Halbautomatische Verlinkung in großen Informationssystemen

Sebastian Wilhelm

MASTERARBEIT

eingereicht am
Masterstudiengang

Softwareentwicklung und Wirtschaft

an der Technischen Universität Graz

im April 2011

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am ……. ……..
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

……. ……..
date (signature)

Inhaltsverzeichnis

Kurzfassung

Web-Enzyklopädien erfreuen sich in der heutigen Zeit großer Beliebtheit. Der Dokumentenbestand sowie seine Vernetzung sind wichtige Faktoren in diesen Online-Systemen. Zumeist handelt es sich bei solchen Web-Bibliotheken nicht um statische Entitäten. Diese Art von Informationssystem ist eigentlich einer ständigen Erweiterung an Wissen unterworfen, und das sollte logischerweise eine kontinuierliche Anpassung des Verlinkungsnetzwerks zur Folge haben.

Das begründet einen hohen Wartungsaufwand, was dazu führt, dass manuelle Vernetzung bei einem immer größer werdenden Dokumentenbestand nicht mehr zu gewährleisten ist.

Eine Lösung wäre womöglich die Editorenanzahl zu erhöhen um dem Aufwand entgegenzuwirken. Da stellt sich aber die Frage der qualitativen Konsistenz der Verlinkungen. Studien haben gezeigt, dass bei derartigen kollaborativen Systemen je nach unterschiedlichem Informationsstand die Verlinkungsstrategien der Editoren voneinander abweichen[41].

Als Alternative zum Browsen über Verlinkungen kann die Suche angesehen werden. Oft enthalten derartige Enzyklopädien eine ausgeklügelte Suchfunktion. Navigation über den vernetzten Hypertext aber spielt im Vergleich zur Suche oft eine größere Rolle, denn sie gibt dem User einen besseren Einblick in den Kontext und schafft Beziehungen zwischen den Dokumenten[117].

Unverknüpfte oder wenig verknüpfte Dokumente können deshalb als großes Problem angesehen werden. Sie geben dem Benutzer keine weiteren Möglichkeiten als bestenfalls die Suche zu bemühen oder die Seite eben wieder zu verlassen, wobei diese Option prozentuell viele Benutzer im Fall von Wikipedia und dem Austria-Forum wählen[54][63].

Das Ziel meines Diplomprojektes ist es den Vorgang der Verlinkung zu erleichtern und dadurch den Wartungsaufwand zu verringern. Der manuelle Vorgang nämlich, den ein Autor durchführen muss, um sein Dokument zu vernetzen, verspricht wenig Anreiz und ist besonders in Dokumenten mit entsprechend langem Inhalt mühsam. Zur Lösung dieser Probleme liegt die Idee einer Automatisierung der Verlinkung nah.

Diese Diplomarbeit handelt von einem von mir programmierten Verlinkungs-Tool. Es wird ein semi-automatischer Ansatz benützt um die Dokumente zu

verlinken. Semi-automatisch deshalb, da es als Unterstützungstool fungiert und in diesem Sinne dem interessierten Autor Linkvorschläge bietet. Dieser vermag die Vorschläge zu evaluieren und zu speichern. Das erspart ihm das Suchen nach passenden Zusammenhängen sowie das manuelle Linksetzen im Text.

Mein Tool soll den Anreiz bieten die Dokumente sinnvoll und einfach zu verlinken, um damit den Editier-Aufwand zu verringern und die Informationen besser zu vernetzen. Im Großen und Ganzen aber soll es dabei helfen, die Dokumente informativer zu gestalten, um dem Benutzer mehr Möglichkeiten zu bieten sein Wissen zu erweitern.

Abstract

Web-Encyclopedias enjoy a great poularity at the moment. The set of documents plus their interlinkings play an important role in such kind of systems. Mostly these online-librarys aren't static entities. This kind of information-system is dealing with an immanent expansion of knowledge and for this reason the hypertext-network should also be adjusted to these changes.

This causes a huge effort in respect of maintenance. In particular the manual process of trying to keep the linkage in a fast growing set of document up to date seems to be quite impossible.

A solution to a better distribution of the maintenance effort might be the increase of the editor-count.This might lead to a problem. Studies have shown, that the linking at this kind of collaborative systems depends on the information space of the authors. So the links might differ strongly depending on who is linking[41].

A search engine might be an alternative towards browsing through links. Such information systems often have sophisticated search functions included. However, navigation through linked hypertext is often playing a more important role compared to the searching-function, because it gives the user more insight into the context and creates relationships between the documents[117].

Therefore less linked documents might be thought as a great problem in such information systems. It leaves the user to the only possibility to exit the page or in best case using the search-function. No surprise, thinking of the aforenamed problems that the percentage of users, which continue surfing on web-encyclopedias like Wikipedia or Austria-Forum, after entering the site, is really low[54][63].

One important goal of my diploma-project is to relieve the manual process of interlinking and as a matter of fact decrease the maintenance effort. The manual linking procedure, which has to be applied by the author, won't increase the attraction of it, and can be quite tedious by editing huge documents. A solution of this problems might be a tool which could help in the authoring process.

This diploma thesis is about such a program. A semi-automatic approach for linking documents. Semi-automatic to the effect that it serves as a support-tool for the editor, and gives the interested author link-suggestions.

This approach offers the possibility of evaluation by the editor, who doesn't need to search for related documents and to create the linking manually inside the text.

The program should help to link the articles in a meaningful and easy way, in order to decrease the authoring-effort and to make information more valuable. All things considered should this tool help to improve the article's information content, to give the user more opportunities for increasing his knowledge.

Kapitel 1

Einleitung

1.1 Motivation, Zielsetzung und Beitrag der Arbeit

Vannevar Bush[1], der Urvater des Personal Computers, bezeichnete den Hypertext als *computer glue*[2] zwischen den Dokumenten, der die Informationen, welche darin enthalten sind, zusammenbindet [55]. "Computer Klebstoff" im Kontext des Webs hat sich in der Möglichkeit der Verlinkung der Webdokumente etabliert.

Online-Enzyklopädien dienen heute als wichtige Quellen zu Informationen. Das bekannteste Beispiel ist Wikipedia[3], welches über drei Millionen Einträge, bzw. Dokumente enthält und an die 5000 uneigennützige "Reviewer" beschäftigt, die die Dokumente nach den von der Enzyklopädie vorgelegten Kriterien begutachten[114].

Es existieren in diesem System nicht nur menschliche Prüfer, auch computerisierte Helfer, sogenannte Bots durchlaufen die Seiten. Einer davon sucht seit Februar 2009 nach unverlinkten Informationen in dem Bestand[112]. Die Kriterien sind die Quantität der Verlinkungen auf das dementsprechende Dokument. Es wird also, wenn keine internen Links auf ein solches zeigen, von einem verwaisten Dokument gesprochen.

Im englischen Wikipedia existieren ungefähr 19% Artikel dieser Art (ca. 180.000 Dokumente)[113]. Im deutschen Wikipedia findet sich über die Anzahl keine wirkliche Angabe[115].

Um diesem suboptimalen Zustand entgegenzuwirken wurde in Wikipedia das Projekt ORPHAN gegründet, welches den Auftrag hat, verwaiste Artikel in das Informationsystem wieder einzugliedern, wobei sich bei "ORPHAN" 86 aktive Mitglieder beteiligen[116]. Die Effektivität dieser Gegenmaßnahme ist in Zweifel zu ziehen, betrachtet man das Wachstum an verwaisten Dokumenten in Abbildung 1.1.

[1] Mehr Informationen zur Person siehe 2.2.1

[2] Frei übersetzt.: Computer-Klebstoff

[3] Siehe mehr dazu in Abschnitt 2.6.3

Bezogen auf den User selbst ergeben sich weitere Fragen. Zum Beispiel auf sein Verhalten in Relation zur Verlinkung eines Web-Dokuments.

Eine Studie von David Gleich et al.[54] hat bezüglich dem User-Verhalten in der Web-Enzyklopädie Wikipedia[4] ergeben, dass zwischen 32.5 und 42.5% der User, die die Seite betreten, anhand den Kanten bzw. Links andere Dokumente im System aufsuchen. Der Rest verlässt die Seite wieder.

Bei der österreichischen Web-Enzyklopädie Austria-Forum[5] verhält es sich ähnlich[63].

Zum Vergleich: Bei der Film-Plattform HelloMovies[6] ist die Durchschnittswahrscheinlichkeit, dass das Surfen auf der Seite fortgesetzt wird zwischen 60 und 72.5%.

Die Gründe für die hohe Wahrscheinlichkeit des Verlassens der Seiten von Wikipedia oder des Austria-Forums durch die Benützer sind noch nicht erforscht. Möglicherweise bietet mein Projekt aber einen Lösungsansatz zu diesem Problem.

Diese Diplomarbeit nämlich stellt ein Verlinkungs-Tool vor, dass den Zweck hat, automatisch generierte Linkvorschläge dem Benützer zu präsentieren und diese, falls sie akzeptiert werden, in das Dokument zu integrieren. Das Tool wurde für die Web-Enzyklopädie Austria-Forum implementiert.

Dieses Verlinkungs-Tool soll also mithelfen die Zahl der verwaisten Dokumente bzw. Sackgassen-Artikel zu vermindern, indem es die Verlinkungs-Arbeit der Editoren erleichtert. Es soll aber auch auf das Verhalten des Users positiv einwirken, indem es themenbezogene und dadurch interessante Verlinkungen bietet.

Im Grunde generiert das Programm Linkvorschläge; aus dem Grund wird die Vernetzung auch semi-automatisch bezeichnet. Diese Vorschläge berücksichtigen den Kontext und die Relevanz von den zu verlinkenden Wörtern.

Es werden nicht Verweise zu allen Substantiven vorgeschlagen. Mein Programm berechnet mittels den Methoden des Information Retrievals [7] als Kandidaten nur die relevanten Keywords für das Dokument. Auch zusammengesetzte Wörter werden einbezogen.

Die Vorgehensweise, dass die Wörter, die verlinkt werden sollen, vorher selektiert werden, verhindert eine Überfülle an Linkvorschlägen, was womöglich einer Überforderung durch den Autor zur Folge haben könnte oder, falls er zuviele dieser Vorschläge speichert, eine Beeinträchtigung der Lesbarkeit des Textes bewirken könnte. Es soll also ein sogenanntes Spaghetti-Dokument, welches zuviele Links im Text inkludiert hat, vermieden werden[45].

[4] Siehe mehr über Wikipedia unter 2.6.3

[5] Siehe Abschnitt 6.3.4 für weitere Details.

[6] http://www.hellomovies.com/

[7] Für mehr Informationen siehe Kapitel 5.

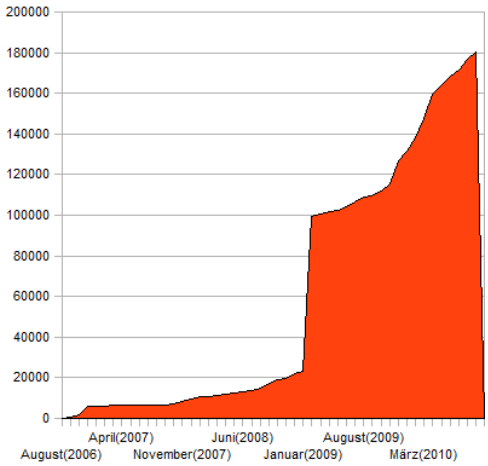

Abbildung 1.1: Die Steigerung der Anzahl der verwaisten Dokumente in Wikipedia seit 2006. Die radikale Kurve ab Februar 2009 existiert aus dem Grund, da ein automatisiertes Programm zur Identifizierung der verwaisten Seiten ab diesem Zeitpunkt eingesetzt wurde.

1.2 Aufbau und Kapitelübersicht

Insgesamt ist die Arbeit in zwei Abschnitte geteilt. Der erste Teil befasst sich hauptsächlich mit dem theoretischen Zugang zur Verlinkung in Hypertextsystemen, während der zweite Teil, den praktischen Part des Projektes thematisiert, also den Fokus auf den Verlinkungsassistenten hat.

Das Kapitel 2 zeigt die jahrelange Entwicklung der Hypertext-Systeme bis zum Wiki-System. Insbesondere wird auf ein wissenschaftliches Paper von Halasz, wo sieben Kriterien für zukünftige Hypertext-Systeme beschrieben werden, eingegangen[60].

Danach werden im Kapitel 3 andere Verlinkungs-Projekte sowie ihre Eigenschaften betrachtet. Außerdem werden die Projekte einer spezifischen Einteilung nach ihren Verlinkungskriterien unterworfen.

Desweiteren finden sich im Kapitel 4 Beschreibungen zu linktheoretische Paradigmen, welche nicht nur in Hypertext-Systemen zu finden sind. Insbesondere das Small-World-Problem sowie aber auch Netzwerk-Modelle von Strogatz-Watts werden ausgeführt. Die Navigation in solchen Netzwerken spielt in dem Kapitel eine besondere Rolle, da sie mit dem Browsen eines Users im Hypertext verglichen werden kann.

Im Kapitel 5 kann man mehr über die Methoden des Information-Retrieval, auf dem wichtige Algorithmen des Projekts basieren, in Erfahrung bringen.

Der praktische Teil wird im Kapitel 6 genauer ausgeführt. Das Kapitel beginnt mit der Problemstellung, welche zu dem Projekt führte, siehe dazu Abschnitt 6.1. Danach wird auf die Anforderungen des Programm eingegan-

gen, siehe dazu Abschnitt 6.2. Unter Abschnitt 6.3 folgen Beschreibungen über die Technologien, welche das Tool benützt, sowie über die Umgebung, in der es intergriert wurde. Darauf folgt unter Abschnitt 6.4 der Aufbau des Tools und die Einteilung in serverseitige, clientseitige und verknüpfende Komponenten, welche unter den Sektionen 6.5 6.6 6.7 weiter ausgeführt werden.

Am Ende findet sich unter Kapitel 7 die Konklusion und der Ausblick in die Zukunft des Projektes.

Kapitel 2

Hypertext-Systeme

2.1 Was ist Hypertext?

Hypertext kann als asynchron aufgebauter Text gesehen werden, welcher Links zu anderen Dokumenten enthält, während der Begriff Hypermedia einen Hypertext bezeichnet, der auch Bilder und Videos darin einbettet[121]. Oft wird der Ausdruck Hypertext auch für Hypermedia verwendet. Beide Definitionen stammen von Ted Nelson[1].

Hypertext-Systeme haben einen Bestand an Dokumenten, welche miteinander vernetzt sind und den sie verwalten. Sie werden in dieser Arbeit auch als Informationssysteme bezeichnet.

2.2 Entstehungsgeschichte

2.2.1 Memex

Die Idee des Hypertexts entstammt aus den vierziger Jahren. 1945 beschrieb Vannevar Bush, damals wissenschaftlicher Berater von dem amerikanischen Präsidenten Roosevelt, in "As We May Think" ein Informationssystem, in seinen Eigenschaften ähnlich dem assoziativen Denken des Menschen[119][25]:

> "The Human mind operates by association. With one item in its grasp, it snaps instantly to the next that is suggested by the association of thoughts, in accordance with some intricate web of trails carried by the cells of the brain."[2]

An die voran genannten Eigenschaften sollte sich sein Gedankenkonstrukt namens Memex, welches er beschreibt, annähern[25]:

[1] Siehe dazu in Abschnitt 2.2.3

[2] Frei übersetzt: Der menschliche Geist funktioniert über Assoziationen. Hat er eine Sache in seiner Umklammerung, kann er sofort zur nächsten gedanklichen Assoziation springen, aber nur in Übereinstimmung mit den vernetzten Pfaden der Gehirnzellen.

> "Consider a future device for individual use, which is a sort of mechanized private file and library. It needs a name and to coin one at random "memex" will do. A memex is a device in which an individual stores all his books, records and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory."[3].

Eine Maschine also, die ähnlich dem menschlichen Geist funktioniert, und welche als eine Erweiterung des Gedächtnis ihren Einsatz findet. Verglichen mit der heutigen Zeit wäre Memex nichts anderes als ein Computer mit Internetzugriff.

Vannevar Bush sah auch die Entstehung von Online-Enzyklopädien voraus [25]:

> "Wholly new forms of encyclopedias will appear, ready-made with a mesh of associative trails running through them, ready to be dropped into the memex and there amplified." [4]

2.2.2 Douglas Engelbart

Das Gedankenexperiment von Vannevar Bush wurde in den nächsten Jahren zur Inspiration vieler Computerpioniere, einer der besonders herausragt ist Douglas Engelbart. Er ist nicht nur Erfinder der Maus sondern entwickelte eines der ersten Hypermedia-Systeme. Es wurde anfangs NLS[5] bezeichnet, und später, als es verkauft wurde, in Augment umbenannt. Das System wurde ab 1962 am Stanford Research Institute entwickelt. Wissenschafter konnten zum Beispiel ihre akademischen Arbeiten in das System hochladen und darüber hinaus diese miteinander verlinken [88][42].

NLS enthielt nicht nur richtungsweisende Features das Forschungsfeld Hypertext betreffend. Am 9. Dezember 1968 gab Doug Engelbart eine Demonstration des Systems vor ungefähr tausend Computerwissenschafter, bei der er nicht nur die von ihm entwickelte Maus, dynamische Dateienverlinkung und eine Kollaboration über Netzwerk zweier Testpersonen mittels Video und Audioübertragung zeigte, sondern auch die Hypermedia-Funktionen von NLS präsentierte. Diese Vorstellung wurde später auch als "Mother of all Demos" bezeichnet[102].

[3] Stellen Sie sich vor, dass in Zukunft ein Apparat existiert, welcher als eine Art mechanisierte private Bibliothek funktioniert. Es braucht einen Namen und ich nenne es Memex. Ein Memex ist ein Gerät, in welchem der Benutzer seine Bücher, Aufnahmen, Dialoge speichert und welches in der Weise funktioniert, dass das Gespeicherte mit hoher Geschwindigkeit und Flexibilität abgerufen werden kann. Es soll für den Benutzer als eine Art Ergänzung zu seinem Erinnerungsvermögen dienen.

[4] Frei übersetzt: Neue Formen von Enzyklopädien werden entstehen, durchzogen von assoziativen Pfaden, bereit um ins Memex geladen zu werden und dieses zu vergrößern

[5] NLS steht für oN-Line System.

2.2.3 Project Xanadu

Wie gesagt stammt der Ausdruck Hypertext, sowie auch der weniger bekannte Term Hypermedia von Ted Nelson aus dem Jahre 1965. Hypertext war nichts anderes als der schon von Bush angedeutete elektronische Text mit eingefügten Verlinkungen. Mit Hypermedia wurde von Nelson eine Mischung zwischen Hypertext und Multimedia, also verlinkter Text mit Bilder, Videos und Töne, beschrieben.

Ted Nelson entstammt nicht den Computerwissenschaften sondern hat einen Abschluss in Philosophie. Die Idee eines allumfassenden Hypertext-Systems, welches er Xanadu benannte, ist seit über dreißig Jahren in Arbeit. Es ist somit eines der längsten Software-Projekte in der Computer-Geschichte. Mitlerweile existiert eine Xanadu Operating Company, die aber nur Teile des Projektes Xanadu vertreibt, denn bis heute war es nicht möglich, alle Vorstellungen von Ted Nelson zu realisieren[88].

Das Projekt Xanadu war als ein virtuelles Universallexikon geplant. Es sollte eine allumfassende Bibliothek werden, worin unterschiedlichste Publikationen miteinander verknüpft werden würden[120]. Folgende Features waren damals in Planung[88, 92, 93, 109, 120]:

Einzigartige Server-IDs Die Xanadu-Server sollten eindeutig identifizierbar sein.

Allgemeine Verlinkungsfreigabe Das Verlinken sollte jedem Benutzer in allen Artikeln gestattet sein.

BackLinks Endpunkte einer Verlinkung sollten Informationen über deren Anfangspunkte innehaben.

Hohe Usability des User-Interface

Lokale und externe Speicherung Die oft benutzten Dokumente sollten in eine lokale Datenbank gespeichert werden, während ein externes Backend für die ausgefallenen Informationen zuständig wäre.

Transklusion Diese Technik sollte es erlauben, dass beim Einbinden eines Zitats nicht vom Autor, den man zitieren möchte, Wort für Wort kopiert wird, sondern, dass es möglich ist Teile des Dokuments, also der Datei, die man benötigt, einzubinden.

Transcopyright Über Transklusion werden zusammengesetzte Files (Virtual Files) erstellt und der Rechteinhaber muss zur Nutzung dieser Funktion im Vorhinein seine Zustimmung erteilen.

Bezahlsystem Auch ein Bezahlsystem sollte inkludiert werden, das den Zugriff auf die Dokumente regelt. Das Geld sollte an die Autoren je nach Größe ihres Beitrags aufgeteilt werden. Ein durch Transklusion inkludierter Autor z.B. sollte auch entsprechend entlohnt werden. Diese Idee wurde bei den Micropayements wieder aufgegriffen.

Versioning Jede Änderung an einem Dokument sollte als neue Version gespeichert werden. Das alte Dokument sollte man zur Wiederherstellung

archivieren. Verlinkungen sollten auf alte Versionen bestehen bleiben und nicht automatisch auf die neueste Version zeigen. Die Löschung eines Dokuments bzw. der Verknüpfungen zwischen diesen wäre damit unmöglich.

Suche Die Durchführung der Suche sollte ohne Wissen über weitere Dokumenteigenschaften wie den Speicherort geschehen können.

Xanadu war und ist ein sehr ambitioniertes Projekt. Die meisten Features wurden schon in den 1960er Jahren von Nelson angedacht, was es umso bemerkenswerter macht. Es ergab sich in den 1980er Jahren auch eine Zusammenarbeit mit der Firma Autodesk, welche aber durch Nichteinhaltung von Abgabe-Terminen wieder gelöst wurde. Die Entwickler vermochten keinen funktionsfähigen Prototyp in der vertraglich festgelegten Zeit zu liefern[120].

Dass es nie zu einer befriedigenden Durchführung des Projektes kam, mag auf mehrere Faktoren zurückzuführen sein[88, 120]:

- Mangelnde Erfahrung von Ted Nelson bezüglich Software bzw. Hardware. Er hatte nur einen Abschluss in Philosophie und war im Grunde für die Visionen zuständig. Obwohl man hinzufügen muss, dass ihm diese Arbeit letztlich von Roger Georgery abgenommen wurde.
- Hardwareprobleme. Eine allumfassende Online-Bibliothek beinhaltet eine enorme Anzahl an Dokumenten. Nimmt man die Speicherung jeder Änderung in ein neues Dokument, so erhöht sich der Speicherbedarf exponentiell.
- Komplexe Features. Transklusion zum Beispiel war für die Entwickler ein sehr schwieriges Unterfangen.
- Mögliche Copyright-Probleme.

Trotz des Scheiterns wurden viele Ideen aus dem Projekt Xanadu in anderen umgesetzt, wie zum Beispiel im World Wide Web oder Hyper-G[92].

2.2.4 FRESS

Andries van Dam war einer der frühen Hypertext-Pioniere und wurde von Nelson beeinflusst. 1967 bis 1968 entwickelte er zusammen mit ihm an dem Hypertextsystem HES[6], welches von IBM finanziert wurde[101].

HES war eines der ersten funktionierenden Hypertext-Systeme seiner Art. Nach der Fertigstellung fand sich das Houston Manned Spacecraft Center als Käufer, wo HES zur Dokumentation der Apollo Missionen benutzt wurde[32].

Nach der Entwicklung von HES verfolgte Van Dam mithilfe seiner Studenten ein neues Projekt namens FRESS. Es enthielt eine Liste an Innovationen gegenüber älteren Systemen[36]:

- Keine Limitierung bei der Anzahl der Dokumente
- Keine Limitierung bei der Größe der Dokumente

[6] Hypertext Editing System

- Keywords konnte man zu den Knoten und Links anhängen
- Bidirektionale Links
- Rückgängig-Funktion (Undo)

Fress galt als ein sehr stabiles System, welches zwanzig Jahre auf den IBM-Mainframes der Brown Universität lief[88].

Andries van Dam hat mithilfe von Fress auch einen Beitrag zum E-Learning gebracht. Sein Projekt fand Anwendung an einer Schule als Unterstützungstool in einem Poesie-Kurs. Textbücher wurden von dem System elektronisch gespeichert. Studenten verfassten Online-Essays über einzelne Inhalte, und verknüpften diese gleichzeitig. Lehrende nutzten FRESS um Evaluierungen über die Essays zu verfassen und Verlinkungen zu setzen[38].

2.3 Hyper-G

Hyper-G ist ein an der TUGraz entwickeltes Hypertext-System. Es wurde zur ungefähr gleichen Zeit wie das Web erstellt und stand somit unter direkter Konkurrenz mit diesem, wobei es in manchen Features dem Web bis heute voraus ist[5]:

- Ausgeklügeltes User-Interface.
- Unterstützung von Multi-Lingualität
- Bidirektionale Links.
- Erleichterung der Kommunikation zwischen den Hyper-G-Servern.
- Interne Suche

Als Browser wurde Harmony eingesetzt[5].

Später emigrierte Hyper-G auch ins Web. Als kommerzielles Produkt names Hyperwave ist es zu finden unter http://www.hyperwave.com/d/.

2.4 7 Issues

Im Jahr 1988 kurz vor der Enstehung des World Wide Web veröffentlichte Frank G. Halasz Vorschläge in Form eines Artikelbeitrag namens "7 Issues, Reflections on Notecards". Es geht in dem Paper um sieben Probleme, die zukünftige Hypertext-Systeme zu lösen haben[60]:

1. Suchen über Querys. Eine Erweiterung zum einfachen Browsen soll entwickelt werden, womit es möglich sein soll über inhaltsbasierte Anfragen nach Dokumente im System zu suchen.
2. Erweiterung des Knoten und Linksystems in Richtung Zusammenstellungen sogenannten Composites, sodass man z.B. mehrere Knoten bzw. Dokumente in einem Knoten verbinden könne.

3. Dynamisches Netzwerk bzw. virtuelle Strukturen. Die Verlinkung zwischen den Knoten (Dokumenten) soll sich mit wechselnder Information ändern.
4. Automatisierung über künstliche Intelligenz. Ein Inferenzsystem zum Beispiel das neue Information zu einem Knoten erkennt und dann eine Verknüpfung herstellt.
5. Versioning. Zu vergleichen mit den heutigen Versioning-Tools wie Subversion[7] oder CVS[8]. Die Änderungen an den Knoten sollen gespeichert werden, so dass das veränderte Dokument wiederhergestellt werden kann. Die Forderung ist schon älter, vergleicht man dazu die angestrebten Features beim Projekt Xanadu[9].
6. Kollaboratives Arbeiten. Gleichzeitiges Editieren eines Artikel soll möglich gemacht werden.
7. Erweiterbarkeit und Anpassbarkeit. Das System soll so erweiterbar sein, dass es sich der Datenstruktur des Benutzers anpasst und nicht umgekehrt.

Die sieben Probleme waren anfangs von Halasz für eher kleine Hypertext-Systeme gedacht. Die Idee eines Projekts in Größenordnung Xanadu wurde da eher als unmöglich angesehen. Als das erste Paper von Tim Berners-Lee, dem Erfinder des Webs, veröffentlicht wurde, bezeichnete Halasz dieses abschätzig als weitere "Nelsonian Vision", also als weitere unmögliche Vision von Ted Nelson[59, 84].

Halasz gliederte auch Hypertext-Systeme nach fünf Dimensionen[58, 60]:

- Umfang. Es geht hier um die Frage für welche Größenordnung das System ausgelegt wird. Ein System zum Beispiel der Größe der allumfassenden Bibliothek Xanadu braucht andere Ressourcen, als eine einfache Notizen-Software.
- Schmökern vs Schreiben. Welche Tätigkeit also im Hypertext-System mehr im Vordergrund steht. Ob in dem Informationssystem mehr geschrieben oder mehr konsumiert wird.
- Aufgaben-Spezifizierung. Dieser Maßstab unterscheidet zwischen Fokusierung auf spezielle oder auf allgemeinere Aufgaben.
- Navigator vs. Architekt. Der Fokus des Navigators liegt auf den Knoten (Dokumente) und deren Inhalt und seine Aktivität ist diese zu durchforsten. Während dem Architekten die ganze Struktur des Hypertexts interessiert und diese zu manipulieren.
- Literat vs. Virtuellist. Insofern ist damit der Vergleich zwischen einer manuellen oder einer dynamischen Knotenstruktur gemeint. Ob das Netzwerk per Mensch oder per Programme erstellt wird.

[7] http://subversion.tigris.org/
[8] http://www.cvshome.org/
[9] Siehe Abschnitt 2.2.3

Im nächsten Abschnitt findet sich ein Überblick über die Entstehungsgeschichte des World Wide Webs und seinen Stationen vom Anfang dieses Hypertexts bis zur Erweiterung zum Web 2.0. Auch wird das Web aus der Sicht von Halasz durchleuchtet und an die vorher genannten Dimensionen gemessen.

2.5 World Wide Web

Tim Berners-Lee ist der Erfinder des World Wide Webs. Er arbeitet im Moment als Professor am Massachusetts Institute of Technology und ist der Vorsitzende des W3C-Konsortium[10] .

1980 entwickelte er ein Notizverwaltungs-Programm mit dem Namen ENQUIRE. Es enthielt die Möglichkeit Verlinkung zwischen Notizen zu setzen. Auch die Rückverfolgung der Links waren als Feature inkludiert[28].

Er arbeitete dann im Jahr 1989 im Cern. Dort existierten parallel viele Großprojekte, welche jahrelange Forschung bedurften, und es gab Probleme in der Kommunikation sowie beim Datenaustausch. Tim Berners-Lee dachte ein Hypertext-System an, um diese Schwierigkeiten zu vermindern.

1989 verfasste er einen Vorschlag in schriftlicher Form, welches ein Hypertext-System beschrieb, und sendete es an die ihm vorgesetzte Etage. In diesem Schreiben erwähnte er auch Ted Nelsons Xanadu. Nach einigen weiteren Bemühungen seitens Berners-Lee erhielt er schließlich von der Chefetage einen NeXT-Computer und den Hinweis ein solches Systen zu entwickeln[17,24,28].

Er programmierte in der Folge das World Wide Web und einen Browser-Prototyp. Am 17. Mai 1991 ging das Web offiziell auf den Cern-Rechnern an den Start und wurde am 6. August 1991 per FTP öffentlich zugänglich gemacht.

Das Web ist aus den folgenden Standards / Technologien aufgebaut:

HTTP-Protokoll Das HyperText-Transfer-Protokoll ist ein simples Protokoll für die Kommunikation zwischen Server und Browser. In seiner ersten Version 0.9 beinhaltete das Protokoll nichts weiteres als das GET-Kommando und nach einem Leerzeichen den entsprechende Universal Resource Identifier, wodurch man auf eine Ressource am Server zugreifen konnte[28].

1996 wurde das Protokoll dann zur Version 1.0 erweitert. Es kamen Befehle wie POST, HEAD und ein einfaches Caching-System hinzu. POST wurde dafür hinzugefügt um Daten an den Server zu schicken. HEAD brauchte man fürs Cachen, einfach gesagt, lieferte es einen Teil des Headers der Antwort vom Server zurück[19].

Beide Protokolle HTTP 1.0 und HTTP 0.9 hatten das Problem, dass für jede Übertragung, sei es Bilder oder Dokumente, eine neue TCP-

[10] www.w3.org

Verbindung gemacht wurde. Das verlangsamte die Übertragung für den Benutzer[15].

Aktueller Standard ist heute das HTTP 1.1., in dieser Version wurde die Übertragung beschleunigt, in dem nicht immer neue TCP-Verbindungen notwendig waren. Das Caching und das Authentifizieren wurden verbessert, in der Version 1.0 verschlüsselte man über Base64, was praktisch keine Sicherheit bot. Fünf neue Befehle wurden hinzugefügt[15, 43]:

- OPTIONS. Über den Befehl bekam man eine Auflistung der Möglichkeiten um eine Ressource abzurufen.
- PUT. Damit lassen sich Ressourcen auf einen HTTP-Server hochladen.
- DELETE. Damit lassen sich Ressourcen wieder löschen.
- TRACE. Zur Verfolgung welche Stationen das Dokument genommen hat.
- CONNECT. Soll hauptsächlich verwendet werden bei Proxies und Tunneln.

Uniform Resource Locator Um eine Ressource im WWW eindeutig zu identifizieren existiert die sogenannte URL. Im vorherbeschriebene HTTP-Protokoll werden URLs benützt, um Ressourcen eindeutig zuzuweisen. Sie besitzen die folgende spezifizierte Syntax:

```
http://<Host>:<Port>/Pfad/zu/der/Ressource?Suchteil
```

Der Port kennzeichnet die Schnittstelle zum HTTP-Server, wobei dieser, wenn nicht festgelegt, standarmäßig auf 80 liegt[20].

Das `http` am Anfang soll das Protokoll anzeigen mit dem gesendet wird[20]. Es existieren noch andere Protokolle wie zum Beispiel FTP[11] oder gopher.

Die Schrägstriche im Pfad zu der Ressource sind den Unix-Systemen entlehnt, während der doppelte Schrägstrich nach dem "`http:`" das Kennzeichen hätte sein sollen, dass danach der Hostname folgt. Diese Idee wurde vom Apollo Domain System kopiert[18].

HTML Die HyperText Markup Language , das auf dem SGML[12]-Format beruht, ist die Sprache mit der ein Hypertext-Dokument im WWW verfasst wird. Am Anfang in der Ur-Version von 1992 existierten hauptsächlich sogenannte text-orientierte Tags, sowie z.B.[27]:

`<TITLE></TITLE>` zur Kennzeichnung des Titels des Hypertext-Dokuments

`<A HREF= "" NAME= "" TYPE = ""></A>` zum Einfügen eines Links. Diesem konnte sogar ein Typ gegeben werden, was aber optional war.

