

Technische Universität Kaiserslautern
Fachbereich Informatik
AG Wissensbasierte Systeme
Professor Dr. Andreas Dengel

Colleague–2–Colleague Information Retrieval

Nutzung von Kompetenzeinschätzungen für
die Dokumentrecherche in P2P-Netzwerken

Diplomarbeit

von
Stefan Weisenberger

Betreuer:

Sven Schwarz	Heiko Maus	Ludger van Elst
schwarz@dfki.uni-kl.de	maus@dfki.uni-kl.de	elst@dfki.uni-kl.de

Kaiserslautern, 1. September 2005

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur angefertigt habe.

Kaiserslautern, 1. September 2005

Stefan Weisenberger

Zusammenfassung

Bei ständig wachsenden Mengen an Informationen bekommen leistungsstarke Suchmechanismen eine immer größere Bedeutung. Die Beurteilung von Suchergebnissen ist subjektiv. Sie ist von den Einschätzungen der anfragenden Person abhängig. Diese entscheidet auf Grund ihres Wissens und ihrer Interessen, ob eine Information für sie passend ist. Damit ein Information Retrieval System die Anfrage der Nutzer optimal beantworten kann, ist es nötig, Informationen über deren Interessen zu sammeln und diese in die Anfrageberechnung mit einzubeziehen.

In dieser Arbeit wird ein Ansatz für eine personalisierte Suche in P2P-Netzwerken vorgestellt und umgesetzt. Dabei sollen Einschätzungen der Nutzer bezüglich der Qualität ihrer Daten und der Daten anderer Nutzer zur Verbesserung der Informationssuche verwendet werden. Die Nutzer können durch Veröffentlichung ihrer Kompetenzen und durch Bewertung der Kompetenzen anderer Anwender die Ergebnisse des Retrieval Prozesses auf ihre persönlichen Interessen abstimmen.

Inhaltsverzeichnis

1	Einführung	4
1.1	EPOS	5
1.2	Ziel und Aufbau der Arbeit	6
2	Grundlagen	7
2.1	Information Retrieval	7
2.1.1	IR-Modelle	8
2.1.2	Indexierung	11
2.1.3	Anfrageauswertung	13
2.1.4	Relevanz Feedback	14
2.1.5	Bewertung und Vergleich von IR-Systemen	15
2.2	P2P-Systeme	17
2.2.1	Eigenschaften von P2P-Netzwerken	17
2.2.2	Hybride P2P-Systeme	18
2.2.3	„Echte“ P2P-Systeme	18
2.3	Verteiltes Information Retrieval	19
2.3.1	Semantisches Routing	21
2.3.2	Soziales P2P-Netzwerk	26

3	C2CIR-Einführung	28
3.1	Ausgangsszenario	28
3.2	Anforderungen an das System	31
3.3	Use Cases	31
4	Konzeptionelle Umsetzung	36
4.1	Das C2CIR-Netzwerk	36
4.2	Nutzung von persönlichen Strukturen	37
4.3	Persönliche Sicht auf das Netzwerk	39
4.3.1	Nutzung von Kompetenzen	41
4.3.2	Nutzung von Sucherfahrungen	41
4.4	Personalisierte Suche	43
4.4.1	Anfrageerstellung	43
4.4.2	Inhaltsbasierte Anfrageweiterleitung	44
4.4.3	Dokumentranking	45
4.4.4	Bewertung der Suchergebnisse	47
5	Implementierungsaspekte	49
5.1	Systemaufbau	49
5.1.1	Vorgegebene Funktionalität	50
5.1.2	C2CIR-Peer	52
5.1.3	Competency-Manager	53
5.1.4	Confidence-Manager	54
5.1.5	Das C2CIR-Suchmodul – Query-Manager	59
5.1.6	Umsetzung der Suchanfragen	60
5.2	C2CIR-GUI	67

5.2.1	Aufbau	67
6	Diskussion und Ausblick	74
A	brainFiler	77
A.1	Systembeschreibung	77
A.2	brainFiler-API	79
A.2.1	Klasse Document	80
A.2.2	Klasse Category	80
A.2.3	Klasse DocumentClassificationStore	80
A.2.4	Klasse Query	82
A.2.5	Klasse RankedItemList	82

Kapitel 1

Einführung

Die Verwaltung und Verbreitung von Wissen der Mitarbeiter innerhalb einer Firma bekommen einen immer höheren Stellenwert. Ein großer Teil des Wissens liegt in Form von Textdokumenten vor. Die Mitarbeiter schreiben Dokumente und verwalten sie auf ihrem PC. Dieses Wissen der Mitarbeiter ist ein wichtigster Bestand einer Firma. Wissensmanagementsysteme (kurz: WMS) versuchen das Wissen zu erfassen und für alle zugänglich zu machen.

In letzter Zeit werden in verschiedenen Projekte dezentrale Wissensmanagementlösungen auf der Grundlage von P2P-Netzwerken erforscht. Die dezentrale Struktur hat den Vorteil, dass jeder Anwender seine Daten auf dem Arbeitsplatzrechner selbstständig in eigenen Strukturen (z. B. Verzeichnisstrukturen) organisiert, hat er jederzeit die Kontrolle über seine Informationen. Im Gegensatz zu zentralen Information Retrieval Servern erfordern unstrukturierte Netzwerke intelligente Suchmechanismen zur Informationsgewinnung.

Wenn man annimmt, dass jeder Peer immer von demselben Anwender benutzt wird, kann man die Vernetzung auf Computerebene auch auf die soziale Ebene, als Vernetzung von Menschen, übertragen. Bei der Suche in einem sozialen Netzwerk nutzen die Anwender ihr Hintergrundwissen (z. B. Kenntnis der anderen Nutzer) zum Finden der richtigen Ansprechpartner für die gewünschten Informationen. Überträgt man diesen Ansatz auf die Suche auf Computerebene, so kann auch dies die Ergebnisse verbessern. Damit das System die Suche auf die persönlichen Wünsche des Nutzers abstimmen kann, ist es notwendig, die Interessen und Erfahrungen des Anwenders zu erfassen. Dies kann z. B.

durch Auswertung der lokalen persönlichen Datenstrukturen oder aus Bewertungen von Ergebnissen früherer Anfragen geschehen.

Die Nutzung dieser persönlichen Strukturen zur Informationsgewinnung ist ein Ziel des *EPOS*-Projekts.

1.1 EPOS

Die Arbeit wurde im Rahmen des EPOS-Projektes¹ am DFKI² in Kaiserslautern durchgeführt. Der Begriff EPOS steht für „Evolving Personal to Organizational Knowledge Spaces“.

Das Grobziel des Projektes wird in [6] folgendermaßen beschrieben:

Das Ziel des EPOS-Projektes ist die Evolution des Arbeitsplatzes eines Nutzers mit seinen unterschiedlichen nativen Informationsstrukturen zu seinem persönlichen Wissensraum. In der Kooperation mit anderen Arbeitsplätzen trägt dieser zum organisatorischen Wissensraum bei und somit zum Unternehmensgedächtnis.

Im Zentrum des Projekts steht der persönliche Arbeitsplatzrechner eines Arbeiters. Die Informationen und Strukturen sowie die Arbeitsweisen des Nutzers spiegeln die Aktivitäten, Konzepte und Sichtweisen des Nutzers wider. Durch Beobachtung der Aktivitäten kann man dem Nutzer immer die Informationen bereitstellen, die er gerade benötigt. Da die Arbeiter Teil des Unternehmens sind, können deren lokale Informationen und Strukturen als Grundlage zur Entwicklung von globalen Modellen und Ontologien verwendet werden. Das System soll auch Mechanismen zur Förderung des Wissensaustausches unter den Nutzern basierend auf Kompetenzen, Sucherfahrungen und persönlichen Sichten bereitstellen. Damit die vielfältigen und unterschiedlichen Informationen durch die Nutzer besser erfasst werden können, werden auch flexible und erweiterbare Visualisierungsmöglichkeiten erforscht.

¹<http://www.dfki.uni-kl.de/epos>

²Deutsches Forschungszentrum für Künstliche Intelligenz – <http://www.dfki.de>

Ein Teilbereich des Projektes befasst sich damit, die Effektivität des Information Retrieval Prozesses aus der Interaktion mit dem Nutzer zu verbessern, dieser Ansatz wird *Collaborative Information Retrieval* genannt.

1.2 Ziel und Aufbau der Arbeit

Ziel der Diplomarbeit ist es, ein P2P Information Retrieval System zu entwickeln, das die Anwender beim Wissensaustausch unterstützt. Das System soll den Nutzern nur die Informationen bereitstellen, die für ihn interessant sind. Dazu soll es die lokalen Strukturen auf den PCs der Nutzer sowie deren Kenntnisse und Erfahrungen über andere Anwender erfassen und diese Informationen bei der Berechnung der Suchergebnisse berücksichtigen. Das System arbeitet mit der Annahme, dass das Netzwerk nicht aus anonymen Verbindungen von verschiedenen Computern besteht, sondern dass die Teilnehmer sich untereinander kennen. Ein typisches Anwendungsszenario für unser System ist zum Beispiel die Abteilung einer Firma. Deshalb benennen wir das System *Colleague-to-Colleague-Information-Retrieval-Netzwerk* oder kurz *C2CIR-Netzwerk*.

Die Diplomarbeit besteht aus zwei Teilen: Im theoretischen Teil wird erforscht, wie man eine personalisierte Suche in P2P-Systemen realisieren kann. Der praktische Teil der Arbeit ist die konkrete Umsetzung dieser Vorschläge durch die Implementierung eines Prototyps.

Dieses Dokument enthält die theoretischen Grundlagen für das implementierte System und ist folgendermaßen untergliedert: Kapitel 2 enthält eine Einführung in die Grundlagen des Information Retrieval sowie die Vorstellung mehrerer Projekte, die im Bereich Verteiltes Information Retrieval angesiedelt sind. In Kapitel 3 werden Annahmen und Anforderungen für das zu entwickelnde System beschrieben. Kapitel 4 behandelt die grundlegenden Konzepte unseres Systems. Kapitel 5 beschäftigt sich mit den Ansätzen zur konkreten Implementierung des Systems und stellt die Suchalgorithmen vor. Kapitel 6 fasst das Ergebnis der Arbeit zusammen und gibt einen Ausblick auf mögliche Erweiterungen.

Kapitel 2

Grundlagen

Am Anfang dieser Arbeit wollen wir kurz die wichtigsten Grundlagen für unsere Forschungen vorstellen und folgende Fragen beantworten:

1. Was ist Information Retrieval?
2. Was sind P2P-Netzwerke?
3. Was sind verteilte IR-Systeme?
4. Welche Lösungen für verteiltes IR mit P2P-Netzwerken gibt es?

2.1 Information Retrieval

Information Retrieval (IR) befasst sich mit der Aufbereitung, Speicherung und Recherche von Informationen. Das IR-System soll den Nutzer bei der Suche nach den gewünschten Informationen unterstützen.

IR kann als komplexer Prozess beschrieben werden (siehe Abb. 2.1). Damit das System Anfragen und Dokumente vergleichen kann, müssen diese in geeignete vergleichbare Repräsentationen umgewandelt werden. Die Darstellung hängt von dem zugrunde liegenden IR-Modell (siehe Abschnitt 2.1.1) ab. Die meisten IR-Systeme arbeiten mit der Annahme, dass man den Inhalt eines Dokuments durch eine Teilmenge seiner Wörter repräsentieren kann. Der Extraktionsprozess, der Dokumente auf ihre Schlüsselwörter reduziert, wird *Indexierung*

(siehe Abschnitt 2.1.2) genannt. Um Anfragen zu beantworten, vergleicht das System die Anfragedarstellung mit den einzelnen Dokumentdarstellungen und gibt eine Menge von passenden Dokumenten zurück. Die meisten IR-Systeme bieten dem Anwender die Möglichkeit, die zurückgegebenen Dokumente auf ihre Relevanz zu bewerten, die so genannte *Relevanzrückkopplung* (*relevance feedback*). Die Informationen aus der Bewertung werden zur Verbesserung der Suche genutzt, entweder durch Veränderung der Anfrage oder Verbesserung der Dokumentdarstellung (siehe Abschnitt 2.1.4).

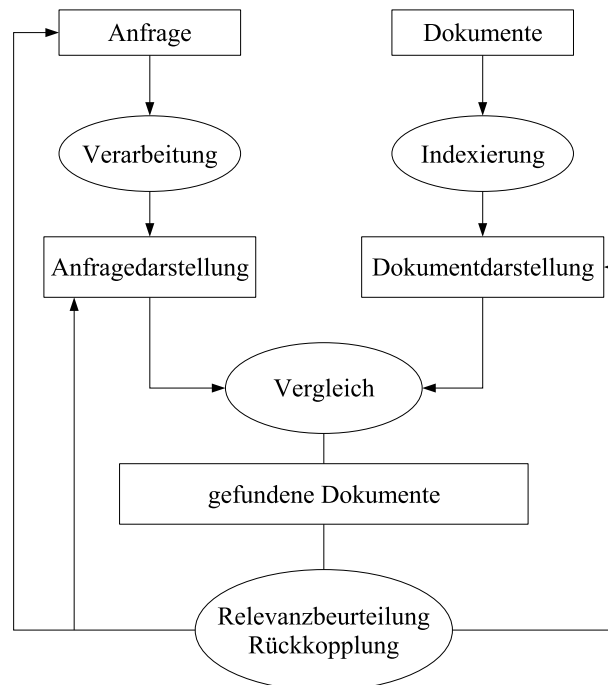


Abbildung 2.1: Der Information Retrieval Prozess nach [16]

2.1.1 IR-Modelle

Im folgenden Abschnitt werden die zwei bekanntesten IR-Modelle, das *Boolesche Retrieval Modell* und das *Vektorraum Retrieval Modell*, vorgestellt. Eine ausführliche Beschreibung und Informationen zu weiteren IR-Modellen, wie dem *Probabilistischen Retrieval Modell*, findet man in der Literatur [1, 7].

Boolesches Retrieval Modell

Das älteste, aber noch sehr häufig verwendete IR-Modell ist das Boolesche Retrieval Modell. Das Modell basiert auf der Mengenlehre und der Booleschen Algebra. Dokumente werden durch eine Anzahl von Attributen repräsentiert, z. B. Schlagworte oder Metadaten, wie *Autor*, *Jahr*, usw. Suchanfragen sind Terme bestehend aus Attributmengen, die durch boolesche Operatoren AND, OR, NOT verknüpft werden. AND bildet die Schnittmenge zweier Mengen, OR die Vereinigung. Der Operator NOT bildet das Komplement einer Menge: Da diese Menge häufig sehr groß ist, wird dieser in den meisten Systemen nur zusammen mit dem AND-Operator als AND NOT verwendet.

Beispiel: Für zwei Terme t_1 und t_2 gilt

- t_1 AND $t_2 \Rightarrow$ alle Dokumente die t_1 und t_2 enthalten
- t_1 OR $t_2 \Rightarrow$ alle Dokumente die t_1 oder t_2 enthalten
- t_1 AND NOT $t_2 \Rightarrow$ alle Dokumente die t_1 aber nicht t_2 enthalten

Aus diesen Termen lassen sich beliebig verschachtelte Anfragen konstruieren. Dadurch kann man unter Kenntnis der richtigen Attribute sehr präzise Anfragen stellen. Bei der Auswertung der booleschen Anfrageterme wird zwischen zwei Zuständen unterschieden: *wahr*, das heißt ein Dokument erfüllt die Anfrage oder *falsch*, das Dokument erfüllt die Anfrage nicht. Dieses Verfahren wird *Exact-Match-Verfahren* genannt. Die Ergebnisse dieser Retrieval Verfahren sind ungeordnete Mengen. Das führt bei einer ungenauen Anfrage mit einer großen Ergebnismenge dazu, dass ein gewünschtes Dokument nur schwer zu finden ist.

Das Boolesche Retrieval Modell hat sich im Bibliothekswesen bewährt, da Bücher meist über wenige einfache Attribute gesucht werden, z. B. über Titel, Autor, Schlagworte. Bei einer Suche, die sich auf den Inhalt eines Dokuments bezieht, lässt sich die Anfrage nicht so einfach stellen, denn die Semantik eines Textes kann man kaum durch wenige Attribute beschreiben. Die Informationssuche unterscheidet sich auch dadurch, dass man Informationen nicht einfach in gut oder schlecht unterteilen kann. Stattdessen versucht man die Dokumente nach ihrer Relevanz zu ordnen, indem man für jedes Dokument einen Wert

berechnet, der die Wahrscheinlichkeit angibt, dass es die Anfrage erfüllt. Diese Strategie wird auch das *Best-Match-Verfahren* genannt. Dieses Verfahren wird z. B. durch das *Vektorraum Retrieval Modell* realisiert.

Vektorraum Retrieval Modell

Das *Vektorraum Retrieval Modell* geht von einem festen Vokabular bestehend aus n verschiedenen Worten aus. Dokumente und Anfragen werden als Vektoren repräsentiert. Ein Dokumentvektor d und Anfragevektor q haben folgende Form:

$$d = \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix} \quad q = \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix}$$

Die Gewichte w_1, \dots, w_n beschreiben den Anteil der einzelnen Terme t_1, \dots, t_n an der Repräsentation des Dokumentinhaltes.

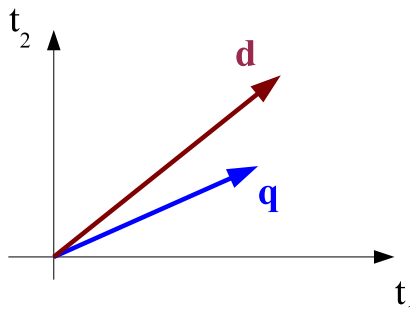


Abbildung 2.2: Anfrage- und Dokumentvektor im Raum

Zur Berechnung der Ähnlichkeit zwischen Dokumenten und einer Anfrage vergleicht man die Lage von Anfragevektor q und den Dokumentvektoren d_j im n -dimensionalen Raum (Abb. 2.2). Ein oft benutztes Vergleichskriterium ist der Winkel zwischen den beiden Vektoren. Ein Ähnlichkeitsmaß, das auf dem

Winkel basiert, ist z. B. das Cosinusmaß:

$$\cos(d_j, q) = \frac{\vec{d} \bullet \vec{q}}{|\vec{d}| \times |\vec{q}|} = \frac{\sum_{i=1}^t w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^t w_{i,j}^2} \times \sqrt{\sum_{i=1}^t w_{i,q}^2}}$$

Da die Gewichte immer positiv zwischen 0 und 1 liegen, hat auch das Cosinusmaß einen Wertebereich zwischen 0 und 1, wobei gilt: je mehr die einzelnen Terme in den Vektoren übereinstimmen desto größer wird der Wert der Funktion. Deshalb lässt sich das Ergebnis zur Relevanzbewertung eines Dokumentes nutzen.

Eine ausführliche Erklärung des Cosinusmaßes und weitere Ähnlichkeitsmaße findet man in der Literatur (siehe [7] S.75 ff).

2.1.2 Indexierung

Zu Beginn des IR-Prozesses müssen die Dokumente im System so repräsentiert werden, dass sie wieder gefunden werden können. Der Prozess der Erstellung der Dokumentdarstellung nennt man *Indexierung*. Bei der Indexierung versucht das System den Dokumentinhalt zu erfassen und ihn durch sprachliche Elemente (Terme) zu repräsentieren. Man unterscheidet zwei Verfahren: die *intellektuelle Indexierung*, bei der Experten eine manuelle Verschlagwortung auf der Grundlage eines genormten Vokabulars durchführen und die *automatische Indexierung*, die die Semantik eines Textes aus dessen Wörtern extrahiert. Wir wollen hier nur die automatische Indexierung genauer vorstellen. Sie besteht aus folgenden Schritten:

1. Syntaxanalyse (*tokenisation*)
2. Entfernung der Stoppworte (*stopword removal*)
3. Stammbildung (*stemming*)
4. Gewichtung der Terme (*term weighting*)

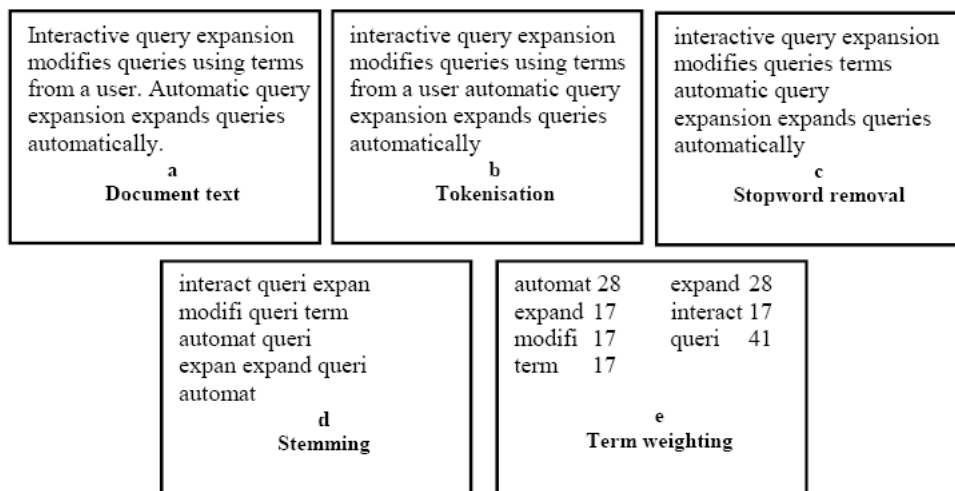


Abbildung 2.3: Indexierung eines Dokuments aus [18]

Die meisten IR-Systeme erfassen die Texte nur syntaktisch, also als reine Menge von Worten. Aus diesem Grund werden im ersten Schritt, der Tokenisation (vgl. Abb. 2.3b), zuerst alle Satz- und Sonderzeichen aus dem Text entfernt. Meistens wird der Text auch noch in Kleinschreibung umgewandelt.

