

Entwicklung von Verfahren zur Clusterbildung in einem Graphen für die Vorabübertragung von Webseiten

Diplomarbeit im Fach Informatik

vorgelegt von

Timo Pfahl

geb. am 01. August 1979 in Bietigheim-Bissingen

angefertigt am

**Institut für Parallele und Verteilte Systeme
Universität Stuttgart**

Betreuerin: Dipl. Inf. Susanne Bürklen

Beginn der Arbeit: 01. Juni 2005

Abgabe der Arbeit: 01. Dezember 2005



Zusammenfassung

Aus unserem täglichen Leben sind mobile Endgeräte wie Mobiltelefone oder PDAs nicht mehr wegzudenken. Durch die zunehmende Verbreitung von Funknetzen ist der mobile Zugriff auf das Internet mit solchen Geräten an immer mehr Orten möglich. Von einer flächendeckenden Versorgung ist man jedoch noch weit entfernt. Eine Möglichkeit, dem Benutzer auch in Zeiten ohne Netzanbindung den Zugriff auf Inhalte des Internets bieten zu können, besteht darin, Webseiten in Phasen guter Netzanbindung vorab auf das Endgerät zu übertragen. Die Herausforderung dieser Vorgehensweise, im englischen Hoarding genannt, liegt darin, eine gute Vorhersage über zu erwartendes Benutzerverhalten zu treffen. Hierzu wird das vergangene Benutzerverhalten beobachtet und ausgewertet. Auf Basis dieser Auswertung erfolgt dann eine Vorhersage über das zukünftig erwartete Verhalten.

Im Rahmen dieser Arbeit wird ein solches Verfahren zur Vorhersage zukünftigen Zugriffsverhaltens auf Webseiten erarbeitet. In einem existierenden Verfahren wird das vergangene Zugriffsverhalten verschiedener Benutzer in einer Graphstruktur repräsentiert. Die Auswahl der Webseiten erfolgt dann durch eine Traversierung des Graphen nach dem Tiefen- oder Breitensuchprinzip. In dieser Arbeit wird ein Verfahren erstellt, das durch Clusterbildung jene Webseiten in diesem Graphen zu Informationseinheiten zusammenfasst, zwischen denen eine semantische Beziehung besteht. Das bedeutet, dass wenn eine Webseite eines Clusters aufgerufen wird, es sehr wahrscheinlich ist, dass auch auf die anderen enthaltenen Seiten zugegriffen wird. Um den Speicherplatz der Endgeräte effizienter zu nutzen, werden Cluster dann entweder komplett oder gar nicht vorab übertragen. Dadurch wird erwartet, dass eine bessere Auswahl der vorab zu übertragenden Webseiten getroffen wird.

Die Idee hinter der Clusterbildung basiert darauf, dass Benutzer beim Durchsuchen des Internets häufig eine Folge von Webseiten aufrufen, bis die gesuchte Information gefunden wurde. Dies ermöglicht eine Unterteilung der Seiten in so genannte Navigations- und Inhaltsseiten. Auf Basis dieser Unterteilung werden dann im Graphen Pfade von Webseiten gesucht, die eine solche Suche repräsentieren. Die einzelnen Seiten eines solchen so genannten Suchpfades werden dann zu einem Cluster zusammengefasst.

Es wird gezeigt, dass durch die Clusterbildung und -bewertung eine sehr viel bessere Vorhersage über zukünftiges Benutzerverhalten getroffen werden kann. Auf Basis von Bewertungsmetriken, die zur Ermittlung der Zufriedenheit der Benutzer dienen, werden bis zu dreimal bessere Ergebnisse erzielt als für das existierende Tiefen- und Breitensuchverfahren.



