellen Zustands (neues Wissen über den Benutzer oder den Kontext der Interaktion ableiten),

4. Entscheidung über die Notwendigkeit und Art der Adaption in Berücksichtigung des Modells des aktuellen Zustands,
5. Anwendung oder Instanziierung der Adaption.

Die erstellten Kriterien sind bei der Evaluierung eines adaptiven Systems nützlich und in die eben genannten Ebenen kategorisierbar [29, S. 38] (die deutsche Übersetzung wurde von Neumayr [28, S. 14-17] übernommen):

- *Transparenz/Verständlichkeit (transparency, comprehensability):* Wissen und verstehen die Benutzer, was das System aufzeichnet (Ebene 1), interpretiert (Ebene 2), modelliert (Ebene 3) und warum? Welche Adaptionen hat das System durchgeführt und warum (Ebene 4)? Wie fand die Adaption statt (Ebene 5)?
- *Vorhersagbarkeit (predictability):* Können die Benutzer vorhersagen, welchen Effekt ihre Aktionen auf die Vorstellungen (Ebenen 2 und 3) und Entscheidungen (Ebene 4) des Systems hat? Wird die User-Experience nicht zu inkonsistent durch die Adaption (Ebene 5)? Werden die Benutzer gebeten grundlegende Änderungen am Systemverhalten oder Systemaussehen zu bestätigen (Ebenen 4 und 5)? Hält sich das System an Konventionen jener Applikationen, die seine Benutzer normalerweise verwenden (Ebene 5)? Werden die Adaptionen dermaßen durchgeführt, wie dies im Einklang mit den Benutzer-Erwartungen aus der realen Welt steht? (Ebene 5)?
- *Privatsphäre (privacy):* Werden die Benutzer über die Daten informiert, die aufgezeichnet werden (Ebene 1), welche Rückschlüsse daraus gezogen werden (Ebenen 2 und 3) und wie diese Daten gespeichert und genützt werden (Ebenen 2, 3 und 4)? Können die Benutzer mitentscheiden, welche Daten über sie gespeichert werden (Ebene 1), welche Rückschlüsse gezogen werden dürfen (Ebenen 2 und 3), welche Adaptionen gezeigt werden (Ebene 5), welche Daten gespeichert (Ebenen 2 und 3) und wofür sie verwendet werden (Ebene 4)? Werden persönliche Daten vergleichbar wie im realen Leben geschützt (Ebenen 2 und 3)?
- *Steuerbarkeit/Durchschaubarkeit (controllability, scrutability):* Können die Benutzer Interpretationen des Systems (Ebene 2), Aktionen am Benutzermodell (Ebene 3) oder Adaptionsentscheidungen (Ebene 4) ruückgängig machen oder verändern? Können Benutzer beeinflussen, wie Adaptionen angewandt (Ebene 5) und wie Rückschlüsse (Ebene 3) und Entscheidungen (Ebene 4) getroffen werden (zum Beispiel durch das Setzen von Parametern, die das Systemverhalten steuern)?
- *Umfang der Erfahrung, Zufallsfund (breadth of experience, serendipity):* Können jene Benutzer noch immer auf Material zugreifen, von dem das System denkt, dass es weniger angemessen für diese Benutzer ist (Ebene 5)? Wird es den Benutzern ermöglicht, unerwartete, erfreuliche Entdeckungen zu machen, anstatt die Erfahrung einzuschränken

(Ebenen 4 und 5)?

- *Unaufdringlichkeit (unobtrusiveness):* Sind Erklärungen des Systemverhaltens nicht unnötig oder zu oft störend für die Benutzer (Ebene 5)? Wird zu häufig eine Bestätigung der Benutzer des Systemverhaltens eingeholt, obwohl die gar nicht nötig wäre (Ebenen 2, 3 und 4)?
- *Zeitgerechtheit (timeliness):* Ist das Timing des Systemverhaltens (zum Beispiel Meldungen) angemessen an die Benutzeraktivitäten und den Kontext angepasst (Ebenen 4 und 5)?
- *Ästhetik (aesthetics):* Sind automatische Änderungen des Systemaussehens ästhetisch ansprechend (Ebene 5)?
- *Angemessenheit/Notwendigkeit (appropriateness, necessity):* Wie notwendig war die Aktion, über die das System soeben entschieden hat (Ebene 4)? Wie angemessen war die Aktion, über die das System entschieden hat, wenn man den derzeitigen Zustand der Interaktion, den Verlauf der Interaktionen und die adaptive Theorie des Systems beachtet (Ebene 4)?

Bisherige Forschungsarbeiten, die sich mit dem konkreten Design von bzw. Problemen bei der Erstellung von Empfehlungssystemen beschäftigt, wurden von Pu et al. [35] in ihrer Publikation vereint. Dabei wurden zwei wichtige Fragen untersucht: Wie interagieren Benutzer mit einem Empfehlungssystem und wie motivierend sind die wahrgenommenen Qualitäten bzw. Eigenschaften des Empfehlungssystems für die Benutzer, um das System anzunehmen? Daraus entstanden 20 Richtlinien bezüglich Benutzerfreundlichkeit und -oberflächendesign, die die Entwicklung von effektiveren, zufriedenstellenden Empfehlungssystemen unterstützen soll. Die Richtlinien, die je nach Art des Recommenders nicht immer alle einsetzbar sind, wurden dabei in Usability-Probleme eingeteilt. Nachstehend folgt ein Ausschnitt aus den Richtlinien [35]:

- Um das erste Problem, den Aufwand des Benutzers, zu vermeiden, empfehlen die Autoren diesen möglichst gering zu halten und/oder den Prozess der Datenerhebung für den Benutzer adaptiv und unterhaltsam zu gestalten.
- Der Benutzer soll weiters immer selbst Kontrolle darüber haben, was er bewertet.
- Bei Systemen, die auf der Persönlichkeit des Benutzers basieren, können unterhaltsame Persönlichkeitstests eingesetzt werden, um ihm Präferenzen zu entlocken.
- Damit die optimale Erhebungsmethode gefunden werden kann, sodass der Ausgleich zwischen Aufwand und Genauigkeit geschaffen werden kann, sollen mehrere vergleichende Benutzerstudien stattfinden.
- Den Benutzern soll bewusst sein, welchen einzigartigen Beitrag sie leisten.