[11] File Transfer Protokoll
[12] Standard Generalised Mark-up Language

In weiterer Folge wurde die Markup-Sprache erweitert um Bilder einzubetten, Tabellen zu zeichnen, zum Einbinden von CSS[13], womit man die Darstellungs-Eigenschaften definieren vermag, und vielem mehr. Zur Zeit ist HTML 5 aktuell. An weiteren Versionen von HTML und weiteren Standards wird am W3C[14], am Word Wide Web Consortium, an dem Tim Berners-Lee Vorsitzender ist, gearbeitet

Ein Kernpunkt des Erfolgs vom Web war auch die Erstellung von effizenten Programmen zur Anzeige der HTML-Informationen, sogenannten Browsern. Wäre die Entwicklung bei den Browsern nicht fortgeschritten, hätte der Erfolg vom Web nicht diese Dimensionen erreicht.

Der Ur-Vater des Browsers wurde von Berners-Lee WorldWideWeb genannt und hatte nicht nur lesende Eigenschaften, sondern über diesen ließen sich auch Homepages erstellen.

Anfänglich liefen die Browser hauptsächlich auf unix-basierten Betriebssystemen. Der erste Browser welcher auf Windows-Systemen lief, hieß LineMode und wurde von Nicola Pellow 1991 entwickelt[87].

Als Durchbruch bei den Browsern galt Mosaic. Mit diesem war es möglich den Hypertext mit Bildern darzustellen. Er wurde von Marc Andreessen und anderen programmiert. Andreesen entwickelte in späterer Folge den auf Mosaic basierenden Browser Netscape[85].

2.5.1 Das World Wide Web aus der Sicht von Halasz

2001 äußerte sich Halasz in einem Paper zum Web und verglich die bisherigen Entwicklungen dieses Hypertext-Systems mit seinen Kriterien, welche er in seinen vorangegangenen Arbeiten definiert hat[15][59].

Zum einen verglich er die Eigenschaften des Webs mit den sieben Merkmalen, die er 1988 für die nächste Generation der Hypertextsysteme gefordert hatte. Seine Sicht bezieht sich auf das ganze Web, nicht auf einzelne Web-Applikationen:

1. Suche. Die Möglichkeit dazu findet sich laut Halasz über die Suchmaschinen. Das Web tendiert hin zu inhaltsbasierte Suchanfragen anstatt zum strukturbasierten Ansatz, abgesehen zum Beispiel von dem verlinkungsorientierten Aspekt im PageRank-Algorithmus.
2. Zusammenstellungen(Composites). Die Möglichkeit Webseiten zusammenzufügen lässt sich in der über HTML 4 standardisierten Frames-Methode finden. Mithilfe von Frames ist es möglich verschiedene Artikel bzw. Ressourcen ohne Restriktion auf ihren Standort miteinander verbinden.
3. Dynamisches Netzwerk bzw. virtuelle Strukturen. Es finden sich Seiten

[13] Cascading Style Sheets
[14] http://www.w3c.org
[15] Weitere Informationen darüber findet man unter 2.4

im Web, die erst bei Abruf berechnet und erstellt werden. Auch existieren dynamische Verlinkungen, die über Datenbankanfragen berechnet werden.

4. Artificial Intelligence. Berechnungen bei Web-Applikationen, die den Hypertext aktuell halten existieren, jedoch in den meisten Fällen ohne die Hilfe von einer Artificial Intelligence.
5. Versioning. Existiert praktisch bei solchen Webpages, die einen Nutzen davon ziehen. Wikis wären ein Beispiel.
6. Kollaboratives Arbeiten. Erleichterung des kollaborativen Arbeitens wird im Web angestrebt. Halasz unterscheidet zwischen Infrastruktur zur Unterstützung von Zusammenarbeit und eine Hilfe für das soziale Interagieren im Netz.
7. Erweiterbarkeit und Anpassbarkeit. Das Web ist ein System, welches Erweiterbarkeit und Anpassbarkeit fördert. Die WWW-Technologien sind unkompliziert zu lernen, so dass auch unbegabte Programmierer interessante Web-Projekte erstellen vermögen.

Wie schon unter 2.4 erwähnt, kreierte Halasz ein Dimensions-Schema für Hypertext-Systeme. Diese Schema wird nun auf das Web angewandt:

- Umfang. Das Web könne sich perfekt skalieren, ob für die Größe einer Applikation wie es Ted Nelson vorschwebt oder einem kleineren System in der Größe von NoteCards.
- Schmökern vs Schreiben (Browsing vs Authoring). Im Web veröffentlichen wenige, was viele lesen, meint Halasz im Jahre 2001, wobei die Tendenz zu mehr Authoring existiert. Bezogen auf das zukünftige Web 2.0 siehe Abschnitt 2.5.2, scheint er da richtig zu liegen.
- Aufgaben-Spezifizierung. Spezifizierung von Aufgaben ist beim Web schwer möglich, man kann es für alle möglichen Anwendungsgebiete einsetzen, wobei Standards für bestimmte Aufgaben entwickelt werden, z.B.: SMIL[16] oder Common Micropayment Markup[17].
- Navigateur vs Architekt. Das Web geht laut Halasz eindeutig in die Richtung Navigateur. Es wird mehr Wert auf die Knoten (also Dokumente) gelegt, als auf die ganze Struktur.
- Literat vs. Virtuellist. Beide sollen im Web existieren , wobei eine Gruppe vergessen wurde, nämlich die Geschäftsleute, die das Internet als Quelle neuer Einkommen sehen.

2.5.2 Web 2.0

Der Begriff Web 2.0 kann mehrere Bedeutungen beinhalten. Es ist ein Begriff der vielfältig aber auch oft verwendet wird, vielfach auch aus Marketing-

[16] http://www.w3.org/AudioVideo/
[17] http://www.w3.org/TR/WD-Micropayment-Markup/

Web 1.0	**Web 2.0**
DoubleClick	AdSense
OFoto	Flickr
Britannica Online	Wikipedia
Persönliche Websites	Blogging
Content Management System	Wikis [18]
Taxonomie	Folkonomie

Tabelle 2.1: Vergleich Web 1.0 - Web 2.0 anhand der Gegenüberstellung von Projekten, Webtechnologien, Klassifikationsschemen des alten und des weiterentwickelten Webs[91]

Gründen. Worin sich viele Web 2.0-Definitionen einig sind, ist, dass das Web eine Art Evolution durchläuft, sich sozusagen verändert.

Der Begriff Web 2.0 impliziert auch ein Web 1.0. Den Unterschied zwischen Web 1.0 und Web 2.0, also die Merkmale dieses Übergangs, lässt sich vereinfacht an einer Gegenüberstellung von bekannten Web-Unternehmen, Klassifikationsschemen und Web-Technologien , die einerseits als Vertreter des alten und andererseits als Vertreter des neuen Webs gesehen werden können, beschreiben[91]. Siehe dazu Tabelle 2.1.

Auch lässt sich als Charakteristikum für das Web 2.0 eine Reihe an technologischer Neuerungen, die einige der vorangegangenen Applikationen nutzen, sowie Standards festmachen, die in der folgenden Aufzählung besprochen werden[4]:

Ajax Ajax ist eine Abkürzung für Asynchronous JavaScript + XML[52] und hat vielfältigen Einsatz im WWW. Man vermag zum Beispiel desktopähnliche Webapplikationen mithilfe von Ajax entwickeln. Die Keykomponente von Ajax ist, dass mithilfe von Javascript neue Inhalte zum Server gesendet sowie vom Server empfangen werden können, ohne dass der Benützer manuell eingreift. Diese Eigenschaft wird über das XMLHttpRequest gesteuert.

HTML/XHTML Standardisierung der Repräsentation im Browser.

CSS Über CSS lassen sich Stylingdefinitionen setzen. Es trennt praktisch den Inhalt von der Darstellung.

Document Object Model Für Javascript ist das die Schnittstelle zur Manipulation des Dokuments.

XML Wird als Standard verwendet zum Datenverkehr von Informationen.

XSLT Wird als Standard verwendet zum Datenverkehr von Styling-Attributen.

REST REST ist eine Abkürzung für *Represential State Transfer* und ist im Grunde nur ein architektonisches Prinzip, wie das Web funktionieren sollte. Es soll nicht verwechselt werden mit einem Standard.

Jede Ressource sollte nach REST eine URI zugewiesen werden und die Kommunikation erfolgt über das HTTP-Protokoll.

SOAP SOAP bedeutet *Service Object Access Protocol* verwendet komplexe Protokolle zum Datenverkehr im Gegensatz zum von REST propagierten HTTP-Protokoll. es wird oft kombiniert mit WSDL, was soviel bedeutet wie *Web Service Description Layer*. Das Ganze wird für Web-Services eingesetzt.

Micro-Formats Micro-Formats werden benützt um zusätzliche oft semantische Informationen in die Seite einzufügen, die maschinell interpretiert werden kann. Ein Beispiel dazu wäre das hCard-Format, mithilfe man persönliche sowie organisatorische Daten einbinden vermag.

Open-APIs Application Programming Interfaces sind Schnittstellen, welche Programmierer verwenden können zum Zwecke der Erweiterung ihrer Programme. In dem Fall sind Schnittstellen gemeint, die über das Web verwendet werden, um Services zu erreichen, die das Webangebot erweitern. Ein Beispiel dazu wäre die GoogleMaps API, welches eine Schnittstelle zu der populären Geo-Webapplikation bietet.

Ein wichtiger Punkt, der nun im nächsten Abschnitt extra behandelt wird, da er im Kontext meines Projekts eine potente Rolle spielt, ist das Wiki-System, bzw. Wiki-Frameworks.

2.6 Wikis

Wikis sind Teil der Web 2.0-Evolution und spielen im heutigen Web eine wichtige Rolle. Abgesehen davon wurde mein Projekt in einem Wiki-System implementiert .

Der Name Wiki, eigentlich WikiWiki, entstammt aus dem Hawaiianischen und bedeutet soviel wie: *Fast, speedy; to hurry, hasten; quick, fast, swift* also auf gut Deutsch "schnell, flink, hurtig". Ein Wiki soweit definierbar ist eine frei erweiterbare Sammlung von miteinander verlinkten Dokumenten, worin jede Seite ohne irgendwelche zusätzliche Erweiterungen vom Browser aus editierbar ist[73]. Die bekannteste Wiki-Applikation stellt Wikipedia dar[19].

2.6.1 Geschichte

Das Wiki-Konzept wurde von Ward Cunningham 1993 entworfen. Nachdem er die Entwicklung abgeschlossen hatte und sie veröffentlichen wollte, schrieb er einem Freund am 16. März 1993 [20]:

[19] http://wikipedia.org
[20] http://c2.com/wiki/mail-history.txt

> [...] I've put up a new database on my web server and I'd like you to take a look. It's a web of people, projects and patterns accessed through a cgi-bin script. It has a forms based authoring capability that doesn't require familiarity with html. I'd be very pleased if you would get on and at least enter your name in RecentVisitors. I'm asking you because I think you might also add some interesting content. I'm going to advertise this a little more widely in a week or so. The URL is http://c2.com/cgi-bin/wiki. Thanks and best regards. – Ward [21]

Am 25. März schließlich, dem eigentlichen Geburtsdatum des Wikis, stellte Cunningham sein Wiki wie folgt vor:

> Think of it as a moderated list where anyone can be moderator and everything is archived. Its not quite a chat, still, conversation is possible[22][1].

Im Grunde sollte es "quick-web" genannt werden, jedoch entschied sich Cunningham einen seiner Meinung nach interessanteren Titel zu formulieren und nannte seine Implementation WikiWikiWeb, welches wie gesagt aus dem Hawaiianischen entstammt. Der Grund für den hawaiianischen Einschlag lässt sich zurückführen auf eine Erinnerung von Cunningham an einen sogenanten Wiki Wiki Bus, einem Shuttlebus , betrieben an einem hawaiianischen Flughafen[30].

Das sogenannte WikiWikiWeb, welches unter http://c2.com/cgi-bin/wiki zu finden ist, wuchs langsam jedoch stetig und wurde zu einem der meistbesuchtesten und größten Webportalen, die sich mit Design Patterns beschäftigen. Siehe dazu Tabelle 2.2 und Abbildung 2.1 für weitere Statistiken.

2.6.2 Das Wiki-Konzept

Was ist also ein Wiki und welche Eigenschaften, abseits von der technischen Realisierung, sollte es haben um danach benannt zu werden? Cunningham und andere haben darauf in mehreren Punkten Antwort gegeben[39][73]:

- Freigabe von allen Dokumenten zur Veränderung.
- Editieren über den Browser ohne zusätzliche Plugins.

[21] Frei übersetzt: Ich habe eine neue Datenbank auf meinem Web-Server installiert, und es wäre schön, wenn du sie dir mal ansiehst. Es ist ein Web für Leute, Projekte und Patterns, zugänglich gemacht über ein cgi-Skript. Es braucht keine Html-Kenntnisse um Text zu editieren. Ich wäre sehr erfreut wenn du deinen Namen in den RecentVisitors eintragst. Ich frage nur, weil ich glaube dass du interessanten Inhalt hinzufügen könntest. Ich werde in der nächsten Woche oder so noch einmal Werbung dafür machen. Die URL ist http://c2.com/cgi-bin/wiki.

[22] Frei übersetzt: Man sollte es als eine moderierte Liste ansehen, wo jeder Moderator ist und alles archiviert wird. Es ist kein Chat, jedoch Konversation ist möglich.

Datum	Seitenanzahl
29.November 1994	-
15.Dezember 1995	2426
1. Dezember 1996	5134
31. Dezember 1997	10600
25. März 1996	14554
2. Dezember 2000	62919

Tabelle 2.2: Anzahl der Seiten von WikiWikiWeb zu verschiedenen Zeitpunkten(http://c2.com/cgi-bin/wiki) [31]

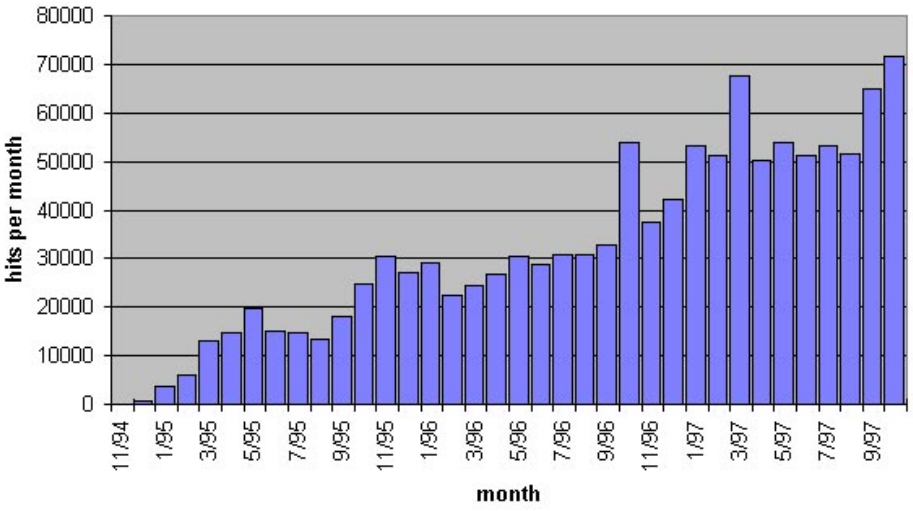

Abbildung 2.1: Statistik zu den Besucherzahlen von WikiWikiWeb

- Förderung von Identifizierung der Thematik des Dokuments. Treffendende Dokument-Titel in den URLs sollen die Verlinkung erleichtern.
- Keine fertigen statischen Seiten, sondern Stärkung von Zusammenarbeit an den Dokumenten.
- Nichtlineare Hypertextstruktur. Über assoziative verlinkte Texte erfolgt Navigation und Befriedigung des Informationbedürfnis.
- Wenig Vorkenntnisse zum Editieren. Der User benötigt keine ausgedehnten HTML-Kenntnisse um die Seiten nach seinen Vorstellungen zu editieren bzw. formatieren und verlinken. Dafür verwendet man die Wiki-Syntax.
- Sozialer Hypertext. Technik steht weniger im Vordergrund als sozialer Kontext im Projekt, inhaltliche Divergenzen und Arbeitsabläufe.

Den wahren Durchbruch erfuhr das Wiki eigentlich erst mit der steigenden Popularität von Wikipedia.

2.6.3 Wikipedia

Die heute sehr populäre Online-Enzyklopädie wurde 2000 mit dem Namen Nupedia von Jimmy Wales und Larry Sanger gegründet und war eigentlich dazu gedacht geprüftes Expertenwissen zu veröffentlichen. Vom technischen Standpunkt basierte Nupedia auf keinem Wiki-System und wurde von Expertenteams verwaltet. Im ersten halben Jahr wurden nur zwei Artikel von dem Prüfungsverfahren abgesegnet, was das ganze System in Frage stellte[44].

Es mussten Änderungen her und die zündende Idee kam von Ben Kovitz einem alten Freund Larry Sangers, welcher sich mit dem WikiWikiWeb und speziell mit dem Wiki-Konzept von Cunningham beschäftigte und Sanger darüber erzählte. Dieser war begeistert von dem Konzept und überlegte sogleich ob Wikis nicht für Enzyklopädien geeignet wären[75]:

> Instantly I was considering whether wiki would work as a more open and simple editorial system for a free, collaborative encyclopedia, and it seemed exactly right[23].

Larry Sanger schlug schließlich vor eine Wiki-Engine einzusetzen und praktisch alle Artikel zum Editieren freizugeben. Wikipedia.com ging am 15. Januar 2001 online und beinhaltete nach dem ersten Jahr 20.000 Artikel[44].

Eine alte Version der Wiki-Software 2001 findet sich auf http://nostalgia.wikipedia.org/wiki/HomePage.

In technischer Hinsicht wurde bei Wikipedia anfangs als Wiki-Engine das UseModWiki eingesetzt. Die Seiten wurden nicht in eine Datenbank sondern als einzelne Files gespeichert, was im Endeffekt aufgrund der wachsenden Benutzeranzahl das System verlangsamte. Die Lösung dazu erfolgte über den deutschen Programmierer Magnus Manske, auch Teil der Wikipedia-Community. Er implementierte das Wiki-Engine namens MediaWiki in PHP[24]. Im Grunde baute diese neue Version auf dem UseModWiki auf, wobei eine der wichtigsten Erweiterungen die Datenanbindung an MySQL[25] darstellte. Eine weitere wichtige Funktion war `diff`, denn dadurch war es möglich verschiedene Versionen eines Dokuments miteinander zu vergleichen [75].

Heute existieren Ableger der Wikipedia in allen möglichen Sprachen. Die meisten Seiten beinhaltet die englische Version, das an die 3,5 Millionen Dokumente verwaltet[114].

[23] Frei übersetzt: Sofort überlegte ich ob ein Wiki als ein offeneres und simples Editier-System für eine freie kollaborative Enzyklopädie funktionieren würde und es schien genau richtig.

[24] http://www.php.net/

[25] http://www.mysql.com/

Larry Sanger stieg 2002 aus dem Projekt aus, da es seiner Meinung nach an gegenseitigen Respekt fehlte und zumeist Trolle oder schwierige Menschen an dem Projekt tätig wären[99]. So gründete er eine eigene Online-Enzyklopädie namens Citizendium[26].

Die Verbindung von Wikis mit Web 2.0 erfolgte nur aus dem Grund, da Wikipedia zu der Zeit populär wurde, als der Begriff Web 2.0 aufkam. Wikis, betrachtet man die Geschichte von dem WikiWikiWeb von Cunnigham, existierten schon viel früher.

2.6.4 Wiki-Frameworks

Es existiert eine Vielzahl von Wiki-Frameworks, über welche sich Wikis realisieren lassen. Über http://c2.com/cgi/wiki?WikiEngines findet sich ein Versuch alle Wiki-Frameworks zu dokumentieren. Außerdem kann man unter http://www.wikimatrix.org/ durch die Eingabe verschiedener Attribute ein für sich passendes Wiki-Framework suchen. Hier folgt nun eine Auswahl von wichtigen Wiki-Frameworks[26]:

MediaWiki Wurde konzipiert für Wikipedia, basiert auf PHP und benützt MySQL als Daten-Backend. Es ist dank Wikipedia eines der populärsten Wiki-Frameworks, und existiert mittlerweile in der Version 1.16. Da es in einer der meistbesuchtesten Seiten erfolgreich eingesetzt wird, kann man davon ausgehen, dass es für große Wiki-Projekte ein geeigneter Kandidat ist. Weitere Informationen finden sich unter http://www.mediawiki.org und unter dem Abschnitt 2.6.3

MoinMoin Ist in python geschrieben und kann auch als eine Art Personal-Wiki verwendet werden. Die Daten werden in einfache Files geschrieben. Es unterstützt Subpaging, also hierarchische Strukturen, sowie Templates mit denen das Aussehen der Seiten geprägt werden kann. Es existiert ein modulares Plugin-System, welches den Basis-Code unangetastet lässt. Weitere Informationen finden sich unter http://moinmo.in/ .

PHPWiki Ist ein Klon des ersten Wikis WikiWikiWeb und wurde, wie der Namen schon andeutet, in PHP entwickelt. Jeder kann Seiten verändern, und es existiert eine hohe Flexibilität bei den Daten-Backends (MySql, PostgreSQL, Einfache Files etc.). Das Projekt wurde 2000 gegründet und existiert zur Zeit des Schreibens der Diplomarbeit in der Version 1.4. Weitere Informationen finden sich unter http://sourceforge.net/projects/phpwiki/

TikiWiki Im Grunde ist die Wiki-Software nur ein Bruchteil dessen, was im Angebot von TikiWiki vorhanden ist, denn inkludiert ist auch eine CMS-Funktion, Blog-Funktion, Forums-Funktionalität(phpBB) uvm.

[26] http://c2.com/cgi/wiki?TopTenWikiEngines

Es existieren Diskussionen darüber wie dieses All-In-One-Paket umgesetzt wurde, siehe http://c2.com/cgi/wiki?TikiWiki. Weitere Informationen finden sich unter http://tiki.org.

WikkaWiki Ist ein Fork von dem alten WakkaWiki, das seit 2004 eingestellt wurde und basiert auf PHP. Es setzt auf Flexibilität, Geschwindigkeit und Erweiterbarkeit. Weitere Informationen finden sich unter http://wikkawiki.org/HomePage.

FoswikiEngine Kann man als Erweiterung zu TWiki sehen. Aufgrund von Missstimmigkeiten zwischen der Community und Autoritäten von TWiki[27] spaltete sich FosWiki ab. Es orientiert sich in Richtung Enterprise-Wiki und ist in Perl geschrieben. Es beinhaltet einen sehr mächtigen WYSIWYG-Editor und setzt auf Dynamisierung der Seiten, was laut Homepage besonders für Unternehmen interessant sein soll. Im Daten-Backend werden reine Files verwendet. Das Wiki ist im Übrigen kompatibel zu TWiki. Weitere Informationen finden sich unter http://foswiki.org/.

MojoMojo Ist programmiert in Perl und eigentlich mit der Herausgabe der ersten Version im Jahr 2007 relativ jung. Es setzt im Gegensatz zu vielen anderen Wikis auf Baumhierarchien bei den Seitenstrukturen. Weitere Informationen finden sich unter http://mojomojo.org/

JSPWiki Siehe dazu Abschnitt 6.3.3.

2.7 7 Issues und das Web 2.0

Der Vergleich Web mit Halasz wurde insofern in dem Unter-Abschnitt 2.5.1 gezogen als dass das World Wide Web gesamtheitlich anhand den halaszschen Kriterien damit verglichen wurde. Dieser Vergleich beschäftigt sich mit einzelnen Web-Applikationen, die zum Web 2.0 zu zählen sind und ihre Erfüllung der Kriterien von Halasz, wobei auch Web-Projekte im Akademischen Bereich hinzugezählt werden.

Es sollen in dem Abschnitt Web 2.0 Projekte nicht nur an den originalen sieben Kriterien gemessen werden, sondern die Kriterien wurden eine genaueren Einteilung unterzogen, welche nachfolgend beschrieben werden[83]:

Suche Suche ist ein relativ allgemeiner Begriff, so wird er unterteilt in:

- Inhaltszentrierte Suche. Einfache Suche. Simpel gesagt herrscht die Frage ob ein Wort oder Phrase im Text vorhanden ist.
- Kontextzentrierte Suche. Metadaten werden genutzt um zusammenhängende Dokumente zu finden.
- Strukturelle Suche. Das Hypertext-System wird als großer Graph verstanden in dem nach Mustern gesucht wird.

[27] http://twiki.org/

Zusammenstellungen(neue Strukturen) Hier geht es nicht nur um sogenannte Composites(Zusammenstellungen) sondern mittlerweile um die Erweiterung des Hypertextes durch andere Strukturen:

- Typisierte n-ary Links. Was soviel bedeutet, dass man erstens Links einen Typen zuweisen kann, z.B. Link auf eine Quelle und zweitens dass der Link mehrere Ziele haben kann.
- Zusammenstellungen(Composites). Wurde schon früher öfters erwähnt. Mehrere Seiten werden zu einer einzelnen zusammengestellt. Ähnlich der Transklusion von Ted Nelson.
- Erweiterte Navigations-Strukturen. z.B.: Geführte Touren.
- Spuren(Trails). So dass der Benutzer weiß, wo er gewesen ist, oder um dem Benutzer durch die Analyse der Trails Empfehlungen über weitere Seiten zu erstellen.

Dynamisch, Adaptiv Dynamische und Adaptive Systeme stehen in einem engen Verhältnis zueinander, deshalb werden diese in einer Kategorie gesehen. Folgende vier Kriterien werden gesetzt:

- Dynamischer Inhalt. Damit ist im Allgemeinen dynamische Erstellung von Knoten und deren Inhalt gemeint.
- Dynamische Strukturen. Dynamische Erstellung von Verlinkungen, Touren und anderen Strukturen.
- Berechnungen das Hypertext-Netzwerk betreffend. z.B.: Berechnungen durchführen aufgrund von Linkpfade, die verfolgt wurden.
- Personalisierung. Der Benutzer kann seine persönliche Ansicht auf die Dokumente und Links einstellen.

Versioning

- Knotenbezogenes Versioning. Es werden (inhaltliche) Änderungen gespeichert, die Dokumente betreffen. Eine History wird zumeist erstellt von all den Veränderungen zum Zwecke der Revidierung.
- Netzwerkbezogenes Versioning. Es liegt der Augenmerk mehr auf Verlinkungen und auf die Netzwerke, die die Links bilden. Speicherung der Änderungen an solchen Netzwerken wird durchgeführt um diese später zu revidieren.

Benutzerrechte + Kollaboration Alles was mit Editieren zusammenhängt.

- Privates Kommentieren. Das Benutzerrecht eigene Knoten auch zu kommentieren.
- Öffentliches Kommentieren. Jeder Benützer darf jeden öffentlichen Knoten kommentieren.
- Globale Zusammenarbeit. Alle Benützer haben die Fähigkeit unabhängig von anderen Benützern am gleichen Dokument zu arbeiten.

- Eingeschränkte Zusammenarbeit. Die Fähigkeit, dass nicht alle Benützer sondern spezielle Gruppen am gleichen Dokument unabhängig voneinander arbeiten.
- Erweiterbarkeit. Die Möglichkeit des Benützers das System an Funktionalität zu erweitern.

Die vorher genannten Kriterien wurde von David E. Millard und Martin Ross an bekannten Web 2.0- sowie akademischen Applikationen getestet und eine Tabelle unter Abbildung 2.2 mit den Ergebnissen erstellt. Kurzbeschreibung zu den Applikationen:

Flickr Eine Applikation zum Fotos hinaufladen/herzeigen/verwalten etc. [28]

MediaWiki Siehe Beschreibung unter 2.6.4 unter Abschnitt 2.6.3

TWiki Ein populäres Wiki-Engine basierend auf Perl[29].

WordPress Sehr bekannte Blogging Software, mit CMS-Eigenschaften [30].

Annotea W3C-Projekt, zum Austausch von Metadaten. Ist implentiert im Amaya-Browser. Es existieren auch Anstrengungen, diese Funktionalität für Firefox zu implementieren[31].

Xspect Implementiert die XLink-Spezifikation, eine XML Linking Sprache über welche die Verlinkung von Ressourcen möglich wird, wobei diese nicht nur unidirektionalen Charakter hat. Es lassen sich auch multidirektionale Verweise setzen[32].

OHM / WWW Damit ist gemeint ein Open Hypermedia System basierend auf dem Word Wide Web, speziell getestet wurde über das ältere Dexter-Based Framework, welches Applets verwendet und bi-direktionale Verlinkung als Feature enthält[57].

Interessant zum Anmerken an Abbildung 2.2 ist, dass im Hinblick auf erweiterte Verlinkung also mehreren Linktypen und die Fähigkeiten mehrere Ziele als Link anzusteuern, sowie Zeiger auf ein Dokument zurückzuverfolgen, nur im Forschungsbereich implementiert wurde. Wobei man hier einen reinen pragmatischen Ansatz zieht und die Frage stellen muss, ob eine erweiterte Verlinkung überhaupt gebraucht wird in den anderen Systemen.

Versioning wird hauptsächlich dort verwendet, wo es einen Sinn macht, nämlich bei Wikis, die starken Änderungen an den Knoten sowohl auch an den Kanten unterworfen sind.

Der Vergleich Web 2.0 und den Kriterien des Hypertext-Pioniers Halasz ist womöglich im Endeffekt etwas schwierig, da die Evolution zum Web 2.0 und ihren Applikationen anhand den Bedürfnissen des Users gebaut wurden.

[28] www.flickr.com
[29] http://twiki.org
[30] www.wordpress.com
[31] http://www.w3.org/2001/Annotea/
[32] http://www.w3.org/TR/xlink/

	Web 2.0					Academic / Research	
	Flickr	MediaWiki	TWiki	WordPress	Annotea	Xspect	OHS/WWW
Content Search	X	X	X	X	X	+	X
Context Search	X	+	+	X	X	+	X
Structural Search	+				X	+	+
Trails	X			X			
Composition	X	X	X	X	X	X	X
Dynamic Content	X	X	X	X	X	X	+
Dynamic Structure	+			X	X		+
Typed n-ary links						X	X
Other Navigational Structures	X	X	X	+	+	X	X
Computation over the Network	X	X	X	X		X	X
Entity Versioning		X	X				+
Network Versioning			X				
Private Annotation	X	X	X	X	X	X	+
Public Annotation	X	X	X	X	X	X	+
Global Collaboration	X	X	X	X	+	+	+
Restricted Collaboration	X	X	X	X	+	+	+
Personalisation	X	+	X	X	X	X	X
Extensibility	X	X	X	X	+	X	X

Abbildung 2.2: Die Eigenschaften von Web 2.0 Applikationen verglichen mit den Kriterien von Halasz[83]. (*) voll implementiert. (+) teilweise implementiert.

Ein theoretischer Ansatz wie der von Halasz war klarerweise keinem so ausgedehnten Feedback-Zyklus unterworfen, wie der bei der Weiterentwicklung des Webs.

Kapitel 3

Automatische Verlinkung

Nachdem auf Hypertext-Systeme im vorangegangenen Kapitel der Fokus gelegt wurde, widmet sich dieses Kapitel den wissenschaftlichen Arbeiten bzw. Projekten, welche sich in den vergangenen Jahren mit automatischer Verlinkung in Hypertext-Systemen beschäftigten. Dadurch soll ein Überblick über die akademischen Arbeiten in diesem Forschungsfeld geschaffen werden, welche eine hohe Themenrelevanz zu dem Projekt meiner Diplomarbeit innehat.

Wie im Abschnitt 2.4 ausgeführt, empfahl Halasz u.a. ein dynamisches Hypertext-Netzwerk und eine Art höherwertiges Berechnungsverfahren, welches unterstützend in das System inkludiert werden solle. Die Kombination beider Features kann als Vorschlag für einen automatische Verlinkungs-Algorithmus in solch einem Informationssystem gesehen werden.

Es existieren jedoch auch viele Probleme die für eine automatische Verlinkung in großen Informationssystemen sprechen:

Wartungsaufwand Für große dynamische Dokumentsammlungen ist der Wartungsaufwand durch Editoren zu hoch[106]. Im Falle der Online-Enzyklopädie Austria-Forum, welches über 60.000 Artikel enthält, die einer ständigen Editierung und Erweiterung unterworfen sind, ist es praktisch unmöglich die Verlinkungen der Dokumente untereinander händisch aktuell zu halten.

Fehlende Konsistenz bei Verlinkungen In kollaborativen Informations-Systemen werden die Dokumente von mehreren Personen editiert. Jeder dieser Autoren hat ein spezifisches Wissen und verlinkt dadurch die Dokumente verschiedenartig[41].

Verschiedenartiges Informationsbedürfnis Jeder User hat ein eigenes Informationsbedürfnis, je nachdem welche Ressourcen er vorher besucht hat. Dynamische Verlinkungsalgorithmen könnten sich diesem Wissen anpassen und daran abgestimmte Vorschläge bringen[117].

Verwaisung und Sackgassen-Artikel Dokumente, die außerhalb des Hypertext-Netzwerks liegen oder solche die keine weiterführende Verknüpfungen im Text haben und praktisch eine Sackgasse bilden, sind sicherlich

Problemfälle, welche man bei manueller Vernetzung in Informationssystemen öfters antrifft.

Tote Links Das sind jene Links, die auf keine vorhandene Ressource zeigen und normalerweise zu einem 404-HTTP-Fehler führen. Das Web selbst hat keinen Mechanismus implementiert, die toten Links zu markieren. Es muss zu einer externen Lösungsmöglichkeit gegriffen werden[33].

Es existieren mehrere Arten der automatischen Verlinkung. Zum einen gibt es den Ansatz der dynamischen Links, welche nicht in den Text eingebunden sind und sich wechselnder Veränderungen des Dokumentenbestands anpassen. Zum anderen gibt es statische Links, die fester Bestandteil des Hypertext werden und sich nach der Erstellung nicht mehr ändern.

