Entwicklung und Bereitstellung eines Recommender Bundles für das PHP Framework Symfony 2

Bachelorarbeit

vorgelegt am: 02.08.2013

am Fachbereich Mathematik und Informatik der
Freien Universität Berlin

Name:	Jonas Cwojdzinski
Geburtsdatum:	02.05.1988
Matrikelnummer:	4298490
Fachbereich:	Mathematik und Informatik
Studiengang:	Informatik
Erstgutachter:	Prof. Dr. Adrian Paschke
Zweitgutachter:	Prof. Dr. Elfriede Fehr

Eidesstattliche Erklärung

Ich erkläre, dass ich meine Bachelor-Arbeit “Entwicklung und Bereitstellung eines Recommender Bundles für das PHP Framework Symfony 2“ selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe und dass ich alle Stellen, die ich wörtlich oder sinngemäß aus Veröffentlichungen entnommen habe, als solche kenntlich gemacht habe. Die Arbeit hat bisher in gleicher oder ähnlicher Form oder auszugsweise noch keiner Prüfungsbehörde vorgelegen.

Berlin, den 02.08.2013

Jonas Cwojdzinski

Inhaltsverzeichnis

Abbildungsverzeichnis

Beispielverzeichnis

Abkürzungsverzeichnis

Ajax Asynchronous JavaScript and XML

DBS Datenbanksystem

FOAF Friend of a Friend

N3 Notation 3

OWL Web Ontology Language

RAP RDF API for PHP

RDF Resource Description Framework

RDFS RDF Schema

SPARQL SPARQL Protocol And RDF Query Language

SQL Structured Query Language

URI Uniform Resource Identifier

URL Uniform Resource Locator

XML Extensible Markup Language

Web World Wide Web

W3C World Wide Web Consortium

Zusammenfassung

Empfehlungssysteme werden häufig zum Vorschlagen von Produkten im eCommerce eingesetzt. Das liegt zum einen an ihrer vielseitigen Anwendbarkeit, zum anderen an ihrer verkaufsfördernden Wirkung. Die am häufigsten verwendeten Empfehlungssysteme sind der Content-based, der Collaborative und die Hybrid Recommender, wozu auch der Semantic Recommender gehört.
Im Fokus dieser Arbeit steht die Entwicklung eines Symfony 2 Bundles, welches einen Content-based und einen Semantic Recommender enthält. Diese Empfehlungssysteme werden parallel in einer Produktivumgebung eingesetzt und anschließend anhand verschiedener Metriken ausgewertet. Das Bundle wird Open Source freigegeben und kann frei verwendet und weiterentwickelt werden.

1 Einleitung

In dieser Arbeit wird untersucht, was notwendig ist für ein Empfehlungssystem, um einem Benutzer das optimale Produkt aus einer Menge von Produkten anzubieten. In die Berechnung fließen implikative Faktoren ein, die in dieser Arbeit genauer untersucht werden. Des Weiteren wird evaluiert, wie erfolgreich das Empfehlungssystem Produkte vorschlägt anhand der Auswertung von Konvertierungsraten und Umsatz.
Das optimale Produkt definiert sich dadurch, dass es

- für den Benutzer von Interesse ist,
- der Benutzer in der Zielgruppe des Produktherstellers liegt und
- der Produktanbieter - der nicht mit dem Hersteller identisch sein muss - mit diesem Produkt Umsatz erzielt [Xiao & Benbasat 2007].

Ein Produktabschluss umschließt je nach Produkttyp ein Kauf des Produkts, die Übermittlung von Kontaktdaten und Bestätigung der Teilnahmebedingungen eines Gewinnspiels oder die vollständige Beantwortung einer Umfrage.
Je zufriedener der Benutzer mit dem angebotenen Produkt war, desto eher kehrt er wieder zu dem Anbieter zurück [Liang et al. 2007]. Die Benutzerzufriedenheit ist ein wichtiger Faktor, der nicht unterschlagen werden darf. Die Zufriedenheit erhöht sich signifikant, wenn der Benutzer das Gefühl hat im Empfehlungsprozess mitzuwirken. [Liang et al. 2007].
Hersteller von gegenständlichen Produkten positionieren sich auf dem Markt unter anderem durch ihre Preisstrategie, die ihnen z.B. nur zahlungskräftige Kunden beschert, da das Produkt einen hohen Anschaffungspreis besitzt. Für Marketingunternehmen kann ein Produkt auch eine Werbekampagne sein, die sich nur an einen bestimmten Personenkreis richtet. Dort ist es wichtig, dass ihr Produkt nur durch spezifizierte Faktoren gefilterten Personen angeboten wird. Alle Personen, die nicht in dieses Raster fallen, sollten ausgeschlossen werden, da sie im neutralen Fall keinen Umsatz generieren. Meistens entstehen jedoch Kosten, da die Schaltung des Werbebanners, durch den der Benutzer

auf die Werbeseite kommt, Kosten verursacht.
Vorverarbeitung von Daten ist im Internet bereits allgegenwärtig, um Ergebnisse in Echtzeit zu liefern. Als bekanntestes Beispiel lässt sich die Suchmaschine Google nennen. Ohne Vorverarbeitung wäre es für ein System nicht möglich eine Suchanfrage im Millisekunden-Bereich zu bearbeiten. Im Beispiel Google wird das gesamte Internet mittels Crawler regelmäßig durchsucht und indexiert gespeichert. Jede Suchanfrage generiert einer Datenbankabfrage, deren Ergebnis dem Benutzer ausgegeben wird[1].
Zur Zeit (Stand Juli 2013) arbeitet Google an einer semantischen Suche[2], die auf eine Suchanfrage statt Links zu passenden Webseiten direkt die möglichen Antworten wiedergibt. Dazu wird eine Wissensbasis benötigt, die Verbindungen zwischen Begriffen speichert. Anhand dieses Wissens ist es möglich auf Fragen direkt zu antworten, wie etwa die größten Städte an der Spree aufzulisten.
In dieser Bachelorarbeit wird ein Grundstock an Wissen aufgebaut, um die semantische Suche auf Werbekampagnen zu beziehen. Dies ermöglicht mit einer gewissen Wahrscheinlichkeit Produktvorschläge vorherzusagen, die dem Benutzer gefallen. Dazu wird im Kapitel 2 die Hypothese aufstellen, dass ein semantisches Empfehlungssystem erfolgreicher Werbekampagnen vorschlägt als ein inhaltbasiertes Empfehlungssystem. In Kapitel 3 werden die theoretischen Grundlagen der Empfehlungssysteme und des semantischen Webs vorgestellt. Das folgenden Kapitel 4 geht auf die implizite Bewertung ein, die in Empfehlungssystemen eine große Rolle spielt. Die implizite Bewertung beschreibt die Auswertungen von Verhalten einer Person zu einer Handlung. Das Augenmerk liegt in dieser Arbeit auf dem Browserverhalten. Dazu gehört u.a. welches Produkt angeklickt wird oder wohin gescrollt wird. Die entwickelte Methodik wird dazu in Kapitel 5 eruiert. Nachfolgend wird Kapitel 6 die beiden entwickelten Empfehlungssysteme anhand verschiedener Faktoren vergleichen und bewerten. Im letzten Kapitel wird ein Fazit sowie ein Ausblick auf kommende Verbesserungen der entwickelten Empfehlungssysteme gegeben.

[1] http://www.google.de/intl/de/insidesearch/howsearchworks/crawling-indexing.html (abgerufen am 22. Juli 2013)
[2] http://googleblog.blogspot.de/2012/08/building-search-engine-of-future-one.html (abgerufen am 22. Juli 2013)

2 Fragestellung und Hypothese

2.1 Fragestellung

Diese Arbeit ermittelt, mit welchen Mitteln man die große Informationsvielfalt von Produktvorschlägen im Bereich Online-Marketing für den Benutzer aufbereiten kann und anhand welcher Eigenschaften das am besten passendste Produkt für den Benutzer herausgefiltert wird.
Des Weiteren sollen implikative Indikatoren und deren Einflüsse sondiert werden, die Aufschluss über die Konvergenz zwischen dem angebotenen Produkt und dem Benutzer geben. Diese Informationen fließen in das selbstlernende Empfehlungssystem ein und verbessern stetig die nächsten Produktvorschläge. Das Hauptaugenmerk liegt ganz bewusst auf mobilen Endgeräten (Laptop, Tablet, Smartphone), die im Jahr 2012 schon 41% des weltweiten Internettraffics ausmachten und diesen Anteil bis 2017 voraussichtlich auf 55% steigern werden[1]. Es gilt zu ergründen, was für Informationen sich aus dem Mobilgerät filtern lassen und welche Schlüsse daraus gezogen werden können.

2.2 Hypothese

Laut Schätzungen verdoppelt sich das menschliche Wissen alle fünf bis sieben Jahre [Rüttgers 1999]. Dabei spielt das Internet eine wichtige Rolle bei der Verbreitung von Informationen. Im Jahr 2002 enthielt das World Wide Web etwa 170 Terabyte an Wissen [Lyman & Varian 2004]. Mit steigendem Wissen erhöht sich auch der Aufwand dieses Wissen zu verarbeiten. Dementsprechend muss stark differenziert werden, welches Wissen von Nutzen ist und benötigt wird. Diese Abwägung ist für die Wissensbasis des semantischen Empfehlungssystems (Semantic Recommender) essenziell.
Es ist davon auszugehen, dass der Semantic Recommender mit seiner Wissensbasis erfolgreicher arbeitet als der Content-based Recommender je umfassender die Wissensbasis

[1] http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html (abgerufen am 27. Juli 2013)

ist. Je besser das vorgeschlagene Produkt zu den Vorlieben des Benutzers passt, desto eher schließt er das Produkt ab und besucht die Webseite bzw. den Produktanbieter wieder. Durch die eingangs erwähnten Indikatoren der impliziten Bewertung, sowie die im Kapitel 6 folgenden Evaluierungsparameter werden diese Annahmen überprüft.

3 Grundlagen

3.1 Recommender

Empfehlungssysteme sind eine Technologie, die einem Benutzer für ihn nützliche Artikel vorschlagen. Artikel können je nach Anwendungsfall **materielle** Produkte (Bücher, CDs, Kleidung), wozu auch die **digitalen** Produkten (eBooks, MP3s) zählen und **immaterielle** Produkte (Werbekampagnen, Urlaubsreisen, Suchergebnisse) sein [Kotler 2010]. Empfehlungssysteme sind relativ neu im Gegensatz zu Suchmaschinen und Datenbanken und wurden Anfang der 90er Jahre [Hey 1991, Resnick et al. 1994, Hill et al. 1995] das erste Mal erwähnt. Recommender Systems sind häufig im eCommerce verbreitet. Das liegt zum einen an ihrer vielseitigen Anwendbarkeit, zum anderen an ihrer verkaufsfördernden Wirkung [Schafer et al. 2001]. Um zu analysieren, was einem Benutzer gefällt, benötigt man im Idealfall viele Informationen über die Produktpalette und den Benutzer. Der wichtigste Arbeitsschritt ist dabei, Produkte und Benutzer richtig zueinander in Beziehung zu setzen. Um die Interessen des Benutzers zu identifizieren, ist es nützlich, Informationen durch implizites und explizites Feedback über ihn zu sammeln. Auf die Vor- und Nachteile dieses Feedbacks wird gesondert in Kapitel 4 eingegangen. Anhand der gesammelten Daten / der identifizierten Informationen können dem Benutzer Produkte vorgeschlagen werden. Diese personalisierten Vorschläge unterscheiden sich von den nicht-personalisierten, die z. B. die beliebtesten Produkte aller Benutzer umfassen. Zur Einschätzung wird dem vorzuschlagenden Produkt ein Wert zugewiesen, der ausdrückt, inwieweit der Artikel den Interessen des Benutzers entspricht. Diese Charakterisierung kann sich an das Benutzerverhalten anpassen. Ziel ist es, die Benutzerzufriedenheit und dadurch die Loyalität zu erhöhen [Schafer et al. 1999]. Dies geschieht nach [Xiao & Benbasat 2007] z. B. durch:

- Einbeziehung von Benutzerfeedback in die Vorschlagergebnisse
- Zeitersparnis - der Benutzer muss nicht nach dem Produkt suchen und bekommt im Optimalfall das gesuchte Produkt vorgeschlagen

- Verständnis für die Interessen des Nutzer
- Vorhersage von nicht gesuchten, aber potenziell interessanten Produkten
- Vorschlag/Einbeziehung von andere beliebte Produkte (cross-sell)

Je nach Bedarf bietet es sich an einen Zufallsgenerator zu verwenden, der das Ergebnis mit einigen Produkten erweitert und so die Möglichkeit bietet weitere Interessensgebiete des Benutzers zu entdecken.
Die am häufigsten verwendeten Empfehlungssysteme sind der Content-based Recommender und der Collaborative Recommender. Nachfolgend werden zusätzlich die Hybrid Recommender vorgestellt, die eine besondere Gattung der Empfehlungssysteme darstellen (Abbildung 3.1). Gesondert wird auf den Semantic Recommender eingegangen, der ein wesentlicher Teil dieser Arbeit ist.

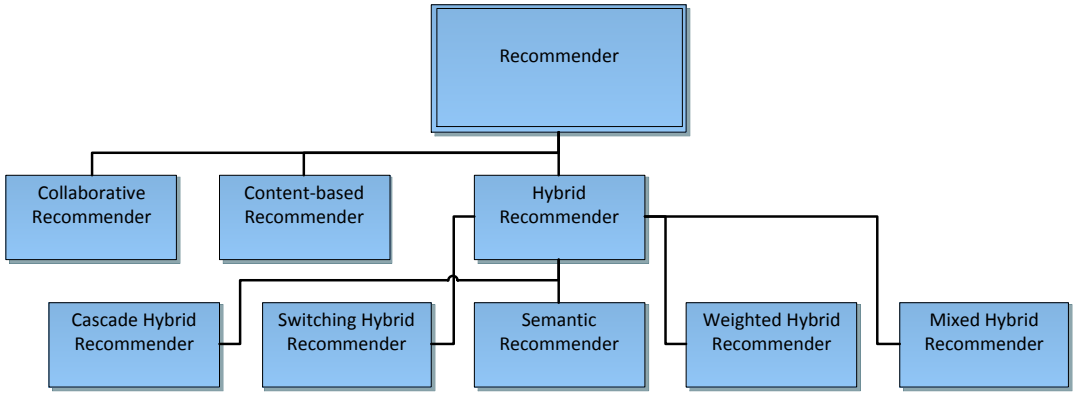


Abbildung 3.1: Recommender Organigramm

3.1.1 Content-based Recommender

Der Content-based Recommender hat seinen Ursprung im *Information Filtering* [Belkin & Croft 1992]. Das Forschungsgebiet beschäftigt sich mit der automatisierten Bereinigung von redundanten oder ungewollten Informationen in einem Datenstrom. Verwendet wird das System u.a. bei der Filmdatenbank Internet Movie Database[1].
Content-based Recommender schlagen Elemente vor, die gleiche oder ähnliche Ausprägungen haben wie bereits vorgeschlagene Objekte, die den Benutzer interessieren. Jedes einzelne Objekt wird anhand fest definierter diskreter Attribute charakterisiert. Jedes Attribute besitzt eine Gewichtung, die positive oder negative Werte annehmen kann.

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

Um Elemente mit den Benutzerinteressen vergleichbar zu machen, wird ein Profil anhand der bereits positiv bewerteten Elemente erstellt [Pazzani & Billsus 2007]. Anhand eines Ähnlichkeitsmaßes - z. B. Cluster Analyse oder Entscheidungsbaum - lässt sich die Ähnlichkeit von Benutzerprofil und den Elementen feststellen. Das System lernt von den Vorlieben des Benutzers indem es sein Verhalten beobachtet.
Problematisch sind neue Benutzer, deren Profile noch keine ausreichende Detailtiefe besitzen. Dieser Umstand wird auch *New User* Problem genannt. Es gibt unterschiedliche Methoden einen neuen Benutzer zu beurteilen. Eine Option ist es, den Nutzer auf Basis seiner demografischen Daten zu beurteilen, sofern diese verfügbar sind. Andernfalls wäre es auch möglich, die beliebtesten Elemente aller Benutzer oder zufällige Elemente vorzuschlagen [Rashid et al. 2002].
Eine weitere Gefahr ist die Überspezialisierung eines Profils. Dies ist der Fall, wenn dem Benutzer nur Produkte vorgeschlagen, die zu ähnlich oder schon bekannt für den Benutzer sind. Im Optimalfall werden Elemente aus verschiedenen Bereichen unterbreitet und so ein breites Spektrum an Empfehlungen auswählbar gemacht.