Es gibt in jeder Sprache eine Menge von Wörtern, die in sehr vielen Dokumenten vorkommen, so genannte *Stoppworte*. Zu den Stoppwörtern gehören z. B. Artikel, Präpositionen, aber auch sehr häufig verwendete Verben, wie z. B. *haben* und *ist*. Durch ihr häufiges Auftreten in der Dokumentensammlung sind sie ungeeignet, um zwischen verschiedenen Dokumenten zu differenzieren. Deshalb werden diese Worte nicht in den Index übernommen (siehe Abb. 2.3c).

Nach diesem Prozess besteht die Darstellung aus einer Menge von charakteristischen Wörtern. Die Wörter kommen in einem Text meistens in verschiedenen syntaktischen Formen vor: zum Beispiel Baum, Bäume, Baumes. Da diese Formen eines Wortes inhaltlich die gleichen Dinge und Sachverhalte beschreiben, werden sie zusammengefasst. Dabei kommt ein *Stammbildungsverfahren* (*stemming*) zum Einsatz. Dieses Verfahren basiert auf den Forschungsergebnissen von Kuhlen [13]. Der Index wird dadurch komprimiert, dass Wörter durch Reduzierung auf ihren Wortstamm zusammengefasst werden. Das Ergebnis ist eine Liste für jedes Dokument mit einer Menge von beschreibenden Wortstämmen (vgl. Abb. 2.3d).

Viele IR-Verfahren nehmen zusätzlich eine Gewichtung der Terme vor (vgl. Abb. 2.3e). Bei den Gewichten unterscheidet man zwischen globalen dokumentunabhängigen und lokalen dokumentabhängigen Bewertungen. Für die globale Termbewertung wird die Häufigkeit eines Terms in der gesamten Dokumentsammlung betrachtet. Die *Dokumenthäufigkeit* (*document frequency*) eines Terms ist die Anzahl der Dokumente, welche diesen beinhalten. Für die Unterscheidung von Dokumenten sind vor allem die Terme interessant, die in relativ wenigen aber bestimmten Dokumenten enthalten sind. Ein Term ist also umso wichtiger je weniger er in der Dokumentsammlung vorkommt, deshalb gewichtet man den Term mit der *inversen Dokumenthäufigkeit* (*inverse document frequency - IDF*).

Die Verfahren für die lokale Termbewertung stützen sich meist auf die Annahmen von Luhn [15], der vorschlägt, die Aussagekraft eines Terms an dessen Häufigkeit im Dokument (*term frequency - TF*) zu messen.

Die Bedeutsamkeit eines Terms ist abhängig von globalen und lokalen Gewichtseinflüssen. Für die Unterscheidung zwischen Dokumenten sind vor allem die Terme interessant, die in speziellen Dokumenten häufig vorkommen, im Gesamtdatensatz aber nur selten. Deshalb berechnet man das Gewicht eines Terms als Kombination von Termhäufigkeit und invertierte Dokumenthäufigkeit mit der so genannten *TF-IDF-Formel* (*term frequency inverse document frequency*).

$$TF-IDF = \frac{TF}{IDF}$$

2.1.3 Anfrageauswertung

Kleine Dokumentsammlungen können in angemessener Zeit vollständig durchsucht werden. Bei großen Dokumentmengen ist dies sehr zeitaufwendig. Um einen schnellen Zugriff auf Dokumente mit gleichen Termen zu haben, arbeiten die meisten IR-Systeme mit einem so genannten *invertierten Index*. Das sind Speicherstrukturen, die zu jedem Term die Zugriffspfade auf die Dokumente speichern, in der dieser enthalten ist. Oft enthalten diese Listen weitere Angaben, wie z. B. die Häufigkeiten der Terme (siehe Abb. 2.4).

Term	DF	(Dokumentverweis, TF)
workflow	12	(1,10);(2,3);...
ontology	3	(3,23);...
user	70	(1,12);(2,31);...
⋮	⋮	⋮

Abbildung 2.4: Invertierter Index mit Termgewichten

Durch die Nutzung der invertierten Liste können Anfragen sehr effizient beantwortet werden.

Beispiel: Dieses Beispiel erläutert die Verwendung des invertierten Index unter Benutzung des Booleschen Retrieval Modells. Beim Vektorretrieval werden noch die Gewichte mit die Bewertung eingerechnet. Wir haben drei Dokumente d_1, d_2, d_3 und drei Terme t_1, t_2, t_3 . Dokument d_1 enthält t_1 und t_2 , Dokument d_2 enthält t_2 und t_3 und Dokument d_3 enthält t_1 und t_3 . Aus diesen Informationen können wir folgende Matrix erstellen:

	d_1	d_2	d_3
t_1	1	0	1
t_2	1	1	0
t_3	0	1	1

Um die Anfrage „ t_1 AND NOT t_3 “ zu beantworten, muss das System nur die beiden Zeilen für t_1 und t_3 auswerten, um das Ergebnis d_1 zu erhalten.

2.1.4 Relevanz Feedback

Am Ende des IR-Prozesses bieten viele Systeme dem Nutzer die Möglichkeit durch eine *Relevanzrückkopplung* (*relevance feedback* – *RF*) den Retrieval Prozess zu beeinflussen, indem er die gefundenen Dokumente auf ihre Relevanz bewertet. Die Bewertung wird zur Verbesserung und Präzisierung der Suche benutzt. Vor allem das Vektorraum Modell eignet sich sehr gut für einen solchen iterativen Verbesserungsprozess. Die Informationen aus der Bewertung können in unterschiedlicher Weise benutzt werden: entweder wird damit die

Anfrage präzisiert oder die Dokumentdarstellung verbessert.

Bei der Anfrageverfeinerung wird der Anfragevektor um die charakteristischen Terme der relevanten Dokumente erweitert, z. B. durch die Addition der Gewichte. Dieser neue Vektor wird dann für eine neue Anfrage benutzt. Dieses Verfahren ist vor allem dann sehr nützlich, wenn der Nutzer nicht genau spezifizieren kann, was er sucht. So beginnt er mit einer relativ einfachen Anfrage und verbessert die Suche durch sein Feedback immer weiter, bis er die gewünschten Dokumente gefunden hat.

Eine zweite Möglichkeit ist die Dokumentdarstellung anzupassen. Da ein positives Feedback bedeutet, dass ein Dokument gut zu einer Anfrage passt, wird der Dokumentvektor weiter in Richtung des Anfragevektors verschoben, indem die Terme des Anfragevektors zu dem relevanten Dokumentvektor hinzugefügt werden. Im Gegensatz zur Anfragemodifikation, die nur kurzfristig im Bezug auf die Anfrage wirkt, kann die Dokumentmodifikation die Suchergebnisse dauerhaft verbessern.

2.1.5 Bewertung und Vergleich von IR-Systemen

Es ist im Prinzip nicht möglich IR-Systeme objektiv zu bewerten, denn die Frage, ob ein System gute Ergebnisse zu einer Anfrage liefert, hängt immer von der subjektiven Einschätzung des Anfragenden ab. Um dennoch einen Vergleich durchführen zu können, testet man jedes System mit den gleichen Daten und trifft die Annahme: Wenn ein System unter diesen speziellen Bedingungen gute Ergebnisse liefert, wird es auch im allgemeinen Fall gut funktionieren.

Zur Messung der Qualität von IR-Systemen oder zu deren Vergleich haben sich zwei Maße etabliert, *Recall* und *Precision*:

Der *Recall* oder die Vollständigkeit des Ergebnisses gibt an, welcher Anteil der relevanten Dokumente vom System gefunden wurde.

$$recall = \frac{\# \text{ Relevante Dokumente im Ergebnis}}{\# \text{ alle Relevanten Dokumente im Ergebnis}}$$

Ein Recall von 1 würde bedeuten, dass alle relevanten Dokumente gefunden wurden, 0 bedeutet, kein relevantes Dokument wurde gefunden.

Die *Precision* oder Genauigkeit misst den Anteil der relevanten Dokumente innerhalb der Antwortmenge.

$$precision = \frac{\# \text{ Relevante Dokumente im Ergebnis}}{\# \text{ Dokumente im Ergebnis}}$$

Wenn alle Dokumente im Ergebnis relevant sind, hat die *Precision* den Wert 1. Sollten keine relevanten Dokumente im Ergebnis sein, so ist die *Precision* 0.

Die beiden Maße sind in der Regel gegenläufig. Ein höherer *Recall* kann man z. B. durch Vergrößerung der Antwortmenge erreichen. Dies führt in der Regel dazu, dass im Ergebnis auch mehr irrelevante Dokumente vorkommen, was die *Precision* verschlechtert. Eine Verkleinerung der Antwortmenge erhöht normalerweise die *Precision*, da im vorderen Bereich der Ergebnisliste eher relevante als irrelevante Dokumente vorkommen. Durch die Verkleinerung der Menge werden aber auch die relevanten Dokumente, die weiter hinten im Ergebnis lagen, nicht mehr berücksichtigt, was eine Verschlechterung des *Recall* bewirkt.

Die Berechnung der *Precision* ist noch relativ einfach zu realisieren. Es reicht, wenn ein Experte die Dokumente des Ergebnisses auf ihre Relevanz bewertet. Bei der Berechnung des *Recalls* hat man das Problem, dass die Dokumente des gesamten Datensatzes auf ihre Relevanz beurteilt werden müssten, was nur mit sehr viel Aufwand möglich ist. Deshalb behilft man sich mit fertig zusammengestellten Dokumentsammlungen, den so genannten *Testkollektionen*.

Eine Testkollektion besteht aus einer Dokumentsammlung, einer Liste mit typischen Anfragen und Angaben über die relevanten Dokumente zu jeder Anfrage. Viele Testkollektionen arbeiten mit der bekannten Zusammenstellung *Reuters-21578* [14]. Es handelt sich um eine Sammlung von Nachrichtenartikeln der gleichnamigen Nachrichtenagentur aus dem Jahre 1987. Die Nachrichten wurden von Experten in viele verschiedene Kategorien unterteilt.

2.2 P2P-Systeme

Peer-to-Peer-Systeme (kurz: P2P-System) sind verteilte Computersysteme. Im Gegensatz zur klassischen Client-Server-Architektur des Internets sind bei diesem Konzept alle Computer, genannt *Peer* (engl. peer $\hat{=}$ der Gleiche, der Ebenbürtige), gleichberechtigt, denn jeder übernimmt sowohl die Rolle eines Servers als auch die eines Clients.

P2P-Systeme erlangten große Bekanntheit durch die Nutzung zum Austausch von Multimedia-Dateien (Musik, Filme), auch *Filesharing* genannt. Weitere erfolgreiche Einsatzgebiete von P2P-Netzen sind Kommunikation (Instant Messaging) und verteilte Berechnungen.

2.2.1 Eigenschaften von P2P-Netzwerken

Die P2P-Technik hat im Gegensatz zur Client-Server-Architektur mehrere Vorteile:

- *Skalierbarkeit*: Das Netz kann jederzeit durch Hinzunahme von weiteren Peers vergrößert werden.
- *Robustheit*: Nach dem Ausfall eines oder mehrerer Peers funktioniert das Netz immer noch, lediglich die Daten der Peers fehlen im Netzwerk.
- *Dezentrale Kontrolle*: Die Daten werden auf den jeweiligen Peers verwaltet.
- *Lastenverteilung*: Berechnungen werden auf mehrere Peers verteilt. So kann man freie Kapazitäten besser ausnutzen.

Grundsätzlich unterscheiden zwei Arten von P2P-Systemen:

1. P2P-Systeme mit zentraler Verwaltung (hybride P2P-Systeme)
2. P2P-Systeme ohne zentrale Verwaltung („echte“ P2P-Systeme)

2.2.2 Hybride P2P-Systeme

Bei hybriden P2P-Systemen handelt es sich im Gegensatz zu echten Peer-to-Peer Ansätzen um Netzwerke, die zur Verwaltung auf zentrale Server zurückgreifen. Beispiele für hybride P2P-Systeme sind z.B. Instant Messaging Systeme oder die ersten Filesharing-Programme wie z.B. *Napster*¹.

Napster verfügte über eine dezentrale Datenhaltung und der Austausch der Dateien erfolgte ebenfalls von Peer zu Peer. Doch die Organisation des Netzwerkes und die Anfrageauswertung übernahmen jedoch zentrale Server.

Beim Eintritt ins Netzwerk musste sich jeder Peer bei einem zentralen Server anmelden. Dieser Server speicherte den Online-Status und die IP-Adresse der Peers. Die Suchanfragen wurden zentral von einem Server bearbeitet. Deshalb übertrug jeder Peer eine Liste mit seinen bereitgestellten Dateien, damit diese zentral indexiert werden konnten. Nach einer Anfrage schickte der Server eine Liste mit Peers zurück, die die gesuchte Datei bereitstellen. Zum Download der Datei wählte der anfragende Peer anschließend den leistungsstärksten Peer, abhängig von seiner Latenzzeit aus.

2.2.3 „Echte“ P2P-Systeme

Die „echten“ P2P-Systeme bestehen aus einem Netzwerk gleichberechtigter Peers, die sich ohne zentrale Kontrolle selbst organisieren.

Die meisten P2P-Systeme arbeiten nach dem Prinzip von *Gnutella*, dem ersten System dieser Art. Da es keinen zentralen Einstiegspunkt ins Netzwerk gibt, muss ein Peer zuerst die IP-Adresse eines Peers, der schon im Netzwerk angeschlossen ist, kennen. Aufgrund der hohen Fluktuation im Netz müssen die Peers ständig ihre Peerliste aktualisieren. Die Kommunikation der Peers im Gnutellanetzwerk arbeitet ohne feste Verbindungen zwischen den Peers rein auf Nachrichtenbasis. Die Nachrichten werden über das Netzwerk durch das so genannte *Fluten (flooding)* weiterverteilt. Zur Auffindung der gewünschten Dokumente im Netz leitet jeder Peer die Anfragen an möglichst viele Teilnehmer weiter. Damit die Nachrichten nicht endlos im Netzwerk verteilt werden und ein Peer eine Nachricht nicht mehrfach beantwortet, bekommt jede Nachricht

¹<http://www.napster.com>

eine eindeutige *ID* sowie einen so genannten *Time-To-Live-Zähler*. Dieser gibt an, über wie viele Stationen die Nachricht maximal verteilt werden darf. Das Protokoll ist recht einfach gehalten und besteht aus nur 5 verschiedenen Nachrichten: *ping* für die Suche von anderen Peers, *pong* als Antwort auf *ping* mit der IP-Adresse eines Peers, *query* zum Absetzen von Suchanfrage und *query-hit* als Antwort auf eine *query* mit dem Ergebnis. Zur Kommunikation hinter einer Firewall gibt es noch die *push* Nachricht. Im Gegensatz zu Napster kann das System nicht gewährleisten, dass ein existierendes Dokument auch gefunden wird. Da der Flooding-Algorithmus nur einen Teil des Netzwerks erreichen kann.

2.3 Verteiltes Information Retrieval

Zentrale IR Lösungen haben einige Nachteile im Bezug auf Performance, Skalierbarkeit, Ausfallsicherheit und Kosten. Bei immer größer werdenden Dokumentmengen steigen die Kosten, die für die Speicherung und Wartung aufgebracht werden müssen, schnell an. Da die Anfragen nur von diesem einen Server beantwortet werden können, wird dieser Server schnell zum *Single-Point-Of-Failure*.

Das *distributed Information Retrieval (dIR)* versucht diese Probleme durch Verteilung des IR-Systems auf mehrere Systeme zu vermeiden. Erste Ansätze, wie z. B. *Harvest*², lösen die Zentralität damit auf, dass sie mehrere IR-Systeme verwenden. Die Anfragen der Clients werden an speziellen Anfrageserver, genannt *Brokern*, gestellt, die dann den richtigen IR-Server anfragen, die Ergebnisse zusammenfassen und anschließend diese den Clients übermitteln [11]. Da dieses System immer noch mit zentralen Komponenten arbeitet, ist hier die Ausfallsicherheit nicht gewährt.

Zur Zeit wird in einigen Forschungsprojekten an Lösungen für verteilte IR-Systeme auf Basis von P2P-Netzwerken gearbeitet. Die Vorteile dieses Ansatzes sind u. a. geringe Wartung, da die Nutzer des Peers ihre Dokumente selbst verwalten, bessere Ausfallsicherheit, bei Ausfall eines Peers gehen dem Netzwerk nur dessen Dokumente verloren, sowie bessere Ausnutzung der Ressour-

²<https://harvest.sourceforge.net>

cen. Die P2P-Architektur hat den Vorteil, dass man relativ schnell und einfach eine Vernetzung zwischen PCs und somit den Zugriff auf Daten der anderen Anwender realisieren kann. Durch das Verbinden mit weiteren Peers kann man das Netzwerk leicht erweitern. Der P2P-Ansatz kommt dem Modell der „natürlichen“ sozialen Verbindungen zwischen Menschen sehr nahe. Die Personen haben nicht zu allen anderen Menschen Kontakt, sondern jeder kennt nur einen Teilbereich. Man kann aber über einen Bekannten Verbindungen zu weiteren Personen aufbauen. Milgram [17] hat in seiner Studie sogar gezeigt, dass man normalerweise über durchschnittlich sechs Personen jede Person in der Welt erreichen kann, das sogenannte *Small World Phänomen*.

Im Gegensatz zu zentralen IR-Systemen bringt das Wiederfinden von Informationen in P2P Information Retrieval Systemen einige Schwierigkeiten mit sich. Die Verfahren des klassischen Information Retrievals lassen sich nicht so einfach auf die P2P-Netzwerke übertragen. Denn Verfahren, wie zum Beispiel die bekannte *TF-IDF*-Formel, benötigen Informationen über die gesamte Dokumentmenge. Außerdem eignet sich das *Floodingverfahren*, wie es in P2P-Netzwerken angewandt wird, nur schlecht für die Suche in verteilten IR-Systemen, da es relativ ineffizient alle Peers abfragt. Viele *dIR*-Systeme versuchen deshalb zuerst die Zahl der anzufragenden Peers auf die Menge einzugrenzen, die mit hoher Wahrscheinlichkeit passende Dokumente bereithalten. Das Information Retrieval in verteilten IR-Systemen läuft laut [1] in folgenden Schritten:

1. Auswahl der richtigen Peers für die Anfrage
2. Verteilung der Anfrage an die Peers
3. Parallele Ausführung der Anfrage auf den Peers
4. Zusammenfassung der verteilten Ergebnisse zum Endergebnis

Erste Ansätze zur Peersuche, wie Chord [22], benutzen verteilte Hashlisten (distributed hash table - DHT) zum Auffinden der richtigen Peers. Wenn ein Dokument ins System eingefügt werden soll, wird zu diesem zuerst ein eindeutiger Schlüssel in Form eines Hashwertes berechnet. Die Dateien werden abhängig von dem Wert auf einem bestimmten Peer gelagert. Die Suche nach einem bestimmten Dokument wird durch Nachschlagen in der DHT durchgeführt. Dieses Verfahren eignet sich nur für eine genaue Dateisuche und ist

für ein Information Retrieval ungeeignet. Für die inhaltliche Suche wird eine Strategie benutzt, die *semantisches Routing* genannt wird. Eine Anfrage wird abhängig von ihrem Thema nur an solche Peers weitergeleitet, die potenziell gute Antworten liefern können. Im nächsten Abschnitt betrachten wir mehrere Ansätze für diese Strategie.

2.3.1 Semantisches Routing

Unter *semantischem Routing* verstehen wir die gezielte Weiterleitung von Anfragen zu den Peers, die am wahrscheinlichsten relevante Dokumente liefern können.

Einige Projekte, die diese Weiterleitungsstrategie umgesetzt haben, werden hier vorgestellt. *PlanetP* baut mittels einem speziellen Verteilalgorithmus einen globalen Index auf, *NeuroGrid* arbeitet mit schlüsselwortbasierten Routingtabellen. Das Projekt *Bibster* arbeitet mit festen Ontologien und realisiert eine kompetenzabhängige Weiterleitungsstrategie. Abschließend wollen wir den Ansatz von Simon Schenk präsentieren, der sich in seiner Diplomarbeit [19] mit der Suche in sozialen Netzwerken befasst hat.

PlanetP

PlanetP³ [5] ist eine P2P-Infrastruktur zur inhaltsbasierten Suche. Das Projekt zielt auf Gemeinschaften, die eine große Zahl an Dokumenten austauschen wollen. Das System arbeitet mit der Vorgabe, dass die Dokumente durch Text beschrieben sind. Textdokumente werden selbst indexiert. Andere Dokumente, wie diverse Multimediaformate, müssen mit XML-Textstücken beschrieben werden, um indexiert werden zu können.