Inhaltsverzeichnis

1	Einführung	1
1.1	Einleitung und Motivation	1
1.2	Kapitelübersicht	3
2	Grundlagen	5
2.1	Systemmodell und existierende Verfahren	5
2.1.1	Systemmodell	5
2.1.2	Bisheriges Verfahren	7
2.1.3	Erwartete Vorteile durch Clusterbildung	7
2.2	Datenstrukturen	8
2.2.1	Aufbau der Logdateien	8
2.2.2	Ableitbare Attribute für Logdateien	9
2.2.3	Semantik und Struktur des Informationsgraphen	11
2.3	Anforderungen an Clusterverfahren	14
2.4	Taxonomie	15
2.4.1	Partitionierende und Hierarchische Verfahren	16
2.4.2	Clusterbildung in Graphen	17
2.5	Verwandte Arbeiten	18
2.5.1	SEER	21
3	Clusteransätze	25
3.1	Allgemeines zur Vorgehensweise	25
3.1.1	Kriterien zur Clusterbewertung	26
3.2	Intuitive Ansätze	26
3.2.1	Clustersemantik	27
3.2.2	Tiefenansatz	28
3.2.3	Aufwandsverminderung	30
3.2.4	Clusterbewertung	32
3.2.5	Vor- und Nachteile	33
3.2.6	Breitenansatz	33
3.3	Aufwärtsansatz	34
3.3.1	Clustersemantik	34
3.3.2	Einfache Version	34

3.3.3	Bewertung der Cluster	36
3.3.4	Aufwandsverminderung	37
3.3.5	Detaillierter Aufwärtsansatz	40
3.3.6	Vor- und Nachteile	42
4	Clusterverfahren	43
4.1	Semantik der Cluster	43
4.2	Clusterstruktur	44
4.2.1	Inhaltseigenschaft von Knoten	44
4.2.2	Aufbau eines Suchpfades	44
4.2.3	Festlegung der Clusterstruktur	45
4.3	Bewertung der Cluster	48
4.3.1	Größe und inneres Gewicht	48
4.3.2	Aufstellen der Bewertungsfunktion	49
4.4	Entwicklung des Algorithmus	50
4.4.1	Eingrenzung des Aufwands	50
4.4.2	Semantische Distanzen	52
4.4.3	Verwendung einer Warteschlange	54
4.4.4	Algorithmus	54
5	Vorabübertragungsliste	59
5.1	Erstellung	59
5.1.1	Problematik der Clusterüberlappung	59
5.1.2	Optimierung des Aufwands	61
5.1.3	Effiziente Verfahren	62
5.1.4	Inkrementeller Ansatz	64
5.2	Bewertung	66
6	Evaluierung	69
6.1	Verwendetes Simulationsmodell - User Centric Walk	69
6.2	Metriken zur Verfahrensbewertung	69
6.2.1	Interpretation der Logdateien zur Evaluierung	70
6.2.2	Herleitung der Metriken	71
6.3	Evaluierungsaufbau	75
6.4	Auswertung	76
6.4.1	Bestimmung der besten Parameterwerte	76
6.4.2	Vergleich mit bisherigem Verfahren	90
6.4.3	Fazit	92
7	Zusammenfassung	95
7.1	Erkenntnisse dieser Diplomarbeit	95
7.2	Ausblick	96

A Implementierung	97
A.1 Struktur der Klassen	97
A.2 Beschreibung der Schnittstellen	98
B Begriffslexikon	103
Literaturverzeichnis	105

Abbildungsverzeichnis

2.1	Systemmodell	6
2.2	Beispiel - Logdatei	11
2.3	Beispiel - Informationsgraph	13
2.4	Überlappung von Clustern	14
2.5	Taxonomie (Jain et al.)	16
2.6	Hierarchisches Verfahren - Ergebnis (Jain et al.)	17
3.1	Clustersemantik - Beispiel	27
3.2	einfacher Tiefenansatz - Beispiel	29
3.3	schlechtester Fall - Tiefenansatz	30
3.4	Verwendung einer Knotenliste	31
3.5	Clusterbewertung - Problem	32
3.6	Aufwärtsansatz - Beispiel	36
3.7	Clusterbewertung - Inneres Gewicht	37
3.8	Aufwärtsansatz - Zyklenproblem	38
3.9	Summe aller Pfadgewichte - Beispiel	39
3.10	Detaillierter Aufwärtsansatz - Beispiel	41
4.1	Suchpfade - Beispiel	45
4.2	Wo beginnt ein Cluster?	46
4.3	Wo endet ein Cluster?	47
4.4	Visualisierung Steigungsansatz	50
4.5	Wahl von w_{th}	53
4.6	Ablauf des Clusteralgorithmus	56
5.1	Beispiel für Bewertungsänderung von Clustern	60
5.2	Ablauf des Bubble Sort-Verfahrens	63
5.3	Inkrementeller Ansatz - Nachteil	65
5.4	Berücksichtigung der Anzahl der Inhaltsknoten	67
6.1	Arten von Logdateien - Beispiel	71
6.2	Bewertung mittels Logdatei - Beispiel	72
6.3	Logsuchpfad - Beispiel	73
6.4	Laufzeitvergleich: Heap- und Bubble Sort-Verfahren	77