3.1.2 Collaborative Recommender

Die Empfehlungen des Collaborative Recommenders basieren auf Elementen, die von Benutzern mit ähnlichem Interesse positiv bewertet wurden [Balabanović & Shoham 1997]. Als Grundlage dient dazu das Clustering von Interessen. Personen mit gleichen Interessen mögen häufig ähnliche Dinge. Die Bewertung kann implizites und explizite Feedback (siehe Kapitel 4) eines Benutzer umfassen, aus dem ein Interessenprofil mit Vorlieben und Abneigungen erstellt wird. Dazu gibt es zwei verschiedene Algorithmen [Breese et al. 1998]:
Der *memory-based* Algorithmus basiert auf statistischen Techniken, die die gesamte Datenbank nach den nächsten Nachbarn - d.h. ähnlichen Profilen - durchsuchen (*k-nearest neighbour*-Algorithmus [Fix & J. L. Hodges 1951]). Dazu wird zuerst die durchschnittliche Bewertung $\overline{v}_i$ eines Benutzers i für alle von ihm bewerteten Elemente I_i mit der Bewertung $v_{i,j}$ berechnet (*mean vote* Formel).

$$\overline{v}_i = \frac{1}{|I_i|} \sum_{j \epsilon I_i} v_{i,j}$$

Mit der Formel $p_{a,j}$ kann das Interesse vom aktuellen Benutzer a an Element j abgeschätzt werden. Dazu wird eine Gewichtungsfunktion $w(a, i)$ verwendet, die den Abstand der Bewertung zwischen dem aktuellen und n anderen Benutzern wiedergibt. Häufig wird

dafür der *Pearson* Algorithmus [Pearson 1895] oder die Kosinus-Ähnlichkeit verwendet [Breese et al. 1998]. Zusätzlich wird der Parameter k zur Normalisierung eingeführt. Die Bewertungsfunktion $v_{i,j}$ stellt die Evaluierung von Benutzer i zu Element j dar.

$$p_{a,j} = \overline{v}_a + k \sum_{i=1}^{n} w(a,i)(v_{i,j} - \overline{v}_i)$$

Der *model-based* Algorithmus nutzt die Datenbank um ein Benutzermodell anzulegen, welches für die Prognose verwendet wird. Der geschätzte Erwartungswert $E(v_{a,j})$ von Benutzer a für Element j kann Werte zwischen 0 und m annehmen. Für die Vorhersage werden häufig ein *Cluster Modell* oder ein *Bayesian Netzwerk*[2] angewandt.

$$p_{a,j} = E(v_{a,j}) = \sum_{i=0}^{m} Pr(v_{a,j} = i | v_{a,k}, k \in I_a) i$$

Collaborative Recommender finden häufig Anwendung bei großen Medienportalen und sozialen Netzwerken, wie z.B. YouTube oder last.fm. Facebook verwendet Empfehlungssysteme u.a. zur Freundesuche [Zheng et al. 2008]. Dort werden dem Benutzer die Personen mit den meisten gemeinsamen Freunden angeboten. Die oben genannten Portale besitzen alle Medieninhalte, die nicht oder nur schwer objektiv eingeschätzt werden können. Durch die subjektive Einschätzung wird diese Einschränkung vermieden.
Die kollaborative Komponente birgt auch einige Unwägbarkeiten. Zum einen ist die Suche nach potenziellen Nachbarn in Millionen Datensätzen ziemlich aufwändig (**Skalierbarkeit**). Zum anderen wird eine "kritische Masse" an Daten benötigt, um dienliche Ergebnisse zu liefern (**Datendichte**) [Balabanović & Shoham 1997]. Des Weiteren sind Empfehlungen des System durch manipulierte Bewertung angreifbar (**Manipulation**) [Herlocker et al. 2004]. Zudem ist eine ausgereifte *cold start* Strategie dienlich, die den Umgang mit neuen Benutzern bzw. neuen Elementen regelt.

3.1.3 Hybrid Recommender

Hybride Empfehlungssysteme kombinieren zwei oder mehrere Empfehlungssysteme und verbinden deren Stärken zu einem verbesserten System. Häufig werden dabei Collaborative Recommender mit anderen Techniken verknüpft. Hierbei unterscheidet man zwischen *Cascade*, *Weighted*, *Switching* und *Mixed* Hybrid Recommender [Burke 2002]. Kaskadierende (*Cascade*) Recommender verknüpfen zwei Recommender miteinander

[2]Graph mit Knoten als Zufallsvariablen und Kanten als bedingte Wahrscheinlichkeit

und verwenden die Ergebnisse eines ersten in einem zweiten Recommender. Durch diese erneute Berechnung erhalten die Vorschläge die Bewertung beider Empfehlungssysteme.
Ein *weighted* Recommender sammelt die Ergebnisse der einzelnen Recommender und gewichtet die resultierenden Elemente nach einem festen Faktor. Ein großer Vorteil daran ist die nachträgliche Anpassung der Ergebnisreihenfolge.
Der Fokus des *switching* Recommender liegt auf der Elementebene. Hier wird das System gewechselt, falls das aktuell verwendete System keine zufriedenstellenden Elemente liefern kann oder fehlerhaft arbeitet. Wenn beispielsweise ein Content-based Recommender eine zu geringe Datendichte besitzt, um dienliche Ergebnisse berechnen zu können, kann ein Collaborative Recommender verwendet werden, der seine Berechnung anhand der nächsten Nachbarn trifft.
Ein *Mixed Hybrid Recommender* mischt die Ergebnisse aller kombinierten Empfehlungssysteme zusammen. Das kann vorteilhaft sein, um das sogenannte *New Item* Problem zu vermeiden. Falls ein neues Element noch keine Bewertungen besitzt, besteht trotzdem die Möglichkeit, dass es aufgrund seiner Ausprägungen angeboten wird.
Im Folgenden wird gesondert auf den Semantic Recommender eingegangen, da er ein Hauptbestandteil des angebotenen Bundles ist.

Semantic Recommender

Der Semantic Recommender ist ein besonderer Hybrid Recommender und benutzt semantische Informationen zur Generierung von Vorschlägen. Dieses Wissen wird meist als Konzeptdiagramm (z.B. Taxonomie oder Thesaurus) oder Ontologie modelliert. Dazu ist es notwendig, das Wissen über einen Benutzer und die vorzuschlagenden Elemente interpretierbar zu machen. Resource Description Framework (RDF)-Triple speichern die Informationen maschinenlesbar in einer semantischen Wissensbasis. Diese Daten fließen in die Vorschläge ein. Um aussagekräftige Ergebnisse zu erzielen, bedarf es einer Mindestmenge an Informationen über Benutzer bzw. Element.
Der große Vorteil von Wissensbasen sind u.a. die modellierbaren Querverbindungen von Elementen. Somit können Elemente mit unterschiedlichen Ausprägungen aber ähnlichen charakteristischen Ausprägungen verbunden werden. Zusätzlich sind die Manipulationsmöglichkeiten wie bei einem Collaborative Recommender ausgeschlossen, da die Empfehlung auf diskreten Werten innerhalb der Wissensbasis basiert. Diese Heterogenität der Informationsrepräsentation ist jedoch gleichzeitig auch ein Defizit des Recommenders. Meist sind die Informationen auf ein Anwendungsgebiet spezialisiert und müssen für andere Bereiche angepasst werden.

Wissenbasen sind für semantische Empfehlungssysteme unerlässlich, daher werden im Weiteren zwei frei zugängliche Wissensbasen vorgestellt.

3.2 Wissensbasen

Wissensbasen sind Speicher für schriftliches Wissen und sind die Grundlage für das Wissensmanagement. Darin gespeichert werden Informationen, wie z.B. Artikel oder Ontologien. Unterteilt werden die Datenbanken in menschenlesbare (Beispiel: Wikipedia) und maschinenlesbare (Beispiel: DBPedia) Wissensbasen [Auer et al. 2007]. Im Weiteren wird beispielhaft eine Basis aus jeweils einem dieser Bereiche kurz vorgestellt.

3.2.1 Wikipedia

Die deutsche Wikipedia ist eine kostenlose Enzyklopädie. Sie basiert auf gemeinschaftlich erstellten Artikeln, die regelmäßig erweitert und aktualisiert werden. Durchschnittlich wird das Onlinelexikon 1,3 Mio. Mal pro Stunde aufgerufen[3]. Die deutschsprachige Version besitzt über 1,6 Mio. Artikel und ist nach der englischen die größte Datenbank.
Jeder erstellte oder geänderte Artikel wird gesichtet, jedoch ist das keine Garantie, dass der Artikel korrekt ist. Textstellen sind wegen der eingeschränkten Qualität und der Anonymität der Autoren nur bedingt zitierfähig.

3.2.2 DBPedia

Die DBPedia extrahiert die Information der Wikipedia und erstellt mit speziellen Algorithmen eine RDF-Repräsentation der Daten. Sie bietet strukturierte, verknüpfte Daten der Wikipedia auf Basis von Ontologien an. Der Zugriff auf die Daten wird über Download der Datensätze oder Abfragen an den SPARQL Protocol And RDF Query Language (SPARQL) Endpunkt ermöglicht. Als Standard wird das RDF benutzt und dadurch wird eine Verknüpfung mit anderen freien Datenbeständen, wie z.B. das Digital Bibliography & Library Project[4] oder das Statistische Amt der Europäischen Union[5], angeboten.

[3] http://de.wikipedia.org/wiki/Wikipedia:Über_Wikipedia (abgerufen am 31. Juli 2013)
[4] http://dblp.uni-trier.de/
[5] http://epp.eurostat.ec.europa.eu/portal/page/portal/eurostat/home/

3.3 Semantic Web

Das Semantic Web ist ein Konzept zur Erweiterung des World Wide Web (Web) zum sogenannten Web 3.0. Ziel ist es Informationen der Menschen für Computer interpretierbar zu machen und die Daten zwischen realer und digitaler Welt zu verknüpfen. Um Informationen deutbar für Maschinen zu machen, müssen sie nach ihrer Bedeutung verknüpft werden. Das Web 3.0 gilt als Vorbereitung zum Internet der Dinge[6]. Die Grundlage zum Semantic Web sind der Uniform Resource Identifier (URI) und die Extensible Markup Language (XML). Das Resource Description Framework-Modell und Ontologien, wie z.B. die Web Ontology Language, bauen auf diesem Unterbau auf. In der Abbildung 3.2 sind die Konzept des Semantischen Webs dargestellt. Im Folgenden werden einige dieser Technologien vorgestellt.

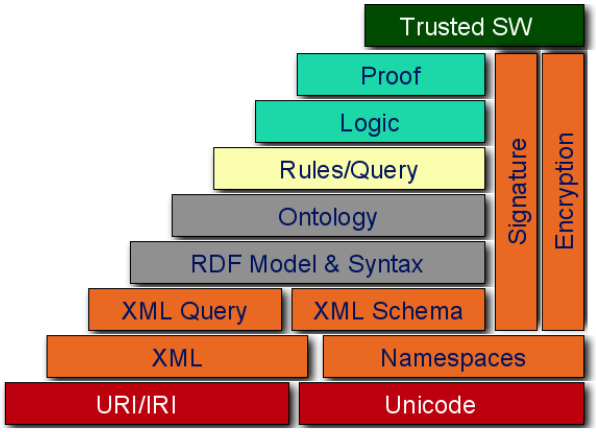


Abbildung 3.2: Semantic Web Stack[7]

3.3.1 Resource Description Framework

Das RDF ist eine formales Modell zur Beschreibung von logischen Aussagen über Ressourcen im Web. RDF gilt als Grundlage für das Semantische Web. Mittels dieses Frameworks ist es möglich strukturierte, maschinenlesbare Beschreibungen von Aussagen zu erstellen. Das RDF-Modell besteht aus einer Sammlung von Aussagen, die jeweils durch ein **Tripel** definiert werden. Jedes Tripel besitzt drei Elemente (Subjekt, Prädikat

[6]Physische Geräte besitzen digitalen Ableger im Internet.
[7]Grafik von der offiziellen W3C Website entnommen: http://www.w3.org/2004/Talks/1117-sb-gartnerWS/slide18-0.html (31.07.2013)

und Objekt) [Decker et al. 2000]. **Subjekt** und **Objekt** beschreiben jeweilig eine Ressource, deren Beziehung zueinander über das **Prädikat**, was auch eine RDF-Ressource sein kann, definiert wird. Eine Ressource kann ein Literal[8] oder eine RDF-Ressource sein, die über einen eindeutigen Bezeichner identifiziert wird. Um die Eindeutigkeit von RDF-Ressourcen zu wahren, werden meist URI verwendet, die einer Uniform Resource Locator (URL) ähneln. Beispiel 3.1 zeigt zwei Beispiele für den URI einer Ressource.

Beispiel 3.1: Beispiele für URI von RDF-Ressourcen

```
http://xmlns.com/foaf/0.1/name
http://www.w3.org/TR/rdf-schema/#ch_resource
```

Das Framework ermöglicht es, Informationen aus verschiedenen Quellen zusammenführen. Das ist beispielsweise sinnvoll, wenn Synonymen bei der Volltextsuche verbunden werden sollen. RDF ist unabhängig von fest definierten textuellen Repräsentationen. Als gängigstes Format haben sich aber die XML und Notation 3 (N3) zur Serialisierung[9] verbreitet. Das Beispiel 3.2 wurde von *W3Schools*[10] übernommen und ist eine mögliche RDF Darstellung einer CD von Bob Dylan. Beschrieben wird neben dem Name des Musikers auch das Herkunftsland (Zeile 9), das Plattenlabel (Zeile 10), der Kaufpreis (Zeile 11) und das Erscheinungsjahr (Zeile 12).

Beispiel 3.2: XML-basierte Repräsentation in RDF

```
1  <?xml version="1.0"?>
2    <rdf:RDF
3  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4  xmlns:cd="http://www.recs hop.fake/cd#">
5
6    <rdf:Description rdf:about="
7     http://www.recshop.fake/cd/Empire Burlesque">
8       <cd:artist>Bob Dylan</cd:artist>
9       <cd:country>USA</cd:country>
10      <cd:company>Columbia</cd:company>
11      <cd:price>10.90</cd:price>
12      <cd:year>1985</cd:year>
```

[8] Bei einem Literal handelt sich meist um einen primitiven Datentyp, wie etwa eine Ganzzahl, Wahrheitswert oder Zeichenkette.

[9] Mittels Serialisierung wird ein Objekt in einen Datenstrom z.B. zur Übertragung umgewandelt.

[10] http://www.w3schools.com/rdf/rdf_main.asp

```
13   </rdf:Description>
14 </rdf:RDF>
15  N3: http://www.w3.org/TR/rdf-schema/#ch_resource
```

RDF-Graph

Das RDF-Modell basiert aufgrund seiner Tripel auf einem gerichteten Graphen, der meist in einer Graphendatenbank - auch Tripelstore genannt - gespeichert wird. Das Speichern von Graphen ist im Tripelstore effizienter als in einem konventionellen, relationalen Datenbanksystem (DBS) aufgrund von Traversionsalgorithmen.
Abfragen werden mit der Sprache SPARQL ausgeführt, die Structured Query Language (SQL) ähnelt. Auf SPARQL wird gesondert in Kapitel 3.3.6 eingegangen.

3.3.2 Resource Description Framework Schema

Eine Erweiterung vom RDF ist das RDF Schema (RDFS). Es ist eine Empfehlung vom World Wide Web Consortium (W3C) und erweitert RDF um objekt-orientierte Konstrukte. Zusätzlich bietet es die Grundlage für andere RDF Ontologien eine bestimmte Anwendungsdomäne zu modellieren. Das RDFS unterteilt sich in 2 Bereiche:

- Der Bereich **Klassen** umschreibt die Typen *Class*, *Resource*, *Property* und *Literal*.
- Die Bereich **Eigenschaften** gliedern sich in *subClassOf*, *subPropertyOf*, *domain* und *range* auf.

Viele dieser Komponenten von RDFS werden von anderen Ontologien genutzt, wie z.B. von Friend of a Friend oder der Web Ontology Language.

Friend of a Friend

Friend of a Friend (FOAF) ist ein RDF Schema zur Modellierung von semantischen Informationen zu sozialen Netzwerken. Das Schema gilt als eine der ersten Semantic Web Anwendungen. Mittels FOAF können Angaben über eine Person gemacht werden wie z.B. Name, Email-Adresse, Webseiten, Interessen, Fotos und befreundete Personen. Beispiel 3.3 zeigt ein beispielhaftes RDF Dokument mit den FOAF Angaben zu der Person Max Mustermann.

Beispiel 3.3: beispielhafte FOAF Angaben

```
<rdf:RDF
```

```
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  <foaf:PersonalProfileDocument rdf:about="">
    <foaf:maker rdf:resource="#me"/>
    <foaf:primaryTopic rdf:resource="#me"/>
  </foaf:PersonalProfileDocument>
  <foaf:Person rdf:ID="me">
    <foaf:name>Max Mustermann</foaf:name>
    <foaf:givenname>Max</foaf:givenname>
    <foaf:family_name>Mustermann</foaf:family_name>
    <foaf:knows>
      <foaf:Person>
        <foaf:name>Manuela Mustermann</foaf:name>
      </foaf:Person>
    </foaf:knows>
  </foaf:Person>
</rdf:RDF>
```

Durch die standardisierte Darstellung der Daten ist es möglich, dass z.B. Webcrawler[11] die personenbezogenen Informationen automatisiert auslesen und verarbeiten. Um dieses Risiko zu minimieren, wurde das FOAF+SSL Protokoll entwickelt. Es gewährleistet eine Authentifizierung über ein Zertifikat und stellt sicher, dass die anfragende Person auch berechtigt ist, das FOAF Dokument zu sehen.