- Bestimmte Ziele, die der Benutzer mit seiner Beteiligung erreichen kann, motivieren (z. B. Identifzierungsmöglichkeiten innerhalb der Community durch einen bestimmten Mitgliederrollen oder -status wie „Insider“, „Forscher“ oder „Novize“).
- Sogenannte *explanation interfaces*, die dem Benutzer erklären, welche Vorteile bereitgestellte Daten haben, erhöhen die Transparenz des Systems und reduzieren somit Bedenken hinsichtlich der Privatsphäre.
- Um seine Vorlieben zu ändern oder zu verfeinern, sollte dem Benutzer eine Möglichkeit für Kritik an den bisherigen Daten gegeben werden.
- Dem Benutzer sollte sowohl die Möglichkeit gegeben werden, vom System vorgeschlagene als auch selbst-initiierte Bewertungen vorzunehmen.
- Um das Vertrauen des Benutzers zu gewinnen, sollen auch bereits bekannte Objekte empfohlen werden. Die Balance zwischen Vertrautheit und Neuheit von Objekten soll stets gehalten werden.
- Die empfohlenen Objekte sollen möglichst vielfältig sein. Das System soll brauchbar sein, indem nur kontext-kompatible Objekte empfohlen werden (also beispielsweise nur Bücher, wenn sich der Benuzer für Bücher interessiert).
- Die zu bewertenden Objekte sollen stets genügend Informationen vorweisen.

5.3.2 Evaluierung der Akzeptanz-Anforderungen

Die im vorigen Abschnitt erwähnten Anforderungen, um die Akzeptanz des Benutzers gegenüber dem System zu sichern, sind großteils auch auf *ReadBeyond* übertragbar. Einige wurden bereits in Abschnitt 5.2 im Bezug auf die Anforderungen ausgeführt. Zumal andere nur bei anderen Arten von Empfehlungssystemen Sinn machen, werden einzelne im folgenden Abschnitt herausgegriffen und evaluiert.

Zeitgerechtheit und Rückmeldungen

Diese Anforderung kann mitunter nicht erfüllt werden. *ReadBeyond* ist nicht nur von der Logik auf dem Webserver abhängig, sondern auch von den verschiedenen APIs. Im Gegensatz zum Prototypen werden in der späteren Version nicht nur AlchemyAPI zur Verschlagwortung genutzt, sondern auch OpenCalais eingesetzt. Dieser Schritt soll einem denkbaren Ausfall der API entgegenwirken und es möglich machen, im Fall der Fälle auf die andere API zurückzugreifen. Fällt eine API aus, kann es unter Umständen zu einer Zeitverzögerung kommen, wenn versucht wird, eine Verbindung zu ihr aufzubauen. Hier muss gegebenenfalls ein Zeitlimit gesetzt werden. Ist *ReadBeyond* weit verbreitet, wird natürlich auch die Datenmenge am Webserver

sehr groß. Um hier bei Vergleichen nicht Zeitverzögerungen zu verursachen, sollte in Betracht gezogen werden, Datenbankindizes einzusetzen.

Ein weiteres Problem ist die Begrenzung der API-Aufrufe: Vorausgesetzt das Benutzerkript ist aktiviert, werden beim Laden von jeder einzelnen Seite Aufrufe zu den APIs gemacht. Diese sind bei Zemanta mit 1.000 kostenfreien, bzw. bei Nachfrage 10.000 begrenzten Aufrufen am geringsten. Da momentan Zemanta die einzige Quelle für Artikel ist, könnten also nach dem Überschreiten des Limits nur für Seiten, die bereits im System sind, Empfehlungen gegeben werden. Angesichts der Menge an Webseiten im Netz, ist es vor allem zu Beginn sehr unwahrscheinlich, dass bereits Daten vorhanden sind. Da bei einer gründlichen Recherche mit nur wenigen Benutzern das Limit schnell erreicht wird, müsste hier, z. B. kostenpflichtig, aufgestockt werden. Wichtig ist es in allen Fällen den Benutzern darüber zu informieren, wenn Verzögerungen eintreten oder keine Empfehlungen gefunden werden. Dies kann mitunter grafisch durch Lade-Bilder passieren, oder durch formulierte Rückmeldungen geschehen. Momentan ist dies bei *ReadBeyond* noch nicht umgesetzt.

Bevor das Empfehlungssystem allerdings die längere Testphase durchläuft, muss dies noch ergänzt werden. Damit sich der Benutzer im System zurecht findet und informiert ist, wenn Probleme oder Änderungen zum normalen Verhalten auftauchen, sind Rückmeldungen unumgänglich.

Kontrolle und Privatsphäre

Dem Benutzer wird selbst die Kontrolle darüber gelassen, ob er Empfehlungen bewertet oder nicht. Kann oder möchte er keine Bewertungen abgeben, wird er nicht dazu gezwungen.

ReadBeyond legt kein Benutzerprofil an und benötigt daher auch keine Daten der Benutzer. Es wird nur gespeichert, wann welche Seite aufgerufen wurde. Dies wird aber nicht mit einem Profil in Verbindung gebracht und die Person dahinter bleibt anonym. Da keine persönlichen Daten über den Benutzer gespeichert werden, muss er auch nicht um Erlaubnis gefragt oder speziell informiert werden.

5.4 Zusammenfassung der Ergebnisse

Im ersten Testdurchlauf des Algorithmus haben die Ergebnisse großteils den Erwartungen entsprochen. Ein Fehler ist in einem Sonderfall aufgetaucht und muss noch näher untersucht werden. Der Algorithmus bedarf noch weiterer Evaluierungen, am besten von möglichen Endbenutzern über einen längeren Zeitraum. Grundsätzlich eignet sich der Algorithmus gut, um *ReadBeyond* als Merkmal-Anreicherungs Hybrid umzusetzen. Die Schwächen der einzelnen Komponenten werden überwunden und zusätzliche wichtige Faktoren wie das Altern miteinberechnet.

Die an das System gestellten Anforderungen konnten mit dem Prototyp großteils erfüllt werden. Eine große Schwierigkeit stellt die Qualität der empfohlenen Webseiten dar. Um dem Benutzer einen tatsächlichen Mehrwert zu bieten, müssen mehr hochwertige Artikel oder Arbeiten empfohlen werden; eine Anknüpfung an wissenschaftliche Datenbanken wäre hier sinnvoll.

