Hochschule Neubrandenburg
University of Applied Sciences

Hochschule Neubrandenburg

Studiengang Geoinformatik

Werkzeuge für ein verhaltensbasiertes Empfehlungssystem für Community Inhalte

Bachelorarbeit

Zum Erlangen des akademischen Grades

„Bachelor of Engineering“ (B.Eng.)

vorgelegt von

Toni Wojciak

thesis 2012-0645-8

Erstprüfer: Prof. Dr. Dipl.-Ing. Andreas Wehrenpfennig

Zweitprüfer: Dipl.-Math. Holger Diener

2012

Danksagung

Ich bedanke mich bei Holger Diener und Andreas Müller für die Betreuung meines Praktikums und meiner Bachelorarbeit im Fraunhofer Institut für graphische Datenverarbeitung. Ein weiterer Dank gebührt Herrn Professor Wehrenpfennig für die Betreuung meiner Bachelorarbeit an der Hochschule Neubrandenburg.

Kurzfassung

Diese Arbeit thematisiert Empfehlungssysteme und ob ein Solches entwickelt werden kann, das sowohl ohne explizite Bewertungen der Inhalte, als auch ohne explizite Informationen über die Nutzer zurechtkommt. Zunächst wird ein Überblick über den aktuellen Forschungsstand von Empfehlungssystemen aufgezeigt. Dabei werden die drei Hauptsysteme von Empfehlungssystemen erklärt: das inhaltliche, gemeinschaftsbasierte und hybride Empfehlungssystem. Am Ende des Forschungsstandes werden die Methodiken zum Sammeln von Daten erläutert. Im Anschluss daran wird ein Empfehlungssystem konzipiert, das die oben genannten Anforderungen berücksichtigt. Indes wird ein geeignetes Empfehlungssystem ausgewählt und drei Empfehlungsfunktionen entwickelt und beschrieben. Dabei handelt es sich sowohl um die Beliebtheit und Aktualität von Seiten im Community Portal Liferay, als auch um die Ähnlichkeit von Nutzer. Diese drei Funktionen werden anschließend in einer neuen Funktion kombiniert. Diese Funktion berechnet aus der Kombination aller Funktionen einen Empfehlungswert. Der Empfehlungswert wiederum, wird dazu verwendet einen Farbwert im RGB System zu berechnen, welcher die Wichtigkeit einer Seite visuell in Liferay ausdrücken soll. Um einen Einblick in die Funktionsweise des Empfehlungssystems zu ermöglichen, wird anschließend ein Werkzeug zur Konfiguration und Überprüfung der Ergebnisse konzipiert. Abschließend wird das konzipierte System in Form eines Portlets, mit der Programmiersprache JAVA und dem darauf aufbauenden Webanwendungs-Framework Vaadin, umgesetzt.

Abstract

This work deals with recommendation systems especially if such can be developed that manage without explicit feedback of the content and no explicit information about the user. First an overview of the current state of the art of recommendation systems is provided. Here, the three main systems of recommendation systems are explained: the content, collaborative, and hybrid recommendation system. After the methodology for collecting data a recommendation system that considers the above mentioned requirements is designed. An appropriate recommendation system will be selected and three recommendation functions are developed and described. It is the popularity and relevance of sites, as well as to the similarity of users in the user community portal Liferay. These three features are then combined in a new, hybrid function. This function calculates a recommendation value out of all the functions. This is used to calculate a color value in the RGB system, which visually expresses the importance of a page in Liferay. To provide an insight into the functioning of the recommendation system, a tool for configuring and monitoring the results will be designed. Finally, the designed system will be implemented in form of a Portlet with the programming language Java and the according web application framework Vaadin.

Inhaltsverzeichnis

Abkürzungsverzeichnis

IES - Inhaltsbasiertes Empfehlungssystem

GES - Gemeinschaftsbasiertes Empfehlungssystem

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Recommender Systems (deutsch: Empfehlungssysteme) sollen die Entscheidungen eines Menschen unterstützen, indem Empfehlungen für ein bestimmtes Objekt angezeigt werden. Ein Objekt ist, in Bezug auf diese Arbeit, etwas, das von einem Nutzer betrachtet, gekauft, bewertet oder auch empfohlen werden kann. In einem typischen Empfehlungssystem geben Menschen direkt oder indirekt eine Bewertung für ein Objekt ab. Diese Bewertung wird als Eingabe für das Empfehlungssystem genutzt [RV97]. Durch die Anzahl getätigter Bewertungen in einem Online Shop, kann beispielsweise für ein käufliches Objekt eine Empfehlung berechnet werden, welche die Kaufentscheidung des Nutzers unterstützen soll.

Doch Empfehlungssysteme können auch in anderen Bereichen eingesetzt werden, als nur zur Unterstützung eines Einkaufs in einem Online Shop. So kann eine Firma z.B. ein Empfehlungssystem verwenden, welches ihr ermöglicht den Mitarbeitern Internetseiten vorzuschlagen, die ihnen helfen sollen, bestimmte Probleme zu lösen. Sucht ein Nutzer nach einem Objekt im Internet, wird es durch die Fülle von Informationen, immer schwieriger dieses in kürzester Zeit zu finden. Empfehlungssysteme sollen die Nutzer personalisiert zu interessanten Objekten zu leiten [GDS11]. Dafür nutzen Empfehlungssysteme die Meinungen der Nutzer. Diese wiederum sollen den Nutzern bei ihrer Entscheidung für oder gegen ein Objekt unterstützen [COS03].

Im Buch „Mining of Massive Datasets“ [RLU12] wird das sogenannte „long tail“ Phänomen beschrieben. Dieses zeigt, dass es viele Produkte gibt, die nicht bekannt gemacht oder allgemein als „Flop“ bezeichnet wurden. Resultierend daraus können diese Produkte aber im Internet dennoch angeboten und mit den sonstigen Top-Sellern, in beispielsweise einem Buchladen, gleichgestellt werden. Ein Buchladen hat hunderte von Büchern, während einem Menschen bei Amazon.com Millionen von Büchern zur Auswahl stehen. Eine Zeitung wiederum bietet am Tag mehrere dutzend Artikel an, während in einem Online Nachrichtenservice mehrere hundert angeboten werden. Somit wird gezeigt, dass nicht alle Objekte für z.B. einen normalen Laden empfohlen werden können, da es dafür zu viele sind. Dieses Phänomen verdeutlicht die Bedeutung von Empfehlungssystemen, vor Allem für Online Shops.

Schafer et al. [SKR99] beschreiben den Nutzen von Empfehlungssystemen auf E-Commerce-Webseiten. Ein Empfehlungssystem kann dem aktuellen Nutzer beispielsweise Top-Seller eines Online-Shops empfehlen. Außerdem ist es möglich, Objekte zu empfehlen, die ein ähnlicher Nutzer betrachtet, oder gekauft hat. Die Autoren erklären an mehreren Beispielen, welche Methoden die einzelnen E-Commerce Seiten nutzen. Amazon.com nutzt z.B. den Book Matcher, welcher es einem Nutzer ermöglicht bereits gelesene Bücher zu bewerten. Aufgrund dieser Bewertungen kann sich der Nutzer dann Bücher empfehlen lassen, die ihn möglicherweise interessieren könnten. Die Webseite „Moviefinder.com“ wiederum ver-

wendet z.B. den Match Maker. Dieser erlaubt es einem Nutzer ein bestimmtes Thema oder Genre auszuwählen, um sich danach Filme empfehlen zu lassen.

2 Aktueller Forschungsstand

Kategorisierung der Systeme

Nach Adomavicius und Tuzhilin [AT05] lassen sich Empfehlungssysteme in drei Hauptbereiche einteilen:

- Inhaltsbasierte Empfehlungssysteme
- Gemeinschaftsbasierte Empfehlungssysteme
- Hybride Empfehlungssysteme

Bei den inhaltsbasierten Empfehlungssystemen werden einem Nutzer Objekte vorgeschlagen, die denen ähnlich sind, die er auch in der Vergangenheit bevorzugt hat. Das gemeinschaftsbasierte Empfehlungssystem untersucht, wie ähnlich der aktuelle Nutzer einem anderen Nutzer ist. Im Anschluss werden dem aktuellen Nutzer Objekte vorgeschlagen, welche ähnliche Nutzer bevorzugen. Durch einen kombinierten Einsatz der zwei genannten Empfehlungssysteme, entsteht ein hybrides Empfehlungssystem. Dieses kann einige Probleme der einzelnen Systeme überwinden, um so präzisere Empfehlungen zu berechnen.

In den folgenden Kapiteln des aktuellen Forschungsstandes werden die drei genannten Empfehlungssysteme beschrieben. Daraufhin folgt das Kapitel über die Methodik zum Sammeln von Informationen, die als Eingabe für die Empfehlungssysteme dienen sollen.

2.1 *Inhaltsbasierte Empfehlungssysteme*

Ein inhaltsbasiertes Empfehlungssystem (IES) nutzt die betrachteten Objekte eines Nutzers aus der Vergangenheit, um neue und für den Nutzer interessante Objekte zu finden. Dabei vergleicht das IES die Inhalte der Objekte und berücksichtigt gleichzeitig die getätigten Bewertungen der Nutzer. Um dies zu ermöglichen, wird ein Profil für einen Nutzer angelegt [PB07]. Dieses kann aus verschiedenen Typen von Informationen bestehen. Beispielsweise kann das Profil die Nutzervorlieben beinhalten oder auch die getätigten Interaktionen. Die Nutzervorlieben bestehen aus den enthaltenen Objekten. Ein Objekt wird dabei durch „Features“ beschrieben, wie beispielsweise das Genre, den Titel oder die Wörter, die am meisten in einem Text vertreten sind. Ein Objekt wird dann als ein Vektor mit Features beschrieben, welche anschließend verglichen werden können [KAS06]. Bei den vergangenen Interaktionen wird darauf Wert gelegt, welche Objekte sich ein Nutzer z.B. angesehen oder welche dieser gekauft hat. Das IES analysiert das Profil eines Nutzers und kann daraus die Vorlieben ermitteln[PB07][MS00].
Aufgrund dieser Vektoren können andere Objekte gefunden und vorgeschlagen werden, die inhaltlich ähnliche Features enthalten. In [RLU12] stellen A. Rajaraman und J. D. Ullman in mehreren Beispielen dar, welche Inhalte von IES genutzt werden können. Sie beschreiben,

dass jedes Objekt wichtige Eigenschaften besitzt, durch die es möglich ist, die Objekte zu repräsentieren. So kann sich ein Nutzer auf einer Filmkritik Webseite beispielsweise Filme empfehlen lassen. Jedem Film sind dann z.B. der Regisseur, die Hauptdarsteller und das Genre zugeordnet. Bewertet ein Nutzer nun einen Film, so werden diesem Filme mit einem ähnlichen Genre empfohlen. Das IES sucht also zuerst nach Filmen mit diesem Genre und sortiert das Ergebnis der Suche nach den besten Bewertungen. Online Bücher Shops sind da ähnlich. Bewertet oder kauft ein Nutzer ein Buch, so können ihm Bücher mit einem ähnlichen Genre, oder auch weitere Bücher des Autors empfohlen werden. Die Bewertungen oder Betrachtungen eines Objektes können im Profil eines Nutzers gespeichert werden. Durch diese Informationen können dem Nutzer in Zukunft immer neue Objekte empfohlen werden, die ihn interessieren könnten. In [AT05] beschreiben die Autoren, dass neue Bewertungen von den bisher getätigten Bewertungen abhängig sind. So werden beispielsweise nur Filme empfohlen, die der Nutzer in der Vergangenheit hoch bewertet hat. Durch die Features eines Films, wie bereits beschrieben, ist das Finden von ähnlichen und gut bewerteten Filmen mit IES möglich.
Durch die Informationen über die Vorlieben des Nutzers, kann ein Profil angelegt werden, welches fähig ist, durch das Sammeln dieser Daten zu lernen. So werden die gesammelten Daten generalisiert, um immer wieder ein neues Profil für den Nutzer zu erstellen [LGS11].

Vor- und Nachteile

Jedes der Empfehlungssysteme besitzt einige Vor- und Nachteile. So haben die IES den Vorteil, dass neue Objekte, auf Grund der gespeicherten Features, vorgeschlagen werden können. Andererseits sind das „Limited Content Analysis“ und „new User Problem“ Nachteile von IES. Der erstgenannte Nachteil entsteht, wenn nur wenige inhaltliche Beschreibungen des Objektes vorhanden oder diese in einer, vom Computer nicht automatisch zu erfassenden Form gegeben sind. Außerdem können zwei unterschiedliche Objekte, die die gleichen inhaltlichen Beschreibungen haben, nicht unterschieden werden[AT05].
Das new User Problem ist der bekannteste Nachteil. Sofern sich ein Nutzer, in beispielsweise einem Online-Shop neu angemeldet hat, können diesem auf Grund der wenig zur Verfügung stehenden Informationen keine präzisen Empfehlungen angezeigt werden.

2.2 *Gemeinschaftsbasierte Empfehlungssysteme*

„Collaborative Filtering is the process of filtering or evaluating items using the opinion of other people“[SCH07]. Auch die gemeinschaftsbasierten Empfehlungssysteme (kurz GES) legen ein Profil für einen Nutzer an, welches die Vorlieben getätigter Betrachtungen oder auch Einkäufe speichern kann. Allerdings beziehen sich die GES auf die Ähnlichkeit der Profile von Nutzern. So werden einem aktuellen Nutzer die Objekte eines anderen Nutzers vorgeschlagen, welcher ähnliche Vorlieben in seinem Profil aufweist [SCH07][AT05].

GES lassen sich in zwei Algorithmen Bereiche [AT05] einteilen: speicher- und modellbasierte Algorithmen. Speicherbasierte Algorithmen nutzen den vollständigen Bestand an getätigten Bewertungen der Nutzer für ein Objekt. Daraus berechnet dieser Algorithmus eine Empfehlung. Dafür ist es wichtig Nutzer herauszufiltern, die ein ähnliches Profil aufweisen, wie das des aktuellen Nutzers. Dafür wird beispielsweise der Pearson-r Korrelationskoeffizient verwendet (siehe Abb. 2.1). In dieser Formel seien a und b zwei Nutzer, die in einer Funktion P_sim(a,b), miteinander verglichen werden sollen. P_{aj} sei die Bewertung des Nutzers a für das Objekt j und P_{bj} die Bewertung des Nutzers b für dasselbe Objekt. $\bar{P_a}$ und $\bar{P_b}$ seien die durchschnittlich getätigten Bewertungen der Nutzer a und b. Je dichter das Ergebnis dieser Berechnung an eins liegt, desto ähnlicher sind sich die Nutzer [LP08].

$$P\_sim(a,b) = \frac{\sum_j (P_{aj} - \overline{P_a})(P_{bj} - \overline{P_b})}{\sqrt{\sum_j (P_{aj} - \overline{P_a})^2} \sqrt{\sum_j (P_{bj} - \overline{P_b})^2}}$$

Abb. 2.1 - Berechnung des Pearson-r Korrelationskoeffizient zweier Nutzer a und b

Diese Formel wird vor allem dazu verwendet die Ähnlichkeit zweier Nutzer zu bestimmen. Auch die Ähnlichkeit der Objekte muss bestimmt werden. Der erste Schritt dafür ist die Separierung der Nutzer, die die zu vergleichenden Objekte bewertet haben. In einer Nutzer-Objekt-Matrix seien die Nutzer in Zeilen und die Objekte in Spalten unterteilt. So wird die Ähnlichkeit zweier Nutzer entlang der Zeilen berechnet und die Ähnlichkeit der Objekte entlang der Spalten [SAR01][HEU10]. Somit gibt es eine unterschiedliche Größenordnung, die zu beachten ist. Ein Lösungsansatz dafür ist die adjusted cosine similarity (deutsch: angepasste Kosinus Ähnlichkeit), wie in Abb. 2.2 dargestellt. Hierbei wird die Ähnlichkeit zweier Objekte i und j, sim(i,j), bestimmt. Dabei sollen $R_{u,i}$ und $R_{u,j}$ die Bewertungen des Nutzers u für die Objekte i und j darstellen und $\bar{R_u}$ die durchschnittliche Nutzerbewertung eines Nutzers u.

$$sim(i,j) = \frac{\sum_{u \in U} (R_{u,i} - \bar{R_u})(R_{u,j} - \bar{R_u})}{\sqrt{\sum_{u \in U} (R_{u,i} - \bar{R_u})^2} \sqrt{\sum_{u \in U} (R_{u,j} - \bar{R_u})^2}}.$$

Abb. 2.2 - Angepasste Kosinus Ähnlichkeit zweier Objekte i und j

Doch dieser Algorithmus der GES besitzt auch einige Nachteile. So müssen die angelegten Profile der Nutzer erhalten bleiben, was durchschnittlich ein Sicherheitsrisiko darstellen kann.

Ebenso ist die Skalierbarkeit dieses Algorithmus unzureichend. Dadurch entsteht eine hohe Anforderung an die Speicherressourcen und es hat Einfluss auf die Rechenzeit [HOF03].
Die modellbasierten Algorithmen hingegen benötigen nur einen Teil der getätigten Bewertungen der Nutzer. Aus diesen Bewertungen wird ein Modell erschaffen, welches dann eine Empfehlung für ein, vom Nutzer noch nicht betrachtetes, Objekt berechnet [SAR01]. Die „Clustering“ Methode ist ein übliches Verfahren bei der Nutzung der modellbasierten Algorithmen. Es überwindet das Problem der Skalierbarkeit der speicherbasierten Algorithmen, indem Nutzer und Objekte in Segmente gefiltert und somit nur ein Teil der gesamten Daten benötigt werden [GON10]. Nutzer werden dabei in Gruppen ähnlicher Bewertungen sortiert, so auch die Objekte.