Das System arbeitet mit zwei Indizes: einem *lokalen Index*, der die lokalen Dokumente beschreibt und einem *globalen Index*, der den Inhalt der bekannten Peers beschreibt. Der globale Index ist eine Tabelle mit Informationen zu jedem bekannten Peer (siehe Tabelle 2.1): ein systemweit eindeutiger Bezeichner, die IP-Adresse, der Status (online, offline) und eine Liste mit den Schlüsselworten aus dessen lokalen Dokumenten. Um Speicherplatz zu sparen, wird für

³<http://www.planetP.com>

die Liste eine kompakte Datenstruktur verwendet, nämlich Bloomfilter [2]. Ein Bloomfilter speichert Objekte auf der Grundlage von Hashfunktionen und bietet dadurch eine effiziente Auswertung der Zugehörigkeit eines Objektes zu der Liste. Die Komprimierung wird dadurch erreicht, dass bei dem Test ein gewisser Prozentsatz von Objekten fälschlicherweise der Liste zugeordnet wird, obwohl diese nicht Teil der Menge sind, sogenannte *false positives*.

Name	Status	IP	Bloomfilter
Stefan	online	192.168.0.1	<i>BF</i> [...]
Sven	offline	192.168.0.2	<i>BF</i> [...]
...

Tabelle 2.1: PlanetP: globaler Index

Da sich die Daten in einem P2P-Netzwerk häufig ändern, ist es besonders wichtig, den Index auf allen Peers aktuell zu halten. Deshalb müssen die Peers ständig Informationen über ihren Datensatz untereinander austauschen. Zur Aktualisierung der Indizes auf den Peers nutzt *PlanetP* das „Gossiping-Verfahren“. Bei dem Verfahren schicken die Peers in gewissen Abständen Informationen zu Änderungen des globalen Indexes an einen zufällig ausgewählten Peer. Dieser überprüft, ob die Änderungen neu sind und schickt in diesem Fall Information an einen anderen Peer weiter.

Die Suche nach Dokumenten läuft in zwei Schritten ab:

1. Peer Ranking: Berechne eine Liste mit Peers geordnet nach der Relevanz.
2. Anfrage der relevantesten Peers: Jeder angefragte Peer berechnet eine geordnete Liste mit Dokumenten passend zur Anfrage. Der fragende Peer sammelt dann alle Ergebnisse, fügt sie zusammen und ordnet sie nochmals neu.

Zur Berechnung der Peerliste dient der globale Index. Jede Anfrage besteht aus einer Menge von Termen. Das System vergleicht die Suchterme mit den Schlüsselwörtern jedes Peers und berechnet daraus einen Rang. Zur Berechnung wurde ein neues Maß eingeführt - die *Inverse Peer Frequency (IPF)*. Ähnlich wie bei der *IDF* (vgl. Abschnitt 2.1.2) werden hier Terme, die auf

allen Peers vorkommen, geringer gewichtet als Terme, die nur speziell auf wenigen Peers vorkommen, da diese nicht geeignet sind, um zwischen verschiedenen Peers zu unterscheiden. Die IPF für einen Term t wird folgendermaßen berechnet:

$$IPF_t = \log\left(1 + \frac{N}{N_t}\right)$$

wobei N die Anzahl aller Peers im Netzwerk ist und N_t die Anzahl der Peers, die den Term enthalten. Für jede Anfrage wird mit Hilfe des globalen Indexes und der IPF eine Liste mit den relevanten Peers berechnet. Die Anfragen werden anschließend an die Peers in der Liste der Reihe nach von oben nach unten weitergeleitet. Ein angefragter Peer berechnet mit Hilfe seines lokalen Indexes sowie der globalen Liste die besten Dokumente und liefert dem anfragenden Peer eine gerankte Liste mit den Dokument-URLs zurück. Dieser vereinigt die Ergebnisse und fragt den nächsten Peer an. Das Verfahren stoppt, wenn es keines der Dokumente von einem angefragten Peer unter die K besten Dokumente in der Ergebnisliste schafft.

NeuroGrid

*NeuroGrid*⁴ ist eine dezentrale lernfähige Suchmaschine für P2P-Systeme.

Das NeuroGrid-Netzwerk besteht aus einem Netz von Instanzen der NeuroGrid-Software, auch Knoten genannt. Die Knoten verwalten Dokumente bzw. Bookmarks und stellen Suchmechanismen bereit. Die Daten sollen nicht wie z. B. in Dateisystemen hierarchisch geordnet werden, sondern in einer netzähnlichen Form, indem ein Nutzer Dokumente durch Zuweisung von beliebig vielen Schlüsselworten zu Konzepten zusammenfasst.

Das System arbeitet auf der Annahme, dass Dokumente mittels einer Menge von Termen gesucht werden. Für die Suche des richtigen Peers benutzt NeuroGrid folgende sich ergänzenden Komponenten: semantisches Routing und einen Lernmechanismus.

⁴<http://www.neurogrid.de>

Die Suche in NeuroGrid ist laut [12] in drei Abschnitte unterteilt:

1. finde heraus, was du suchst
2. finde heraus, wo es sich befindet
3. download des Objekts

Das System arbeitet ähnlich wie PlanetP mit zwei getrennten Indizes. Ein lokaler Index speichert Verweise von Schlüsselwörtern auf die lokalen Dokumente und ein Peerindex führt Verweise von Schlüsselwörtern auf andere Knoten. Das Ranking von Peers und Dokumenten ist von der Anzahl der mit der Anfrage übereinstimmenden Schlüsselwörter abhängig.

Eine Anfrage wird von einem Knoten folgendermaßen beantwortet: Zuerst werden durch den lokalen Dokumentindex Dokumente auf der eigenen Datenbasis gesucht. Anschließend wird eine Liste mit den besten Knoten berechnet und die Anfrage an diese weitergeleitet. Die angefragten Knoten liefern aus ihrer Datenbasis die besten Dokumente zurück und leiten ihrerseits die Anfrage weiter.

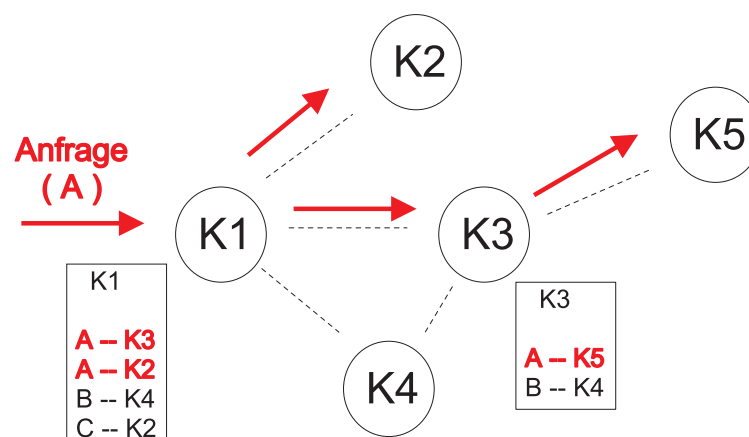


Abbildung 2.5: NeuroGrid: Weiterleitung der Suchanfragen

Zum Verhindern von Schleifen und unendlicher Weiterleitung der Anfragen werden die gleichen Mechanismen wie im Gnutellanetzwerk genutzt: *Time-To-Live (TTL)* Zähler zur Begrenzung der Weiterleitung und eindeutige Bezeichner der Knoten zur Verhinderung doppelter Anfrageauswertung.

Das System lernt aus den erfolgreichen Suchprozessen. Nach einer erfolgreich gekennzeichneten Suche, d. h. wenn das gewünschte Dokument gefunden wurde, werden die Indizes aktualisiert. Der anfragende Knoten speichert einen direkten Verweis zu dem Knoten, in dessen Datenbasis das gewünschte Dokument gespeichert ist, indem er die Schlüsselworte der Anfrage mit dem Knoten verbindet.

Bibster

Das Projekt *Bibster*⁵ untersucht, wie man in P2P-Netzwerken die richtigen Peers zur Beantwortung einer Anfrage finden kann [4, 8, 9, 10].

Bibster leitet die Anfragen in Abhängigkeit der Kompetenzen der Peers weiter. Die Kompetenz eines Peers wird hier als Abstraktion des Inhalts der Datenbasis gesehen. Damit die einzelnen Peers die Kompetenzen austauschen können, wird dabei eine gemeinsame Ontologie zwischen den Peers benutzt. Eine Ontologie ist eine systemweit gültige Kategorisierung. Die Kompetenz eines Peers wird durch eine Auswahl dieser Kategorien dargestellt. Diese Kompetenzen dienen als Grundlage für die inhaltsbasierte Weiterleitung. Deshalb muss jeder Peer seine Kompetenzen dem Netzwerk bekannt geben.

Das Thema einer Suche wird ebenso durch Kategorien der Ontologie ausgedrückt, somit sind Kompetenzen und Anfrage vergleichbar. Erhält ein Peer eine Anfrage, so berechnet er mit seinem Wissen über die Kompetenzen der bekannten Peers eine geordnete Liste von Peers abhängig von der Ähnlichkeit zwischen Kompetenz und Anfrage (siehe Abb. 2.6). Die Anfrage wird also nur an einen Teil der Netzwerkteilnehmer weitergeleitet. Durch das Wissen über die Kompetenzen der Peers bildet sich eine *semantische Topologie*, d. h. die Verbindungen zwischen den Peers sind abhängig von dem Inhalt einer Anfrage.

Das System wurde für die Suche im Bibliothekswesen entwickelt. Das P2P-System ist spezialisiert auf Wissenschaftler, die beim Verwalten, Suchen und Publizieren von bibliografischen Metadaten (z. B. Bibtex) unterstützt werden. Die Beschränkung auf den bibliografischen Bereich hat den Vorteil, dass es

⁵<http://bibster.semanticweb.org>

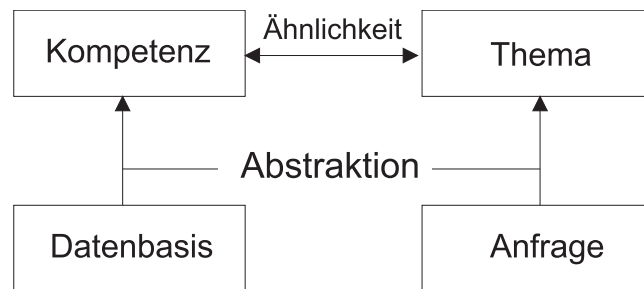


Abbildung 2.6: Bibster: kompetenzabhängiger Vergleich

z. B. mit der *ACM Topic-Hierarchie*⁶ schon leistungsfähige Ontologien gibt.

Dokumente, die in das System eingeführt werden sollen, müssen in die Ontologie eingeordnet werden. Eine Anfrage wird durch Auswahl von Themen aus den Ontologien aufgebaut. Jeder Peer publiziert dem Netzwerk seine fachliche Kompetenz durch eine Menge von ACM-Themen. Auch die Anfrage wird durch ACM-Themen ausgedrückt. Zur Berechnung der Ähnlichkeit zwischen Kompetenz und Anfrage untersucht das System, wie nahe die Themen in der ACM-Hierarchie zusammenliegen.

2.3.2 Soziales P2P-Netzwerk

Simon Schenk widmete sich in seiner Diplomarbeit [19, 20] der Frage: Wie kann ein P2P-System die Nutzer beim Ausbau ihres sozialen Netzes unterstützen?

Sein Ansatz geht davon aus, dass jeder Peer nur einen Benutzer hat. Durch die Interaktion des Benutzers mit seinem Peer können immer mehr Informationen (Interessensgebiet, Wissen über andere Peers) über seinen Besitzer gesammelt werden. Es findet praktisch eine Abbildung vom realen sozialen Netzwerk auf das P2P-Netzwerk statt. Er spricht deshalb auch vom *sozialen P2P-Netzwerk*.

Die System von Schenk soll die Gruppierung von Peers mit ähnlichen Interessen fördern. Laut Schenk beeinflusst die Gruppierung auf der technischen Ebene auch die Bildung von Interessengruppen im sozialen Netzwerk positiv.

⁶<http://www.acm.org/class/1998/>

Eine wichtige Frage zur Bildung einer Informationsgemeinschaft ist: Wie können Peers mit ähnlichen Interessen gefunden werden? Schenk führt dazu drei Faktoren ein:

1. *Relevanz*
2. *Reputation*
3. *Kompetenz*

Als *Relevanz* eines Peers bezüglich eines Themas oder einer Anfrage bezeichnet er die Wahrscheinlichkeit, dass dieser gute Dokumente liefern kann. Die *Relevanz* ist ein subjektives Maß, d. h. sie ist abhängig von dem Kontext und Wissen des Anfragenden.

Der zweite Faktor ist die *Reputation* eines Peers. Sie resultiert aus den Erfahrungen des Nutzers aus früheren Anfragen. Nach einer erfolgreichen Kommunikation mit dem Peer wird die Reputation erhöht, ansonsten wird der Wert verringert.

Das arithmetische Mittel der beiden oben genannten Faktoren nennt Schenk die *Kompetenz* des Peers.

Anders als bei der normalen Gnutella-Suche werden die Anfragen nicht an alle bekannten Peers weitergeleitet, sondern es findet eine Einschränkung des Suchraums statt. Jeder Peer berechnet zur Anfrage eine geordnete Liste mit kompetenten Peers und nur die n besten Peers werden angefragt.

Nachdem wir die Grundlagen und mehrere Ansätze aus bestehenden Systemen vorgestellt und analysiert haben, können wir jetzt im nächsten Kapitel, die Anforderungen für das zu entwickelnde System festlegen.

Kapitel 3

C2CIR-Einführung

3.1 Ausgangsszenario

In einer Abteilung arbeitet eine überschaubare Menge von Kollegen, die damit beschäftigt sind, Wissen zu produzieren (schreiben, lesen, publizieren). Jeder hat eine Menge von Dokumenten auf seinem Rechner. Die Wissenschaftler verwalten die Dokumente mit einem lokal installierten Dokumentmanagementsystem *brainFiler* (siehe Anhang A), welches ihnen erlaubt, zur besseren Navigation Dokumente zu kategorisieren. Um das System für den Nutzer einfach und flexibel zu halten, gibt es keine vorgegebene Ontologie, sondern jeder kann nach seinen Vorstellungen Kategorien anlegen und Dokumente in diese Kategorien einfügen.

Damit die Kollegen Dokumente untereinander austauschen können, sind die Rechner durch ein P2P-Netzwerk verbunden und jeder kann seine Dokumente für andere Kollegen zur Ansicht freigeben. Da die Kollegen zum Teil an gleichen Projekten arbeiten, dort aber andere Teilaufgaben verrichten, gibt es Überschneidungen bei den Interessen, d. h. die Kollegen haben Dokumente, die für andere der Abteilung ebenfalls interessant sind.

Die Datenmengen auf den einzelnen Peers werden mit der Zeit immer größer. Das hat zur Folge, dass es immer schwieriger und zeitaufwendiger wird, manuell nach Information zu suchen. Die Freiheit der Nutzer, ihre Dokumente in eigenen Strukturen ordnen und die Konzepte dabei beliebig benennen zu

können, führt dazu, dass die Datenbasis mit steigender Zahl der Dokumente immer unübersichtlicher wird. Bei einer großen Anzahl von verschachtelten Konzepten ist es schon für den Ersteller kaum noch möglich seinen eigenen Datensatz zu überblicken. Für andere Anwender, die auf seine Daten zugreifen wollen, ist dies noch schwieriger. Diese persönlichen Strukturen enthalten viel Hintergrundwissen des Erstellers, z. B. die Namensgebung oder die Art, wie Dokumente zusammengefasst werden. Dieses Wissen haben fremde Zugreifer nicht. Darum führt ein manuelles Durchsuchen von fremden Strukturen nur selten zum Erfolg. Die Anwender müssen also durch ein Suchsystem unterstützt werden.

Bei der Informationssuche, wie sie in unserem Kollegennetzwerk stattfindet, spielt die Herkunft der Information eine wichtige Rolle. Bei der Suche nach Dokumenten zu einem bestimmten Arbeitsthema sind z. B. die Dokumente der Teammitglieder besonders interessant. Für andere Themen gibt es Spezialisten im Kollegenkreis. Die Informationssuche zu deren Spezialgebieten sollte dann an diese Experten weitergeleitet werden.

Die Benutzer können aus ihrem Wissen über Teammitglieder und Experten einschätzen, welche der Kollegen zu einem gesuchten Thema Informationen liefern können. Sie können abhängig vom Thema der Suche beurteilen, dass die Informationen von manchen Kollegen wichtiger sind als andere. Die persönliche Sicht auf das Netzwerk ist also von Thema zu Thema unterschiedlich. In [8] werden solche themenabhängige Verbindungen auch *semantische Topologie* des Netzwerkes genannt.

Beispiel: Stefan befasst sich unter anderem mit den Themen „A“ und „B“. Aus seiner Erfahrung weiß er, dass für „A“ vor allem Ludger und Heiko interessante Dokumente besitzen, zu „B“ liegen auf Svens Peer passende Dokumente. Bei der Informationssuche zu Thema „A“ würde er also nur Ludger und Heiko anfragen, zu Thema „B“ Sven. In der physikalischen Topologie sind in diesem Beispiel die Peers zwar voll vernetzt, die semantische Topologie zu einer Anfrage hingegen nicht. Abbildung 3.1 zeigt die semantische Topologie aus der Sicht von Stefans Peer.

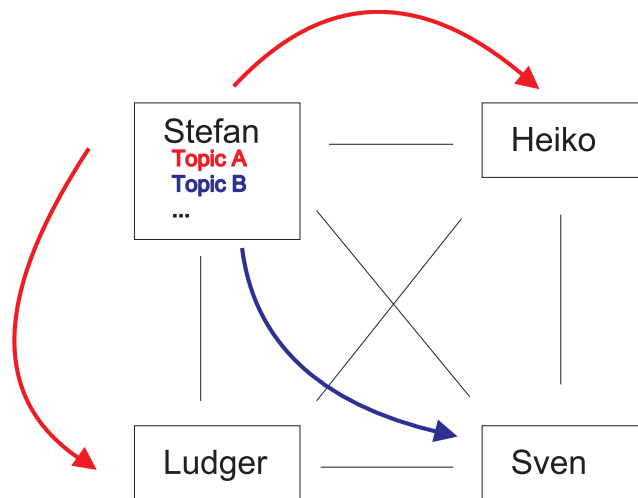


Abbildung 3.1: Semantische Topologie

Innerhalb des Kollegenkreises kennen sich alle Nutzer, weshalb jeder selbst entscheiden kann, welchen Kollegen er zu einem Thema anfragt. Viele der Anwender arbeiten aber nicht nur mit den Personen innerhalb der Abteilung zusammen, sondern sie sind auch Mitglied von Firmen übergreifenden Gruppen. Das heißt, sie kennen weitere Personen mit ähnlichen Interessen und Wissen außerhalb des Kollegenkreises. Diese *Homophily*¹ kann man für die Informationssuche ausnutzen [21], in dem Anfragen über bekannte Personen an weitere Informanten weitergeleitet werden.

¹Homophily bezeichnet die Tendenz, dass sich Ähnliches mit Ähnlichen verbindet.

3.2 Anforderungen an das System

Wir wollen ein System implementieren, das es den Nutzern vereinfacht, passende Dokumente zu ihren Arbeitsthemen zu finden. Dabei soll vor allem das Wissen der Nutzer über andere Netzwerkteilnehmer und die Erfahrungen aus vorherigen Anfragen dazu benutzt werden, passende Dokumente zu finden und unpassende Dokumente herauszufiltern. Das Suchsystem basiert auf drei Konzepten:

1. konzeptbasierte und komplexe Anfragen: Suche mit Termen und Konzepten, Suche mit mehreren Konzepten
2. persönliche Sicht auf das Netzwerk (Kompetenzen, Zufriedenheit)
3. personalisierte Suche: Berücksichtigung von Interessen des anfragenden Nutzers

3.3 Use Cases

Die Entwicklung unseres Suchsystems wurde in mehrere kleine Teilprobleme unterteilt, die aufeinander aufbauen (siehe Abb. 3.2). Damit kann besser dokumentiert werden, wie sich verschiedene Erweiterungen der Suchalgorithmen auf die Qualität der Ergebnisse auswirken.

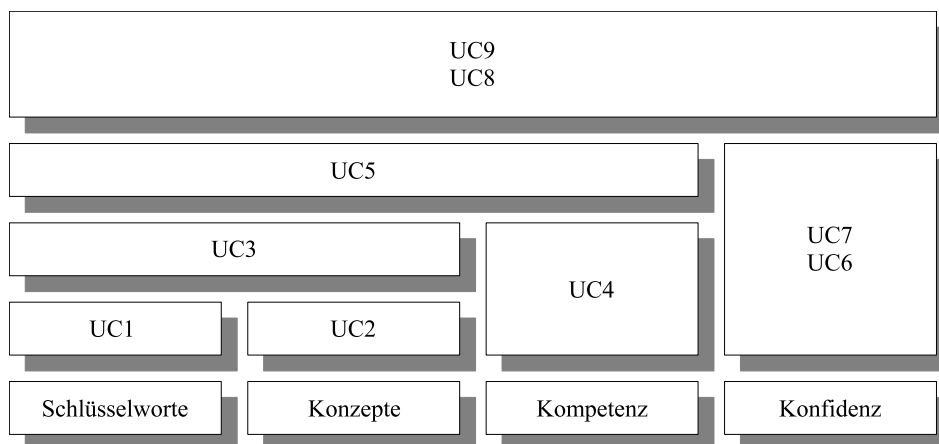


Abbildung 3.2: Use Cases

USE CASE 1: Volltextsuche

Wir beginnen mit einer einfachen Volltextsuche. Der Nutzer stellt eine Anfrage durch eine Menge von Schlüsselwörtern. Das System sucht auf allen Peers nach Dokumenten, die diese Schlüsselwörter enthalten.