3.3.3 Web Ontology Language

Die Web Ontology Language (OWL) ist eine Deklaration des W3C und erweitert die Mächtigkeit vom RDFS. Durch das große Vokabular gibt es mehr Interpretationsmöglichkeiten als durch XML, RDF und RDFS. OWL wird in drei Untersprachen unterteilt:

- OWL Lite besitzt Klassenhierarchien und schlichte Abhängigkeiten um einfache Taxonomien gestalten zu können. Unterstützt werden z.B. Hierarchien von Unterklassen und Restriktionen auf Eigenschaften.
- OWL DL bietet Entwicklern Ausdrücke für die Modellierung von Berechenbarkeit und Entscheidbarkeit. Die Untersprache enthält alle OWL Sprachkonstrukte,

[11] Webcrawler durchsuchen und analysieren automatisiert das Web und werden häufig von Suchmaschinen verwendet.

deren Benutzung aber gewissen Einschränkungen unterliegt. Beispielsweise darf eine Klasse nicht eine Instanz einer anderen Klasse sein. Der Zusatz DL steht für Beschreibungslogik (description logic).

- OWL Full verfügt über die selben Konstrukte wie OWL DL, jedoch ohne die bei OWL DL genannten Einschränkungen. Dadurch sind die angelegten Ontologien nicht immer entscheidbar.

3.3.4 Mikroformate

Mikroformate sind semantische Markups in HTML oder XHTML. Sie bieten den Vorteil, dass die Dokumente von Maschinen und Menschen gleichzeitig lesbar sind. Insgesamt gibt es ca. 90 Elementtypen zu Annotation. Jedoch sind Mikroformate für Maschinen nicht vollständig semantisch, da es z.B. keinen Standard für den Tag *address* existiert. In Beispiel 3.4 ist eine Addresse gezeigt. Maschinenlesbar ist lediglich die Website der Firma. Die restlichen Informationen sind nur menschenlesbar.

Beispiel 3.4: Mikroformat in HTML

```
<address>
Mustermotorenwerke GmbH<br />
Obergasse 1a<br />
12345 Musterhausen<br />
Web: <a href="http://www.mmw.de/">www.mmw.de</a>
</address>
```

Dublin Core

Dublin Core ist ein Vokabular zur Beschreibung von Dokumenten im Semantic Web. Das Vokabular besitzt 15 Merkmale, die ein Dokument spezifizieren. Dazu gehören u.a. Identifier (Bibliotheksnummer, ISBN, etc.), Format für digitale Dokumente (MIME-Type), Sprache, Titel, Kurzzusammenfassung, Autor, Lizenz, Lebenszyklus und referenzierte Dokumente. Die Verwendung vom Dublin Core hält sich in Grenzen, da die führenden Suchmaschinen diese Metadaten ignorieren. In der Vergangenheit wurden durch manipulierte Dokumentbeschreibungen die Ergebnisse zu sehr verfälscht, wodurch sich das Interesse der Suchmaschinen am Dublin Core verringerte[12].

[12] http://dublincore.org/resources/faq/#whatsearchenginessupport (abgerufen am 31. Juli 2013)

GoodRelations

GoodRelations ist eine OWL DL Ontologie für den eCommerce und wird im Gegensatz zum Dublin Core von den Suchmaschinen Google und Yahoo unterstützt. Für das Vokabular existieren Erweiterungen für die gängigsten Shopsysteme. Im Gegensatz zu anderen Ontologien wird GoodRelations nicht als XML Dokument, sondern als HTML Markup eingebunden. Beschrieben werden mit diesem Vokabular Produkte und Firmen. Als zusätzliches Feature bietet es eine standardisierte Angabe zu den Öffnungszeiten eines Geschäfts an. Unternehmen erhoffen sich durch diese suchmaschinen-optimierten Angaben eine bessere Auflistung bei den Suchergebnissen [Hepp 2008].

3.3.5 RDF-Frameworks

RDF-Frameworks erleichtern die Arbeit mit RDF-Graphen und bieten häufig viele Funktionen zum Editieren. Weiterführend werden drei Frameworks kurz vorgestellt.

Jena

Jena[13] ist ein Open-Source Framework zur Abfrage und Editierung von RDF-Graphen. Aufgrund der Realisierung in Java ist Jena plattformunabhängig. Die Schnittstelle unterstützt den Import oder Export von RDF- und XML-Dokumenten, relationalen Datenbanken, N3 und Dokumenten in der RDF-Serialisierungssprache Turtle. Jena arbeitet auf Modellen im Speicher oder in Dokumenten/Datenbanken und bietet als Abfragesprache SPARQL an.

RDF API for PHP (RAP)

Das PHP Framework RAP[14] ist genau wie Jena ein Framework für RDF-Graphen. RAP ist jedoch in der Skriptsprache PHP geschrieben und bietet einige zusätzliche Funktionen wie RSS Parser oder eine Schnittstelle für Ontologie-Modelle.

ARC2

ARC2[15] ist ebenso wie RAP ein PHP Framework für semantische Webanwendungen. Standardmäßig werden bei ARC2 die Daten in eine MySQL Datenbank gespeichert und

[13] https://jena.apache.org/index.html
[14] http://wifo5-03.informatik.uni-mannheim.de/bizer/rdfapi/
[15] https://github.com/semsol/arc2/wiki

über eine Mapping-Datei referenziert. Für das Framework existieren einige Plugins, die die Funktionalität erweitern.

3.3.6 SPARQL

SPARQL ist eine Abfragesprache für RDF-Graphen und wird seit 2008 von der W3C für den Zugriff auf RDF-Daten empfohlen. Die Syntax ähnelt sehr der weit verbreiteten Abfragesprache SQL für relationale Datenbanken. Sie bietet die Möglichkeit erforderliche und optionale Graphenmuster zu suchen. Das Ergebnis der Abfrage kann eine Kollektion von RDF-Graphen sein. In SPARQL können Präfixe genutzt werden, um die Abfrage leserlicher zu machen. Beispiel 3.5 zeigt eine Abfrage, die alle Hauptstädte in Afrika in der DBPedia abfragt. Dazu wird in der ersten Zeile das Präfix *abc* definiert, was in den nachfolgenden Zeilen verwendet wird.

Beispiel 3.5: SPARQL Abfrage

```
PREFIX abc: <http://example.com/exampleOntology#>
 SELECT ?capital ?country
 WHERE {
    ?x abc:cityname ?capital ;
       abc:isCapitalOf ?y .
    ?y abc:countryname ?country ;
       abc:isInContinent abc:Africa .
 }
```

Folgende Abfrage-Engines werden unterstützt:

- ARQ (Engine innerhalb vom Jena-Framework),
- D2R Server (Wrapper für relationale Datenbanken),
- ARC (Engine für PHP) sowie einige mehr.

Nach den Grundlagen zur Verwendung von semantischen Informationen befasst sich das folgende Kapitel mit dem impliziten und expliziten Benutzerfeedback.

4 Implizite und explizite Bewertung

Als implizite Bewertung bezeichnet man das Interpretieren von Benutzerverhalten ohne explizites Feedback des Benutzers [Joachims et al. 2007]. Dazu wird das Verhalten einer Person aufgezeichnet und später manuell oder automatisiert ausgewertet. Die Bandbreite der Registrierung und Auswertung von Verhalten einer Person reicht von einer Videoaufzeichnung im Kaufhaus mit anschließender Analyse, wie lange vor welchem Regal gestanden wurde und welches Produkt in den Warenkorb gelegt wurde, bis zur Nutzung der Verkehrsdaten eines Mobiltelefons (siehe Abschnitt 4.3). Implizit erhobene Benutzerinformationen zu verarbeiten birgt Risiken, die im Unterkapitel 4.4 vorgestellt werden.
Im ersten Abschnitt wird kurz die explizite Bewertung vorgestellt. Die nachfolgenden Abschnitte beschäftigen sich mit den Themen, warum man implizite Bewertung benötigt (4.2), zeigen deren Verwendungsmöglichkeiten (4.3) und Gefahren (4.4) auf. Abschließend werden verschiedene Parameter der impliziten Bewertung erläutert (4.5).

4.1 Explizite Bewertung

Die explizite Bewertung ist das ausdrückliche Feedback einer Person auf ein Ereignis, Tätigkeit oder Objekt [Pazzani & Billsus 2007]. Die Anwendungsdomänen dazu sind sehr vielfältig und gehen von Meinungsumfragen auf der Straße zu Evaluierungen auf Webseiten. Insbesondere bei Webseiten mit sozialem Bezug wird auf die persönliche Einschätzung Wert gelegt. Dazu zählen z.B. die Transaktionsbewertung bei Ebay, der Facebook Like-Button und das Bewerten eines Videoclips auf YouTube. Meist werden dadurch die Interessen des Unternehmens (Produktevaluierung) mit dem Mitteilungsdrang Einzelner verbunden. Dass Beiträge jedoch nicht immer der Wahrheit entsprechen, lässt sich z.B. an den Amazon Produktrezensionen über das Wenger Schweizer Offiziersmesser Giant[1] zeigen. Dort werden dem Taschenmesser teils sehr humoristische Funktionen angedichtet. Solche subjektive Einschätzungen sollten nicht der alleinige Bestandteil ei-

[1] http://www.amazon.de/product-reviews/B000R0JDSI (abgerufen am 10. Juli 2013)

ner Produktbewertung sein, da eine explizite Taxierung immer eine bewusste Bewertung ist. Somit ist es nicht auszuschließen, dass es sich um eine wohlüberlegte Manipulation handelt. Besser wäre eine Kombination von impliziten und expliziten Einschätzungen, um das Risiko einer Manipulation zu minimieren.
Explizites Feedback kann z.B. über ein Formular mit binären Antworten (positiv/negativ) ermöglicht werden, welches neben dem auswählbaren Element positioniert wird. Ist das Feedback negativ, wäre eine zusätzliche Antwortmöglichkeit (Element schon bekannt, Element nicht interessant, etc.) für weitere Vorschläge von Vorteil.

4.2 Vergleich implizite und explizite Bewertung

Recommender Systems brauchen für die Berechnung eines Produktvorschlages möglichst genaue Informationen über den Benutzer. Meist reichen dazu die explizit erhobenen Daten nicht aus. Dies kann z.B. an der Qualität, Quantität oder Objektivität der Informationen liegen [Pazzani & Billsus 2007, Oard, Kim et al. 1998]. Des Weiteren ist die Benutzersensibilität im Bezug auf Datenschutz in Zeiten von Facebook und Google stark gestiegen. Viele Benutzer beantworten ungern einem ihnen fremden Unternehmen persönliche Fragen [Hu et al. 2008]. Erschwerend kommt der Zeitaufwand und die eigene Sorgfalt hinzu, die beim Ausfüllen eines solchen Formulars entsteht. [Yun & Trumbo 2000] haben gezeigt, dass Personen, die einen Fragebogen ausfüllen, bisweilen falsche Angaben machen. Um diese Fehlerquelle zu minimieren, bietet es sich an, die implizite Bewertung in die Bewertung einfließen zu lassen. Zusätzlich ermöglicht die implizite Bewertung eine neutrale und objektive Abschätzung des Benutzers, die die Subjektivität bei der Selbstbewertung meist nicht zulässt [Van Meteren & Van Someren 2000]. Jedoch lässt sich negatives Feedback mit impliziter Bewertung nur schwierig messen. Dazu bietet sich eher eine explizite Bewertung an.

4.3 Beispiele für implizite Bewertung

Die Schutzgemeinschaft für allgemeine Kreditsicherung (Schufa) ist eine nicht-staatliche Auskunftei, die Informationen zu der Kreditwürdigkeit von Privatpersonen sammelt. Das umschließt Kontaktdaten, Handy-/Leasingverträge, Bankkonten, ausstehende Zahlungsaufforderungen und anderes mehr. Zur Zeit[2] verfügt die Schufa über 514 Millionen

[2]Stand Mai 2013

Datensätze von 66,2 Millionen Personen[3]. Aus diesen Daten berechnen komplexe Algorithmen die Kreditwürdigkeit einer Person. Dies geschieht ausschließlich über implikative Bewertung ohne Zutun der Person. Mehr als 90% der eingetragenen Personen erhalten eine positive Schufa-Auskunft.
Die Funkzellenabfrage (FZA) wird zum Zweck der Strafverfolgung ausgeführt und stellt fest, welche Mobiltelefone zu einem Zeitpunkt in einer bestimmten Funkzelle eingewählt waren. Diese personenbezogene Datenabfrage kann nur von einem Richter genehmigt werden und wird meist für verdeckte Ermittlungen genutzt. In den letzten Jahren wurde häufig diskutiert, unter welchen Umständen eine FZA gerechtfertigt ist. Bei einer Abfrage ist nicht auszuschließen, dass auch unbeteiligte Personen mit ihren Mobiltelefonen in der Funkzelle angemeldet sind.
Neben Google mit seinem Kartendienst Google Maps (siehe Abbildung 4.1) bietet Apple (siehe Abbildung 4.2) auch in seinen Karten Stauinformationen an. Diese Informationen stammen von den Verkehrsdaten der mobilen Endgeräte der anonymisierten Kartennutzer. Darin enthalten sind die GPS-Position, WLAN-Zugänge in der Umgebung (zur genaueren Positionsbestimmung) und die Bewegungsgeschwindigkeit. Aus diesen Daten werden aktuelle und zukünftige Stauinformationen berechnet. Je nach Verkehrsfluss werden bei Google Maps die Straßen grün, gelb oder rot markiert. Bei Apple werden neuralgische Punkte mit einer rot-gestrichelten Linie signalisiert.
Die Online-Handelsplattform Amazon ist bekannt für ihre prominenten Produktvorschläge (Kunden, die Produkt A kauften, haben auch Produkt B gekauft). Ein weiteres Feature von Amazon ist die Produktempfehlung per Email, nachdem ein Benutzer sich ein Produkt ansah, aber nicht kaufte. Die Grundlage für diese zeitversetzte Produkterinnerung stammt aus der impliziten Bewertung [Schafer et al. 1999]. Schaut sich ein Kunde ein Produkt an und kauft es nicht, ist er ein möglicher zukünftiger Käufer. Durch implizite Faktoren, wie z.B. Verweildauer auf der Produktseite, ist es möglich zu analysieren, ob das angezeigte Produkt den potenziellen Kunden interessiert oder er aus Versehen ein ihm uninteressantes Produkt angeklickt hat [Fox et al. 2005].

4.4 Gefahren der impliziten Bewertung

Neben den genannten positiven Anwendungsmöglichkeiten der impliziten Bewertung birgt diese jedoch auch einige Risiken, auf die im Folgenden eingegangen werden soll.

[3] http://www.schufa.de/de/private/unternehmen/zahlendatenfakten/zahlendatenfakten.jsp (abgerufen am 25. Juli 2013)

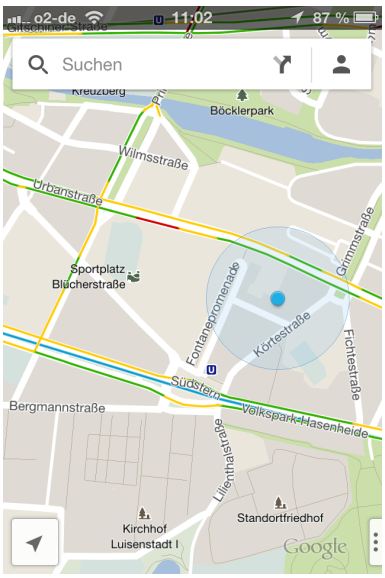


Abbildung 4.1: Google Maps im iPhone 4 (eigene Darstellung)

Abbildung 4.2: Apple Maps im iPhone 4 (eigene Darstellung)

Bei explizitem Feedback hat der Benutzer immer die Möglichkeit keine Rückmeldung zu geben, indem er z.B. die Umfrage nicht ausfüllt oder eine Antwort zu verweigern. Jenes ist bei der impliziten Bewertung selten möglich, da der Benutzer keinen Einfluss auf die Beurteilung hat und in vielen Fällen auch nicht registriert, wann eine Beurteilung stattfindet. Das trifft z.B. auf die Schufa zu. Bei Eröffnung eines Bankkontos wird in den meisten Fällen eine Schufa-Anfrage getätigt. Kunden, die regelmäßig Schulden machen, diese aber auch tilgen, sind bei den Banken sehr beliebt. Die kreditvergebende Banken verdienen an den Darlehenszinsen. Nimmt eine Person nie ein Darlehen auf, ist es für die Bank schwerer einzuschätzen, ob diese Person pünktlich ihr Darlehen tilgt. Ein weiteres Beispiel für unbemerktes Bewerten ist der Einfluss des Wohnortes beim Schufa-Scoring. Die Wohnlage kann positiv oder negativ auf den Score einwirken, wobei die Wohnlage nicht ausschließlich den Score beeinflussen darf[4]. In die Kritik geraten war die Schufa, nachdem sie eine Kooperation mit dem Hasso-Plattner-Institut eingegangen war, um soziale Netzwerke als Quelle für die Bewertung zu erforschen. Diese Vereinbarung wurde kurz danach gekündigt[5]. Insgesamt ist das Socring der Schufa kritisch zu sehen, da die Berechnung nicht offengelegt wird und nicht alle Einflüsse einzusehen sind[6].