2.2.1 Beispiel

In [LSY03] beschreiben die Autoren einen, von Amazon, selbsterstellten Algorithmus für ein gemeinschaftsbasiertes Empfehlungssystem. Dieses „Item-to-Item Collaborative Filtering“ kann Empfehlungen von Millionen von Artikeln in Echtzeit berechnen. Dazu vergleicht der unterliegende Algorithmus (siehe Abb. 2.3) ein einzelnes Objekt, nach bestimmten Kriterien, mit anderen Objekten. Dafür seien I_1 jedes Objekt im Produktkatalog, C jeder Käufer des Objektes I_1 und I_2 jedes Objekt, gekauft von C.

```
For each item in product catalog, I1
  For each customer C who purchased I1
    For each item I2 purchased by
       customer C
      Record that a customer purchased I1
       and I2
  For each item I2
    Compute the similarity between I1 and I2
```

Abb. 2.3 - Item-to-Item kollaborativer Filterungsalgorithmus

Seit dem 3. September 1995 gibt es eBay, das weltweit größte Internetauktionshaus. Wer einen Artikel ersteigern möchte, muss den anderen überbieten. Es gibt Millionen von Artikeln auch per Sofort-Kauf. Jeder, der bei eBay angemeldet ist, besitzt ein Bewertungsprofil [KLSS09]. „Jedes eBay-Mitglied hat ein Bewertungsprofil. Es ermöglicht allen Mitgliedern abzuschätzen, was sie bei einer Transaktion mit einem anderen Mitglied erwarten können. Mit einer positiven Bewertung können Sie anderen mitteilen, dass etwas gut gelaufen ist. Als Käufer können Sie auf diese Weise anderen Mitgliedern Verkäufer empfehlen, mit denen Sie gute Erfahrungen gemacht haben. Als Verkäufer haben Sie mit Bewertungen die Möglichkeit, Kunden zu ermuntern, wieder bei Ihnen einzukaufen. Das Bewertungsprofil informiert somit andere Mitglieder über die Zuverlässigkeit eines Handelspartners“ [EF12]. Hat beispielswei-

se ein eBay-Mitglied einen Kauf getätigt, darf und kann dieser danach eine Bewertung über den Verkäufer abgeben. Je nachdem, ob eine positive, neutrale oder negative Bewertung abgegeben wurde, bekommt der Verkäufer einen Punkt in seinem Bewertungspunktestand addiert, es bleibt unverändert oder es wird ein Punkt abgezogen. Während der Verkäufer nur eine positive Bewertung und einen kurzen Bewertungstext abgeben kann, hat der Käufer eine präzisere Möglichkeit den Verkäufer zu bewerten. So kann der Käufer den Verkäufer in verschiedenen vorgegebenen Kriterien bewerten. Dafür besteht die Option, wie in Abb. 2.4, für jedes Kriterium eine Punktzahl von eins bis fünf, in Form von Sternen, anzugeben. Dabei bedeuten fünf Sterne „sehr gut“ und 1 Stern „sehr schlecht“. Durch die getätigte Bewertung verändert sich die Gewichtung des Verkäufers. Je besser die Bewertungen eines Verkäufers sind, desto eher werden dessen Objekte empfohlen [BS97].

Abb. 2.4 - Ebay Verkäuferbewertung

Haben im Beispiel eBay viele Nutzer den Verkäufer mit sehr guten Bewertungen ausgezeichnet, so steigt seine Gewichtung. Der Verkäufer wird dann mit einem Titel ausgezeichnet, der ihm den Vorteil bringt, seine Waren als einer der ersten vorzustellen. Hat der Verkäufer einen hohen Rang, so erhält dieser den Titel „Verkäufer mit Top-Bewertung“. Sucht ein Käufer nun ein Objekt, werden alle Objekte von allen Verkäufern analysiert. Anschließend werden die Objekte mit einer hohen Übereinstimmung dem Nutzer angezeigt, wobei die „Verkäufer mit Top-Bewertung“ zuerst dargestellt werden.

2.2.2 Vor- und Nachteile

Die GES können einige Nachteile der IES überwinden, wie die Limited Content Analysis oder auch die Overspecialization. Allerdings besitzt auch die GES Nachteile, die es zu beachten gibt. Darunter zählt auch das bereits im IES beschriebene new User Problem. Weitere Nachteile sind das „new Item“- und das „Sparsity“-Problem. Ersteres zeigt auf, dass Objekte nur vorgeschlagen werden können, wenn diese bereits Bewertungen besitzen. Letzteres wiederum ist dicht mit dem new Item Problem verbunden. So besitzt ein Objekt beispielsweise

schon Bewertungen, jedoch reichen diese evtl. nicht aus, um eine präzise Empfehlung zu berechnen.

2.3 *Hybride Empfehlungssysteme*

In [AT05] wird beschrieben, dass mittlerweile mehrere Empfehlungssysteme den bekanntesten hybriden Ansatz nutzen, indem sie die inhaltsbasierte und gemeinschaftsbasierte Methode kombinieren. Dies soll dabei helfen, die von den beiden grundlegenden Methoden ausgehenden Probleme zu eliminieren. Diese Probleme können dabei unter anderen die Seltenheit eines Objektes, der eingeschränkte Umfang, die Skalierbarkeit oder auch das Cold-Start-Problem [AT05][BUR02] sein. Letzteres beinhaltet die bereits erklärten Nachteile des „New User"- und „New Item"-Problem. Inhaltsbasierte Methoden können zum Beispiel das „New-Item"-Problem überwinden, indem sie textuelle Beschreibungen der Artikel nutzen und diese dann vergleichen. Reine inhaltsbasierte Empfehlungssysteme benötigen wiederum mehr zusätzliche Informationen über den Inhalt von Artikeln (z.B. Zugehörigkeit zur Musik, zu Filmen, zu Restaurants, etc.).
Unter den hybriden Empfehlungssysteme gibt es verschiedene Methoden für die Empfehlungsberechnung, wie „weighted", „switching", „mixed", „Feature Combination" und „Meta – Level".

Weighted

Bei dieser Methode wird das Ergebnis eines empfohlenen Artikels durch alle verfügbaren Empfehlungstechniken berechnet. Das P-Tango System z.B. nutzt gemeinschaftsbasierte und inhaltsbasierte Techniken. Zuerst haben Beide eine gleiche Gewichtung, aber diese wird schrittweise angepasst, da sich Vorhersagen über Nutzerbewertungen bestätigen oder nicht bestätigen. Ein weiteres Beispiel, das diese Methodik nutzt, ist das kombinierte Hybridsystem von Pazzani. Dieses System verwendet sowohl die inhaltsbasierte, gemeinschaftsbasierte, als auch die demographische Technik. Dabei gebraucht das System jede Empfehlungstechnik als einen Satz von Bewertungen, die dann im Anschluss kombiniert werden.

Switching

In diesem Ansatz wird je nach Situation entschieden, welche Empfehlungstechnik genutzt werden soll. Das „Daily Learner" System z.B. bringt eine Kombination aus inhaltsbasierter und gemeinschaftsbasierter Technik zum Einsatz. Zuerst wird die inhaltsbasierte Methode genommen. Ist diese nicht effizient genug, dann wird die gemeinschaftsbasierte Technik verwendet. Tran & Lohan nutzen in ihrem Ansatz die vergangenen Bewertungen der Nutzer und die Empfehlungen jeder verfügbaren Technik, um für die nächsten Empfehlungen die beste Technik auszuwählen und anzuwenden.

Mixed

Werden Empfehlungen von verschiedenen Empfehlungsdiensten zur gleichen Zeit ausgegeben, dann wird die „mixed“-Methode verwendet. Das PTV-System z.B. nutzt textuelle Beschreibungen von beispielsweise TV-Shows, was die inhaltsbasierte Technik darstellt. Zur selben Zeit wird die gemeinschaftsbasierte Technik genutzt, um die Vorlieben anderer Nutzer zu verwenden. Diese Kombinierung der beiden Techniken bringt den Vorteil, das „new-Item“ Problem zu überwinden, jedoch das „new User“-Problem bleibt bestehen.

Feature Combination,

Hier werden die Eigenschaften verschiedener Empfehlungen zusammen in einem Empfehlungsalgorithmus eingebunden. Das System „Ripper“ zieht dabei die getätigten Bewertungen von Nutzern und inhaltliche Eigenschaften der Artikel heran. Die Artikel werden auf die Anzahl der Nutzer reduziert, die eine Bewertung abgegeben haben. Durch den gemeinschaftsbasierten Ansatz können hier erhebliche Verbesserungen im Bereich der Genauigkeit erzielt werden.

Cascade

Wenn ein Empfehlungsanbieter die Empfehlungen von einem andern Anbieter verfeinert, spricht man von der „Cascade“-Methode. EntreeC ist ein Restaurant Empfehler und nutzt die „knowledge-based“ und gemeinschaftsbasierte Technik. Verwendet werden dabei das Wissen über Restaurants und Nutzervorlieben für neue Empfehlungen. Hierbei werden Empfehlungen, die gleiche Vorlieben darstellen kombiniert. Die gemeinschaftsbasierte Technik bricht dabei die Empfehlungen wieder auf und sortiert sie in eine Rangfolge.

Feature Augmented

Diese Methode nutzt die ausgegebenen Empfehlungen der einen Technik als Eingabe für eine andere Technik. Das Libra System ermöglicht die inhaltsbasierte Empfehlung von Büchern aufbauend auf Textdaten, die bei Amazon gefunden werden. Diese Daten beinhalten z.B. verwandte Titel, Genres oder auch Autoren, die durch die gemeinschaftsbasierte Methode ermöglicht werden.

Meta-Level

Das Meta-Level nutzt ein ganzes Modell, welches von einem Empfehlungsdienst lernt, als Eingabe für einen anderen Dienst. Ein Beispiel dafür ist das von Pazzani beschriebene „collaboration via content“ oder auch LaboUr[SKK01]. Letzteres nutzt instanz-basiertes Lernen

um inhaltsbasierte Profile zu erstellen, die dann in gemeinschaftsbasierter Art und Weise verglichen werden.

Beispiele

Das, die Switching Methode nutzende, System „AdaRec“ [AB10] verwendet Eigenschaften der Artikel und Nutzerbewertungen, um seine Schaltstrategie je nach Leistung der Vorhersagetechnik umzuschalten. Durch die Kombination aus inhaltsbasierten und gemeinschaftsbasierten Algorithmen können sehr gute Resultate erzielt werden. Schaltstrategien sind anpassbar und sollen ermitteln, welche Vorhersagetechnik für die nächste Empfehlung am besten geeignet ist. Vorhersagetechniken können z.B. „user Average“, CBR (Case Based Reasoning) oder auch topNDeviation sein. In Abb. 2.5 wird dies veranschaulicht. Die grau eingefärbten Ovale sind dabei die Vorhersagealgorithmen und die Pfeile repräsentieren die Attribute und Schwellwerte.

.

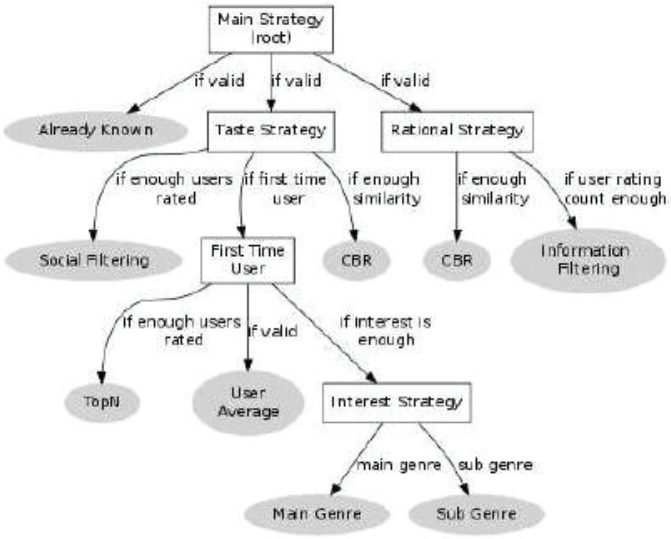

Abb. 2.5 - Entscheidungsbaum zur Auswahl einer Vorhersagestrategie

Das Bild zeigt ein Beispiel, welche Vorhersagetechnik in welcher Situation verwendet wird. Im AdaRec-System hat jeder Nutzer ein Nutzerprofil, das wichtige Informationen beinhaltet, wie z.B. Vorlieben oder Feedbacks. Das System besteht aus zwei Teilen:

- Recommender Engine
- Learning Module

Die Recommender Engine ist verantwortlich für die Vorhersagen und nutzt dabei das Nutzerprofil und die Inhalte des aktuellen Artikels. Darauf beruhend erfolgt eine Vorhersage je nach Vorhersagestrategie. Das „Learning Module“ erstellt hingegen neue Vorhersagestrate-

gien auf Grundlage der vorherigen Leistungsergebnisse der Vorhersagetechniken. Dadurch können wiederum neue Entscheidungsbäume entstehen.

Das Fab System ist ebenso ein hybrides Empfehlungssystem und ist der meta-level Methode zuzuordnen. Dieses kann in zwei Bereiche eingeteilt werden. Als erstes werden Artikel gesammelt um eine Datenbank aufzubauen und als zweites die auf die Datenbank aufbauende Auswahl von Artikeln für ähnliche Nutzer [BS97]. Die umgesetzte Architektur besitzt drei Hauptkomponenten. Der „Collection Agent“ ist dafür zuständig Webseiten zu finden, die zu einem bestimmten Thema passen. Der „Selection Agent“ wiederum sucht Webseiten, die zu einem bestimmten Nutzer passen. Das Profil der ersteren Methode stellt dabei die aktuellen Themen dar, während das Profil des Letzteren die Interessen eines einzelnen Nutzers darstellt. Die letzte Komponente ist der Router. Dieser nimmt die vom „Collection Agent“ gefundenen Webseiten entgegen und reicht sie zu den Nutzern weiter, die dazu passen. Die Abb. 2.6 zeigt einen Überblick über diese Architektur. Fab bringt erhebliche Vorteile im Vergleich zu reinen inhaltsbasierten oder gemeinschaftsbasierten Systemen:

- Nutzung der Erfahrung anderer Anwender
- Nutzung der inhaltsbasierten Methode, um Artikel, die von anderen Anwendern noch nicht gesehen wurden, vorzuschlagen
- Nutzung des Profils um gute Empfehlungen zu erstellen, trotz ungleicher Nutzer
- Artikel können herausgefiltert werden
- Empfehlungen können ausgegeben werden, selbst für Nutzer, die nicht dieselben Artikel bewertet haben

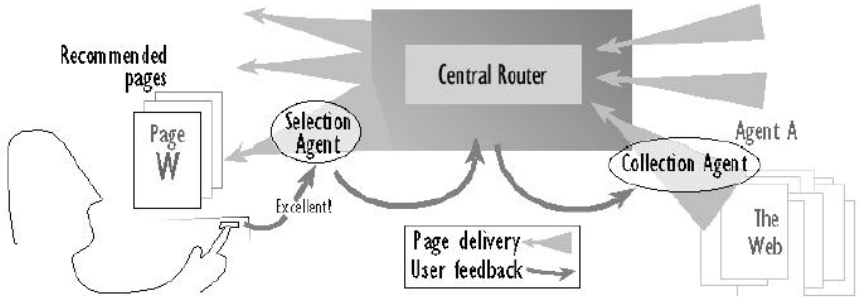

Abb. 2.6 - Überblick über die Fab-Architektur

2.4 *Methodik zur Datenerfassung*

Um für einen Nutzer Empfehlungen berechnen zu können, benötigt es Daten, welche dies ermöglichen. Das Sammeln von Daten ist in zwei verschiedenen Methoden möglich. So gibt es die Variante des expliziten und des impliziten Feedbacks.

2.4.1 Explizites Feedback

Die Variante des expliziten Feedbacks sammelt die Bewertungen von Objekten durch die direkte und bewusste Eingabe eines oder mehrerer Nutzer. Dabei gibt es verschiedene Verfahren, wie Bewertungen angegeben werden können. Die zwei bekanntesten und gebräuchlichsten Methoden sind das Betätigen eines „gefällt mir“ oder „gefällt mir nicht“-Schalters und das Verteilen von Punktzahlen. Ersteres wird beispielsweise von Chip.de [CHIP12] verwendet, während Letzteres unter anderen Online-Shops, wie eBay [EBAY12], nutzen. Durch eine Analyse mehrerer Bewertungen ist es möglich, für ein Objekt eine gemittelte Bewertung zu berechnen. Daraus wiederum kann im Vergleich mit anderen Objekten eine Empfehlung berechnet werden.

2.4.2 Implizites Feedback

Das implizite Feedback sammelt Informationen von Nutzern aus dem Hintergrund, sodass eine direkte Eingabe nicht nötig ist. Diese Variante reflektiert also die Meinung der Nutzer durch ihr beobachtetes Verhalten und ist daher nicht der robusteste Weg der Informationssammlung[HKV08]. So ist beispielsweise negatives Feedback nur zu vermuten. Wenn z.B. ein Mensch eine TV-Show nicht gesehen hat, dann kann das mehrere Gründe haben. Derjenige könnte die Show nicht mögen, wusste nichts von dieser oder er hatte keine Zeit, sich die Show anzusehen. Auch positives Feedback, das implizit erfasst wird, kann nicht eindeutig bestimmt werden. Ein gekauftes Objekt könnte als Geschenk gekauft worden sein oder der Käufer war eventuell unzufrieden mit der Ware. Allerdings lassen gewisse Aktionen eines Nutzers auf eine positive oder negative Reaktion deuten [HB10]. Aktionen wie drucken, Lesezeichen setzen, häufiges ansehen oder auch das lange Verbringen auf einer Seite und das Kopieren von Textpassagen können unter Anderen ein positives Interesse aufweisen. Das abrupte Schließen einer Seite oder auch, eine kurze Weile auf einer Seite zu verbringen kann wiederum negatives Interesse bedeuten.

2.4.3 Beispiele

Robin van Meteren und Maarten van Someren präsentieren in ihrem Artikel [MS00] ein inhaltsbasiertes Empfehlungssystem, welches dynamische Hyperlinks für eine Webseite erstellt, um es dem Nutzer einfacher zu machen, interessante Artikel zu finden. Das beinhaltende Nutzermodell ist sehr dynamisch und lernt nur von positiven Stellungnahmen. Wenn ein Nutzer das Web durchsucht, entsteht dadurch eine Sammlung von Informationen, die die Interessen des Anwenders darstellen. Das „Personalized Recommender System“, kurz PRES, empfiehlt Dokumente, welche textuelle Informationen über „Do it yourself“-Heimverbesserungs-Tipps beinhaltet. Dabei beachtet das System, dass für die Nutzer ein bestimmtes Thema nur von kurzer Zeit von Bedeutung ist. Das bedeutet, dass sich die Inte-

ressen des Besuchers der Seite auf längere Sicht ändern können. Als Fazit daraus, haben die Entwickler beschlossen das System sehr dynamisch zu machen. Aufgrund dessen laufen berechnete Bewertungen nach einer gewissen Zeit ab. Diese werden durch das implizite Feedback gesammelt, da explizite Resonanzen unvorteilhaft sind. Ein Nutzer würde ständig gebeten werden, eine Bewertung zu tätigen. Eine implizite Variante ist daher die beste Variante für das System und wird auch genutzt.

Wie bereits erwähnt, ist das explizite Feedback die robustere Methode, Informationen für Empfehlungen zu sammeln. So haben es Parra und Amatriain in [PA11] geschafft implizite Feedbacks auf Expliziten abzubilden. Somit können explizite Methoden mit impliziten Daten genutzt werden. Die Daten, die genutzt werden, können unter anderem die Anzahl der Betrachtungen sein, wie oft ein Titel gespielt wurde, oder auch, wie oft eine Seite während einer Sitzung besucht wurde. In [PKYA11] nutzen die Autoren die Ansätze aus ihrer vorigen Arbeit und verwenden sogenannte MAP[MRS08]-und nDCG[JK02] Metriken, um zu zeigen, dass diese neuentwickelte Methode vergleichbar ist mit den modernsten Methoden zur Verwendung vom impliziten Nutzer-Feedback.