3.1 Dynamische Vernetzung

Das Hauptkriterium der dynamischen Verlinkung ist, dass die Links erst dann berechnet werden, wenn der Benützer diese braucht.

Es folgt eine Auflistung von Vorteilen und Nachteilen, welche die dynamische Vernetzung mit sich bringt[10]:

Vorteile

\+ Aktualität. Gibt den aktuellen Stand der Dokumente wieder.

\+ Kontext-Sensitivität. Kann sich neuen Informationen anpassen.

\+ Ökonomisch. Links werden nur erstellt, wenn sie gebraucht werden.

Nachteile

\- Wiederholung von gleichen Berechnungen. Bei eher statisch tendierenden Daten ist es ein Nachteil, bei sich schnell verändernden Dokumenten ein Vorteil.

3.1.1 Projekte

Ältere Projekte, die eine dynamische Verlinkung implementierten, finden sich Anfang der 90er Jahre. Es existierte zu der Zeit das Open Hypertext-System Microcosm, welches eine Art Verlinkung-Overlay für das Microsoft Windows anstrebte, oder das an der TUGraz entwickelte Online-Informationssystem Hyper-G, das in Sektion 2.3 schon umschrieben wurde.

Beide Projekte kreieren dynamische Links mit ähnlichen technologischen Lösungen:

Hyper-G Hyper-G stand anfangs in Konkurrenz zum Web, welches bis heute das Problem der toten Links nicht gelöst hat. Hyper-G aber präsentierte eine Lösung in Form eines Verlinkungsalgorithmus. Die Links werden dabei nicht in das Dokument eingegliedert, sondern in einer eigenen Datenbank gespeichert. Mithilfe dieses Systems ist es möglich

die Konsistenz der Vernetzung zu verwalten und Read-Only-Medien wie z.B. eine CD zu verlinken[5].

Außerdem kann man mithilfe der Links auf einzelne Dokumente oder auf eine Dokumentensammlung[1] zeigen.

Im Gegensatz zum unidirektionalen Ansatz des Webs ermöglichen bidirektionale Verlinkungen die Rückverfolgung der Vernetzung[5].

Microcosm Microcosm ist ein quelloffenes Hypertext-System, welches Anfang der 90er Jahre für den IBM PC entwickelt wurde. Es ist im Gegensatz zu Hyper-G nicht als Online-Applikation gedacht sondern für den Einzelgebrauch. Mithilfe von Microcosm soll man prinzipiell jedwedes Dokument-Format miteinander verknüpfen können. Erweiterbarkeit soll durch die offenen Sourcen gewährleistet sein.

Die Links werden extern gespeichert. Ziel ist es über diese externe Link-Datenbank eine Wissensbasis aufzubauen, sodass neue Dokumente, die in das System hinzugefügt werden, mithilfe der gespeicherten Information vernetzt werden können.

Beide Systeme benützen eine externe Linkspeicherung um die Dynamik der Vernetzung zu gewährleisten. Während es bei Hyper-G mehr um die Konsistenz der Vernetzung geht, versucht Microcosm Wissensbasen aufzubauen.

Heute hat das Web diese Hypertext-Systeme längst verdrängt. Die Idee aber von externer Link-Speicherung bzw. dynamischen Links ist geblieben. Obwohl es keinen allgemeinen Ansatz hin zur dynamischen Vernetzung im Web gibt, lassen sich die Web-Applikation entsprechend erweitern, was in spezifischen Situationen Vorteile bringen könnte[61]:

- Abnahme des Wartungsaufwands. Der Link wird nur in der Datenbank geändert und damit auch in jedem Dokument, in dem er verwendet wird. Dadurch lassen sich auch Verweise ins Leere vermeiden.
- Eigene Links. Jeder User bekommt eine eigene Link-Datenbank, welche er selbst erstellen und verwalten kann. Er ist nicht mehr angewiesen auf die statisch-gesetzten Verweise eines Editors. Es wäre sicher möglich diese eigene Datenbank mit anderen Benützern zu teilen.
- Entkoppelung von Information und Verlinkung. Link-Datenbanken und Dokumente könnten ausgetauscht werden.

Das Projekt COHSE[2] ist zum Beispiel ein aktuellerer Versuch, Webdokumente dynamischer zu vernetzen. COHSE ist im Grunde ein dynamisches Verlinkungs-Service, welches Semantic Web- Technologien inkludiert. Es besteht aus dem sogenannten "Knowledge-Service", das OWL-Ontologien einbezieht und Verknüpfungen zu Konzepten herstellt, und aus

[1] Den sogenannten Collections in Hyper-G
[2] Conceptual Open Hypermedia Service

einem "Ressource-Service", der eine URI[3] mit Konzepten verbindet[122].

3.2 Statische Vernetzung

Bei der statischen Vernetzung geht es vorwiegend um Links, die in das Dokument gespeichert und damit auch ein Teil davon werden. Statische Verlinkung bedeutet, dass nachdem der User das Dokument verlässt im Gegensatz zu dem dynamischen Ansatz die Verweise nicht gelöscht werden. Sie können auch precomputed Links genannt werden[10].

Hier findet sich eine Auflistung, die Vorteile und Nachteile der sogenannten precomputed Links beschreibt[10]:

Vorteile

+ Systemschonend. Im Gegensatz zum dynamischen Ansatz ist das Erstellen der Links nicht bei jedem Dokumentabruf durchzuführen.
+ Dokumente werden schneller abgerufen, denn es ist nur eine Berechnung beim Initialisieren aller Links nötig.

Nachteile

- Möglichkeit von Lücken im Kontext. Im Falle der Änderung des Dokuments oder des Dokumenten-Bestands.
- Fehlende Link-Integrität. Das Löschen eines Dokuments im System führt möglichweise zu einem toten Link.
- Redundanz. Sollte der Verlinkungsalgorithmus auf die ganze Dokumentenbasis angewendet werden, werden möglicherweise Verweise für niemals abgerufene Dokumente berechnet.

3.2.1 Projekte

Es werden die Projekte in drei verschiedene Kategorien eingeteilt. Diese orientieren sich an den grundsätzlichen Kriterien, mit denen die Verlinkungsalgorithmen die Dokumente vernetzen[108].

Struktur

Der Fokus wird hier auf die logische Struktur der Dokumente zur Erstellung der Verweise gelegt. In diesem Ansatz kommt es zur Verknüpfung von erkennbaren Struktur-Elementen. Diese Verlinkung vermag zwischen den Dokumenten oder zwischen Elemente, die im gleichen Dokument zu finden sind, kreiert werden[108].

Ein Beispiel dazu findet sich im Projekt von Raymond und Tompa[94], welches eine Art Hypertext-Userinterface für das Oxford English Dictionary erstellten. Die Querverweise waren ihrer Meinung nach die geeignetsten

[3] Universal Ressource Identifier

Abbildung 3.1: Unterschied zwischen einem Information Retrieval System und Automatischen Verlinkungstools basierend auf IR[108]

Kandidaten unter den strukturellen Elementen, um Verlinkungen zu setzen. Sie hatten jedoch Probleme automatisch ein geeignetes Ziel für die Links zu bestimmen.

Glushko zeigte 1989 wie eine große Ingenieurs-Enzyklopädie in Hypertext umzuwandeln ist. Über die Struktur der Dokumente erstellte er ein Inhaltsverzeichnis, Indizes, sowie Links, die Teile wie Fussnoten und Anmerkungen im gleichen Dokument vernetzten[55].

Auch die Umwandlung des Unix-Manuals in Hypertext, durchgeführt von Franke und Wahl, kann als weitere strukturbezogene Verlinkung gesehen werden[46].

Eine ganz andere Benutzergruppe adressierte das xlinkit Projekt von Nentwich[86]. Während die Vorhergenannten eher den Hypertext-Betrachter im Fokus hatten, adressierte xlinkit eher Betreiber und Editoren von Hypertext-Systemen. Aufgaben von dem Projekt bestanden in einem verteilten Dokumentenbestand die Elemente auf Konsistenz zu vergleichen und zu verlinken. Der Algorithmus, der da hinter steht, basiert auf einem Regelwerk um die Konsistenz-Bedingungen zwischen den Dokumenten zu definieren.

Statistisch

Projekte dieser Art nützen zur Berechnung der Verlinkungen mathematische Funktionen. Genauer gesagt verwenden diese Projekte die Methoden des Information Retrieval, welche genauer im Kapitel 5 beschrieben werden, um die Dokumenten miteinander zu verknüpfen. Dabei sind diese Verlinkungstools nicht mit Information Retrieval Systemen zu verwechseln. Wie Abbildung 3.1 zeigt, sind die beiden unterschiedlich aufgebaut[108].

Eines der frühen Projekte war der Hypertext Apprentice von Bernstein

im Jahr 1992, welcher anhand eines simplen Ähnlichkeitsmaß die Dokumente miteinander verglich[21].

Später implementierte James Allan einen ausgeklügelteren Vernetzungsalgorithmus basierend auf dem Techniken des Vector Space Model[4]. Er erarbeitete auch eine Technik um die erstellten Links nach einer bestimmten Taxonomie zu klassifizieren.

Es basierte nicht nur Allans Projekt auf dem Vektor Space Model, sondern viele andere Projekte so wie mein Diplom-Projekt oder das SMART Retrieval System von Salton[97] etc.

Automatische Velinkungstools basierend auf Information Retrieval sind im Grunde sehr forschungsbezogen. Existieren neue Forschungsergebnisse betreffend diesem Fachgebiet, ist die Wahrscheinlichkeit sehr groß, dass eine Unmenge an Verlinkungstools, die neuen Erkenntnisse als Basis besitzend, erstellt werden[108].

Eine relativ neue Technologie im Information Retrieval ist das Latent Semantic Indexing von Bellcore. Kurz genannt LSI, involviert es in simplen Term-Frequenz-Systemen konzeptuelle Ideen. Term-Dokument-Matrizen werden praktisch durch dieses Verfahren reduziert und in der Folge ein semantischer Raum geschaffen, worin konzeptuell verwandte Terme nah zueinander platziert werden[34].

Blustein zum Beispiel verwendete LSI zur Verlinkung eines wissenschaftlichen Dokumentenbestands[22].

Semantisch

Semantische Verlinkungstools orientieren sich nicht wie die Statistischen an der Verteilung von Termen sondern legen Wert auf die Bedeutung der Terme. Wörter werden in dieser Richtung im Kontext ihres konzeptuellen Hintergrunds gesehen.

Um die Terme, bzw. in Folge die Dokumente richtig zu kategorisieren, wird ein Regelwerk benötigt, wobei die Erstellung eines solchen einen hohen Aufwand erfordert.

Im Grunde wird die semantische Verlinkung in zwei Phasen geteilt[108]:

Hypothese In dem vorhergenannten Regelwerk existieren mehrere Patternmengen. Patterns sind eigentlich ein oder mehrere Wörter. Je mehr im Artikel Überschneidungen mit der Patternmenge zu finden sind, desto eher fällt diese in die engere Wahl. Es können mehrere Patternmengen einem Dokument zugeordnet werden.

Bestätigung Bei dieser Phase werden die weniger zugehörigen Patternmengen über bestimmte den Mengen inkludierte Regeln aussortiert.

Als Beispiel: Der Titel *Atombosse rüsten zum Kampf gegen Merkel* in einem Nachrichtenartikel würde zu der Pattern-Menge *Krieg* passen.

[4]Siehe mehr dazu Kapitel 5.

Jetzt existiert aber kein Hinweis über verschiedene Länder oder Waffen in dem Artikel, deshalb würde er aussortiert werden.

IMAD[5] von Hayes und Pepper zum Beispiel benützte die beschriebene Technik um Verlinkungen zwischen den Dokumenten herzustellen[62].

Aktuelle semantische Hypertext-Tools verwenden NLP[6]-Techniken[108]. Einige Projekte in der Richtung wären z.B. Green[56], welcher lexikalischen Ketten bei der Vernetzung verwendet oder Basili[14], der wissensbasierte Informationsextraktion zur Verlinkung einsetzt.

Hybride

Nicht zu vernachlässigen ist die Kombination zwischen den vorher beschriebenen Techniken.

Ein sehr bekanntes Projekt in dieser Richtung ist Citeseer, eine Suchmaschine für wissenschaftliche Arbeiten. Sie soll dabei helfen die Suche nach web-basierte Publikationen zu automatisieren und dadurch erheblich zu erleichtern[23].

Citeseer kombiniert mehrere Methoden. Es verwendet maschinelle Lernheuristiken, die konventionelle Suchmaschinen nach Publikationen durchsuchen. Außerdem existiert eine Übersetzungsfunktion, welche Postscript und PDF Dateien ins ASCII-Format überträgt mit der Aufgabe die wissenschaftliche Arbeiten zu identifizieren. Noch dazu sucht Citeseer im Dokument nach Autor, Titel, Referenzen uvm., um damit mehr Informationen über das Dokument in das Webportal übertragen zu können.

Die Weiterentwicklung von Citeseer wird CiteseerX genannt und ist zu finden unter http://citeseerx.ist.psu.edu/ .

3.3 Automatische Verlinkung im Web

Es folgt in diesem Abschnitt eine kleine Aufstellung von Projekten, welche sich mit automatischer Verlinkung speziell im Web bzw. in Web-Enzyklopädien beschäftigen. Es existieren verständlicherweise bei weitem mehr Verlinkungstools dieser Art.

NNexus Dieses Verlinkungstool wird in der kollaborativen Online-Enzyklopädie PlanetMath[7] verwendet. Es verlinkt die Beiträge anhand ihrer Metadata-Einträge in dem Netzwerk, wobei dahinter semantische Konzepte stehen. Es soll zur Unterstützung der Autoren verwendet werden[50].

Linkator Ein Verlinkungstool welches Semantic Web Technologien und Informationsextraktion kombiniert um Verknüpfungen herzustellen. Außerdem benützt es http://linkeddata.org/ als externe Wissensquelle[9].

[5] integrated maintenance advisor
[6] Neuro-linguistic-programming
[7] http://planetmath.org/

Wikifiy Identifiziert die wichtigen Konzepte eines Dokuments und verlinkt dieses mit dem Dokumentenbestand von Wikipedia[29]. Wobei das nicht das einzige Verlinkungstool ist, welches Wikipedia als Wissensbasis benützt. Siehe zum Beispiel Knoth [72] oder Gardner [51].

Kapitel 4

Navigation in Hypertext-Netzwerken

Die Navigation in Hypertext-Netzwerken spielt eine wichtige Rolle in der Interaktion mit diesen Systemen. Im Gegensatz zur Benützung der Suchfunktion bekommt der User bei der Navigation ein besseres Verständnis des Kontexts und es ist ihm möglich eine Beziehung zwischen den Dokumenten herzustellen[117].

Der Dokumentenbestand eines Informations-Systems, in welchem der Benutzer navigiert, kann aus einem anderen Blickwinkel heraus betrachtet werden als z.B. der des Users. Diese andere Sicht der Dinge stellt die vernetzten Dokumente als sogenannte Graphen dar. Ein solcher Graph zerlegt sich in Knoten, welche als die Dokumente verstanden werden, und in Kanten, die die Verknüpfungen des Netzwerks darstellen.

Dieses Kapitel beschäftigt sich mit der Vernetzung aus dieser graphentheoretischen Sicht. Es werden sowohl Gegebenheiten als auch Gesetzmäßigkeiten, welche mit solchen Netzwerken zusammenhängen, betrachtet.

Es wird aber auch die Navigation aus dieser Sicht analysiert. Ein User, der durch ein derartiges Informationsystem navigiert, wird dabei mit einem dezentralen Suchalgorithmus gleichgesetzt. Das eröffnet einem neue Möglichkeiten zur Evaluierung dieser Systeme.

4.1 Das Kleine-Welt-Paradigma

4.1.1 Geschichte

Der amerikanische Psychologe Stanley Milgram führte im Jahr 1967 ein Experiment durch, welches sich mit weitreichenden sozialen Verknüpfungen in der Bevölkerung beschäftigte. Es sollten zufällige Testpersonen aus dem Raum Wichita und Omaha ein Paket zu einem festgelegten Empfänger senden, jedoch durfte der Person das Paket nicht direkt zugeschickt werden,

sofern der Empfänger dem Sender nicht persönlich bekannt war. Es sollte nur an jene Personen weitergesandt werden, welche auch mit dem Sender per Vornamen kommunizierten. Der temporäre Besitzer des Pakets versandte also dieses weiter an einem Bekannten, mit der Überlegung, dass dieser die Zielperson kennen könnte. Die Teilnehmer waren auch angehalten zur Nachvollziehbarkeit der Kette eine Postkarte mit persönlichen Informationen an die Wissenschafter zu schicken, wann immer sie das Paket weitersendeten[82].

Von den 60 Paketen erreichten 3 ihr Ziel, wobei die Verkettung eine Durchschnittslänge von 5.5 Teilnehmer bzw. Knoten aufwies. Daraus folgerte man, dass jeder US-Bürger über sechs Personen mit jedem anderen Mitbürger bekannt war, worauf sich auch die Bezeichnung "Small World Phenomenon[1]" bezieht[81]. Es wird aber auch als die "Six-degrees of separation" beschrieben.

Ein zweites Experiment, die diese Small-World-Methode benutzte, wurde von Milgram zwei Jahre später durchgeführt. 296 Personen aus Nebraska und Boston sollten über die vorher besprochene Weise ihr Paket an eine Zielperson in Massachusetts senden. Es erreichten 65 ihr Ziel und der Durchschnitt der Zwischenstationen betrug im Durchschnitt 5.2.

Die Experimente werfen die Frage auf, inwiefern ein Netzwerk aufgebaut sein muss, wo es zu ähnliche Resultaten wie im Small-World-Experiment kommt.

4.1.2 Small-World-Netzwerk

Die Gegebenheiten, welche sich auf unser soziales Netz abbilden lassen, können auch auf andere Beispiele erweitert werden. In der Tat existieren einige natürlich entstandene Netzwerke, worin alle Knoten durch kurze Pfadlänge verbunden sind[110]:

- Das neuronale Netz des Wurms Caenorhabditis elegans.
- Das Stromnetz der USA.
- Ein Graph, welcher die Zusammenarbeit von Filmschauspielern darstellt. (Siehe auch dazu die Bacon-Zahl[2], die angibt um wieviele Ecken Schauspieler mit Kevin Bacon zusammengearbeitet haben. Zum Beispiel: Die Bacon-Zahl 1 bedeutet, er hat direkt mit ihm zusammengearbeitet und die Bacon-Zahl 2 heisst soviel wie, dass der Schauspieler mit einem anderen zusammengearbeitet hat, welcher schon mit Kevin Bacon in einem anderen Film zusammenspielte etc.
- Teile des WorldWideWeb. Es wurden dazu u.a. .edu-Seiten analysiert und auf ihre Eigenschaften untersucht, wobei die Ergebnisse auf ein Small-World-Netzwerk hinwiesen[2].

[1] übersetzt: Kleine-Welt-Phänomen
[2] http://oracleofbacon.org/

Es existieren zwei bekannte Modelle, die das Small-World-Netzwerk abbilden. Das erste das beschrieben wird ist das Strogatz-Watts-Netzwerkmodell und das zweite das Barabási-Albert-Modell.

4.1.3 Strogatz-Watts-Netzwerk

Duncan Watts und Steven Strogatz entwarfen ein Netzwerk-Modell, welches wie das Small-World-Netzwerk aufgebaut werden sollte.

Zum Aufbau eines Strogatz-Watts-Modell nimmt man anfangs ein Gitter, das ringförmig aufgebaut ist (siehe dazu das erstes Bild links bei Abbildung 4.1). Die einzelnen Punkte bekommen einen weiteren Link, der über die Wahrscheinlichkeit p zufällig ausgesucht wird, wobei p zwischen 0 und 1 zu suchen ist. Wenn $p = 1$ bedeutet das einen Zufallsgraphen (siehe dazu die Abbildung 4.1 ganze rechts).

$0 < p < 1$ bedeutet auch, dass p selbst nur eingegrenzt ist und der optimale Wert noch erforscht werden muss. Zusammenhänge lassen sich jedoch durch Tests deuten.

Es werden solch einem Graphen zwei Eigenschaften zugewiesen[110]:

Charakterische Pfadlänge ($L(p)$) Es wird der kürzeste Pfad zwischen zwei Knoten genommen. Dieser wird zwischen allen Knoten berechnet und daraus ein Durchschnittwert gezogen.

Clustering Koeffizient ($C(p)$) Im Paper von Steven Strogatz und Duncan Watts wird bei diese Kennzahl auch von dem Maßstab der Cliquenhaftigkeit gesprochen. Die Erklärung ist recht simpel. Aus der Sicht eines Social Networks, worin ein Knoten einen Mensch und eine Kante zwischen diesen Freundschaft darstellen soll, ist das folgendes: Wieviel Freunde, habe ich, die die gleichen Freunde haben. Also in dem Sinne eine Art geschlossener Freundeskreis, deshalb auch der Name Cliquenhaftigkeit. Der Clustering Koeffizient gibt den Durchschnitt solcher Freundeskreise auf jeden Knoten an.

In Abbildung 4.2 finden sich Testergebnisse, welches die Entwicklung von $C(p)$ und $L(p)$ bei steigendem Zufall der Vernetzung darstellt. Wie man sieht hält sich der Clustering Koeffizient relativ hoch und sinkt drastisch bei hohem p. Während jedoch die charakterische Pfadlänge $L(p)$ schnell sinkt (Das passiert schon im Tausendstel von p). Sprich das Small-World-Netzwerk, welches zwischen dem Zufälligkeitwert p von 0 und 1 liegt, ist vom Charakteristikum ein Netzwerk mit kurzen Wegen.

Steven Strogatz und Duncan Watts zeigten über ihr Netzwerk-Modell einen realistischen Ansatz inwiefern Small-World-Netzwerke aufgebaut werden könnten. Es existiert aber noch ein anderer Ansatz.

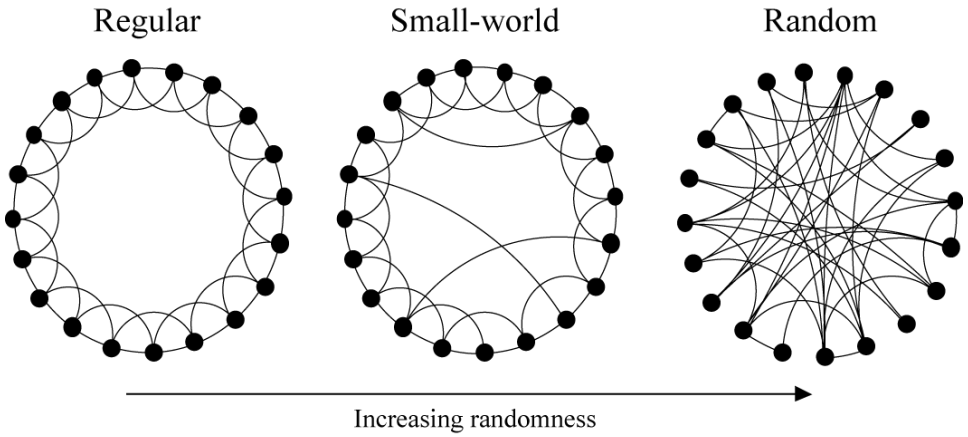

Abbildung 4.1: Übergang von einem Gitter mit Ringgeometrie zu einem "Small World"-Netzwerk bis zum zufällig aufgebauten Netzwerk[110]

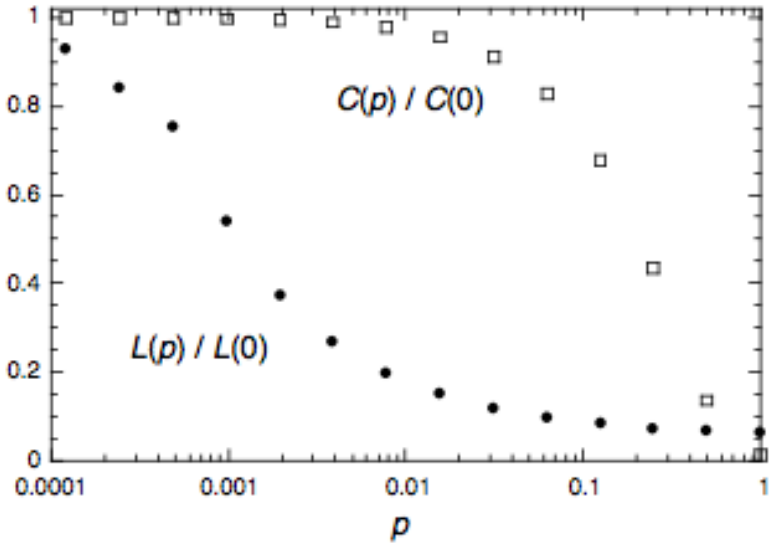

Abbildung 4.2: Das Verhältnis von $L(p)$ (durchschnittliche Pfadlänge zwischen allen Knoten) und $C(p)$ (dem Clustering Koeffizient) zu p (dem steigenden Zufall bei der Vernetzung)[110]

4.1.4 Barabási-Albert-Netzwerk

Ihr Netzwerk-Modell basiert auf der Einsicht, dass komplexe Netzwerke einer skalenfreien Verteilung, die auf Potenzgesetze beruht, folgt.

Sogenannte skalenfreie Netzwerke besitzen folgende Eigenschaften:

- Die Verteilung der Knoten pro Kante folgt keiner Gauß-Verteilung.

Netzwerkname	γ
WWW	2.1
Schauspieler-Kollaborations-Graph	2.3
Elektrisches Netz der USA	4

Tabelle 4.1: Hier werden bekannte komplexe skalenfreie Netzwerke mit dem von der Barabási-Albert Formel (Siehe Formel4.1.4) stammende γ aufgelistet [13]

(Die Knoten haben also nicht gleichmäßig viele Kanten). Die Verteilung der Kanten der skalenfreie Netzwerke wird Pareto-Verteilung genannt. Diese schließt ein, dass es Knoten gibt die Abweichungen zu dem Durchschnitt haben, siehe dazu nächsten Punkt über die Hubs[68].

- Einige Knoten haben überdurchschnittlich viele Verbindungen (Hubs). Das basiert auf der Annahme dass die Verteilung der Kanten pro Knoten Potenzgesetzen folgt.
- Hohe Ausfallssicherheit bei zufälliger Entfernung eines Knotens. (Jedoch Anfälligkeit für gezieltes Entfernen, besonders wenn die Hubs aus dem Netzwerk genommen werden.)

Den Grundstein für das Modell von Barabási und Albert entstammt dem Vorhandensein von Daten, die sich mit komplexen Netzwerke beschäftigen. Während z.B. Erdös-Renyi in den sechziger Jahren davon ausging, dass es sich bei solchen Netzwerken vorwiegend um Zufallsgraphen handelte, also um zufällig verbundene Knoten, konnten Barabási und Albert wie erwähnt auf schlüssigere Daten zurückgreifen und daraus ihr Modell entwerfen[13].

So wurden große Netzwerke, wie das WorldWideWeb, ein Kollaborations-Graph von Filmschauspielern und das elektrische Netz der USA untersucht und Statistiken u.a. wie die Verteilung der Knoten (siehe dazu Abbildung 4.3 berechnet, wobei sich herausstellte, dass diese nach dem Potenzgesetz aufgeteilt sind[13].

Folgende Wahrscheinlichkeit $P(k)$ errechnet nach Barabási und Albert für den Knoten k, wieviele Kanten er besitzen soll[13]: $P(k) \sim k^{-\gamma}$

Nach den Berechnungen von Barabási und Albert liegt γ in der Regel zwischen 2 und 4[68], für genauere Zahlen zu Netzwerken siehe Tabelle 4.1.

Aufbau des Barabási-Albert-Netzwerks

Die Idee zum Aufbau des Netzwerk-Modell von Barabási-Albert unterscheidet sich in dem Punkt von dem Modell von Strogatz-Watts, da es mehr auf reale Daten aufbaut and dadurch als wirklichkeitsnäher bezeichnet werden kann.

Die Idee von Strogatz-Watts war, dass es zu Anfang sich um ein reguläres

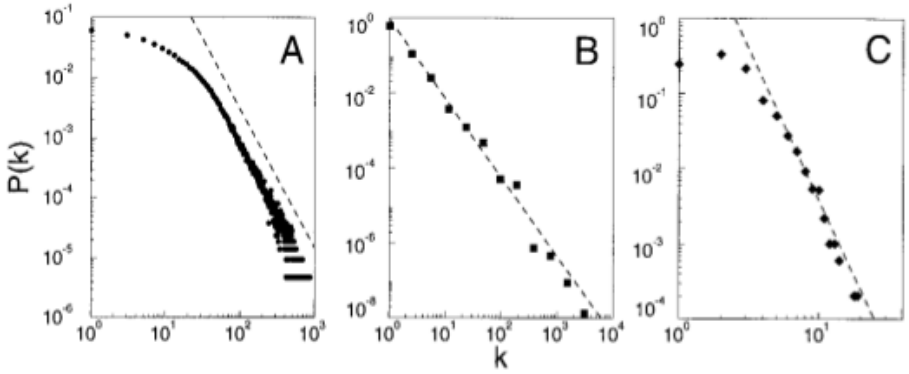

Abbildung 4.3: Die Wahrscheinlichkeitsverteilung $P(k)$ der Kanten pro Knoten k von folgenden Netzwerken: **(A)**dem Filmschauspieler-Kollaborationsgraphen (Durschnittsanzahl von k pro Knoten: 28,78), **(B)** dem WorldWideWeb (Durschnittanzahl von k pro Knoten: 5,46), **(C)** dem elektischen Netz der USA (Durschnittsanzahl von k pro Knoten: 2,67)[13].

Gitter handelt, und dann jeder Knoten mit einer bestimmten Wahrscheinlichkeit mit einem anderen (fernen) Knoten verbunden wird. Barabási und Albert nahmen den Fall an, dass immer neue Knoten zum Netzwerk hinzugefügt werden. Ähnlich dem Vorgang wie wenn eine neue Seite im Web hinzukommt oder ein unbekannter junger Schauspieler in einem neuen Film mitspielt.

Nimmt man das Beispiel des neuen unbekannten Schauspielers her, kann man davon ausgehen, dass er nicht in einer Hauptrolle spielen wird, sondern ein populärer erfahrener Schauspieler diese Rolle übernimmt. Der erfahrene Schauspieler hat schon aufgrund seiner Karriere mit mehr Schauspielern zusammengearbeitet, also hat er aus Kollaborations-Graphensicht schon viel mehr Kanten als der junge unbekannte Schauspieler. Das nennt Barabási und Albert "preferential attachement", was soviel bedeutet wie, wenn ein neuer Knoten zu dem Netzwerk hinzukommt, besteht eine große Wahrscheinlichkeit, dass dieser sich mit den sogenannten Hubs verbindet[13].

Unterschiede des Barabási-Albert-Modell zu den Anforderungen des standardmäßigen Small-World-Netzwerks existieren auch insofern, als dass bei ersterem Clustering bzw. die Cliquenhaftigkeit keine Berücksichtigung findet. Das bedeutet, dass ein skalenfreies Netzwerk ein Small-World-Netzwerk sein kann, aber nicht in jedem Fall, wobei das beim Strogats-Watts-Modell immer zutrifft.

4.1.5 Dezentrale Suchalgorithmen im Small-World-Netzwerk

Es wurde in dem Kapitel schon viel über die Architektur eines sogenannten Small-World-Netzwerks und um seine speziellen Eigenschaften berich-

tet. Milgrams Small-World-Phänomen hat nicht nur gezeigt, dass Netzwerke mit den sogenannten "six degrees of separation" existieren, sondern zeigte auch, dass die Leute ihr lokales Wissen nutzen konnten um die Pfade zur festgelegten Person zu konstruieren.

In diesem Abschnitt geht es vorwiegend um die Frage des optimalen Suchablauf durch diese Art von Netzwerk. Es sollen die Eigenschaften eines dezentralen Suchalgorithmus beschrieben werden, der aufgrund seines lokalen Wissens Pfade in dem Netzwerk herstellt. Dieser Ablauf lässt sich mit dem Browsen des Users durch das Web vergleichen, was im Grunde auch für mein Projekt interessant ist.

Der Suchalgorithmus wird deshalb dezentral genannt, da es keine Komponente gibt, die jeden Suchschritt steuert, sondern weil die Knoten aufgrund ihres "Wissen" selbst entscheiden, wohin sie im Falle von Milgram das Paket weitersenden.

Die Grundeigenschaften eines solchen Knoten bzw. seinem Netzwerk benennt Kleinberg wie folgt[70]:

- Der Knoten existiert auf einem Gitternetz.
- Der Knoten hat Kanten zu Nachbarn, die neben ihm im Gitternetz liegen, und Kanten zu sogenannten Fernkontakten.
- Der Knoten weiß wo der Fernkontakt im Gitter liegt und weiß auch über seine Position im Gitter Bescheid.
- Der Knoten weiß jedoch nicht welche Fernkontakte seine Bekannten im Gitter haben.

Das Ziel des Suchalgorithmus von Kleinberg ist, dass dieser die Entfernung von dem Anfangsknoten v und dem Endknoten w in polylogarithmischer Zeit in Relation zu den n Knoten im Gitter schafft.

Der Test seines Algorithmus am Strogatz-Watts-Modell erreichte das vordefinierte Ziel laut Kleinberg nicht, obwohl es als ein Modell des Small-World-Netzwerks dies hätte erreichen sollen. Der Grund liegt in der zufälligen Auswahl der Fernkontakte im Strogatz-Watts-Modell. Daher wurde ein Clusteringkoeffizent α von Jon Kleinberg vorgeschlagen, welcher diese Zufälligkeit reduziert.

Nimmt man ein zu niedriges α so werden die Fernkontakte zu zufällig für eine dezentralen Suchalgorithmus gewählt, sprich also $\alpha = 0$ entspricht dem Strogatz-Watts-Modell. Nimmt man aber ein zu hohes α so ist die Auswahl der Fernkontakte nicht zufällig genug, um damit in der benötigten Zeit den Endknoten zu erreichen.