[4]nach § 28b Abs.3 BDSG in der Fassung vom 20.12.1990
[5]Frankfurter Allgemeine: ‘Schufa will Facebook-Profile auswerten‘, unter: http://www.faz.net/aktuell/wirtschaft/pruefung-der-kreditwuerdigkeit-schufa-will-facebook-profile-auswerten-11776537.html (abgerufen am 12.06.2013)
[6]Urteil Landgericht Wiesbaden vom 01.12.2011 - Aktenzeichen: 8 O 100/11

Darüber hinaus ist ein Missbrauch von den gesammelten Daten nicht auszuschließen. Der Missbrauch kann vom Anlegen eines Persönlichkeitsprofils bis hin zum Verkauf von personenbezogenen Datensätzen reichen. Als Beispiel lässt sich hier die Rasterfahndung benennen. Dort werden verschiedene Datenquellen wie z.B. Melderegister, Handelsregister, Krankenkassen geprüft. Einer der Kritikpunkte ist dort die aufgehobene Unschuldsvermutung, da die sich im Raster befindlichen Personen als verdächtig gelten. Eine weitere Unwägbarkeit ist ein falsch gedeutetes Benutzerinteresse auf Basis von fehlerhaftem Feedback. Die Folgen sind u.a. nicht zutreffende Produktvorschläge, unzufriedene Benutzer und Verminderung der Elementaufrufe. Daraus kann ein Kunden-, Image- und Umsatzverlust für den Anbieter der Empfehlungen resultieren.

4.5 Merkmale in der implizite Bewertung

Wie eingangs schon erwähnt, können charakteristische Ausprägungen dazu dienen, Elemente (z.B. Produkte) und Benutzer beschreibbar und vergleichbar zu machen. Dazu gibt es beispielsweise die Möglichkeit Elemente anhand der Merkmale der Benutzerzielgruppe zu charakterisieren. So kann etwa dem Produkt iPhone 5 das Merkmal *internetaffin* zugeordnet werden, da es sich um ein Smartphone mit internetaffiner Zielgruppe handelt. Durch das Benutzerverhalten können verschiedene Merkmale festgestellt werden, die im Weiteren vorgestellt werden. Die Ausprägungen wurden von einer Psychologin für die Jaduda GmbH entwickelt.

Sozio-demographische Ausprägungen

Sozio-demographische Daten beschreiben die unterschiedlichen Lebensumstände von Personen und bieten die Möglichkeit die Zielgruppe zu kategorisieren. Die Daten umfassen u.a. Geschlecht, Alter, Haustiere, Haus/Wohnung, berufliche Tätigkeit, Religion, Haushaltsgröße, Kinder, Familienstand, Schulabschluss, Einkommen und politische Einstellung.

Psychographische Ausprägungen

Der psychographische Bereich umfasst Informationen, die z.B. Essen, Aktivitäten, Interessen, Lifestyle, ökologisches, gesundheitliches, soziales Bewusstsein, Risikobereitschaft, Sicherheitsbedürfnis, Aufgeschlossenheit, Zufriedenstellung umschließt. Diese Informationen werden meist explizit erfasst, da selten das Benutzerverhalten für diese Annahmen

ausreicht.

Einkaufsverhaltensbezogene Ausprägungen

Ein wichtige Rubrik im eCommerce sind die einkaufsverhaltensbezogene Ausprägungen. Diese beinhalten u.a. Einkaufverhalten und -affinität sowie das Bewusstsein für Preis, Marken, Qualität und Tradition. Zu beachten ist dabei, dass sich die Ausprägungen zwischen Ladengeschäften und Onlineshops stark unterscheiden können. Mangelndes Vertrauen in Produktqualität, fehlende Anprobe und die Versanddauer können sich negativ für den Onlinebereich auswirken. Die zeitliche Unabhängigkeit und eine größere Angebotsvielfalt sprechen u.a. für den eCommerce.

Ausprägungen der mobilen Endgeräte

Der Begriff mobile Endgeräte umspannt die Gruppe der Mobiltelefone, wie etwa Smartphones und den Bereich der Notebooks. Zusätzlich werden auch MP3-Player und GPS-Geräte mit diesem Begriff verbunden, auf die aber nicht weiter eingegangen wird, da nur Geräte mit Mobilfunkdienst und Webbrowser analysiert werden.
Mittels externen Bibliotheken, wie z.B. Wurfl[7], kann neben dem Markennamen und Typ des Endgeräts auch der Browser und das Betriebssystem ermittelt werden. Des Weiteren liefert der Browser auch Angaben zur IP-Adresse des Geräts und der gewünschten Sprache. Anhand der IP-Adresse kann auf das Land geschlossen werden, wo sich der Benutzer gerade aufhält.

Ausprägungen der Benutzerinteraktion

Das Verhalten des Benutzers auf einer Website zu registrieren und analysieren ist ein wichtiger Schritt, um zu verstehen, was den Benutzer interessiert. Der leichteste Schritt ist die Aufzeichnung, was der Benutzer **angeklickt** hat. Dazu können die observierten Elemente z.B. mit dem Javascript-Event *onClick* verknüpft werden, was eine Selektion erfasst. Des Weiteren kann die **Verweildauer** genutzt werden. Über diese kann ein gewisses Interesse prognostiziert werden. Jedoch ist die Lesezeit nicht allein ausschlaggebend für das Interesse an dem Seiteninhalt [Kelly & Belkin 2004]. Des Weiteren unterstützen einige Browser keine Asynchronous JavaScript and XML (Ajax)-Requests[8], bevor sie die Seite verlassen, was eine Terminierung der Dauer erschwert. Eine weite-

[7] http://wurfl.sourceforge.net/
[8] Ein Ajax-Request ist eine asynchrone Übertragung vom Browser ohne die Seite neu zu laden.

re Verhaltenserfassung ist die **Scroll Detection**. Hier wird mittels einer Javascript-Bibliothek festgestellt, wohin der Benutzer den Bildschirminhalt verschoben hat und welche Elemente für ihn sichtbar waren.
Analysiert man die Betrachtungszeit, die Scroll Detection und die bisher registrierten Interessen des Benutzers über einen längeren Zeitraum, ist eine Erfassung der Entscheidungsfreude möglich. Hat der Benutzer z.B. sich ein Element nur kurz angesehen, was seinen Interessen entsprochen hat und wurde es daraufhin angeklickt, spricht das für eine schnelle Entscheidungsfreude [Joachims et al. 2007].
Scrollt der Benutzer an das Ende der Vorschläge, könnte man das anhand der Scroll Detection feststellen und weitere Vorschläge einblenden lassen. Abhängig von dem weiteren Verhalten ließe sich feststellen, ob die vorherigen Vorschläge nicht signifikant für den Benutzer waren.

5 Methode

5.1 Grundlagen

Für das Symfony 2 Framework wird ein Webserver mit PHP 5 benötigt. Hier bietet es sich an den weit verbreiteten Webserver Apache zu verwenden. Zum dauerhaften Speichern der Daten wird ein Datenbankserver genutzt. Der Datenbankabstraktionslayer Doctrine ist im Symfony 2 Framework integriert und unterstützt alle gängigen relationalen Datenbanken (MySQL, PostgreSQL und Microsoft SQL), sowie die NoSQL-Datenbank MongoDB.
Das Symfony 2 Framework ist ein MVC Framework für Webentwicklung in der Skriptsprache PHP. MVC ist ein Architekturmuster, das sich in Datenmodell (**Model**), Benutzeroberfläche (**View**) und Ablaufsteuerung (**Controller**) aufteilt. Durch die strikte Trennung der drei Bereiche ist eine flexible, dynamische und abstrakte Entwicklung möglich.
Symfony 2 zeichnet sich vor allem durch seine Performance und Erweiterbarkeit aus. Viele Funktionen lassen sich durch externe Bundles[1] hinzufügen. Ein eigenständiges Recommender-Bundle bereitzustellen, bietet die Möglichkeit den Programmcode Open-Source zu verteilen, Feedback zu erhalten und Updates besser zu verteilen.
Der Composer ist ein Manager für Abhängigkeiten in PHP. Symfony 2 nutzt den Composer zur Installation und Aktualisierung vom Framework und Bundles. In einer externen Datei wird festgelegt, welche Programmversion und Bundles installiert bzw. aktuell gehalten werden sollen. Mit jedem Composer-Aufruf werden außerdem alle Abhängigkeiten, die ein Bundle besitzt, aufgelöst.
Für das Recommender-Bundle bestehen für die Umsetzung folgende Anforderungen:

1. Push information model - dem Benutzer werden Produktvorschläge anhand seines Benutzerprofils generiert. Gegensätzlich davon ist das pull information model, wo der Benutzer explizit eine Anfrage stellt (z.B. Suchmaschine) [Gnasa et al. 2005].

[1] Ein Symfony 2 Bundle ist mit einem Plugin vergleichbar und stellt eine bestimmte Funktionalität bereit.

2. Active filtering system [Boutilier et al. 2003] - dem Benutzer werden nur Produkte vorgeschlagen, die er noch nicht abgeschlossen hat und die zu seinem Profil passen. Viele Werbekampagnen sind auf bestimmte Personengruppen zugeschnitten. Um zu erreichen, dass auch nur Personen aus dieser Gruppe diese Werbekampagne zu Gesicht bekommen, werden Filter eingesetzt. Die Filter beinhalten verschiedene Parameter, wie etwa demographische, soziale, geographische oder technische Restriktionen (siehe Abschnitt 4.5). Anhand dieser Werte kann bestimmt werden, ob der Benutzer in der Zielgruppe der Werbekampagne liegt und ihm die Kampagne angeboten werden kann.
   Ein weiterer wichtiger Punkt in der Kampagnenfilterung ist der Leadfilter. Für die meisten Werbekampagnen wird nach Abschluss für den Benutzer ein Lead erstellt. Ein Lead enthält die Kontaktdaten einer Person, die sich für das werbende Unternehmen bzw. beworbene Produkt interessiert. Dieser Lead wird nach Abschluss der Kampagne mit Einverständnis der Person an das werbende Unternehmen weitergeleitet. Um Leadduplikate zu vermeiden, werden Kampagnen, für die ein Lead bereits existiert, herausgefiltert.

3. Product-based filtering system [Resnick et al. 2004] - Beide Recommender Systems erhalten für ihre Berechnung eine Kollektion von Produkten, die dem Benutzer vorgeschlagen werden können, und geben die Produkte sortiert wieder, je nachdem wie gut sie zu dem Benutzer passen. Berechnet wird dabei wie ähnlich die Ausprägungen der bereits abgeschlossenen Kampagnen zu den noch möglichen Kampagnen ist. Gegensätzlich dazu berechnet ein User-centred filtering system [Xin et al. 2005] die Similarität zwischen Benutzerprofilen.

5.2 Datenbank

Aufgabe dieses entwickelten Bundles ist eine Verbindung zwischen Produkten und Benutzern herzustellen und zu berechnen, welche Produkte sich für einen Benutzer eignen. Um dies möglichst einfach zu vollziehen besitzt das Bundle zwei Schnittstellen.
Das **Produkt**-Modell ist eine systeminterne Entität, die an eine systemübergreifende Entität über eine One-to-One-Relation angegliedert werden muss. Im Testsystem geschieht das über die SubCampaign-Entität, die über das Feld *product_id* ein Produkt referenziert. Die *SubCampaign* ist das Datenmodell der vorzuschlagenden Werbekampagne in diesem Anwendungsfall.
Die zweite Schnittstelle ist das **Profile**-Interface. Das Interface besitzt zwei zu imple-

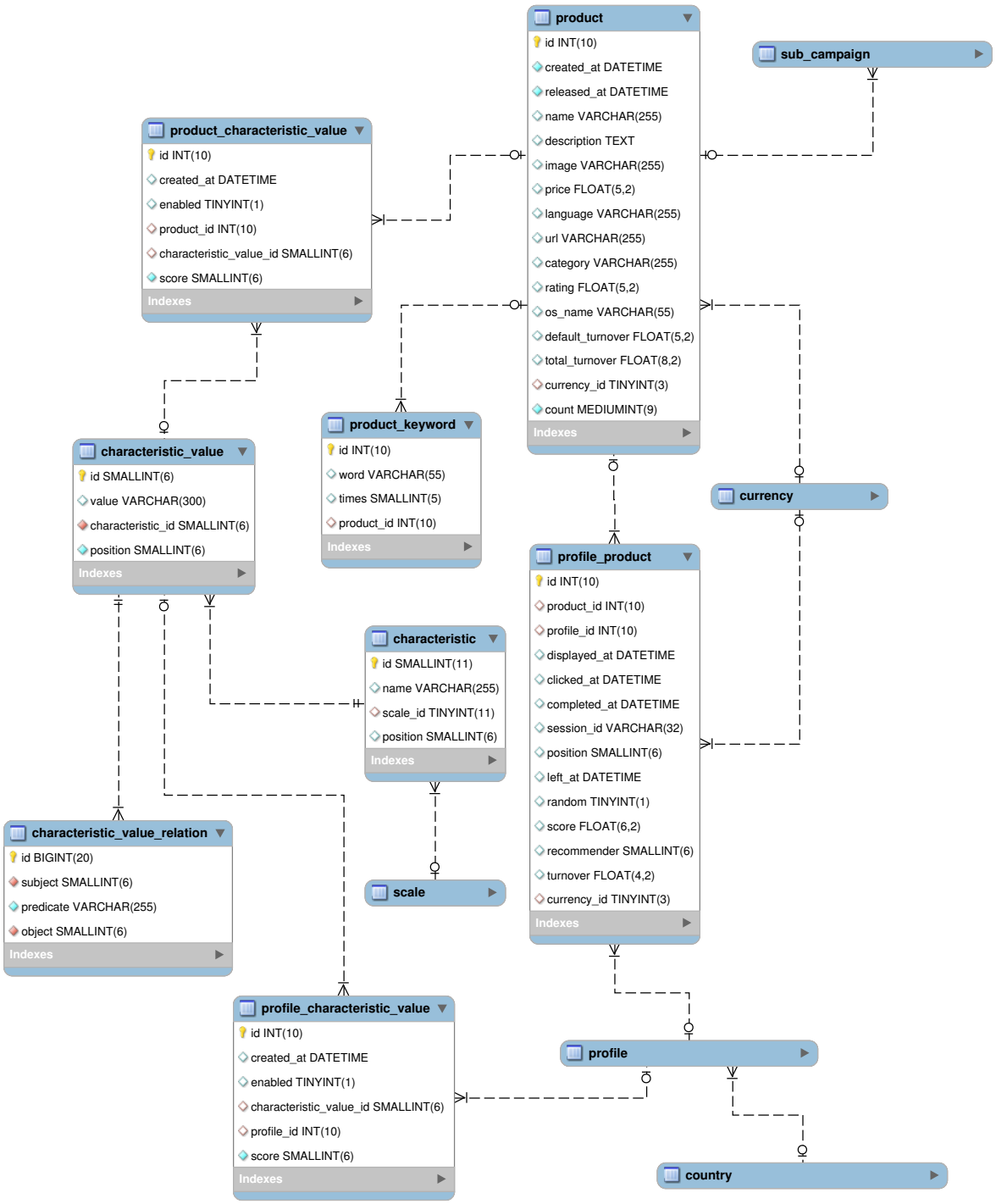


Abbildung 5.1: ER Modell

mentierenden Methoden, die die Vorlage zur Rückgabe des Profil-Identifiers und der charakteristischen Ausprägungen bereitstellen. Ist dies geschehen, müssen die vorhandenen Produkte anhand ihrer Ausprägungen charakterisiert werden. In der Tabelle *characteristic_value* gibt es 234 Ausprägungen, die jedem Produkt und Benutzer zugeordnet werden. Jede zugeordnete Ausprägung besitzt außerdem einen Wert zwischen -100 und +100. Der Wert gibt an, wie genau die Ausprägung dem referenzierten Objekt entspricht. Initial sind alle Werte auf 0 - Neutralität - gesetzt.
In Abbildung 5.1 ist ein Auszug des verwendeten Datenschemas zu sehen. Die Hauptbestandteile des Schemas gliedern sich in drei Bereiche auf:

- Die *profile*-Tabelle beschreibt das Benutzerprofil mit Attributen, wie z.B. Vorname, Nachname, Email-Adresse und Geburtsdatum. Zum Profil gehören die Tabellen *country*, *profile_characteristic_value*, *profile_characteristic_value_raw* und *profile_product*. Die Tabelle *country* gibt das Land an, in dem sich der Benutzers zur Zeit aufhält. Die Tabelle *profile_characteristic_value* enthält die aktuellen charakteristischen Ausprägungen der Benutzer und *profile_characteristic_value_raw* speichert alle jemals zugeordneten Ausprägungen eines Benutzers. Die Tabelle *profile_product* zeichnet alle getätigten Produktvorschläge auf.

- In der *product*-Tabelle werden die einzelnen Produkte gespeichert. Ein Produkt besitzt u.a. einen Namen, einen Beschreibungstext, ein Bild, den summierten produzierten Umsatz und den durchschnittlichen Umsatz. Der durchschnittliche Umsatz wird explizit mit dem Produkt verknüpft, da er im Scoring (siehe Abschnitt 5.4.4) verwendet wird und ein Produkt verschiedene Umsätze haben kann. Je nach Zielgruppenausrichtung der Werbekampagne kann der Umsatz variieren. Um trotzdem die Abstraktion zu gewährleisten und die Komplexität beim Scoring gering zu halten, wird ein durchschnittlicher Umsatz dem Produkt zugewiesen. Die Produktattribute *default_turnover* und *total_turnover* werden auf eine Währung umgerechnet. Die verwendete Währung kann von Kampagne zu Kampagne unterschiedlich sein. Durch die zugehörige *currency*-Tabelle wird die verwendete Währung bestimmt.
  Wie ein Benutzerprofil besitzt auch ein Produkt charakteristische Ausprägungen. Diese werden in Tabelle *product_characteristic_value* gespeichert. Die Ausprägungen eines Produktes werden nicht dynamisch angepasst, weswegen die Verwendung einer *product_characteristic_value_raw*-Tabelle entfällt. Je nach Produkt gibt es einen Beschreibungstext, anhand dessen eine Charakterisierung auf Basis

von Schlagworten durchgeführt wird. Die vorkommende Anzahl der Wortstammes, der Wortstamm selbst und das zugehörige Produkt eines jeden Schlagwortes werden in der Tabelle *product_keyword* erfasst.