Beispiel: Ich suche Dokumente mit einem Term bestehend aus den Schlüsselworten „workflow“ und „ontology“. Als Ergebnis bekomme ich eine gerankte Liste mit Dokumenten zurück, die zu diesem Term passen. Je häufiger die Schlüsselwörter im Dokument vorkommen, desto höher ist der Rang.

USE CASE 2: Einfache konzeptbasierte Dokumentsuche

Ausgehend von einem (oder mehreren) eigenen Konzepten suchen wir nach Dokumenten, die zu diesen Konzepten inhaltlich passen, weil sie thematisch ähnliche Dokumente beinhalten.

Beispiel: Ich suche Dokumente zu meinem Konzept „workflow“. Dem Konzept sind schon einige Dokumente zugeordnet. Für meine Anfrage gebe ich nun dieses Konzept als Eingabeparameter an und das System gibt mir eine Liste von Dokumenten zurück, die zu diesem Thema passen, weil sie inhaltlich ähnlich sind.

USE CASE 3: Konzeptbasierte Dokumentsuche mit Schlüsselwort abhängiger Filterung

Dieser Fall kombiniert die Volltextsuche mit der Konzeptsuche. Wir suchen zuerst, wie in Use Case 2, Dokumente, die zu den gegebenen Konzepten passen. Um die Menge der Dokumente einzuschränken, kann der Nutzer durch die Angabe von Schlüsselwörtern das Ergebnis weiter filtern.

Beispiel: Ich suche nach Dokumenten über Klassen in Java. Ich habe einen Ordner „JAVA“ mit einer Vielzahl von Dokumenten über Javaprogrammierung. Da ich aber speziell nur Dokumente brauche die Informationen über Klassen enthalten, gebe ich als Term noch „Klasse“ an. Als Ergebnis bekomme ich Dokumente zurück, die ähnlich zu meinem Konzept „Java“ sind und in denen

das Schlüsselwort „Klasse“ vorkommt.

USE CASE 4: Explizite Kompetenzverbreitung

Zur Verbesserung der Suche legen die Nutzer die Kompetenz ihres Peers fest. Die Kompetenz wird durch eine Untermenge der eigenen Konzepte ausgedrückt. Die Nutzer wählen *die* Konzepte von ihrem Datensatz aus, die gute Dokumente zu einem Thema enthalten.

Beispiel: Auf meinem Peer befinden sich viele Dokumente zu verschiedenen Themen in unterschiedlichen Konzepten, unter anderem habe ich ein Konzept „workflow“. Da ich mich intensiv mit dem Thema „workflow“ beschäftige, habe ich viele Dokumente zu diesem Thema auf dem Peer abgelegt und weiß, dass diese Dokumente gut zu der Thematik passen. Deshalb kennzeichne ich das Konzept „workflow“ als Kompetenz.

USE CASE 5: Konzeptbasierte Dokumentsuche mit Kompetenzzinbeziehung

Die Suche wird durch Hinzunahme der Kompetenzzinschätzungen verfeinert. Dokumente, die einem Kompetenzkonzept zugeordnet sind, werden stärker gewichtet als andere Dokumente des Peers.

Beispiel: Ich suche nach Dokumenten zum Thema „workflow“. Auf einem Peer werden einige Dokumente gefunden, wobei einige zu einem als Kompetenz gekennzeichneten Konzept gehören. Als Ergebnis bekomme ich eine Liste mit Dokumenten zurück, in der die Dokumente des Kompetenzkonzepts einen höheren Rang haben als andere Dokumente.

USE CASE 6: Anfrageunabhängige, explizite Qualitätseinschätzung fremder Konzepte

Um die eigenen Einschätzungen bei der Suche zu berücksichtigen, kann jeder Nutzer eines Peers die Qualität der Daten von anderen Peers bewerten. Unser System erlaubt eine Bewertung der Daten auf Konzeptebene.

Bei der einfachen Qualitätseinschätzung bewerten wir die Konzepte anfrageunabhängig, d. h. Dokumente aus schlecht bewerteten Konzepten werden generell niedriger eingestuft als Dokumente von positiv bewerteten Konzepten.

Beispiel: Ich suche nach Dokumenten über Javaprogrammierung. Das System gibt mir als Ergebnis auch Dokumente zurück, die sich mit Indonesien befassen, da ein Kollege ein Konzept „JAVA“ mit Dokumenten für eine Dienstreise angelegt hat. Dokumente über diese Insel sind für mich generell uninteressant, da sich meine Arbeit um Softwareentwicklung dreht. Deshalb sage ich dem System explizit, dass es den Ordner „JAVA“ des Kollegen mit seinen Dokumenten bei der Auswertung der Suche niedriger einstufen soll.

USE CASE 7: Anfrageabhängige, explizite Qualitätseinschätzung fremder Kategorien

Meistens kann man keine *grundsätzliche* Aussage über die Qualität eines Konzepts machen, weil die Dokumente des Konzepts nicht schlecht, sondern nur unpassend zu einer bestimmten Anfrage sind. Wir brauchen also eine Möglichkeit Konzepte anfrageabhängig zu bewerten. In unserem System sollen Anfragen normalerweise aus Konzepten bestehen, deshalb bedeutet anfrageabhängig in unserem Fall auch konzeptabhängig.

Beispiel: Bei der Suche nach Dokumenten zu meinem Konzept „workflow“, wurden unter anderem Dokumente aus verschiedenen Kategorien von Heiko zurückgegeben. Nach Sichtung der Dokumente habe ich festgestellt, dass Dokumente aus der Kategorie „wf“ gut zu meiner Anfrage passen. Die Dokumente aus „ontology“ sind für mich zwar generell nicht uninteressant, passen aber nicht zum gesuchten Thema. Durch mein Feedback kann ich dem System mitteilen, dass bei der nächsten Anfrage zum Thema „workflow“ Dokumente aus „wf“ bevorzugt angezeigt werden, die aus „ontology“ möglichst nicht mehr zu diesem Thema angezeigt werden. Bei einer Anfrage zu einem anderen Konzept spielt diese Einschätzung keine Rolle.

USE CASE 8: Konzeptbasierte Dokumentsuche mit voller Kompetenzeinschätzung

Hier werden zur Suche nun sowohl die Kompetenzeinschätzung der Peers als auch die Qualitätseinschätzung hinzugezogen.

Beispiel: Ich suche nach Dokumenten zum Konzept „workflow“. Auf Heikos Peer liegen passende Dokumente in verschiedenen Kategorien unter anderem „wf“ und „java“. In „wf“ habe ich schon oft passende Dokumente zu „workflow“ gefunden, wogegen mir die Dokumente aus „java“ zu diesem Thema noch nie weitergeholfen haben. Deshalb sollen die Dokumente aus „wf“ besser bewertet werden als die Dokumente aus „java“. Da die Kategorie „wf“ Teil der Kompetenz von Heikos Peer ist, sind diese Dokumente noch bevorzugt zu behandeln.

USE CASE 9: Suche mit Anfrageweiterleitung

Im letzten Fall wird unser Szenario erweitert. Da die Mitarbeiter nicht nur mit Kollegen zusammen arbeiten sondern auch Kontakte zu weiteren Wissenschaftlern außerhalb der Abteilung haben, werden Anfragen über einen bekannten Peer aus dem lokalen Netzwerk zu weiteren (kompetenten) Peers weitergeleitet.

Beispiel: Ich suche nach Dokumenten zum Thema „workflow“. Da ich weiß, dass mein Kollege Heiko einige gute Dokumente zu diesem Thema auf seinem Peer gespeichert hat, stelle ich die Anfrage an ihn. Heiko arbeitet gerade intensiv an diesem Thema und kennt aus seinen Forschungen Bernd aus einer anderen Abteilung, der auch in diesem Gebiet arbeitet. Deshalb leitet er meine Anfrage zu Bernd weiter. Als Ergebnis bekomme ich jetzt Dokumente von Heiko und von Bernd zurück, obwohl ich diesen selbst nicht kenne.

Kapitel 4

Konzeptionelle Umsetzung

In diesem Kapitel werden die Konzepte für die Entwicklung eines dezentralen Information Retrieval Systems vorgestellt, welches die Anforderungen aus dem vorherigen Kapitel umsetzt.

4.1 Das C2CIR-Netzwerk

Definition (C2CIR-Peer): *Das C2CIR-Netzwerk besteht aus einer Menge von C2CIR-Peers p , die miteinander zu einem P2P-Netzwerk verbunden sind. Mit dem Peer kann ein Benutzer seine Dokumente verwalten und für andere Nutzer zugänglich machen. Er stellt Anfragemechanismen für die Suche sowohl auf seiner eigenen Datenbasis als auch auf anderen Peers bereit. Der Peer erfasst persönliche Informationen aus den Daten und Vorlieben des Nutzers, um die Anfrageauswertung auf dessen Interessen abzustimmen.*

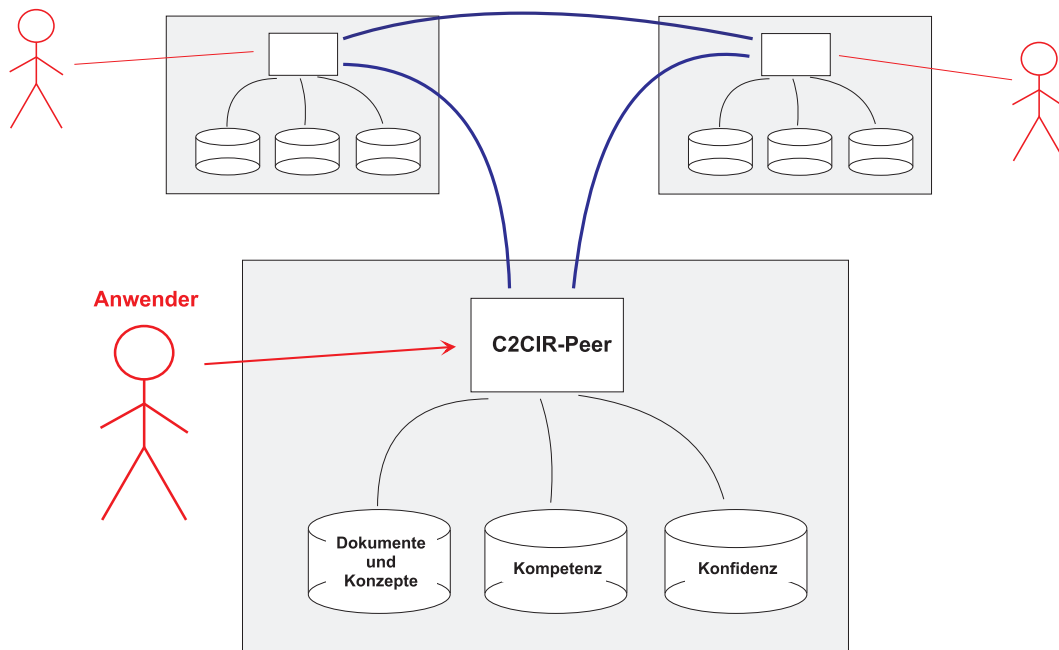


Abbildung 4.1: Das C2CIR-Netzwerk

4.2 Nutzung von persönlichen Strukturen

Traditionelle P2P-Systeme stellen die Daten dem Netzwerk unstrukturiert bereit. Bei der Suche nach Dokumenten werden alle Dokumente aus dem Datenbestand gleichberechtigt auf Ähnlichkeitskriterien untersucht. Diese flache Datenstruktur hat den Nachteil, dass keine Zusammenhänge zwischen den einzelnen Dokumenten erkennbar sind. Deshalb ist es schwer ausgehend von einem guten Dokument weitere Dokumente zu dem gleichen Thema zu finden.

Andere Systeme, wie z. B. *Bibster* (siehe Abschnitt 2.3.1), strukturieren die Daten durch eine gemeinsame Klassifikationsstruktur, einer Ontologie. Die Dokumente müssen nach einem festgelegten Schema in verschiedene Kategorien eingeteilt werden. Dadurch wird die themenorientierte Suche erleichtert. Der Einsatz von Ontologien hat den Nachteil, dass diese, um sie sinnvoll nutzen zu können, von den Nutzern verstanden werden müssen, d. h. jeder Nutzer weiß, welche Dokumente zu welcher Thematik gehören. Meist werden diese festen Strukturen von den Anwendern nur ungern benutzt. Da die Ontologie die Strukturierung sehr stark einschränkt und die Nutzer ihre Daten somit nicht nach eigenen Interessen und Wissen ordnen können. Für eine Organisation hat

die Ontologie den Nachteil, dass in einer flexiblen Arbeitswelt, mit sich ständig ändernden Arbeitsthemen, viel Aufwand zur Aktualisierung der globalen Strukturen betrieben werden muss.

Aufgrund der beschriebenen Nachteile, sollen in unserem System keine Ontologien zur Anwendung kommen. Es nutzt statt dessen die persönlich angelegten Strukturen der einzelnen Anwender. Die Nutzer werden in der Strukturierung ihrer Daten nicht eingeschränkt. Diese ordnen ihre Dokumente meist schon auf ihrem Rechner, indem sie z. B. eigene hierarchische Ordnerstrukturen aufbauen (siehe Abb. 4.2). Sie fassen Dokumente, die ihrer Meinung nach zusammengehören, in einem Ordner des Dateisystems zusammen. Ein solcher Ordner ist ein Beispiel für eine themenorientierte Zusammenfassung von Dokumenten, die man auch *Konzept* nennt.

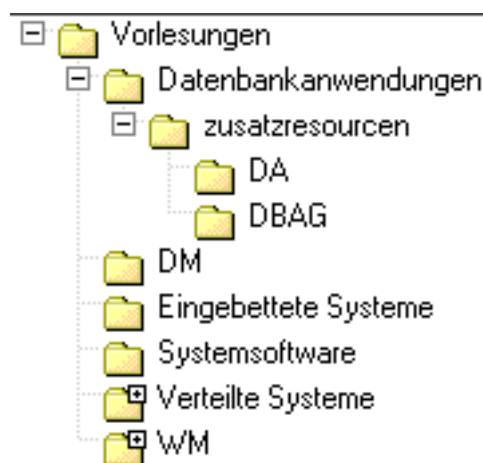


Abbildung 4.2: Hierarchisch geordnete Konzepte

Definition (Konzept): *Ein Konzept c ist eine Menge von Dokumenten eines Peers mit einem eindeutigen Bezeichner. Ein Konzept fasst thematisch ähnliche Dokumente unter einem gemeinsamen Namen zusammen.*

Im Gegensatz zu einfachen Dateisystemen ist es möglich, dass ein Dokument beliebig vielen Konzepten zugeordnet ist. Die Konzepte können wie im Dateisystem hierarchisch angeordnet werden. Durch die Hierarchie können Beziehungen zwischen den Konzepten hergestellt werden. Für die manuelle Suche haben

Hierarchien den Vorteil, dass der Nutzer relativ schnell durch ein themenorientiertes Absteigen in der Baumstruktur zu seinen gewünschten Dokumenten gelangen kann.

Da dem Nutzer keine Struktur vorgegeben wird, kann er mit seinem Hintergrundwissen die Daten ordnen. Da er die Struktur selbst angelegt hat, gibt es für ihn keine Verständnisprobleme und er findet sich in seinen Daten leicht zurecht. Anderen Anwender, die seinem Datenbestand durchsuchen, fehlt das implizite Wissen des Erstellers, deswegen ist es für sie meist viel schwieriger diese fremden Strukturen zu verstehen. Sie kennen nicht die genauen Zusammenhänge der Strukturen. So kommt es oft vor, dass verschiedene Anwender, obwohl sie an ähnlichen Themen arbeiten, ein Konzept mit ähnlichen Daten unterschiedlich benennen bzw. unter einem bestimmten Konzeptnamen andere Dokumente erwarten. Das heißt, dass die Dokument- und Konzeptnamen keine verlässlichen Kennzeichen zum Auffinden von relevanten Informationen sind.

Unser System soll den Anwender bei der Informationssuche auf fremden Systemen unterstützen, indem es Beziehungen zwischen eigenen und fremden Konzepten anzeigt und so das Auffinden von ähnlichen Informationen erleichtert. Um diese Beziehungen aufzudecken, müssen die persönlichen Sucherfahrungen der Nutzer ausgewertet werden. Die Nutzung von persönlichen Einschätzungen ist Thema des folgenden Abschnitts.

4.3 Persönliche Sicht auf das Netzwerk

Traditionelle P2P-Netzwerke sind per Definition Verbunde von gleichberechtigten Rechnern (siehe Abschnitt 2.2). Die Netzwerke versuchen durch Vernetzung der Rechner, eine große gemeinsame Datenbasis aufzubauen. Die Suche arbeitet in der Regel mit dem *Flooding-Algorithmus*. Da die Netze einer hohen Fluktuation sowohl der Teilnehmer als auch der Daten ausgesetzt sind, lassen sich schlecht längerfristig Informationen über die einzelnen Peers aufbauen.

Unser Kollegennetzwerk arbeitet unter anderen Bedingungen. Die Netzwerkstruktur ist im Allgemeinen statisch mit einer festen Zahl an Teilnehmern. Die Peers im Netzwerk arbeiten längerfristig zusammen, sodass das Sammeln von Informationen über die einzelnen Teilnehmer möglich ist. Da in der Regel

immer die gleiche Person an einem Peer arbeitet, gehen wir davon aus, dass die Themen der Dokumente eines Peers einen Teilbereich des Wissens seines Nutzers abbilden. Das bedeutet insbesondere auch, dass ein Peer, mit dem schon einmal erfolgreich Informationen über ein Thema ausgetauscht wurde, bei späteren Anfragen immer noch Informationen zu diesem Thema besitzt.

Als Grundlage für unsere personalisierte P2P-Suche wollen wir betrachten, wie ein Wissenschaftler ohne EDV-Unterstützung nach Informationen suchen würde:

Wenn ein Wissenschaftler nach Informationen im Kollegenkreis sucht, geht er nicht, wie bei der Suchstrategie der P2P-Netzwerke, in jedes Büro und fragt jeden Mitarbeiter, sondern er spricht ganz gezielt die Kollegen an, die er kompetent für dieses Thema hält. Die gezielte Anfrage kann er nur deshalb durchführen, weil er die anderen Kollegen und deren Erfahrungen kennt.

Beispiel: Ich suche nach Informationen zum Projekt „EPOS“. Das System soll nicht die Peers aller Mitarbeiter nach Informationen durchsuchen, sondern nur Dokumente von Peers, deren Nutzer am Projekt mitarbeiten.

Eine Suchsystem kann ohne zusätzliche Information über seinen Nutzer nur einen syntaktischen Vergleich von Dokumenten anhand des Inhalts durchführen. Damit der Peer eine personalisierte Suche durchführen kann, ist es also wichtig, das Wissen des Nutzers über andere Kollegen zu erfassen.

In unserem System werden folgende persönliche Einschätzungen eines Benutzer erfasst:

1. *Kompetenz*: Seine Einschätzungen zu der Qualität seiner eigenen Konzepte (Nicht alle Konzepte enthalten gute Dokumente!)
2. *Konfidenz*: Seine Einschätzungen zu der Kompetenz seiner Kollegen (Nicht alle fremden Konzepte sind für ihn interessant!)

4.3.1 Nutzung von Kompetenzen

Die Konzepte auf einem Peer sind nicht alle von gleich großer Bedeutung. Ein Teil der Konzepte enthält Dokumente, mit denen der Besitzer aktiv arbeitet. Sie sind gut ausgewählt und passen thematisch zusammen. Andere Konzepte bestehen aus noch ungeordneten Dokumentsammlungen, z. B. Dokumente, die der Nutzer noch lesen will.

Beispiel: Auf dem Peer von Stefan befinden sich unter anderem Konzepte mit den Namen „Programmierung“ und „Diplomarbeit“, welche Informationen von und über seine Arbeit beinhalten. Sein Datensatz enthält aber auch Konzepte „drafts“ und „toRead“, welche vorläufige oder ungeordnete Dokumente beinhalten. Außerdem hat er auch noch Konzepte mit Dokumenten, die nichts mit seiner Arbeit zu tun haben, z. B. „privates“, „fun“.