Der ideale Exponent α berechnete Kleinberg dann als $\alpha = 2$. Hat der Exponent diesen Wert, erreicht der Algorithmus die gesetzte Anforderung, der polylogarithmischen Zeit. Jeder andere Exponent erreichte eine viel höhere Zeit [69].

Anwendungszwecke

Natürlich entsteht die Frage für welchen Anwendungszweck dezentralisierte Suchalgorithmen verwendet werden können:

Peer-to-Peer-Protokolle Die Filesharing-Netzwerke zwischen Benützern entsprechen einem Small-World-Netzwerk. Im Grunde wurden, bevor es Probleme bezüglich der Legalität gab, die ganzen Anfragen zentral von den File-Sharing-Servern gespeichert. Dadurch ließen sich auch die Anfragenden zurückverfolgen bzw. ihre IP. Das ist nur der eine Punkt, der für die Anwendung von dezentralisierten Suchalgorithmen spricht, auch würde das System dadurch stabiler und mächtiger werden, je robuster es aufgebaut ist. Man denke nur, wenn der zentrale Server ausfällt. Es existieren dazu einige Arbeiten, die sich mit diesem Thema beschäftigen siehe dazu [76][64][67].

Fokusierter Web-Crawler Hier geht es nicht um den normalen Web-Crawler der alle möglichen Daten speichert, also alle Kanten eines Knoten verfolgt, sondern um einen auf spezielle Inhalte fokusierten Crawler. Dieser folgt den Kanten von Page zu Page und versucht über spezielle Regeln zu entscheiden, welche Links er verfolgen soll. Die Frage ist, ob es möglich sei, solche Crawler zu programmieren, welche fähig sind, eine nicht indizierte aber für den User relevante Page aufzuspüren. Darüber existieren Testberichte, die durchwegs positiv ausfielen[80].

Daten von sozialen Netzen Da soziale Netzwerke prinzipiell Small-World-Netzwerke sind, sollten sich diese zur Anwendung der dezentralisierten Algorithmen eignen. Ein Beispiel dafür findet sich z.B. im geographischen Routing in einem Social-Netzwerk von Liben-Nowell[74]. Dabei wurden Daten vom Blogging-Portal http://www.livejournal.com, welches laut Selbstangaben über 30 Millionen Blogs und Communities beherbergt[77], genommen und ein geographisch-fokusierter dezentraler Suchalgorithmus darauf angewendet. Der Algorithmus ging von Person zu Person und suchte in der Kontaktliste den geographisch am nächsten liegenden Kontakt zur Zielperson.

Ein weiterer Versuch Suchalgorithmen an Daten von sozialen Netzen zu testen, wurde von Adamic et al. durchgeführt[3]. Betrachtet wurde der E-Mail-Verkehr von HP Labs[3]. Ein Netzwerk-Modell von dem Email-Verkehr findet sich unter Abbildung 4.4. Die Suchalgorithmen-Tests verliefen bei dem E-Mail-Netzwerk positiv.

[3] http://www.hpl.hp.com/

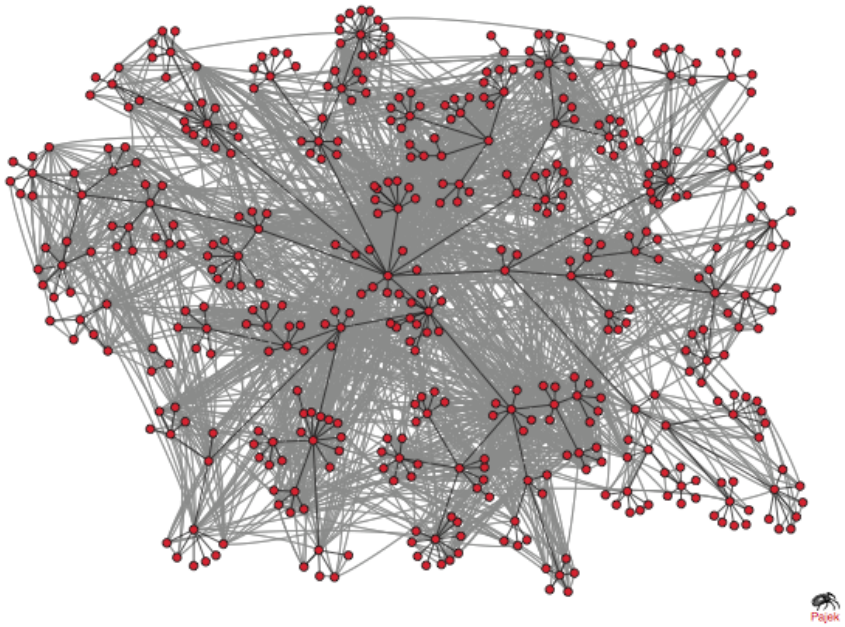

Abbildung 4.4: Darstellung des Emailverkehrs von HP Labs unter Berücksichtigung der hierarchischen Einheiten. Die grauen Striche sind der Email-Verkehr, die schwarzen Strich zeigen die hierachische Zugehörigkeit[3].

4.2 Analyse

Der graphentheoretische Ansatz bei Informationssystemen eröffnet neue Arten der Analyse von derartigen Systemen.

Dieser Ansatz wäre womöglich geeignet zur Evaluierung der Vernetztheit der Informationsbestände. Es könnte über den Verlinkungsgrad des Hypertext-Netzwerks Aussagen über die Befähigung zur Navigation in dem System gemacht werden. Die Annahme liegt nicht fern, dass je niedriger der Diameter eines Netzwerks ist, desto höher die Wahrscheinlichkeit, dass der Benützer sein Informationsbedürfnis befriedigen kann.

Es wäre auch möglich über den dezentralen Suchalgorithmus einen Benutzer zu simulieren und dadurch weitere Erkenntnisse der Navigations-Tauglichkeit des Netzwerk zu gewinnen.

Interessant wäre dabei auch die Simulation der Auswirkungen von Navigationstools auf das Netzwerk, bevor man diese überhaupt in das System integriert. Es existiert bereits ein Projekt von Helic et al. welches in diese Richtung forscht[63].

Im Großen und Ganzen könnten über die graphentheoretische Richtung der Hypertext-Forschung analytische Tools entstehen, die einen neuwertigen Einblick und wertvolle Erkenntnisse für die Weiterentwicklung dieser Infor-

mationssysteme liefern würden.

Kapitel 5

Information Retrieval

5.1 Einleitung

Das Wort Information Retrieval lässt sich schwer ins Deutsche übersetzen. Das Online-Wörterbuch http://www.dict.org übersetzt Information Retrieval als Informationsabruf, Informationswiedergewinnung oder Wiederauffinden von Information. Ob diese Beschreibungen umfassend zutreffen, ist zweifelhaft, betrachtet man die später folgenden Definitionen.

Nach Salton sollte sich Information Retrieval mit Strukturierung, Analyse, Organisation, Speicherung, Suche und Abbruf von Information beschäftigen [96].

Es stellt sich die Frage was Information eigentlich bedeutet. Kann man Wissen mit Information gleichsetzen? Wie hängt das mit den reinen Da-

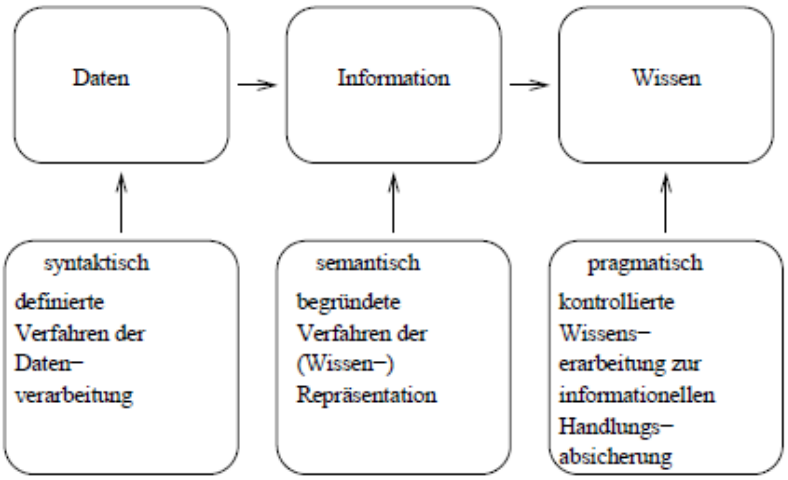

Abbildung 5.1: Unterschiede Daten, Information, Wissen[48]

ten zusammen. Eine Definition (wie zu sehen unter Abbildung 5.1) versucht zuerst die einzelnen Begriffe abzugrenzen.

In der heutigen Zeit wird Information Retrieval oft gleichzeitig mit dem World Wide Web verknüpft, jedoch diese Relation ist nur zum Teil richtig[78].

Information Retrieval als Forschungfeld ist bevor dem Aufkommen des Webs vorwiegend im Bibliothekswesen oder im Patentamt sowie in verschiedenen wissenschaftlichen Bereichen ein Begriff gewesen. Die Grund dafür kann sich darauf beziehen, dass in diesen Arbeitsfeldern es zu Tätigkeiten zählt über einen großen Dokumentenbestand zu walten.

Das Web hat also Information Retrieval nicht erfunden. Die Beschaffenheit des Webs selbst förderte Innovationen im IR-Bereich, wobei die Verfügbarkeit und Quantität der ins Web freigegebenen Dokumente wahrscheinlich eine Rolle spielt[78].

Im Web wurde die Basis für das Anwachsen von öffentlich zugänglichen Dokumenten geschaffen. Jeder Benutzer vermochte auf einmal sein Anliegen darin zu verbreiten. Aus dieser Entwicklung folgte so gleich der Ruf nach einer effizenten Suche, die auf Anfrage des Users für ihn relevante Dokumente zurückgibt. Dies ist ein Kerngebiet des Information Retrieval.

Die Gesellschaft für Information Retrieval sieht den Schwerpunkt dieses Forschungsfeldes eher im Wissenstransfer vom Informationssystem zum Anfragenden, wobei hier folgende Spezifikationen gelten sollten[47]:

- **Vagheit** Das Informationsbedürfnis sollte nicht vollkommen ausformuliert sein, sondern eine Vagheit sollte existieren
- **Unsicherheit** Das IR-System sollte den Inhalt der Dokumente nicht in ihrer Vollständigkeit kennen.

Ein Information Retrieval System oder ein auf Deutsch Informationsabruf-System soll nicht das Wissen des Users über die Angelegenheit verändern, nach der er sucht. Es informiert ihn jedoch über die Existenz von einigen Dokumente, die sich auf seine Anfrage beziehen könnten. So könnte man Aktivitäten im Bereich von Information Retrieval zusammenfassen.

5.1.1 Daten Retrieval vs Information Retrieval

Man darf Information Retrieval nicht mit Datenretrieval verwechseln. Es gibt einige markante Unterschiede, die berücksichtigt werden müssen.

Erstens geht Daten-Retrieval in die Richtung Datenbankabfragen. Zumal in dieser Richtung klar durchformulierte Anfragen gestellt werden, die durch die Exaktheit der Anfrage auch genau bestimmte Menge an Suchergebnisse zurückliefern. Jedes falsche Ergebnis der Suche wird als absoluter Fehler des Systems gesehen.

Im Information Retrieval-Bereich sind teilweise fehlerhafte Suchergebnisse nichts Ungewöhnliches. Es wird eher darauf geachtet, ob die Information für das Informationsbedürfnis des Users befriedigend war oder nicht.

Vergleichsparameter	Data Retrieval	Information Retrieval
Art des Treffers	Exakt z.B. eine Abfrage bei einer Datenbank nach Einträge mit bestimmten Datum. Wenn die Suchergebnisse Dokumente mit anderem Datum zurückliefert, wird das als vollkommenes Versagen des DR-Systems angesehen.	Teilweise exakt. Wie gesagt werden Ergebnisse zurückgeliefert mit mehr oder minderen Relevanz zum User. Relevanz lässt sich nicht anhand Richtig oder Falsch messen.
Folgerung	Deduktiv	Induktiv, IR-Systeme berechnen Ergebnisse aufgrund von einer Mehrzahl an Beobachtungen.
Modell	deterministisches Modell	auf Wahrscheinlichkeit beruhend
Klassifikation	monothetisch	polithetisch
Anfragesprache	formale Sprache	natürliche Sprache
Fragespezifikation	auf Vollständigkeit beruhend	Vagheit (s.o)
gesuchte Objekte	der Query-Spezifikation folgend	auf Relevanz basierend
Datenfehler-Reaktion	empfindlich	keine Reaktion

Tabelle 5.1: Unterschiede Data Retrieval - Information Retrieval

Und bezogen auf die Vagheit der Anfragen zu eine IR-System, wie vorher vermerkt, sind jene bei weitem nicht so durchformuliert, im Vergleich zum Daten-Retrieval-System.

Der Begriff der Relevanz hat im Information Retrieval einen hohen Stellenwert, und die Dokumente werden auch danach gereiht, je weniger ein Dokument dem Informationsbedürfnis des Users hilft, desto eher wird es als Fehler angesehen. Das Ziel eines Information Retrieval System ist es so wenig nicht relevante Dokumente und so viel wie möglich für den Benützer relevante Dokumente zurückzuliefern [12].

Insbesondere lassen sich Unterschiede an einigen markanten Punkten festmachen, siehe dazu Tabelle 5.1[95].

5.1.2 Evaluierung von Retrieval-Systemen

Sicherlich ist es möglich für den User eine Art Feedback-Funktion zu implementieren, worin er seine Zufriedenheit über die Treffer ausdrückt, es kann auch objektivere Wege geben.

Rechenmethoden existieren nämlich um die Treffer der Retrieval-Systeme zu evaluieren und somit auch das System selbst. Wichtige Kennzahlen in

dieser Hinsicht sind `Recall` und `Precision`, wobei beide in Formel wie folgt dargestellt werden können[118]:

$$Precision = \frac{\text{Anzahl der Dokumente in der Treffermenge, die relevant sind}}{\text{Anzahl der Treffermenge}} \tag{5.1}$$

$$Recall = \frac{\text{Anzahl der Dokumente in der Treffermenge, die relevant sind}}{\text{Anzahl aller relevanten Dokumente}} \tag{5.2}$$

Betrachtet man die Formeln, so kann man feststellen, dass es bei `Precision` darum geht, den prozentuellen Anteil von den relevanten Dokumenten aus der Dokumentenmenge, die man insgesamt durch das Retrieval System zurückbekommen hat, berechnet. Das mag durch menschliches Urteil einordenbar sein.

Betrachtet man aber `Recall` sieht man, dass es um den prozentuellen Anteil der erhaltenen relevanten Dokumente in Bezug auf die gesamte relevante Dokumentenmenge im System geht. Stellt man sich nur vor, man müsste diese z.B. aus einem System wie dem Austria-Forum, dass an die sechzig Tausend Dokumente enthält, händisch beurteilen, kann man wohl sagen, dass sich die `Recall`-Berechnung als schwierig erweist.

Recall-Annäherung

Zur Berechnung der vollständigen Menge an relevanten Dokumente existieren Methoden, die diese abschätzen versuchen[48]:

- Stichproben nehmen. Was insgesamt einen hohen Aufwand in großen Dokumentmengen bedeutet.
- Source-Dokument-Methode. Es werden Stichproben genommen, danach werden Fragen formuliert, die nach der Meinung der Beurteiler relevant zu den Dokumenten sind. Darauf folgt dann die Eingabe dieser Fragen ins Information-Retrieval-System, und der Vergleich mit der Treffermenge, ob das Dokument vorhanden ist. Daraus lassen sich Wahrscheinlichkeiten berechnen wie häufig ein relevantes Dokument in der Treffermenge existiert, was zu einer Näherung des Recalls führt. Ein Nachteil dazu ist, dass die Fragen nicht unbedingt den Fragen des normalen User gleichen müssen.
- Verallgemeinerung der Fragen für die Source-Dokument-Methode. Man erhält eine größere Menge in denen sich die Dokumente befinden.
- Externe Quellen. Ein Beispiel dazu: Fachleute befragen welche relevanten Dokumente sie zu entsprechenden Themen kennen. Danach die Liste der von der externen Quelle benannten relevanten Dokumente mit der Treffermenge vergleichen. In vielen Situationen aber existieren dazu keine externen Quellen, die man kontaktieren könnte.

- Anwendung verschiedener Information-Retrieval-Systeme. Man nimmt gleiche Anfragen her und wendet mehrere Retrieval-Systeme auf den gleichen Dokumentenbestand an.

5.1.3 IR in der Praxis

Hier findet sich die Vorstellung einiger Projekte, die die Techniken des Information Retrieval für unterschiedliche Zwecke benützen[48, 78]:

- Adhoc. Eine spontan-formulierte Anfrage mit darauf folgender Antwort des IR-Systems. z.B.: Google [1], Bing [2], Yahoo [3].
- Klassifikation. Es wird eine Einteilung über Klassifikationskriterien verwendet. z.B.: Yahoo Directory[4], Open Directory[5], Spam-Filter
- Klustering. Dokumente nach Ähnlichkeitsmerkmalen "klustern". z.B.: vivisimo[6]
- Informationsextraktion. Eine Filterung von spezifischen Informationen aus dem Dokumentenbestand. z.B.: Aktiendaten bei yahoo financial news [7]
- Textzusammenfassung. Automatische Zusammenfassung von Texten wie bei Nachrichtenüberblicks-Seiten,z.B.: google news[8], yahoo news[9]
- Frage-Antwort-System. Der Benützer stellt eine durchformulierte Frage, das System gibt eine Antwort. z.B.:wolfram alpha[10]
- Empfehlungs-System. Das System gibt aufgrund des Verhaltens des Benützers eine Empfehlung für weitere Schritte im System aus. z.B.: last.fm[11], amazon.com[12]

5.2 Klassische Modelle des IR

Grundsätzlich existieren drei klassische Modelle im Information Retrieval: Boolsches Modell, Probabilistisches Modell und Vektor Modell. Im Grunde trennt die Modelle die mathematische Ansicht auf Dokumente und Anfragen.

Boolsches Modell Es ist das älteste Modell, operiert über die Mengenlehre um dem Anfragen entsprechende Dokumente zu liefern. Querys wer-

[1] www.google.com
[2] http://www.bing.com
[3] http://www.yahoo.com
[4] http://dir.yahoo.com/
[5] http://www.dmoz.org/
[6] http://de.vivisimo.com/
[7] http://de.finance.yahoo.com/nachrichten
[8] http://news.google.at/
[9] http://news.google.at/
[10] http://www.wolframalpha.com/
[11] http://www.last.fm/
[12] http://www.amazon.de

den über logische Operatoren formuliert (AND,OR,NOT), wobei dies nur für das simple boolsche Modell zutrifft. Das "extended" boolsche Modell umfasst auch einen Operator, den sogenannten "proximity operator" `NEAR`, der die Nachbarschaft der Terme einbezieht. Die Treffer der Anfrage werden ohne ein Ranking wiedergegeben, und die Unterscheidung erfolgt binär also entweder relevant oder nicht relevant. Suchmaschinen verwenden das boolsche Modell nur teilweise[12, 78].

Vektor Space Modell Die Dokumente werden differenzierter angesehen als im boolschen Modell. Ein Dokument kann im Gegensatz zu der binären Logik des boolschen Modell für die User-Anfrage mehr oder minder relevant sein . Mathematische Berechnungen werden über Vektoren realisiert. Das Modell verwendet für die Score rationale Zahlen, welche normalerweise zwischen 0 und 1. Man spricht vom Ähnlichkeitsgrad[13] zwischen den Dokumenten[12, 78]. Ein Beispiel dafür ist Term Frequency/Inverse Document Frequency-Algorithmus, wobei dieser später noch genauer beschrieben wird.

Probabilistisches Modell Dieses Modell basiert auf Wahrscheinlichkeitsberechnungen. Es existieren mehrere unterschiedliche Techniken, sowie das BM25 Gewichtungsschema, Bayesian Netzwerk oder Binary Independence Modell. Das letztere zum Beispiel arbeitet anhand von Annahmen über relevante und nicht relevante Dokumente. Diese Annahmen können per User-Aktion oder durch weitere Annahmen verfeinert werden. Es hat ähnliche Ansätze wie das Boolsche Modell, in der Art binären Unterscheidung zwischen relevanter und nicht relevanter Dokumente[12, 78].

5.2.1 Stärken- Schwächen.

Das boolsche Modell mag für den Anfang am Intuitivsten erscheinen. Trotzdem unterscheidet das Modell nur zwischen relevant und nicht relevant. Nimmt man zum Beispiel das Austria-Forum mit seinen über 60.000 Dokumenten, und bekommt durch eine dementsprechende User-Anfrage eine Trefferliste von über Hundert Dokumente, die alle als gleich relevant gekennzeichnet sind, bleibt die Arbeit beim Benutzer den Treffern ein Ranking zu geben. Ab einer gewissen Menge an Dokumenten bedeutet das einen sehr großen Aufwand. Ein schon berechnetes Ranking zwischen den Dokumenten, sowie es das Vektor Space Modell durchführt, erscheint da praktikabler.

Es existieren einige Studien zum Vergleich der Modelle, in diesem Zusammenhang wurde festgestellt, dass das Vector Space Modell gleich gut wenn nicht besser als alle seine Alternativen im Bereich Dokumenten-Ranking ist. Noch dazu lässt es sich einfach und simpel durchführen. Daraus folgt auch seine Beliebtheit in der Praxis[12].

[13] degree of similarity

5.3 TF-IDF

5.3.1 Beschreibung

Wie schon vorher öfters erwähnt wurde bei meinem Projekt eine Information-Retrieval-Methode verwendet um topik-relevante Keywords zu identifizieren. Diese Methode nennt sich TF-IDF und steht als Abkürzung für Term Frequency - Inverse Document Frequency, wobei diese Methode zu den Vektor-Space-Modellen zählt.

Term-Frequency bezeichnet prinzipiell in diesem Kontext die Häufigkeit eines bestimmten Terms in einem Dokument. Es wird aber auch häufig normiert zu der Dokumentenlänge.

Inverse Document Frequency ist da etwas komplizierter zu erklären. Gut zu wissen in dieser Hinsicht ist, dass Document Frequency die Anzahl der Dokumente bezeichnet, die einen bestimmten Term enthält. IDF ist eine Art umgekehrte Document Frequency, was in den Formeln auch ersichtlich ist. Es bedeutet prinzipiell, dass je weniger Dokumente den Term enthalten, desto "wichtiger" wird er.

Das geht augenscheinlich auf folgende Gründe zurück. Ein Wort wird dann wichtiger, je öfter es im Dokument erwähnt wird. Jetzt gibt es aber Wörter, die sehr oft vorkommen und dennoch nichts über das Dokument aussagen. Je häufiger ein Term nun auch in anderen Dokumenten existiert, desto niedriger ist seine Wichtigkeit[119].

5.3.2 Formeln

Im Grunde passiert nichts anderes, als dass beide Maße miteinander multipliziert werden:

$$TF * IDF \tag{5.3}$$

Es existieren mehrere Abwandlungen von TF , der Term-Frequency (Term-Häufigkeit) [12, 119]:

$$tf_{i,j} = \frac{freq_{i,j}}{max_l freq_{l,j}} \tag{5.4}$$

$$tf_{i,j} = 1 + \log freq_{i,j} \tag{5.5}$$

$$tf_{i,j} = (K + (1 - K) * freq_{i,j}) \tag{5.6}$$

wobei $freq_{i,j}$ die Häufigkeit von Term i im Dokument j darstellt, $max_l freq_{l,j}$ den häufigsten Term im Dokument und K eine selbst gewählte Konstante zwischen 0 und 1 ist.

Weiters existieren mehrere Versionen für IDF der Inverse Document Frequency:

$$idf_i = \log \frac{N}{n_i} \tag{5.7}$$

$$idf_i = \log \frac{N - n_i}{n_i} \tag{5.8}$$

$$idf_i = \log(1 + \frac{n_m}{n_i}) \tag{5.9}$$

wobei N die Anzahl aller Dokumente im ganzen Bestand darstellt und n_i die Anzahl der Dokumente die den Term i enthalten und n_m der höchste IDF-Wert eines Terms in dem Dokumentenbestand ist. Da nun die Anzahl aller Dokumente durch die Anzahl der Dokumente, die den Term enthält, dividiert wird, folgt dahin, dass je weniger Dokumente den Term enthalten, der idf_i größer wird.

Am Ende werden beide Maße zusammengeführt:

$$w_{i,j} = f_{i,j} * idf_i \tag{5.10}$$

wobei nun $w_{i,j}$ die Gewichtung für den Term i des betreffenden Dokument j darstellt, und kann bezogen auf anderen Terme verglichen und entsprechend gereiht werden.

Dies ist natürlich nur ein Vorschlag den TFIDF zu berechnen, es existieren auch Abwandlungen dazu. Eine davon stammt von Gerald Salton et al.[98]:

$$w_{i,q} = (0.5 + \frac{0.5 * freq_{i,j}}{freq_{l,i}}) * \log \frac{N}{n_i} \tag{5.11}$$

Wobei $freq_{i,j}$ die Häufigkeit des zu bearbeitenden Wortes ist und $freq_{l,i}$ die Anzahl des am häufigst auftretenden Wortes im Dokument darstellt. N ist die Dokumentanzahl im System und n_i die Anzahl der Dokumente, wo das ausgewählte Wort vorkommt. Diese Formel wird auch in unserem Projekt zur TFIDF-Berechnung angewandt.

5.3.3 TF-IDF-Heuristiken

Auch lassen sich Heuristiken bezogen auf den TF-IDF anwenden. Im Grunde stellen diese Werte dar, die selbst zu wählen sind, um dadurch wenn möglich Rechenzeit zu sparen. In meinem Projekt wurde zwecks der Performanzeverbesserung solche Heuristiken angewandt. Es existieren zwei Schwellenwerte, die sich einerseits auf den TF andererseits auf den IDF beziehen.

- Unteres Limit an Häufigkeiten. Terme werden erst ab einer gewissen Häufigkeit in dem Dokument in die Berechnungen miteinbezogen.
- Länge des Wortes. Es gibt Untersuchungen, die gezeigt haben, dass die höhere Wortelänge auch einen höheren IDF-Wert bedeutet[100].

Beide Schwellenwerte sind nach eigenem Ermessen zu wählen, und in meinem Projekt wurde ein Interface implementiert, anhand man diese Heuristiken austesten kann.

Besondere Verwendung des TF-IDF findet sich in der Suchmaschinen-Engine Lucene, welches im Abschnitt 6.3.3 genau betrachtet wird.

Kapitel 6

Halbautomatische Verlinkung in großen Informationssystemen

Hier beginnt der praktische Teil der Diplomarbeit. Im nun folgenden Abschnitt wird die Problemstellung, die zu der Entwicklung eines solchen Verlinkungsassistenten führte, betrachtet. Darauf folgt eine Beschreibung der Anforderungen an das Verlinkungstool. Anschließend gibt es eine Ausführung über das Environment und die darin zugrundeliegenden Technologien, die teilweise auch im Projekt verwendet wurden. Schließlich widmet sich die Arbeit dem Aufbau des Verlinkungsassistenten und der genauen Beschreibung der einzelnen Komponenten.

6.1 Problemstellung

Es ist ein sehr großer Dokumentenbestand[1], den die Web-Enzyklopädie Austria-Forum[2] zu verwalten hat. Zu diesem Verwalten zählt es auch informative und hilfreiche Verknüpfungen zwischen den Dokumenten herzustellen.

Vernetzungen der Informationen spielen besonders in derart Informationssystemen eine große Rolle, wobei diese oft nur auf manueller Verlinkung basiert. Das kann zu einigen Problemen führen:

- Erhöhter Wartungsaufwand eines sich dynamisch ändernden Dokumentenbestands.
- Steigender Verwaisungsgrad der Dokumente in Web-Enzyklopädien. Siehe Abbildung 1.1 für die Statistik unverlinkter Dokumente in Wikipedia.

[1] An die 135.000 Dokumente, Anhänge dazugezählt[11]
[2] Mehr Informationen zum Austria-Forum findet sich unter Abschnitt 6.3.4.

- Fehlende Konsistenz der Verknüpfungen. Bei einem kollaborativen Hypertext-System stellt das aus dem Grund ein Problem dar, weil verschiedene Autoren mit verschiedenartigem Wissen die Dokumente anders verlinken[41].
- Sinkendes User-Interesse. Prozentuell wenig Benutzer folgen den Verlinkungen im Text und verlassen die Seite sofort wieder. Sowohl in Wikipedia als auch im Austria-Forum stellt das ein Problem dar[54][63].

Zum Entgegenwirken der vorhergenannten Probleme, wurde beschlossen eine Erweiterung des Austria-Forums zu implementieren, welche das Vernetzen der Dokumente erleichtern soll.

6.2 Anforderungen

Ziel ist es einen Verlinkungsassistenten für die Web-Enzyklopädie Austria-Forum zu entwickeln, welcher folgende Anforderungen erfüllen soll:

Präsentation Das Programm soll dem User Links präsentieren, die über Techniken des Information Retrieval berechnet werden.

Interaktion Die vorgeschlagenen Links sollen durch den User überprüfbar und auswählbar sein.

Session Es soll möglich sein für mehrere Wörter pro "Verlinkungs-Session" Linkvorschläge zu erstellen. Das heisst der User kann mehrere Wörter pro Session verlinken.

Speicherung Das Programm soll die Möglichkeit haben das Dokument zu verändern, um den gewählten Link bzw. die gewählten Links einzufügen. In anderen Worten muss es die Rechte haben, um auf den Dokumentenbestand schreibend einzuwirken.

UserInterface Das Tool soll für den Gebrauch ein UserInterface enthalten. Der GUI-Teil soll in der Funktionalität unabhängig zu der Hauptlogik, der Berechnung der Links sein.

Modularität Der Teil zur Berechnung der Links soll möglichst modular implementiert werden, um dadurch die Wiederverwendbarkeit bei einer anderen Projektumgebungen zu steigern.

Erweiterbarkeit Gemäß der Möglichkeit einer Änderung der Umgebung soll die Bibliothek an den Schnittstellen zur Datenbank oder Syntax-Filterung erweiterbar bzw. anpassbar sein.

6.3 Environment

Prinzipiell sollen hier alle technologischen Komponenten, auf dem das Austria-Forum aufbaut, beschrieben werden. Das Austria-Forum kann aus technischer Sicht als ein JSPWiki-Derivat gesehen werden. Das Wiki-Framework JSPWiki wird im Abschnitt 6.3.3 beschrieben. Es baut auf Java-Webtechnologien

auf, welche unter Abschnitt 6.3.2 näher betrachtet werden. Das Austria-Forum selbst und seine Erweiterungen werden unter 6.3.4 ausgeführt. Der folgende Abschnitt handelt von der Programmiersprache Java, welche die Basis für die vorhergenannten Technologien ist.

6.3.1 Java

Java wurde von Sun entwickelt und ist mittlerweile im Besitz von Oracle[3]. Die Wahl dieser Programmiersprache beinhaltet folgende Vorteile[90]:

Objekt-Orientiert Java ist eine objektorientierte Programmiersprache. Objekte werden aus Klassen instanziert, sie enthalten Funktionen und Datenelemente bzw. auch Objekte selbst. Klassen vermögen voneinander abgeleitet werden, dadurch lassen sich Eigenschaften vererben. Außerdem kann man Objekte voneinander abkapseln, so dass diese nur bei spezifizierten Schnittstellen angesprochen werden können. Noch dazu erleichtert die Intuitivität der Objektorientiertheit von Java ein verständlicheres und einfaches Designen von Software.

Umfangreiche Java-API Die Java-API enthält eine große Sammlung von Funktionen, die für alle mögliche Operationen unterstützend eingesetzt werden kann. Hier ein Link zu der offizielle API-Spezifikation der 5.0 Version von Java: http://download.oracle.com/javase/1.5.0/docs/api/.

Syntax Die Syntax hat Ähnlichkeiten zu anderen objektorientierten Sprachen wie C++ und C#. Das erleichtert die Umstellung.

Plattformunabhängig Java selbst läuft in einer Virtual Machine, die für jede Plattform ob Windows, Linux, OS X etc. frei erhältlich ist. Großer Vorteil dabei ist, dass der Code sich plattformunabhängig ausführen lässt.

Garbage-Collector Der Garbage-Collector ist praktisch der Teil der Virtual Machine, der den nicht mehr benötigten Speicher verwaltet. Das heisst, dass der Programmierer sich nicht mehr direkt um die Speicherfreigabe, wie in anderen Sprachen üblich, kümmern muss.

Robustheit Java ist robust. Es wurde ausgelegt, dass es mithilfe der Virtual Machine die Programmier-Fehler früh erkennt, was das Entwickeln erheblich erleichtert. Pointer, wie sie in C und C++ verwendet werden und welche den Speicher korrumpieren können, sind nicht erlaubt, was zusätzlich zur Robustheit beiträgt.

6.3.2 Apache Tomcat + Java EE

Um das Wiki-Framework JSPWiki auszuführen ist ein Server nötig. Im Fall des Austria-Forums handelt sich um den Apache Tomcat.

[3] http://www.oracle.com/at/index.html

Der Apache-Tomcat-Server ist ein javafähiger Server. Er wird auch als Servlet-Container bezeichnet. Die neueste stabile Version von Tomcat ist 6.0.30 und die Applikation wird als Open-Source bereitgestellt[8].

Zur Entwicklung von Java im Web wird die Java Enterprise Edition der Version 5 verwendet, die der Tomcat-Server unterstützt. Die wichtigsten Komponenten darin sind Servlets, JavaServerPages und Filter.

Servlets sind im Grunde Java-Klassen, welche vom javax.servlet.HttpServlet abgeleitet sind und dementsprechende Funktionalität inkludieren und leicht zu erweitern sind. Sie spielen eine zentrale Rolle in Java-Webapplikationen, und werden in den MVC-Web-Architekturen als Controller eingesetzt[104].