Kapitel 6

Schussbemerkungen

In dieser Arbeit wurde versucht eine Verbindung zwischen Empfehlungssystemen und Semantic APIs herzustellen. Um die Nachteile der verschiedenen Ansätze zu überwinden, wurden hybride Empfehlungssysteme erläutert, die die Vorteile von mehreren verknüpfen. Als eine der besten Kombinationen ging der Merkmals-Anreicherungs Hybrid hervor. Dieser wurde an der Beispielsanwendung *ReadBeyond* umgesetzt. Die Implementierung bot einige Herausforderungen wie den eigens entwickelten Algorithmus oder die vorher unbekannten Eigenheiten von Benutzerskripten. Letztlich stellte sich heraus, dass Benutzerskripte trotz ihrer relativen Unbekanntheit großes Potenzial bieten und für diesen Zweck geeignet sind. Mit *ReadBeyond* ist ein hybrides Empfehlungssystem entstanden, das den Benutzern – vorausgesetzt es wird weiterentwickelt und verbessert – einen wirklichen Mehrwert bei Recherchetätigkeiten bieten kann. Empfehlungssysteme werden nun schon längere Zeit im Netz eingesetzt. Eine Verbindung dieser Systeme mit Semantic APIs hat meiner Meinung nach durchaus Zukunftspotenzial. Das Angebot an bereits bestehenden Benutzerskripten ist mir erst im Zuge der Recherchen zu dieser Arbeit bewusst geworden. Vor allem für Personen mit Javascript-Kenntnissen bieten sie eine gute Möglichkeit selbst schnell kleine Funktionen auf Webseiten zu ergänzen.

Empfehlenswert ist die Durchführung einer längeren Studie von mehreren Wochen mit potenziellen Endnutzern. Das Ziel, *ReadBeyond* unter einer großen Menge an Benutzern zu verbreiten, bietet natürlich auch zusätzliche Herausforderungen. Zum einen müssen gegebenenfalls die täglich begrenzten Aufrufe an die Semantic APIs aufgestockt werden, da das Limit schnell erreicht ist. Zum anderen muss natürlich auch die Implementierung von *ReadBeyond* effizienter gestaltet werden. Um mit der künftig großen Datenmenge Verzögerungen zu verhindern, wäre auch der Einsatz von Datenbankindizes sinnvoll. Die Kommunikation mit dem Benutzer ist ebenfalls verbesserungswürdig. Momentan erhält dieser kaum Rückmeldungen, wenn Probleme oder Verzögerungen auftauchen. Die Auswahl der Semantic APIs und der Kon-

stanten bei Algorithmus wurden in dieser Arbeit an die Ergebnisse anderer Arbeiten angelehnt. Wünschenswert wäre hier eine Studie, welche APIs und Werte sich speziell für *ReadBeyond* eignen.

Anhang A

Testfälle

A.1 Testfall 1

Nachstehend werden die Daten aufgelistet, mit denen der Algorithmus aus Abschnitt 4.2.2 befüllt wird. Neben den Aufrufen an dem jeweiligen Tag ist auch eine Verteilung der Aufrufe auf die Bewertungsskala ersichtlich: (1,0,0,0,4) bedeutet z. B. dass ein Benutzer die Empfehlung mit einem Stern und vier Benutzer die Empfehlung mit fünf Sternen bewertet haben.

	Aufrufe an Tag x	*Verteilung der Aufrufe*
Tag 1	14	(1,0,3,5,5)
Tag 2	19	(3,0,1,10,5)
Tag 3	28	(0,2,2,9,15)
Tag 4	25	(2,0,2,7,14)
Tag 5	13	(1,0,0,4,8)
Tag 6	17	(2,0,0,5,10)
Tag 7	6	(1,0,2,0,3)

Tabelle A.1: Testdaten des 1. Falls

In der folgenden Tabelle sind die Ergebnisse des Testprogramms ersichtlich. Aufgeführt sind die veränderte, berechnete Relevanz r_c, der Korrekturfaktor h, gegebenenfalls der verändete Korrekturfaktor h, der einfache m und der gewichtete g Mittelwert der Bewertungen. Die angegebenen Werte wurden auf drei Kommastellen gerundet.

	r_c	h	h *verändert*	m	g
Tag 1	0,672	1	-	3,929	3,842
Tag 2	0,664	1,152	-	3,182	3,795
Tag 3	0,721	1,259	-	4,049	4,119
Tag 4	0,736	1,204	-	4,105	4,205
Tag 5	0,742	0,924	1	4,141	4,241
Tag 6	0,745	0,897	1	4,155	4,255
Tag 7	0,570	0,616	0,770	4,131	4,231

Tabelle A.2: Ergebnisse des 1. Testdurchlaufs

A.2 Testfall 2

Nachstehend werden die Daten aufgelistet, mit denen der Algorithmus aus Abschnitt 4.2.2 befüllt wird. Neben den Aufrufen an dem jeweiligen Tag ist auch eine Verteilung der Aufrufe auf die Bewertungsskala ersichtlich: (1,0,0,0,4) bedeutet z. B. dass ein Benutzer die Empfehlung mit einem Stern und vier Benutzer die Empfehlung mit fünf Sternen bewertet haben.

	Aufrufe an Tag x	*Verteilung der Aufrufe*
Tag 1	5	(0,0,1,1,3)
Tag 2	4	(0,0,1,3,1)
Tag 3	0	(0,0,0,0,0)
Tag 4	0	(0,0,0,0,0)
Tag 5	0	(0,0,0,0,0)
Tag 6	0	(0,0,0,0,0)
Tag 7	0	(0,0,0,0,0)
Tag 8	0	(0,0,0,0,0)
Tag 9	0	(0,0,0,0,0)
Tag 10	0	(0,0,0,0,0)
Tag 11	0	(0,0,0,0,0)
Tag 12	0	(0,0,0,0,0)
Tag 13	0	(0,0,0,0,0)

Tabelle A.3: Testdaten des 2. Falls (1 von 2)

	Aufrufe an Tag x	*Verteilung der Aufrufe*
Tag 14	0	(0,0,0,0,0)
Tag 15	0	(0,0,0,0,0)
Tag 16	0	(0,0,0,0,0)
Tag 17	0	(0,0,0,0,0)
Tag 18	0	(0,0,0,0,0)
Tag 19	0	(0,0,0,0,0)
Tag 20	3	(1,0,0,1,1)
Tag 21	5	(0,0,2,0,3)
Tag 22	7	(0,0,2,2,3)