Eine schon alte, aber immer noch bewährte Methode ist, darauf zu achten, wie lange ein Nutzer in einer Sitzung insgesamt auf einer Seite verbracht hat [KOR01]. Lee und Park haben sogar ein Empfehlungssystem entworfen, dass genau dies berücksichtig. In [LP08] entwickeln die Autoren eine Möglichkeit implizites Feedback als Pseudo-Bewertungsdaten zu nutzen und daraus eine Pseudo-Bewertungsmatrix zu erstellen. Sie nutzen dabei die Zeit, wann der Nutzer einen Artikel gekauft hat und wie lange sich der Nutzer diesen Artikel angesehen hat, um eine Entscheidung zu treffen. Der erste Schritt zu diesem angestrebten Ziel ist es, wie in Abb. 2.7 eine simple Pseudo Matrix zu erstellen. Diese zeigt, ob ein Artikel gekauft wurde, was mit einer „1“ dargestellt wird.

	Item 1	Item 2	Item 3	Item 4
User A	1		1	1
User B		1	1	
User C	1			1

Abb. 2.7 - Simple Pseudo Bewertungsmatrix, erstellt durch Kaufinformationen

Wird nun noch die Zeit mit einbezogen, wann ein Objekt ausgewählt und wann es gekauft wurde, so ergibt sich eine Nutzer-Objekt-Matrix, wie in Abb. 2.8. Dabei steht LTime für Launched Time (Zeit, wann ein Objektausgewählt wurde) und PTime für purchased Time (Kaufzeit nach Auswahl des Objektes) steht.

	Item 1 :LTime 1	Item 2 :LTime 2	Item 3 :LTime 3	Item 4 :LTime 4
User A	PTime 1		PTime 2	PTime 3
User B		PTime 4	PTime 5	
User C	PTime 6			PTime 7

Abb. 2.8 - Zugriffszeit eines Artikels und Zeit bis kauf durch Nutzer

Werden nun die beiden eben genannten Zeiten im selben Moment betrachtet, so können daraus Informationen gewonnen werden, ob der Nutzer diesen Artikel mag oder nicht. Sei w(pi, lj) der Bewertungswert für ein Objekt und einen Nutzer, wobei pi die verstrichene Kaufzeit eines Objektes darstellt und lj die Zugriffszeit [LP08]. Dabei entsteht „Pseudo-Rating-Matrix“, wie die in Abb. 2.9.

	Item 1	Item 2	Item 3	Item 4
User A	w(p1,l1)		w(p2,l3)	w(p3,l4)
User B		w(p4,l2)	w(p5,l3)	
User C	w(p6,l1)			w(p7,l4)

Abb. 2.9 - Pseudo Bewertungsmatrix unter Berücksichtigung der Zugriffszeit des Artikels und der Zeit bis zum Kauf eines Objektes

Um Empfehlungen zu berechnen durchläuft ihr System dabei 4 Phasen:

- Sammeln von impliziten Feedbacks
- Erstellen einer Pseudo – Bewertungsmatrix
- Ähnliche Artikel finden
- Artikel empfehlen

Im letzten Schritt werden dann die Top n Artikel empfohlen, siehe Abb. 2.10. PS(a, j) steht dabei für die Funktion zur Berechnung des Punktestands eines Nutzers a und dem Artikel j. Die Ähnlichkeit zweier Nutzer a und t sei definiert als sim(a,t). Außerdem ist P_{tj} die Bewertung des Artikels j von einem Nutzer t, während $\bar{P_t}$ die durchschnittlich getätigten Bewertungen des Nutzers t sei.

$$PS(a,j) = \frac{\sum_t sim(a,t)(P_{tj} - \overline{P_t})}{\sum_t sim(a,t)}$$

Abb. 2.10 - Vorausberechnetes Ergebnis eines Nutzers a und Objektes j

Zusammenfassend ist zu sagen, dass die explizite Methode, Informationen zu sammeln die genauere und sicherste Methode ist, Empfehlungen zu berechnen. Da dies aber nicht immer möglich ist und Nutzer meist nicht gewillt sind ein Feedback abzugeben, greift man auf die implizite Methode zurück, die vielleicht nicht so exakt ist, aber dennoch hinreichende Daten zur Verfügung stellt. In der Abb. 2.11 soll dies verdeutlicht werden.

	Implicit feedback	Explicit feedback
Accuracy	Low	High
Abundance	High	Low
Context-sensitive	Yes	Yes
Expressivity of user preference	Positive	Positive and Negative
Measurement reference	Relative	Absolute

Abb. 2.11 - Charakteristiken vom impliziten und expliziten Feedback

3 Konzeption

In diesem Kapitel werden zunächst die Voraussetzungen und Anforderungen an ein Empfehlungssystem beschrieben. Dabei wird auf die im aktuellen Forschungsstand beschriebenen Empfehlungssysteme und die Methodik zum Sammeln von Daten eingegangen. Der darauffolgende Abschnitt beschreibt die genutzten Funktionen verschiedener Empfehlungssysteme. Nach dem Abschnitt der Funktionen wird auf die Eingabe der Werte eingegangen. Im letzten Abschnitt erfolgt eine Zusammenfassung und Bewertung.

Im Rahmen des MEMO-Projektes soll ein Empfehlungssystem für Community Inhalte prototypisch umgesetzt, sowie ein Werkzeug zur Konfiguration und Überprüfung der Ergebnisse entwickelt werden. Aus einem Protokoll eines Projekttreffens [IGD11] geht hervor, dass die Bereitschaft Inhalte zu bewerten, oder zu erstellen, gering ist. Dies wurde im Laufe der ersten 18 Monate des MEMO-Projektes festgestellt. Ziel ist es, zu untersuchen, ob ein Empfehlungssystem für Community Inhalte entwickelt werden kann, das ohne explizite Bewertungen der Nutzer auskommt.

Das Memo-Projekt beschäftigt sich mit Online Lern- und Kollaborationsdiensten. Die Zielgruppen sind Handwerker und Experten des KFZ-Handwerks und der Elektromobilität, die Informationen austauschen und darüber in einer Community diskutieren wollen. Auch andere Online-Portale und Kollaborationsdienste können Hilfe und interessante Anwendungen in diesem Projekt finden. „Die MEMO Lern- und Kollaborationsdienste sollen in Wissens- und Bildungsangebote von Bildungsanbietern, Berufsverbänden und Kammern, in Firmenportale und weitere Internet-Portal- und Kollaborationsdienste eingebunden werden können, um dort Qualifizierungsangebote zu ergänzen und um sie mit modernen, aktuellen, nutzerorientierten Diensten zu erweitern“ [MEM12]. Dies wiederum hat zum Einen den Vorteil, dass große Zielgruppen erreicht werden, welche sich über Neuerungen informieren und zum Anderen stellt dies die nachhaltige Nutzung von MEMO-Diensten sicher. Die genutzten Dienste sollen den Nutzern beim Lernen, Zusammenarbeiten und Austauschen von Informationen helfen. Um beispielsweise die Kenntnisse über Hochvolt-Systeme zu erweitern, werden immer neue Methoden zum Austausch von Informationen entwickelt.

3.1 *Überblick*

Anforderungen

Es ist ein geeignetes Empfehlungssystem zu entwickeln, das Empfehlungen für eine Seite ausgeben soll. Durch die Empfehlungen soll erreicht werden, dass den Nutzern interessante Seiten vorgeschlagen werden. Sucht ein Nutzer zu lange nach einer hilfreichen Seite, so

verschwendet er wichtige Arbeitszeit. Da das MEMO-Projekt Liferay als Community Portal nutzt, werden das Empfehlungssystem und dessen Visualisierung als Portlet implementiert. Die durch das Portlet entstehenden Empfehlungswerte sollen die Suchzeit nach informativen Seiten verkürzen.
Dabei ist zu beachten, dass dieses System nur implizit Informationen von Nutzern sammeln kann. Dafür muss eine Funktion entwickelt werden, die diese Informationen verwendet und daraus einen Empfehlungswert berechnet. Zur Beobachtung des Nutzerverhaltens soll das Empfehlungssystem eine geeignete Ausgabe ermöglichen. Eine hohe Anzahl von Nutzern und Seiten muss dargestellt werden können. Das Werkzeug zur Untersuchung der Funktionsweise des Empfehlungssystems soll so einfach wie möglich zu verstehen und anzuwenden sein. Um eine geeignete Untersuchung zu ermöglichen, soll dieses Werkzeug das Ergebnis verschiedenster Parameterkonfigurationen visualisieren können.

Inhaltliches System

Im aktuellen Forschungsstand ist beschrieben, dass es zwei grundlegende Empfehlungssysteme gibt (inhaltlich, gemeinschaftsbasiert).Werden diese Beiden vereint, entsteht ein hybrider Ansatz eines Empfehlungssystems. Ein rein inhaltlich basiertes Empfehlungssystem benötigt viele inhaltliche Informationen. So sucht dieses System beispielsweise nach ähnlichen Seiten, die der Nutzer in der Vergangenheit gut bewertet hat. Sind die Informationen nicht vorhanden oder nur gering vertreten, so leidet auch die Präzision. Da die Präzision für diese Arbeit zu gering und dieses System zur Untersuchung verschiedener Parameterkonfigurationen alleine nicht tauglich ist, wird diese Art von Empfehlungssystem nicht verwendet.

Gemeinschaftsbasiertes System

Im Gegensatz zu einem inhaltlichen System bezieht sich das gemeinschaftsbasierte System auf die Ähnlichkeit der Nutzer. In Bezug auf das zu erstellende System, können Empfehlungswerte für Seiten berechnet werden, die im Profil ähnlicher Nutzer vorhanden sind. Wie bereits erwähnt, sollen aber verschiedene Parameterkonfigurationen getestet werden. Ein rein gemeinschaftsbasiertes Empfehlungssystem hat in Bezug auf diese Arbeit Nachteile, wie das new Item-Problem, welches es unpräzise macht. Ein reiner Ansatz für diese Arbeit ist somit nicht denkbar.

Hybrides System

Ein hybrider Ansatz eines Empfehlungssystems würde die zwei beschriebenen Empfehlungssysteme vereinen. Durch die Nutzung dieses Systems können mehrere Funktionen vereint werden. So können verschiedenste inhaltlich- und gemeinschaftsbasierte Funktionen

verwendet werden, um das Ergebnis unterschiedlicher Parameterkonfigurationen zu visualisieren. Die Forschung im Bereich hybrider Empfehlungssysteme hat festgestellt, dass es durch die Verbindung von inhaltlichen und gemeinschaftsbasierten Systemen zu einem hybriden System möglich ist, den Nachteilen der einzelnen Systeme entgegenzuwirken [AT05][BUR02]. Im aktuellen Forschungsstand wurde aufgezeigt, dass es verschiedene hybride Methoden gibt. Jede dieser Methoden besitzt Eigenschaften, die es für das Werkzeug nützlich machen oder nicht. Die „Switching“ Methode ist denkbar ungeeignet. Sie wählt je nach Situation eine Empfehlungstechnik aus. Eine Kombination aus den beiden verwendeten Empfehlungsmethoden berechnet Empfehlungswerte, die eine hohe Genauigkeit besitzen. Für diese Arbeit soll jedoch keine Empfehlungstechnik ausgewählt, sondern mehrere vereint werden. Der „Mixed“ Ansatz gibt hingegen mehrere Empfehlungswerte von verschiedenen Empfehlungsdiensten zur gleichen Zeit aus. Dies macht es jedoch für diese Arbeit unbrauchbar, da nur ein Empfehlungswert verwendet werden soll. Die „Feature Combination“ nutzt die Eigenschaften verschiedener Empfehlungen zusammen in einem Empfehlungsalgorithmus. Dabei wird nicht nur Wert auf die Nutzerähnlichkeit gelegt, sondern auch auf die abgegebenen Bewertungen der Nutzer. Da die Nutzer in dem zu entwickelnden System aber nicht explizit Bewertungen vornehmen und somit Daten nur implizit gesammelt werden können, ist diese Methode für einen hybriden Ansatz untauglich. Es sollen ebenso keine Empfehlungen von anderen Empfehlungsanbietern verfeinert werden, daher entfällt auch die „Cascade“ Methode. Das „Feature Augmented“ wiederum nutzt die ausgegebenen Empfehlungen der einen Technik als Eingabe für die andere Technik. Da allerdings nur wenig inhaltliche Informationen der Seiten vorhanden sind und die Eingabe für die andere Technik diese aber benötigt, ist diese Methode ungeeignet. Bei der „Meta-Level“ Methode wird ein Modell verwendet, welches von einem Empfehlungsdienst lernt. Dieses dient wiederum als Eingabe für einen anderen Dienst. Die „Weighted“ Methode nutzt alle verfügbaren Empfehlungstechniken. Das Ergebnis ist ein Empfehlungswert, welches durch das gewichtete Mittel der Funktionen und Parameter berechnet wird. Da im zu konzipierenden System mehrere Funktionen verwendet werden sollen, ist die Weighted Methode die beste Lösung für einen hybriden Ansatz eines Empfehlungssystems. Durch die Nutzung dieser Methode würde die gestellte Anforderung erfüllt werden.

Explizites und implizites Feedback

Damit jedoch überhaupt Daten für eine Empfehlungsberechnung vorhanden sind, benötigt es eine Methode, um diese zu sammeln. Diese Methoden wurden ebenfalls aktuellen Forschungsstand erläutert. In Bezug auf Empfehlungen, wäre die beste Lösung immer noch das explizite Feedback, da es eine hohe Genauigkeit besitzt und exakt das ausdrückt, was der Nutzer denkt. Der Nachteil dieses Feedbacks ist, dass die Nutzer gewissenhaft Bewertungen

abgeben müssten. Diese Arbeit soll jedoch herausfinden, ob ein Empfehlungssystem ohne explizite Bewertungen auskommt. So ist im Gegensatz zum expliziten Feedback das Implizite nicht so exakt und aufschlussreich. Dennoch wollen die Nutzer, ohne selbst einen großen Aufwand betreiben zu müssen, einen Artikel vorgeschlagen bekommen. Durch die zwei genannten Aspekte, ist das implizite Feedback die beste Lösung in dieser Community, um die benötigten Daten zu sammeln. Die aus dem Nutzerverhalten gesammelten Informationen sind keine hundertprozentige Sicherheit darüber, ob es wirklich ein positives Feedback ist, da hier keine direkte Bewertungseingabe durch den Nutzer erfolgt. Durch den Vorteil des Sammelns von Informationen im Hintergrund, kann der Nutzer jedoch Zeit bei der Suche nach hilfreichen Seiten sparen.

3.2 *Funktionen*

Die implizit gesammelten Daten sollen der Berechnung der Empfehlungen dienen. Für diese Arbeit werden drei Funktionen vorgestellt, die vom Verfasser selbst entwickelt wurden. Das Ergebnis jeder Funktion ist ein Wert zwischen null und eins. Sie bilden die Grundlage für das zu entwickelnde Werkzeug.

3.2.1 Ausgangsdaten

Im aktuellen Forschungsstand wurde schon beschrieben, dass einige Aktionen auf eine positive Stellungnahme hindeuten. So lassen folgende Beobachtungen auf eine positive Resonanz schließen:

- Eine lange Zeit auf einer Webseite zu verbringen
- Häufiges Betrachten derselben Seite
- Wurde etwas gedruckt
- Wurde etwas markiert
- Wurde etwas kopiert
- Lesezeichen setzen
- Schlagwortsuche

Im Gegensatz dazu kann auch auf eine negative Resonanz geschlossen werden:

- Eine Webseite nur selten ansehen
- Eine geringe Zeit auf einer Webseite verbringen
- Eine Kombination aus beiden

Die Ansätze, Aktionen zu beobachten, wie Drucken, Markieren, Kopieren oder wie lange ein Nutzer auf einer Seite verbringt, würde der Verfasser gerne einbringen. Diese Methoden der impliziten Überwachung würden einen besseren Rückschluss darüber geben, ob eine Seite interessant und zu gebrauchen ist. Allerdings ist es mit der Liferay Community Edition tech-

nisch nicht möglich diese Aktionen zu beobachten. Diese Version von Liferay stellt nur eine gewisse Anzahl an Funktionen bereit. Mögliche Aktionen in der Liferay Community Edition sind Lesezeichen setzen und auf Grund der getätigten Schlagwortsuche eine Seite zu betrachten. Dennoch sind dies optionale Möglichkeiten, da dies einerseits eine Aktivität des Nutzers erfordert und andererseits nicht jeder diese Funktion nutzt. Des Weiteren werden vor allem bei der Schlagwortsuche auch die Wörter gespeichert, die falsch eingetippt worden sind. In den Berechnungen für einen Empfehlungswert einer Seite wiederum werden dann auch die alten Schlagwörter genutzt, die nun aber keine Rolle mehr spielen könnten. Wird eine Seite viel besucht, lässt sich daraus schließen, dass die Resonanz positiv ist. Die Anzahl der Betrachtungen ist also eine Möglichkeit, die am besten Aufschluss darüber gibt, ob eine Seite interessant ist. Durch eine bereitgestellte Funktion kann jeder Besuch auf einer Seite gespeichert werden. Somit können diese Daten problemlos im Hintergrund gesammelt und in einem hybriden Ansatz eingebunden werden.

3.2.2 Häufigkeitsfunktion

Mit Liferay und den unterliegenden Funktionen ist es möglich eine Speicher-Funktionalität bereitzustellen. Im Laufe dieser Arbeit werden Daten aus dem Beobachten des Nutzerverhaltens gespeichert. Rechtlich gesehen ist es in Bezug auf den Datenschutz bedenklich und somit für den Einsatz in Unternehmen unvorteilhaft. Dies ist jedoch ein Testsystem, welches keine realen Nutzer beinhaltet.

Datenerfassung

Die Aktivität eines Nutzers in der Community kann verfolgt werden. So ist es möglich, die Besuche auf einer Liferay-Seite zu speichern und jedem Nutzer zuzuordnen. Somit kann ermittelt werden, wie oft ein Nutzer eine Seite betrachtet hat. In Bezug dazu kann auch ermittelt werden, welche der zur Verfügung gestellten Seiten insgesamt am meisten betrachtet wurde. Einige Nutzer haben den Wunsch eine Seite empfohlen zu bekommen, welche eine hohe Beliebtheit besitzt. Da allerdings die Nutzer keine explizite Bewertung abgeben, kann nicht so einfach erschlossen werden, ob eine Seite wirklich hilfreich ist. Zu erkennen ist auch, welche Seiten nur selten oder gar nicht betrachtet worden sind. Intuitiv ist davon auszugehen, dass häufig betrachtete Seiten interessanter sind, als selten betrachtete Seiten.