Des weiteren existieren JavaServerPages. Sie bestehen aus Java-Code und HTML und werden zur Darstellung verwendet. Die JavaServerPages, welche die Dateiendung .jsp haben, werden intern vom Tomcat zu Servlets kompiliert [89].

Filter bzw. Servlet-Filter, eine weitere Komponente in Java-Webapplikationen, bieten die Möglichkeit Code, bevor spezifizierte Servlets aufgerufen werden, auszuführen. Man kann definieren, wo und bei welchen Requests der Filter eingesetzt wird. Es lassen sich auch Filter-Ketten, also mehrere Filter hintereinander, einrichten[103].

6.3.3 JSPWiki

Das JSPWiki[66] wurde von dem Finnen Janne Jalkannen, welcher ein studierter Physiker ist, 2001 ins Leben gerufen. Beruflich ist er als Chief Technology Officer bei der Firma Thinglink, einem Unternehmen, das einen Alternativ-Service zur DOI [4] entwickelt, tätig[105].

Jalkannens JSPWiki ist OpenSource und über die Webseite http://jspwiki.org erhältlich. Es lässt sich per WAR-Paket, einem Web-Container zu Deployement-Zwecken, sehr einfach installieren. Das Wiki basiert auf der Java-Version J2EE 1.4. Es besteht hauptsächlich aus JavaServerPages und einigen Servlets sowie auch Filtern etc. [5].

JSPWiki-Features

Hier muss angemerkt werden, dass aufgrund der logischerweise fortschreitenden Implementation einige JSPWiki-Features von Version zu Version anders sind. Es wird hauptsächlich auf die Basis-Features eingegangen[40]:

- Verwendung der JSP-Technologie. Ermöglicht Trennung von Darstellung, Logik und Daten.
- Mächtiges Plugin-System. Es lassen sich Plugins erstellen, die in jede Wiki-Seite eingebunden werden können. Vorraussetzung zur Entwicklung solcher Plugins sind Kenntnisse in Java. Mehr dazu folgt später.

[4] Digital Object Identifier

[5] Mehr Informationen dazu findet sich unter Abschnitt 6.3.2

- Beinhaltet ein Revision Control System, dadurch lässt sich eine Versionskontrolle der Wiki-Dokumente durchführen.
- Flexible Datenspeicherung. Mithilfe des JDBC-Treibers lässt sich eine Vielzahl von Datenbanken einbinden, wobei auf Plain-Text-Speicherung auch umgestellt werden kann. Die Verwaltung der Seiten des JSPWiki kann auch durch WebDAV erfolgen.
- Ausgereiftes Privilegien-System für Gruppen und Benützer. Die Speicherung erfolgt in XML und kann wahlweise dort oder über das JSPWiki-Interface editiert werden.
- Implementation der Lucene-API ist vorhanden und wird u.a. zur Volltext-Suche verwendet[6].
- Dank Java lässt sich das Wiki einfach in gängige IDEs einbinden (z.B. Netbeans, Eclipse etc) .

Die Abbildung 6.1 zeigt den Aufbau der JSPWiki-Klassen . Wie die einzelnen Komponenten in Echtzeit miteinander kommunizieren, lässt sich im Sequenz-Diagramm in der Abbildung 6.2 betrachten.

Lucene

Eine wichtige JSPWiki-Komponente ist das Such-Engine Apache Lucene, welches zur Volltext-Suche eingesetzt wird. Es ist dafür zuständig die Dokumente zu indizieren und für die Suche zu optimieren. Da Lucene auch in Java entwickelt wurde und eine große Anzahl an Features enthält, ist es sehr geeignet für das JSPWiki.

Es folgt eine Aufzählung mehrere Eigenschaften der Lucene-Bibliothek[7]:

Hohe Performance Es braucht erstens nur wenig Arbeitsspeicher zur Ausführung, zweitens kann es 20 Megabyte pro Sekunde indizieren, auch wenn die Hardware, auf der es läuft, durchschnittlich ist[7]. Drittens komprimiert es die indizierten Dokumente zu 20-30% der normalen Größe.

Vielfalt der Suchanfragen Wildcards mit Regular Expressions sowie Suchanfragen über Zeiträume sind möglich. Eine detailiertere Beschreibung befindet sich in Abschnitt 6.3.3.

Definieren nach Felder Ein wichtiges Werkzeug in Lucene sind die Felder, die so eine Art Namespaces darstellen. Ein Dokument hat z.B.: normalerweise Felder wie: Autor, Titel, Jahr, etc. Intern wird beim Indizieren je nach Implementation die Daten nach Felder gespeichert. Auch nach diesen lässt sich suchen, z.B.: Autor: Kant. Weitere Erklärungen dazu folgen später.

Multithreaded Erlaubt gleichzeitiges Suchen und Indizieren.

[6] Mehr Information dazu unter Abschnitt 6.3.3

[7] Z.B.: ein Pentium M 1.5GHz

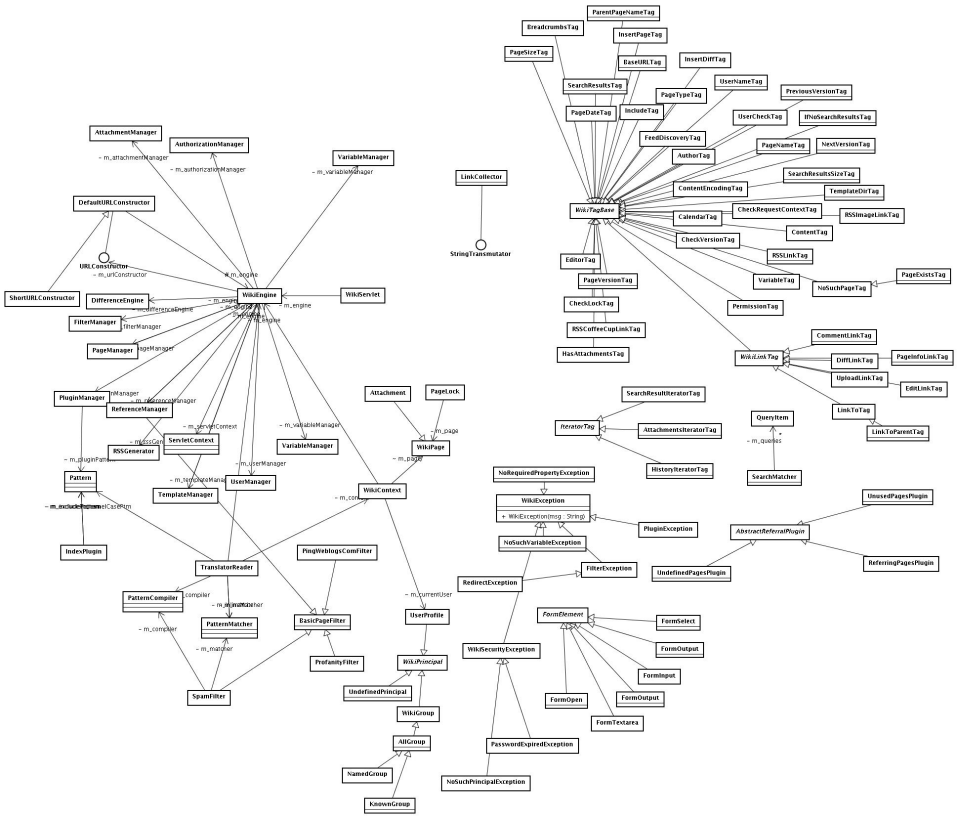

Abbildung 6.1: Umfassendes Klassendiagram der JSPWiki-Applikation[66]

Open-Source Die Community wird dadurch eingebunden, was die Weiterentwicklung bzw. Verbesserung unterstützt. Außerdem ist die Lizenz (Apache License), so ausgelegt, dass sich Lucene nicht nur privat, sondern auch kommerziell nutzen lässt.

Implementation in mehren Sprachen Obwohl es ursprünglich in Java geschrieben wurde, gibt es Implementation in C++, C, C#, Perl etc. [8]

Da nun Lucene nur eine Bibliothek ist, muss zur Anbindung an dieser eine Implementation her. Im Austria-Forum bzw. JSPWiki ist diese unter com.ecyrd.jspwiki.search gespeichert. Die Klasse LuceneSearchProvider spielt darin eine wichtige Rolle. In Abbildung 6.3 befindet sich ein Klassendiagramm von com.ecyrd.jspwiki.search.

Suchanfragen-Syntax Wie gesagt, existiert eine große Vielfalt bzw. einige Möglichkeiten Suchanfragen zu definieren:

[8] http://wiki.apache.org/lucene-java/LuceneImplementations

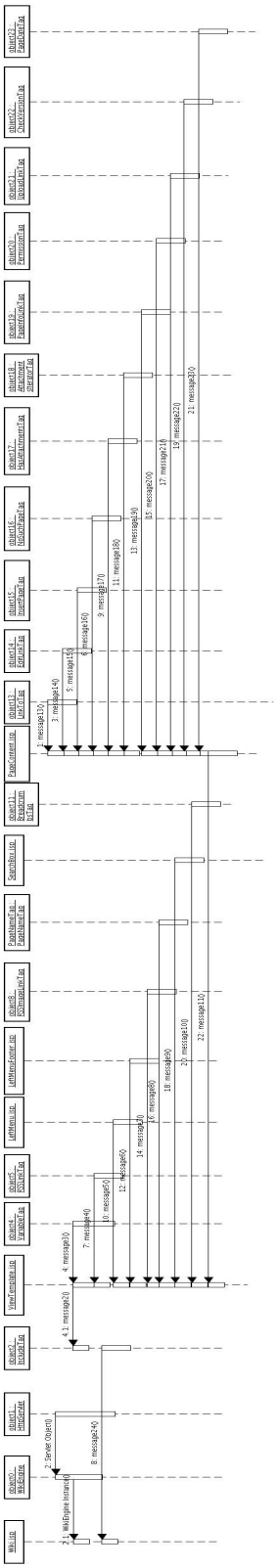

Abbildung 6.2: Sequence Diagramm der JSPWiki-Applikation[66].(Vertikal gestellt)

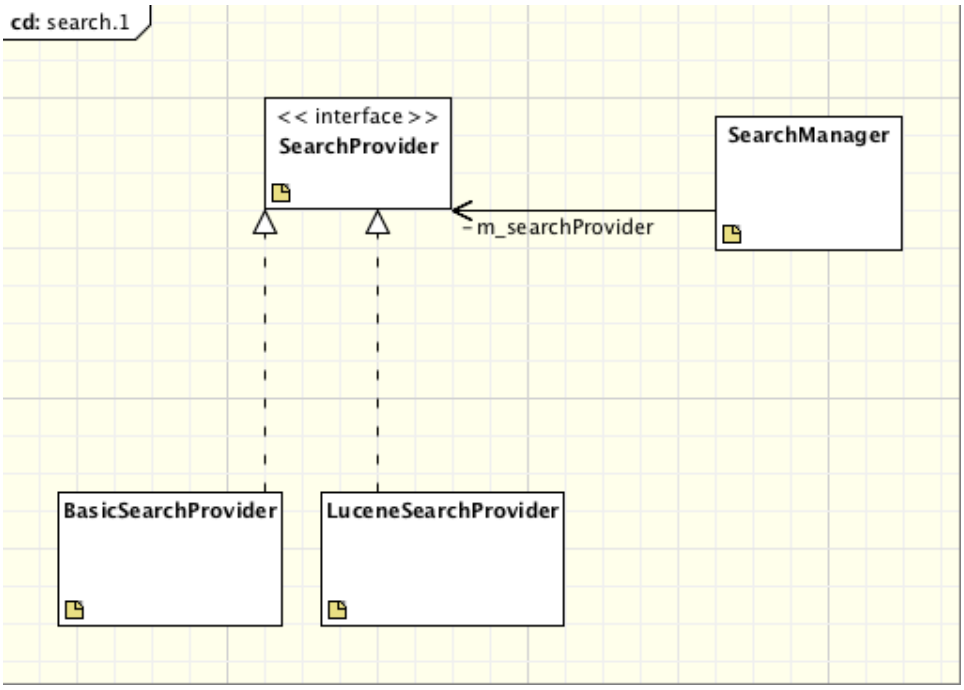

Abbildung 6.3: Ein Klassendiagramm von com.ecyrd.jspwiki.search, wo praktisch die Suche und das Indizieren von JSPWiki implementiert ist. (Erstellt mit UML Poseidon[53])

Einfacher Text Ein einfacher Text wie "Parkwächter Strafzettel" kann von dem Lucene Engine ausgewertet werden und die Dokumente werden dementsprechend gewichtet zurückgegeben. Die Gewichtung wird später erläutert.

Felder Felder kann man wie gesagt als Namespaces deuten. Z.B. folgende Suchanfragen: "Autor:Immanuell Kant", wird alle Dokumente die mit diesem Feld verbunden sind zurückgeben. Falls das Feld nicht existiert, gibt es eine Fehlermeldung.

Boolsche Operatoren Über die Boolschen Operatoren lassen sich die Terme miteinander verbinden, sollte kein Operator definiert sein, wird eine ODER-Verknüpfung angenommen. Die Anfrage "Parkwächter Strafzettel" wird also als "Parkwächter OR Straftzettel" interpretiert. Andere selbsterklärende Operatoren sind AND und NOT.

Ein weiterer Operator stellt + dar, wobei man leicht glauben kann, dass man dadurch die gleiche Funktion, wie die von AND bekommt, jedoch bedeutet dieser Operator folgendes: Nimmt man z.B. "+Parkwächter Strafzettel" , so heisst das für Lucene: Gib mir ein Dokument wo auf jeden Fall Parkwächter inkludiert ist, und vielleicht (nicht zwingend) auch Strafzettel.

Range Hat man ein Feld definiert, welches einen Integer nimmt, z.B. ein Feld, das eine Jahreszahl speichert, so vermag man auch nach einem Zeitraum zu suchen. Z.B.: Durch `Erstellungsjahr:[2008 TO 2010]` durchsucht man nun alle Dokumente, die innerhalb 2008 und 2010 erstellt wurden. Während aber z.B.: `Erstellungsjahr:{2008 TO 2010}` nur Dokumente aus dem Jahr 2009 zurückliefert. Die Range-Eigenschaft lässt sich nicht nur auf Felder mit Datumswerten anwenden, sondern auch auf normale Wörter, wobei das Kriterium für Lucene in dem Fall die alphabetische Reihenfolge ist.

Wildcards Über Wildcard, die auch bei den regulären Ausdrücke benützt werden, lassen sich ungenaue Suchbegriffe angeben. Es existieren zwei operatoren "?" und "*" , über das Fragezeichen kann man nach einem Buchstaben suchen lassen, den man noch nicht kennt: z.B. "L?st" könnte nun List gemeint sein oder Last, uvm. Über den *-Operator lässt sich angeben, dass an der Stelle noch mehrere unbekannte Buchstaben positioniert sind, z.B.: "Park*" könnte Parkwächter oder Parkbank gemeint sein. Andererseits ist die Positionierung nicht nur am Anfang des Wortes oder am Ende sondern auch inmitten des Wortes: "P*wächter" möglich. Wobei der *-Operator für keinen Buchstaben auch stehen kann, sprich "Park*" kann auch nur Park sein.

Boosting-Term Falls der Benützer einen Term hervorheben möchte, kann er das über eine bestimmte Methode höher gewichten. Wenn man nach einem Ort in der Steiermark suchen möchte, und zum Beispiel bei der Suchanfrage: `Ort Steiermark` immer nur Dokumente mit allgemeinen Informationen über die Steiermark zurückbekommt, kann dem Term `Ort` mehr Wichtigkeit verliehen werden, indem man das Caret-Zeichen verwendet, dass in der Mathematik auch bei Potenzen zu finden ist. Mit der Suchanfrage `Ort^4 Steiermark` würden in dem Fall Dokumente mit dem Term `Ort` im Ranking bevorzugt werden. Die genauen Gründe dazu hängen mit dem internen Scoring von Lucene zusammen und werden im nächsten Abschnitt erläutert.

Scoring der Dokumente Lucene erstellt eine Einteilung der Dokumente, bezogen auf die Suchanfrage. Die Bewertung der Dokumente wird über einen erweiterten TFIDF[9] erstellt. Allgemeine Informationen über den TFIDF-Algorithmus kann man unter Abschnitt 5.3 finden.

In Lucene wurde eine eigene komplexere Abwandlung des TFIDF-Algorithmus implementiert. Die wie folgt aussieht[6]:

$$score(q,d) = coord(q,d) * queryNorm(q) * \sum_{tinq} tf(tind) * idf(t)^2 * t.getBoost() * norm(t,d) \tag{6.1}$$

[9] Termfrequency * Inverse Document Frequency.

Zur Erklärung der Formel:

1. $score(q, d)$. Ist die Bewertung des Dokuments d auf die Query q.
2. $coord(q, d)$. Dieser Faktor berechnet, wieviel Terme der Suchanfrage gleichzeitig im Dokument sind, je mehr desto höher ist gewöhnlich das Ranking.
3. $queryNorm(q)$. Stellt einen normalisiernden Faktor dar, welcher keinen Einfluss auf das Dokumenten-Ranking nimmt, da alle Dokumente mit dem gleichen Faktor multipliziert werden. Es sollen dadurch verschiedene Suchanfragen vergleichbar werden.
4. $tf(tind)$. Termhäufigkeit, wird über folgende Formel intern berechnet: $tf(tind) = \sqrt[2]{haeufigkeit}$
5. $idf(t)$. Die Inverse Document Frequency, sie ist in Lucene als diese Formel realisiert: $idf(t) = 1 + \lg(\frac{numDocs}{docFreq+1})$ Wobei $numDocs$ die Anzahl aller Dokumente im System darstellt und $docFreq$ die Anzahl der Dokumente zeigt, die den Term beinhalten.
6. $t.getBoost()$. Dieser Faktor lässt sich über den sogenannten Boosting-Term direkt beeinflussen, damit man einzelne Terme in der Suchanfrage wichtiger machen kann.
7. $norm(t, d)$. Hier werden einzelne Faktoren hineinberechnet, die schon vor dem Indexen des Dokuments gesetzt wurden, hinzukommt auch ein Maßstab, der sich mit der Länge der einzelnen Felder im Dokument, beschäftigt.

Plugins im JSPWiki

Plugins spielen eine wichtige Rolle für die Entwickler im JSPWiki. Es lassen sich dadurch auf einfache Weise Erweiterungen in das System eingliedern, welche durch den Editor mit wenig Vorwissen verwendet werden kann. Unter Abschnitt 6.3.4 bzw. 6.3.4 werden von mir erstellte Plugins beschrieben. Eine ausgiebige Aufzählung der gesamten Austria Forum-Plugins findet sich unter http://austria-lexikon.at/af/Hilfe/Plugins.

Ein Plugin wird dadurch eingefügt, indem man im Wikieditor einen speziellen Code eingibt. Er besteht grundsätzlich aus der Zeichenfolge "[{" und wird abgeschlossen durch "}]". Das erste Wort nach der eröffnenden Zeichenfolge stellt den Namen des Plugins dar. Wird der Name des Plugins falsch eingegeben, bekommt der Benützer auf der entsprechenden Seite einen Plugin-Fehler. Nachdem das Plugin richtig benannt wird, kann der Editor dem Plugin einige Parameter übergeben. Ein allgemeines Beispiel dazu:

```
[{Plugin-Name Parametername='Parameterwert'}]
```

Plugins befinden sich unter dem Package `com.ecyrd.plugins` im JSPWiki. Wie die Klassen in dem Package organisiert sind, kann man unter

Abbildung 6.4 betrachten. Das Pluginsystem wird bei dem Verlinkungstool genutzt, um das Einsetzen der Daten in das User-Interface zu erleichtern. Es wurde ein LinkSuggestionPlugin erstellt, durch welches es möglich ist, auf einfache Weise Linkvorschläge in den Text zu inkludieren. Unter 6.7.1 findet sich mehr über das LinkSuggestionPlugin.

Wiki-Tags

Neben dem Plugin-System enthalten die Wiki-Tags eine weitere Möglichkeit der Erweiterung des Systems. Um einen Tag hinzufügen müssen folgende Konventionen eingehalten werden:

- Die Klasse, in der die Tag-Funktionalität entstehen soll, wird im normalen Fall von WikiTagBase abgeleitet.
- Es müssen durch die Vererbung spezielle Funktionen implementiert werden.
- Der Name der Klasse wird als Tagname verwendet. Im Fall von z.B.: `<wiki:translate>` handelt es sich um die Klasse TranslateTag, welche JSPWiki-Dokumente ins HTML konvertiert.

Unterschiede zu Plugins sind: Tags lassen sich in die JavaServerPages eingliedern, können also nur serverseitig hinzugefügt werden, während Plugins über die JSPWiki-Dokumente einzubinden sind und somit auch lokal verändert werden können.

Tags werden eher für technische Funktionen eingesetzt. Es ist ein Feature, das mehr auf der Entwicklerseite Verwendung findet im Gegensatz zu Plugins, die hauptsächlich von den Editoren gebraucht werden.

Das Klassendiagramm zu den WikiTags im Austria-Forum lässt sich in der Abbildung 6.5 betrachten.

6.3.4 Austria-Forum

Hier folgt eine umfassende Beschreibung des Austria-Forums, das wie gesagt auf dem JSPWiki basiert, mit einer Liste an Erweiterungen, die ich dafür entwickelt habe.

Als Erstes gibt es einen Abschnitt über die lobenden Pressestimmen, als die Enzyklopädie online ging. Danach folgt die lange Geschichte dieses Web-Lexikons. Dann wird auf die Grundsätze und Struktur eingegangen. Schlussendlich gibt es eine Aufzählung der Features durch welches sich das Austria-Forum vom normalen JSPWiki abhebt, sowie eine Liste meiner eigens entwickelten Erweiterungen.

Pressestimmen

"Neue Österreich-Enzyklopädie geht online" titelte die Kleine Zeitung am 9.10.2009. Als "kostenlose Internet-Enzyklopädie" über Österreich bezeich-

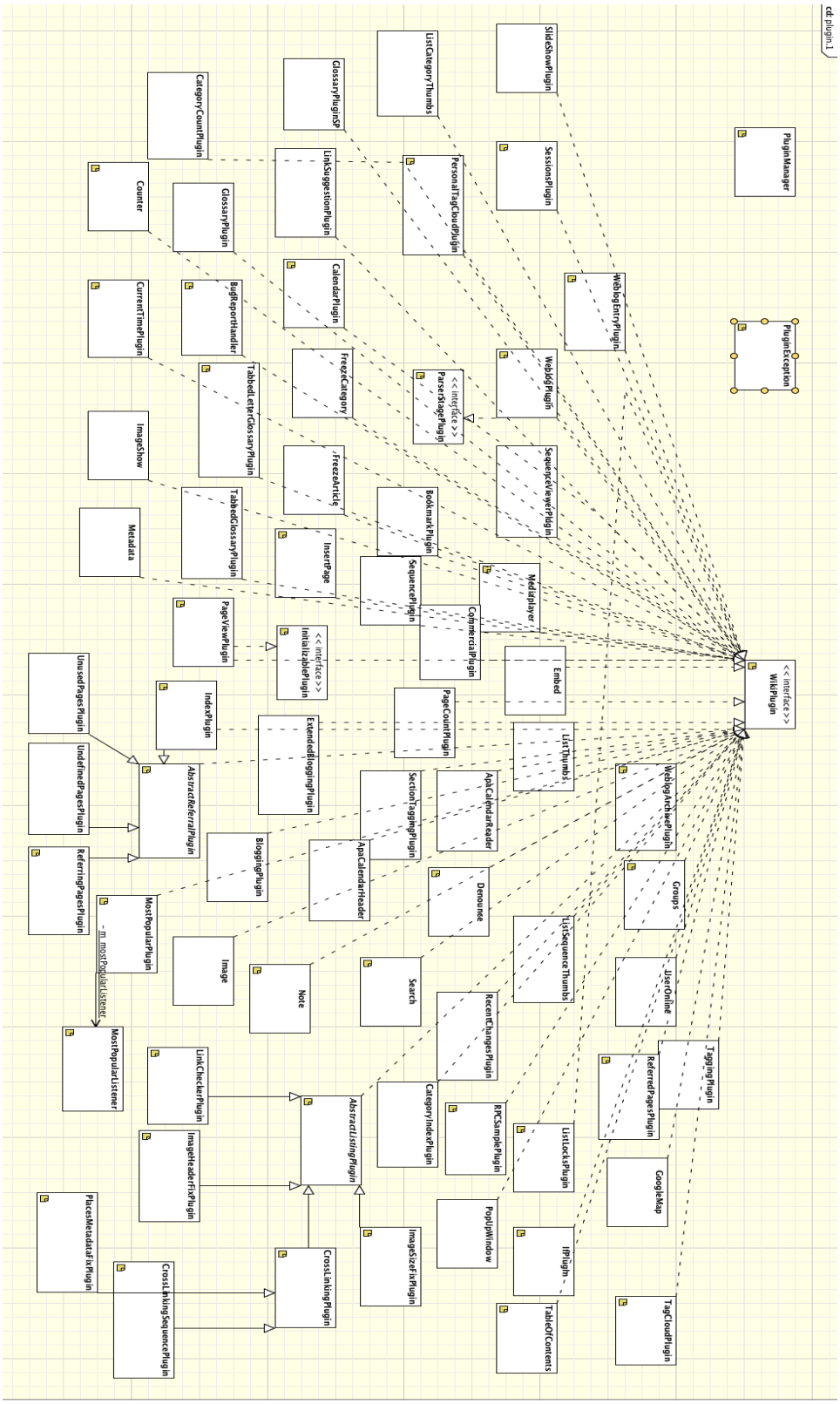

Abbildung 6.4: Ein Klassendiagramm des com.ecyrd.plugins Packages, worin das ganze Plugin-System von JSP-Wiki implementiert ist. Wie man sieht, werden die Plugins, um inkludiert zu werden von WikiPlugin abgeleitet.

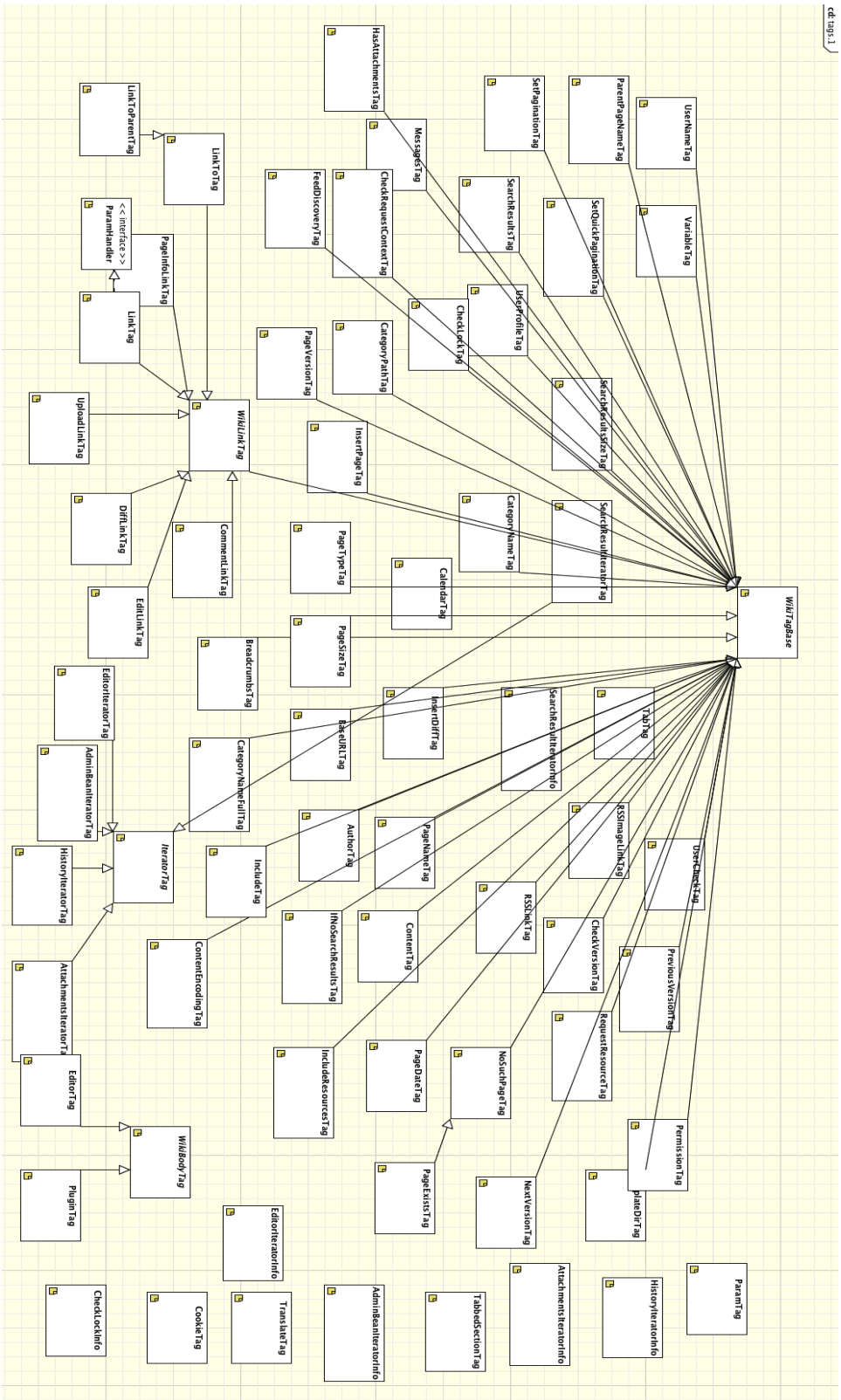

Abbildung 6.5: Klassendiagramm von com.ecyrd.tag von dem Austria-Forum

nete der Artikel das Projekt. Damals umfasste die Sammlung an Datenbeständen, also Text und Anhänge, eine Zahl von über 97.000 Objekte[71].

Und die Zeitung "die Presse" ging noch genauer ins technische Detail und verriet, dass nach Prof. Hermann Maurer, dem Initiator des Austria-Forums, es sich hier nicht nur um ein normales Lexikon handelt, jedoch vielmehr um einen Lexikon-Cluster. Hervorgehoben wurde in dem Artikel auch, dass jeder Beitrag mit einer Begriffswolke versehen ist, dadurch ist es möglich nach ähnlichen Beiträgen zu suchen, und die Zugehörigkeit zu Themengebieten zu erschließen. "Structered semantic Wiki" wird das Austria-Forum in dem Zusammenhang bezeichnet [37].

Der Standard widmet sich der Sache in einem knappen relativ neuen Artikel und titelt "Wikipedia in Rot-Weiß-Rot". Darunter wird im Vergleich zu Wikipedia hervorgehoben, dass Editierrechte hauptsächlich von Wissenschaftern verfasst werden und von normalen Usern nicht verändert aber kommentiert werden können[35].

Was bleibt nun von den anfänglichen Pressemeldungen bis jetzt. Konnte das Austria-Forum die Euphorie befriedigen? Vielleicht lässt sich das in Zahlen ausdrücken.

Wie schon vorher erwähnt hatte das Austria-Forum beim Launch am 1.9.2009 eine Anzahl von 97.000 Dokumente und Anhänge. Nach ungefähr zehn Monate hat sich die Zahl laut Statistik vom 22.07.2010 auf über 135.000 Objekte erhöht. Das macht summa summarum einen Zuwachs von ungefähr 40% in ungefähr neun Monaten[11].

Geschichte

Die Grundidee des Austria-Forum ist schon älter und es existieren einige Vorgänger-Versionen zu dem Projekt.

Schon 1996 entstand unter dem damaligen Namen "AEIOU" ein System, das als die erste Vorversion zum Austria-Forum gesehen werden kann. Unter anderem fanden sich im AEIOU Fernsehaufnahmen des Österreichischen Rundfunks in digitaler Form, sowie Musikalben mit Beethoven, Mozart und vielen mehr[107].

Dann im Jahre 2006 folgte eine Weiterentwicklung des AEIOU unter dem Projektname Alexander[65]. Über das Wissensportal Alexander sollten lexikalische sowie journalistische Quellen mit Benützerbeiträgen verknüpft und damit etwas Neuartiges gegenüber der mächtigen Online-Enzyklopädie Wikipedia geschaffen werden[16].

Vom technischen Blickwinkel baute Alexander auf Hyperwave einem Wissensmanagementsystem spezialisiert auf Collaborativem Enterprise Content Management, auf.

HyperWave ist die kommerzielle Version von Hyper-G[10], einem Publishing-System, mit eingebauten Features , die das Web in einigen Aspekten über-

[10]Siehe für weitere Beschreibung Abschnitt2.3

flügeln soll[111].

Vorerst erfolgte ein Testbetrieb an denen nur Abonnementen der Zeitung "die Presse" teilnehmen durften. Die geschah aus dem Grund, da der Besitzer der Presse die Styria Sponsor war. Mit dabei beim Sponsoring war auch der Brockhaus-Verlag, welcher dem lexikalischen Teil die Professionalität geben sollte[49].

Mit dem Absprung eines wichtigen Mitsponsors dem Meyers Verlag nahm das Projekt einen schweren Rückschlag. Aus diesen und anderen Gründen schaffte es Alexander nicht aus dem Testbetrieb hervorzutreten, und das Projekt wurde eingestellt[107].

Letztlich führte das Prof. Hermann Maurer, dem Initiator der vorhergenannten Enzyklopädie-Projekte, zu der Idee des Austria-Forums, welches dann in seiner ersten Version implementiert wurde.

Zwei Jahre lief das Austria-Forum im inoffiziellen Status. In dieser Zeit fand vorwiegend Editorentätigkeit statt. Nach einer Evaluierung und der Filterung der Probleme wurde eine technologische Neuorientierung überlegt und durchgeführt. Dieser Prozess wird genau beschrieben in der Diplomarbeit[107] von Christoph Trattner, welcher eine Evaluierung von Wiki-Hypertextsystememem durchführte und eine Empfehlung für das JSPWiki aussprach, dieser man auch folgte. Das neue Austria-Forum wurde nun auf der Basis von JSPWiki[11] von Janne Jalkannen aufgebaut[107].