Definition (Kompetenz): *Die Kompetenz $comp$ eines Peers p ist die Teilmenge seiner Konzepte, die gute Dokumente enthalten. Die Einschätzung wird von dem Nutzer des Peers vorgenommen und stellt somit dessen subjektive Sicht auf seine Konzepte dar.*

Durch die Kennzeichnung der Kompetenz unterstützt der Benutzer des Peers *andere* anfragende Kollegen bei der Suche auf *seinem* Datenbestand. Er grenzt die für ihn besonders gut durchdachten Dokumentsammlungen von losen, willkürlich zusammengetragenen Dateien ab. Bei Anfragen auf den Datenbestand seines Peers werden die Dokumente aus den Kompetenzen bevorzugt behandelt.

4.3.2 Nutzung von Sucherfahrungen

Die Bewertung von Informationen ist subjektiv und hängt von dem Wissen und dem aktuellen Kontext (aktueller Arbeitsbereich, Interessen) des Suchenden ab. Was für den einen gute Dokumente bzw. interessante Themen sind, ist für den anderen schlecht oder uninteressant.

Das System soll dem Nutzer nur die Informationen bereitstellen, die ihn wirklich interessieren. Das wollen wir dadurch erreichen, dass das System die Zufrie-

denheit des Nutzers mit den Ergebnissen vorhergehender Anfragen auswertet und die Anfrageauswertung daran anpasst.

Definition (Konfidenz): *Die Konfidenz $conf$ ist die Beurteilung des Nutzers eines Peers, zur Qualität von fremden Konzepten.*

Wir nehmen an: Wenn ein Dokument den Nutzer nicht interessiert, sind auch alle anderen Dokumente dieses Themas für ihn wertlos. Aus diesem Grund bewerten wir durch die Konfidenz immer die Qualität auf Konzeptebene und nicht nur die eines einzelnen Dokumentes.

Wir unterscheiden zwei verschiedene Arten der Konfidenz:

1. *generelle Beurteilung ($gconf(\text{concept})$):* Die Qualität wird unabhängig von der Anfrage bewertet, d. h. Dokumente mit geringer Konfidenz sind generell uninteressant und sollen bei keiner Anfrage berücksichtigt werden. Dokumente mit hoher Konfidenz sind für den Nutzer grundsätzlich von Interesse und sollen bevorzugt behandelt werden.
2. *anfrageabhängige Beurteilung ($qconf(\text{query}, \text{concept})$):* Oft sind Dokumente für den Nutzer nicht generell uninteressant, sondern nur unpassend zur aktuellen Anfrage. Durch $qconf$ bietet das System dem Benutzer die Möglichkeit, die Qualität eines Konzepts in Abhängigkeit zu einer Anfrage zu bewerten.

Die Funktion $gconf(\text{concept})$ kennt drei verschiedene Zustände:

- *neutral* - keine Bewertung
- *1* - Dokumente des Konzepts sind gut
- *0* - Dokumente des Konzepts sind schlecht

Die Funktion $qconf(\text{query}, \text{concept})$ bewertet ein Konzept concept abhängig von dem Thema der Anfrage query . Die Bewertung wird durch das Feedback des Nutzers beeinflusst. Dieser kann wie bei der $gconf$ eine der drei Bewertungen abgeben. Im Gegensatz zu $gconf$, dessen Wertebereich nur aus 0 und

1 besteht, wird bei der *qconf* das Verhältnis der positiven Bewertungen zu der Gesamtzahl der Bewertungen berechnet:

$$qconf(query, concept) = \frac{\# \text{ positive Bewertungen } (query, concept)}{\# \text{ alle Bewertungen } (query, concept)}$$

Der Wertebereich dieser Funktion liegt zwischen 0 (nur negative Bewertungen) und 1 (nur positive Bewertungen).

4.4 Personalisierte Suche

Die personalisierte Suche hat zum Ziel den Nutzer mit den Informationen zu versorgen, die für ihn von Interesse sind. Diese Suche besteht aus vier Abschnitten:

1. Anfrageerstellung
2. Anfrageweiterleitung
3. Berechnung und Darstellung der Ergebnisse
4. Bewertung der Ergebnisse

4.4.1 Anfrageerstellung

Jede Suche beginnt damit, dass der Nutzer dem System übermittelt, zu welchem Thema er Informationen sucht.

Die meisten Information Retrieval Ansätze wie z. B. PlanetP oder NeuroGrid benutzen zur Umschreibung der gesuchten Informationen Schlüsselwörter. Um eine Thematik eindeutig zu spezifizieren, ist meist eine Vielzahl von Schlüsselwörtern nötig. Viele Wörter haben mehrere Bedeutungen, die nur aus dem Kontext eines Textes erfasst werden können. Die ungenaue Beschreibung der Schlüsselwortsuche führt deshalb dazu, dass Dokumente unterschiedlicher Themen gefunden werden.

Die Schlüsselwortsuche ist für unsere Kollegensuche nicht ausreichend, weil die themenbezogene Dokumentsuche bei Projektmitarbeitern im Mittelpunkt

steht. Wir nehmen an, dass die Nutzer vor allem weitere Dokumente zu einem ihrer Arbeitsbereiche suchen, wobei sie die Dokumente zu einem bestimmten Arbeitsbereich mittels Konzepten zusammengefasst haben.

Die Anfrage zu Dokumenten für ein eigenes Arbeitsgebiet lässt sich so sehr präzise durch die Nutzung von Konzepten darstellen. Zur Eingrenzung der Dokumentmenge kann der Nutzer noch Terme zur Suche hinzufügen.

Definition (Anfrage): *Eine Anfrage query ist eine Menge von Termen und/oder Konzepten, mit der der Nutzer eines Peers spezifizieren kann, nach welchen Informationen er sucht.*

4.4.2 Inhaltsbasierte Anfrageweiterleitung

Der nächste Schritt der Suche ist die Anfrage an die Peers weiterzuleiten, die passende Informationen bereithalten. Es wird also ähnlich wie in den vorgestellten Systemen (siehe Abschnitt 2.3.1) ein semantisches Routing vorgenommen.

Die meisten verteilten Wissensmanagementsysteme berechnen aus den Schlüsselwörtern eine Vorauswahl der Peers. Unser System arbeitet konzeptbasiert, deshalb soll auch die Auswahl der richtigen Peers für eine Anfrage durch Konzepte berechnet werden. Wir wollen mit Hilfe von inhaltlichen Vergleichen zwischen Konzepten der bekannten Peers zusammen mit den persönlichen Einschätzungen Beziehungen zwischen eigenen Konzepten und fremden Konzepten finden und diese in die Suche einbeziehen. Bei diesem sogenannten *Mapping* berechnet der Peer zu jeder konzeptbasierten Anfrage eine bewertete Liste von Kategorien aller bekannten Peers. Mit Hilfe dieser Konzeptliste werden dann auch die Dokumente bewertet. Je besser ein Konzept zu der Anfrage passt, desto besser sind auch dessen Dokumente zu bewerten.

Dieses Mapping von Konzepten dient auch als Grundlage für die Weiterleitung über mehrere Peers hinweg, das sogenannte *Hopping*.

Beispiel: Von Stefans Peer wird eine Suchanfrage zu dessen Konzept „Wissensmanagement“ abgesetzt. Der Peer berechnet, aus dessen Konfidenz und der Kompetenz, dass das Konzept „WM“ zu der Anfrage passt. Deshalb werden viele Dokumente dieses Konzepts an Stefan übermittelt. Da Heikos Peer

zusätzliche Verbindungen zu weiteren Peers hat, versucht er die Anfrage an diese Peers weiterzuleiten. Heikos Peer berechnet zu seinem Konzept „WM“ auch eine Peerliste. Seine Berechnung ergibt, dass Bernd mit „KM“ ein ähnliches Konzept besitzt, deshalb leitet er die Anfrage an Bernd weiter. Stefan erhält also die Dokumente von Heiko und von Bernd, obwohl Stefan und Bernds Peers vorher keine direkte Verbindung hatten.



Abbildung 4.3: Hopping auf Konzeptbasis

4.4.3 Dokumentranking

Definition (Relevanz): Die Relevanz rel eines Dokuments bezüglich einer Anfrage bezeichnet die Wahrscheinlichkeit, dass ein Dokument zu einer Anfrage passt. Die Relevanz ist ein subjektives Maß, also abhängig von den Einschätzungen des Anfragenden. Die Funktion $rel(doc, query)$ belegt reelle Werte zwischen 0 (Dokument passt überhaupt nicht) und 1 (Dokument passt perfekt zur Anfrage).

Wir treffen eine erste einfache Annahme zur Relevanz eines Dokuments: Wenn ein Benutzer Informationen zu einem bestimmten Thema sucht, dann sind Dokumente relevant, die einen ähnlichen Inhalt haben wie die Anfrage.

Definition (Ähnlichkeit): Als Similarity sim eines Dokumentes doc zu einer Anfrage $query$ bezeichnen wir eine vom System berechnete Ähnlichkeit, wobei $sim(doc, query) = 1$ identisch und $sim(doc, query) = 0$ keine Ähnlichkeit bedeutet.

Similarity vergleicht nicht, ob die Anfrage semantisch zum Dokument passt. Die Funktion arbeitet mit dem syntaktischen Vergleich der charakteristischen Terme von Anfrage und Dokument.

Der inhaltliche Vergleich auf Wortebene reicht uns aber für die Relevanzbetrachtung nicht aus. Wichtig für die Relevanz aus Nutzersicht ist die Qualität des Dokuments. Das System kann die Qualität nicht aus dem Inhalt des Dokumentes erschließen. Die Bewertung ist subjektiv aus der Sicht des Lesenden, also abhängig von dessen Kontext und Wissen. Aus diesem Grund haben wir zuvor zwei subjektive Maße der Qualität *Kompetenz* und *Konfidenz* eingeführt, die wir mit in die Relevanzberechnung einbeziehen werden.

Die Relevanz eines Dokuments *doc* zu einer Anfrage *query* ist abhängig von den oben definierten drei Faktoren:

1. Ähnlichkeit *sim* zwischen *query* und *doc*
2. Kompetenzen *comp*
3. Konfidenz *conf*

comp und *conf* sind auf Konzepten definiert. Die Dokumente in einem Konzept haben dieselben Werte, wie die Konzepte denen sie zugeordnet sind. Wir berechnen also *comp* und *conf* für Dokumente folgendermassen:

$$comp(doc) = comp(concept), \quad \text{mit } doc \in concept \quad (4.1)$$

$$conf(query, doc) = conf(query, concept), \quad \text{mit } doc \in concept \quad (4.2)$$

Bei Dokumenten die zu mehreren Konzepten gehören werden die Werte gemittelt. Die Relevanz eines Dokumentes berechnen wir so:

$$rel(doc, query) = \frac{\alpha \cdot sim(doc, query) + \beta \cdot comp(doc) + \gamma \cdot conf(doc, query)}{\alpha + \beta + \gamma} \quad (4.3)$$

Durch α, β und γ wird der Anteil der drei Funktionen an der Relevanz festgelegt. Die Verhältnisse dieser Faktoren sind abhängig vom Zustand des Systems.

Phase 1 (Initialzustand): Da im Initialzustand dem System noch keine bzw. nicht ausreichende Informationen über den Nutzer bekannt sind, wird die Relevanz am Anfang hauptsächlich durch *sim* bestimmt, d.h.

$$\alpha \gg \beta, \gamma.$$

Dies führt dazu, dass das Ergebnis Dokumente mit ähnlichem Text quer über alle Peers und durch alle Konzepte gesucht werden.

Phase 2 (Einführung von Kompetenzen): Die Angabe von Kompetenzen *comp* auf den verschiedenen Peers führt dazu, dass die Dokumente aus Kompetenzkategorien einen höheren Stellenwert bekommen als die anderen Dokumente der Peers, d.h.

$$\alpha = \beta \gg \gamma.$$

Jetzt werden vor allem Dokumente aus den Kompetenzklassen zurückgegeben.

Phase 3 (Einführung der Konfidenz): Durch jede Anfrage lernt das System immer mehr über die Interessen des anfragenden Nutzers. Das führt dazu, dass die Funktion *conf* immer aussagekräftiger wird, deshalb soll γ mit der Anzahl der gegebenen Feedbacks ansteigen. Wenn das System ausreichend Feedback bekommen hat, wird die Relevanz hauptsächlich durch *conf* dominiert, d.h.

$$\gamma \gg \beta, \alpha.$$

4.4.4 Bewertung der Suchergebnisse

Damit ein Peer die Einschätzungen seines Nutzers lernen kann, muss das System feststellen, welche Dokumente für den Nutzer interessant und welche für ihn unpassend sind. Wir nutzen dazu die expliziten Angaben der Benutzer. In weiteren Forschungen wäre zu überprüfen, wie man das System unabhängiger von den expliziten Angaben machen kann, indem das System die Einschätzungen implizit aus den Verhalten und Aktionen der Nutzer erfasst.

Wir benutzen Relevanz-Feedback. Nach einer Anfrage muss der Anwender die gefundenen Dokumente und Konzepte bewerten. Passend zu unseren beiden

verschiedenen Versionen der Konfidenz (anfrageabhängig, unabhängig) kann er zwischen 5 verschiedenen Qualitätsstufen auswählen (siehe Abschnitt 4.3.2):

1. *always good* - Konzept ist für den Nutzer immer von hohem Interesse
2. *good* - Konzept passt gut zur Anfrage
3. *neutral* - keine Bewertung
4. *bad* - Konzept passt nicht zur Anfrage
5. *always bad* - Konzept interessiert den Nutzer nicht.

Aus der Gesamtmenge dieser Bewertungen wird die Konfidenz berechnet.

Bei der Wirkungsweise der Bewertungen unterscheiden wir zwei verschiedene Zeiträume:

1. kurzfristige Auswirkung
2. langfristige Auswirkung

Um den Nutzer zu motivieren, den Feedbackmechanismus zu benutzen, werden die Bewertungen sofort ins Ergebnis miteingerechnet, indem z. B. schlecht bewertete Dokumente oder Konzepte sofort aus dem Ergebnis herausgefiltert werden.

Langfristig gesehen ist das Feedback einer einzelnen Anfrage nur wenig aussagekräftig. Die Qualität der Konfidenz steigt erst mit der Anzahl der Bewertungen, sodass ein einziges negatives oder positives Feedback das Ergebnis einer neuen Anfrage nicht so stark beeinflussen darf. Erst eine gewisse Menge von Bewertungen kann das Ergebnis signifikant verändern.

Nachdem wir nun die grundlegenden Konzepte unseres Systems vorgestellt haben, zeigen wir im folgenden Kapitel wie man die Ansätze konkret in ein Programm umsetzen kann.

Kapitel 5

Implementierungsaspekte

5.1 Systemaufbau

Wir implementieren das System als Erweiterung eines schon existierenden dezentralen Wissensmanagementsystems, des *brainFilers* (siehe Anhang A).

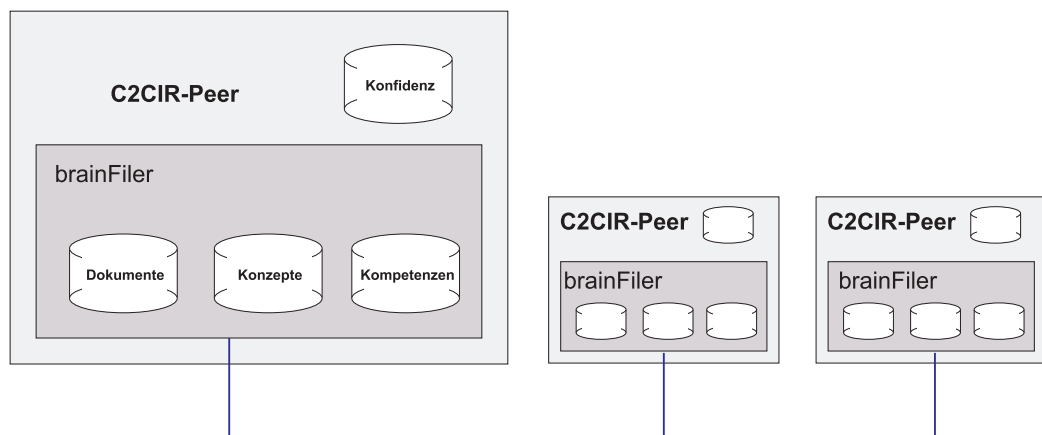


Abbildung 5.1: C2CIR: System-Design

5.1.1 Vorgegebene Funktionalität

Die Implementierung unseres C2CIR-Systems nutzt folgende Funktionalitäten des brainFilers:

1. Verwaltung von Dokumenten und Kategorien (Indexierung, Kategorisierung)
2. Inhaltliche Dokumentsuche mit Schlüsselworten oder Kategorien
3. Netzwerkzugriff

Verwaltung von Dokumenten und Kategorien: Die Verwaltung von Dokumenten ist im brainFiler sehr ausgereift. Man kann dem brainFiler Dateisystemverzeichnisse und Emailordner angeben, die in den Datenbestand aufgenommen werden. Zusätzlich lassen sich Dokumente zu beliebig vielen Konzepten zusammenfassen. Diese werden im brainFiler als *Kategorie*¹ bezeichnet. Zur Verwaltung der Kategorien erlaubt der brainFiler das Anlegen von beliebig vielen Taxonomien (*multikriterielle Dokumentenklassifikation*). Durch diese Strukturierung in Konzepte und Taxonomien ermöglicht der brainFiler eine themenorientierte Dokumentsuche.

Der brainFiler arbeitet mit dem Vektorraummodell (siehe Abschnitt 2.1.1), d. h. alle Dokumente werden über einen Termvektor repräsentiert. Der Vektor enthält die charakteristischen Terme eines Dokuments und deren Häufigkeit. Auch für die Kategorien werden Termvektoren berechnet. Da eine Kategorie eine Menge von Dokumenten repräsentiert, hängt deren Termvektor von den zugeordneten Dokumenten ab. Der Termvektor der Kategorie wird bei jeder Veränderung der Dokumentsammlung neu berechnet.

Netzwerkzugriff: Da wir uns in dieser Arbeit hauptsächlich auf die Entwicklung von Suchalgorithmen konzentrieren wollen verzichten wir auf die Implementierung eines eigenen Netzwerkmoduls einschließlich Protokoll und übernehmen für eine erste Implementierung die Netzwerkfunktionalität des brainFilers. Das System kann über die brainFiler-Schnittstelle mit jeder anderen

¹Da die Implementierung eng mit dem brainFiler zusammenhängt, nennen wir die Konzepte in diesem Kapitel Kategorien.

brainFiler-Instanz im Netzwerk kommunizieren, auf dortige Dokumente und Kategorien zugreifen und Suchanfragen (Schlüsselwortsuche, Konzeptsuche) durchführen. Das reicht für das einfache Szenario, bei dem jeder jeden kennt, vollkommen aus. Für die Erweiterung des Netzwerks um eine Anfrageweiterleitung wie sie in Use Case 9 beschrieben wurde, ist mit dem brainFiler allein nicht zu realisieren.

Suchfunktionalität: Der brainFiler bringt mehrere leistungsstarke Suchfunktionen mit sich. Da alle brainFiler-Instanzen miteinander verbunden sind, kann man Anfragen auf den Daten jedes beliebigen Peers im Netzwerk ausführen. Sowohl Dokumente als auch Konzepte sind durch Termvektoren beschrieben, dadurch ergeben sich einige Vergleichsmöglichkeiten. brainFiler bietet auf Basis dieser Termvektorvergleiche folgende Suchmöglichkeiten an:

Dokumentsuche über Schlüsselworte: Die Termvektoren der Dokumente werden nach einem oder mehreren Schlüsselworten durchsucht.

Dokumentsuche über Dokumente: Ausgehend von dem Termvektor eines Dokumentes werden weitere ähnliche Dokumente gesucht.

Dokumentsuche über Kategorien: Durch den Vergleich der Kategorievektoren mit Vektoren der einzelnen Dokumente können passende Dokumente zu einem Konzept gefunden werden.

Kategoriesuche über Dokumente: Durch Vergleich mit dem Dokumentvektor werden ähnliche Kategorien zu einem Dokument gesucht.

Kategoriesuche über Kategorien: Hier werden die Termvektoren der Kategorien verglichen.