Die Häufigkeitsfunktion

Durch die gespeicherten Besuche eines Nutzers auf einer Seite, können viele Informationen gewonnen werden. Es kann ein Bezug zu der Anzahl eines jeden Besuchs auf einer Seite und der Anzahl der Besuche auf der Seite, die am meisten besucht wurde, hergestellt werden. In der folgenden, für das zu konzipierende System definierten, Häufigkeitsfunktion h(s)

einer Seite s, sind n der aktuelle Nutzer in der Liferay Community und N die Gesamtanzahl aller Liferay Nutzer. Die Seite s wird in der Liferay Community zur Verfügung gestellt. Die Anzahl der Betrachtungen einer Seite s von einem Nutzer n sei gegeben als AS(s, n) und die Anzahl der Betrachtungen der Seite, die in der Community am häufigsten betrachtet wurde ist hS, so ergibt sich folgende Formel:

$$h(s) = \frac{\sum_{n=1}^{N} AS(s,n)}{hS}$$

Beispiel

In der Liferay Community haben die Nutzer der Liferay Community die Seite s_1 insgesamt 18-mal betrachtet. Die Anzahl der Betrachtungen, der am häufigsten vertretenen Seite, beträgt 23. Somit ergibt sich die Rechnung:

$h(s_1) = \frac{18}{23} = 0.7826.$

Das Ergebnis ist ein Wert zwischen null und eins. Dieser vermittelt die Beliebtheit einer Seite. In diesem Fall besitzt die Seite s_1 eine hohe Beliebtheit. Im Vergleich dazu ist eine andere Seite s_2 8-mal betrachtet worden. Das Ergebnis, 0.3478, ist ein wesentlich kleinerer Wert. Die Seite s_2 weist im Vergleich zu der am meisten betrachteten Seite nur eine geringe Beliebtheit auf. Je geringer also die Anzahl der betrachteten Seite ist, desto geringer ist auch der Empfehlungswert. Dabei kann diese Anzahl niemals den Maximalwert übersteigen.

Vor- und Nachteile

Diese Funktion bezieht die implizit gesammelten Informationen ein. Obwohl ein Nutzer noch keinerlei Seiten betrachtet hat, kann dieser dennoch sehen, welche Seiten am häufigsten betrachtet wurden. Somit wird also dem new User Problem entgegengewirkt. Da präzise Empfehlungen aber erst ab einer gewissen Anzahl von Betrachtungen mehrerer Nutzer erfolgen können und neue Artikel nicht mit eingebunden werden, bestehen noch immer die Nachteile des new Item- und des Sparsity Problems. Selten betrachtete Seiten können jedoch auch interessant sein und somit vernachlässigt werden.

3.2.3 Aktualitätsfunktion

Datenerfassung

Durch das Setzen eines Zeitstempels, sobald eine Seite in Liferay erstellt oder modifiziert wurde, kann die Aktualität einer Seite bestimmt werden. Dieser Zeitstempel beinhaltet sowohl das Erstellungs- als auch das Modifizierungsdatum. Über eine interne Abfrage kann diese Zeitdifferenz abgerufen werden. Intuitiv ist davon auszugehen, dass sich ein Nutzer mehr für aktuelle, als für ältere Seiten interessiert. Durch die Nutzung dieser Funktion kann

der Nutzer ältere Seiten zwar immer noch aufrufen, aber die neueren Seiten werden ihm empfohlen. Ein Grund dafür ist beispielsweise, dass ältere Seiten in Bezug auf den Fortschritt einer Technik nicht mehr aktuell sind.

Die Aktualitätsberechnung

Mit der Voraussetzung, dass es mehrere Seiten mit unterschiedlichen Zeitstempeln des Modifizierungsdatums gibt, kann ein Bezug der aktuell betrachteten Seite zu der Seite, die das älteste, bzw. niedrigste Datum besitzt, hergestellt werden. Das Ergebnis der Aktualitätsfunktion einer jeden in Liferay befindlichen Seite, kann sich vollkommen verändern. Dies geschieht, indem die Seite mit dem niedrigsten Modifizierungsdatum erneut bearbeitet wird. Durch das Bearbeiten dieser Seite wird dann ein neuer Zeitstempel gesetzt, der das aktuelle Datum darstellt. In der Liferay Community soll A(s) die Aktualität einer Seite s sein, die definiert ist durch folgende Variablen: s ist die Seite, für die ein Empfehlungswert berechnet werden soll. Diese wird in Liferay zur Verfügung gestellt. M(s) ist das Modifizierungsdatum der Seite s. Außerdem ist AD das aktuelle Datum und NMD das niedrigste modifizierte Datum, welches bereits am Anfang dieses Unterkapitels beschrieben wurde. Somit ergibt sich folgende Formel:

$A(s) = \frac{M(s)-NMD}{AD-NMD}$.

Beispiel

In der folgenden Tabelle 3.1 soll dargestellt werden, welche unterschiedlichen Werte Seiten bekommen können, die ein unterschiedliches Modifizierungsdatum besitzen. Dabei sei das aktuelle Datum AD der 18.08.2012 und NMD_1 das zuerst betrachtete niedrigste modifizierte Datum. Nach einer Bearbeitung, bzw. Aktualisierung der zu NMD_1 gehörenden Seite, ist das neue niedrigste modifizierte Datum NMD_2. Dieser Vergleich zeigt also auch, was mit den Empfehlungswerten

	NMD_1 = 14.06.2012	NMD_2 = 27.06.2012
1. Betrachtete Seite mit M(s) = 29.06.2012	A(s) = 0,1225	A(s) = 0,0349
2. Betrachtete Seite mit M(s) = 04.07.2012	A(s) = 0,1979	A(s) = 0,1178
3. Betrachtete Seite mit M(s) = 16.08.2012	A(s) = 0,9668	A(s) = 0,9635

Tabelle 3.1 - Unterschiede der Aktualität verschiedener betrachteter Seiten vor und nach der Aktualisierung der Seite mit dem niedrigsten modifizierten Datum

In der Tabelle 3.1 werden drei verschiedene Seiten aufgezeigt, die eine unterschiedliche Aktualität besitzen. Während die dritte betrachtete Seite mit 96.68% die aktuellste der drei Seiten ist, hat die zweite Seite nur eine Aktualität von 19,79% und die Erste sogar nur 12,25%. Die betrachteten Seiten beziehen sich auf das niedrigste zu findende Datum NMD_1. Die Seite, zu der NMD_1 gehört soll nun modifiziert werden und befindet sich in Bezug auf die Aktualität, an der höchsten Stelle. Dies bedeutet, dass nun eine andere Seite das niedrigste modifizierte Datum besitzt, nämlich NMD_2. Durch diese Umstellung ist der Abstand der Seite mit dem niedrigsten modifizierten Datums geringer zu den restlichen Seiten, wodurch sich alle Werte (einige mehr, andere weniger) verändern. Die dritte Seite zum Beispiel verändert sich nur geringfügig, während die zweite Seite einen Abfall von knapp 8% verzeichnet und Seite eins sogar 9%. Dadurch ist die erste betrachtete Seite am dichtesten an der Seite mit dem niedrigsten modifizierten Datum, was die Aktualität angeht. Warum das modifizierte Datum genommen wird und nicht das Erstellungsdatum, soll eine ähnliche Tabelle 3.2 aufzeigen. Anstatt des $NMD_{\#}$ erscheint dort nun das Erstellungsdatum E(s) einer Seite s. Dieses wiederum wird in der Formel anstelle des NMD eingesetzt. Das aktuelle Datum wird dabei der 18.08.2012 bleiben. Die für diese Berechnung entstandene Formel ist diese:
$A(s) = \frac{M(s) - \mathrm{E(s)}}{AD - \mathrm{E(s)}}$.

In Bezug auf das Erstellungsdatum ist zu erkennen, dass die Ergebnisse immer kleiner werden, je dichter sich das modifizierte Datum am Erstellungsdatum befindet. Je weiter sich dieses Datum also vom Erstellungsdatum entfernt, desto größer wird auch das Ergebnis. In Bezug auf die Aktualität einer Seite ist dies also keine nutzbare Funktion, da genau das Gegenteil benötigt wird. Des Weiteren lässt sich nun kein Bezug zu anderen Seiten herstellen, da nur die eigenen Werte der Seite genommen werden. Da aber zu erkennen sein soll, wie aktuell eine Seite in Bezug auf andere Seiten ist, wird das niedrigste modifizierte Datum verwendet.

	E(s)	A(s)
1 Betrachtete Seite mit M(s) = 29.06.2012	20.06.2012	0,1248
2 Betrachtete Seite mit M(s) = 04.07.2012	22.06.2012	0,2016
3 Betrachtete Seite mit M(s) = 16.08.2012	06.08.2012	1.1544625711435828E-9

Tabelle 3.2 - Nutzung des Erstellungsdatum, anstatt des niedrigsten modifizierten Datums

Vor- und Nachteile

Der Vorteil dieser Funktion ist, dass erkannt werden kann, welche Seite neu im Bezug zu anderen Seiten ist. Dies ist ein Ansatz zur Lösung des new User Problems. Da die ganz neuen Seiten mindestens einmal betrachtet werden müssen, um sie anderen Nutzern vorschlagen zu können, ist das Problem jedoch nicht ganz behoben.

3.2.4 Nutzerähnlichkeitsfunktion

Datenerfassung

Im Bereich der Empfehlungssysteme ist die Nutzerähnlichkeit eine erfolgreiche Methode, um dem Nutzer genaue Vorschläge für Seiten zu berechnen. In dieser Arbeit ist ein Nutzer einem anderen ähnlich, wenn sie gemeinsame Seiten in der Datenbank aufweisen. Je mehr diese übereinstimmen, desto ähnlicher sind sie sich. Wenn die Nutzer nach einem gewissen Muster sortiert werden, lässt sich diese Ähnlichkeit feststellen. So hat der Nutzer im MEMO Projekt den Wunsch etwas über Elektromobilität zu erfahren und informiert sich über die Wartung von Elektroautos. Ein anderer Nutzer, der ähnliche Interessen in Bezug auf die Elektromobilität aufweist, hat diesen Artikel noch nicht gesehen. Durch das Ergebnis der Funktion für die Nutzerähnlichkeit kann ihm dann genau diese Seite, über die Wartung von Elektroautos, empfohlen werden. Auch in dem zu konzipierenden Empfehlungssystem kann diese Nutzerähnlichkeit verwendet werden. Voraussetzung ist, dass ein Nutzer sich bereits Seiten angesehen hat. Dafür dient das implizite Sammeln von Daten. Durch das Beobachten, welcher Nutzer welche Seite betrachtet hat und durch die Anzahl vorhandener Einträge im Nutzerprofil kann herausgefunden werden, was die Nutzer für eine Übereinstimmung besitzen. Intuitiv ist davon auszugehen, dass die Vorschläge für einen Nutzer bei einer geringen Ähnlichkeit nicht von großem Interesse sind. Ist die Übereinstimmung aber hoch, so kann darauf geschlossen werden, dass diesen Nutzer auch die Seiten interessieren könnten, die er noch nicht gesehen hat.

Die Bestimmung der Nutzerähnlichkeit

Durch das Speichern eines jeden Besuchs auf einer Seite im Profil eines Nutzers, ist es möglich die Nutzerähnlichkeit zu anderen Nutzern zu ermitteln. Es soll beispielsweise für den aktuellen Nutzer N_a, der Liferay Community, ein Empfehlungswert für die Seite s von allen bereits betrachteten Seiten berechnet werden. Dafür wird ermittelt, welcher Nutzer welche Seite wie oft betrachtet hat. Verglichen werden dabei der aktuelle Nutzer N_a mit allen anderen Nutzern. Der Nutzer, der im Augenblick zum Vergleich steht, soll mit N_i bezeichnet werden. Für die Ähnlichkeit$sim(N_a, N_i)$ der zwei Nutzer N_a und N_i wird die Anzahl gleich betrach-

teter Seiten durch das Maximum der betrachteten Seiten dieser beiden Nutzer geteilt. Das Ergebnis ist ein Ähnlichkeitswert für N_a in Bezug auf N_i.

$$sim(N_a, N_i) = \frac{Anzahl\ gleicher\ Seiten\ von\ N_a\ und\ N_i}{Maximum\ der\ betrachteten\ Seiten\ der\ Nutzer\ N_a\ und\ N_i}$$

Dieser Ähnlichkeitswert wird dann mit der Gesamtanzahl der Betrachtungen von s durch N_i multipliziert. Dadurch entsteht ein numerischer Wert, welcher die Wichtigkeit der Seite s vom Nutzer N_i in Bezug zu N_a ausdrückt. Dieser numerische Wert und die weiteren Übereinstimmungswerte in Bezug auf alle anderen Nutzer werden summiert. Mit der Gesamtanzahl der Betrachtungen aller Liferay Nutzer, für die aktuell betrachtete Seite, lässt sich dann ein Verhältnis berechnen.
Definiert wird eine Funktion SW(N_a, s) die den Ähnlichkeitswert für einen aktuellen Nutzer N_a und einer Seite s berechnet. Sei A(N_i, s) die bereits angesprochene Anzahl der Seite s im Profil eines Nutzers N_i und GA(s) die Gesamtanzahl der Betrachtungen der Seite s aller Nutzer, so ergibt sich folgende Formel:

$$SW(N_a, s) = \frac{\sum\big(sim(N_a, N_i) * \mathrm{A}(N_i, s)\big)}{GA(s)}$$

Beispiel

In Tabelle 3.3 soll gezeigt werden, welcher Nutzer sich welche Seite angesehen hat. Im Anschluss daran erfolgt eine Berechnung eines Empfehlungswertes für eine Seite in Bezug auf die Nutzerähnlichkeit. Dabei sind N_a der aktuelle Nutzer, N_1 ist der erste Nutzer in der Liferay Community und N_2 der Zweite. Das $s_\#$ ist jeweils eine Seite in Liferay, wobei # für die Ziffer der Seite stehen soll. Das "!" soll der Artikel sein, dessen Empfehlungswert im Anschluss berechnet wird. Jeder dargestellte Nutzer und jede dargestellte Seite besitzt einen numerischen Wert. Dieser ist die Anzahl, wie oft ein Nutzer eine Seite betrachtet hat.

	s_1	s_2	s_3	s_4	s_5
N_1	5	2	9		6
N_2	8		4	12	
N_a	!	5	8		2

Tabelle 3.3 - Nutzer-Seite-Matrix zur Veranschaulichung, welcher Nutzer welche Seite betrachtet hat

Zu erkennen ist, dass N_1 und N_a drei übereinstimmende Artikel haben und N_1 verschiedene Seiten betrachtet hat, ergibt insgesamt vier. Damit ergibt sich eine Nutzerähnlichkeit dieser beiden Nutzer von 0,75. Dieser Wert wird nun mit der Anzahl der Betrachtungen der vorzuschlagende Seite s_1 von N_1 multipliziert, also 0,75 * 5. Somit ergibt sich ein Wert von 3,75. Im Vergleich dazu hat der Nutzer N_2 nur eine Seite, die übereinstimmend ist mit N_a. Der Maximalwert verschieden betrachteter Seiten ist hier 3 und dadurch ergibt sich eine Nutzerübereinstimmung von 0,33 für diese beiden Nutzer. Wird dieser Wert nun mit der Anzahl der Betrachtungen für die vorzuschlagende Seite, also 8, multipliziert, so ergibt sich der Wert von 2,64. Die beiden Ergebnisse, 3,75 und 2,64, werden summiert und im Anschluss durch die Gesamtanzahl der Betrachtungen, aller Nutzer für s_1, geteilt. Da die Gesamtanzahl der Betrachtungen für s_1 13 ist, ergibt sich ein Empfehlungswert von 0,4915. Dadurch kann ausgesagt werden, dass die Seite s_1 in Bezug auf die Nutzerähnlichkeit nur eine mittelmäßige Relevanz für N_a besitzt.

3.2.5 Gewichtete Empfehlungswertfunktion

Im Rahmen dieser Arbeit soll ein Werkzeug entstehen, welches die Möglichkeit bietet, verschiedenste Parameterkonfigurationen zu testen. Dafür wird eine Funktion benötigt, welche die zuvor beschriebenen Funktionen logisch zusammenfasst.
Durch das Zusammenfassen der Häufigkeits-, Aktualitäts- und Ähnlichkeitsfunktion entsteht ein hybrider Ansatz eines Empfehlungssystems. Im Überblick der Konzeption wurde bereits erläutert, dass für den hybriden Ansatz, die Weighted-Methode verwendet werden soll. Dabei wird jeder verwendeten Funktion eine eigene Gewichtung zugeordnet. Durch das Setzen dieser Gewichte für jede vorgestellte Methode und das Teilen durch die Summe der Gewichte entsteht die Gewichtungsfunktion.

Die gewichtete Empfehlungswertfunktion

Die Funktion hat als Ergebnis das gewichtete Mittel der drei Funktionen. Dieser Wert liegt ebenfalls zwischen null und eins. Dabei ist zu beachten, dass die Summe aller Gewichte größer als null sein muss und die einzelnen Gewichte größer oder gleich null sein sollen. Die hier definierte Empfehlungswertfunktion $EW(N_a, s)$ berechnet den Empfehlungswert für einen Nutzer N_a aller Nutzer M und einer Seite s. Diese Funktion ist definiert durch F_n , welches die verwendete Funktion darstellt und g_n ist die zugehörige, vom Nutzer festgelegte, Gewichtung für die verwendete Funktion F_n.

$$EW(N_a, s) = \frac{\sum_{n=1}^{M} (g_n * F_n)}{\sum_{n=1}^{M} g_n}$$

In Bezug auf die drei genannten Funktionen würde die Funktion also so aussehen:

$$EW(N_a, s) = \frac{Gewicht_h * h(s) + Gewicht_A * A(s) + Gewicht_{SW} * SW(n, o)}{Gewicht_h + Gewicht_A + Gewicht_{SW}}$$

Beispiel

Um diese Funktion noch weiter zu verdeutlichen, soll das folgende Beispiel dienen. So soll in einer Nutzer-Seiten-Matrix für eine Seite eine Empfehlung vorgeschlagen werden. Dafür soll einem Nutzer der Liferay Community für eine in Liferay befindliche, ausgewählte Seite ein Empfehlungswert berechnet werden. In diesem Beispiel soll ein Vergleich zwischen dem Ergebnis einer einzeln genutzten Funktion und einer Mischung aus allen Funktionen aufgezeigt werden. Legt ein Nutzer zum Beispiel nur Wert auf aktuelle Seiten, so sind folgende Gewichte für die Berechnung zu wählen:

- Häufigkeit = 0
- Aktualität = 1
- Nutzerähnlichkeit = 0

Der aktuelle Nutzer N_a und die aktuelle Seite s besitzen für den Vergleich beispielsweise folgende Werte:

- Häufigkeit (s) = 0,15
- Aktualität(s) = 0,65
- Nutzerähnlichkeit (N_a) = 0,2

Durch diese Werte ergibt sich die folgende Rechnung:

$$EW(N_a, s) = \frac{0 * 0{,}15 + 1 * 0{,}65 + 0 * 0{,}2}{0 + 1{,}0 + 0} = 0{,}65$$

Zu erkennen ist eine Aktualität von 65% für die ausgewählte Seite s. Ändern sich nun aber die Belange des Nutzers und dieser legt etwas Wert auf die Häufigkeit einer Seite und vor allem auf eine hohe Nutzerähnlichkeit, so sind die Gewichte beispielsweise folgendermaßen definiert:

- Häufigkeit = 0,35
- Aktualität = 1
- Nutzerähnlichkeit = 0,85

Werden diese Gewichte miteinbezogen, so entsteht diese Rechnung:

$$EW(N_a, s) = \frac{0{,}35 * 0{,}15 + 1 * 0{,}65 + 0{,}85 * 0{,}2}{0{,}35 + 1{,}0 + 0{,}85} = 0{,}3966$$

Zu erkennen ist ein Abfall des Empfehlungswertes. Dadurch, dass der aktuelle Nutzer eine allgemein niedrige Nutzerübereinstimmung mit allen anderen Nutzern besitzt und dieser genau darauf Wert legt, verringert sich der Empfehlungswert. Selbstverständlich spielt auch die Häufigkeit einer Seite eine Rolle, jedoch fällt diese nicht sehr ins Gewicht. Durch diese Funktion ist es möglich mehrere Empfehlungsfunktionen zu nutzen und verschiedene Tests durchzuführen. Nun fehlt lediglich ein Werkzeug, welches es dem Benutzer ermöglicht die Gewichte selbst zu setzen und das Ergebnis zu betrachten.