6.5	Ergebnisse: Inhaltswahrscheinlichkeit	79
6.6	Ignorieren der Inhaltswahrscheinlichkeit	80
6.7	Ergebnisse: Minimales Kantengewicht bei statischer semantischer Distanz	81
6.8	Ergebnisse: Minimales Kantengewicht bei dynamischer semantischer Distanz	83
6.9	Laufzeitvergleich: Minimales Kantengewicht (Dynamische sem. Distanz)	83
6.10	Bewertungsvergleich: Statische und Dynamische semantische Distanz	84
6.11	Laufzeitvergleich: Statische und Dynamische semantische Distanz	85
6.12	Bewertungen: Minimales Pfadgewicht	86
6.13	Laufzeitverhalten: Minimales Pfadgewicht	86
6.14	Inhaltsseiten-Vollständigkeit für variierende α und β	88
6.15	Suchpfad-Vollständigkeit für variierende α und β	88
6.16	Indirekte Gewichtung von innerem Gewicht zur Clustergröße	89
6.17	Verhalten bei großen Vorabübertragungslisten	89
6.18	Existierendes Breiten- und Tiefensuchverfahren - Ergebnis	90
6.19	Verfahrensvergleich anhand Inhaltsseiten-Vollständigkeit	91
6.20	Verfahrensvergleich anhand Suchpfad-Vollständigkeit	92
6.21	Einfluss der Faktoren auf Ergebnisverbesserung	93
A.1	Klassendiagramm	98

Kapitel 1

Einführung

In diesem Kapitel wird eine Einleitung und Vorstellung des Themas gegeben. Zunächst wird auf die heutige Bedeutung drahtloser Kommunikation eingegangen. Darauf folgt eine Vorstellung der Grundidee des Hoarding und dessen Alternativen. Anschließend wird kurz das existierende Verfahren zur Vorabübertragung von Webseiten erklärt, auf das in dieser Arbeit aufgebaut wird. Daraufhin wird die Motivation zu dieser Diplomarbeit gegeben, indem erläutert wird, welches Ziel mit der Clusterbildung verfolgt wird und was für das Erreichen dieses Ziels zu tun ist. Abschließend wird eine Kapitelübersicht über die Struktur dieser Diplomarbeit aufklären.

1.1 Einleitung und Motivation

Aus unserem alltäglichen Leben sind mobile Endgeräte wie etwa Mobiltelefone, Notebooks oder PDAs nicht mehr wegzudenken. Mit ihrer zunehmenden Verbreitung erschließen sich immer wieder neue Anwendungsgebiete wie beispielsweise SMS (engl. *short message system*), Navigationssysteme oder der mobile Zugriff auf das Internet.

Es existieren heute mehrere Möglichkeiten, wie diese mobilen Geräte drahtlos miteinander kommunizieren oder auf das Internet zugreifen können. Das gegenwärtige Problem ist, dass solche Funkverbindungen nicht immer stabil sind, eventuell keine genügend hohe Bandbreite bieten oder an manchen Orten gar nicht erst zur Verfügung stehen. Als Beispiel lässt sich dies an der WLAN-Technologie (engl. *wireless local area network*) demonstrieren, die in letzter Zeit eine große Verbreitung erfuhr. Benutzern mobiler Endgeräte wird es ermöglicht, sich durch sogenannte *Access Points* beispielsweise in das Internet einzuklinken. Zwar finden sich öffentliche Access Points mittlerweile an immer mehr Orten, von einer flächendeckenden Versorgung, die von jedem beliebigen Ort eine Anbindung an das Internet ermöglicht, ist man aber noch weit entfernt.