Tabelle A.4: Testdaten des 2. Falls (2 von 2)

In der folgenden Tabelle sind die Ergebnisse des Testprogramms ersichtlich. Aufgeführt sind die veränderte, berechnete Relevanz r_c, der Korrekturfaktor h, gegebenenfalls der verändete Korrekturfaktor h, der einfache m und der gewichtete g Mittelwert der Bewertungen. Die angegebenen Werte wurden auf drei Kommastellen gerundet.

	r_c	h	h *verändert*	m	g
Tag 1	0,7525	1	-	4,4	4,3
Tag 2	0,718	0.889	1	4,2	4,1
Tag 3	0,4	0,44	0,55	4,2	4,1
Tag 4	0,195	0,218	0,272	4,2	4,1
Tag 5	0,097	0,108	0,135	4,2	4,1
Tag 6	0,048	0,053	0,067	4,2	4,1
Tag 7	0,024	0,026	0,033	4,2	4,1
Tag 8	0,012	0,013	0,016	4,2	4,1
Tag 9	0,006	0,007	0,008	4,2	4,1
Tag 10	0,003	0,003	0,004	4,2	4,1
Tag 11	0,001	0,002	0,002	4,2	4,1
Tag 12	0,001	0,001	0,001	4,2	4,1
Tag 13	0,0004	0,0004	0,0005	4,2	4,1
Tag 14	0,0002	0,0002	0,0002	4,2	4,1

Tabelle A.5: Ergebnisse des 2. Testdurchlaufs (1 von 2)

	r_c	h	h *verändert*	m	g
Tag 15	8,54	9,522	0,0001	4,2	4,1
Tag 16	4,227	4,713	5,892	4,2	4,1
Tag 17	2,092	2,333	2,916	4,2	4,1
Tag 18	1,036	1,155	1,444	4,2	4,1
Tag 19	5,127	5,717	7,146	4,2	4,1
Tag 20	0,684	2,5	-	4	3,91
Tag 21	0,697	4,326	-	4,056	3,982
Tag 22	0,705	5,35	-	4,08	4,03

Tabelle A.6: Ergebnisse des 2. Testdurchlaufs (2 von 2)

A.3 Testfall 3

Nachstehend werden die Daten aufgelistet, mit denen der Algorithmus aus Abschnitt 4.2.2 befüllt wird. Neben den Aufrufen an dem jeweiligen Tag ist auch eine Verteilung der Aufrufe auf die Bewertungsskala ersichtlich: (1,0,0,0,4) bedeutet z. B. dass ein Benutzer die Empfehlung mit einem Stern und vier Benutzer die Empfehlung mit fünf Sternen bewertet haben.

	Aufrufe an Tag x	*Verteilung der Aufrufe*
Tag 1	150	(35,26,15,33,41)
Tag 2	140	(10,8,39,43,40)
Tag 3	186	(12,65,35,54,20)
Tag 4	128	(53,5,18,46,6)
Tag 5	71	(17,6,7,26,15)
Tag 6	209	(47,38,16,56,52)
Tag 7	237	(45,6142,60,29)
Tag 8	162	(17,4,37,56,48)
Tag 9	167	(12,3,47,48,57)
Tag 10	135	(2,35,62,20,16)
Tag 11	137	(3,2,54,20,58)
Tag 12	252	(40,37,64,48,63)
Tag 13	179	(14,27,59,57,22)

Tabelle A.7: Testdaten des 3. Falls (1 von 4)

	Aufrufe an Tag x	*Verteilung der Aufrufe*
Tag 14	117	(22,9,26,60,0)
Tag 15	128	(9,6,3,56,54)
Tag 16	212	(60,58,24,57,13)
Tag 17	162	(41,61,15,30,15)
Tag 18	123	(7,5,52,6,53)
Tag 19	142	(50,21,15,44,12)
Tag 20	138	(37,0,21,64,61)
Tag 21	116	(21,7,1,24,63)
Tag 22	156	(56,19,56,14,11)
Tag 23	105	(4,56,6,19,20)
Tag 24	114	(21,27,25,8,33)
Tag 25	190	(12,59,55,27,37)
Tag 26	189	(1,65,38,22,63)
Tag 27	118	(33,44,4,34,3)
Tag 28	149	(2,25,23,59,40)
Tag 29	185	(34,63,30,41,17)
Tag 30	170	(50,62,44,9,5)
Tag 31	149	(12,22,64,1,50)
Tag 32	118	(35,2,49,7,25)
Tag 33	151	(47,40,4,51,9)
Tag 34	175	(8,54,34,31,48)
Tag 35	198	(9,65,45,39,40)
Tag 36	197	(63,23,37,41,33)
Tag 37	249	(43,54,56,41,55)
Tag 38	151	(40,11,58,24,18)
Tag 39	161	(18,5,59,23,56)
Tag 40	177	(2,31,45,37,62)
Tag 41	162	(27,46,62,7,20)
Tag 42	140	(37,4,44,9,46)
Tag 43	125	(11,52,34,1,27)
Tag 44	119	(23,42,38,16,0)

Tabelle A.8: Testdaten des 3. Falls (2 von 4)

	Aufrufe an Tag x	*Verteilung der Aufrufe*
Tag 45	205	(57,35,5,50,58)
Tag 46	204	(62,53,23,41,25)
Tag 47	54	(19,3,6,16,10)
Tag 48	159	(26,53,14,4,62)
Tag 49	170	(60,16,48,28,18)
Tag 50	174	(10,52,60,49,3)
Tag 51	228	(60,40,38,65,25)
Tag 52	197	(31,62,13,54,37)
Tag 53	154	(38,8,40,44,24)
Tag 54	141	(51,5,11,65,9)
Tag 55	174	(8,60,26,57,23)
Tag 56	145	(44,2,10,38,51)
Tag 57	155	(13,32,26,52,32)
Tag 58	168	(52,17,28,65,6)
Tag 59	110	(0,38,14,41,17)
Tag 60	162	(38,26,22,50,26)
Tag 61	219	(32,59,20,58,50)
Tag 62	196	(44,36,53,54,9)
Tag 63	203	(39,2,42,65,55)
Tag 64	157	(9,52,7,38,51)
Tag 65	116	(13,38,24,27,14)
Tag 66	159	(41,0,40,64,50)
Tag 67	120	(0,31,44,21,24)
Tag 68	160	(29,0,61,16,54)
Tag 69	217	(4,55,57,46,55)
Tag 70	222	(46,55,41,53,27)
Tag 71	107	(27,1,0,51,28)
Tag 72	150	(14,27,28,55,26)
Tag 73	195	(13,56,57,58,11)
Tag 74	94	(15,21,11,10,37)