Vor- und Nachteile

Durch die Nutzung dieser gewichteten Empfehlungswertfunktion ist es möglich einen hybriden Ansatz für ein Empfehlungssystem zu erstellen, das nicht auf explizite Bewertungen der Nutzer angewiesen ist. Werden mehrere Empfehlungsfunktionen zur selben Zeit verwendet, so wird beispielsweise, dem new User Problem entgegengewirkt, indem Seiten vorgeschlagen werden, die eine hohe Beliebtheit oder Aktualität aufweisen.

3.3 *Eingabe der Werte*

Damit die Ergebnisse der Funktionen unterschiedlich untersucht werden können, wird ein Werkzeug benötigt. Dieses soll jeder Funktion ein Gewicht zuweisen, wodurch das Ergebnis für einen Empfehlungswert verändert wird. Die Gewichte für die einzelnen, bereits vorgestellten Funktionen, können nur umständlich durch Verändern des Programmcodes verändert werden. Daher sollte es eine Möglichkeit geben, dass der Benutzer dieses Werkzeuges, die Gewichte in einer geeigneten und einfachen Form einstellen kann. Somit können verschiedenste Ergebnisse überprüft werden, die wiederum in einer verständlichen und schnell einzuschätzenden Art, visuell dargestellt werden sollen.

3.3.1 Eingabemöglichkeiten

Um die Gewichte einstellen zu können muss eine Eingabemöglichkeit ausgewählt werden. Folgende Möglichkeiten stehen dabei zur Auswahl:

- Ein Eingabefeld, um einen Wert selbst einzutippen
- Einen Schalter, um die Gewichte Schritt für Schritt zu setzen
- Ein Aufklappmenu mit vorgegebenen Werten
- Einen Wert durch einen Schieberegler setzen

Der Benutzer des Werkzeuges hat mit einem Eingabefeld die Möglichkeit ein Gewicht für eine Funktion per Tastatur einzugeben. Dies hat zwar den Vorteil, dass exakte Werte verwendet werden können. Intuitiv ist aber davon auszugehen, dass die Motivation des Benutzers stark sinken wird, immer wieder neue Werte über die Tastatur eingeben zu müssen. Das Betätigen von Schaltern, bis ein gewünschter Wert erreicht wurde, ist ebenfalls umständlich und zeitaufwändig. In Online Shops ist dies eine geeignete Variante, um die gewünschte Stückzahl für ein käuflich erwerbbares Objekt anzugeben. Zum Einstellen der Werte in diesem Werkzeug ist es jedoch nicht geeignet. Eine zweckdienliche Variante ist ein Aufklappmenu. So kann der Benutzer ganz einfach einen Wert auswählen. Durch die Unübersichtlichkeit, bei zu vielen Werten oder der fehlenden Genauigkeit, bei zu wenigen Werten, entfällt jedoch diese Möglichkeit. Die allerdings für dieses Werkzeug passendste Variante ist die Verwendung von Schiebereglern, die Werte zwischen einem Minimal- und Maximalwert annehmen können. Dabei hat der Benutzer die Möglichkeit, durch einfaches Verschieben, einen Wert für die Gewichte festzulegen.

3.3.2 Visuelle Darstellungsmöglichkeiten

Im Fraunhofer IGD gibt es eine eigene Abteilung für den Bereich der „visual analytics“ [IVA12]. Dabei werden „Lösungen für die interaktive Visualisierung großer Datenmengen, so genannte Visual Analytics-Technologien“ [IVA12] erforscht und entwickelt. So gibt es verschiedenste Möglichkeiten dem Nutzer Informationen visuell darzustellen. Das menschliche Auge erfasst Informationen in einer sehr detaillierten und prägnanten Weise. Diese visuellen Informationen werden vom menschlichen Gehirn sehr schnell analysiert und interpretiert. Deswegen ist die visuelle Informationsaufnahme eine wichtige menschliche Eigenschaft. Schon früher wurden visuelle Darstellungen genutzt, um die Relevanz von Informationen deutlich zu machen und auszuwerten. So wurden beispielsweise Landkarten zur Vorbereitungen eines Angriffs auf einen Feind genutzt, um die eigenen und feindlichen Truppen in Bezug auf ihren Standort festzulegen und daraus eine Strategie zu entwickeln. Ein weiteres Beispiel im Bereich der visuellen Informationen ist die von John Snow erstellte Ghost Map [LEH10], siehe Abb. 3.1. Diese Karte zeigt exakt die Wohnorte der Cholera-Opfer von 1854. Durch die Analyse dieser Karte konnte festgestellt werden, dass die Wohnorte um eine Wasserpumpe verlaufen. Heutzutage werden ähnliche Karten in Rettungs- und Ordnungsdiensten genutzt. Dabei gehen diese Dienste mit der Zeit und nutzen digitale Karten, die noch exakter und Informationsreicher sind. Planung, Durchführung und auch die Auswertung sind durch den Fortschritt dieser Methode leichter und präziser geworden.

Abb. 3.1 - Ghost Map von John Snow - Wohnorte der Cholera Opfer von 1854

Visuelle Ausgabe der Empfehlungswerte

Eine leicht verständliche und trotzdem gut analysierbare Visualisierung zu finden ist keine leichte Aufgabe. In [LEH10] von Lehman et al. werden mehrere Möglichkeiten beschrieben, wie z.B. Tabellen, Streudiagramme oder das Zeichnen von Graphen. Da schon des Öfteren eine Nutzer-Seiten-Matrix angesprochen wurde, wird die visuelle Darstellung der berechneten Empfehlungswerte in einer Tabellenform dargestellt. Auch Lehman et al. sagen: „eine intuitive und wenig aufwändige Möglichkeit ist die textuelle Darstellung der Daten in Tabellen“ [LEH10]. Tabellen können Datensätze vollständig darstellen. Allerdings besitzen auch Tabellen gewisse Nachteile, die es zu beachten gibt. So nimmt die Übersichtlichkeit mit Zuname an Datensätzen ab. Außerdem nutzen Tabellen Normalerweise textuelle Beschreibungen für die Ausgabe von Ergebnissen. In einer Tabelle mit je 100 Nutzern und 100 Seiten, wobei jeder Nutzer und jede Seite zusammen einen Wert besitzen, können nur schwer, oder auch gar keine Muster erkannt werden. Diese Form der Ausgabe ist also nur teilweise von Vorteil. Allerdings werden in [LEH10] auch sogenannte graphisch komprimierte Tabellen erwähnt und beschrieben. Diese können enorme Datensätze darstellen, die anstatt textuellen Beschreibungen eine logisch nachzuvollziehende Visualisierung dieser qualitativen Datensätze nutzen. Auch das Ergebnis der Empfehlungswertfunktion, welches zwischen null und eins liegt, ist ein qualitativer Datensatz. So kann der prozentuale Wert genutzt werden, um zum Beispiel eine gewisse Farbe zu berechnen. Jeder Farbe wird ein mathematischer Wert zugeordnet, wie 0% = Rot, 50% = Gelb und 100% = Grün. Durch diese Zuordnung, kann das menschliche Auge vorhandene Muster leicht erkennen. Dies ist Vorteilhaft für die Auswertung und Überprüfung der verschiedenen Ergebnisse, welche durch das Setzen verschiedener Gewichte entstehen. Welche Farbe einen guten und welche einen schlechten Empfehlungswert besitzt soll ein Farbverlauf verdeutlichen, welcher von Rot (schlechter Empfehlungswert), über Gelb (mittelmäßiger Empfehlungswert) bis hin zu Grün (guter Empfehlungswert) verläuft. Indem der Benutzer des Werkzeuges, die Schieberegler verschiebt, ent-

stehen neue Gewichtungen für die Berechnung des Empfehlungswertes. Nachdem das Ergebnis der Empfehlungswertfunktion feststeht, soll dieses an eine neue Funktion weitergeleitet werden, welches eine Farbe berechnet, die zu dem qualitativen Datensatz passt. Um eine, für die Empfehlung geeignete Farbe zu berechnen, wird das RGB- System verwendet. Damit ein Empfehlungswert von einem Anderen unterschieden werden kann, dienen mehrere Farben. Das bereits genannte Beispiel eines Farbverlaufes und warum mehrere Farben einen besseren Unterschied deutlich machen, soll in Abb. 3.2 veranschaulicht werden.

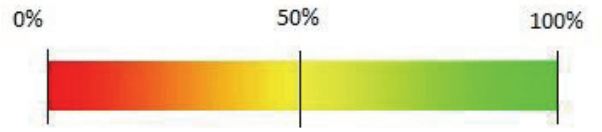

Abb. 3.2 - Farbverlauf für die Relevanz eines Empfehlungswertes

Im Folgenden wird erläutert, warum diese Farben ausgewählt wurden. Dabei geht der Verfasser auf die Bedeutung der Farben ein [IMM09].
Rot gilt allgemein als eine aggressive und bedrohliche Farbe und dient auch als eine Signalfarbe. So wird beispielsweise im Straßenverkehr die Farbe Rot in einer Ampel genutzt, um dem Fahrer deutlich zu machen „Stopp“, die Weiterfahrt ist im Moment nicht möglich und wird Ihnen verboten. In Verbindung dieses Beispiels mit dem errechneten Empfehlungswert, kann rot auch hier als Signalfarbe „Stopp“ dienen und dem Nutzer mitteilen, dass diese Seite nicht empfehlenswert ist. Die Farbe Gelb hingegen bedeutet unter anderem zweifelhaft und gilt als eine Warnfarbe. Auf das Ampelbeispiel bezogen, bedeutet dies „Achtung“, die Ampel wird gleich entweder auf grün oder rot umspringen. In Verbindung mit diesem Beispiel, lässt sich auch hier wieder ein Zusammenhang zu dem Empfehlungswert feststellen. Ist der berechnete Empfehlungswert beispielsweise bei 0,5, also 50%, so bedeutet dies „Achtung“, diese Seite ist vom durchschnittlichen Interesse. Somit kann die Seite zwar empfohlen werden, allerdings mit einer Warnung, dass es nur zum Teil den Wünschen entspricht. Die Farbe Grün hat nach [IMM09] auf Menschen eine beruhigende Wirkung und gilt unter anderem als zuverlässig. Als Ampelfarbe dient Grün als ein Signal zum Passieren der Kreuzung oder der Straße. In Verbindung der Farbe Grün mit einem berechneten Empfehlungswert, kann gesagt werden, dass diese Seite als zuverlässig angesehen wird.

3.3.3 Berechnung eines Farbwertes durch den Empfehlungswert

Damit einem Empfehlungswert eine Farbe zugeordnet werden kann, wurde im vorigen Kapitel beschrieben, welcher Bereich für welche Farbe gelten könnte. So wird der Farbwert für einen hohen Empfehlungswert in den grünen Bereich tendieren, während ein niedriger Wert in den roten Bereich tendieren wird. Wie bereits erwähnt, werden die Farben des RGB-Systems verwendet. Eine Farbkomponente kann dabei für jeden der drei grundlegenden

Farben einen Wert von 0 bis 255 annehmen. Es gibt einen zusammengesetzten Farbverlauf. Der erste Teil des gesamten Farbverlaufes verläuft von Rot zu Gelb und spiegelt die Empfehlungswerte von 0 bis 0,5 wieder. Die Empfehlungswerte ab 0,5 und darüber werden von Gelb bis Grün dargestellt. Diese zwei Farbverläufe setzen sich unterschiedlich zusammen. So wird bei dem Farbverlauf von Rot zu Gelb der Rot-Wert auf 255 und der Blau-Wert auf 0 gesetzt, während der Grün-Wert berechnet wird. Bei dem Verlauf von Gelb zu Grün wiederum wird der Grün-Wert auf 255 und der Blau-Wert auf 0 gesetzt, während der Rot-Wert berechnet wird. Bei Empfehlungswerten unter 0,5 entsteht zur Berechnung einer Farbe von Rot zu Gelb folgende Formel:

$$Grünwert = \left(\frac{Empfehlungswert}{0{,}5}\right) * 255$$

Der entstehende Grün-Wert wird genutzt, um eine Mischfarbe zu erstellen. Bei Empfehlungswerten ab 0,5 entsteht zur Berechnung einer Farbe von Gelb zu Grün diese Formel:

$$Rotwert = 255 - (255 * Empfehlungswert)$$

Die Subtraktion vom maximal anzunehmenden Farbwert ist nötig, da mit ansteigendem Empfehlungswerten der Rot-Wert immer kleiner werden muss.

3.3.4 Resultat

Es wurde aufgezeigt, wie ein Empfehlungswert berechnet und dieser an eine Farbberechnung weitergegeben wird. Ebenso wurde beschrieben, welche Darstellungsmöglichkeit genutzt werden soll. Das Ergebnis ist eine graphisch komprimierte Tabelle, wobei jedem Nutzer für eine Seite eine Farbe ausgegeben werden soll, die den Empfehlungsgrad darstellt. In Liferay gibt es verschiedene Kategorien, wie „Wiki“, „Blogs“ oder „Forums“. Damit der Nutzer den Überblick behalten kann, welcher Artikel zu welcher Kategorie gehört, soll die Ausgabe dabei nach Kategorien geordnet sein. Um die einzelnen Kategorien voneinander zu trennen dienen schwarze Trennlinien zur besseren Erkennbarkeit. Des Weiteren sollen für den Nutzer nur Empfehlungswerte für Seiten berechnet werden, die dieser selbst noch nicht gesehen hat. Daher werden bereits angesehene Artikel grau dargestellt. Diese Farbe ist gleichbedeutend mit der Farbe Silber. Im Allgemeinen steht Silber für „Gleichgültigkeit“ [IMM09]. Wird diese Bedeutung Assoziiert auf die Ausgabe einer bereits gesehenen Seite, so kann gesagt werden, dass diese Seite „bereits gesehen“ wurde und somit nicht mehr für den Nutzer zu berechnen ist.

3.4 *Zusammenfassung und Bewertung*

Das Konzept zeigt drei grundlegende Funktionen auf, die noch weiterentwickelt werden können. Im Rahmen dieser Bachelorarbeit sollen diese lediglich als Funktionen zum Testen des Werkzeuges dienen. Jede dieser Funktionen hat gewisse Vor- und Nachteile. Werden diese zusammen in einer einzelnen Funktion genutzt, kann einigen Nachteilen entgegengewirkt werden. Die Empfehlungen sind präziser, wenn alle Schieberegler beispielsweise hoch gesetzt sind. Die Aktualität empfiehlt die neuesten Seiten, sofern diese wenigstens einmal betrachtet wurden. Somit ist dies ein Ansatz zur Behebung des new Item Problems. Die Beliebtesten Seiten zeigen neuen Nutzern, ohne, dass diese sich etwas angesehen haben, welche Objekte sie interessieren könnten. Dies wiederum ist ein Ansatz zur Behebung des new User Problems. Die kombinierte Nutzung dieser Funktionen in einer einzigen Funktion zeigt auf, dass ein hybrider Ansatz eines Empfehlungssystems verwendet wird. Dabei wird die Weighted Methode der hybriden Empfehlungssysteme genutzt. Des Weiteren erhält jede Funktion ein Gewicht. Der Empfehlungswert, der das qualitative Ergebnis dieser kombiniert genutzten Funktionen ist, dient als Übergabewert für eine Farbberechnung. Diese soll für das qualitative Ergebnis eine geeignete Farbe berechnen. Da jede Farbe ihre eigene Bedeutung aufweist, soll Rot als „nicht zu empfehlen“, Gelb als „zweifelhaft“, Grün als „empfehlenswert“ und Grau als „bereits gesehen“ dienen. Mit Hilfe dieser Informationen kann ein Werkzeug entwickelt werden, das zur Untersuchung verschiedener Konfigurationsparameter dienen soll. Der Gebrauch der Schieberegler bietet den Vorteil einer leichten Handhabung. Als Visualisierung der Ergebnisse wird eine graphisch komprimierte Tabelle genutzt, die sowohl Nutzer, als auch bereits in Liferay betrachtete Seiten enthalten soll. Bei der Visualisierung der qualitativen Ergebnisse wird beachtet, zu welcher Kategorie die jeweiligen Seiten gehören und es erfolgt eine geordnete Ausgabe. Diese Ausgabe ist dabei logisch nachzuvollziehen. Daraus folgt, dass sie einfach zu verstehen ist und somit schnell vom menschlichen Auge analysiert und interpretiert werden kann. Mit Hilfe dieser Informationen ergibt sich ein System, dass in Abb. 3.3 veranschaulicht wird.