Es bieten sich drei Möglichkeiten an, wie in Gebieten ohne Netzanbindung verfahren werden kann:

1. Schlichter Verzicht auf eine Netzanbindung und ausschließlich lokales Arbeiten auf dem Gerät selbst.

2. Versuch mittels anderer Techniken, wie beispielsweise WWAN (engl. *wireless wide area network*), eine Netzanbindung zu erhalten.
3. Bei guter Netzanbindung wird eine Vorabübertragung von solchen Informationen durchgeführt, die der Benutzer in Zeiten ohne Netzanbindung vermutlich anfordern wird (engl. *Hoarding*).

Da heutzutage Daten häufig nicht lokal gespeichert werden, um sie beispielsweise mittels Server-Strukturen mehreren Benutzern gleichzeitig zugänglich zu machen, ist Punkt 1 in vielen Situationen nicht tragbar. Neben den technischen Voraussetzungen, die für Punkt 2 erfüllt sein müssen, kommt hinzu, dass anderweitige Verbindungsarten eventuell sehr langsam, teuer oder gar beides zugleich sind. So bleibt mit der Verwendung von Hoarding die Möglichkeit, eine Netzanbindung zu simulieren, indem die angeforderten Informationen bereits im Voraus lokal gespeichert wurden. Die Herausforderung, die es auch in dieser Arbeit zu meistern gilt, liegt darin, eine möglichst gute Vorhersage über die zukünftig benötigten Informationen zu treffen. Dies ist notwendig, da sowohl der Speicherplatz auf dem Endgerät als auch die verfügbare Zeit zur Vorabübertragung begrenzt ist.

Das existierende Verfahren für diese Vorhersage, auf das diese Arbeit aufbaut, ist sehr einfach aufgebaut. Es arbeitet auf einer Graphstruktur, die das Zugriffsverhalten von Benutzern auf Webseiten charakterisiert. Der Graph wird aus der Analyse vergangenen Benutzerverhaltens aufgebaut. Die Vorhersage erfolgt durch eine Traversierung nach dem Tiefen- oder Breitensuchprinzip. Dabei werden angetroffene Graphknoten direkt in die Liste der vorab zu übertragenden Webseiten übernommen, wobei jeder Knoten für eine Webseite steht. Da der Graph gewichtete Kanten besitzt, kann aufgrund dieser Eigenschaft bei der Traversierung die Reihenfolge der Knotenbesuche etwas beeinflusst werden. Weitere Möglichkeiten zu Vergleich und Sortierung von Knoten existieren hier jedoch nicht.

Durch die Verwendung von Clusterbildung wird nun erwartet, dass Knoten des Graphen, die semantisch zusammengehören, gruppiert werden können. Was unter dieser Semantik genau zu verstehen ist und wie die Clusterbildung in einem Algorithmus ablaufen kann, ist Teil dieser Diplomarbeit. Ziel ist es, dass fertige Cluster eine Menge von Webseiten enthalten, für die folgendes gilt:

Wird eine im Cluster enthaltene Seite angefordert, dann ist es wahrscheinlich, dass auch auf die anderen Seiten dieses Clusters zugegriffen wird.

Bei der Auswahl der vorab zu übertragenden Seiten wird ein Cluster dann entweder komplett übertragen oder gar nicht. Dadurch wird der Speicherplatz auf dem Endgerät effizienter genutzt, da einzelne, verstreute Seiten nicht übertragen werden. Die Auswahl der zu übertragenden Cluster erfolgt durch eine Clusterbewertung und den damit ermöglichten Vergleich zwischen Clustern. Die Bewertung kann auf verschiedenen Clustereigenschaften basieren und ist damit wesentlich flexibler als die Besuchsreihenfolge der Knoten, die beim bisherigen Verfahren die Auswahlreihenfolge bestimmt. Die Festlegung der Clustereigenschaften sowie die Erarbeitung dieser Bewertung erfolgt ebenfalls in dieser Diplomarbeit.