- Die *characteristic_value_relation*-Tabelle ist die wichtigste Komponente des Semantic Recommenders, da sie seine Wissensbasis ist. Hier werden Verbindungen zwischen charakteristischen Ausprägungen gespeichert.

5.3 Backend

Zur Administration der Produkte wurde im Backend eine Maske zur Charakterisierung der einzelnen Produkte erstellt (Abbildung 5.2). Über Auswahlfelder lassen sich die Ausprägungen selektieren und mit dem danebenliegenden Reglern die Gewichtungen anpassen. Über die Checkbox links neben der Ausprägung lassen sich bestimmte Werte aktivieren bzw. deaktivieren. Darüber hinaus besteht die Möglichkeit den durchschnittlichen Umsatz des Produktes und die verwendete Währung anzupassen. Das Look-and-feel der Seite wurde mit dem Frontend Framework Twitter Bootstrap[2] und jQuery UI[3] realisiert und orientiert sich an gängigsten Webstandards.

5.4 Recommender Bundle

Das Testsystem ruft das Recommender Bundle im *SubCampaignDetector* auf. Der *SubCampaignDetector* bekommt beim Aufruf die zu verwendende Recommender Engine (Content-based oder Semantic Recommender) übergeben und leitet am Ende die fertigen Produktvorschläge an den zuständigen Controller. Zuerst wird geprüft, ob noch Produktvorschläge in der Session (siehe Kapitel 5.4.1) gespeichert sind. Ist dies nicht der Fall, werden alle Produkte gesucht, die dem Benutzer angeboten werden können. Je nach Produkt gibt es verschiedene Restriktionen, die einen Benutzer ausschließen können. Die Ursache kann z.B. eine Verschiedenheit in der Zielgruppe sein oder das Produkt wurde bereits ohne Erfolg vorgeschlagen.
Die verbleibenden Produkte werden vor der Vorschlagerstellung an die Recommender Engine gesendet. Die Vorfilterung der Produkte verkürzt die Laufzeit bereits bei 1000 Produkten erheblich. Da jedes Produkt initial 234 Merkmale besitzt, ist die Einschrän-

[2]http://twitter.github.io/bootstrap/
[3]http://jqueryui.com/

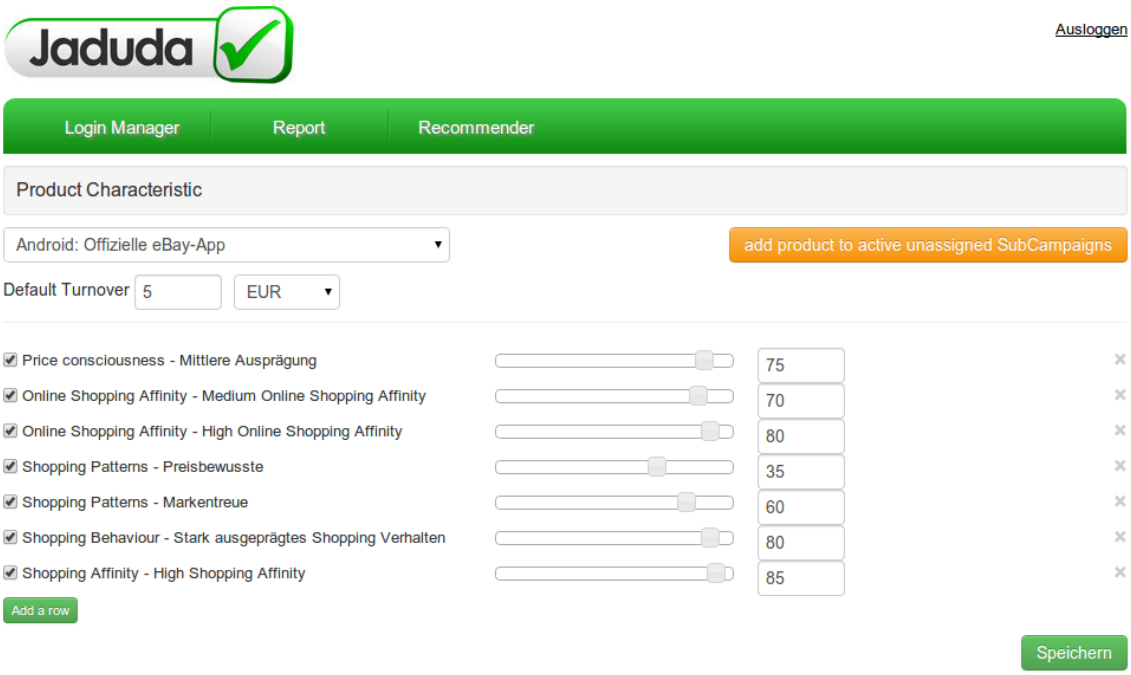


Abbildung 5.2: Zuordnung der Produktcharakteristika

kung der Produkte sehr sinnvoll. Anschließend wird die Methode *getProductIds* der Recommender Engine aufgerufen.

5.4.1 Abstract Recommender

Der Abstract Recommender ist eine abstrakte Klasse, von der die beiden Recommender Engines erben, siehe Abbildung 5.3. Die Klasse stellt die ausführbaren Methoden *getProductIds* und *getEngineId* bereit, sowie einige andere zu implementierende Methoden. Die Methode *getProductIds* prüft, ob in der Benutzersession Produktvorschläge gespeichert sind oder ruft andernfalls die Methode *calculateProductIds* auf, um Produktvorschläge zu generieren und diese in der Benutzersession zu speichern. Abschließend werden standardmäßig die Top5 Produkte zurückgegeben, die noch nicht abgeschlossen wurden. Die Methode *getEngineId* wird für die spätere Evaluation benötigt, um festzustellen, welcher Produktvorschlag welchem Recommender entstammt. Pro Benutzersession werden nur einmal die Produktvorschläge berechnet und für die weitere Verwendung in der Session gespeichert. Dieser Vorgang hat zwei Vorteile: Für jeden weiteren Vorschlag können die in der Session gespeicherten Produkte verwendet werden, somit verkürzt sich die **Laufzeit**. Außerdem wird verhindert, dass dem Benutzer andere Produkte vorgeschlagen werden, wenn er die Vorschlagseite aktualisiert. Es ist nicht unwahrscheinlich, dass

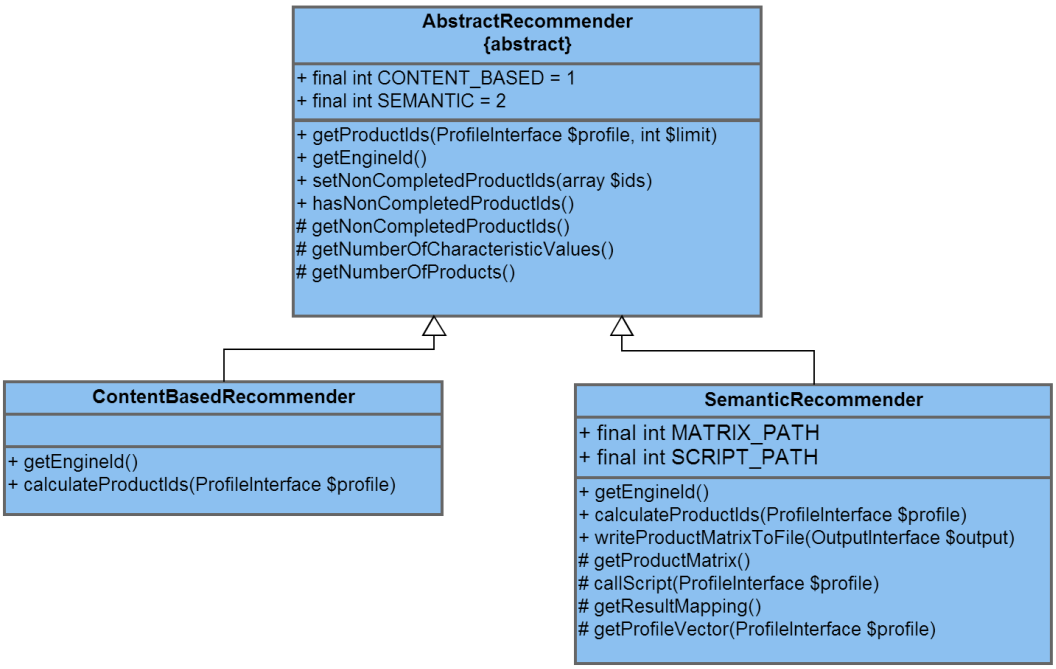


Abbildung 5.3: UML Diagramm der Recommender Engine

Produkte innerhalb kurzer Zeit Feedback erhalten, was deren Empfehlung verändern kann. Mit Hilfe der Session wird eine **Vorschlagpermanenz** gewährleistet.

5.4.2 Content-based Recommender Engine

Der Content-based Recommender ist sehr kompakt programmiert. Eine SQL-Abfrage sucht Verbindungen zwischen einem Profil und einer gegebenen Menge an Produkten. Hier wird eine native Abfrage statt dem üblichen Doctrine Querybuilder verwendet, da Doctrine bei der Tabellenreferenzierung nur Verweise auf den Primärschlüssel und keine Fremdschlüssel-Joins zulässt. Somit wäre keine direkte Verknüpfung zwischen der *profile_characteristic_value* und der *product_characteristic_value*-Tabelle möglich, da hier jeweils der Fremdschlüssel *characteristic_value_id* referenziert wird.

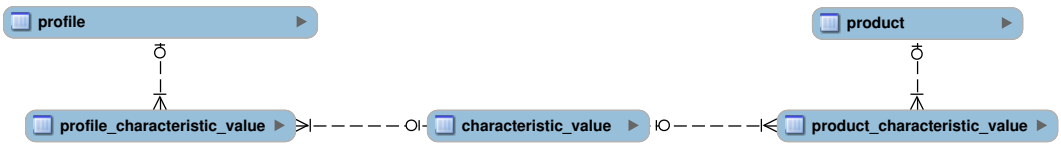


Abbildung 5.4: ER-Modell: Verbindung zwischen Profil und Produkt

Die Abfrage (siehe Beispiel 5.1) beginnt bei dem aufzurufenden Profil und referenziert

alle zugehörigen charakteristischen Ausprägungen (Abbildung 5.4). Als nächster Schritt werden die Produktausprägungen mit den Ausprägungen des Profils verknüpft. Anschließend werden die Produktausprägungen mit den Produkten verbunden. Zusätzlich zu den ProduktIds *pd.id* (Zeile 2) wird die Scoremultiplikation von Profil- und Produktausprägung summiert und als *productScore*, auf zwei Nachkommastellen gerundet, ausgegeben (Zeile 3-4). Der Produktscore verdeutlicht die Übereinstimmung der Profil- und Produktmerkmale. Jedes Merkmal kann einen ganzzahligen Wert zwischen -100 und +100 annehmen. Im Beispiel 5.1 wird die Abfrage für das Profil mit der Id 7000 (Zeile 15) und den ProduktIds 5,8,9,12 (Zeile 16) gestellt.

Beispiel 5.1: SQL Abfrage des Content-based Recommender

```
1  SELECT
2      pd.id,
3      ROUND(SUM(pdcv.score * pfcv.score) / 10000, 2)
4        AS productScore
5    FROM product pd
6      LEFT JOIN product_characteristic_value pdcv
7        ON pdcv.product_id = pd.id AND pdcv.enabled = 1
8      LEFT JOIN profile_characteristic_value pfcv
9        ON pfcv.characteristic_value_id =
10         pdcv.characteristic_value_id
11       AND pfcv.enabled = 1
12     LEFT JOIN profile pf
13       ON pfcv.profile_id = pf.id
14   WHERE
15     pf.id = 7000
16     AND pd.id IN (5, 8, 9, 12)
17   GROUP BY pd.id
18   ORDER BY productScore DESC
```

5.4.3 Semantic Recommender Engine

Der semantische Recommender nimmt den Aufbau des Content-based Recommender als Grundlage für seine Berechnungen und erweitert diese mit einer Wissensbasis. Anfänglich wurde eine Lösung mit einem externen C-Skript angestrebt. Die Idee basiert auf einer Skalarmultiplikation, bei der jeder Vektor die Ausprägungen eines Produkts bzw.

Profils darstellt. Ein Benchmark-Test zeigt, dass bei 1000 Produkten mit jeweils 234 Merkmalen die Sprache PHP durchschnittlich 850 ms, hingegen die systemnahe C Simulation nur 200 ms benötigt. Diese um Faktor 4 beschleunigte Berechnung rechtfertigt die Auslagerung. Um diese Berechnung weiter zu optimieren, wurden die Produktvektoren in einer temporären Datei ausgelagert und mittels Cronjob[4] regelmäßig aktualisiert. Der positive Nebeneffekt war eine Pufferung von Schwankungen im Produktscore.
Da die Skalarmultiplikation nur beschränkt die Wissensbasis zum Einsatz bringt, wird eine Lösung mit RDF-Tripeln angestrebt. Dazu werden Ähnlichkeiten zwischen den charakteristischen Merkmalen von Profil und Produkten gesucht, wie es der Content-based Recommender ebenfalls realisiert. Jedoch werden in der Wissensbasis Querverbindungen zwischen den Merkmalen gespeichert und für die Suche verwendet. Diese Querverbindungen werden über ein Frontend eingegeben und in einer relationalen Datenbank gespeichert.
Unter Zuhilfenahme der ARC2 Bibliothek wurde eine SPARQL-Abfrage auf der Datenbank ausgeführt. Da kein üblicher Triple-Speicher, sondern eine MySQL-Datenbank verwendet wurde, stellte eine Mapping-Datei die notwendige Verbindung her. Angesichts der Tatsache, dass das Bundle leicht zu implementieren und ohne zusätzliche Installationen verwendet werden soll, bot sich hier eine relationale Datenbank an, die bei Symfony 2 eine Grundvoraussetzung ist. Nach einigen Testläufen zeigte sich, dass die Bibliothek von ARC2 deutlich stabiler war als die von RAP, was ausschlaggebend für deren Benutzung im Bundle war.
In der Mapping-Datei (Beispiel 5.2) werden die RDF Abbildungen für Profil und Produkt beschrieben. Die D2RQ Mapping Engine bietet die Möglichkeit Tabellen einer relationalen Datenbank in eine virtuelle RDF-Repräsentation zu überführen. Für das Profil wird eine Klasse mit den Prädikaten *ProfileId* und *Products* definiert (Zeile 1-15). Das Prädikat *Products* verbindet - ähnlich wie das SQL-Query des Content-based Recommenders - das Profil mit seinen charakteristischen Ausprägungen. Jedoch werden diese Ausprägungen erst mit der Wissensbasis verknüpft (Zeile 20-25) und dann mit den Ausprägungen des Produkts verbunden (Zeile 26-30).

Beispiel 5.2: Auszug aus der D2RQ Mapping-Datei

```
1 map:Profile a d2rq:ClassMap;
2   d2rq:class :Profile;
3   d2rq:bNodeIdColumns "profile.id";
```

[4]Ein Cronjob ist ein Prozess im Betriebssystem, der regelmäßige Aufgaben ohne Zutun des Benutzers automatisiert ausführt.

```
  d2rq:dataStorage map:Database1;
  .
map:ProfileId a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:Profile;
  d2rq:property :ProfileId;
  d2rq:datatype xsd:integer;
  d2rq:column "profile.id";
  .
map:ProfileProducts a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:Profile;
  d2rq:property :Products;
  d2rq:refersToClassMap map:Product;
  d2rq:alias "characteristic_value_relation
    AS characteristic_value_relation2";
  d2rq:join
    "profile.id = profile_characteristic_value.profile_id";
  d2rq:join
    "profile_characteristic_value.characteristic_value_id =
    characteristic_value_relation.subject";
  d2rq:join
    "characteristic_value_relation.object =
    characteristic_value_relation2.subject";
  d2rq:join
    "characteristic_value_relation2.object =
    product_characteristic_value.characteristic_value_id";
  d2rq:join
    "product_characteristic_value.product_id = product.id";
  d2rq:condition "profile_characteristic_value.enabled = 1 AND
    profile_characteristic_value.score > 0 AND
    product_characteristic_value.enabled = 1 AND
    product_characteristic_value.score > 0";
```

Das SPARQL-Query in Beispiel 5.3 zeigt eine beispielhafte Abfrage des Semantic Recommenders. Die Variable *?profile* beschreibt die Profil-Klasse und wird durch deren eindeutiges Prädikat *:Products* identifiziert. Das Objekt *?profileId* liefert die Id des Profils. Das Prädikat *:Products* stellt alle mit dem Profil verknüpften Produkte bereit. Ab-

schließend wird das Ergebnis auf das Profil mit der Id 7000 und die Produkte 5,8,9,12 beschränkt.