Das Austria-Forum in der Form, in der Form es heute existiert, ging schließlich am 01.09.2009 offiziell online.

Grundsätze & Struktur

Die Grundsätze des Austria-Forums[79]:

- Bekenntnis zur Demokratie und der Verfassung
- Politische Unabhängigkeit
- Publiziert durch unabhängige Wissenschafter

Struktur des Austria-Forums[79]:

AEIOU Österreich Lexikon Darin sind die alten Teile des AEIOU enthalten (welche von der Vorgängerversion stammen), wobei große Überarbeitungen zur Aktualität beitragen. Auch inkludiert in diesem sind Teile des Ueberreuter Lexikon 2003 "Österreich von A-Z".

Spezialisierte Wissenssammlungen (Speziallexika) Es wird in fächerspezifische Themen eingeteilt, sowie österreichische Flora, Briefmarken, Symbole und viele andere Kategorien.

Community Jeder Benutzer kann hier seine Interessen und auch alles Sonstige, was nicht in die anderen Kategorien passt, thematisieren.

[11] Weitere Informationen dazu findet sich unter Abschnitt 6.3.3

Allgemeine Austria-Forum-Erweiterungen

Es geht hier vorwiegend um eine Auswahl an Erweiterungen des Austria-Forums zum JSPWiki[107]. Für die Features der normalen Version des Wikis siehe Abschnitt 6.3.3.

- **Strukturierter Inhalt.** Ziel dabei war es dem Wissen, das im Austria-Forum gespeichert und editiert wird, eine dem Denken ähnliche Strukturierung zu geben, um dem Benützer den wichtigen Kontext zu übermitteln und ihm dadurch einen größeren Überblick zu ermöglichen. Um diese Ziele zu erreichen, wurde die Technik des infiniten Subpaging verwendet, mit dem sich Wissens-Hierarchien abbilden lassen. Die Hierarchen werden über die URLs ersichtlich. Außerdem wird die Filestruktur auf dem Server nach genau denselben Kategorien abgebildet. Zusätzlich verfügen die Seiten über sogenannte Breadcrumbs, um dem User die Erfassung des Kontexts zu erleichtern.
- **Austria-Forum-Plugins** Das InsertPage-Plugin wurde erweitert. Kategoriespezifische Plugins wie das CategoryIndexPlugin, GlossaryPlugin sowie TabbedGlossaryPlugin wurden erstellt.
- **Tagging.** Tagclouds wurden zu den Dokumenten hinzugefügt, über welche ein intuitives Navigieren möglich ist.
- **Such-Erweiterungen.** Suche nach Kategorien wurde konsequenterweise hinzugefügt. Der User kann diese selbst spezifizieren, nach denen er suchen möchte, oder automatisch in der Kategorie suchen lassen, in welcher er sich gerade befindet

Meine Austria-Forum-Erweiterungen

Hier findet sich die Liste der wichtigsten Features, die ich für das Austria-Forum programmiert habe.

Doubleclick and Search

Im Austria-Forum existiert am rechten oberen Teil der Seite ein immanentes Suchfeld . Dieser "Suchkasten" ist statisch angelegt, und greift intern auf die mächtige Such-Engine Lucene[12] zu, sobald der Benützer eine Suche durchführen möchte. Also klickt der User im normalen Fall auf das Formularfeld im Suchkasten, gibt seine gewünschte Anfrage ein und bestätigt diese entweder mit der Taste Enter oder durch einen Klick auf das Symbol der Lupe neben dem Feld.

Das erfordert einigen Aufwand. Was wäre z.B., wenn der Benützer ein interessantes Wort, worüber er mehr wissen will, inmitten des Text des Dokuments findet. Er müsste um danach zu suchen es entweder kopieren und in das Suchkästchen einfügen oder es dort manuell eintippen.

[12] Siehe Abschnitt 6.3.3 für mehr Informationen.

Um diesen Fall zu erleichtern habe ich eine Hilfsfunktion entwickelt.

Nimmt man das vorige Beispiel noch einmal: Der User findet ein Wort von Interesse im Text. So kann er nun das Wort mithilfe meiner Implementation einfach per Maus doppelklicken, und bekommt damit eine automatische Suchanfrage auf das für den Benutzer interessante Wort.

Im Grunde wurde dieses Feature auf Basis von Javascript realisiert. Da Javascript bekannterweise bei einigen Browser verschieden implementiert wurde, musste die Funktion quasi doppelt erstellt werden. Einmal um es dem Internet Explorer verständlich zu machen und dann für den Rest der existierenden Browsern.

Das Feature lässt sich nur in "normalen" Austria-Forum-Seiten durchführen, nicht aber in statischen "Special-Pages", z.B. bei der Administrations-Seite, oder der Biographien-Suche etc. Der Doppelklick kann nur im normalen editierbaren Text zur Suche eingesetzt werden. "Doubleclick and Search" verlinkt praktisch jedes Wort im Wiki-Text mit der Suche.

Standard-Metadata-Feld

Ein wichtiges Merkmal des Austria-Forum ist Qualitätssicherung. Um den Überblick bewahren zu können, welche Dokumente schon auf Qualität überprüft worden sind, sollte ein Metadaten-Feld verwendet werden, dass dem Editier-Team ermöglicht, das Dokument als kontrolliert zu kennzeichnen. Durch das Metadaten-Feld können zusätzliche Informationen in das Dokument inkludiert werden, ohne dass diese im Inhalt aufscheinen.

Der folgende Codeschnippsel sollte an jeder Seite angehängt werden:

```
[{Metadata Suchbegriff=' ' Kontrolle='Nein'}]
```

Für Weitere Erklärungen zum Plugin-Syntax im JSPWiki siehe Abschnitt 6.3.3.

Welchen Zweck erfüllt dieses Plugin also? Lucene, welches im Hintergrund als Suchengine eingesetzt wird, hat eine interne Datenspeicherung die über Felder basiert. Jedes Feld kann einen beliebigen Namen haben, sowie auch einen beliebigen String als Wert. In realen Projekten, wie dem Austria-Forum, ist es nicht gerade ratsam, jedem Dokument in einen eigenen Namespace, also einen eigenes Feld zuzuweisen um dann nach 40.000 Feldnamen zu suchen. Es wäre dadurch kein internes Ranking möglich und die Suche wertlos.

Im Austria-Forum existieren folgende Felder, die von Lucene indiziert werden: contents, author, name. Zu den Feldern in Lucene wurde in Abschnitt 6.3.3 schon eingegangen.

Was wenn diese Felder nicht genug sind? Und aus irgendeinem Grund weitere gebraucht werden, die aber nur händisch eingeben werden können. Aus dem Grund existiert ein Metadata-Plugin, über jenes lässt sich ein beliebiges Feld mit eigenen Namen und Wert definieren .

Aufgrund des Wunsches nach besserem Ranking der Sucherergebnisse führte man ein Suchbegriff-Feld ein, dass der Editor dazu nützen kann, den Inhalt des Dokumentes mit ein paar Keywords zu beschreiben. Außerdem existiert das Kontrolle-Feld wie gesagt dazu, um zu kontrollieren, ob die Qualität des Beitrags schon überprüft wurde.

Da nun über 40.000 Dokumente ohne dieses Plugin existieren, sollte ein Programm jedes Dokument im Austria-Forum dementsprechend editieren.

Es werden aus den von Lucene indizierten Daten die Pfade zu den Dokumenten extrahiert und eine Liste erstellt. Danach wird über diese Aufstellung die Änderungen vorgenommen.

Probleme Als die Testläufe alle wunderbar funktionierten, habe ich in der Nacht zum 12.03.2010 entschieden, es an den echten Daten durchzuführen. Eine Sache wurde leider von mir übersehen, nämlich, dass das JSPWiki das Änderungsdatum von den Dateien auf der Festplatte nimmt. Am unteren Rand jedes Wiki-Dokument findet sich das letzte Änderungsdatum. So kam es dazu, dass nun nach dem Schreiben alle Dokumenten, diese ein Datum vom 12.03.2010 bekamen.

Eine Anpassung im Code hätte das verhindern können, jedoch wird der fortschreitende Editiervorgang den Änderungsdatum-Fehler korrigieren. Die Formatierungen oder andere Eigenschaften bzw. Inhalte der Seiten waren nicht betroffen.

BiographySearch

Im Austria-Forum existiert eine eigene Biographien-Kategore, welche auf die Lebenswege berühmter österreichischer Persönlichkeiten spezialisiert ist. Zu finden ist sie unter http://austria-lexikon.at/af/Wissenssammlungen/Biographien. Eine Suche nach dem Lebensweg dieser berühmten Personen umfasst andere Kriterien als z.B. eine Suche nach einer steirischen Ortschaft oder nach einem historischen Ereignis. Diese Unterschiede führten zu der Idee einer eigenen Suche nach biographischen Merkmalen.

Es wurden mehrere Such-Kriterien aufgestellt:

- Geburtsjahr.
- Geburtsland
- Geburtsort
- Arbeitort
- Arbeitsgebiet
- Todesort
- Todesjahr
- Todesland

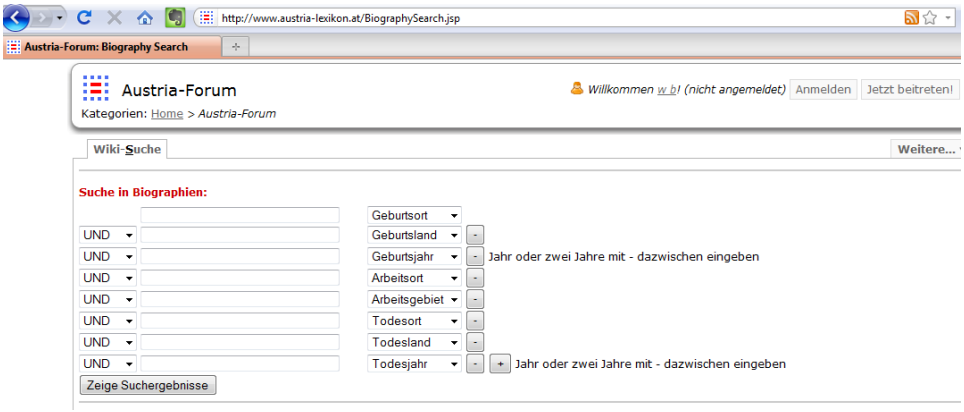

Abbildung 6.6: Erweiterte Suche nach Biographischen Kriterien

Zur Speicherung der biographischen Kriterien lässt sich ein weiteres Metadata-Feld vom Editor einfügen. Ein Beispiel für ein solches Metadata-Feld, wäre zum Beispiel:

```
[{Metadata Geburtsort='Wien' Geburtsland='Österreich' Geburtsjahr='1844'
 Arbeitsgebiete='Physiker, Physik, Naturwissenschafter' Arbeitsorte='Wien,
  Heidelberg, Graz' Todesjahr='1906' Todesort='Triest, Duino'
  Todesland='Österreich, Italien'}]
```

Nach diesen Metadaten sollten sich auch Suchanfragen spezfizieren lassen und so fand sich in der ersten Version des "BiographySearch"-Features unter der speziellen Seite http://austria-lexikon.at/Search.jsp ein Formularfeld, worin es möglich war eine Eingabe zu tätigen. Es existierte dort auch eine Auswahl-Box, in der sich bestimmen ließ, nach welchen Kriterien gesucht werde sollte.

Es ließ sich aber bei jeder Anfrage nur entweder nach Geburtsjahr, Geburtsland etc, also nur nach einem biographischen Merkmal suchen. Damit vermochte man die Suchanfrage nicht weiter zu spezifizieren.

Die Grundidee der Erweiterung baute sich aus dem Gedanken auf, mehrere Felder zur Sucheingabe zu verwenden, um dadurch nach mehreren Kriterien suchen zu können.

Diese spezielle Seite wurde aufgrund der möglichen inhaltlichen Überladung nicht auf der allgemeinen Such-Seite integriert. Stattdessen inkludierte ich die Biographien-Suche in einer speziell dafür vorgesehenen Wiki-Seite namens http://www.austria-lexikon.at/af/BiographySearch. Diese Vorgangsweise erleichtert auch die Verlinkung von "normalen" Wikiseiten zu jener spezifischen Suche.

Im Laufe der Programmierung entstand nicht nur die Möglichkeit eine Mehrfach-Suche nach Kriterien durchzuführen, sondern eine zeitgerechtere Anpassung, die folgende Features enthält:

Dynamische Auswahl der Kriterien Bei jedem Inputfeld, bei der der Term zur Suchanfrage eingetippt werden kann, ist es auch möglich über ein Auswahl-Feld auszuwählen, nach welchem Kriterium gesucht werden soll.

Logische Verknüpfung zwischen den einzelnen Suchanfragen Das heisst, jedes Feld lässt sich untereinander über OR oder AND dynamisch verknüpfen. z.B. Wenn ich eine Anfrage stellen möchte, ob derjenige in Wien ODER in Graz gearbeitet hat, verwende ich die OR-Verknüpfung. Im Falle wenn ich wissen möchte, ob derjenige in Wien UND in Graz seine Arbeitsstätte hatte, verwende ich die AND-Verknüpfung.

Unbegrenzte Erweiterung der Suchanfragenfelder Am rechten Rand findet sich ein Plus oder ein Minus-Button, wobei beim Ersteren ein weiteres Feld am Ende erzeugt wird und beim Minus, das Feld links vom Button damit verschwindet. Der Entfernen-Button soll auch zur besseren Übersicht beitragen, der Benützer kann ihn benützen um z.B. unnötige leere Felder zu entfernen. Wobei der Plus-Button ihm die Möglichkeit gibt immer weitere Details zur Anfrage hinzuzufügen.

Wird die Suchanfrage durchgeführt, bekommt der Benützer eine Auflistung von allen gefundenen Dokumente, die, wenn die Anfrage mehr als zwanzig Seiten im Ergebnis retourniert, zerteilt werden, und über eine fortlaufende Nummerierung erreicht werden können. Siehe auch Abbildung 6.6 für die fertige Implementierung.

Die Erweiterung der Suche nach Biographien sollte als eine Dynamisierung der alten Biographien-Suche verstanden werden, und das vor allem im Hinblick auf die Usability aber auch auf die Mächtigkeit der Anfragen, die nun dynamisch konstruiert werden können.

Erweiterte Navigation

Im Austria-Forum herrschte bevor dieses Feature hinzugefügt wurde eine einfache Browsing-Strategie. Simpel gesagt war der Benutzer abhängig von den manuellen Links und den Plugins, die erst eingebunden werden mussten, die Navigation durch die Dokumente durchzuführen. Beispiele für dementsprechende Plugins wären das CategoryIndexPlugin, welches eine Übersicht darüber gibt, welche Seiten der Kategorie zugehörig sind, oder das ähnlich funktionierende TabbedGlossaryPlugin.

Die speziellen Übersichts-Plugins erfüllen ihren Zweck, zumal der Benutzer eine Zusammenfassung der hierarchisch darunter liegenden Dokumente bekommt.

Was aber, wenn einer der Editoren aufgrund von Qualitätskontrolle, die vielen hundert Seiten in einer Subkategorie durchbrowsen möchte. Der Autor müsste um dies durchführen zu können, jedesmal wenn er eine neue Seite in der Subkategorie erreichen möchte, einen Schritt von der Subkategorie

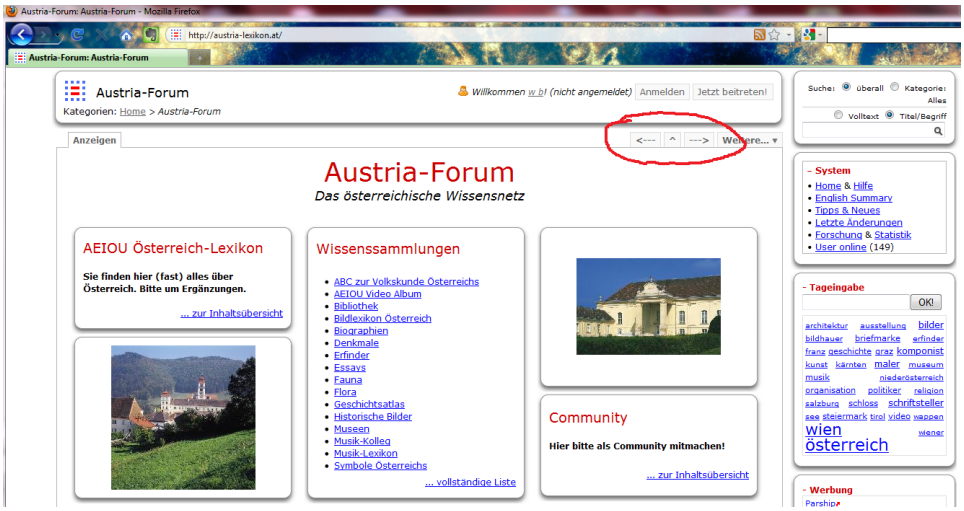

Abbildung 6.7: Innovative Page-Navigation

zurückgehen und über eines der erwähnten Plugins, das nächste noch nicht kontrollierte Dokument suchen. Das macht eine Qualitätskontrolle recht aufwendig.

Dieses Problem existierte schon vor meiner Zeit beim Austria-Forum, und der erste Denkansatz einer Lösung war, die Seiten einer Kategorie über den Wikicode zu verlinken, was äußerst mühsam gewesen wäre. In dem Fall müssten bei jedem Dokument manuell Links gesetzt werden. Mir kam dann die Idee, dieses Navigation-Feature als statischen Baustein ins Austria-Forum zu implementieren, so dass es möglich wird, auf jeder Austria-Forum-Seite in alphabetischer Reihenfolge (aufsteigendend und absteigend) die nächste Page der gleichen Hierarchiebene zu besuchen.

Außerdem verfügt diese erweiterte Navigationleiste die Möglichkeit hierarchisch nach oben zu gehen also zur Überkategorie. In der Abbildung 6.7 findet sich die Navigationsleiste am rechten oberen Rand.

Die Steuerung erfolgt durch Symbole:

Pfeil nach links Gehe zur nächsten Page in der gleichen Kategorie in entgegengesetzer alphabetischer Reihenfolge.

Pfeil nach rechts Gehe zur nächsten Page in der gleichen Kategorie in alphabetischer Reihenfolge.

Zirkumflex Gehe zu der Überkategorie, also zu dem entsprechenden Elternknoten.

Probleme Da die ganze Datenstruktur im Hintergrund baumartig aufgebaut ist, ließ sich das nach einigen Mühen verwirklichen. Die ersten Probleme entstanden bei der Kodierung von Seiten die für das Deutsche spezielle

Abbildung 6.8: Eine Übersicht aller Kommentare, erstellt durch das BloggingPlugin.

Zeichen verwendeten, sowie z.B. Umlaute. Das folgt aus dem Grund, dass Java hauptsächlich in den USA entwickelt wurde, und damit man auch Wörter mit Umlaute am Anfang sortieren vermag, ist eine zustätzliche Implementation nötig.

Kommentar-Funktion

Um die Qualität der Beiträge zu erhöhen, sollte eine Möglichkeit hinzugefügt werden, um Feedback über den geschriebenen Artikel zu geben. Es wurde vor meiner Zeit eine Kommentar-Funktion dafür implementiert, jedoch nicht fertiggestellt. Mein Auftrag war dieses Feature weiterzuentwickeln, so dass es vollständig funktionsfähig wird.

Die fertiggestellte Kommentar-Funktion kann durch den Benützer folgendermaßen genützt werden:

1. Der User klickt auf "Kommentieren", welches in der oberen Steuerleiste der Seite zu finden ist.
2. Ein Editier-Feld öffnet sich mit gleichen Funktionen, wie wenn jemand eine neue Seite erstellen möchte.
3. Der Benützer gibt seine Anmerkung ein und klickt OK, außerdem lässt sich durch den Vorschauknopf eine Preview des Kommentars erstellen.

Wird ein Kommentar erstellt, passiert im Hintergrund folgendes:

1. Dem Autor des Beitrags wird eine Email geschickt, in der ihm bekanntgegeben wird, dass ein Kommentar auf einem seiner Beiträge geschrieben wurde. Der Autor erhält auch einen Link zu dem kommentierten Dokument und einen Link auf die User-Seite des Kommentierenden. Dadurch kann er sich in eine Diskussion mit dem Feedback-Gebenden begeben.

2. Eine Email, wird an die Adresse af@austria-lexikon.org versendet, um dem Admin die Nachricht mitzuteilen, dass kommentiert wurde. Dieser Vorgang passiert rein zur Kontrolle.
3. Eine neue Seite wird unter http://www.austria-lexikon.at/Community/Neue_Kommentare_(Nicht_öffentlich!) erstellt und mit einem Zeitstempel versehen. Diese Seite könnte zum Beispiel `Wilhelm Sebastian_1283507946000` heissen. In dem Beispiel handelt es sich bei `Wilhelm Sebastian` um den Namen des Kommentierenden im System, während die Zahl `1283507946000` einen Zeitstempel darstellt. Dieser Vorgang erfolgt weiterhin aus Kontrollegründe um den Überblick über die Kommentare zu behalten. Wie gesagt findet sich unter der URL http://www.austria-lexikon.at/Community/Neue_Kommentare_(Nicht_öffentlich!) eine Aufstellung aller Kommentare dargestellt durch das BloggingPlugin. Unter Abbildung 6.8 befindet sich ein Screenshot von der Neue-Kommentare-Seite.

Die Kommentar-Funktion ist jedem angemeldeten Benützer erlaubt. Es reicht eine einfache Anmeldung ohne Editierrechte zu besitzen.

Probleme Anfangs änderte das Hinzufügen eines Kommentars auch das Änderungsdatum des Dokuments, da jede Umgestaltung an diesem als neue Version vom System gesehen wurde. Deshalb musste, bevor das Kommentar im Wiki-Text eingefügt wurde, das Datum des Beitrags zwischengespeichert werden und nach der Änderung wieder korrigiert werden.

ExtendedBloggingPlugin

Im JSPWiki existiert ein Blogging-Plugin[13]. Es erstellt eine Liste der Dokumente aus der Subkategorie, relativ dazu wo das Plugin gesetzt wurde, und ordnet sie nach dem Erstellungsdatum.

In der Liste des Blogs werden nur die Pagenamen und das Erstellungsdatum angezeigt. Der User muss also über die Namen beurteilen, ob dieser "Blogeintrag" für ihn geeignet ist oder nicht. Es kann davon ausgegangen werden, dass nicht alle Dokumentnamen aussagekräftig genug sind, um diese Auswahl zu beschleunigen.

Im Sinne der Modernisierung dieses Blogging-Plugins und um den User entgegenzukommen, wurde eine Erweiterung angedacht, die ich dann in dem ExtendedBloggingPlugin verwirklichte.

Das Plugin ermöglicht dem Autor eine Beschreibung zu dem Thema in den Dokumenten, relativ zu der Seite wo ExtendedBloggingPlugin erstellt wurde, hinzuzufügen. Diese Beschreibung beginnt mit dem Zeichen "|". Diese Zeile wird, falls vorhanden bis zu einer gewissen Zeichenanzahl als Beschreibung in die Blogging-Liste eingegliedert. Wenn diese Art von Syntax nicht existiert, werden die ersten Zeichen des Dokuments verwendet.

[13]Für weitere Erklärungen zum Thema Plugins im JSPWiki siehe Abschnitt 6.3.3

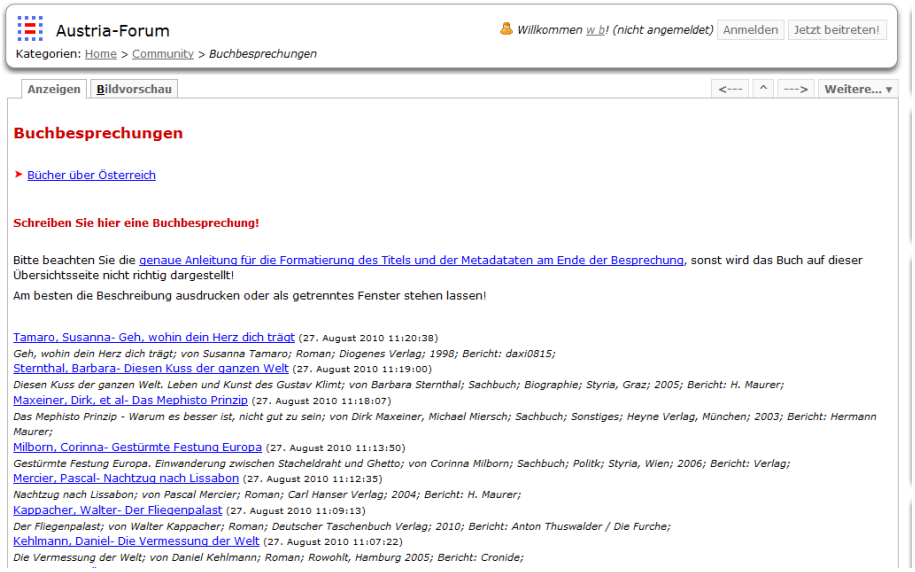

Abbildung 6.9: Buchbesprechung als Blog

Im Communitybereich existiert ein gutes Beispiel zur Verwendung dieses Plugins, in Form des Buchbeschreibungs-Blog zu finden unter der URL http://austria-lexikon.at/af/Community/Buchbesprechungen(Siehe Abbildung 6.9).

Probleme Im Großen und Ganzen ist die Implementierung der Erweiterung recht simpel, jedoch sollte auch eine Fail-Safe-Strategie hinzugefügt werden, falls die Keyword-Zeile nicht existiert. In diesem Fall wird, wie schon berichtet, die ersten Zeichen des Textes wiedergegeben. Im JSP-Wiki können die Texte per Wiki-Markup-Sprache und per HTML abgerufen werden. Würden die ersten Zeichen Formatierungen enthalten, z.B. ein CSS-Styling würden irritierenden Resultate entstehen.

Käme man also auf die Idee die WikiMarkup-Sprache ohne Änderungen von der Seite in die Zusatzinformation einzubinden, wäre der Text voll unerwünschter Zeichen, die nur zur Verwirrung beitragen.

Die Lösung ist durch einen Filter realisiert, der auch im Hauptprojekt benützt wird. Dieser Filter, der nichts mit dem Filter aus der JavaWeb-Welt zu tun hat, macht nichts anderes, als dass er alle unnötigen Wiki-Zeichen, sowie alle möglichen Plugins und Wiki-CSS-Stylings entfernt und den reinen Text wiedergibt.

Abbildung 6.10: Das durch das CommercialPlugin erstellte Werbungs-Feld

CommercialPlugin

Das CommericalPlugin[14] soll Verhindern, dass die Werbung im Austria-Forum Überhand nimmt. Es geht um die Frage inwiefern mehrere Sponseren auf einer Seite dargestellt werden können, ohne dass die Seite dadurch überladen wird.

Die Lösung findet sich in der Idee nicht die vollständige Liste an Sponsoren auf einmal zu zeigen, sondern nur eine gewisse Menge, die der Administrator selbst bestimmen kann.

Die Frage der Sichtbarkeit der Sponsoren wird über einen Zufallsgenerator bestimmt, um etwaige Benachteiligung auszuschließen.

Unter Abbildung 6.3.4 findet sich die Ausgabe des Commercial-Plugins im Austria-Forum.

Es wird im Grunde nur die Austria-Forum-Seite http://www.austria-lexikon.at/af/Werbung eingebunden. Klickt man auf dieser Seite unter `Weitere` auf `Seitenquelltext einblenden` bekommt man zum Beispiel folgenden Code, welcher die Benützung des Plugins veranschaulicht:

```
[{CommercialPlugin
linktext1='Christian Brandstätter' url1='http://www.brandstaetter-verlag.at'
 description1='Der bedeutende Kunstverlag'
linktext2='IMAGNO' url2='http://www.imagno.com'
description2='Große Datenbank historischer Bilder'
linktext3='Technisches Museum Wien' url3='http://www.tmw.at'
description3='100 Jahre jung!'
linktext4='Verlag Ed.Hölzel' url4='http://www.hoelzel.at'
 description4='Traditionsreicher Atlas- und Bildungsverlag'
linktext5='content company:Styria' url5='http://www.styria.com/de'
description5='Großer österr. Medienkonzern'}]
```

[14]Für weitere Erklärungen zum Thema Plugins im JSPWiki siehe Abschnitt 6.3.3

Support-Aufgaben

Im Rahmen meiner Tätigkeit half ich auch einigen Studenten Projekte in und um das Austria-Forum zu verwirklichen. Den größten Beitrag zu diesen Support-Aufgaben lieferte aber die Einführung eines allgemein erreichbaren SVN-Repository, wobei die Idee ursprünglich auf mich und Herbert Mikl zurückgeht, und letzterer die Installation übernahm. In diesem Repository wurde die neueste Version des Austria-Forums hochgeladen und aktuell gehalten.

Meiner Meinung nach sollte jeder Student, der Zugriff auf das Repository bekommt, auch eine funktionsfähige Version erhalten.

Die Daten wurden zuvor per USB-Stick weitergetragen und der neue Mitarbeiter/Student musste an dem JSPWiki-Derivat erst Anpassungen vornehmen, damit der Code in seiner Entwicklungsumgebung ausführbar wurde. Wenn der Student also neue Features entwickelt hat und diese dann in die Online-Version integriert werden sollten, kostete das viel Mühe und Zeit.

Eine Änderung dieses Konzepts war nötig, nicht nur um Zeit einzusparen sondern um auch das Austria-Forum für Weiterentwicklungen zugänglicher zu machen.

Netbeansed Austria-Forum Bei Netbeans handelt es sich um eine IDE[15], die auf Java spezialisiert ist, jedoch ist es auch möglich für Projekte in anderen Sprachen sowie Ruby, PHP etc. Programme zu entwickeln. Abgesehen von den reichlichen Features, die diese Entwicklungsumgebung umfasst, existiert auch ein integriertes SVN-Tool[16].

Mithilfe von Netbeans ist es möglich ein Projekt mit einem SVN-Repository aktuell zu halten, also seine Änderungen mit einem SVN-Server abzugleichen. Über diese SVN-Funktion habe ich schließlich den Code des Austria-Forums auf den von Herbert Mikl installierten SVN-Server hochgeladen, um dadurch den Grundstein für eine zeitgemäßere Weiterverteilung des Projekts zu legen.

Folgende Schritte sind erforderlich um das Austria-Forum in Netbeans auszuführen:

1. Die neueste Version von Netbeans laden installieren und dann starten
2. Klicke auf `Team` (in der oberen Leiste), dann auf `Subversion` und `Checkout`.
3. Unter dem Eingabe-Feld Repository URL: svn://url.zu.unserem.svn eintragen. Unter User und Passwort, den dementsprechend zugewiesenen Benutzer-Name und das Passwort eingeben. Auf "Next" klicken um weiterzugehen.
4. Ein Häkchen bei "Scan for Netbeans Projects after Checkout" setzen und auf "Finish" klicken.

[15] Integrierte Entwicklungsumgebung
[16] http://netbeans.org/

5. Netbeans ladet das Projekt auf die Festplatte und fragt, wenn der Vorgang beendet ist, ob ein neues Projekt für diesen Checkout erstellt werden soll. "OK" klicken und das Projekt ist fertig für die Ausführung und Änderung.

SVN-Konzept Als nächstes stellte sich die Frage nach der Konsistenz des SVN-Repository, an denen mehrere Personen Zugriff haben. Der Code in dem SVN-Repository sollte kompilierbar bleiben, und Kollisionen von unterschiedlichen Versionen sollte vermieden werden.

Das Wechseln auf alte Versionen ist in SVN möglich, zum Beispiel wenn jemand durch fehlerhaften Code die Ausführbarkeit behindert. Wenn viele Änderungen revidiert werden müssen, kann die Übersichtlichkeit aber verloren gehen.

Daraus folgte dann, dass es eine Differenzierung geben sollte, wer nun auf das wirklichen Haupt-Projekt im Subversion-Server zugreifen darf und wer nicht. Ich führte dass sogenannte "Branchen" als Lösung ein. "Branchen" bedeutet im Allgemeinen mit Verzweigungen im SVN zu arbeiten[17].

Das kann man sich anhand eines Beispiels vorstellen. Ein Mitarbeiter einer Einheit einer Firma, die ein bestimmtes Handbuch erstellt, bekommt den Auftrag kleine Änderungen an dem entsprechenden Handbuch vorzunehmen, da eine andere Einheit der Firma eine abgeänderte Version braucht. Der Mitarbeiter kopiert das Handbuch und schreibt die Änderungen hinein. Dadurch erstellt er einen so genannten Branch bzw. Zweig. Im Entwicklungsprozess kann es als eine eigene Linie im Projekt gesehen werden, die unabhängig von der anderen exisitert.

Wenn ein Student einen Branch des Hauptprojekts zugewiesen bekommt, stellt man dadurch sicher, dass er den vollständig ausführbaren Code erhält, jedoch auch, dass seine Änderungen, die er hinzufügt, nicht die Hauptentwicklung stören.

Der Aufbau unseres SVN-Repository kann man sich folgendermaßen vorstellen:

- `svn://url.zu.unserem.svn/AFSVN/trunk` Diese Url führt zum Hauptprojekt. Den Zugriff haben nur interne Mitarbeiter des Austria-Forums.
- `svn://url.zu.unserem.svn/AFSVN/branches/<Name_des_Studenten-Projektes>/` Unter /branches findet sich für jedes Studenten-Projekt ein Unterordner mit eigenem Namen. Den Zugriff erhalten der dazugehörige Student und die Admins.

Zugriffsteuerung auf spezifische Ordner funktioniert im SVN über eine Access Control List, namens authz unter dem "conf"-Ordner im Repository. Ein Beispiel eines solchen Files:

```
[/AFSVN/branches/CategoryIndexPluginExtension]
```

[17] Mehr zu dem Thema findet sich unter http://svnbook.red-bean.com/en/1.1/ch04.html

```
rmoser = rw

[/AFSVN/branches/SearchExtension]
dstein = rw
```

Der Benützer "dstein" darf zum Beispiel auf `/AFSVN/branches/SearchExtension` schreibend und lesend zugreifen[18].