Resultierend wird erwartet, dass durch die Clusterbildung und -bewertung die Trefferrate der Vorhersage gegenüber den bisherigen Verfahren steigt. Dies bedeutet, dass Benutzer,

nur unter Verwendung der vorab übertragenen Menge von Webseiten, einen größeren Anteil der gewünschten Webseiten in dieser Menge vorfinden als mit dem existierenden Breiten- und Tiefensuchverfahren. Die Bestätigung dieser Erwartung wird im Rahmen dieser Arbeit geliefert.

1.2 Kapitelübersicht

Die Kapitel dieser Diplomarbeit wurden folgendermaßen strukturiert:

- In Kapitel 2 werden zunächst das Systemmodell und das Verfahren, auf welches aufgebaut wird, vorgestellt. Danach wird auf die zu Grunde liegenden Datenstrukturen eingegangen, auf denen das Clusterverfahren aufbauen muss. Nach der Feststellung der Anforderungen an das Clusterverfahren wird ein Überblick über Clustertechniken allgemein und verwandte Arbeiten gegeben.
- Die Clusteransätze, die im Fortschritt dieser Arbeit nach und nach erarbeitet werden und welche durch die aus ihnen gewonnenen Erkenntnisse als Grundlage für das letztendlich verwendete Verfahren dienen, werden in Kapitel 3 vorgestellt.
- In Kapitel 4 wird das Verfahren vorgestellt, das letztendlich angewendet wird. Es wird detailliert auf die Semantik und Struktur der zu erstellenden Cluster eingegangen. Nachfolgend wird die Bewertung der Cluster definiert und der verwendete Algorithmus vorgestellt.
- Vorgehensweisen zur Erstellung und Bewertung der Vorabübertragungsliste finden sich in Kapitel 5.
- In Kapitel 6 werden zunächst das verwendete Simulationsmodell sowie die Metriken zur Bewertung des Verfahrens aus Sicht des Benutzers erläutert. Anschließend erfolgt die Beschreibung des Evaluierungsaufbaus. Schließlich folgen die Ergebnisse der Evaluierung, die Informationen über die besten Parameterkombinationen enthalten und einen Vergleich des Clusterverfahrens mit den bisherigen Verfahren bieten.
- Eine abschließende Zusammenfassung sowie ein Ausblick erfolgen in Kapitel 7.
- Im Anhang befinden sich Erläuterungen zur Implementierung und ein Begriffslexikon.

Kapitel 2

Grundlagen

In diesem Kapitel wird zunächst ein kurzer Überblick über das Systemmodell gegeben und auf die grundlegenden Datenstrukturen eingegangen, auf die das Clusterverfahren aufzubauen ist. Anschließend werden die Anforderungen an jenes Verfahren festgehalten. Die Erläuterung der Anforderungen ermöglicht es, die Clusterverfahren verwandter Arbeiten auf ihre Anwendbarkeit auf dieses Problem hin zu untersuchen. Hierzu wird zuerst ein Allgemeinüberblick über existierende Clusterverfahren gegeben, bevor konkret auf einzelne Arbeiten eingegangen wird.

2.1 Systemmodell und Überblick über das bisherige Verfahren

Diese Arbeit wurde im Rahmen des Sonderforschungsbereichs Nexus [22] erstellt. Kurz zusammengefasst, ist es das Ziel von Nexus, Weltmodelle zu definieren und realisieren, die in kontextbezogenen Anwendungen verwendet werden können. Hierzu werden Methoden bereitgestellt, welche die Verwaltung solcher Modelle sowie die kontextbezogene Kommunikation unterstützen. Das Prinzip des Hoarding, das in Kapitel 1 kurz erklärt wurde, stellt eine solche Methode zur Unterstützung kontextbezogener Kommunikation dar.

In diesem Abschnitt wird zunächst das Systemmodell zur Versorgung der Endgeräte mit Daten erklärt, das im Rahmen des von Nexus verwendeten Hoarding benutzt wird und auf das auch diese Arbeit aufbaut. Danach wird das bisher verwendete Verfahren zur Vorabübertragung genauer beleuchtet, woraufhin erläutert wird, warum durch Clusterbildung eine Verbesserung dieses Verfahrens erwartet wird.