Tabelle A.9: Testdaten des 3. Falls (3 von 4)

	Aufrufe an Tag x	*Verteilung der Aufrufe*
Tag 75	160	(0,15,27,57,61)
Tag 76	189	(16,38,51,58,26)
Tag 77	78	(13,19,27,14,5)
Tag 78	152	(56,28,32,19,17)
Tag 79	174	(58,33,8,50,25)
Tag 80	171	(19,65,46,31,10)
Tag 81	145	(18,32,26,45,24)
Tag 82	206	(21,62,62,7,54)
Tag 83	138	(23,21,8,51,35)
Tag 84	223	(13,41,64,45,60)
Tag 85	128	(16,38,28,24,22)
Tag 86	162	(53,44,22,34,9)
Tag 87	217	(33,52,41,59,32)
Tag 88	194	(65,15,29,62,23)
Tag 89	110	(17,19,44,26,4)
Tag 90	129	(13,39,46,12,19)
Tag 91	182	(41,28,57,3,53)
Tag 92	162	(14,56,31,37,24)
Tag 93	135	(41,4,11,16,63)
Tag 94	92	(43,16,13,7,13)
Tag 95	157	(36,24,32,14,51)
Tag 96	148	(37,28,24,18,41)
Tag 97	205	(44,59,4,36,62)
Tag 98	204	(57,50,53,22,22)
Tag 99	138	(12,63,26,23,14)
Tag 100	101	(24,1,30,38,8)

Tabelle A.10: Testdaten des 3. Falls (4 von 4)

In der folgenden Tabelle sind die Ergebnisse des Testprogramms ersichtlich. Aufgeführt sind die veränderte, berechnete Relevanz r_c, der Korrekturfaktor h, gegebenenfalls der verändete Korrekturfaktor h, der einfache m und der gewichtete g Mittelwert der Bewertungen. Die angegebenen Werte wurden auf drei Kommastellen gerundet.

	r_c	h	h *verändert*	m	g
Tag 1	0,565	1	-	3,127	3,227
Tag 2	0,611	0,966	1	3,393	3,493
Tag 3	0,586	1,064	1	3,25	3,35
Tag 4	0,562	0,951	1	3,109	3,209
Tag 5	0,563	1,113	1	3,116	3,224
Tag 6	0,559	0,792	0,99	3,124	3,224
Tag 7	0,555	1,131	1	3,069	3,169
Tag 8	0,568	1,065	1	3,149	3,249
Tag 9	0,582	1,046	1	3,225	3,325
Tag 10	0,58	0,943	1	3,214	3,314
Tag 11	0,59	0,905	1	3,271	3,371
Tag 12	0,59	1,214	1	3,266	3,365
Tag 13	0,59	1,141	1	3,265	3,365
Tag 14	0,59	0,926	1	3,254	3,354
Tag 15	0,595	0,859	1	3,299	3,399
Tag 16	0,584	1,075	1	3,238	3,338
Tag 17	0,577	1,028	1	3,194	3,294
Tag 18	0,581	0,89	1	3,218	3,318
Tag 19	0,576	0,886	1	3,191	3,291
Tag 20	0,580	1,01	1	3,215	3,315
Tag 21	0,584	0,863	1	3,237	3,337
Tag 22	0,577	0,919	1	3,2	3,3
Tag 23	0,57	0,791	0,988	3,192	3,292
Tag 24	0,547	0,76	0,95	3,188	3,288
Tag 25	0,575	0,985	1	3,183	3,283
Tag 26	0,577	1,088	1	3,195	3,295
Tag 27	0,573	0,912	1	3,173	3,273
Tag 28	0,576	0,933	1	3,192	3,292
Tag 29	0,573	1,052	1	3,172	3,272
Tag 30	0,566	1,062	1	3,135	3,235
Tag 31	0,567	1,001	1	3,142	3,242

Tabelle A.11: Ergebnisse des 3. Testdurchlaufs (1 von 4)

	r_c	h	h *verändert*	m	g
Tag 32	0,566	0,875	1	3,136	3,236
Tag 33	0,563	0,919	1	3,119	3,219
Tag 34	0,565	1,016	1	3,126	3,226
Tag 35	0,565	1,132	1	3,128	3,228
Tag 36	0,565	1,183	1	3,116	3,216
Tag 37	0,562	1,36	1	3,113	3,213
Tag 38	0,561	1,143	1	3,105	3,205
Tag 39	0,563	1,068	1	3,118	3,218
Tag 40	0,566	1,078	1	3,134	3,234
Tag 41	0,564	1,037	1	3,123	3,223
Tag 42	0,564	0,95	1	3,124	3,224
Tag 43	0,563	0,862	1	3,119	3,219
Tag 44	0,561	0,801	1	3,106	3,206
Tag 45	0,561	1,038	1	3,106	3,206
Tag 46	0,558	1,149	1	3,091	3,191
Tag 47	0,516	0,739	0,924	3,09	3,19
Tag 48	0,558	0,868	1	3,091	3,191
Tag 49	0,556	0,965	1	3,08	3,18
Tag 50	0,556	1,025	1	3,076	3,176
Tag 51	0,554	1,218	1	3,068	3,168
Tag 52	0,554	1,215	1	3,067	3,167
Tag 53	0,554	1,08	1	3,067	3,167
Tag 54	0,553	0,974	1	3,063	3,163
Tag 55	0,554	1,023	1	3,065	3,165
Tag 56	0,555	0,958	1	3,069	3,169
Tag 57	0,556	0,957	1	3,074	3,174
Tag 58	0,555	0,997	1	3,068	3,168
Tag 59	0,555	0,838	1	3,071	3,171
Tag 60	0,555	0,922	1	3,070	3,170
Tag 61	0,555	1,138	1	3,072	3,172
Tag 62	0,554	1,171	1	3,066	3,166