5.1.2 C2CIR-Peer

Der *C2CIR-Peer* ist die Schnittstelle zwischen dem *C2CIR*-System und dem Nutzer. Dieser erweitert die Funktionalität des *brainFilers* um folgende Funktionen (siehe Abb. 5.2):

- Festlegen von Kompetenzen
- Einschätzung der Qualität der Kategorien von bekannten Peers
- Komplexe Suchanfragen aus Schlüsselworten und Kategorien
- Personalisierte Suchen

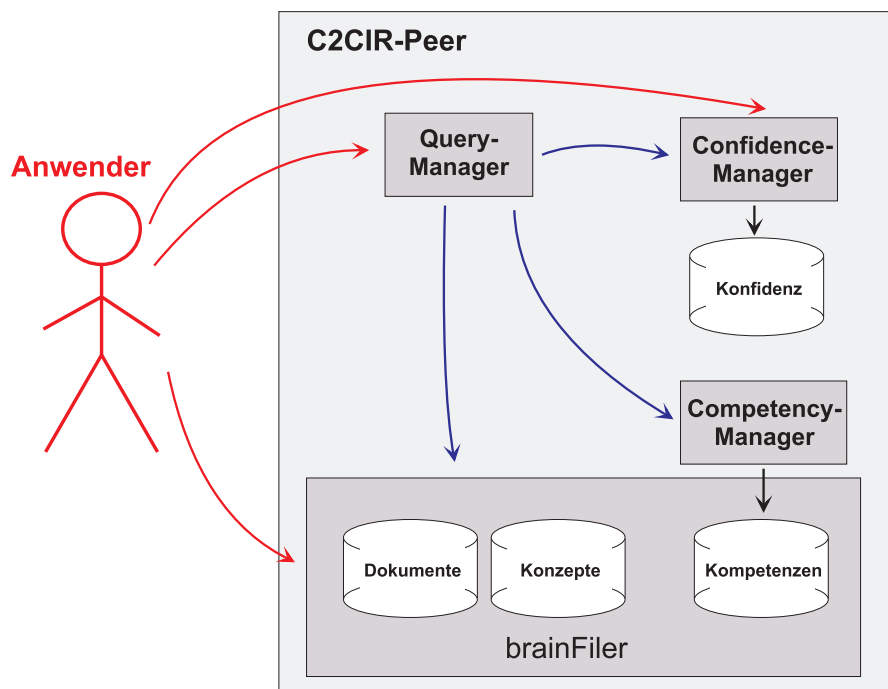


Abbildung 5.2: Der C2CIR-Peer

Die Dokumente und Konzepte des Systems werden durch den *brainFiler* organisiert. Für die Verwaltung der Kompetenzen eines Peers ist der so genannte *Competency-Manager* zuständig. Die persönlichen Einschätzungen zu der Qualität der Daten von anderen Peers verwaltet der *Confidence-Manager*. Der

Query-Manager ist für die gesamte Abwicklung einer Anfrage zuständig. Dieser kann für die Ausführung einer Anfrage auf alle Komponenten des Peers zugreifen.

Der Anwender kann über den *Query-Manager* Anfragen absetzen, durch den *Confidence-Manager* Kategorien bewerten und durch den integrierten brainFiler Dokumente und Kategorien verwalten. Kompetenzen sind spezielle Kategorien und werden ebenso durch den brainFiler organisiert.

In den folgenden Abschnitten wollen wir den Aufbau und die Arbeitsweise der einzelnen Komponenten genauer erläutern.

5.1.3 Competency-Manager

Der *Competency-Manager* verwaltet die Kompetenzen eines *C2CIR-Peers*. Der Peer wird um die Möglichkeit erweitert, Kategorien aus dessen Datenbasis als Kompetenz zu kennzeichnen. Die Auswahl der Kompetenz wird durch den Nutzer des Peers manuell festgelegt. Dieser wählt die Kategorien, die er für die Kompetenz seines Peers hält, aus und übergibt diese dem *Competency-Manager*.

Für die Berechnung der Kompetenz eines Dokuments stellt der *Confidence-Manager* eine Funktion bereit, wir nennen sie hier $comp(d)$. Diese berechnet zu jedem Dokument d , ob dieses Teil einer Kompetenzkategorie ist oder nicht.

Berechnung von $comp(d)$

- **Eingabe:**

Dokument d

- **Berechnung:**

- Berechne Kompetenz von d :

$$comp(d) = \begin{cases} 1 & d \in \text{Kompetenz} \\ 0 & \text{sonst} \end{cases}$$

Die Kompetenzen werden mit dem brainFiler in der Datenbasis eines Peers gespeichert. Dazu erstellen wir für die Kompetenzen eine eigene brainFiler-Kategorie. In unserem Prototyp haben wir die Kategorie „Kompetenzen“ genannt. Eine Kategorie wird durch eine Verknüpfung mit der Kompetenztaxonomie zu einer Kompetenzkategorie. Diese Kategorie wird in der aktuellen Version des Systems nicht hierarchisch betrachtet, d. h. als Kompetenzkategorien sind einzig die direkten Unterkategorien des Wurzelknotens zu behandeln.

Durch die Integration der Kompetenzverwaltung in die brainFiler-Strukturen kann man relativ leicht ohne ein zusätzliches Tool Kompetenzen auszeichnen und publizieren. Um eine Kategorie zu seiner Kompetenz zu erheben, kopiert man eine Verknüpfung in die Kompetenzkategorie (siehe Abb. 5.3).

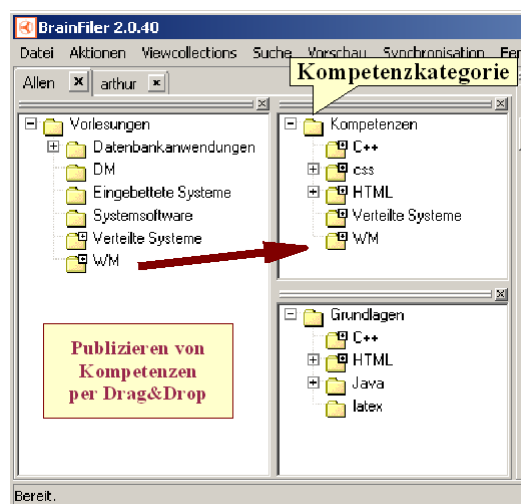


Abbildung 5.3: Kompetenzen werden in einer speziellen Kategorie „Kompetenzen“ gespeichert

5.1.4 Confidence-Manager

Der *Confidence-Manager* verwaltet die persönlichen Erfahrungen und Einschätzungen, die Konfidenz (siehe Abschnitt 4.3.2), von Informationen anderer Peers. Da wir bei der Konfidenz zwei Fälle unterscheiden (anfrageabhängig, anfrageunabhängig), muss der *Confidence-Manager* zwei Speicherstrukturen bereitstellen. Wir nutzen dazu Matrizen.

Anfrageunabhängige Konfidenzmatrix (Use Case 6) Durch die anfrageunabhängige Konfidenz $gconf(category)$ kann der Nutzer die Bewertung zu der Qualität eines fremden Konzeptes in die Suche einbeziehen. Die $gconf$ bewirkt ähnlich wie die Kompetenz eine bessere Bewertung dieser Kategorien. Die Bewertungen werden lokal in einer Matrix abgespeichert. Diese Matrix ordnet einer fremden Kategorie entweder den Wert 1 (gute Kategorie) oder 0 (schlechte Kategorie) zu. Kategorien, die nicht in der Matrix gespeichert sind, haben einen neutralen Wert, d. h. sie werden nicht bewertet.

Kategorie	Konfidenz	
Ludger, Ontology	1.0	immer gut
Sven, User Observation	1.0	immer gut
Sven, drafts	0.0	immer schlecht
Heiko,	

Abbildung 5.4: anfrageunabhängige Konfidenzmatrix

Die Berechnung der anfrageunabhängigen Konfidenz für ein Dokument greift auf die Matrix zu.

Berechnung von $gconf(c)$

- **Eingabe:**

Kategorie c

- **Berechnung:**

- Suche c in Tabelle und gib Konfidenzwert zurück

$$gconf(c) = \begin{cases} 1 & \text{gute Bewertung} \\ 0 & \text{schlechte Bewertung} \\ NULL & \text{keine Bewertung} \end{cases}$$

Anfrageabhängige Konfidenzmatrix (Use Case 7) Da die Anfragen in der Regel immer mit eigenen Kategorien abgesetzt werden, soll diese Matrix das Feedback in Abhängigkeit zu der Anfragekategorie speichern. Die Matrix speichert die Konfidenz zu einem Paar bestehend aus einem eigenen Konzept

und einem fremden Konzept. Die Konfidenz wird durch Feedback beeinflusst. Für die Aussagekraft der Einschätzung ist statistisch gesehen die Anzahl der abgegebenen Bewertungen ein wichtiges Kriterium: je mehr Bezugsdaten, desto genauer ist die Einschätzung. Dieses Phänomen nennt man auch das *Gesetz der großen Zahlen*. Deshalb reicht es nicht, nur die aktuelle Konfidenz zu speichern, sondern es muss immer die Anzahl der abgegebenen Bewertungen mitgeführt werden. Die Gesamtzahl dient zur Gewichtung der anfrageabhängigen Konfidenz.

eigene Kategorie	fremde Kategorie	Zufriedenheit	
WM	Ludger, Ontology	10/20	passt gut zu „WM“
WM	Ludger, WV	2/10	passt nicht zu „WM“
WM	Heiko, KM	9/10	
WM	Sven, workflow	9/10	

Abbildung 5.5: anfrageabhängige Konfidenzmatrix

Die Konfidenzfunktion $qconf(ownC, foreignC)$ gibt zu einer gegebenen Kategorie $ownC$ den Wert der Konfidenz und das Gewicht in Abhängigkeit zu einer eigenen Kategorie $foreignC$ zurück.

Berechnung von $gconf(ownC, foreignC)$

- **Eingabe:**

eigene Kategorie $ownC$

fremde Kategorie $foreignC$

- **Berechnung:**

- Suche Paar $ownC, foreignC$ in Tabelle und gib Konfidenzwert und das Gewicht zurück

- **Ergebnis:** Vektor $v = \begin{pmatrix} Konfidenz \\ Gewicht \end{pmatrix}$

Die beiden Konfidenzen werden in der Funktion $conf(ownC, foreignC)$ zu einem Wert zusammen gefasst.

Berechnung von $conf(ownC, foreignC)$

- **Eingabe:**

eigene Kategorie $ownC$

fremde Kategorie $foreignC$

- **Berechnung:**

- Berechne Wert $value - gconf$ der anfrageunabhängigen Konfidenz:

$$gconf = gconf(foreignC)$$

- Berechne Wert $value_qconf$ der anfrageabhängigen Konfidenz:

$$qconf = qconf(ownC, foreignC)$$

- Berechne die allgemeine Konfidenz $conf$ aus der anfrageunabhängigen Konfidenz und der anfrageabhängigen Konfidenz:

$$conf(ownC, foreignC) = \frac{gconf + qconf}{2}$$

- **Ergebnis:** Vektor $v = \begin{pmatrix} Konfidenz \\ Gewicht \end{pmatrix}$

Explizite Bewertung von Kategorien Der Nutzer muss dem System explizit mitteilen, welche Kategorien er für passend hält. Er kann Kategorien durch Auswahl einer dieser 5 Bewertungsstufen bewerten:

1. *always good* - Kategorie ist immer gut
2. *good* - Kategorie passt gut zur Anfrage
3. *neutral* - neutrale Bewertung
4. *bad* - Kategorie passt nicht zur Anfrage
5. *always bad* - Kategorie ist immer schlecht

In der Regel findet die Bewertung nach einer Anfrage als Feedback statt. Das Feedback wird an den *Confidence-Manager* übertragen. Dieser aktualisiert damit die jeweiligen Konfidenzwerte. Die Bewertungen 1, 3 und 5 wirken sich auf die anfrageunabhängige Matrix aus, die Bewertungen 2 und 4 auf die anfrageabhängige Matrix.

Anfrageunabhängige Bewertung Die anfrageunabhängige Konfidenz *gconf* kennt nur drei Zustände: „*always good*“ setzt die *gconf* einer Kategorie auf 1, „*always bad*“ auf 0. Mit „*neutral*“ wird eine Kategorie wieder aus der Liste gelöscht und der Rank somit nicht mehr durch die *gconf* bewertet.

Anfrageabhängige Bewertung Die anfrageabhängige Konfidenz *qconf* berechnet sich aus dem Quotienten von positiven Bewertungen und der Gesamtzahl aller Bewertungen:

$$qconf = \frac{\textit{positiv}}{\textit{positiv} + \textit{negativ}}$$

5.1.5 Das C2CIR-Suchmodul – Query-Manager

Der *Query-Manager* ist für die Abwicklung der Suche zuständig. Zur Anfrageauswertung muss er mit den anderen Komponenten des *C2CIR-Peers* und mit anderen Peers kommunizieren können.

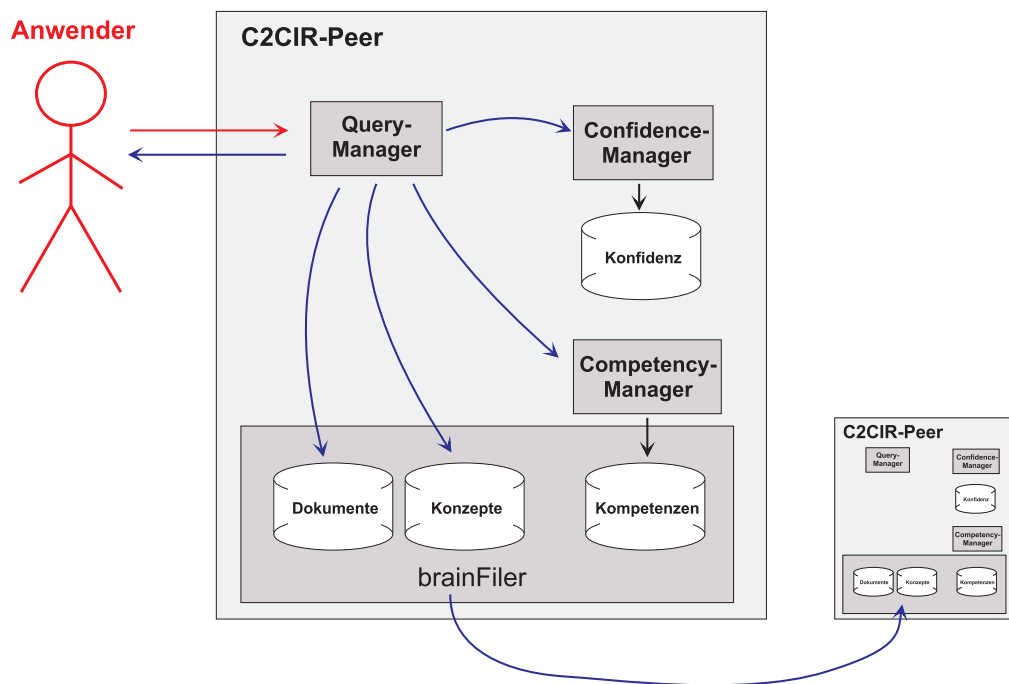


Abbildung 5.6: C2CIR: Anfragebearbeitung

Die aktuelle Umsetzung mit dem brainFiler als Kommunikationskomponente erlaubt keine direkte Verbindung zwischen den C2CIR-Peers. Da aber jeder Peer über den brainFiler auf die Dokumente, Konzepte und Kompetenzen aller anderen Netzwerkteilnehmer zugreifen kann (siehe Abb. 5.6) und die Konfidenzbewertung der Anfrage nur durch die Einschätzungen des lokalen Nutzers beeinflusst werden, ist auch keine direkte Verbindung nötig. Eine Anfrage kann komplett durch den lokalen Query-Manager bearbeitet werden.

Im Allgemeinen läuft die Suche in folgenden Schritten ab:

1. Die Suche wird durch eine Anfrage des Nutzers gestartet. Zur Formulierung der Anfrage wählt er aus seinem Datenbestand Kategorien aus und fügt eventuell noch eine beliebige Menge an Schlüsselworten zur Einschränkung der Suchergebnisse hinzu.
2. Die Anfrage wird an den eigenen Peer gestellt, der diese sofort an seine Instanz des *Query-Managers* weiterleitet.
3. Der *Query-Manager* berechnet mit Hilfe des *brainFilers* eine Liste von passenden Dokumenten auf allen bekannten Peers.
4. Durch den *Competency-Manager* werden Kompetenzen der einzelnen Peers in die Bewertung des Ergebnisses miteinbezogen.
5. Durch den *Confidence-Manager* werden die Ergebnisse nach den persönlichen Interessen des Nutzers geordnet.
6. Die Ergebnisse werden dann dem Nutzer als gerankte Liste präsentiert.

5.1.6 Umsetzung der Suchanfragen

Am Anfang unserer Überlegungen haben wir Anforderungen für das zu implementierende System durch eine Menge von Use Cases aufgestellt (siehe Abschnitt 3.2). In den folgenden Abschnitten wollen wir uns jetzt genauer mit der konkreten Umsetzung der Suchalgorithmen beschäftigen.

Wie oben erklärt, startet eine Suche mit der Formulierung der Anfrage q durch den Benutzer. Die Anfrage $q = (T, C)$ ist ein Vektor, mit dem Term T als Menge von Schlüsselworten t_0, \dots, t_n und C als eine Menge von Kategorien c_0, \dots, c_m .

Die Ergebnisse der Suchalgorithmen sind sogenannte gerankte Listen. Das sind geordnete Mengen von Dokumenten, zu denen ein Wert zwischen 0 und 1 für die Relevanz zur Anfrage, der sogenannte *Rank*, zugeordnet ist.

Dokument	Rank
Information-Retrieval.pdf	0.9
Einführung-Wissensmanagement.pdf	0.87
Knowledge-Management.doc	0.85
⋮	⋮

Tabelle 5.1: gerankte Liste

In den Algorithmen wird eine Funktion $rank(L, d)$ benutzt. Diese gibt für ein bestimmtes Dokument d den Rang aus einer gegebenen gerankten Liste L zurück. Wenn das Dokument nicht in der Liste vorhanden ist, hat die Funktion den Wert 0 als Ergebnis.

$$rank(L, d) = \begin{cases} rank(L, d) & \forall d \in L \\ 0 & sonst \end{cases}$$

Use Case 1 - Volltextsuche

Der erste Anfragefall ist die Volltextsuche. Der Nutzer stellt seine Anfrage in Form eines Terms, welcher einen oder mehrere Suchbegriffe enthält. Der *Query-Manager* berechnet mit diesem Term t über die brainFiler-Volltextsuche, hier $bF_{term}(t)$ genannt, eine Liste mit passenden Dokumenten von den bekannten Peers und gibt diese Liste als Endergebnis an den Nutzer zurück (siehe auch Abschnitt A.2.3).

Berechnung von $textSearch(q)$

- **Eingabe:**

Anfragevektor $q = (T, -)$ mit T ist Menge von Termen t_1, \dots, t_n

- **Berechnung:**

- Berechne mit brainFiler Ergebnisliste für Terme T :

$$L = bF_{term}(T)$$

- **Ergebnis:** geordnete Liste mit Dokumenten

$$textSearch(q) = L$$

Use Case 2 - Konzeptsuche

Im zweiten Fall sucht der Nutzer Dokumente durch eine Menge von Konzepten. Die Anfrage wird durch eine Menge von brainFiler-Kategorien formuliert. Zur Auswertung greift der *Query-Manager* auf die Kategorie-Suchfunktion des *brainFilers* zu, hier $bF_{category}(c)$ genannt, diese berechnet zu einer gegebenen Kategorie c eine Menge von passenden Dokumenten (siehe Abschnitt A.2.3).

Zusammenfassung des Algorithmus: Wir berechnen mit den Funktionen des *brainFilers* zu jeder Kategorie eine Liste von ähnlichen Dokumenten. Die einzelnen gerankten Listen werden zu einer gemeinsamen Liste vereinigt. Bei der Zusammenfassung wird der Rang eines Dokuments als Mittelwert des Dokumentrangs aus den einzelnen Listen berechnet. Dabei gilt, wenn ein Dokument in einer Liste nicht vorhanden ist, bekommt es den Rang 0.

Berechnung von *conceptSearch(q)*

- **Eingabe:**

Anfragevektor $q = (-, C)$ mit C ist Menge von Kategorien c_1, \dots, c_m

- **Berechnung:**

- Berechne mit *brainFiler* eine Ergebnisliste für jede einzelne Kategorie c_i : $L_i = bF_{category}(c_i)$
- Zusammenfassung der Ergebnislisten zu $L = L_1 \cup \dots \cup L_m$
- Berechnung des Rangs für jedes Dokument $d \in L$ durch Mittelwertbildung des Rangs aus den verschiedenen Listen

$$rank(L, d) = \frac{rank(L_1, d) + \dots + rank(L_m, d)}{m}$$

- **Ergebnis:** geordnete Liste mit Dokumenten

$$conceptSearch(q) = L$$

Use Case 3 - Berechnung der Ähnlichkeit durch Konzepte und Text

In diesem Fall sollen Volltextanfragen und Kategorieanfragen zu einer komplexen Anfrage verknüpft werden. Die Ergebnisliste aus der Kategorieanfrage wird durch Angabe von Schlüsselworten neu geordnet.

Zusammenfassung des Algorithmus: Der *Query-Manager* berechnet die Dokumentlisten für die Kategorien (UC2) und für die Terme (UC1). Die beiden Listen werden zusammengefasst, indem der neue Rang eines Dokuments durch Mittelwertbildung der Dokumentränge aus den beiden Listen berechnet wird. Im Ergebnis haben dann vor allem diejenigen Dokumente einen hohen Rang, die sowohl in der Liste der Volltextsuche als auch in der Liste der Konzeptsuche vorhanden sind.

Berechnung der *similarity*(*q*)**• Eingabe:**

Anfragevektor $q = (T, C)$ mit T ist Menge von Termen und C ist eine Menge von Kategorien c_1, \dots, c_m .

• Berechnung:

- Berechne Teilergebnisliste für Terme T :

$$L_1 = \text{textSearch}(q)$$

- Berechne Teilergebnislisten für die Kategorien c_i :

$$L_2 = \text{conceptSearch}(q)$$

- Zusammenfassung der Teilergebnisse $L = L_1 \cup L_2$

- Berechnung des Rangs für jedes Dokument $d \in L$ durch Mittelwertbildung der Ranks aus den verschiedenen Ergebnislisten

$$\text{rank}(L, d) = \frac{\text{rank}(L_1, d) + \text{rank}(L_2, d)}{2}$$

• Ergebnis: geordnete Liste mit Dokumenten

$$\text{similarity}(q) = L$$

Use Case 6 - Kompetenzen

In Use Case 6 werden Kompetenzen der einzelnen Peers beim Dokumentranking mitberücksichtigt. Die Dokumente aus den Kompetenzkategorien werden höher gewichtet als andere Dokumente.