Beispiel 5.3: SPARQL-Query des Semantic Recomender

```
SELECT DISTINCT ?productId
WHERE {
    ?profile :ProfileId ?profileId .
    ?profile :Products ?products .
    ?products :ProductId ?productId .
  FILTER (
    ?profileId = 7000 &&
    ?productId IN (5, 8, 9, 12)
  )
}
```

5.4.4 Product Scoring Manager

Um die Komplexität bei der Berechnung der Produktvorschläge in Grenzen zu halten und trotzdem zusätzliche Einflüsse geltend zu machen, wurde ein Scoring Manager verwendet. Der Manager erweitert den in der Recommender Engine berechneten Score mit aktuellen Erfahrungswerten. In die Prognose fließen dazu folgende Werte ein:

- Die Produkteinblendungen - **Impressions** genannt - geben wieder, wie vielen Personen das aktuelle Produkt angezeigt wurde. Die Impressions liefern einen Eindruck, wie oft das Produkt angezeigt wurde.
- Die Konvertierungsrate (**Conversion Rate**) bildet den prozentualen Anteil ab, wie viele von den Impressions die Produkt abgeschlossen haben. Ein Produktabschluss umschließt je nach Produkttyp ein Kauf des Produkts, die Übermittlung von Kontaktdaten und Bestätigung der Teilnahmebedingungen eines Gewinnspiels oder die vollständige Beantwortung einer Umfrage.
- Als dritter Faktor übt der durchschnittliche Umsatz des Produktes Einfluss in das Scoring aus (**Average Turnover**). Wie schon erwähnt, kann ein Produkt verschiedene Umsätze abhängig vom Filter besitzen. Dies kann z.B. eine Filterung nach Nationalität umfassen, um den Umsatz in entsprechender Landeswährung abzurechnen. Um den Average Turnover besser abschätzen zu können, wird jedem Produkt ein arithmetischer Mittelwert zugewiesen.

- Zusätzlich wurde ein **Impressions Barrier** installiert, die die *New Item* Problematik reduzieren soll. Neue Produkte besitzen noch keine stabile *Conversion Rate*, da zu Beginn jeder Kampagnenabschluss/-abbruch die Quote maßgeblich beeinflusst. Initial wird deshalb die *Conversion Rate* erst ab der 100. *Impression* in das Scoring einbezogen.

Der Score eines Produktes erhöht sich je besser die Ausprägungen zu dem Profil bzw. Produkt passen und je mehr Ausprägungen Profil und Produkt gemein haben. Abschließend gibt der Scoring Manager die Produkte sortiert anhand ihres neu berechneten Scores an die Recommender Engine zurück.

5.4.5 Prozessparameter

Ziel des Recommender Bundles ist es auch Daten für eine umfangreiche Auswertung zu sammeln. Dazu werden unter anderem die Produktvorschläge gespeichert (vgl. Kapitel 5.1). Zu jedem einzelnen Vorschlag gehören die Attribute: verwendete Recommender Engine, berechneter Score und Position in der Darstellungsreihenfolge. Ferner werden fünf optionale Datumsangaben gespeichert, die die Phasen vom Vorschlag bis zum Abschluss erfassen. In der ersten Phase wird die **Erstellung** eines Produktvorschlags registriert. In der zweiten Phase erfasst ein Tracker, ob das Produkt vollständig im Sichtbereich des Browsers **angezeigt** wurde. Realisiert wurde der Tracker in Javascript mit dem jQuery Framework und der Erweiterung Waypoints[5]. In der dritten Phase wird festgehalten, wann der Benutzer die Vorschlagseite **verlassen** hat. Die Phasen vier und fünf sind die interessantesten, da erhoben wird, ob der Benutzer Interesse am Produkt hat, es **angeklickt** wurde und ob er selbiges **abgeschlossen** hat.
Das Registrieren eines Abschlusses wird mit einem *Event Listener* umgesetzt. Symfony 2 bietet den Vorzug das *Observer Pattern* zu verwenden und bei Änderungen an Objekten diesen Vorgang für Beobachter zu signalisieren. Schließt ein Benutzer ein Produkt ab, wird dieser Vorgang über das Event *ProfileLeadCreation* signalisiert und aktiviert den *ProfileLeadCreationListener*, welcher die interne Verarbeitung dieses Events abschließt.

5.4.6 Management der Profilmerkmale

Die Verknüpfung von Merkmalen mit Profilen und Produkten ist ein Zuordnungskreislauf, vgl. Abbildung 5.5. Initial werden dem Produkt mehrere Merkmale anhand seiner vermuteten Zielgruppe zugewiesen. Bei einem Gewinnspiel für ein Smartphone z.B.

[5] http://imakewebthings.com/jquery-waypoints/

wird eine gewisse Technikaffinität, Markenbewusstsein und Risikointeresse vorausgesetzt. Schließt ein Benutzer ein Produkt ab, speichert der *Profile Characteristic Value Raw Manager* die Merkmale des Produkts als Rohdaten für den Benutzer ab. Diese Merkmale werden dem Benutzer zwar zugeordnet, aber noch nicht für das Profil verwendet. Ein Symfony 2 Kommando wird regelmäßig von einem Cronjob aufgerufen und weist dem Benutzerprofil die Merkmale seiner abgeschlossenen Produkte bis zu einem Zeitpunkt rückwirkend zu. Der Zeitraum kann flexibel gewählt werden und auch später verändert werden, da die Daten langfristig verfügbar sind. In regelmäßigen Abständen werden die Ausprägungen der Produkte mit den Ausprägungen der Benutzer verglichen, die dieses Produkt absolviert haben, und bei starken Abweichungen werden die Ausprägungen des Produkts angepasst. Dieser flexible Umgang mit Merkmalen ermöglicht eine dauerhafte Optimierung der Produktcharakterisierung sowie ein selbst lernendes Benutzerprofil.

Abbildung 5.5: Kreislauf Kampagne-Profil

6 Ergebnisse und Diskussion

Die Jaduda GmbH ist ein kleines Start-Up Unternehmen, was im Online und Performance Marketing tätig ist und im Jahr 2009 gegründet wurde. Die charakteristischen Ausprägungen wurden von einer Psychologin für die Jaduda GmbH entwickelt, um Produkte mit Merkmalen zu versehen. Die semantische Wissensbasis setzt diese Ausprägungen miteinander in Relation und ermöglicht so Querverbindungen zwischen Ausprägungen festzustellen.
Das Testsystem besitzt vier Prozessorkerne mit jeweils 2,8 GHz und einen Festspeicher von 2 TB. Der Arbeitsspeicher hat eine Größe von 24 GB. Als Betriebssystem wird Ubuntu Linux in der 64-Bit-Version ausgeführt.

6.1 Empfehlungssimulator

Bevor die beiden Empfehlungssysteme in den Live-Betrieb gingen, wurden sie ausgiebig getestet. Dafür wurden fünf Produkte und ebenso viele Profile angelegt. Jedes Profil und Produkt besaß eine feste Anzahl an charakteristischen Ausprägungen. Des Weiteren gab es jeweils ein Profil und ein Produkt, welche dieselben Merkmale hatten und dadurch optimal korrespondieren.
Die Simulation generiert eine bestimmte Anzahl an Produktvorschlägen für die angegebene Recommender Engine. Variabel eingestellt werden kann das Verhältnis der aufrufenden Profile sowie der Umsatz zu dem dazugehörigen Produkt. Fest definiert sind die Auswahlwahrscheinlichkeiten der simulierten Benutzer. Die Auswertung der Live-Datenbank ergab, dass **40 %** aller Benutzer einen der fünf Produktvorschläge abschlossen. Die Vorhersagen mit der ein Benutzer ein Produkt an Position i auswählte, orientierte sich an den Ergebnissen von [Joachims et al. 2007]. Die Werte wurden dahingehend angepasst, dass in den meisten Fällen auf einem mobilen Endgerät nur die ersten zwei Produkte angezeigt werden und erst durch scrollen die weiteren Vorschläge sichtbar werden. Somit werden die ersten beiden Vorschläge am häufigsten ausgewählt.
In Abbildung 6.1 sind die relativen Abhängigkeiten basierend auf der Produktpositi-

on zu sehen. Wenn beispielsweise ein Benutzer nicht Produkt 1 abschließt, was nach der Gegenwahrscheinlichkeit in **42 %** aller Fälle passiert, dann wird mit einer **51 %**igen Wahrscheinlichkeit Produkt 2 abgeschlossen. Daraus ergibt sich ebenfalls, dass sich Produkt 3 in fast 8 von 100 Fällen abgeschlossen wird.

$$Pr(Produkt_3) = (1 - 0.58) * (1 - 0.51) * 0.36$$

$$Pr(Produkt_3) = 0,074088$$

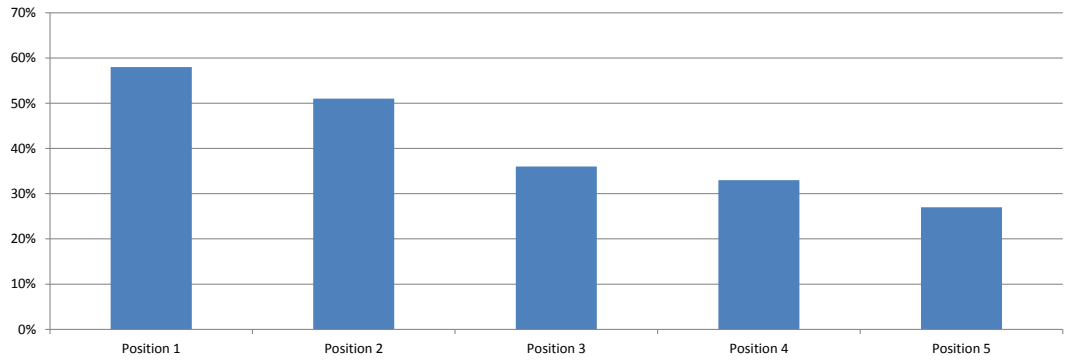


Abbildung 6.1: Klick-Wahrscheinlichkeit in Abhängigkeit der Position des Produktvorschlags

Für die Qualitätskontrolle der Empfehlungssysteme sind zwei Anforderungen zu überprüfen:

1. Dämpft die Impressions Barrier die Impressions-Schwankungen bei *New Item*-Produkten und erhöht dadurch den Umsatz?

2. Erhöht sich auch der Umsatz, wenn die Häufigkeit des Profils erhöht wird, dem das Produkt mit dem höchsten Umsatz zugeordnet ist?

Dazu werden jeweils 1000 Benutzer simuliert, deren Anzahl sich auf die fünf Profile unterteilt. Wie in Abbildung 6.2 zu entnehmen, wird jedem Produkt ein Umsatz zugewiesen. Dieser Umsatz blieb bei allen Simulationen unverändert. Zuerst wurde betrachtet, wie sich der Umsatz verändert, wenn der Impressions Barrier aktiv oder inaktiv ist. Hierfür wurden zwei Simulationen gestartet. Der Umsatz mit deaktivierter Barriere belief sich auf 1250 €. Mit aktivierter Barriere erhöhte sich der Umsatz um **2,4 %** auf 1280 €.

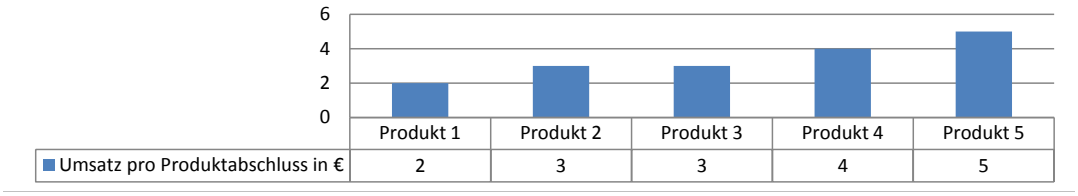


Abbildung 6.2: Produktumsatz in der Simulation

Zu jedem simulierten Profil gibt es ein optimales Produkt, welches bei Abschluss einen fest definierten Umsatz generiert. Wenn sich die Häufigkeit von Benutzerprofilen erhöht, die ein Produkt mit hohem Umsatz besitzen, dann sollte sich der Gesamtumsatz vergrößern. Dazu wurde die Häufigkeit der unterschiedlichen Profile variiert. Die Abbildung 6.3 zeigt die Gesamtumsätze der vier Durchläufe. Bei jedem Durchlauf erhielt das Lastprofil einen 40 %igen Anteil und die anderen Profile jeweils eine 15 %ige Verteilung. Die Auswertung ergab, dass sich der Umsatz wie erwartet steigerte je höher der Umsatz des Lastprofils war.

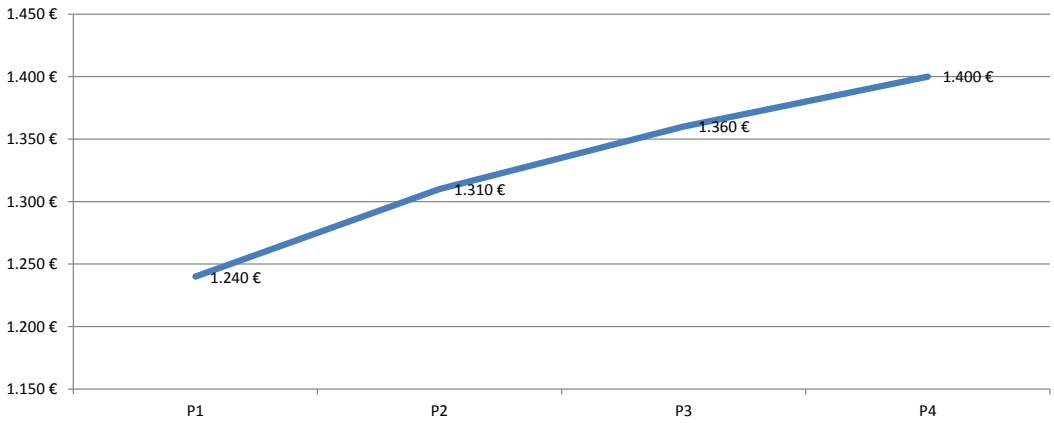


Abbildung 6.3: Gesamtumsätze bei unterschiedlicher Profilverteilung

6.2 Versuchsaufbau

Im Produktivbetrieb ist die Webseite der Produktvorschläge über zwei Wege erreichbar. Ist der Benutzer bereits registriert, erreicht er direkt nach der Anmeldung die Empfehlungsseite. Im anderen Fall gelangt er über ein Werbemittel (Textlink, Werbebanner, Gif-Animation, etc.) auf eine Landing Page[1]. Nach dem Abschluss wird der Benutzer auf

[1]Eine Landing Page ist eine Zielgruppen-optimierte Webseite, die eine kurze Interaktion (Anfrageformular, Gewinnspielteilnahme, etc.) mit dem Interessenten ermöglicht.

die Empfehlungsseite weitergeleitet und erhält die Produktvorschläge. Im alten Empfehlungssystem wurde nur registriert, ob der Benutzer auf die Seite der Produktvorschläge gelangt und welches Produkt er abgeschlossen hat. Bei dem neuem Empfehlungssystem hingegen wurde zusätzlich gespeichert, welche Produkte in welcher Reihenfolge angezeigt wurden, welches Produkt angeklickt bzw. abgeschlossen wurde und wann der Benutzer die Seite wieder verlassen hat.
Der Einstieg über die Landing Page bietet die Möglichkeit das *New User* Problem zu verringern. Jede Landing Page besitzt - genau wie ein Produkt - charakteristische Merkmale. Schließt ein neuer Benutzer die erste Landing Page ab, werden ihm die Merkmale initial übertragen. Diese **First Campaign Adoption** bietet die Grundlage für die ersten Produktvorschläge. Weitere Produktabschlüsse fließen in die Profilmerkmale ein und erweitern diese zu einem immer detaillierteren Interessenbild des Benutzers.
Der Beobachtungszeitraum beträgt bei beiden Empfehlungssystemen jeweils fünf Wochen. Betrachtet wurden die Top 5 Produkte, die in den zehn Wochen auf häufigsten abgeschlossen wurden. Bei den ausgewerteten Produkten handelte es sich um *Pay per Lead*-Werbekampagnen, die eine Vergütung pro Lead vorsehen.

6.3 Vergleich altes und neues System

Das alte Empfehlungssystem bezog in den Produktvorschlägen keine Benutzerinteressen ein. Stattdessen wurden die vorzuschlagenden Produkte nach ihrem Umsatz und ihrer Konvertierungsrate absteigend sortiert ausgegeben. Zu der Zeit wurden noch keine Produktvorschläge getrackt. Somit ist nur eine zweistufige Betrachtung (Clicked, Completed) möglich. Aus Rücksicht auf die Kunden der Jaduda GmbH wurden die ausgewerteten Produkte anonymisiert.
Ein wichtiger Indikator für den Erfolg von “Pay per Lead”-Produkten ist die Anzahl der generierten Leads[2]. Hier wurde die Anzahl der Leads wochenweise betrachtet. In den ersten fünf Wochen wurden durchschnittlich 386 Leads pro Woche erstellt. Die drauffolgenden Wochen erbrachten 388 Leads pro Woche.

[2]Ein Lead ist ein Kontaktdatensatz eines interessierten Kunden für ein bestimmtes Produkt.