Tabelle A.12: Ergebnisse des 3. Testdurchlaufs (2 von 4)

	r_c	h	h *verändert*	m	g
Tag 63	0,555	1,206	1	3,074	3,174
Tag 64	0,556	1,082	1	3,079	3,179
Tag 65	0,556	0,896	1	3,078	3,178
Tag 66	0,558	1,046	1	3,084	3,184
Tag 67	0,558	0,891	1	3,086	3,186
Tag 68	0,559	0,938	1	3,091	3,191
Tag 69	0,56	1,135	1	3,198	3,298
Tag 70	0,559	1,244	1	3,092	3,192
Tag 71	0,56	0,946	1	3,096	3,196
Tag 72	0,56	0,932	1	3,099	3,199
Tag 73	0,56	1,062	1	3,097	3,197
Tag 74	0,6	0,817	1	3,099	3,199
Tag 75	0,562	0,9	1	3,112	3,212
Tag 76	0,562	1,031	1	3,113	3,213
Tag 77	0,529	0,753	0,941	3,111	3,211
Tag 78	0,560	0,847	1	3,102	3,202
Tag 79	0,56	0,961	1	3,097	3,197
Tag 80	0,559	1,008	1	3,092	3,192
Tag 81	0,559	0,95	1	3,093	3,193
Tag 82	0,559	1,11	1	3,092	3,192
Tag 83	0,56	0,978	1	3,095	3,195
Tag 84	0,56	1,175	1	3,101	3,201
Tag 85	0,56	0,979	1	3,1	3,2
Tag 86	0,559	0,987	1	3,092	3,192
Tag 87	0,558	1,16	1	3,091	3,191
Tag 88	0,558	1,172	1	3,087	3,187
Tag 89	0,557	0,920	1	3,085	3,185
Tag 90	0,557	0,856	1	3,083	3,183
Tag 91	0,557	0,987	1	3,082	3,182
Tag 92	0,557	0,99	1	3,081	3,181
Tag 93	0,557	0,909	1	3,084	3,184

Tabelle A.13: Ergebnisse des 3. Testdurchlaufs (3 von 4)

	r_c	h	*h verändert*	m	g
Tag 94	0,512	0,737	0,921	3,079	3,179
Tag 95	0,556	0,854	1	3,079	3,179
Tag 96	0,556	0,884	1	3,079	3,179
Tag 97	0,556	1,076	1	3,078	3,178
Tag 98	0,555	1,165	1	3,071	3,171
Tag 99	0,554	1,006	1	3,068	3,168
Tag 100	0,554	0,813	1	3,068	3,168

Tabelle A.14: Ergebnisse des 3. Testdurchlaufs (4 von 4)

A.4 Testfall 4

Nachstehend werden die Daten aufgelistet, mit denen der Algorithmus aus Abschnitt 4.2.2 befüllt wird. Neben den Aufrufen an dem jeweiligen Tag ist auch eine Verteilung der Aufrufe auf die Bewertungsskala ersichtlich: (1,0,0,0,4) bedeutet z. B. dass ein Benutzer die Empfehlung mit einem Stern und vier Benutzer die Empfehlung mit fünf Sternen bewertet haben.

	Aufrufe an Tag x	*Verteilung der Aufrufe*
Tag 1	113	(64,22,17,8,2)
Tag 2	105	(15,30,24,5,31)

Tabelle A.15: Testdaten des 4. Falls

In der folgenden Tabelle sind die Ergebnisse des Testprogramms ersichtlich. Aufgeführt sind die veränderte, berechnete Relevanz r_c, der Korrekturfaktor h, gegebenenfalls der verändete Korrekturfaktor h, der einfache m und der gewichtete g Mittelwert der Bewertungen. Die angegebenen Werte wurden auf drei Kommastellen gerundet.

	r_c	h	*h verändert*	m	g
Tag 1	0,329	1	-	1,779	1,879
Tag 2	0,437	0,963	1	2,399	2,499

Tabelle A.16: Ergebnisse des 4. Testdurchlaufs

Anhang B

Inhalt der CD-ROM

B.1 PDF-Dateien

Pfad: /

Muehlboeck_Sabine_2013.pdf Masterarbeit Ëmpfehlungssysteme im Kontext von Semantic APIs"

B.2 Bilder

Pfad: /bilder

*.pdf, *.png . . . . . Verwendete Grafiken in der Masterarbeit in Originalgröße

B.3 Evaluierung

Pfad: /evaluierung

readMe.rdf . . . . . Kurze Informationen zum Testprogramm

testAlgorithm.php . . . Testprogramm für den Algorithmus

testfaelle_ausgabedaten.xls Ausgabedaten der 4 Testfälle, die in der Arbeit beschrieben werden

B.4 Online-Quellen

Pfad: /onlinequellen

alchemyapi_author.pdf AlchemyAPI: Author Extraction

alchemyapi_concept.pdf AlchemyAPI: Concept Tagging

alchemyapi_contentScraping.pdf AlchemyAPI: Content Scraping

alchemyapi_entity.pdf . AlchemyAPI: Entity Extraction

alchemyapi_faq.pdf . .	AlchemyAPI: FAQ
alchemyapi_feed.pdf . .	AlchemyAPI: Feed Detection
alchemyapi_keyword.pdf	AlchemyAPI: Keyword Extraction
alchemyapi_language.pdf	AlchemyAPI: Language Detection
alchemyapi_microformats.pdf	AlchemyAPI: Microformats Parsing
alchemyapi_relation.pdf	AlchemyAPI: Relation Extraction
alchemyapi_sentiment.pdf	AlchemyAPI: Sentiment Analysis
alchemyapi_textCat.pdf	AlchemyAPI: Text Categorization
alchemyapi_textEctraction.pdf	AlchemyAPI: Text Extraction
browser_StatsW3C.pdf	W3C Browser Statistics
calais_commercial.pdf .	OpenCalais: FAQ - Commercial Calais
calais_faq.pdf	OpenCalais: FAQ - General Questions
calais_howto.pdf	OpenCalais: How Does Calais Work?
diCiuccio.pdf	Entity Extraction & Content API Evaluation
greasespot_sandbox.pdf	Sandbox im GreaseSpot Wiki
greasespot_manual.pdf	Greasemonkey Manual: API im GreaseSpot Wiki
greasespot_crossbrowser.pdf	Cross-browser userscripting im GreaseSpot Wiki
openculture2005.pdf . .	Representing Knowledge in the Semantic Web
oewa_Browserstatistik.pdf	ÖWA Browserstatistik
oewa_Organisation.pdf	ÖWA Organisation
stackoverflow_banks.pdf	How jQuery can be used in greasemonkey scripts.
w3c_dom.pdf	What is the Document Object Model?
w3c_owl.pdf	OWL 2: Document Overview
w3c_owl_requirements.pdf	OWL: Use Cases and Requirements
w3c_rdf.pdf	Resource Description Framework (RDF): Concepts and Abstract Syntax
w3c_rdf_primer.pdf . .	RDF Primer
w3c_semanticWeb.pdf	W3C Semantic Web Activity
why_userscripts.pdf . .	Why You Should Use Userscripts And Not Extensions When Possible
zemanta_api_companion.pdf	Zemanta Service: API Companion