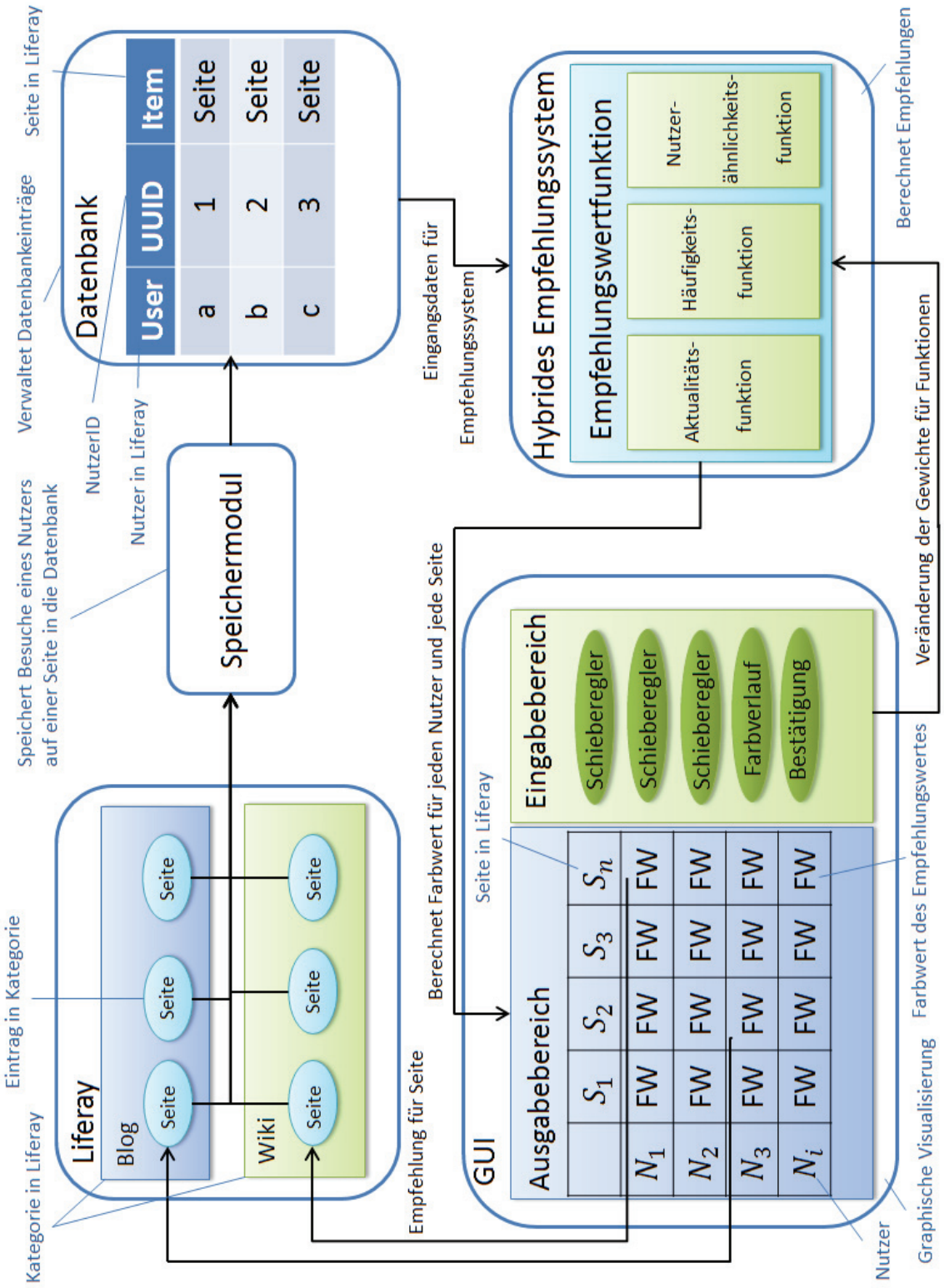

Abb. 3.3 – Systementwurf

Das in Abb. 3.3 veranschaulichte System ist dabei wie folgt zu verstehen. In Liferay befinden sich verschiedene Kategorien, wie Blog oder Wiki, welche JSP-Seiten beinhalten. Durch einen vom Verfasser entwickelten JSP-Code ist es möglich eine Betrachtung dieser Seite in die Datenbank zu schreiben und einem Nutzer zuzuordnen. Die in der Datenbank vorhande-

nen Einträge sollen dem hybriden Empfehlungssystem als Eingabedaten dienen. Das hybride Empfehlungssystem besitzt die bereits erläuterte Empfehlungswertfunktion, die wiederum die drei Funktionen der Aktualität, Häufigkeit und Nutzerübereinstimmung kombiniert. Das Ergebnis dieser Funktion wird in einen Farbwert des RGB-Systems umgerechnet und im Ausgabebereich der GUI an der zugehörigen Position in der Tabelle ausgegeben. Dabei ist eine Farbe die Empfehlung für eine Seite in der Jeweiligen Kategorie und für den jeweiligen Nutzer. Im Eingabebereich kann der Nutzer die Gewichte der einzelnen Funktion der Empfehlungswertfunktion verändern. Dies geschieht durch Schieberegler und einem Schalter zur Bestätigung. Anschließend werden die Farbwerte neu berechnet und ausgegeben.

4 Umsetzung

Zur umsetzung des Konzepts, wurde dieses prototypisch umgesetzt. Hierfür wurden Eclipse Helios [ECL12], die Liferay Community Edition [LR12] und Vaadin [VAA12] verwendet.

4.1 *Grundlagen*

Eclipse Helios

Um das konzipierte System prototypisch umzusetzen, benötigt es nicht nur eine Programmierumgebung, die es ermöglicht Programme in Java zu schreiben, sondern auch Java Portlets für Liferay zu entwickeln. Im Fraunhofer IGD wird dafür Eclipse verwendet. Eclipse ist ein plattformunabhängiges, quelloffenes Programmierwerkzeug. Es wird verwendet, um Software verschiedenster Programmiersprachen zu entwickeln. Ursprünglich wurde Eclipse als IDE für Java entwickelt. Allerdings weist es eine hohe Erweiterbarkeit auf, wodurch Eclipse auch zur Erstellung von Software dient, die in C, C++ oder PHP geschrieben werden sollen. Durch diese Erweiterbarkeit ist es möglich das kostenlose Liferay IDE Plug-In in Eclipse einzubinden. Die neueste Version ist laut [ITRE12] Eclipse Juno.

Liferay

Eine ebenso quelloffene Software ist das Liferay Portal. Es wird vor allem von Unternehmen genutzt, die ein Mitarbeiter – oder auch Geschäftsorientiertes Portal benötigen. Liferay „ist eine Enterprise-Web-Plattform für den Aufbau von Businesslösungen“ [LR]. Da Liferay in Java entwickelt wurde, ist es leicht in Eclipse einzubinden. Es dient dem Austausch von Informationen und Daten und bietet eine personalisierte Oberfläche. Dies ermöglicht dem Nutzer, eine persönliche Seite nach seinen Wünschen zu erstellen. Einige Hauptmerkmale von Liferay sind also die Benutzerpersonalisierung, das rollenbasierte Bereitstellen von Inhalten, die Erstellung von Benutzergruppen und eine leichte Handhabung.

Service Builder

Der Service Builder ist ein Hilfswerkzeug, das durch Liferay zur Verfügung gestellt wird. Dieser ist modellgetrieben und erzeugt Schnittstellen und Klassen für Datenbankeinträge, -abfragen und -aktualisierungen [LRD12]. Der Code kann genutzt werden, um eigene Methoden zu implementieren und Datenbankeinträge zu finden, zu erstellen, zu aktualisieren oder auch zu löschen. Nach der Installation der Liferay IDE in Eclipse Helios und dem Erstellen eines Liferay Projektes, kann der Service Builder genutzt werden.

Vaadin

Um die Visualisierung zu erstellen wird das Vaadin Plug-In für Eclipse verwendet. Dies ist ein Java basiertes Web Applikations Framework und dient der Erstellung interaktiver Applikationen. Diese Applikationen können ohne ein Plug-In im Browser genutzt werden. Vaadin weist eine Server-getriebene Architektur auf. Der Programmierer kann sich somit auf seinen Java-Code konzentrieren, während es die Architektur ermöglicht, HTML, XML und JavaScript aus dem Code zu erzeugen. Nutzt ein Entwickler Vaadin, so stehen ihm alle Java Bibliotheken zur Verfügung. Die meisten Vaadin-Nutzer sind Java – Entwickler für Businessorientierte Produkte. Auch Liferay verwendet Vaadin. So wird für die Erstellung von Portlets in Liferay Vaadin verwendet.

4.2 *Das Liferay Projekt*

4.2.1 Voraussetzungen schaffen

Das zu implementierende Programm nutzt sowohl die „Eclipse Helios Service Release 2“ Version, als auch das Liferay Community Edition Portal und Standard Developer Kit 6.1. Für die Implementierung ist es nötig, das Vaadin Plug-In, als auch die Liferay IDE in Eclipse herunterzuladen und zu installieren. Im Anschluss daran kann der Entwickler nun ein Liferay Projekt anlegen und das Liferay MVC Pattern verwenden.

4.2.2 Den Service Builder nutzen

Die im WEB-INF Ordner erstellete „service.xml“ bietet die Möglichkeit, eine Datenbanktabelle anzulegen und dessen Spalten zu benennen und deren Typ zu definieren. Es wurde festgelegt, dass für das Portlet die Entität „Recommendations“ und folgende in der Tabelle 4.1 aufgezeigte Spalten besitzen soll:

RecommendId	besitzt einen Primärschlüssel des Typs long und dient der Identifikation
content	ist ein String und speichert den Namen der Seite
groupId	ist ebenfalls ein String und speichert die Gruppe der aktuellen Seite
userId	Ist wieder vom Typ long und speichert die Nutzer ID des aktuellen Nutzers
portletId	speichert die ID des Portlets, als Typ String, unter der die aktuelle Seite liegt; String deshalb, weil einige ID’s nicht vom Typ long sind
link	speichert die Adresse als String, die zu der aktuellen Seite führt
modifiedDate	speichert das zuletzt modifizierte Datum der aktuellen Seite im Typ long

Tabelle 4.1 - Ausgewählte Spalten für die Entität Recommendations im Service Builder

Nachdem die gewünschten Entitäten und Spalten angelegt wurden ist noch das Betätigen des „Build Services“-Schalter nötig und der Service Builder generiert alle zugehörigen Packages, Klassen, Interfaces und Quellcode. Bei erfolgreichem Generieren erscheint, wie in der Abb. 4.1 „Build Successfull“ in der Konsole.

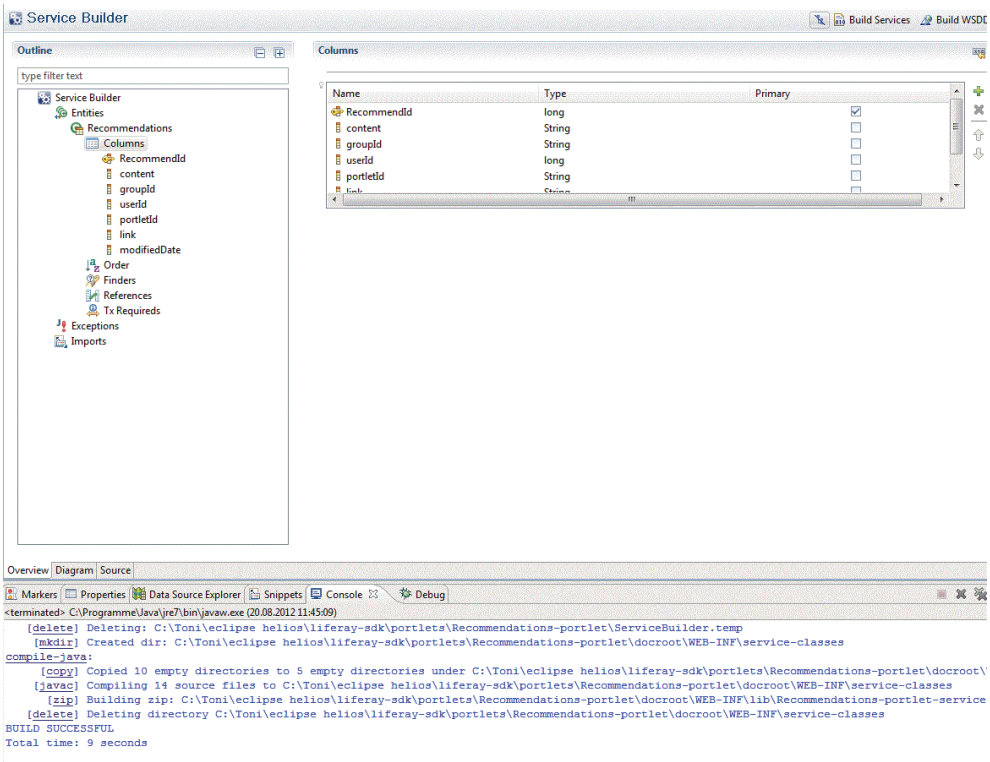

Abb. 4.1 - Graphische Oberfläche vom Service Builder

Damit es später keine groben Fehler gibt, müssen noch einige kleinere Konfigurationen vorgenommen werden. Durch die Feststellung, dass einige gespeicherte Daten zu lang für einen Eintrag sind, wie beispielsweise die Adresse einer aktuell betrachteten Seite, wurde nach einer Möglichkeit gesucht, diese Länge dementsprechend zu definieren. Nach langer Recherche wurde eine Lösung dafür gefunden. Die im docroot/WEB-INF/src/META-INF liegende Datei portlet-model-hints.xml kann verändert werden. Indem jedem nötigen „field“ noch ein konfiguriertes „hint“ hinzugefügt wird, kann die Länge verändert werden. Als Beispiel soll folgender Quellcode dienen:

```
<field name="link" type="String">
        <hint name="max-length">750</hint>
</field>
```

Die bereits angelegte Spalte „link“ vom Typ String wurde nun auf eine maximale Länge von 750 Zeichen gesetzt. Auch in der angelegten Java Klasse „RecommendationsLocalServiceImpl“ sind Veränderungen vorzunehmen. Diese erzeugte Klasse dient als Implementierungsschnittstelle der eigenen Methoden. Für die Implementierung werden zwei Methoden erstellt, die „getUsersRecommendation“ und die „addRecommendation“. Bei der ersten Methode wird der Funktion noch die „userId“ übergeben und über die vom Service Builder erzeugte Klasse „recommendationsPersistence“ kann mithilfe der Methode „findByUSER_ID(userId)“ eine Liste mit allen gespeicherten Einträgen der übergebenen Nutzer ID abgerufen werden. Der zweiten zu implementierenden Methode werden jedoch einige Werte mehr übergeben, nämlich die aus Tabelle 4.1 aufgezeigten Spalten. Über die bereits erwähnte Klasse „recommendationsPersistence“ gibt es die Methoden „create“ und „update“. Die Methode „create“ dient zum Erstellen des Datenbankeintrags, der alle übergebenen Werte hinzugefügt werden. Schlussendlich wird eine Datenbankaktualisierung mit der Methode „update“ durchgeführt, die den Eintrag in die Datenbank schreibt. Der zugehörige Quelltext der „RecommendationsLocalServiceImpl.java“ befindet sich im Anhang [Quellcode RLSI]. Damit jedoch auch alle getätigten Veränderungen der portlet-model-hints.xml und der RecommendationsLocalServiceImpl.java übernommen werden, ist es nötig den Service Builder erneut zu starten. Nach einem erneuten „Build Successfull“ sind alle Änderungen übernommen worden.

4.2.3 Anlegen und Exportieren des Portlets

Das Verfahren zum Anlegen eines Liferay Portlets, ist ähnlich dem Anlegen eines Liferay Projektes. Das Portlet soll dem Liferay Projekt zugeordnet werden, um es später nutzen zu können. Dafür wird ein Liferay Vaadin Portlet erstellt, das den Namen RecommendationsVaadinPortlet erhält und das entsprechende Java Package zugewiesen bekommt, indem auch das Projekt liegt. Bevor jedoch der Finish Button betätigt wird, sollte noch auf eine Einstellungsmöglichkeit hingewiesen werden. Nach mehrmaligem Betätigen des „Next“-Buttons gibt es die Möglichkeit unter „Liferay Display“ einen Namen einzugeben, unter dem das Portlet später in Liferay gefunden werden kann. Nach betätigen des „Finish“-Buttons wird eine Java Klasse erstellt. Nun kann der lokale Server gestartet werden. Wird Liferay nun in einem Browser aufgerufen (localhost:8080), erscheint das Liferay Portal. Nachdem eine Anmeldung stattgefunden hat, kann Liferay genutzt werden. Daraufhin kann mit dem Betätigen der rechten oberen Fläche „Sample Website Built with Liferay“ auf eine personalisierte Beispielseite gelangt werden. Dort können die entsprechenden Portlets per „Drag and Drop“ eingefügt, verschoben oder entfernt werden. Damit das Portlet unter Liferay auch wirklich genutzt werden kann, ist es nötig, es als Web application Archive auf den Server zu exportieren. Wenn ein erfolgreicher Export stattgefunden hat und das Portlet hinzugefügt wird, sollte „Hello Vaadin user!“ beim Aufruf des Portlets in Liferay erscheinen. Nach jeder Veränderung der

RecommendationsvaadinPortlet.java ist ein Export nötig, damit die Veränderungen übernommen werden können. Liferay bietet dafür die sehr gute Funktion, dass nach dem Speichern aller Veränderungen ein automatischer Export erfolgt.

4.3 *Entwicklung eines Empfehlungssystems*

Zur Entwicklung des konzipierten Empfehlungssystems soll es verschiedene Klassen geben, die ihre eigene Bedeutung besitzen. Dabei wurde das MVC Entwurfsmuster verwendet, um eine flexible und strukturierte Architektur zu erstellen.

4.3.1 Projektfunktionen

Die Klasse „RecommendationsMethods“ besitzt eine Vielzahl an Funktionen, die von anderen Klassen genutzt werden. Durch eine direkte Verbindung zu der bereits vorgestellten Implementierungsklasse ist es möglich Datenbankfunktionen aufzurufen. So können Einträge erstellt, aktualisiert oder gelöscht werden. Ebenso ist es möglich mit Hilfe dieser Klasse Nutzer- oder Eintragslisten aufzurufen, deren Größe zu bestimmen ist oder Inhalte abzurufen. Es besteht die Möglichkeit aus einer Eintragsliste eine einfache Liste, bestehend aus einzelnen Strings eines Inhalts zu erstellen. Des Weiteren kann der Verwender dieser Klasse erstellte Listen nach bestimmten Kriterien sortieren, als auch Duplikate entfernen. Durch die Sortierungsfunktionen ist es möglich festzustellen, wann eine neue Gruppe beginnt. Dies ermöglicht eine Unterscheidung der Gruppen, die später durch Trennlinien visuell voneinander getrennt werden. Es ist auch wichtig, dass nicht nur nach Gruppen sortiert wird, sondern im Anschluss eine weitere Sortierung nach dem aktuellsten Datum erfolgt. Weitere Funktionen, die diese Klasse bereitstellt sind das Abfragen des aktuellen Datums, um somit in einer weiteren Methode das niedrigste modifizierte Datum einer Seite zu bestimmen. Auch die Nutzerübereinstimmung und die Seite mit den meisten Betrachtungen können hier bestimmt werden. Diese Funktionen sind die Grundlage für die Gewichtsfunktion, welche sich ebenso in dieser Klasse befindet und später von einer anderen Klasse zur Farbberechnung verwendet wird.