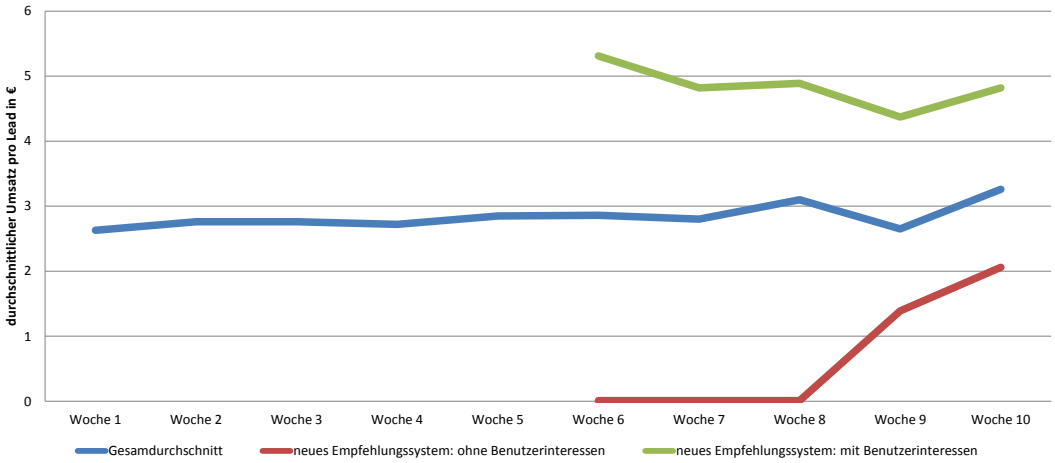


Abbildung 6.4: Durchschnittlicher Umsatz pro Lead

In den ersten fünf Wochen wurde pro Lead ein durchschnittlicher Umsatz von **2,73 €** erzielt. Die drauffolgenden Wochen erhöhten den Umsatz pro Lead um **5,12 %** auf **2,87 €**. Ab Woche 8 wurde der Impressions Barrier im Product Scoring (siehe Abschnitt 5.4.4) überwunden und pushte so die Vorschläge für Benutzer ohne Ausprägungen zusätzlich (Abbildung 6.4).

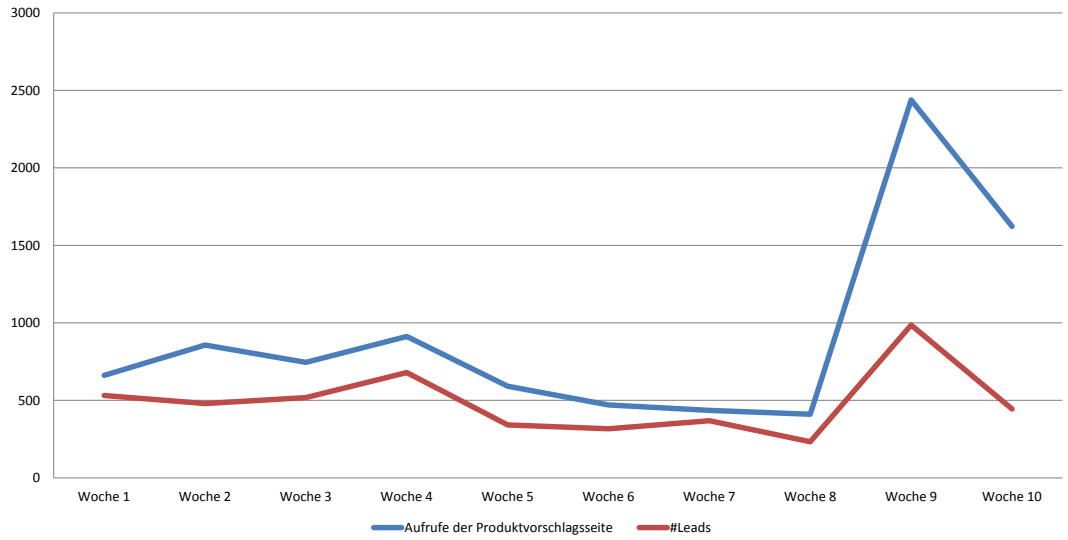


Abbildung 6.5: Verlauf der Produktvorschläge und Leads

Diese nur leichte Optimierung ist drei Umständen geschuldet. Zum einen wurden einige Landing Page Produkte stärker beworben und frequentiert, die keine Weiterleitung zu den Produktvorschlägen besaßen. Dieses Verhalten kann explizit vom Kunden gewünscht werden, da ein exklusiver Kundendatensatz gefordert wird, der nur mit diesem Landing

Page Produkt in Berührung gekommen ist. Somit verringerten sich die Besucher der Produktvorschlagsseite und auch die generierten Leads bis zur neunten Woche (Abbildung 6.5) leicht. Danach war ein sprunghafter Anstieg aufgrund veränderter Landing Page Bewerbung zu sehen, der zu einer starken Steigerung der Leadgenerierung führte und somit auch den absoluten Umsatz erhöhte.
Der zweite Umstand ist der hohe Anteil von Benutzern ohne charakteristische Ausprägungen (Abbildung 6.6). Für das neue Empfehlungssystem wurden nur die am häufigsten verwendeten Landing Pages anhand ihrer Merkmale charakterisiert. Das führte dazu, dass Benutzer, die durch nicht charakterisierte Landing Pages auf die Seite der Produktvorschläge gelangten, selber keine Ausprägungen besaßen. So flossen nur die Werte aus dem Produkt Scoring und keine Benutzerinteressen in die Produktvorschläge ein. Somit werden ähnliche Produkte beworben wie auch bei dem alten Empfehlungssystem.

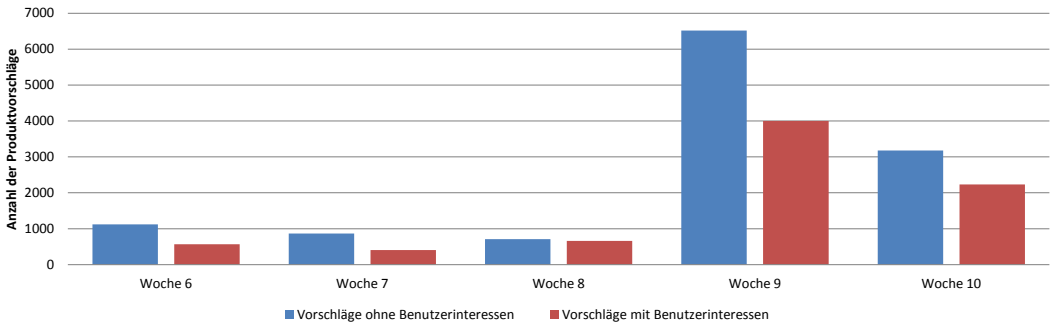


Abbildung 6.6: Vorschläge mit/ohne Benutzerinteressen

Der dritte Umstand war die breitere Produktfächerung durch das neue Empfehlungssystem. Einige Produkte, die geringen bis keinen Umsatz pro Lead produzierten wurden durch die Umstellung mehr vorgeschlagen als früher. Dadurch verringerte sich der Umsatz leicht (Abbildung 6.7). Ab der neunten Woche wurde die Impressions Barrier überwunden, wodurch Produkte mit einer guten Konvertierungsrate häufiger vorgeschlagen wurden und den Umsatz erhöhten.

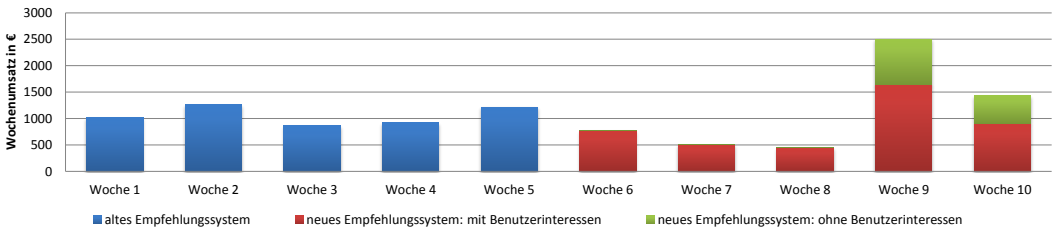


Abbildung 6.7: Umsätze pro Woche

6.4 Vergleich Content-based und Semantic Recommender

Im Beobachtungszeitraum generierte durchschnittlich in 51 von 100 Fällen der Semantic Recommender und in den restlichen 49 Fällen der Content-based Recommender die Vorschläge. Wie in Kapitel 5.4.5 beschrieben, wurde zu jedem Produktvorschlag das aufrufende Benutzerprofil, die Anzeigeposition, die verwendete Recommender Engine und bei welchem Schritt die Benutzer-Produkt-Interaktion (angezeigt, angeklickt, abgeschlossen) endete, gespeichert (Abbildung 6.8).

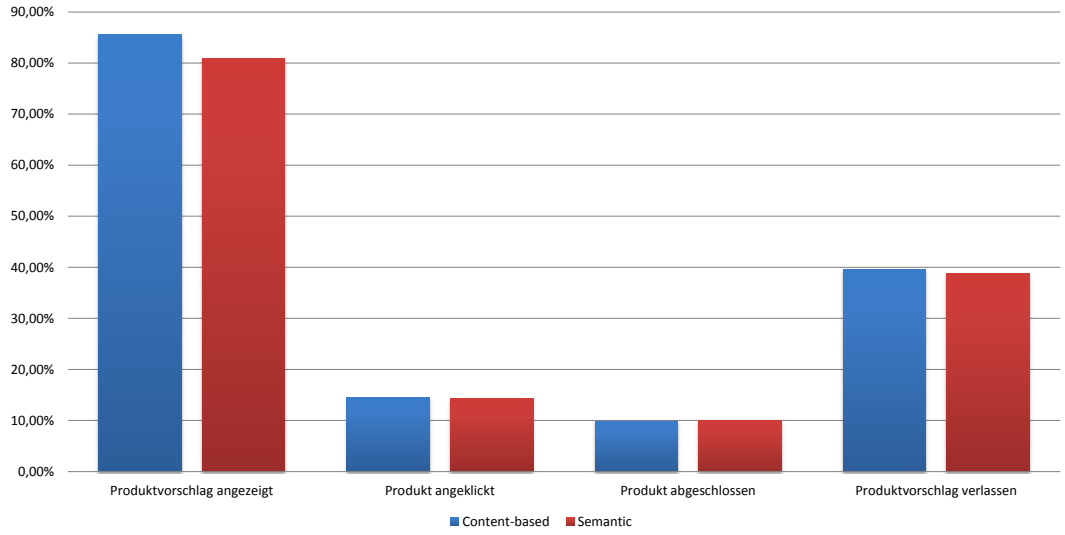


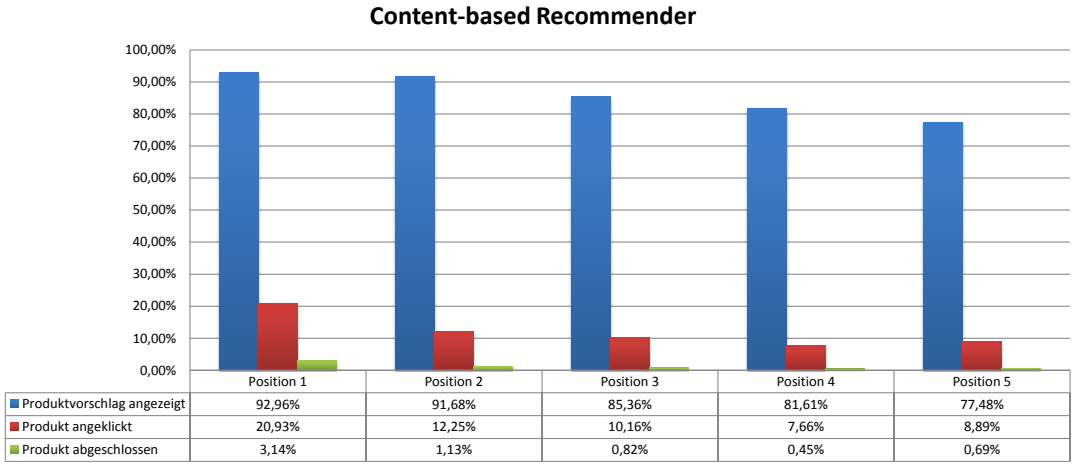
Abbildung 6.8: Vergleich Content-based und Semantic Recommender

Nicht alle Produktvorschläge waren dem Benutzer sofort sichtbar. Das kann z.B. an einem zu kleinen Display liegen. Erst wenn der Produktvorschlag komplett dargestellt wurde, ist er als angezeigt markiert worden. Insgesamt gesehen verhielten sich die Interaktionsraten bei den Produktvorschlägen beider Systeme sehr ähnlich. Die ersten drei Produktvorschläge wurden beim Content-based Recommender mit einer Wahrscheinlichkeit von 92,74 %, 91,44 % und 85,16 % angezeigt. Der Semantic Recommender war nur leicht schlechter mit 90,58 %, 90,25 % und 81,42 %. Hier lässt sich erkennen, dass in einigen Fällen nur die ersten beiden Produkte angezeigt wurden, bevor die Vorschlagseite verlassen wurde. Die Ursache hierfür liegt ebenfalls an den Bildschirmen der Smartphones. Hier werden meist nur die ersten zwei Produkte angezeigt. Um weitere Produkte zu sehen, muss heruntergescrollt werden.
In zwei Fällen wurde die Seite verlassen:

- kein Interesse an den Vorschlägen oder
- ein Produkt war interessant, wurde angeklickt und der Benutzer wurde auf die Produktseite weitergeleitet.

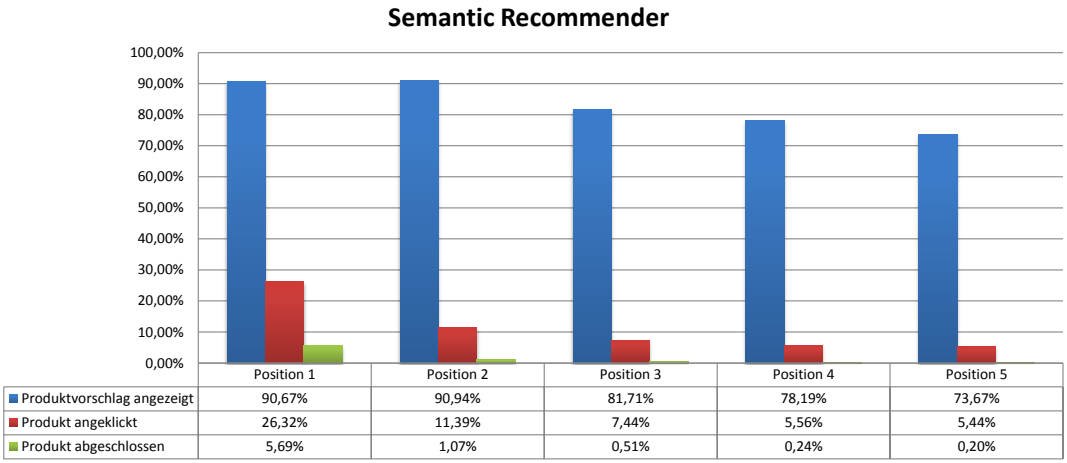
Nicht jedes Verlassen der Vorschlagseite wurde durch den entwickelten Javascript-Tracker (Abschnitt 5.4.5) erfasst. Ursache hierfür war das unterschiedliche Verhalten der verschiedenen Browser bei dem Javascript-Event *onUnload*. Verlässt der Benutzer mit seinem Browser eine Webseite, wird dieses Verhalten durch das *onUnload*-Event signalisiert. Einige Browser unterbinden die Signalisierung dieses Events, da einige Webseitenbetreiber den Benutzer am Verlassen ihrer Präsenz verhindern wollen.

Die Produktposition hatte einen maßgeblichen Einfluss auf den Produktabschluss. Produkte, die an den Positionen 2-5 vom Content-based Recommender (Abbildung 6.9) vorgeschlagen wurden, wurden häufiger angeklickt und abgeschlossen als beim Semantic Recommender (Abbildung 6.10). Jedoch wurde das Produkt an Position 1 beim Semantic Recommender zu 26,32 %, beim Content-based Recommender zu 20,93 % angeklickt. Entsprechend ist die Abschlussrate beim Semantic Recommender mit 5,93 % höher als beim Content-based Recommender mit 3,14 %.



	Position 1	Position 2	Position 3	Position 4	Position 5
Produktvorschlag angezeigt	92,96%	91,68%	85,36%	81,61%	77,48%
Produkt angeklickt	20,93%	12,25%	10,16%	7,66%	8,89%
Produkt abgeschlossen	3,14%	1,13%	0,82%	0,45%	0,69%

Abbildung 6.9: Benutzerinteraktion in Abhängigkeit zur Produktposition beim Content-based Recommender



	Position 1	Position 2	Position 3	Position 4	Position 5
Produktvorschlag angezeigt	90,67%	90,94%	81,71%	78,19%	73,67%
Produkt angeklickt	26,32%	11,39%	7,44%	5,56%	5,44%
Produkt abgeschlossen	5,69%	1,07%	0,51%	0,24%	0,20%

Abbildung 6.10: Benutzerinteraktion in Abhängigkeit zur Produktposition beim Semantic Recommender

Der positionsbezogene Umsatz unterschied sich bei den Empfehlungssystemen beträchtlich (Abbildung 6.11). Generierte ein abgeschlossenes Produkt des Semantic Recommender an Position 1 durchschnittlich **5,44 €**, so erzeugte ein Produkt des Content-based Recommender im Mittel nur **3,84 €**. Überraschenderweise war der Umsatz des Content-based Recommender an Position 3 höher als an Position 2. Absolut gesehen ist die Umsatzkurve des Content-based Recommenders schwacher fallend und harmonischer als die des Semantic Recommender. Beim Semantic Recommender fällt die Umsatzkurve bis Position 5 stark ab und erreicht Beträge zwischen 0,03 und 0,06 €.