Abgesehen von den ganzen Austria-Forum Erweiterungen, die ich für das Austria-Forum entwickelt habe, war das Verlinkungstool, welches als Nächstes beschrieben wird, weitaus das Aufwendigste.

6.4 Aufbau des Verlinkungsassistenten

Der semi-automatische Verlinkungsassistent wurde in Form eines Tools in das vorher beschriebene Austria-Forum eingegliedert. Zur Erfüllung der Anforderungen ist das Tool praktisch in drei Teile gegliedert. Einer soll die Präsentationsschicht, also den clientseitigen Teil, repräsentieren, der andere die serverseitige Logik des Programmes, siehe Abbildung 6.11. Der dritte Part ist eine Art Integrations bzw. Verknüpfungskomponente.

Die clientseitige Komponente für die Präsentation enthält Abhängigkeiten mit dem JSPWiki, während die andere, die serverseitige Komponente, welche die Hauptberechnungen vornimmt, Abhängigkeiten vermeidet. Es soll dadurch möglich sein, den Serverteil auch in andere Projekte einzugliedern.

Die Fragen, die sich hier stellen, betreffen den genauen Aufbau des Prototyps in Form von Diagrammen und die Integration des Tools in die Wiki-Applikation.

Die Trennung der Komponenten, ihre Aufgaben und ihre Verknüpfungen werden in den nächsten Abschnitten besprochen.

6.5 Serverseitige Komponenten

Die serverseitigen Komponenten wurden auch Themnis von mir benannt und bestehen hauptsächlich aus Java-Klassen. In Abbildung 6.12 findet sich ein Klassendiagramm zur Übersicht über die Themnis.

Der Vorgang zur Verlinkung kann im Grunde durch folgenden Ablaufschritte beschrieben werden:

Input Als Input bekommt die Themnis ein spezifisches Dokument bzw. den internen Link zu einem Dokument, den der Editor (in unserem Fall) auswählt.

Filterung Je nach spezifischer Syntax des Dokuments (in unserem Fall handelt es sich um die JSPWiki-Syntax) filtert die Themnis alles bis

[18] rw bedeutet read-write

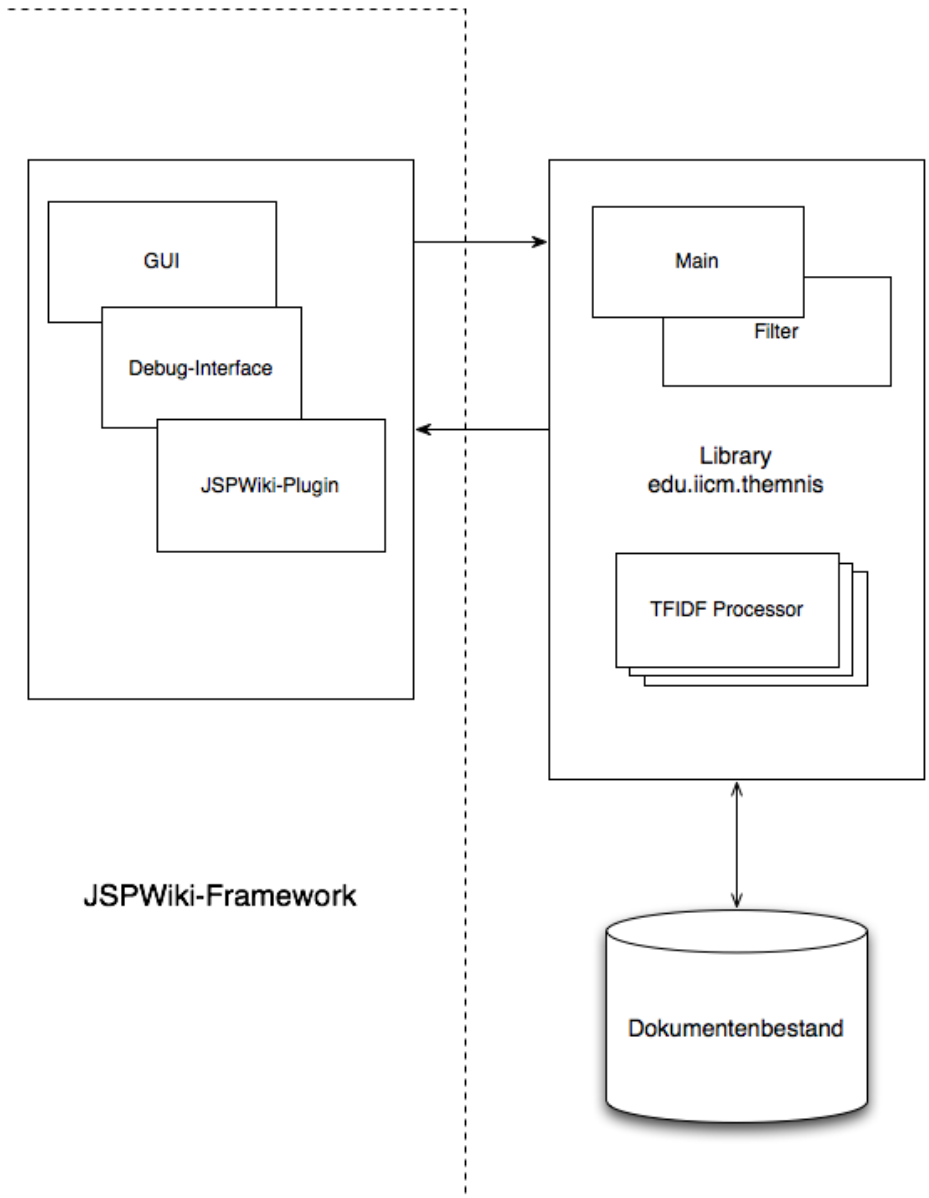

Abbildung 6.11: Allgemeines Diagramm zur Trennung von im System integrierten und getrenntem Teil. Der ins JSPWiki-integrierte Teil besteht aus JSP-Seiten, während die externe Komponente unter edi.iicm.themnis in einem Paket zusammengefasst ist.

auf einzelne sowie zusammengesetzte Substantive aus dem Text. Diese werden dann in einem Array gespeichert, welches für die nächsten Schritte verwendet wird.

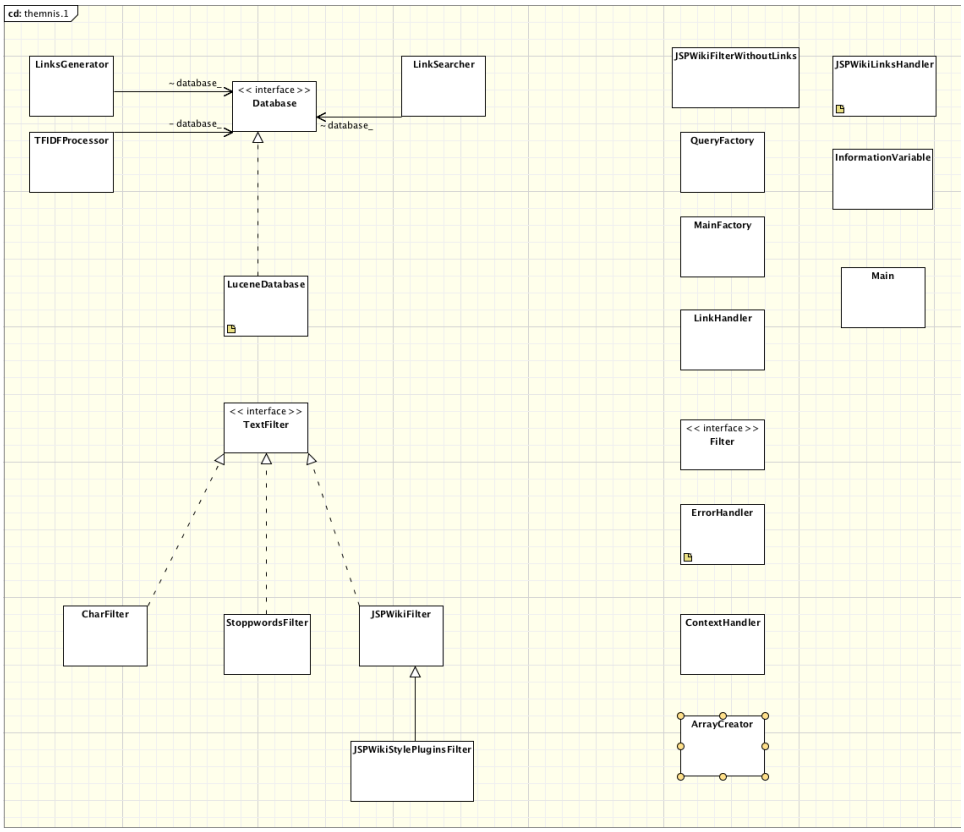

Abbildung 6.12: Ein Klassendiagramm, welches eine Übersicht gibt über die serverseitigen Komponenten des Prototyps.

TF/IDF Über Techniken des Termfrequency / Inverse Term Frequency und mithilfe von Heuristiken[19] werden relevante Wörter für das Dokument bestimmt.

Kontext Für die relevanten Terme wird anschließend eine Art Kontext erstellt. In meinem Fall wird der Satz, in dem das Wort zu finden ist, als solcher benutzt. Es werden nur die Substantive davon gespeichert. Diese werden später in die Suchanfrage eingebaut um die Ergebnisse zu verfeinern.

Querykonstruktion Um nun Linkvorschläge zurückzuliefern müssen Suchanfragen gebaut werden. Die Ergebnisse sollen breitgeträut sein, deshalb werden drei verschiedene Arten von Anfragen losgeschickt. Eine Suchanfrage betrifft nur die Titelzeile, die andere den Inhalt der Dokumente und die dritte bezieht den Kontext des Wortes mit ein.

Suchergebnisse Nachdem die Querys konstruiert wurden, werden sie los-

[19] Siehe Abschnitt 5.3.3 für weitere Informationen

geschickt und die drei im Lucene-Ranking am besten gewerteten Suchergebnisse werden einbezogen.

Linksintegration Die Links die nun pro Wort erhalten wurden, müssen in den Text integriert werden. Dieser Schritt ist je nach Framework, welches als Frontend gewählt wurde, verschieden. Wichtig zu wissen ist, dass die Themnis die Links mithilfe des Linksuggestion-Plugin in das Dokument integriert, welches später unter Abschnitt 6.7 genauer beschrieben wird.

Nachdem nun ein kleiner Überblick über den serverseitigen Ablauf gegeben wurde folgt nun die Beschreibung der Komponenten des serverseitigen Teils im Detail.

6.5.1 Filterung

Filtern heisst nichts anderes, als dass unnötige Wörter aus dem Dokument entfernt werden, um die Übriggebliebenen in ein Array speichern zu können. Die Frage ist berechtigt, wo in den Dokumenten im Austria-Forum unnötige Wörter für die Verlinkung existieren. Die Gründe des Filterns sind vielfältig:

1. JSPWiki besitzt eine eigene Wiki-Syntax, die wohl bei genauerer Beschreibung ein eigenes Kapitel füllen würde. Für fettgedruckten Text zum Beispiel gibt es im JSPWiki einen eigenen Ausdruck: `"__Text fett__"`. Das bedeutet soviel wie **Text fett**. Es existieren in dem Wiki-System Keywords für Schriftgrößenänderungen, Textrahmen, Überschriften, Links, Plugins etc. Es ist wichtig dass diese Zeichenabfolgen nicht mit inhaltswichtigen Wörtern gespeichert werden, da sonst schwere Probleme auftreten können.
2. Aufgrund von Performance werden nur Substantive zur Gewichtung weitergegeben.

Als Ausgabe sollte ein Array mit entsprechenden Substantiven zurückgegeben werden.

Folgende Dateien gehören der Filterung an:

- **CharFilter.java**
- **JSPWikiFilter.java**
- **JSPWikiStylePluginsFilter.java**
- **StoppwordsFilter.java**
- **Textfilter.java**

Unter Abbildung 6.13 findet sich ein Klassendiagramm zu Filterung.

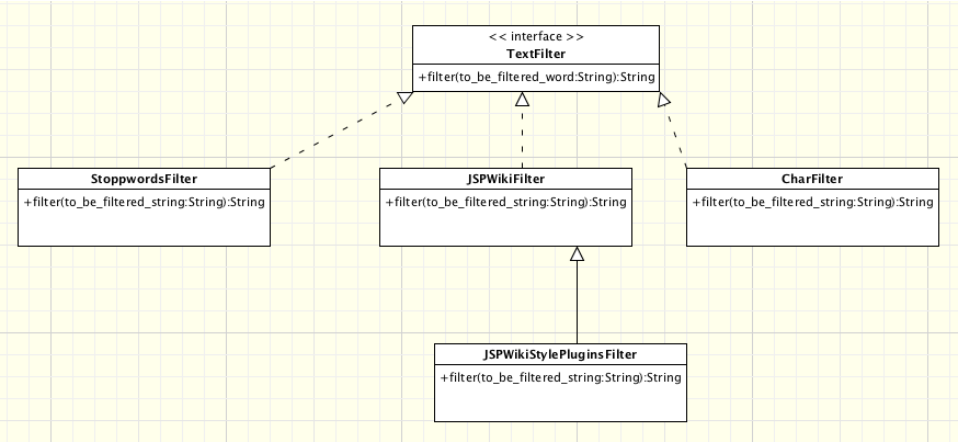

Abbildung 6.13: Klassendiagram über jene Klassen, welcher für die Filterung verantwortlich sind. Textfilter ist die Basisklasse, von der zum Ableiten ist, um einen neuen Filter hinzuzufügen. Dies ermöglicht ein erweiterbares Design.

6.5.2 Gewichtung

Nach der Filterung folgt der Vorgang der Gewichtung der Wörter, welcher über Informations Retrieval-Methoden[20] vorgenommen wird.

Über den TF/IDF[21] werden für den Text wichtige Wörter berechnet. Außerdem wird aus Performance-Gründen vor der Berechnung mittels heuristischer Maßstäbe weitere Kandidaten gefiltert. Die genaue Beschreibung dazu finden sich unter Abschnitt 5.3.3.

Hier folgt eine Aufzählung an Schritten, welche bei der Gewichtung durchgeführt werden:

1. Es wird nachdem der Filter fertig ist, ein Array von einzelnen und zusammen gesetzten Substantiven an die Gewichtungs-Komponente geliefert.
2. Ein originales Dokument wird auch mit übergeben. Das ist zum Beispiel wichtig um Häufigkeiten von Wörtern im Dokument zu bestimmen.
3. Die TFIDF-Heuristiken aus 5.3.3 werden angewandt.
4. Die TFIDF-Formel nach Salton und Buckley, siehe dazu Formel 5.11, wird nun benutzt um relevante Terme zu extrahieren.
5. Das Ergebnis der TFIDF-Berechnung wird mit einem vorher definierten TFIDF-Schwellwert verglichen und gefiltert.
6. Es fallen einige "unwichtige" Wörter durch den Vorgang weg. Die, die

[20] Welche in aller Länge im Kapitel 5 besprochen wird
[21] Siehe Abschnitt 5.3 für mehr Informationen.

bleiben, sind laut Berechnungen für das Dokument und ihrem Inhalt relevant und sollen dadurch verlinkt werden.

Folgende Datei gehört der Gewichtung an:

- **TFIDFProcessor**

6.5.3 Informations-Variablen

Informations-Variablen haben je nach Kontext verschiedene Bedeutungen. In unserem Prototyp soll es als ein Datentyp verwendet werden, der alle wichtigen Informationen die zum Vernetzen notwendig sind speichert.

In diesem Datentyp werden also folgende Variablen gespeichert:

- Wort. (Das Wort selbst.)
- Position. (Die Position des Wortes im Dokument.)
- Links. (Die sogenannten Linkvorschläge)
- Kontext. (Zur Verfeinerung der Linksuche wird der Kontext des Wortes inkludiert.)
- Debug-Informationen (Die ganzen generierten Suchanfragen.)

Nach der Gewichtung weiß man nur, welche Worte für das Dokument relevant sind. Bevor man also nun die ganzen durch den Gewichtungs-Prozess ausgewählten Wörter in Informations-Variablen umwandelt, muss noch die Position im Dokument festgestellt werden. Andernfalls lässt sich kein Kontext des Wortes berechnen und auch kein Verlinkungs-Vorschlag im Text positionieren. Sobald diese Berechnungen vorgenommen wurden kann der Kontext des Wortes mithilfe seiner Position ermittelt werden.

6.5.4 Kontext-Kalkulation

Die Idee hinter der Kontext-Kalkulation ist simpel. Jedes Wort, abhängig wo es im Text steht, hat seine eigene Bedeutung, bzw. seine eigenen Links. Natürlich führt dieses Denkschema zu einer Erhöhung der Arbeitschritte und somit der Rechenzeit, jedoch ist das aufzuwiegen mit der Zufriedenheit des Benützers, denn die die Einbeziehung des Kontexts bedeutet auch eine Verfeinerung der Verlinkungen.

Die Kalkulation erfolgt nach folgendem Schema:

- Der umliegende Satz des entsprechenden Wortes wird aus dem Dokument extrahiert.
- Alles außer die Substantive wird aus dem Satz entfernt.
- Die übriggebliebenen Terme werden zur späteren Verwendung in der entsprechenden Informations-Variable gespeichert.

Der Kontext wird schließlich zur Verbesserung der Suchergebnisse in der Query-Konstruktion (siehe nächsten Abschnitt) verwendet.

Folgende Datei gehört der Kontext-Kalkulation an:

- **ContextHandler**

6.5.5 Query-Konstruktion

Es wird versucht über breitgestreute Suchanfragen dem Editor eine vielfältige Auswahl an passenden Dokumenten zu präsentieren. Der Teil der "Query-Konstruktion" beschäftigt sich mit der Erstellung solcher Anfragen. Dazu werden drei Arten von diesen Querys konstruiert:

Titel-Bezogen Die Anfrage bezieht sich rein auf den Titel der Dokumente im System. In der Lucene-Syntax entspricht das: `title:Suchwort`

Inhalts-Bezogen In dieser Anfrage wird der ganze Inhalt der Dokumente in die Suche involviert. In der Lucene-Syntax entspricht das: `contents:Suchwort`

Kontext-Bezogen Diese Anfrage ist in meinen Augen die Interessanteste, da nicht nur rein das Suchwort einbezogen wird, sondern auch der Kontext des Wortes. Diese werden in die Suchanfrage eingebettet, auch um Mehrdeutigkeiten zu verhindern. In der Lucene-Syntax entspricht das: `contents(Suchwort OR Kontext)`

Diese Suchanfragen sollen breitgestreute Linkvorschläge zurückgeben und somit den Editor optimal unterstützen.

Folgende Datei gehört der Query-Konstruktion an:

- **QueryFactory.java**

6.5.6 Datenbank

Es ist einfach möglich über das Datenbank-Interface andere Datenbanken zur Dokumentensuche einzubinden. Dafür reicht eine Ableitung von `Datenbase.java` und Modifikationen in `MainFactory.java` und `Main.java`.

Es ist jedoch nicht genug irgendeine Datenbank einzubinden. Für das Ergebnis ist es sehr förderlich, wenn diese ein Ranking nach Relevanz der Dokumente erstellen kann.

Für unseren Zweck wurde das ins Austria-Forum integrierte Such-Engine Lucene verwendet.

Lucene

Hier findet sich die praktische Anwendung von Lucene in dem Projekt. Für allgemeine Informationen zu Lucene siehe Abschnitt 6.3.3. Das Lucene-Engine ist im Projekt für zwei verschiedene Aufgaben zuständig:

- Zur Berechnung der Document-Frequency, die für den IDF (Inverse Document Frequency) gebraucht wird. Es wird praktisch nach der Anzahl der Dokumente gesucht, die ein bestimmtes Wort enthalten.
- Bearbeitet die Suchanfragen und liefert dazu relevante Dokumente zur Speicherung in den Informations-Variablen zurück.

Als Voraussetzung zur Durchführung dieser Aufgaben muss ein Verzeichnispfad spezifiziert sein, der auf die Files zeigt, worin Lucene die Dokumente indiziert. Außerdem muss ein IndexSearcher-Objekt verwendet werden, welches die Anfragen durchführt.

Folgende Dateien sind wichtig in dem Kontext:

- **Database.java**
- **LuceneDatabase.java**

6.5.7 Format-Problem

Serverseitig wird rein mit einem Dokument in JSPWiki-Format gearbeitet. Es wäre auch ein Dokument in HTML-Form möglich gewesen. In diesem Abschnitt wird der Denkprozess zu dieser Entscheidung veranschaulicht.

Die essentielle Frage der Formate ist für die serverseitigen Komponenten folgende: Was schickt mir der Client? Was erwartet der Client von mir? In welcher Form nimmt er die Links und bindet sie ein? Diese Diskussion wird im Abschnitt 6.6.3 fortgeführt. Hier sollen die Schwierigkeiten veranschaulicht werden, die anfangs bestanden, als es geplant war HTML serverseitig zu filtern und zu bearbeiten.

Es gibt einige Überlegungen zu der HTML-Lösung. Natürlich lassen sich die Filter soweit erweitern, dass sie erfolgreich HTML-Tags etc. entfernen können. Wenn der clientseitige Teil die Position des Wortes im HTML erwarten würde, um über Javascript die Linkvorschläge einzubinden, wäre dieser Weg nicht unpraktisch.

Doch bei diesem Lösungsweg finden sich Probleme. Die Dokumente beinhalten bekanntlich Plugins, und das wird zum Problem. Denn es ist anhand der HTML-Seite nicht auszumachen ob der Text nun Teil des Plugins ist oder des normalen Dokuments. Das lässt sich nur durch die JSPWiki-Version des Dokuments feststellen.

Ein Plugin ist ein generischer Teil des Dokuments, die Änderungen im System je nach Art des Plugins unterworfen sind. Es hat dadurch keinen Sinn ein Wort, dass aus einem Pluginteil herausgenommen wurde, zu verlinken. Jedoch lässt sich auch ohne komplexe Operationen nicht erkennen, welcher Teil des HTML-Dokuments der Plugin-Teil oder der normale Textteil ist. Aus dem Grund kann der Lösungsweg nicht angewendet werden, ohne grobe Fehler zu erzeugen.

Letztends muss noch hinzugefügt werden, dass prinzipiell die Dokumente in JSPWiki-Syntax auf der Festplatte gespeichert werden. Das heisst um die Links zu speichern braucht man die Position des Wortes in der JSPWiki-Syntax.

Wenn der Editor die Links zu den Wörtern speichern möchte, muss die Position der zu verlinkenden Wörter mitgeschickt werden.

Es wäre möglich das System zu erweitern und beide Positionen eines Wortes zu berechnen. Dazu müsste eine Abgleich-Operation inkludiert werden,

die die Position des Wortes der JSPWiki-Version mit der der HTML-Version vergleicht, was die Komplexität des Programm steigern würde.

Der einfachste Weg, diesen ganzen Problemen nicht zu begegnen, ist, die Positions-Berechnungen rein in einem Dokument in JSPWiki-Syntax durchzuführen. Wenn aber die Architektur des Clients es erforderlich macht, mit HTML-Positionen zu arbeiten, schafft das einen zusätzlichen Aufwand. Dazu wird Abschnitt 6.6.3 mehr gesagt.

6.6 Clientseitige Komponenten

Folgende Komponenten sind für den clientseitigen Teil zuständig:

- Der LinkingClient bestehend aus LinkingClient.jsp und LinkingClient-Content.jsp. Auf diesen JavaServerPages passiert der ganze Ablauf zur Verlinkung. Von der Auswahl bis zur Speicherung. Genauere Detail mit Screenshot etc. findet man unter Abschnitt 6.6.2.
- Das LinkingClientJSEngine ist der eigentliche Teil der Komponenten, der die auschlagebende Arbeit macht. Dass es auf Javascript basiert, ist grundsätzlich eine Vorgabe, da es sich um den clientseitigen Teil einer Webapplikation handelt. Natürlich wären auch andere Möglichkeiten, sowie Java-Applet oder Flash in Frage gekommen, was aber in Folge der Anforderungen die der Client vollbringen muss ein Overkill wäre. Es wird grundsätzlich über die Engine ein Fenster erzeugt, worin der User seine Links auswählen kann. Im Detail wird das LinkingClientJSEngine unter Abschnitt 6.6.2 behandelt.
- LinkingUI stellt den Debug-Client für das System dar. Über diesen lässt sich ermitteln, inwiefern die Themnis auf die Links gekommen ist. Details zum Debug-Client findet sich im nächsten Abschnitt.

Die einzelnen Komponenten folgen nun in detailierterer Beschreibung.

6.6.1 Debug-Interface

Diese JavaServerPage wurde aus dem Grund entwickelt, um die serverseitigen Komponenten ohne das User-Interface zu validieren. Die Entwicklung der Serverkomponenten erfolgte vor dem User-Interface, daraus entstand der Nutzen dieser Debug-UI, wobei dies nicht der einzige Grund dafür ist.

Wie im Abschnitt 5.3.3 beschrieben, werden Heuristiken als Filter für die Ergebnisse verwendet. Die Schwellwerte zu den Heuristiken sind nicht festgelegt, sondern müssen experimentell bestimmt werden. Das Debug-Interface kann verwendet werden diese Heurisitken anzupassen, und somit Ergebnisse zu testen. Außerdem vermag man für gefundene Dokumente weitere Links suchen.

Unter Abbildung 6.14 kann man den Debug-Client betrachten.

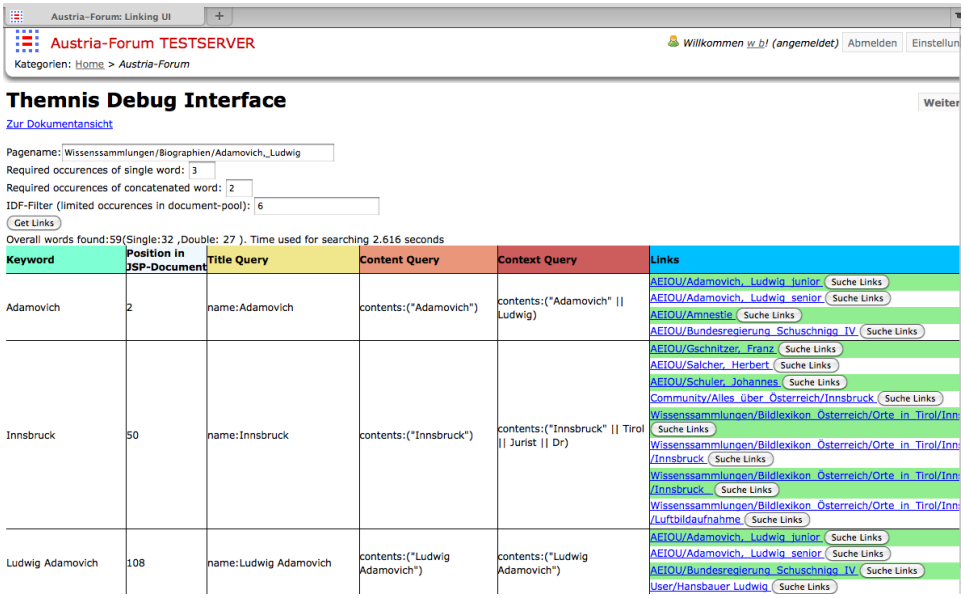

Abbildung 6.14: Das Debug-Interface zu dem Client. Es wird verwendet zum Testen der serverseitigen Komponenten und der Heuristiken die zum Filtern der TFIDF-Ergebnisse verwendet werden

6.6.2 Linking-Client

Der Linking-Client ist praktisch das Frontend für den User, der die Links setzt. Der Benutzer kann im Client folgende Tätigkeiten ausführen (Siehe Abbildung 6.15 für ein Beispiel):

- Er kann eine Auswahl der vorgeschlagenen Wörter treffen, um dafür Links zu erhalten.
- Er kann einen der vorgeschlagenen Links auswählen.
- Er kann diesen temporär speichern, löschen, oder einen anderen auswählen.
- Es ist nur möglich einen Link für ein Wort auszuwählen, jedoch ist es möglich mehrere Links pro Verlinkungs-Session auszuwählen.
- Die ausgewählten Links lassen sich in das Dokument einbetten und somit permanent speichern.

Aus technischer Sicht handelt es sich beim Linking-Client um ein Zusammenspiel aus JavaServerPage und Javascript. Die JavaServerPage wird serverseitig in HTML umgewandelt, und nützt mehrere Plugins sowie Wikitags:

- LinkSuggestionPlugin, siehe genaue Informationen Abschnitt 6.7.1.
- `<wiki:PageExists>` Dies ist notwendig für die interne Prüfung ob dieses Dokument existiert.

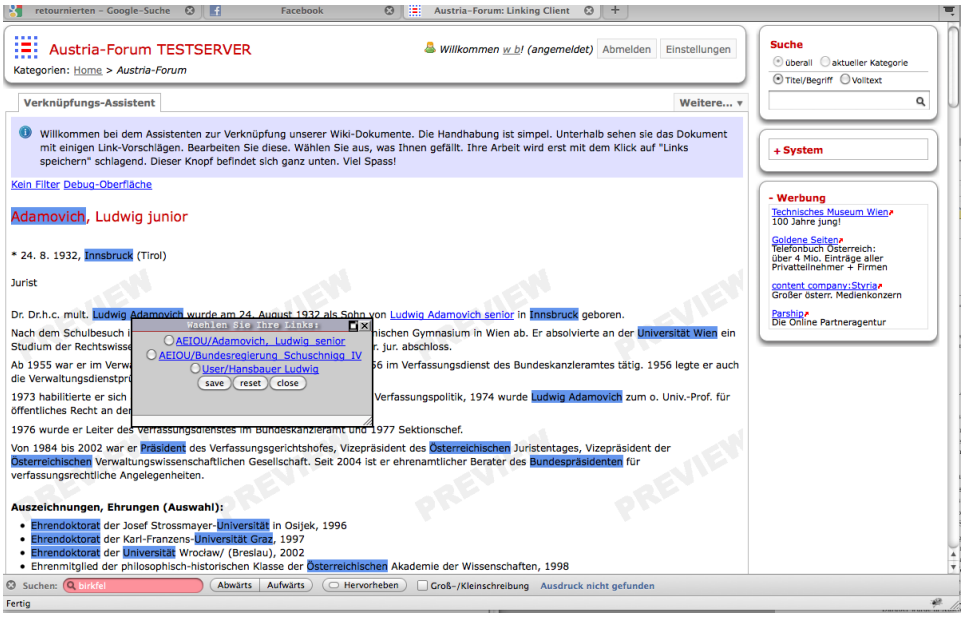

Abbildung 6.15: Der Linking Client in Aktion. Es wurden für die Biographie von dem Juristen Ludwig Adamovich Links angefordert. Die blaugekennzeichneten Wörter zeigen die für das Dokument relevanten Terme. Ein Klick darauf öffnet ein Fenster mit Linkvorschlägen. Es lässt sich darin ein Link auswählen, speichern oder das Fenster wieder schließen. Es existiert am oberen Rand, die Option "kein Filter", welche den User ermöglicht alle Substantive zu verlinken.

- `<wiki:Translate>` Überträgt den vom Server retournierten Text von der JSPWiki-Syntax ins HTML.

Grundsätzlich verfolgt der Client folgenden Plan: Es existiert ein Kontainer-Formular, welches die Links als Parameter zum Server schicken soll. Die ganzen Linkvorschläge enthalten hidden-fields (versteckte Input-Felder), welche den genauen Code zur Verlinkung, die der Server braucht um diese durchzuführen, enthalten. Die versteckten Input-Felder, werden dafür benützt um sie ins Formular-Feld zu speichern. In dem Fall kommt Javascript ins Spiel.

LinkingClientJSEngine

Das Javascript wird im Client für folgende Aufgaben eingesetzt:

- Es erstellt das Javascript-Fenster, und baut die dazu nötigen Elemente ein, die der User braucht um die Verlinkung durchzuführen.
- Es fügt die versteckten Input-Felder (mit dem Verlinkungs-Code) in das Formular-Feld des sogenannten Linkcontainers ein. Damit schafft es einen potentiellen Kanditaten zur Verlinkung.

- Es entfernt potentielle Kanditaten aus dem Linkcontainer, wenn der User seine Meinung ändert.
- Es ändert die Kanditaten für ein spezifisches Wort im Linkcontainer.

Folgende Files zählen zum Verlinkungs-Client:

- **LinkingClient.jsp**
- **LinkingJSEngine.js**

6.6.3 Probleme bei der Client-Entwicklung

Wie schon unter 6.5.7 erwähnt, stellte sich die Frage der Kommunikation, zwischen serverseitigen und clientseitigen Komponenten. Das Verlinkungstool war anfangs folgendermaßen geplant:

- Der Editor soll bei jedem Dokument, per Klick Links anfordern können, die per Ajax in das Projekt eingebunden werden.

Javascript-Styling-Problem

Das Javascript-Styling-Problem verhindert eine einfache Ajax-Lösung für das User-Interface.

Unter JSPWiki nämlich wird es nicht wie üblich CSS[22] überlassen Darstellungseigenschaften der Seite zu definieren, sondern zu diesem Zweck wird Javascript verwendet. Aus dem Grund auch der Name Javascript-Styling-Problem

Über Javascript lässt sich eine Art Styling-Template für JSPWiki erstellen und einbinden. Jedoch dieses Styling verhindert, dass sich das serverseitige vom clientseitigen Dokument gleicht. Es ist auch dadurch zu erkennen, dass die Seite vom Austria-Forum beim ersten Besuch (ohne sie im Cache zu haben) erst formatiert werden muss. Man sieht kurz am Anfang wie die Seite ohne Javascript-Styling aussieht[23].

Durch die unterschiedlichen Dokumentversionen lassen sich schwer eindeutige Linkvorschläge für einen bestimmten Term erstellen, da die Positionen der Wörter im server- und clientseitgen Dokument wie gesagt verschieden sind.