Zusammenfassung des Algorithmus: Die Berechnung der ähnlichen Dokumente wird durch die Funktion $similarity(q)$ durchgeführt. Danach folgt eine Neubewertung der Dokumente abhängig von ihrer Kompetenz. Dies wird durch Mittelwertbildung zwischen Dokumentrang und Kompetenzwert berechnet. Die Kompetenzfunktion gibt nur die Werte 0 oder 1 zurück, das dazu führt, dass im negativen Fall der Rang des Dokuments halbiert und im positiven Fall verbessert wird.

Beispiel:

keine Kompetenz: $rank(d) = \frac{rank(d)+0}{2}$

Kompetenz: $rank(d) = \frac{rank(d)+1}{2}$

Berechnung von $competencyRank(q)$

- **Eingabe:**

Anfragevektor $q = (T, C)$ mit T ist Menge von Termen und C ist Menge von Kategorien c_1, \dots, c_n .

- **Berechnung:**

- Berechne Dokumentliste für Anfrage, wie in UC3

$$L_{sim} = sim(q)$$

- Berechne neue Liste L_{comp} aus den Dokumenten $d \in L_{sim}$ und berechne den neuen Rank abhängig von der Kompetenz:

$$rank(L_{comp}, d) = \frac{rank(L_{sim}, d) + comp(d)}{2}$$

- **Ergebnis:** geordnete Liste mit Dokumenten

$$competencyRank(q) = L_{comp}$$

Use Case 9 - Konfidenz

In diesem Fall werden die Interessen des anfragenden Anwenders in die Bewertung miteinbezogen. Der *Query-Manager* berechnet zuerst eine Liste mit Dokumenten wie in UC6 und ordnet die Dokumente in Abhängigkeit zu den Informationen des *Confidence-Managers* neu.

Zusammenfassung des Algorithmus: Zuerst wird eine Liste mit passenden Dokumenten wie in Use Case 3 berechnet. Die Dokumente werden danach durch die Einbeziehung der Kompetenz- und Konfidenzinformationen neu geordnet. Der Einfluss der einzelnen Informationen wird durch die Gewichte α , β und γ festgelegt. Die Aussagekraft der Konfidenzdaten ist abhängig von der Anzahl der abgegebenen Bewertungen. Bei einer geringen Anzahl von Bewertungen ist diese noch zu speziell und wird deshalb noch nicht voll in die Bewertung miteinbezogen. Mit steigender Anzahl des Feedbacks erhöht sich die Aussagekraft, deshalb wird γ in Abhängigkeit zu der Anzahl der Bewertungen berechnet. Die Werte für α und β werden vom Nutzer festgelegt.

Berechnung von *confidenceRank*(q)

- **Eingabe:**

Anfragevektor $q = (T, c_1, \dots, c_n)$ mit T ist Menge von Termen und c_1, \dots, c_n sind Kategorien.

- **Berechnung:**

- Berechne Dokumentliste für Anfrage, wie in UC3

$$L_{sim} = sim(q)$$

- Berechne neue Liste L_{conf} für alle $d \in L_{sim}$ mit Rank abhängig von der Kompetenz $comp(d)$ und der Konfidenz

$conf(q, d)$:

$$rank(L_{conf}, d) = \frac{\alpha \cdot rank(L_{sim}, d) + \beta \cdot comp(d) + \gamma \cdot conf(q, d)}{\alpha + \beta + \gamma}$$

- **Ergebnis:** geordnete Liste mit Dokumenten

$$confidenceRank(q) = L_{conf}$$

5.2 C2CIR-GUI

Zur Visualisierung der Ergebnisse und zur einfachen Bedienung des Peers haben wir eine relativ einfache Benutzerschnittstelle, auch *Graphical User Interface (GUI)* genannt, implementiert, mit der man alle Funktionen des C2CIR-Peers ausführen kann.

Der Entwurf und die Entwicklung einer funktionalen und nutzerfreundlichen GUI für das C2CIR-System führt über den zeitlichen Rahmen dieser Diplomarbeit hinaus. Zu Demonstrationszwecken haben wir deshalb nur eine einfache GUI entwickelt. Diese soll die brainFiler-GUI um die Suchfunktionen und die Feedback Mechanismen des C2CIR-Peers erweitern. Um alle Funktionen des Systems nutzen zu können, muss man in der aktuellen Version noch parallel mit beiden GUIs arbeiten. Die brainFiler-GUI dient weiterhin zur Verwaltung der Daten, die C2CIR-GUI ist auf die Suchfunktionen spezialisiert. Zu einem späteren Zeitpunkt müssen beide Systeme kombiniert werden, z. B. durch Integration der Suchalgorithmen in das brainFiler-System oder durch die Entwicklung einer GUI mit voller Funktionalität des Systems.

5.2.1 Aufbau

Da unser System auf der Funktionalität des brainFilers arbeitet, haben wir uns bei der Entwicklung der GUI an der aktuellen brainFiler-GUI orientiert (siehe Anhang A).

Das Hauptfenster ist in drei Teile unterteilt (siehe Abb. 5.7). Auf der linken Hälfte befindet sich ein Browser zum manuellen Durchsuchen der Konzepte und Dokumente eines Peers. Die rechte Seite besteht aus zwei Komponenten: Der Bereich oben enthält Informationen über die Anfrageparameter. Rechts unten werden die Ergebnisse in Kategorie- und Dokumentlisten visualisiert.

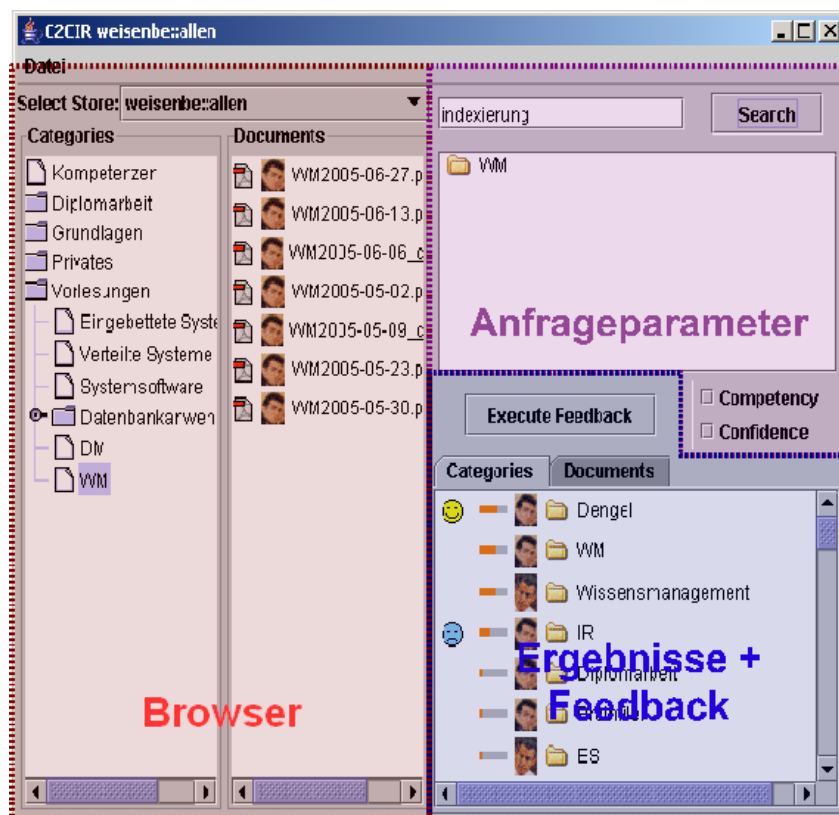


Abbildung 5.7: Aufteilung der C2CIR-GUI

Browser

Das Browserfenster (siehe Abb. 5.8) erlaubt es dem Nutzer sowohl die Konzepthierarchien seines Peers als auch die anderer bekannter Peers manuell zu durchsuchen. Die Auswahl des aktuellen Peers kann der Nutzer mit dem Auswahlmenü oben links durchführen. Wie bei den bekannten Dateisystembrowsern werden bei der Auswahl einer Kategorie die direkt zugeordneten Dokumente angezeigt. Der Browser unterstützt den Anwender bei der Erstellung einer Suchanfrage. Eine ausgewählte Kategorie wird durch einen Rechtsklick mit der Maus zu den Anfrageparametern hinzugefügt.

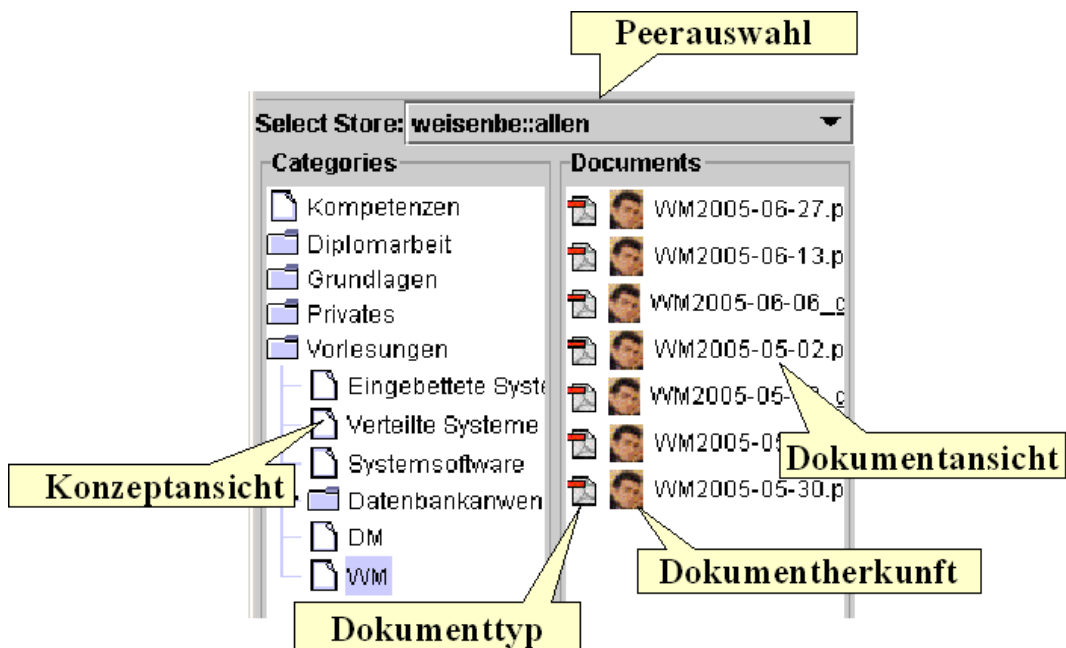


Abbildung 5.8: C2CIR-GUI: Browser-Komponente

Anfrageparameter

Oben im rechten Bereich des Fensters befinden sich die Eingabefelder für die Anfrageparameter (siehe Abb. 5.9). Es besteht aus einem Textfeld zur Eingabe von schlüsselwortbasierten Abfragen und einer Liste zur Verwaltung der Konzeptanfragen. Die Liste wird durch die Auswahl der Kategorien aus dem Browserfenster gefüllt. Zum Entfernen einer Suchkategorie reicht ein Rechtsklick mit der Maus. Der Nutzer kann außerdem durch An- oder Abwahl der beiden Auswahlfelder in der Mitte die Bewertungsstrategien für die Anfrage einstellen: „Competency“, zur Berücksichtigung von Kompetenzen bei der Ergebnisbewertung und „Confidence“, um die persönlichen Sucherfahrungen in die Bewertung mit einzubeziehen.

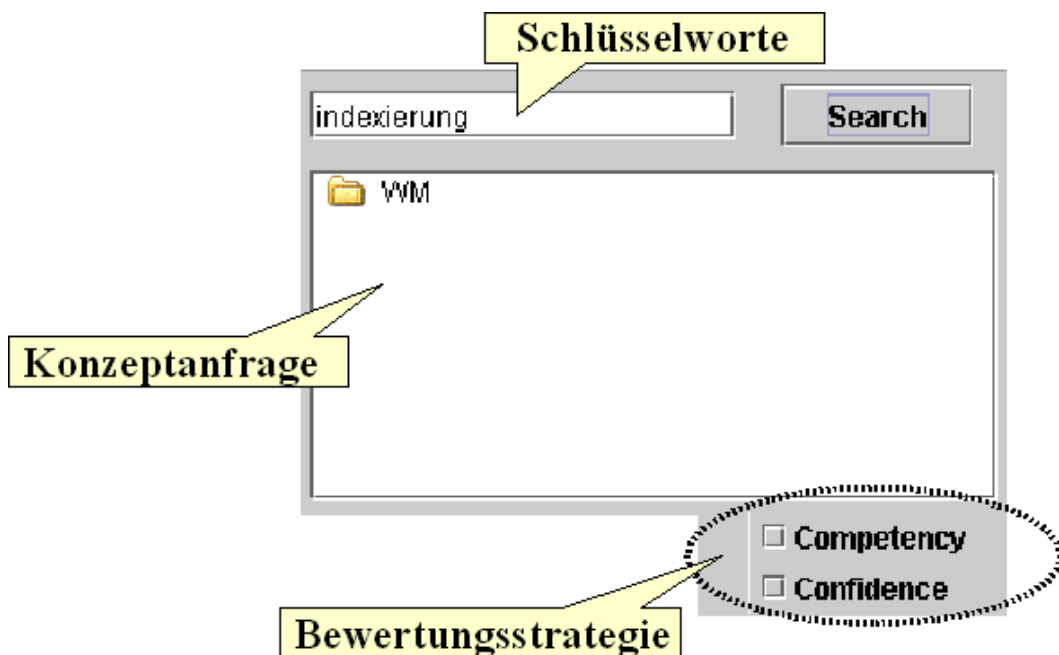


Abbildung 5.9: C2CIR-GUI: Anfrageparameter

Ergebnisvisualisierung und Bewertung

Unten im rechten Bereich befinden sich die Komponenten zur Visualisierung der Anfrageergebnisse. Die Ergebnisse der Anfrage werden sowohl als Liste von Kategorien als auch klassisch als Liste von Dokumenten visualisiert (siehe Abb. 5.11 und Abb. 5.12). Die geordnete Kategorielliste zeigt dem Nutzer Beziehungen zwischen den Abfragekategorien und weiteren Kategorien von bekannten Peers an. Je höher Rang einer Kategorie im Ergebnis ist, desto besser passt sie zur Anfrage. Dadurch soll es erleichtert werden Informationen aus einem ähnlichen Kontext zu finden. Jeder Peer hat ein eigenes Icon. Dieses wird mit jedem Konzept und Dokument angezeigt. Dadurch kann man im Anfrageergebnis sofort sehen, von welchem Peer ein Element der Ergebnisliste stammt. Weiterhin ist es möglich sich die Lage eines Dokuments oder Kategorie in den Taxonomien anzeigen zu lassen (siehe Abb. 5.10). Die Anzeige der Dokumente einer Kategorie erleichtert die manuelle Suche nach weiteren Dokumenten. Die Kenntnis von Dokumenten einer Ergebniskategorie ist auch eine wichtige Grundlage für den Nutzer zur Bewertung der Ergebnisse.

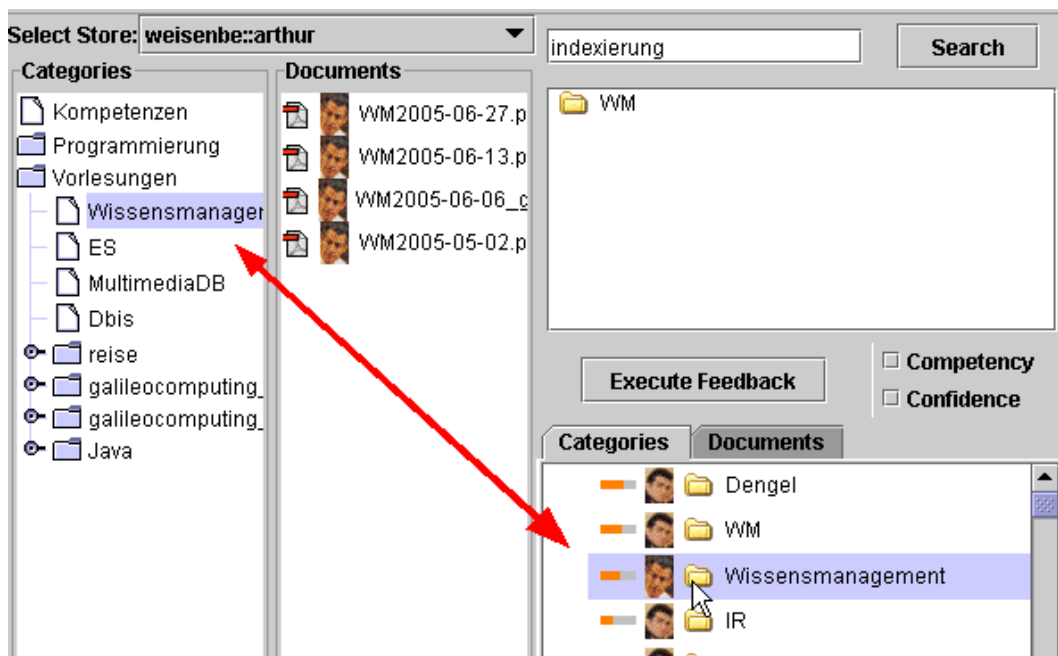


Abbildung 5.10: C2CIR-GUI: Anzeigen des Speicherorts

Die GUI bietet dem Nutzer die Möglichkeit, durch Feedback das Ergebnis zu bewerten. Durch Mausklicks mit der rechten Taste auf eine Kategorie oder ein Dokument kann man deren Bewertung verändern. Durch mehrfaches Klicken auf ein Objekt ändert man das Feedback zwischen *gut*, *schlecht* und *keine Bewertung*. Die aktuelle Bewertung wird optisch durch Icons neben dem Objekt angezeigt: ein lachendes Gesicht bedeutet *gut* und ein trauriges Gesicht *schlecht*. Durch Drücken des Bewertungsknopf „Execute Feedback“ werden die Ergebnislisten neu berechnet. Als kurzfristige Auswirkung werden die Listen abhängig von dem Feedback gefiltert. Die Dokumente von schlecht bewerteten Kategorien werden entfernt und die guten werden besser bewertet. Zur langfristigen Verbesserung der Suche gehen die Bewertungen in die Berechnung der Konfidenz ein.

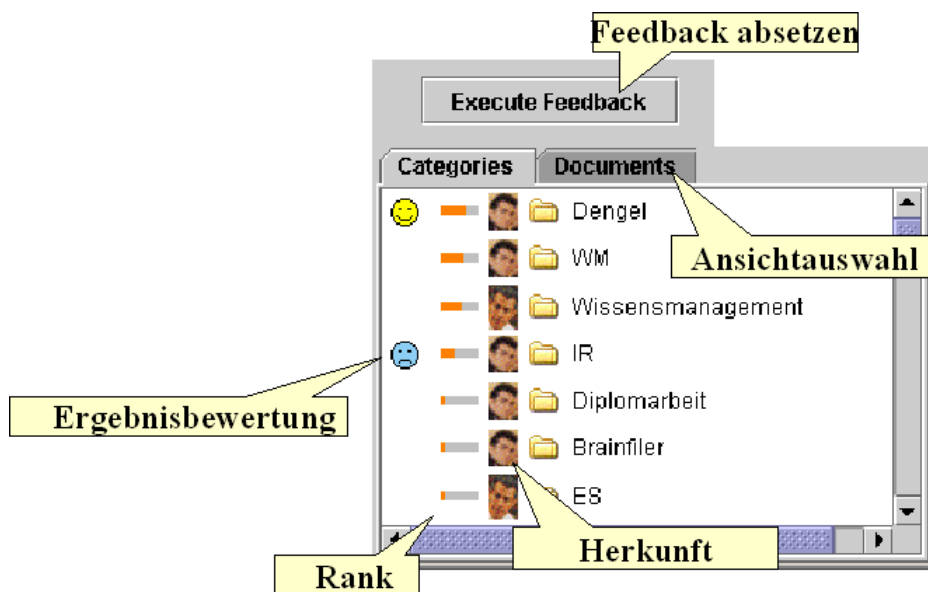


Abbildung 5.11: C2CIR: Kategorieergebnisse und Feedback

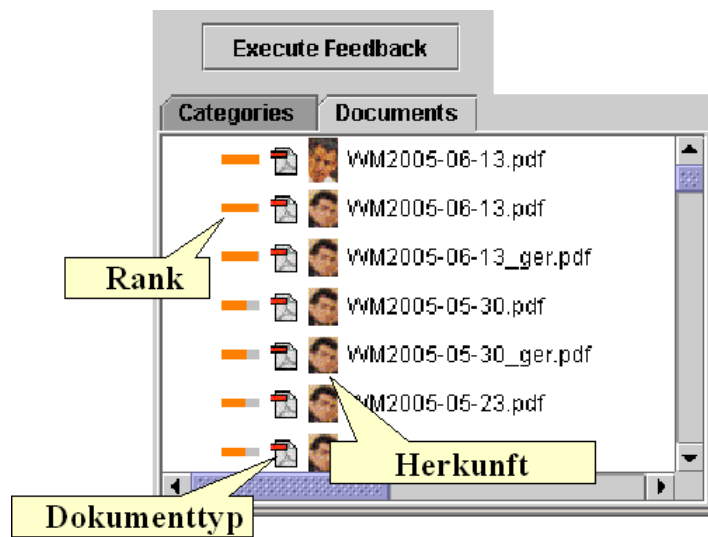


Abbildung 5.12: C2CIR-GUI: Dokumentergebnisse

Kapitel 6

Diskussion und Ausblick

Wir haben ein System entwickelt und implementiert, welches Mechanismen für eine personalisierte Suche auf P2P-Netzwerken zur Verfügung stellt. Dazu werden die Einschätzungen der Netzwerkteilnehmer bezüglich ihrer Daten und die Erfahrungen aus Suchanfragen dazu benutzt, die Ergebnisse des Information Retrievals auf die Bedürfnisse des Anfragenden anzupassen.