B.5 ReadBeyond

Pfad: /readbeyond

ReadBeyond.zip	Projekt: Benutzerskript, das im Zuge der Arbeit entwickelt wurde inklusive Dateien, die am Webserver liegen

Quellenverzeichnis

Literatur

[1] Sofiane Abbar u. a. „Real-time recommendation of diverse related articles“. In: *Proceedings of the 22nd international conference on World Wide Web*. WWW '13. Republic und Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2013, S. 1–12.

[2] Ali Al-Mahrubi u. a. „http://dayta.me - A personal news + data recommender for your day“. entstanden im Zuge der LISC 2011: 1st International Workshop on Linked Science. Okt. 2011.

[3] Dean Allemang und James Hendler. *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.

[4] Vladimir Apostolski, Ljupco Jovanoski und Dimitar Trajanov. „Linked Data-Based Social Bookmarking and Recommender System“. In: *ICT Innovations 2012 Web Proceedings*. Sep. 2012, S. 133–142.

[5] Brian D. Ballentine. In: *Handbook of Research on Open Source Software: Technological, Economic, and Social Perspectives*. Hrsg. von Kirk St.Amant und Brian Still. Idea Group Publishing, 2007. Kap. Greasemonkey and a Challenge to Notions of Authorship, S. 12–22.

[6] Jeremy Banks. *How jQuery can be used in greasemonkey scripts*. Stack Overflow. 2011. URL: http://stackoverflow.com/questions/2246901/how-can-i-use-jquery-in-greasemonkey-scripts-in-google-chrome.

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

[8] Robert Baumgartner. „Methoden und Werkzeuge zur Webdatenextraktion“. In: *Semantic Web - Wege zur vernetzten Wissensgesellschaft*. Hrsg. von Tassilo Pellegrini und Andreas Blumauer. Berlin, Heidelberg: Springer-Verlag, 2006, S. 419–435.

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

[10] Robin Burke. In: *The Adaptive Web: Methods and Strategies of Web Personalization*. Hrsg. von Peter Brusilovsky, Alfred Kobsa und Wolfgang Nejdl. Berlin, Heidelberg: Springer-Verlag, 2007. Kap. Hybrid Web Recommender Systems, S. 377–408.

[11] Fefie Dotsika. „Semantic APIs: Scaling up towards the Semantic Web“. In: *International Journal of Information Management* 30.4 (Aug. 2010), S. 335–342.

[12] Marc Ehring und Rudi Studer. „Semantische Annotationen“. In: *Semantic Web - Wege zur vernetzten Wissensgesellschaft*. Hrsg. von Tassilo Pellegrini und Andreas Blumauer. Berlin, Heidelberg: Springer-Verlag, 2006, S. 469–484.

[13] Randy Farmer und Bryce Glass. *Web Reputation Systems*. O'Reilly Media/Yahoo Press, 2010.

[14] David Flanagan. *JavaScript: The Definitive Guide*. Sixth. O'Reilly Media, Inc., 2011.

[15] Simon Franz. „TRecs - Entwicklung eines Music Recommender auf Basis von Last.fm“. Masterarbeit. Freiburg im Breisgau, Deutschland: Albert-Ludwigs-Universität Freiburg, Sep. 2012.

[16] W3C OWL Working Group. *OWL 2 Web Ontology Language Document Overview (Second Edition)*. W3C Recommendation. W3C, Dez. 2011. URL: http://www.w3.org/TR/owl2-overview/.

[17] Jeff Heflin. *OWL Web Ontology Language: Use Cases and Requirements*. W3C Recommendation. World Wide Web Consortium (W3C), Feb. 2004. URL: http://www.w3.org/TR/2004/REC-webont-req-20040210/.

[18] Pascal Hitzler u. a. *Semantic Web - Grundlagen*. Berlin: Springer, 2008.

[19] Dietmar Jannach u. a. *Recommender Systems: An Introduction*. Cambridge University Press, 2011.

[20] Graham Klyne und Jeremy J. Carroll. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C Recommendation. World Wide Web Consortium (W3C), Feb. 2004. URL: http://www.w3.org/TR/rdf-concepts/.

[21] Marja-Riitta Koivunen und Eric Miller. „W3C Semantic Web Activity“. In: *Proceedings of the Semantic Web Kick-off Seminar in Finland*. http://www.w3.org/2001/12/semweb-fin/w3csw. Nov. 2001.

[22] Raymond Kosala und Hendrik Blockeel. „Web mining research: a survey“. In: *SIGKDD Exploration Newsletter* 2.1 (Juni 2000), S. 1–15.

[23] Philippe Le Hégaret, Lauren Wood und Jonathan Robie. *What is the Document Object Model?* W3C Recommendation. World Wide Web Consortium (W3C), Apr. 2004. URL: http://www.w3.org/TR/DOM-Level-3-Core/introduction.html.

[24] Frank Manola und Eric Miller. *RDF Primer, W3C Recommendation.* W3C Recommendation. W3C, Feb. 2004. URL: http://www.w3.org/TR/2004/REC-rdf-primer-20040210/.

[25] Prem Melville, Raymod J. Mooney und Ramadass Nagarajan. „Content-boosted collaborative filtering for improved recommendations". In: *Eighteenth national conference on Artificial intelligence.* Menlo Park, CA, USA: American Association for Artificial Intelligence, 2002, S. 187–192.

[26] Bamshad Mobasher. „Data Mining for Web Personalization". In: *The Adaptive Web: Methods and Strategies of Web Personalization.* Hrsg. von Peter Brusilovsky, Alfred Kobsa und Wolfgang Nejdl. Berlin, Heidelberg: Springer-Verlag, 2007, S. 90–135.

[27] Raymond J. Mooney und Loriene Roy. „Content-based book recommending using learning for text categorization". In: *Proceedings of the fifth ACM conference on Digital libraries.* DL '00. New York, NY, USA: ACM, 2000, S. 195–204.