Es gibt mehrere Lösungswege zu dem Javascript-Styling-Problem:

- Serverseitiges Nachvollziehen der Javascript-Änderungen. Erscheint als eine der komplexeren Lösungen. Dadurch müsste Javascript-Code serverseitig ausgeführt werden, bevor die Positionen aus dem Dokument gelesen werden.

[22] Abkürzung für Cascading Stylesheets, Mehr Informationen siehe http://de.selfhtml.org/css/

[23] Dieser Zeitraum war vor meiner Arbeit beim Austria-Forum viel größer. Ich habe Javascript dahingehend optimiert, die Änderungen beim ersten Auftreten des DOM-Trees durchzuführen

- Andere Möglichkeiten der Bestimmung der Position: Wörter zählen und im Dokument nummerieren. Diese Möglichkeit funktioniert teilweise, wenn die HTML-Tags entfernt werden, doch auch dann nur bedingt. Man nehme folgenden Fall: Es existiert ein Plugin welches die Links und Beschreibungen zu Seiten, die hierachisch unter der zu verlinkenden Seite, anzeigt. Man entfernt also dort alle Html-Tags und zählt die Wörter. Ein anderer Editor fügt, während dass das System zählt eine weitere Seite im Unterverzeichnis des verlinkenden Dokuments dazu. Das führt dahin, dass der Plugin-Text vergrößert wird, welches die ganze folgende Aufteilung zerstören kann.
- Rückgängigmachen der Javascript-Änderungen. Das ist eine Option, dich ich ausprobiert habe und wieder verwarf. Die Seite wird dadurch merklich verändert, und man bekommt den Eindruck, dass etwas durch die automatische Verlinkung zerstört wurde.
- Lösungsweg abseits Ajax. Letztlich musste ich mich fragen, ob es überhaupt möglich ist, Ajax zur effektiven Lösung dieses Problem einzusetzen und ob nicht eine andere Möglichkeit besser dazu passen würde.

Ich habe mich für letztere Lösung entschieden, da alle anderen entweder zu komplex oder zu instabil waren. Weiters kam mir die Idee den Vorgang der Verlinkung in Form einer Vorschau abzubilden. Das JSPWiki beinhaltet einen Wiki-Tag namens `<wiki:translate>`, welches dem User als Vorschau beim Editieren dient. Weitere Information zu Wiki-Tags finden sich unter Abschnitt 6.3.3.

6.7 Verknüpfungskomponenten

Die Haupt-Komponente, die für die Verknüpfung des serverseitigen und clientseitigen Teils verantwortlich ist, besteht hauptsächlich aus dem Linksuggestion-Plugin. Im Abschnitt 6.6.3 wurde über die Unmöglichkeit berichtet, die Verlinkung über einen Ajaxclient zu vollführen. Im Grunde führte das zu einem anderen UI-Design, welcher über eine Vorschaufunktion (der über einen Wiki-Tag implementiert wurde siehe Abschnitt 6.3.3) funktionieren soll.

Die Idee zur Vorschau entsprang den Problemen, welcher durch den Unterschied zwischen serverseitigen und clientseitigen Dokumenten entsteht. Ein einfacherer Weg war es letzlich die Position schon serverseitig zu integrieren und das Dokument mit Linkvorschläge per Vorschau zurückzugeben. Um die Integration ins Dokument durchzuführen, wurde die Plugin-Technologie ausgewählt.

Die Kommunikation läuft wie folgt ab:

1. Der Client schickt die URI des Dokuments zu den Serverkomponenten.
2. Der serverseitige Teil geht an die Arbeit, filtert, gewichtet die Worte, und holt die Links.

3. Der Server generiert ein neues Dokument, wo die Linkvorschläge per LinkSuggestionPlugins inkludiert werden.
4. Der Server schickt dem Client das vollständige Dokument, das mit Linkvorschlägen modifiziert wurde. Es besteht dabei noch aus reiner JSPWiki-Syntax.
5. Es wird die Vorschaufunktion angewendet und dadurch zu HTML umgewandelt, und dargestellt.

Zur Integration wird also das LinksuggestionPlugin verwendet. Grundsätzlich hat ein JSPWiki-Plugin folgenden Vorzug für mein Projekt:

- Vereinfachte Lesbarkeit (im Gegensatz zum Einfügen von reinem HTML)
- Einfache Wiederverwendbarkeit des Codes.
- Einfache Erweiterbarkeit dank modularem Plugin-System. Siehe Abschnitt 6.3.3 für mehr Informationen über Plugins.

Das Linksuggestion-Plugin versucht beide, server- sowie clientseitige Komponenten auf einfache und schöne Weise zu verbinden.

6.7.1 LinkSuggestionPlugin

Das LinkSuggestionPlugin dient wie schon erwähnt als Integrations-Komponente zwischen Server und Client. Wie schon im Abschnitt 6.3.3 besprochen, sind Plugins vielfältig einsetzbar, in unserem Fall werden sie zum Einsetzen von generischem HTML+Javascript verwendet.

Ein beispielhafte Anwendung eines LinkSuggestionPlugin wäre:

```
[{LinkSuggestionPlugin name="Brot" link="12@4@Link" link="12@4@AndererLink"}]
```

Der Grund für die merkwürdigen Link-Namen liegt darin, dass dieser Code für den Server eindeutig identifizierbar sein muss. Das System im Linking-Client verlangt diese Notation. Die Zahlen sind getrennt voneinander zu verstehen, als Trennzeichen wird der Klammeraffe verwendet. Die Bedeutung der Zeichen 12@4@link lässt sich zerlegen in:

- An **12**ter Stelle im JSPWiki-Dokument stehend.
- Das Wort ist **4** Buchstaben lang (Im Beispiel "Brot")
- **link** zum vorgeschlagenen Dokument, welches verknüpft werden könnte z.B.: `/Weizenprodukte/Brot`

Mithilfe dieses Codes also ist es möglich eine Verknüpfung serverseitig herzustellen. Diese Lösung sollte einen einfachen Weg bieten, die notwendigen Variablen an den Server zu schicken.

Das Plugin in seiner Form wird dann übersetzt in ein für diese Zwecke angepasstes HTML, welches als Teilfunktionalität (abgesehen vom lokalen Javascript) die Präsentation des Linkvorschlags hat. Im Falle des voherigen Beispiels würde der Code wie folgt aussehen:

```
<span onclick="LinkSuggWindow(this)" style="background-color: cornflowerblue;">
Brot
<input type="hidden" value="12@4@Link" name="link">
<input type="hidden" value="12@4@AndererLink" name="link">
</span>
```

Dieser HTML-Code ist das Pendant zum clientseitigen Javascript, welches den Rest der Arbeit erledigt.

Folgende Datei gehört zum LinkSuggestionPlugin:

- **LinkSuggestionPlugin.java**

6.8 Evaluierung

Das Tool wurde zur Zeit des Schreibens der Diplomarbeit nur teilweise evaluiert.

In einem ersten Versuch testete man das Programm an dem Hypertext-Netzwerk des Austria-Forums. Es sollte eruiert werden, wie vollständig das Programm die Dokumente des Lexikons miteinander verlinkt, um damit den Einfluss des Tools auf den Verlinkungsgrad der Dokumentesammlung zu bestimmen. Zur Identifizierung des Maßes wurde die Vernetzung durch das Tool an dem Dokumentenbestand simuliert. Dann zog man strichprobenartig Dokumentenpaare aus dem Bestand, um Linkpfade zwischen diesen zu ziehen[63].

In weiteren Tests liegt der Fokus auf der Usability des Tools und dem Vergleich zwischen menschlicher und computerisierter Verlinkung.

Kapitel 7

Konklusion

Dieses Projekt sollte einen Beitrag zu Halasz[1] vorgetragenen Problemstellungen bringen, wobei insbesondere das "Dynamische Netzwerk" gemeint ist. Nichtsdestotrotz reicht das nur als ein Beitrag in die Richtung, da die Dynamik nach der Speicherung der Links aufhört, wobei der Gedanke einer Erweiterung in eine Art Verlinkungs-Controller nahe liegt.

Der Wert als Programm, welches Linkvorschläge bietet, kann erst in weiteren Tests ermittelt werden. Abhängig von der Usability und der Zufriedenheit des Users mit den Verlinkungs-Ergebnissen lässt sich in der jetzigen Phase noch keine Aussage tätigen. In jedem Fall bietet die semi-automatische Verlinkung eine Alternative und eine mögliche Zeitersparnis, verglichen zum manuellen Weg, sowie eine Erleichterung des Wartungsaufwands in diesem Informationssystem.

Die Adaptierbarkeit für andere Projekte kann als Stärke gesehen werden, bedenkt man rein den Nutzen, welcher der Algorithmus in Projekten wie Wikipedia haben könnte. Adaptierbarkeit, bzw. Erweiterbarkeit z.B. im Datenbank-Bereich lässt sich aufgrund des Designs leicht durchführen.

7.1 Ausblick

Natürlich sind Möglichkeiten zur Verbesserung gegeben. Wie schon in der Konklusion angedeutet, werden die Links nach der Speicherung statisch in den Text geschrieben. In dem Fall wäre ein Umdenken nötig, was in die Richtung gehen könnte, die Verlinkung durch das ganze Wiki dynamischer zu gestalten. Das System also so umgestalten, dass gar keine statischen Links mehr existieren, sondern jede Verknüpfung vor dem Dokumentenabruf dynamisch berechnet wird. Im Falle der Umsetzung würde das der Web-Applikation sicher Performance kosten, jedoch da die Hardware-Kosten kontinuierlich sinken, ist das kein wirklich unlösbares Problem.

[1] Siehe dazu 2.4

Eine andere Frage ist mehr qualitativer Natur. Es geht darum, inwieweit der Algorithmus im Stande ist, im Gegensatz zum Editor qualitativ gleichwertige bzw. höherwertigere Verlinkungen zu liefern. Dieses Problem benötigt natürlich ausführliche Evaluierung.

Es bestehen natürlich noch einige Optimierungsmöglichkeiten. Das Ranking der Linkvorschläge wäre wahrscheinlich zum Anpassen, zumal die Vorschläge im Moment zufällig gereiht sind. Auch werden im Dokument keine Linkvorschläge für schon vorhandene Links gebracht, was auch noch einer Verbesserung bedarf, da sich Informationen in solchen Informations-Systeme ständig ändern.

Es werden im Moment der Diplomarbeit einige Evaluierungen durchgeführt. Die Weiterentwicklung wird sich nach diesen Ergebnissen richten.

Literaturverzeichnis

[1] Abell, J.: *March 25, 1995: First Wiki Makes Fast Work of Collaboration*, 2010. http://www.wired.com/thisdayintech/2010/03/0325wikiwikiweb-first-wiki/, (besucht am 8.11.2010).

[2] Adamic, L.: *The Small World Web*. In: *Proceedings of the Third European Conference on Research and Advanced Technology for Digital Libraries*, ECDL '99, S. 443–452, London, UK, 1999. Springer-Verlag, ISBN 3-540-66558-7.

[3] Adamic, L. und E. Adar: *How to search a social network*. Social Networks, 27, 2005.

[4] Anderson, P.: *What is Web 2.0.: Ideas, technologies, implications for education*. 2007. http://www.jisc.ac.uk/media/documents/techwatch/tsw0701b.pdf.

[5] Andrews, K., F. Kappe und H. Maurer: *The Hyper-G Network Information System*. Journal of Universal Computer Science, 1:206–220, 1995.

[6] Apache Software Foundation: *Similarity*. http://lucene.apache.org/java/2_4_0/api/org/apache/lucene/search/Similarity.html, (besucht am 02.03.2011).

[7] The Apache Software Foundation: *Apache Lucene - Overview*, 2010. http://lucene.apache.org/java/docs/index.html, (besucht am 04.03.2011).

[8] The Apache Software Foundation: *Apache Tomcat - Welcome!*, 2011. http://tomcat.apache.org/index.html, (besucht am 04.03.2011).

[9] Araujo, S., G. J. Houben und D. Schwabe: *Linkator: enriching web pages by automatically adding dereferenceable semantic annotations*. In: *Proceedings of the 10th international conference on Web engineering*, ICWE'10, S. 355–369, Berlin, Heidelberg, 2010. Springer-Verlag, ISBN 3-642-13910-8, 978-3-642-13910-9.

[10] Ashman, H., A. Garrido und H. Oinas-kukkonen: *Hand-made and Computed Links, Precomputed and Dynamic Links.* In: *In Proceedings of Hypermedia - Information Retrieval - Multimedia '97 (HIM '97*, S. 191–208, 1997.

[11] Austria-Forum: *Statistik*, Aug. 2010. http://www.austria-lexikon.at/af/Statistik, (besucht am 23.09.2010).

[12] Baeza-Yates, R. A. und B. Ribeiro-Neto: *Modern Information Retrieval.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999, ISBN 020139829X.

[13] Barabási, Albert-László und Albert, R.: *Emergence of Scaling in Random Networks.* 286:509–512, 1999.

[14] Basili, R., M. T. Pazienza und F. M. Zanzotto: *Inducing Hyperlinking Rules in Text Collections*, 2003.

[15] Baudoin, S.: *TmP - HTTP*, 2002. http://www2.themanualpage.org/http/, (besucht am 04.03.2011).

[16] Baumgartner, C.: *Wikipedia-Konkurrenz: Infoportal Alexander in Graz gestartet*, Sep. 2006. http://www.computerwelt.at/detailArticle.asp?a=106816&n=1, (besucht am 22.09.2010).

[17] Berners-Lee, T.: *A original proposal of the WWW, HTMLized*, 1989. http://www.w3.org/History/1989/proposal.html, (besucht am 04.03.2011).

[18] Berners-Lee, T.: *Why the //, #, etc?*, 2000. http://www.w3.org/People/Berners-Lee/FAQ#etc, (besucht am 04.03.2011).

[19] Berners-Lee, T., R. Fielding und H. Frystyk: *Hypertext Transfer Protocol – HTTP/1.0*, 1996.

[20] Berners-Lee, T., L. Masinter und M. McCahill: *Uniform Resource Locators (URL)*, 1994.

[21] Bernstein, M.: *Hypertext: concepts, systems and applications.* Kap. An apprentice that discovers hypertext links, S. 212–223. Cambridge University Press, New York, NY, USA, 1992, ISBN 0-521-40517-3.

[22] Blustein, J.: *Automatically generated hypertext versions of scholarly articles and their evaluation.* In: *Proceedings of the eleventh ACM on Hypertext and hypermedia*, HYPERTEXT '00, S. 201–210, New York, NY, USA, 2000. ACM, ISBN 1-58113-227-1.

[23] Bollacker, K. D., S. Lawrence und C. L. Giles: *CiteSeer: an autonous Web agent for automatic retrieval and identification of interesting publications.* In: *Proceedings of the second international conference on Autonomous agents*, AGENTS '98, S. 116–123, New York, NY, USA, 1998. ACM, ISBN 0-89791-983-1.

[24] Borchers, D.: *Vor 20 Jahren: Ein schwer vermittelbarer Vorschlag - und der Anfang des Web*, 2009. http://www.heise.de/newsticker/meldung/Vor-20-Jahren-Ein-schwer-vermittelbarer-Vorschlag-und-der-Anfang-des-Web-205966.html, (besucht am 04.03.2011).

[25] Bush, V.: *As We May Think.* Atlantic Monthly, 176, 1945.

[26] Citizendium: *Citizendium. The Citizen's Compendium*, 2010. http://en.citizendium.org/wiki/Welcome_to_Citizendium, (besucht am 04.03.2011).

[27] Connolly, D.: *HyperText Mark-up Language*, 1992. http://www.w3.org/History/19921103-hypertext/hypertext/WWW/MarkUp/MarkUp.html, (besucht am 04.03.2011).

[28] Connolly, D.: *A little History of the Word Wide Web*, 2000. http://www.w3.org/History.html, (besucht am 04.03.2011).

[29] Csomai, A. und R. Mihalcea: *Linking Documents to Encyclopedic Knowledge.* IEEE Intelligent Systems, 23:34–41, September 2008, ISSN 1541-1672.

[30] Cunningham, W.: *Correspondence on the Etymology of Wiki*, 2005. http://c2.com/doc/etymology.html, (besucht am 04.03.2011).

[31] Cunningham, W.: *Wiki History*, 2010. http://c2.com/cgi/wiki?WikiHistory, (besucht am 04.03.2011).

[32] Dam, A. van: *Hypertext '87: keynote address.* Commun. ACM, 31:887–895, July 1988.

[33] Davis, H. C.: *Hypertext link integrity.* ACM Comput. Surv., 31, December 1999, ISSN 0360-0300.

[34] Deerwester, S., S. T. Dumais, G. W. Furnas, T. K. Landauer und R. Harshman: *Indexing by latent semantic analysis.* Journal of the American Society for Information Science, 41(6):391–407, 1990.

[35] Der Standard: *Wikipedia in Rot-Weiß-Rot*, Aug. 2010. http://derstandard.at/1280984536499/Wikipedia-in-Rot-Weiss-Rot, (besucht am 23.09.2010).

[36] DeRose, S. J.: *FRESS: The File Retrieval and Editing System*, 2003. http://www.derose.net/steve/writings/whitepapers/fress.html, (besucht am 04.03.2011).

[37] Die Presse: *Statistik*, Okt. 2009. http://diepresse.com/home/techscience/internet/514199/index.do, (besucht am 22.07.2010).

[38] Durand, D. G. und S. J. DeRose: *FRESS hypertext system (abstract)*. In: *HYPERTEXT '93: Proceedings of the fifth ACM conference on Hypertext*, S. 240, New York, NY, USA, 1993. ACM, ISBN 0-89791-624-7.

[39] Ebersbach, A., M. Glaser, R. Heigl und A. Warta: *Wiki Kooperation im Web*. Xpert.press, Berlin, Deutschland, 2008, ISBN 3-540-35110-8, 978-3-540-35110-8.

[40] Eidenberger, H.: *Freie Wiki-Systeme im Vergleich*, 2009. http://www.heise.de/open/artikel/JSPWiki-224564.html, (besucht am 04.03.2011).

[41] Ellis, D., J. Furner-Hines und P. Willett: *On the measurement of inter-linker consistency and retrieval effectiveness in hypertext databases*. In: *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '94, S. 51–60, New York, NY, USA, 1994. Springer-Verlag New York, Inc., ISBN 0-387-19889-X.

[42] Engelbart, C.: 2007. http://www.dougengelbart.org/history/engelbart.html#Ref-1, (besucht am 15.03.2010).

[43] Fielding, R., J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach und T. Berners-Lee: *Hypertext Transfer Protocol – HTTP/1.1*, 1999.

[44] Fletcher, D.: *A brief history of Wikipedia*, 2009. http://www.time.com/time/business/article/0,8599,1917002,00.html, (besucht am 04.03.2011).

[45] Foss, C.: *Effective browsing in hypertext*. In: *In Proceedings of the Conference on User-Oriented Context-based Text and Image Handling*, RIAO '88, S. 82–98, Paris, France, 1988. Le centre de Hautes Etudes Internationales d'Informatique Documentaire.

[46] Franke, III, C. H. und N. J. Wahl: *Authoring a hypertext UNIX help manual*. In: *Proceedings of the 1995 ACM 23rd annual conference on Computer science*, CSC '95, S. 238–245, New York, NY, USA, 1995. ACM, ISBN 0-89791-737-5.

[47] Fuhr, N.: *Allgemeine Informationen über die Fachgruppe Information Retrieval*, Jan. 1996. http://www.uni-hildesheim.de/fgir/index.php?option=com_content&task=view&id=14&Itemid=41, (besucht am 23.09.2010).

[48] Fuhr, N.: *Einfuehrung in Information Retrieval Skriptum zur Vorlesung im SS 10*, 2010. http://www.is.informatik.uni-duisburg.de/courses/ir_ss10/folien/skript_1-5.pdf, (besucht am 23.09.2010).

[49] futurezone: *Alexander vs. Wikipedia*, Sep. 2006. http://futurezone.orf.at/stories/134782/, (besucht am 22.09.2010).

[50] Gardner, J., A. Krowne und L. Xiong: *NNexus: an automatic linker for collaborative web-based corpora.* In: *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '09, S. 1152–1155, New York, NY, USA, 2009. ACM, ISBN 978-1-60558-422-5.

[51] Gardner, J. J. und L. Xiong: *Automatic link detection: a sequence labeling approach.* In: *Proceeding of the 18th ACM conference on Information and knowledge management*, CIKM '09, S. 1701–1704, New York, NY, USA, 2009. ACM, ISBN 978-1-60558-512-3.

[52] Garrett, J. J.: *Ajax, a new Approach to Web Applications*, 2005. http://www.adaptivepath.com/ideas/essays/archives/000385.php, (besucht am 8.11.2010).

[53] Gentleware: *Gentleware - model to business: poseidon for uml*, 2010. http://www.gentleware.com/index.php?id=products, (besucht am 04.03.2011).

[54] Gleich, D. F., P. G. Constantine, A. D. Flaxman und A. Gunawardana: *Tracking the random surfer: empirically measured teleportation parameters in PageRank.* In: *Proceedings of the 19th international conference on World wide web*, WWW '10, S. 381–390, New York, NY, USA, 2010. ACM, ISBN 978-1-60558-799-8.

[55] Glushko, R. J.: *Design issues for multi-document hypertexts.* In: *HYPERTEXT '89: Proceedings of the second annual ACM conference on Hypertext*, S. 51–60, New York, NY, USA, 1989. ACM, ISBN 0-89791-339-6.

[56] Green, S. J.: *Building Hypertext Links By Computing Semantic Similarity.* IEEE Trans. on Knowl. and Data Eng., 11:713–730, September 1999, ISSN 1041-4347.

[57] Gronbaek, K., N. O. Bouvin und L. Sloth: *Designing Dexter-based hypermedia services for the World Wide Web.* In: *Proceedings of the eighth ACM conference on Hypertext*, HYPERTEXT '97, S. 146–156, New York, NY, USA, 1997. ACM, ISBN 0-89791-866-5.

[58] Halasz, F. G.: *Seven Issues Revisited*, 1991. http://www2.parc.com/spl/projects/halasz-keynote/intro/.

[59] Halasz, F. G.: *Reflections on Seven Issues:Hypertext in the Era of the Web.* ACM J. Comput. Doc., 25(3):109–114, 2001, ISSN 1527-6805.

[60] Halasz, Frank, G.: *Reflections on NoteCards: seven issues for the next generation of hypermedia systems.* Commun. ACM, 31(7):836–852, 1988, ISSN 0001-0782.

[61] Hall, W., G. Hill und H. Davis: *The microcosm link service.* In: *Proceedings of the fifth ACM conference on Hypertext*, HYPERTEXT '93, S. 256–259, New York, NY, USA, 1993. ACM, ISBN 0-89791-624-7.

[62] Hayes, P. und J. Pepper: *Towards an integrated maintenance advisor.* In: *Proceedings of the second annual ACM conference on Hypertext*, HYPERTEXT '89, S. 119–127, New York, NY, USA, 1989. ACM, ISBN 0-89791-339-6.

[63] Helic, D., I. Hasani-Mavriqi, S. Wilhelm und M. Strohmaier: *The Effects of Navigation Tools on the Navigability of Web-Based Information Systems.* 2011. Eingereicht bei der ICWE 2011.

[64] Iamnitchi, A., M. Ripeanu und I. T. Foster: *Locating Data in (Small-World?) Peer-to-Peer Scientific Collaborations.* In: *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, S. 232–241, London, UK, 2002. Springer-Verlag.

[65] IICM: *aLEXander*, Sep. 2006. http://www.iicm.tugraz.at/ask-lex, (besucht am 22.09.2010).

[66] Jalkanen, J.: *JSPWiki:Main*, 2010. http://www.jspwiki.org/, (besucht am 04.03.2011).

[67] Jin, H., X. Ning und H. Chen: *Efficient search for peer-to-peer information retrieval using semantic small world.* In: *Proceedings of the 15th international conference on World Wide Web*, WWW '06, S. 1003–1004, New York, NY, USA, 2006. ACM.

[68] Jonietz, A.: *Skalenfreie Netze.* Proseminar, Universität Trier, 2006. http://www.jonietz.de/personen/ansgar/Skalenfreie_Netze.pdf.

[69] Kleinberg, J.: *Navigation in a small world.* 406:845, 2000.

[70] Kleinberg, J.: *Complex networks and decentralized search algorithms.* In: *Proceedings of the International Congress of Mathematicians*, Madrid, Spain, 2006. European Mathematical Society. www.cs.cornell.edu/home/kleinber/icm06-swn.pdf.

[71] Kleine Zeitung: *Austria-Forum: Neue Österreich-Enzyklopädie geht online*, Jan. 2009. http://www.kleinezeitung.at/steiermark/graz/graz/

2160401/austria-forum-neue-oesterreich-enzyklopaedie-geht-online.story, (besucht am 23.09.2010).

[72] Knoth, P., J. Novotny und Z. Zdrahal: *Automatic generation of inter-passage links based on semantic similarity.* In: *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, S. 590–598, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[73] Leuf, B. und W. Cunningham: *The Wiki way: quick collaboration on the Web.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001, ISBN 0-201-71499-X.

[74] Liben-Nowell, D., J. Novak, R. Kumar, P. Raghavan und A. Tomkins: *Geographic routing in social networks*, 2005.

[75] Lih, A.: *The Wikipedia Revolution. How a Bunch of Nobodies created the World's Greatest Encyclopedia.* Hyperion, New York, NY, USA, 2009, ISBN 978-1-4013-0371-6.

[76] Liu, L., N. Antonopoulos und S. Mackin: *Information Networking. Towards Ubiquitous Networking and Services.* Kap. Small-World Peer-to-Peer for Resource Discovery, S. 223–233. Springer-Verlag, Berlin, Heidelberg, 2008.

[77] LiveJournal: *LiveJournal.com - Erstelle sofort ein kostenloses Blog/-Journal*, 2008. http://www.livejournal.com, besucht am 04.01.2010.

[78] Manning, C. D., P. Raghavan und H. Schtze: *Introduction to Information Retrieval.* Cambridge University Press, New York, NY, USA, 2008, ISBN 0521865719, 9780521865715.

[79] Maurer, H.: *Grundsätze und Struktur des Austria-Forums*, Jan. 2010. http://austria-lexikon.at/af/Infos_zum_AF/Grunds%C3%A4tze, (besucht am 22.09.2010).

[80] Menczer, F.: *Growing and navigating the small world Web by local content.* 2002.

[81] Milgram, S.: *The Small World Problem.* 2:60–67, 1967.

[82] Milgram, S. und J. Travers: *An Experimental Study of the Small World Problem.* 32:425–443, 1969.

[83] Millard, D. E. und M. Ross: *Web 2.0: hypertext by any other name?* In: *Proceedings of the seventeenth conference on Hypertext and hypermedia*, HYPERTEXT '06, S. 27–30, New York, NY, USA, 2006. ACM, ISBN 1-59593-417-0.

[84] Mylonas, E.: *A commentary on Frank Halasz's Reflections on NoteCards: Seven Issues for the Next Generation of Hypertext Systems.* ACM J. Comput. Doc., 25(3):104–108, 2001, ISSN 1527-6805.

[85] National Center for Supercomputing Applications: *About NCSA Mosaic.* http://www.ncsa.illinois.edu/Projects/mosaic.html, (besucht am 01.03.2011).

[86] Nentwich, C., L. Capra, W. Emmerich und A. Finkelstein: *xlinkit: a consistency checking and smart link generation service.* ACM Trans. Internet Technol., 2:151–185, May 2002, ISSN 1533-5399.

[87] Nielsen, H. F.: *WWW - The Libwww Line Mode Browser.* http://www.w3.org/LineMode/, (besucht am 01.03.2011).

[88] Nielsen, J.: *Multimedia and Hypertext: The Internet and Beyond.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995, ISBN 0-12-518408-5. http://www.useit.com/papers/hypertext-history/.

[89] Oracle: *JavaServer Pages Technology.* http://www.oracle.com/technetwork/java/javaee/jsp/index.html, (besucht am 02.03.2011).

[90] Oracle: *Oracle Technology Network for Java Developers*, 2011. http://www.oracle.com/technetwork/java/index.html, (besucht am 04.03.2011).

[91] O'Reilly, T.: *What is Web 2.0*, 2005. http://oreilly.com/web2/archive/what-is-web-20.html, (besucht am 4.11.2010).

[92] Pam, A.: *Xanadu FAQ*, 2002. http://xanadu.com.au/general/faq.html#2, (besucht am 13.10.2010).

[93] Project Xanadu: *Transcopyright for the Web.* http://www.xanadu.com/tco/, (besucht am 04.03.2011).

[94] Raymond, D. R. und F. W. Tompa: *Hypertext and the new Oxford English Dictionary.* In: *Proceedings of the ACM conference on Hypertext*, HYPERTEXT '87, S. 143–153, New York, NY, USA, 1987. ACM, ISBN 0-89791-340-X.

[95] Rijsbergen, C. van: *Information Retrieval.* Butterworths, London, 2. Aufl., 1979. http://www.dcs.gla.ac.uk/Keith/Preface.html.

[96] Salton, G.: *Automatic Information Organization and Retrieval.* McGraw Hill Text, 1968, ISBN 0070544859.

[97] Salton, G.: *The smart document retrieval project.* In: *Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '91, S. 356–358, New York, NY, USA, 1991. ACM, ISBN 0-89791-448-1.

[98] Salton, G. und C. Buckley: *Term-weighting approaches in automatic text retrieval.* Inf. Process. Manage., 24(5):513–523, 1988, ISSN 0306-4573.

[99] Sanger, L.: *Why Wikipedia Must Jettison Its Anti-Elitism*, 2004. http://www.kuro5hin.org/story/2004/12/30/142458/25, (besucht am 04.03.2011).

[100] Sigurd, B., M. Eeg-Olofsson und J. van de Weijer: *Word Length, Sentence Length and Frequency - Zipf revisited.* Studia Linguistica, 58:37–52, 2004.

[101] Simpson, R., A. Renear, E. Mylonas und A. van Dam: *50 years after "As we may think": the Brown/MIT Vannevar Bush symposium.* interactions, 3(2):47–67, 1996, ISSN 1072-5520.

[102] Stanford University: *Doug Engelbart 1968 Demo.* http://sloan.stanford.edu/mousesite/1968Demo.html, (besucht am 13.10.2010).

[103] Sun Microsystem: *Filter(Java EE 5 SDK).* http://download.oracle.com/javaee/5/api/javax/servlet/Filter.html, (besucht am 02.03.2011).

[104] Sun Microsystem: *HttpServlet(Java EE 5 SDK).* http://download.oracle.com/javaee/5/api/javax/servlet/http/HttpServlet.html, (besucht am 02.03.2011).

[105] thinglink: *Janne Jalkanen joins Thinglink as CTO*, 2010. http://www.thinglinkblog.com/2010/04/12/janne-jalkanen-joins-thinglink-as-cto/, (besucht am 04.03.2011).

[106] Thistlewaite, P.: *Automatic construction and management of large open webs.* Inf. Process. Manage., 33:161–173, March 1997, ISSN 0306-4573.

[107] Trattner, C.: *Vom Austria-Forum zum Wiki-Konzept.* Diplomarbeit, Technische Universität Graz, Jan. 2009. http://www.iicm.tu-graz.ac.at/thesis/DA-Trattner.pdf.

[108] Truran, M., J. Goulding und H. Ashman: *Autonomous authoring tools for hypertext.* ACM Comput. Surv., 39, September 2007, ISSN 0360-0300.

[109] Wardrip-Fruin, N.: *Ted Nelson, Copyright, Literary Machines*, 1981. http://dc-mrg.english.ucsb.edu/conference/CNCSC/multimedia/documents/wardrip-fruin.pdf, (besucht am 04.03.2011).

[110] Watts, D. J. und S. H. Strogatz: *Collective dynamics of small-world networks.* 33:440–442, 1998.

[111] whatis.com: *Hyper-G*, Sep. 2005. http://whatis.techtarget.com/definition/0,,sid9_gci212297,00.html, (besucht am 22.09.2010).

[112] Wikipedia: *Orphan*, Sep. 2010. http://en.wikipedia.org/wiki/Wikipedia:Orphan, (besucht am 20.09.2010).

[113] Wikipedia: *Orphaned Articles*, Sep. 2010. http://en.wikipedia.org/wiki/Category:Orphaned_articles, (besucht am 20.09.2010).

[114] Wikipedia: *Statistics*, 2010. http://en.wikipedia.org/wiki/Special:Statistics, (besucht am 20.09.2010).

[115] Wikipedia: *Verwaiste Seiten*, Sep. 2010. http://de.wikipedia.org/wiki/Spezial:Verwaiste_Seiten, (besucht am 20.09.2010).

[116] Wikipedia: *Verwaiste Seiten*, Jan. 2010. http://en.wikipedia.org/wiki/Wikipedia:ORPHAN, (besucht am 20.09.2010).

[117] Wilkinson, R. und A. F. Smeaton: *Automatic link generation*. ACM Comput. Surv., 31, December 1999, ISSN 0360-0300.

[118] Witten, I. H., M. Gori und T. Numerico: *Web Dragons: Inside the Myths of Search Engine Technology (The Morgan Kaufmann Series in Multimedia and Information Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006, ISBN 0123706092.

[119] Witten, I. H., A. Moffat und T. C. Bell: *Managing gigabytes (2nd ed.): compressing and indexing documents and images*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999, ISBN 1-55860-570-3.

[120] Wolf, G.: *The Curse of Xanadu*, 1994. http://www.wired.com/wired/archive/3.06/xanadu_pr.html, (besucht am 13.10.2010).

[121] World Wide Web Consortium: *What is Hypertext*. http://www.w3.org/WhatIs.html, besucht am 28.02.2011.

[122] Yesilada, Y., S. Bechhofer und B. Horan: *COHSE: dynamic linking of web resources*. Techn. Ber., Mountain View, CA, USA, 2007.