Da der zeitliche und inhaltliche Rahmen einer Diplomarbeit beschränkt ist, konnten weitere interessante Aspekte in dieser Arbeit nicht mehr berücksichtigt werden:

Evaluation: Erste Testläufe haben gezeigt, dass der Einsatz unseres C2CIR-Systems die Suchergebnisse aus Sicht des Anwenders verbessern kann. Um genauere Daten über die Auswirkungen der Suchalgorithmen zu erhalten, ist es nötig, eine umfangreiche Evaluation durchzuführen. Dazu müsste im ersten Schritt zuerst einmal eine Testkollektion als *Ground Truth* zusammengestellt werden, die den Anforderungen für unser Kollegennetzwerk entspricht. Mit dieser Testkollektion kann man z. B. die Auswirkungen der einzelnen Suchalgorithmen testen, indem für jeden Use Case eine eigene Precision-Recall Studie durchgeführt und ausgewertet wird (siehe Abschnitt 2.1.5).

Weiter wäre es interessant, eine Nutzerstudie durchzuführen. Es ist wichtig zu überprüfen, ob unser Ansatz für die personalisierte Suche von den Anwendern akzeptiert und benutzt wird. Dazu müsste das Programm im

längeren praktischen Einsatz verwendet werden. Als Testumgebung würde sich insbesondere ein Netzwerk eignen, in dem schon der brainFiler zum Einsatz kommt. Die Anwender sind somit schon mit der Nutzung solcher dezentralen Wissensmanagementsystemen vertraut. Mein Vorschlag ist, das System zusammen mit den Mitarbeitern der Abteilung „Knowledge Management“ am DFKI in Kaiserlautern zu testen. Dort benutzen schon einige Wissenschaftler den brainFiler zum Dokumentaustausch. Auf deren Rechner könnte man leicht das C2CIR-System integrieren. Nach einer längeren Testphase werden diese dann nach ihren Eindrücken und Erfahrungen mit dem System befragt.

Implizite Informationsgewinnung: Die Informationen über die Einschätzungen der Nutzer stammen in der aktuellen Version der Implementierung aus deren expliziten Angaben. Da die Qualität der Ergebnisse der Algorithmen stark von diesen Informationen abhängig ist, erfordert es von den Anwendern, dass diese von den Feedbackmechanismen Gebrauch machen und vor allem die Bewertungen sehr gewissenhaft durchführen. Jede falsche Angabe verschlechtert das Ergebnis. Um das System unabhängiger von den Angaben des Nutzers zu machen, wäre es im nächsten Schritt sinnvoll, zu erforschen, wie man die Informationen über die Zufriedenheit des Nutzers automatisch erkennen kann. Ein Ansatz dafür wäre festzustellen, ob ein Dokument wirklich gelesen oder nur überflogen wurde. Dies ist z. B. Gegenstand eines aktuellen Projekts am DFKI.

Erweiterung der Implementierung: Die aktuelle Version des Programms wird als Erweiterung auf eine lokale brainFiler-Instanz gesetzt. Das hat den Vorteil, dass man einen C2CIR-Peer leicht in ein bestehendes Netzwerk integrieren kann. Die Nutzung des brainFilers als Netzwerkkomponente bringt einen Nachteil mit sich: Die Kommunikation ist eingeschränkt auf dessen Protokoll. In weiteren Forschungen wäre zu überdenken, ob man die Netzwerkkommunikation von dem brainFiler-System abkoppelt und diesen nur noch als lokale Datenbasis verwendet. Dazu müssten für die C2CIR-Peers ein eigenes Netzwerkprotokoll erstellt werden, dass die Anforderungen an unser System direkt erfüllt (Verbreitung von Kompetenz, Anfrageweiterleitung).

Die Akzeptanz eines System hängt sehr stark von dessen Bedienbar-

keit ab. Je komplizierter die Bedienung ist, desto geringer die Nutzung. Das aktuelle System erfüllt die Kriterien der einfachen Bedienbarkeit noch nicht, da man zur Nutzung der vollen Funktionalität mit zwei verschiedenen GUIs arbeiten muss (brainFiler-GUI und C2CIR-GUI). Die GUI wurde auch hauptsächlich zu Test- und Demonstrationszwecken entwickelt. Aus diesem Grund sind weitere Verbesserungen im Bereich der GUI notwendig.

Anhang A

brainFiler

Unser System baut auf der Funktionalität des dezentralen Dokumentmanagementsystems *brainFiler* der Firma *brainbot*¹ auf.

A.1 Systembeschreibung

„Der brainFiler ist eine dezentrale Wissenmanagement-Lösung. Er hilft dem Anwender sein Wissen zu strukturieren und es im Netzwerk allen Beteiligten zur Verfügung zu stellen.“ [3]

Der brainFiler ermöglicht die Verwaltung und Strukturierung von Dokumenten aus verschiedenen Quellen (Textdateien, Emails). Die Struktur des Datenbestandes besteht aus hierarchisch geordneten Containern (genannt *Kategorie*) und ähnelt der Verzeichnisstruktur von Dateisystemen. Durch die bekannten Strukturen wird die Einarbeitung in das System erleichtert. Im Gegensatz zu normalen Dateisystemen ist es beim brainFiler jedoch möglich, Dokumente in mehrere Kategorien einzuordnen, ohne dass Dubletten der Dateien erstellt werden müssen.

brainFiler bietet außerdem die Möglichkeit mehrere Taxonomien anzulegen, dadurch wird dem Nutzer eine multikriterielle Ablage und Suche bereitgestellt. Das System kann dem Benutzer bei der Erstellung von Taxonomien und Einordnen von Dokumenten unterstützen, in dem es Vorschläge macht, zu welchen

¹<http://www.brainbot.de>

Kategorien ein neu einzuordnendes Dokument inhaltlich passen würde (siehe Abb. A.1).

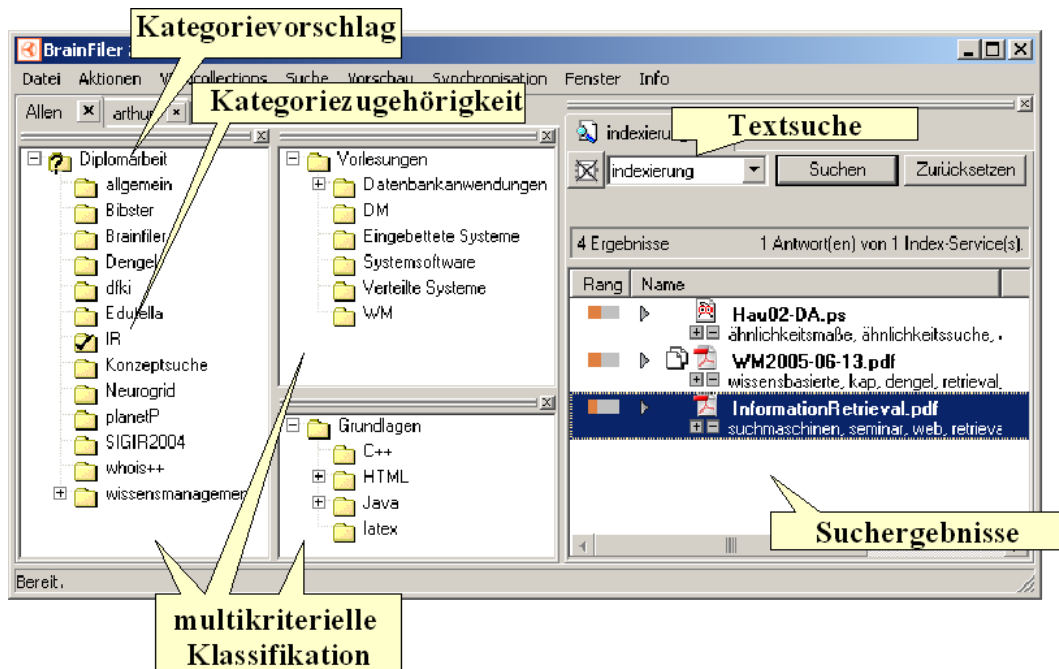


Abbildung A.1: Brainfiler-GUI

A.2 brainFiler-API

Unter dem Paket `de.dfki.util.classification.document` sind Klassen zusammengefasst, die die Nutzung der `brainFiler`-Funktionen in Java-Programmen ermöglichen.

Die wichtigsten Klassen bzw. Interfaces des Paketes sind:

<code>DocumentClassificationStore</code>	Zugriff auf <i>brainFiler</i> -Datenbasis, Klassifizierung von Dokumenten zu Kategorien, umfangreiche Suchfunktionen
<code>StoreId</code>	eindeutiger Bezeichner für eine Datenbasis
<code>Document</code>	Methoden zum Zugriff auf Dokumente
<code>Category</code>	Methoden zum Zugriff auf Kategorien
<code>DocumentSet</code>	Liste von Dokumenten
<code>CategorySet</code>	Liste von Kategorien
<code>Query</code>	Anfragebeschreibung
<code>QueryResult</code>	Anfrageergebnisse
<code>RankedItemList</code>	geordnete Liste bestehend aus <code>RankedItem</code> -Elementen

Wir wollen kurz die wichtigsten Klassen und Methoden vorstellen, die für die Suchalgorithmen relevant sind. Genaue Informationen über die Klassen und deren Verwendung findet man in der Dokumentation.

A.2.1 Klasse Document

Dokumente sind die grundlegenden Informationsträger im *brainFiler*. Diese Klasse stellt Methoden zum Zugriff auf Inhalt und Eigenschaften eines Dokuments bereit.

Man kann ausgehend von einem Dokument eine Konzeptsuche starten. Diese sucht zu einem Dokument passende Kategorien.

```
RankedItemList getRankedCategoryProposal(float threshold)
RankedItemList getRankedCategoryProposal(StoreId store,
                                          float threshold )
```

Das Ergebnis ist eine gerankte Liste von Kategorien.

A.2.2 Klasse Category

Kategorien sind Container mit Dokumenten. Die Klasse besitzt Methoden zum Zugriff auf die Eigenschaften und zum Hinzufügen und Entfernen von Dokumenten.

Die Klasse stellt auch zwei Methoden zum Suchen von ähnlichen Kategorien bereit.

```
RankedItemList getRankedCategoryProposal(float threshold)
RankedItemList getRankedCategoryProposal(StoreId store,
                                          float threshold )
```

Das Ergebnis ist eine gerankte Liste von Kategorien.

A.2.3 Klasse DocumentClassificationStore

Die Klasse *DocumentClassificationStore* stellt den Zugriff auf die *brainFiler*-Instanzen her. Sie ermöglicht Dokumente in taxonomischen Strukturen zu speichern und erlaubt einen einfachen Zugriff auf die Daten.

Suchanfragen

Die Methode `executeQuery` stellt umfangreiche Möglichkeiten zur Dokumentensuche auf dem *brainFiler* Datenbestand bereit.

```
QueryResult executeQuery(Query)
```

Die Methode führt eine Anfrage nach Dokumenten aus. Die Anfrage wird durch die Klasse `Query` (siehe Abschnitt A.2.4) beschrieben. Als Ergebnis liefert die Funktion ein Objekt der Klasse `QueryResult`. Diese enthält eine geordnete Liste von Dokumenten vom Typ `RankedItemList` (siehe Abschnitt A.2.5).

Dokumentsuche über Text: Die Textsuche nach Dokumenten wie sie in Kapitel 5 benutzt wird, kann man folgendermaßen realisieren.

```
\\ Erstelle neues Anfrageobjekt
Query query = new Query();
\\ Setze den Anfragetext
query.setParameter(Query.QUERY_TEXT, "Suchtext");
\\ Lege fest auf welchen Peers gesucht werden soll
query.setParameter(Query.Query_STORES, knownPeers);
\\ Setze die Anfrage über den Store m_store ab
QueryResult results = m_store.executeQuery(query);
```

Dokumentsuche über Kategorie: Die Konzeptsuche nach Dokumenten wie sie in Kapitel 5 benutzt wird, kann man folgendermaßen realisieren.

```
\\ Erstelle neues Anfrageobjekt
Query query = new Query();
\\ Setze Anfragekategorien
query.setParameter(Query.QUERY_CATEGORIES, queryCategories);
\\ Lege fest auf welchen Peers gesucht werden soll
query.setParameter(Query.Query_STORES, knownPeers);
\\ Setze die Anfrage über den Store m_store ab
QueryResult results = m_store.executeQuery(query);
```

A.2.4 Klasse Query

Die Klasse `Query` dient der Beschreibung einer Anfrage nach Dokumenten. Die Klasse enthält folgende Methoden:

```
void setParameter(Parameter, Value)

Object getParameter(Parameter)

void setDefault()
```

Die Anfrage wird durch die Methode `setParameter` modifiziert. Die wichtigsten Parameter sind:

Parameter	Typ	Funktion
<code>QUERY_CATEGORIES</code>	<code>Category_Set</code>	Menge von Kategorien, zu denen ähnliche Dokumente gesucht werden sollen.
<code>QUERY_MAX_RESULT_NUMBER</code>	<code>Integer</code>	Anzahl der maximalen Suchergebnisse
<code>QUERY_SIMILAR_DOCUMENTS</code>	<code>Document</code>	Suche ähnliche Dokumente zu einem Dokument
<code>QUERY_STORES</code>	<code>Collection</code>	Menge von Peerbezeichner zur Eingrenzung des Suchraums
<code>QUERY_TEXT</code>	<code>String</code>	Suche Dokumente über Text

A.2.5 Klasse RankedItemList

Die Ergebnisse der `brainFiler`-Anfragen sind gerankte Listen, diese werden in diesem Paket durch die Klasse `RankedItemList` repräsentiert. Die `RankedItemList` ist eine Liste, die Objekte (Dokumente, Kategorien) zusammen mit einem `double`-Wert speichert. Die Liste ist in absteigender Reihenfolge sortiert.

Abbildungsverzeichnis

2.1	Der Information Retrieval Prozess nach [16]	8
2.2	Anfrage- und Dokumentvektor im Raum	10
2.3	Indexierung eines Dokuments aus [18]	12
2.4	Invertierter Index mit Termgewichten	14
2.5	NeuroGrid: Weiterleitung der Suchanfragen	24
2.6	Bibster: kompetenzabhängiger Vergleich	26
3.1	Semantische Topologie	30
3.2	Use Cases	31
4.1	Das C2CIR-Netzwerk	37
4.2	Hierarchisch geordnete Konzepte	38
4.3	Hopping auf Konzeptbasis	45
5.1	C2CIR: System-Design	49
5.2	Der C2CIR-Peer	52
5.3	Kompetenzen werden in einer speziellen Kategorie „Kompetenzen“ gespeichert	54
5.4	anfrageunabhängige Konfidenzmatrix	55
5.5	anfrageabhängige Konfidenzmatrix	56

5.6	C2CIR: Anfragebearbeitung	59
5.7	Aufteilung der C2CIR-GUI	68
5.8	C2CIR-GUI: Browser-Komponente	69
5.9	C2CIR-GUI: Anfrageparameter	70
5.10	C2CIR-GUI: Anzeigen des Speicherorts	71
5.11	C2CIR: Kategorieergebnisse und Feedback	72
5.12	C2CIR-GUI: Dokumentenergebnisse	73
A.1	Brainfiler-GUI	78

Literaturverzeichnis

- [1] BAEZA-YATES, Ricardo A. ; RIBEIRO-NETO, Berthier A.: *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999. – ISBN 0–201–39829–X
- [2] BLOOM, B. H.: Space/time trade-offs in hash coding with allowable errors. In: *Communications of the ACM* 13(7) (1970), S. 422–426
- [3] BRAINBOT. *brainFiler – Kurzbeschreibung*. http://brainbot.com/site3/dokumente/brainFiler_Kurzbeschreibung.pdf. Juni 2004
- [4] BROEKSTRA, Jeen ; EHRIG, Marc ; HAASE, Peter ; HARMELEN, Frank van ; MENKEN, Maarten ; MIKA, Peter ; SCHNIZLER, Björn ; SIEBES, Ronny: Bibster - A Semantic-Based Bibliographic Peer-To-Peer System. In: *Proceedings of the WWW'04 Workshop on*, 2004
- [5] CUENCA-ACUNA, Francisco M. ; PEERY, Christopher ; MARTIN, Richard P. ; NGUYEN, Thu D.: PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities. In: *Twelfth IEEE International Symposium on High Performance Distributed Computing (HPDC-12)*, IEEE Press, Juni 2003
- [6] DENGEL, Andreas ; ABECKER, Andreas ; BÄHR, Jan-Thies ; BERNARDI, Ansgar ; DANNENMANN, Peter ; ELST, Ludger van ; KLINK, Stefan ; MAUS, Heiko ; SCHWARZ, Sven ; SINTEK, Michael: *EPOS - Evolving Personal to Organizational Knowledge Spaces*. 2002. – Project Proposal, DFKI GmbH
- [7] FERBER, Reginald: *Information retrieval*. 1. Aufl. dpunkt, 2003. – ISBN 3–89864–213–5

-
- [8] HAASE, P. ; SIEBES, R. ; HARMELEN, F. van: Peer Selection in Peerto-Peer Networks with Semantic Topologies. In: *International Conference on Semantics of a Networked World: Semantic Grid Databases*. Paris, Juni 2004
- [9] HAASE, Peter ; EHRING, Marc ; HOTH, Andreas ; SCHNIZLER, Björn: Personalized Information Access in a Bibliographic Peer-to-Peer System. In: *Proceedings of the AAAI Workshop on Semantic Web Personalization*, AAAI Press, Juli 2004, S. pp. 1–12
- [10] HAASE, Peter ; STOJANOVIC, Nenad ; SURE, York ; VÖLKER, Johanna: On Personalized Information Retrieval in Semantics-Based Peer-to-Peer Systems. In: *Proceedings of the BTW-Workshop "WebDB Meets IR"*, 2005
- [11] HARDY, D. ; SCHWARTZ, M. *Harvest user's manual*. 1995
- [12] JOSEPH, Sam: NeuroGrid: Semantically Routing Queries in Peer-to-Peer Networks. In: *Proceedings of the International Workshop on Peer-to-Peer Computing (co-located with Networking 2002)*. Pisa, Italy, Mai 2002
- [13] KUHLEN, Rainer: *Experimentelle Morphologie*. München: Verlag Dokumentation, 1977
- [14] LEWIS, D. D. *Reuters-21578 text categorization test collection*. <http://www.daviddlewis.com/resources/testcollections/reuters21578/>
- [15] LUHN, H. P.: The Automatic Creation of Literature Abstracts. In: *I. B. M. Journal of research and Development* 2(2) (158), S. 159–165
- [16] MEYER-WEGENER, Klaus: *Multimediale Datenbanken : Einsatz von Datenbanktechnik in Multimedia-Systemen*. 2. überarb. und erw. Wiesbaden : B.G. Teubner, 2003 (Leitfäden der Informatik). – ISBN 3–519–12419–X
- [17] MILGRAM, S.: The Small World Problem. In: *Psychology Today* (1967), Mai, S. 60 – 67
- [18] RUTHVEN, I. ; LALMAS, M.: A survey on the use of relevance feedback for information access systems. In: *Knowledge Engineering Review* 18 (2003), Nr. 2, S. 95–145. – ISSN 0269–8889

-
- [19] SCHENK, Simon: *Neuer Ansatz im Wissensmanagement: Unterstützung von sozialen Netzwerken durch Wissenstauschbörsen auf Basis von Peer-to-Peer-Technologie*, Fh Nordakademie, Diplomarbeit, 2004
- [20] SCHENK, Simon: Introducing Social Aspects to Search in Peer-to-Peer Networks. In: *Wissensmanagement*, 2005, S. 217–221
- [21] ŞİMŞEK, Özgür ; JENSEN, David: Decentralized search in networks using homophily and degree disparity. In: *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, 2005
- [22] STOICA, Ion ; MORRIS, Robert ; KARGER, David ; KAASHOEK, M. F. ; BALAKRISHNAN, Hari: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In: *Proceedings of the ACM SIGCOMM '01 Conference*. San Diego, California, August 2001