4.3.2 Erste Visualisierungsschritte

Wie in der Konzeption bereits beschrieben soll am Ende eine graphisch komprimierte Tabelle entstehen, die durch Schieberegler verschiedene Farben für die Empfehlungen ausgeben soll. Am Anfang werden allerdings einige erste Versuche getätigt, um Datenbankeinträge vorzunehmen und sie auch wieder abzurufen und zu löschen. Dafür wird eine Oberfläche mit Vaadin erstellt, die aus einem vertikalen und einem horizontalen Layout, einem Textfeld und zwei Schaltern besteht. So soll in dem Textfeld etwas stehen, was in die Datenbank geschrieben werden soll. Durch die zwei Schalter „hinzufügen“, und „löschen“ wird es ermög-

licht die Datenbankeinträge vorzunehmen und auch wieder zu löschen. Durch einen sogenannten Button.ClickListener ist es möglich einem jeden Schalter beim Betätigen eine Funktion zuzuordnen. So erfolgt beispielsweise nach Betätigen des „hinzufügen“-Schalters ein Datenbankeintrag, der wiederum sofort unterhalb des Textfeldes ausgegeben wird. Durch den zweiten Schalter ist es möglich, die gesamte Datenbank zu löschen und somit auch die Ausgabe. In Liferay sieht die Ausgabe dann aus, wie in Abb. 4.2. Doch das sind nur die ersten Schritte. Der nächste Schritt ist die Erstellung einer Nutzerliste. Dabei wird für jeden Nutzer ein Schalter erstellt, auf dem der Name des Nutzers steht. Wird dann ein Name ausgewählt, so werden die Inhalte des Nutzers angezeigt. Dabei soll es möglich sein, Inhalte dem Nutzer hinzuzufügen, einzelne Inhalte zu editieren oder zu löschen und die gesamten Einträge eines Nutzers zu löschen. Dies wird jedoch im Kapitel „4.3.3 Die Visualisierungsklassen“ beschrieben.

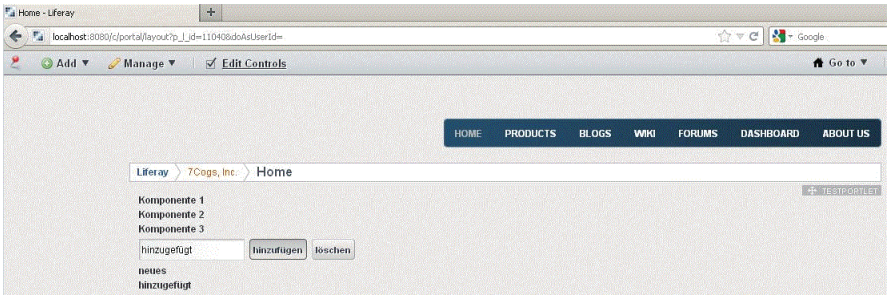

Abb. 4.2 - Datenbankeinträge hinzufügen und löschen

4.3.3 Der JSP – Code

Liferay verwendet für das Darstellen der Seiten, wie im Wiki oder Forums und Blogs Bereich, Java Server Pages, kurz JSP. JSP ist eine Web Programmiersprache und basiert auf Java und HTML. Somit sind eine dynamische Generierung von HTML und XML-Ausgaben und die Erstellung serverseitiger Web-Anwendungen möglich. Durch die Möglichkeit Java und JSP-Code in HTML und XML Seiten einzubringen, können Datenbankabfragen, als auch das Berechnen von Grafiken und die Verarbeitung von Nutzereingaben getätigt werden [JSP12].

Um einen eigenen Programm-Code nutzen zu können, muss beachtet werden, diesen in die vorgesehenen JSP Dateien einzufügen. Die Aufgabe ist es, das Nutzerverhalten zu beobachten. Somit sollte es einen Programm-Code geben, welcher es ermöglicht die Besuche einer Seite als Datenbankeintrag mit einer entsprechenden Kategorie, wie „Products“, „Blogs“, „Wiki“ und „Forums“ zu speichern. Diese Gruppen mit ihren Funktionen sind im Liferay Portal Ordner zu finden unter tomcat/webapps/ROOT/html/portlet. Der einzubettende Code unterscheidet sich für jede JSP-Datei nur geringfügig. Für jede dieser Dateien sind folgende Schritte gleich:

- Das Herausfiltern des aktuellen Nutzers und dessen Nutzer ID
- Das Erstellen einer UUID für den Datenbankeintrag
- Das Speichern des Pfades, unter dem diese Seite zu finden ist
- Die Gruppen ID abrufen
- Das zuletzt modifizierte Datum suchen und speichern
- Den Namen der aktuellen Seite herausfinden
- Der Aufruf der Methode aus dem Model-Bereich für den Eintrag mit allen eben genannten Werten

Der gesamte Code für die Speicher-Funktion der JSP-Seiten wird im Beispiel eines Blog-Beitrages im Anhang [Quellcode] dargestellt.

4.3.4 Die Visualisierungsklassen

Die Hauptklasse ist die am Anfang erstellte RecommendationsVaadinPortlet.java. Sie beschreibt sogenannte „Tab Sheets“, also Registerkarten. Somit ist es möglich zwischen mehreren, selbstdefinierten Ansichten zu wechseln. Beim Betätigen dieser Registerkarten wird das aktuelle Layout zurückgesetzt. Anschließend wird eine andere Visualisierungsklasse aufgerufen. Der Listener, welcher es ermöglicht beim Wechsel einer Registerkarte eine Methode aufzurufen ist der „selectedTabChangeListener“. Standardmäßig wird für die erste Registerkarte die bereits angesprochene Nutzerliste angezeigt. Sofern ein Name ausgewählt wurde, werden wie in Abb. 4.3 alle Einträge des Nutzers angezeigt, welche editier- oder auch löschbar sind. Des Weiteren besteht dort die Möglichkeit, selber Einträge vorzunehmen oder alle Inhalte des Nutzers zu löschen. Bei zu vielen Einträgen entstand das Problem, dass nicht alle Datensätze, bzw. Nutzer angezeigt werden konnten. Bei zu vielen Einträgen sollte normalerweise automatisch eine Bildlaufleiste entstehen. Da dies nicht funktioniert hat, musste eine andere Lösung gesucht werden. So wurde ein Panel erstellt und diesem alle Objekte hinzugefügt. Dieses Panel erstellt automatisch eine Bildlaufleiste, wenn die Visualisierung die Größe des Panels überschreitet. Um eine geordnete Ausgabe zu ermöglichen musste dabei das „AbsoluteLayout“ von Vaadin verwendet werden. Die entstandene Ansicht, Abb. 4.3 dient vor allem zu Testzwecken, wobei jeder Eintrag die Gruppe „Testeinträge“ und das aktuelle Datum zugewiesen bekommt.

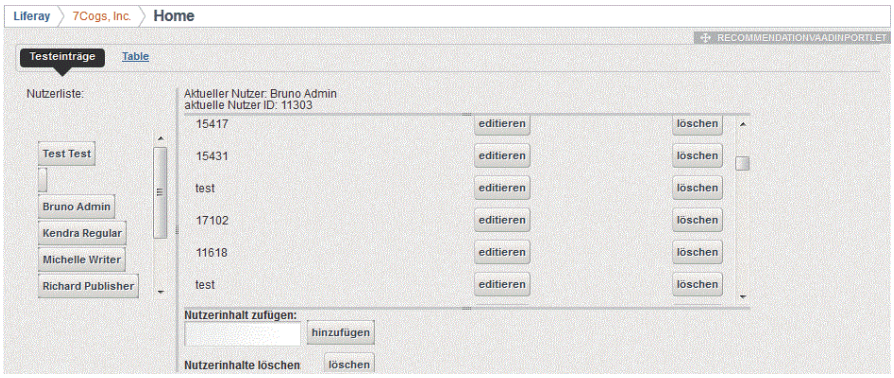

Abb. 4.3 - Nutzerliste und Ausgabe der Inhalte eines Nutzers

Bei der zweiten Registerkarte soll die graphisch komprimierte Tabelle aufgerufen werden. Dafür ist es nötig das übergebene „absoluteLayout“ mit zwei weiteren Layouts zu füllen. Das Erste dieser Beiden ist für das Zeichnen der Tabelle gedacht, das Zweite ist für das Werkzeug. Die Tabelle kann mittels der Klassen StreamResource, Embedded und InputStream visualisiert werden. Dafür muss die Tabelle mit InputStream gezeichnet werden. Wichtig für die Tabelle ist einerseits die bereits gezeigte Nutzerliste. Außerdem wird eine nach Gruppen und anschließend nach Aktualität sortierte Liste ohne Duplikate der Seiten benötigt. Da die sortierte Liste der Seiten zwar ausgegeben werden kann, aber diese als Verweise zu den Seiten dienen sollen, ist es nötig diese auf einem anderen Weg nach dem Zeichnen der Tabelle über das gezeichnete Bild zu legen. Da dafür die Positionen korrekt sein müssen, wird ein bestimmtes Muster zur Sortierung benötigt. Dieses Muster muss dann sowohl für die Verweise, als auch für das Zeichnen der farbigen Quadrate, also die qualitativen Empfehlungswerte, verwendet werden. Durch eine einheitliche Sortierung der Seiten beim Zeichnen der Tabelle und bei dem Erstellen der Verweise, konnte dies realisiert werden. Die zuvor genannten farbigen Quadrate dienen der Visualisierung. Somit ist es nicht vorteilhaft die Seitennamen oder Nutzernamen ausgeschrieben auszugeben, da dies unübersichtlich platzverschwendend wäre. Stattdessen werden Symbole verwendet. Somit ist eine einheitliche Ausgabe von Kästchen möglich. Ein Tooltip soll als Information dienen, um welchen Nutzer es sich handelt, bzw. um welche Seite einer Kategorie es sich handelt. Die Abb. 4.4 zeigt die entstandene graphisch komprimierte Tabelle mit Standardwerten für die Gewichtungen von 0,01. Somit wird verhindert, wie bereits im Konzept angesprochen, dass die Summe der Gewichte 0 sind.

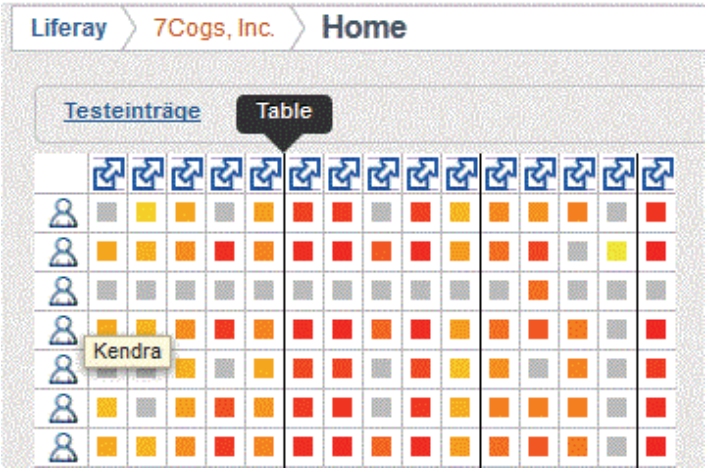

Abb. 4.4 - Graphisch komprimierte Nutzer-Seiten-Tabelle

4.3.5 Parameterkonfiguration

Die Standardwerte für die Gewichte von 0,01 sollen mit einer einfachen Möglichkeit verändert werden können. Im Konzept wurde bereits beschrieben, welches die beste Möglichkeit für dieses System darstellt. Somit wurden Schieberegler entwickelt, die im zweiten Layout, neben der Tabelle aufzufinden sind. Durch das Verschieben dieser Schieberegler können Werte von 0 bis 1 ausgewählt werden. Die entstehenden Gewichte sollen nach Betätigen des „OK“ Schalters an die Klasse zum Zeichnen der Tabelle übergeben werden. Diese wiederum soll für die Farbberechnung die Empfehlungsfunktion für jeden Eintrag verwenden. Durch das Setzen verschiedener Gewichte, wie in Abb. 4.5 und den Daten aus der Datenbank wird für jeden Nutzer und jede Seite ein unterschiedlicher Farbwert berechnet und zurückgegeben. Unter dem Schalter befindet sich eine Farblinie. Diese gibt an, welche Farbe welche Eigenschaft in Bezug auf eine Empfehlung besitzt.

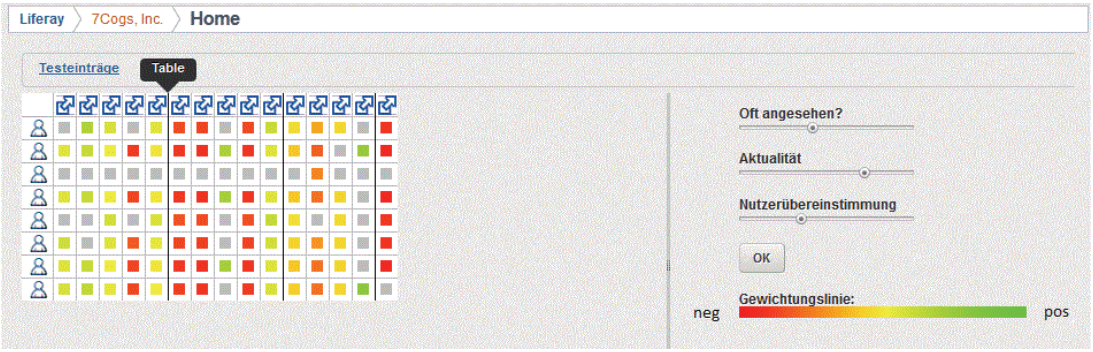

Abb. 4.5 - Graphisch komprimierte Tabelle mit vom Werkzeug verschieden gesetzten Gewichten

4.4 *Anwendungsbeispiele*

Die folgenden Abbildungen werden die drei Funktionen der Nutzerähnlichkeit, Aktualität und der Beliebtheit aufgezeigt. Zu erkennen sind verschiedene Ergebnisse, die durch das Verschieben der Schieberegler entstehen. In Abb. 4.6 wird das Ergebnis einer hohen Nutzerähnlichkeit dargestellt.

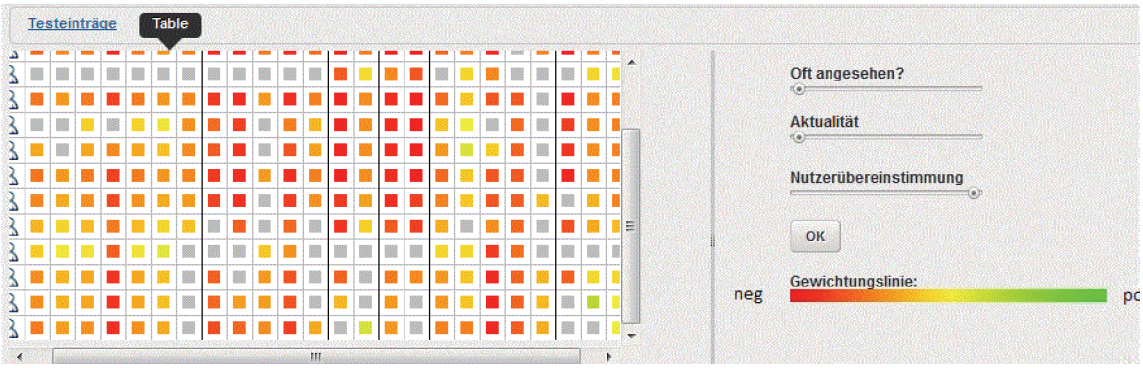

Abb. 4.6 - Ergebnis einer hohen Nutzerähnlichkeit

Im Vergleich dazu soll die Aktualität dienen, welche in Abb. 4.7 dargestellt ist. Dabei ist gut zu erkennen, dass die aktuellsten Einträge links einer Kategorie dargestellt sind

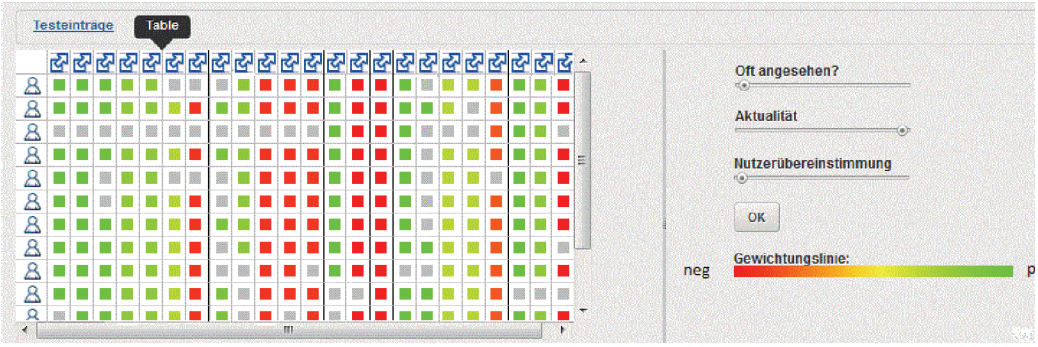

Abb. 4.7 - Ergebnis einer hohen Aktualität

In Abb. 4.8 ist die Beliebtheit einer Seite zu erkennen.

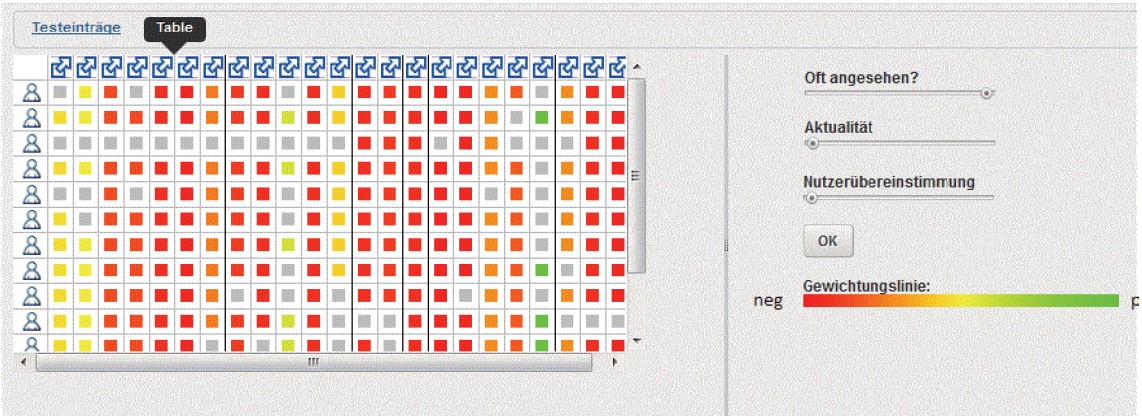

Abb. 4.8 - Ergebnis für hohe Beliebtheit

Durch eine Kombination aus allen Funktionen, lassen sich verschiedene Ergebnisse verzeichnen. In Abb. 4.9 sind die Gewichte für alle drei Funktionen hoch gesetzt, besonders aber für die Beliebtheit einer Seite.