[28] Thomas Neumayr. „Recommender Systeme - Im Spannungsfeld zwischen Vertrauen und Ablehnung". Masterarbeit. Hagenberg: Fachhochschule Oberösterreich Campus Hagenberg, Juni 2012.

[29] Alexandros Paramythis, Stephan Weibelzahl und Judith Masthoff. „Layered evaluation of interactive adaptive systems: framework and formative methods". In: *User Modeling and User-Adapted Interaction* 20.5 (Dez. 2010), S. 383–453.

[30] Michael J. Pazzani und Daniel Billsus. „Content-Based Recommendation Systems". In: *The Adaptive Web: Methods and Strategies of Web Personalization.* Hrsg. von Peter Brusilovsky, Alfred Kobsa und Wolfgang Nejdl. Berlin, Heidelberg: Springer-Verlag, 2007, S. 325–341.

[31] Thomas Pfeiffer und Bastian Koch. *Social Media : wie Sie mit Twitter, Facebook und Co. Ihren Kunden näher kommen.* München, Deutschland: Addison-Wesley Verlag, 2011.

[32] Dimitrios Pierrakos u. a. „Web Usage Mining as a Tool for Personalization: A Survey". In: *User Modeling User-Adapted Interaction* 13.4 (2003), S. 311–372.

[33] Mark Pilgrim. *Greasemonkey Hacks: Tips & Tools for Remixing the Web with Firefox (Hacks).* O'Reilly Media, Inc., 2005.

[34] Christoph Prevezanos. *Computer-Lexikon 2011 - Über 1000 Seiten mit Fachbegriffen: Die ganze digitale Welt zum Nachschlagen*. München, Deutschland: Markt+Technik Verlag, 2010.

[35] Pearl Pu, Li Chen und Rong Hu. „Evaluating recommender systems from the user's perspective: survey of the state of the art". In: *User Modeling and User-Adapted Interaction* 22.4-5 (Okt. 2012), S. 317–355.

[36] Gerald Reif. „Semantische Annotationen". In: *Semantic Web - Wege zur vernetzten Wissensgesellschaft*. Hrsg. von Tassilo Pellegrini und Andreas Blumauer. Berlin, Heidelberg: Springer-Verlag, 2006, S. 405–418.

[37] Martin P. Robillard. „What Makes APIs Hard to Learn? Answers from Developers". In: *IEEE Software* 26.6 (Nov. 2009), S. 27–34.

[38] Peter Van Roy und Seif Haridi. *Concepts, Techniques, and Models of Computer Programming*. Cambridge, MA, USA: MIT Press, 2004.

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

[40] Nigel Shadbolt, Tim Bernes-Lee und Wendy Hall. „The Semantic Web Revisited". In: *IEEE Intelligent Systems* 21.3 (Mai 2006), S. 96–101.

[41] Oreste Signore. *Representing Knowledge in the Semantic Web*. Juni 2005. URL: http://www.w3c.it/papers/openCulture2005.pdf.

[42] Michael K. Smith, Chris Welty und Deborah L. McGuinness. *OWL Web Ontology Language Guide*. W3C Recommendation. W3C, Feb. 2004. URL: http://www.w3.org/TR/owl-guide/.

[43] Rudi Studer, Stephan Grimm und Andreas Abdecker. *Semantic Web Services: Concepts, Technologies, and Applications*. Springer, 2007.

[44] Alvin Toffler. *The Third Wave*. William Morrow & Company, 1980.

[45] S. R. Turner. „Active Browsing". In: *Proceedings of the Fifth IASTED International Conference Internet and Multimedia Systems and Applications (IMSA 2001), August 13-16, 2001, Honolulu, Hawaii, USA*. Hrsg. von M. H. Hamza. IASTED/ACTA Press, 2001, S. 181–186.

[46] Gias Uddin, Barthélémy Dagenais und Martin P. Robillard. „Temporal analysis of API usage concepts". In: *Proceedings of the 2012 International Conference on Software Engineering*. ICSE 2012. Piscataway, NJ, USA: IEEE Press, 2012, S. 804–814.

[47] Liyang Yu. *A Developers Guide to the Semantic Web*. Berlin [u.a.]: Springer, 2011.

Online-Quellen

[48] URL: http://www.zemanta.com/ (besucht am 24.05.2013).

[49] URL: http://www.opencalais.com/ (besucht am 24.05.2013).

[50] URL: http://www.alchemyapi.com/ (besucht am 24.05.2013).

[51] URL: http://www.ghacks.net/2010/04/04/why-you-should-use-userscripts-and-not-extensions-when-possible/ (besucht am 03.08.2013).

[52] URL: http://wiki.greasespot.net/Cross-browser_userscripting (besucht am 08.08.2013).

[53] URL: http://wiki.greasespot.net/Greasemonkey_Manual:API (besucht am 08.08.2013).

[54] URL: http://wiki.greasespot.net/Sandbox (besucht am 08.08.2013).

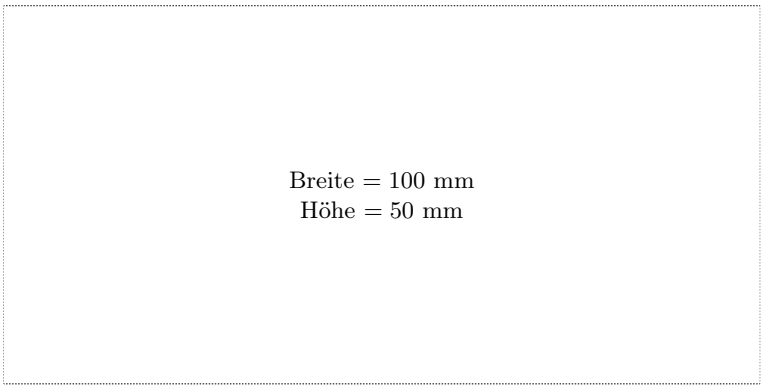
[55] URL: http://www.oewa.at/ (besucht am 02.09.2013).

[56] URL: http://www.w3schools.com/browsers/browsers_stats.asp (besucht am 26.09.2013).

[57] URL: http://blog.viewchange.org/2010/05/entity-extraction-content-api-evaluation/ (besucht am 08.09.2013).

Messbox zur Druckkontrolle

— Druckgröße kontrollieren! —



— Diese Seite nach dem Druck entfernen! —