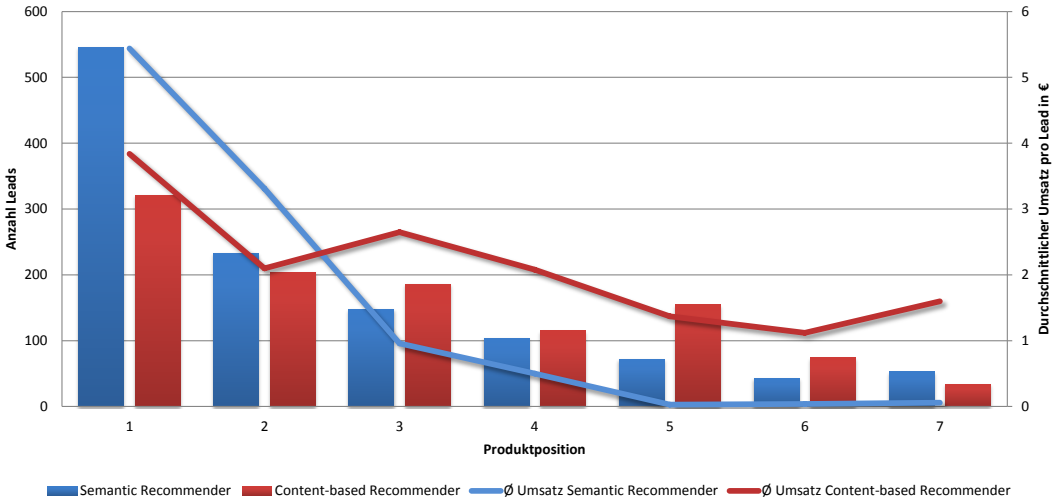


Abbildung 6.11: Abhängigkeit der Produktposition für Leads und Umsatz

Insgesamt gesehen ist der Semantic Recommender mit 3,24 € Umsatz pro Lead deutlich

besser als der Content-based Recommender mit 2,50 € und erzeugt mit den ersten drei Produktvorschlägen über 41 % des gesamten Umsatzes. Der Content-based Recommender produziert mit diesen Vorschlägen lediglich 23 %.

7 Fazit und Ausblick

Das entwickelte Symfony 2 Bundle enthält zwei Empfehlungssysteme. Zum einen enthält es einen Content-based Recommender, der seine Vorschläge anhand der Merkmale der Produkte erstellt. Zum anderen enthält es einen Semantic Recommender, der eine Wissensbasis besitzt. Dadurch werden nicht nur übereinstimmende Merkmale zwischen Benutzer und Produkt gefunden, sondern auch Pfade über benachbarte Merkmale, die die Vorschlagsmenge stark erweitern.
Die Auswertung ergibt, dass der Semantic Recommender in den beobachteten fünf Wochen **45,92%** mehr Umsatz generierte als der Content-based Recommender. Insgesamt steigerte sich der Umsatz mit den beiden entwickelten Empfehlungssystemen um 5,12% pro Lead bei einer leichten Mengensteigerung der Leads im Vergleich zum alten Empfehlungssystem.
Die Umstellung auf das Recommender-Bundle kann als Erfolg gesehen werden, jedoch ist der Wechsel nicht völlig fehlerfrei verlaufen. Zu erwähnen sind einige Unwägbarkeiten, die auftraten.
Die ersten Fassung des Semantic Recommenders ist fehleranfällig, falls dem Benutzer keine Ausprägungen zugeordnet werden. In diesem Fall findet das Empfehlungssystem keine vorzuschlagenden Produkte. Besitzt das Profil keine Ausprägungen, kann somit auch keine Übereinstimmung zwischen den Ausprägungen des Profils und denen der Produkte gefunden werden, was ein leeres Ergebnis liefert. Dadurch können auch keine Produkte vorgeschlagen werden. Die Fehlfunktion wurde behoben, indem Vorschläge ausgehend vom Produkt und dessen Ausprägungen gesucht werden und lediglich die Werte aus dem Product Scoring Manager in den Vorschlag einfließen. Des Weiteren wurde Benutzern, die über ein nicht charakterisierte Landing Page auf die Vorschlagsseite gelangten, initial neutrale Ausprägungen zugeordnet.
Auch wenn das Fazit insgesamt gut zu den beiden Empfehlungssystemen ist, gibt es noch Optimierungspotenzial. Das D2RQ-Mapping und die SPARQL-Abfrage des Semantic Recommenders sind noch keine optimale Lösung. Durch das Mapping ist eine feste Pfadlänge bei der Suche nach Übereinstimmungen von Merkmalen definiert. Da-

durch werden Produkte ausgeschlossen, die eine kürzeren oder längeren Pfad besitzen. Daher ist es sinnvoll, nur noch die einzelnen Tabellen und keine Pfade im Mapping zu definieren. Die Suche kann dann im SPARQL-Query durch Suchalgorithmen der Graphentheorie realisiert werden.
Eine weitere Verbesserung betrifft das Management der Profilmerkmale. Dort werden Ausprägungen des Profils nach einer gewissen Zeit ausgeblendet. Dadurch kann schneller auf geänderte Interessen eines Benutzers reagiert werden. Jedoch kann das zu einer Verschlechterung der Vorschläge führen, wenn die Benutzerinteressen nicht regelmäßig aufgefrischt werden [Billsus & Pazzani 2000].
Auf Wunsch der Marketing-Abteilung der Jaduda GmbH wird der Empfehlungssimulator verfeinert. Ziel ist eine Umsatz- und Abschlussvorhersage zu einem charakterisierten Produkt, bevor es gestartet wird. Die Vorhersage soll sich an ähnlichen Produkten orientieren und einen verlässlichen Wert liefern, wie ein Produkt sich entwickeln wird. Um aussagekräftige Ergebnisse zu erhalten, muss die Datenbank eine ausreichend große Menge an ähnlichen Produkten und dazugehörigen Produktvorschlägen besitzen.
Das Bundle wird Open Source zur Verfügung gestellt. Dieser Schritt führt hoffentlich zu weiterem Feedback und Verbesserungsvorschlägen, mit denen das Bundle weiter optimiert werden kann. Die Resonanz ist vor allem deshalb interessant, da noch kein Recommender Bundle für Symfony 2 existiert und das Interesse dafür nicht absehbar ist.

Literatur

Auer, Sören et al. (2007). »DBpedia: A Nucleus for a Web of Open Data«. In: *The Semantic Web*. Hrsg. von Karl Aberer et al. Bd. 4825. Lecture Notes in Computer Science. Springer Berlin Heidelberg, S. 722–735. ISBN: 978-3-540-76297-3. DOI: 10.1007/978-3-540-76298-0_52. URL: http://dx.doi.org/10.1007/978-3-540-76298-0_52.

Balabanović, Marko & Yoav Shoham (März 1997). »Fab: content-based, collaborative recommendation«. In: *Commun. ACM* 40.3, S. 66–72. ISSN: 0001-0782. DOI: 10.1145/245108.245124. URL: http://doi.acm.org/10.1145/245108.245124.

Belkin, Nicholas J. & W. Bruce Croft (Dez. 1992). »Information filtering and information retrieval: two sides of the same coin?« In: *Commun. ACM* 35.12, S. 29–38. ISSN: 0001-0782. DOI: 10.1145/138859.138861. URL: http://doi.acm.org/10.1145/138859.138861.

Billsus, Daniel & Michael J. Pazzani (2000). »User Modeling for Adaptive News Access«. English. In: *User Modeling and User-Adapted Interaction* 10.2-3, S. 147–180. ISSN: 0924-1868. DOI: 10.1023/A:1026501525781. URL: http://dx.doi.org/10.1023/A%3A1026501525781.

Boutilier, Craig, Richard S. Zemel & Benjamin Marlin (2003). »Active collaborative filtering«. In: *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*. UAI'03. Acapulco, Mexico: Morgan Kaufmann Publishers Inc., S. 98–106. ISBN: 0-127-05664-5. URL: http://dl.acm.org/citation.cfm?id=2100584.2100596.

Breese, John S., David Heckerman & Carl Kadie (1998). »Empirical analysis of predictive algorithms for collaborative filtering«. In: *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*. UAI'98. Madison, Wisconsin: Morgan Kaufmann Publishers Inc., S. 43–52. ISBN: 1-55860-555-X. URL: http://dl.acm.org/citation.cfm?id=2074094.2074100.

Burke, Robin (2002). »Hybrid Recommender Systems: Survey and Experiments«. English. In: *User Modeling and User-Adapted Interaction* 12.4, S. 331–370. ISSN: 0924-1868. DOI: 10.1023/A:1021240730564. URL: http://dx.doi.org/10.1023/A%3A1021240730564.

Decker, S., P. Mitra & S. Melnik (2000). »Framework for the semantic Web: an RDF tutorial«. In: *Internet Computing, IEEE* 4.6, S. 68–73. ISSN: 1089-7801. DOI: 10.1109/4236.895018.

Fix, Evelyn & Jr. J. L. Hodges (1951). *Discriminatory Analysis: Nonparametric Discrimination: Consistency Properties*. Techn. Ber. Project 21-49-004, Report Number 4. USAF School of Aviation Medicine, Randolf Field, Texas, S. 261–279.

Fox, Steve et al. (Apr. 2005). »Evaluating implicit measures to improve web search«. In: *ACM Trans. Inf. Syst.* 23.2, S. 147–168. ISSN: 1046-8188. DOI: 10.1145/1059981.1059982. URL: http://doi.acm.org/10.1145/1059981.1059982.

Gnasa, Melanie et al. (2005). »Personalized Peer Filtering for a Dynamic Information Push«. In: *Foundations of Intelligent Systems*. Hrsg. von Mohand-Said Hacid et al. Bd. 3488. Lecture Notes in Computer Science. Springer Berlin Heidelberg, S. 650–659. ISBN: 978-3-540-25878-0. DOI: 10.1007/11425274_67. URL: http://dx.doi.org/10.1007/11425274_67.

Hepp, Martin (2008). »GoodRelations: An Ontology for Describing Products and Services Offers on the Web«. In: *Knowledge Engineering: Practice and Patterns*. Hrsg. von Aldo Gangemi & Jérôme Euzenat. Bd. 5268. Lecture Notes in Computer Science. Springer Berlin Heidelberg, S. 329–346. ISBN: 978-3-540-87695-3. DOI: 10.1007/978-3-540-87696-0_29. URL: http://dx.doi.org/10.1007/978-3-540-87696-0_29.

Herlocker, Jonathan L. et al. (Jan. 2004). »Evaluating collaborative filtering recommender systems«. In: *ACM Trans. Inf. Syst.* 22.1, S. 5–53. ISSN: 1046-8188. DOI: 10.1145/963770.963772. URL: http://doi.acm.org/10.1145/963770.963772.

Hey, John B. (1991). »System and method for recommending items«. Pat. US4996642. URL: http://patft.uspto.gov/netacgi/nph-Parser?Sect2=PTO1&Sect2=HITOFF&p=1&u=/netahtml/PTO/search-bool.html&r=1&f=G&l=50&d=PALL&RefSrch=yes&Query=PN/4996642.

Hill, Will et al. (1995). »Recommending and evaluating choices in a virtual community of use«. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '95. Denver, Colorado, USA: ACM Press/Addison-Wesley Publishing Co., S. 194–201. ISBN: 0-201-84705-1. DOI: 10.1145/223904.223929. URL: http://dx.doi.org/10.1145/223904.223929.

Hu, Yifan, Y. Koren & C. Volinsky (2008). »Collaborative Filtering for Implicit Feedback Datasets«. In: *Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on*, S. 263–272. DOI: 10.1109/ICDM.2008.22.

Joachims, Thorsten et al. (Apr. 2007). »Evaluating the accuracy of implicit feedback from clicks and query reformulations in Web search«. In: *ACM Trans. Inf. Syst.* 25.2. ISSN: 1046-8188. DOI: 10.1145/1229179.1229181. URL: http://doi.acm.org/10.1145/1229179.1229181.

Kelly, Diane & Nicholas J. Belkin (2004). »Display time as implicit feedback: understanding task effects«. In: *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '04. Sheffield, United Kingdom: ACM, S. 377–384. ISBN: 1-58113-881-4. DOI: 10.1145/1008992.1009057. URL: http://doi.acm.org/10.1145/1008992.1009057.

Kotler, Philip (2010). *Grundlagen des Marketing*. Pearson Deutschland GmbH.

Liang, Ting-Peng, Hung-Jen Lai & Yi-Cheng Ku (Jan. 2007). »Personalized Content Recommendation and User Satisfaction: Theoretical Synthesis and Empirical Findings«. In: *J. Manage. Inf. Syst.* 23.3,

S. 45–70. ISSN: 0742-1222. DOI: 10.2753/MIS0742-1222230303. URL: http://dx.doi.org/10.2753/MIS0742-1222230303.

Lyman, Peter & Hal Varian (2004). *HOW MUCH INFORMATION 2003?* Techn. Ber. URL: http://www.sims.berkeley.edu/research/projects/how-much-info-2003/printable_report.pdf.

Oard, Douglas W., Jinmook Kim et al. (1998). »Implicit feedback for recommender systems«. In: *Proceedings of the AAAI workshop on recommender systems*. Wollongong, S. 81–83.

Pazzani, Michael J. & Daniel Billsus (2007). »Content-Based Recommendation Systems«. In: *The Adaptive Web*. Hrsg. von Peter Brusilovsky, Alfred Kobsa & Wolfgang Nejdl. Bd. 4321. Lecture Notes in Computer Science. Springer Berlin Heidelberg, S. 325–341. ISBN: 978-3-540-72078-2. DOI: 10.1007/978-3-540-72079-9_10. URL: http://dx.doi.org/10.1007/978-3-540-72079-9_10.

Pearson (1895). In: *Royal society proceedings* 58, S. 241.

Rashid, Al Mamunur et al. (2002). »Getting to know you: learning new user preferences in recommender systems«. In: *Proceedings of the 7th international conference on Intelligent user interfaces*. IUI '02. San Francisco, California, USA: ACM, S. 127–134. ISBN: 1-58113-459-2. DOI: 10.1145/502716.502737. URL: http://doi.acm.org/10.1145/502716.502737.

Resnick, Marc L et al. (2004). »Persuasive Design Through Intelligent Recommendation Systems«. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. Bd. 48. 13. SAGE Publications, S. 1503–1507. URL: http://pro.sagepub.com/content/48/13/1503.short.

Resnick, Paul et al. (1994). »GroupLens: an open architecture for collaborative filtering of netnews«. In: *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. CSCW '94. Chapel Hill, North Carolina, USA: ACM, S. 175–186. ISBN: 0-89791-689-1. DOI: 10.1145/192844.192905. URL: http://doi.acm.org/10.1145/192844.192905.

Rüttgers, Jürgen (1999). *Zeitenwende, Wendezeiten.: Das Jahr-2000-Projekt: Die Wissensgesellschaft.* Siedler Verlag.

Schafer, J. Ben, Joseph Konstan & John Riedi (1999). »Recommender systems in e-commerce«. In: *Proceedings of the 1st ACM conference on Electronic commerce*. EC '99. Denver, Colorado, USA: ACM, S. 158–166. ISBN: 1-58113-176-3. DOI: 10.1145/336992.337035. URL: http://doi.acm.org/10.1145/336992.337035.

Schafer, J. Ben, Joseph A. Konstan & John Riedl (2001). »E-Commerce Recommendation Applications«. English. In: *Applications of Data Mining to Electronic Commerce*. Hrsg. von Ron Kohavi & Foster Provost. Springer US, S. 115–153. ISBN: 978-1-4613-5648-6. DOI: 10.1007/978-1-4615-1627-9_6. URL: http://dx.doi.org/10.1007/978-1-4615-1627-9_6.

Van Meteren, Robin & Maarten Van Someren (2000). »Using content-based filtering for recommendation«. In: *Proceedings of the Machine Learning in the New Information Age: MLnet/ECML2000 Workshop*.

Xiao, Bo & Izak Benbasat (2007). »E-commerce product recommendation agents: use, characteristics, and impact«. In: *Mis Quarterly* 31.1, S. 137–209.

Xin, Zhiyun et al. (2005). »Information Push-Delivery for User-Centered and Personalized Service«. In: *Fuzzy Systems and Knowledge Discovery*. Hrsg. von Lipo Wang & Yaochu Jin. Bd. 3613. Lecture Notes in Computer Science. Springer Berlin Heidelberg, S. 594–602. ISBN: 978-3-540-28312-6. DOI: 10.1007/11539506_74. URL: http://dx.doi.org/10.1007/11539506_74.

Yun, Gi Woong & Craig W. Trumbo (2000). »Comparative Response to a Survey Executed by Post, E-mail, & Web Form«. In: *Journal of Computer-Mediated Communication* 6.1, S. 0–0. ISSN: 1083-6101. DOI: 10.1111/j.1083-6101.2000.tb00112.x. URL: http://dx.doi.org/10.1111/j.1083-6101.2000.tb00112.x.

Zheng, Rong, Dennis Wilkinson & Foster Provost (2008). »Social network collaborative filtering«. In: *Stern, IOMS Department, CeDER, Vol.*