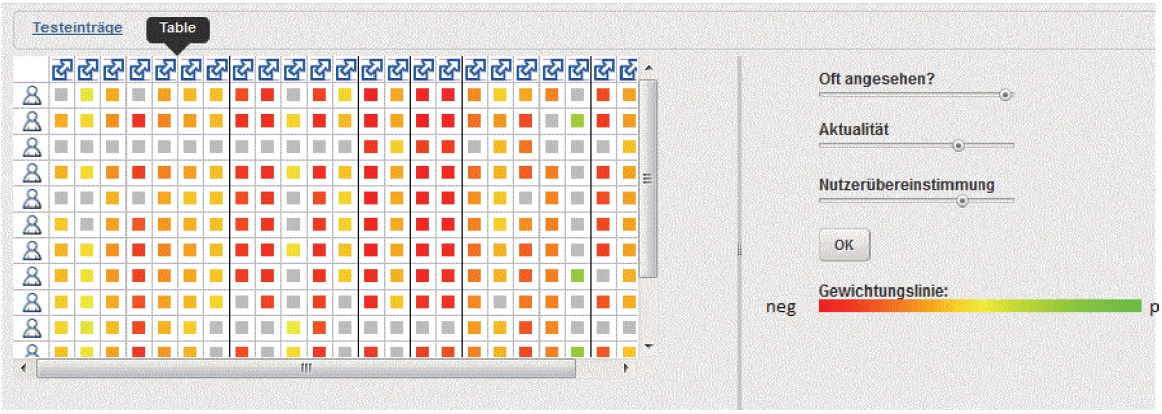

Abb. 4.9 - Ergebnis aus einer Mischung der Schieberegler – 1

Im Vergleich dazu, kann in Abb. 4.5 erkannt werden, welche Empfehlungen berechnet werden, wenn die Gewichte unterschiedlich gesetzt sind.

5 Fazit und Ausblick

5.1 Fazit

Das Ziel dieser Arbeit war es festzustellen, ob ein Empfehlungssystem ohne Nutzung expliziter Bewertungen der Inhalte und ohne explizite Informationen über die Nutzer entwickelt werden kann. Dafür wurden im aktuellen Forschungsstand drei Arten von Empfehlungssystemen und die Methodiken zum Sammeln von Daten beschrieben. In der Konzeption wurde das hybride Empfehlungssystem ausgewählt, das die Weighted Methode nutzt, um Empfehlungen zu berechnen. Für dieses Empfehlungssystem wurden drei Funktionen entwickelt, die in einer neuen Funktion kombiniert werden. Um die Ergebnisse zu überprüfen, wurde ein Portlet für Liferay entwickelt, das ein Werkzeug bereitstellt. Dieses, mit Vaadin erstellte, Werkzeug besitzt eine graphische Ausgabe der qualitativ entstehenden Daten und dient der Überprüfung der Ergebnisse. Das entstandene hybride Empfehlungssystem soll Handwerkern und Experten des KFZ-Handwerks, des MEMO-Projektes, Empfehlungen für Seiten, innerhalb des genutzten Liferay Portals berechnen. Durch die Nutzung dieses Portlets, soll ihre Suchzeit nach nützlichen Informationen deutlich verkürzt werden. Auch der Aspekt, dass explizit keine Bewertungen abgegeben werden müssen, soll den Nutzern den Vorteil verschaffen, ihre Zeit sinnvoller zu nutzen. Die Eingangsdaten für das Empfehlungssystem werden durch das Beobachten des Nutzerverhaltens gewonnen. Dafür wird die Anzahl der Betrachtungen einer Seite genutzt.
Die Ausgabe der qualitativen Daten ist eine graphisch komprimierte Tabelle, wobei für jeden Nutzer-Seiten-Eintrag ein farbiges Quadrat ausgegeben wird. Dies soll die Wichtigkeit der Empfehlung deutlich machen. Dabei sind die Schieberegler eine sehr gute Lösung, um verschiedene Prioritäten, in einfachster und schnellster Weise, zu setzen. Letztendlich ist zu sagen, dass es möglich ist ein Empfehlungssystem zu entwickeln, das nur implizit Daten sammelt.

5.2 Ausblick

Durch die Erstellung des hybriden Empfehlungssystems und dem zugehörigen Werkzeug zur Kontrolle ist es möglich auf Grund weniger Daten trotzdem einen Empfehlungswert zu berechnen. Damit dieser hybride Ansatz verbessert und die Genauigkeit gesteigert werden kann, ist es möglich die inhaltsbasierte Funktion zu verbessern und zu erweitern. Die momentane inhaltliche Funktion sammelt inhaltliche Informationen nur auf geringer Ebene, nämlich das Modifizierungsdatum einer betrachteten Seite in Liferay. Eine Verbesserung dieser Funktion ist eine denkbare Aufgabe für die Zukunft, um dem New User Problem entgegenzuwirken. Eine geeignete Testumgebung für dieses Programm zu schaffen, ist ein weiterer Punkt, der in der Zukunft wichtig wäre. Diese Testumgebung soll es ermöglichen, die

Performance unter realen Bedingungen zu testen. Es wäre wissenswert, wie lange es dauert, bei einer enormen Menge an Seiten und Nutzern, Empfehlungen zu berechnen und die Veränderungen in den Ergebnissen zu beobachten. Ein weiterer Aspekt, der interessant wäre, ist die Nutzung der Liferay Enterprise Edition und der dort zur Verfügung gestellten Möglichkeiten des Monitorings. So können in das bestehende Empfehlungssystem neue Methoden integriert und neue Ergebnisse beobachtet werden.

6 Literaturverzeichnis

[AB10] Aksel, F.; Birtürk, A.: Enhancing Accuracy of Hybrid Recommender Systems through Adapting the Domain Trends. In: Workshop on the Practical Use of Recommender Systems, Algorithms and Technologies(PRSAT 2010), September 30, 2010

[AH03] Abdelnur, A.; Hepper, S.: Java Portlet Specification. Version 1.0, 2003

[AT05] Adomavicius, G.; Tuzhilin, A.: Towards the Next Generation of the State-of-the-Art and Possible Extensions. In: IEEE Transactions on knowledge an Data Engineering, Vol 17, No.6, June 2005

[BHK98] Breese, J. S.; Heckerman, D. & Kadie, C.: Empirical Analysis of Predictive Algorithms for Collaborative Filtering. 1998

[BS97] Balabanovic, M.; Shoham, Y.: Fab: Content-based, collaborative recommendation. In: Communications of the ACM, March, 1997

[BUR02] Burke, R.: Hybrid Recommender Systems: Survey and Experiments. 2002

[CHIP12] http://www.chip.de [Online]: Datum des letzten Zugriffs: 20.08.2012

[COS03] Cosley, D.; Lam, S. K.; Albert, I.; Konstan, J. A.; Riedl, J.: Is seeing believing?: how recommender system interfaces affect users' opinions. In: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 585-592, 2003

[EBAY12] http://www.ebay.de [Online]. Datum des letzten Zugriffs: 20.08.2012

[ECL12] http://www.eclipse.org/downloads/ [Online]. Datum des letzten Zugriffs: 20.08.2012

[EF12] http://pages.ebay.de/services/forum/feedback.html [Online]. Datum des letzten Zugriffs: 06.08.2012

[FUN04] Funk, M.: Recommender Systems in Theorie und Praxis, 2004

[GHJ05] Zhu, T.; Greiner, R.; Häubl, G.; Price, B.; Jewell, K.: Behavior-based Recommender Systems for Web Content. In: Workshop: Beyond Personalization, January 9, 2005

[GM09] Gunawardana, A.; Meek, C.: A unified approach to building hybrid recommender systems. In: Proceedings of the third ACM conference on Recommender systems, pp. 117-124, 2009

[GON10] Gong, S.: A Collaborative Filtering Recommendation Algorithm Based on User Clustering and Item Clustering. In: Journal of Software, Vol.5, No. 7, July, 2010

[HB10] Holub, M.; Bieliková, M.: Behavior Based Adaptive Navigation Support. In: Workshop on the Practical Use of Recommender Systems, Algorithms and Technologies(PRSAT 2010), September 30, 2010

[HEU10] Heunemann, M.: Empfehlungssysteme – Collaborative Filtering. In: Seminararbeit, Bauhaus-Universität Weimar, 28.06.2010

[HKV08] Hu, Y.; Koren, Y.; Volinsky, C.: Collaborative Filtering for Implicit Feedback Datasets. In: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, December 15-19, 2008

[HOF03] Hofmann, T.: Collaborative Filtering via Gaussian Probabilistic Latent Semantic Analysis. In: SIGIR, August 28, 2003

[IGD11] Fraunhofer-Institut für Graphische Datenverarbeitung, Protokoll des MEMO-Projekttreffens vom 7. Juli 2011. Internes Projektdokument

[IMM09] Immoos, F.: Farben – Wahrnehmung – Assoziation – Psychoenergetik

[ITRE12] http://www.itrig.de/index.php?/archives/1228-Eclipse-Juno-Neue-Version-4.2-der-Entwicklerplattform-ist-nun-Standard.html [Online]. Datum des letzten Zugriffs: 15.08.2012

[IVA12] http://www.igd.fraunhofer.de/Institut/Abteilungen/IVA [Online]. Datum des letzten Zugriffs: 10.08.2012

[JK02] Järvelin, K.; Kekäläinen, J.: Cumulated gain-based evaluation of ir techniques. In: *ACM Trans. Inf. Syst.*, October, 2002

[JSK10] Jawaheer, G.; Szomszor, M.; Kostkova, Patty: Comparison of implicit and explicit feedback from an online music recommendation service. In: Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems, September 26, 2010

[JSP12] www.jsptutorial.org/content/introduction [Online]. Datum des letzten Zugriffs: 15.08.2012

[KIR04] Kirchhofer, A.; Gurzki, T.; Hinderer, H.; Vlachakis, J.: Was ist ein Portal - Definition und Einsatz von Unternehmensportalen. In: Fraunhofer IAO Whitepaper, 2004

[KLSS09] Klein, T. J.; Lambertz, C.; Spagnolo, G.; Stahl, K. O.: The actual structure of eBay's feedback mechanism and early evidence on the effects of recent changes. In: International Journal of Electronic Business, 2009

[KOR01] Kim, J.; Douglas, W. O., Romanik, K.: User Modeling for Information Filtering Based on Implicit Feedback. In: ISKO-France, 2001

[LEH10] Lehmann, D. J.; Albuquerque, G.; Eisenmann, M.; Tatu, A.; Keim, D. A.; Schuhmann, H.; Magnor, M. A.; Theisel, H.: Visualisierung und Analyse multidimensionaler Datensätze. In: Informatik Spektrum, pp. 589-600, 2010

[LGS11] Lops, P.; de Gemmis, M.; Semeraro, G.: Content-based Recommender Systems: State of the Art and Trends. In: Recommender Systems Handbook, LLC 2001

[LJB98] Linden, G.D.; Jacobi, J.A; Benson, E.A.: Collaborative recommendations using item-to-item similarity mappings. September 18, 1998

[LP08] Lee, T. Q.; Park, Y.; Park, Y. T.: A Time-based Recommender System using Implicit Feedback. In: Expert Systems with Applications: An International Journal, May, 2008

[LR12] www.liferay.com [Online]. Datum des letzten Zugriffs: 15.08.2012

[LRD12] http://www.liferay.com/documentation/liferay-portal/6.1/development/-/ai/service-build-5 [Online]. Datum des letzten Zugriffs: 24.08.2012

[LSY03] Linden, G.; Smith, B.; York, J.: Amazon.com Recommendations-Item-to-Item Collaborative Filtering; In: IEEE Computer Society, 2003

[MEM12] http://www.memo-apps.de/projekt.html [Online]. Datum des letzten Zugriffs: 20.08.2012

[MMN02] Melville, P.; Mooney, R. J.; Nagarajan, R.: Content-Boosted Collaborative Filtering for Improved Recommendations. In: Proceedings of the Eighteen National Conference on Artificial Intelligence(AAAI-2002), pp. 187-192, July, 2002

[MRS08] Manning, C. D.; Raghavan, P.; Schütze, H.: *Introduction to Information Retrieval. In:* Cambridge University Press, 2008

[MS00] Materen, R. v.; Someren, M. v.: Using Content-Based Filtering for Recommendation, 2000

[OK98] Oard, D.; Kim, J.: Implicit Feedback for Recommender Systems. In: Proceedings of the AAAI Workshop on Recommender Systems, August, 1998

[PA11] Parra, D.; Amatriain X.: Walk the talk: analyzing the relation between implicit and explicit feedback for preference elicitation. In: Proceedings of the 19th international conference on User modeling, adaption, and personalization, 2011

[PB07] Pazzani, M.J.; Billsus, D.: Content-Based Recommender Systems. In: The Adaptive Web, pp. 325 – 341, 2007

[PKYA11] Parra, D.; Karatzoglou, A.; Yavuz, I.; Amatriain, X.: Implicit Feedback Recommendation via Implicit-to-Explicit Ordinal Logistic Regression Mapping. In: CARS 2011, October 23, 2011

[RLU12] Rajaraman, A.; Leskovec, J.; Ullman, J. D.: Recommendation Systems. In: Mining of Massive Datasets, Chapter 9, 2012

[RV97] Resnick, P.; Varian, H. R.: Recommender Systems. In: Communications of the ACM, March 1997 Vol. 40 Nr. 3

[SAR01] Sarwar, B. M.; Karypis, G.; Konstan, J. A. & Reidl, J.: Item-based collaborative filtering recommendation algorithms. In World Wide Web, 2001, pp. 285-295

[SCH07] Schafer, J. B.; Frankowski, D.; Herlocker, J.; Sen, S.: Collaborative Filtering Recommender Systems. In: The Adaptive Web, pp. 291-324, 2007

[SKK01] Schwab, I.; Kobsa, A.; Koychv, I.: Learning User Interests through Positive Examples Using Content Analysis and Collaborative Filtering. In: Internal Memo, GMD, 2001

[SKL09] Spiegel, S.; Kunegis, J.; Li, Fang: Hydra: a hybrid recommender system [cross-linked rating and content information]. In: Proceedings of the 1st ACM international workshop on Complex networks meet information knowledge management, November 6, 2009

[SKR99] Schafer, J. B.; Konstan, J.; Riedi, J.: Recommender systems in e-commerce. In: Proceedings of the 1st ACM conference on Electronic commerce, pp. 158-166, 1999

[TA99] Tuzhilin, A.; Adomavicius, G.: Integrating user behavior and collaborative methods in recommender systems. In: CHI'99 Workshop. Interacting with Recommender Systems, 1999

[VAA12] https://vaadin.com/home [Online]. Datum des letzten Zugriffs: 20.08.2012

[ZDZ07] Zhou, L.; Dai, L.; Zhang, D.: Online Shopping Acceptance Model – A critical Survey of Consumer Factors in Online Shopping

7 Anhang

[Quellcode RLSI] - RecommendationLocalServiceImplement

```
import de.fraunhofer.igd.ide.memo.toni.Recommendations.model.Recommendations;
import de.fraunhofer.igd.ide.memo.toni.Recommendations.service.base.RecommendationsLocalServiceBaseImpl;

/**
public class RecommendationsLocalServiceImpl
    extends RecommendationsLocalServiceBaseImpl {
    /*
    ----------------------------------------------------------------------------------------------------

    public Recommendations addRecommendationById(Recommendations recommendation, long userId, long random)
    throws SystemException, PortalException {
        ------------------------------------------------------------------------------------------------
        Recommendations rec = recommendationsPersistence.create(random);                    //Komponente in DB erzeugen
        ------------------------------------------------------------------------------------------------
        rec.setContent(recommendation.getContent());                                        //Komponente Inhalt setzen
        rec.setUserId(userId);                                                              //UserID setzen
        ------------------------------------------------------------------------------------------------
        return recommendationsPersistence.update(rec, true);                                //Datenbank aktualisieren mit Komponente
        ------------------------------------------------------------------------------------------------
    }

    public Recommendations addRecommendation(long randomRecId, String content, long userId,
        ------------------------------------------------------------------------------------------------
        String groupId, String portletId, String link, long modifiedDate)
            throws SystemException, PortalException {
        ------------------------------------------------------------------------------------------------
        Recommendations rec = recommendationsPersistence.create(randomRecId);               //Komponente in DB erzeugen
        ------------------------------------------------------------------------------------------------
        rec.setRecommendId(randomRecId);                                                    //Komponente Inhalt setzen
        rec.setContent(content);
        rec.setUserId(userId);
        rec.setPortletId(portletId);
        rec.setGroupId(groupId);
        rec.setLink(link);
        rec.setModifiedDate(modifiedDate);
        ------------------------------------------------------------------------------------------------
        return recommendationsPersistence.update(rec, true);
        ------------------------------------------------------------------------------------------------
    }

    public List<Recommendations> getUsersRecommendation(long userId) throws SystemException {
        ------------------------------------------------------------------------------------------------
        try {
            recommendationsPersistence.clearCache();
        }
        catch (Exception e) {
            System.out.println("Cache?!");
        }
        List<Recommendations> recommendations = recommendationsPersistence.findByUSER_ID(userId);
        ------------------------------------------------------------------------------------------------
        return recommendations;
        ------------------------------------------------------------------------------------------------
    }
}
```

Abb. 7.1 - RecommendationsLocalServiceImpl.java

[Quellcode JSP] - Speichern eines Besuchs einer „Blog“-Seite

```
<%
        User user1 = PortalUtil.getUser(request);
        RecDbMethods model = new RecDbMethods();

                                        //aktuelle ThreadId auslesen
        long ThreadId = entry.getEntryId();
        String pageTitle = Long.toString(ThreadId);
                                        //ThreadId in String umwandeln

        Random random = new Random();
        long recId = random.nextLong();
        long UserId = user1.getUserId();

        String link = PortalUtil.getHost(request) + ":" + PortalUtil.getPortalPort() + PortalUtil.getCurrentURL(request);

/*-------------------------------Wie aktuell ist der Eintrag: ----------------------------------*/

        long modifiedDateLong = entry.getModifiedDate().getTime();

                                        Ende Aktualität
/*-------------------------------------------------------------------------------------*/
        String portletId = PortalUtil.getPortletId(request);

        Locale local = new Locale("de");
        String groupId = LanguageUtil.get(local, "javax.portlet.title." + portletId);

        model.addRecommendationWithIdAll(recId, pageTitle, UserId, groupId, portletId, link, modifiedDateLong);

%>
```

Abb. 7.2 - Speichern eines Besuchs einer Blog-Seite

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Bachelorarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Neubrandenburg, den 04.09.2012

Unterschrift