2.1.1 Systemmodell

Grundlegend wird die Annahme getroffen, dass beim Zugriffsverhalten mobiler Benutzer auf Webseiten ein Lokationsbezug zwischen Webseite und geographischem Ort des Aufrufes besteht. Zum Beispiel ist zu erwarten, dass die Webseite der Stuttgarter Staatsgalerie besonders häufig in Bereichen nahe dieser Galerie angefordert wird.

Die zentrale Einheit des Systemmodells ist die *Infostation* (englische Aussprache), welche mit den mobilen Geräten kommuniziert und Daten zur Verfügung stellt. Durch den Lokationsbezug ist jede Infostation nur für die Webseitenaufrufe in einem gewissen geographischen

Bereich zuständig, dem so genannten *Hoarding-Gebiet*. In Abb. 2.1, die in leicht abgeänderter Form [3] entnommen wurde, ist dies grafisch aufgezeigt. Die Entscheidung, welche Webseiten an anfragende Endgeräte übertragen werden sollen, trifft jede Infostation auf Basis der ausschließlich in ihrem Hoarding-Gebiet erfolgten Aufrufe in der Vergangenheit. Informationen über vergangenes Benutzerverhalten werden der Infostation von jedem Endgerät, in Form von Logdateien, beim Betreten des *Übertragungsgebiets* mitgeteilt (*Transmission Area* in Abb. 2.1). Nur im Übertragungsgebiet besteht eine Funkverbindung zwischen Endgerät und Infostation, wodurch sämtlicher Informationsaustausch in diesem Gebiet vollzogen werden muss. Um die Kommunikation mit dem Endgerät und die Vorabübertragung zu ermöglichen, besitzt jede Infostation eine Internetanbindung sowie eine Funknetzchnittstelle nach Standard 802.11 [12]. Detailliertere Ausführungen zur Idee hinter Infostations können [9] entnommen werden.

Ein Endgerät muss neben einem Webbrowser über die benötigte Hoarding-Anwendung verfügen und einen bestimmten Speicherplatz, den so genannten *Hoardcache*, für die vorab übertragenen Seiten bieten. Zudem protokolliert jedes Gerät seine Webseitenaufrufe und verfügt über eine Positionsbestimmung. Somit können Logdateien erstellt werden, die jede Anforderung einer Webseite samt Ort und Zeitpunkt festhalten. Diese Logdateien werden dann, wie erwähnt, von den Infostations benutzt, um das Benutzerverhalten in der Vergangenheit zu analysieren.

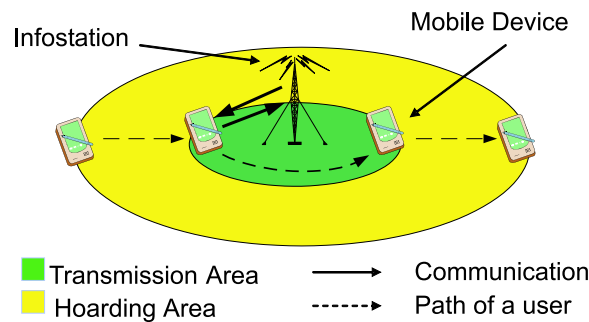


Abbildung 2.1: Systemmodell

Zwischen dem Betreten und Verlassen eines Hoarding-Gebiets spielt sich folgendes ab (siehe Abb. 2.1):

1. Der Benutzer betritt das Hoarding-Gebiet der Infostation. Zu diesem Zeitpunkt ist der Hoardcache des Endgeräts entweder noch leer oder mit den Seiten der vorigen Infostation gefüllt.
2. Beim Betreten des Übertragungsgebiets übermittelt das Endgerät der Infostation seine gegenwärtige Logdatei. Die Infostation, welche die Folgen von Webseitenaufrufen aus den empfangenen Logdateien in einer Graphstruktur speichert, aktualisiert diesen Graphen. Hierbei werden nur Logeinträge berücksichtigt, deren Ortseintrag innerhalb des Hoarding-Gebiets der Infostation liegt. Anschließend erfolgt die Vorabübertragung ausgewählter Webseiten. Die Auswahl dieser Seiten wird in periodischen Abständen

anhand des jeweils aktuellen Graphen getroffen. Die komplette Kommunikation muss geschehen, während der Benutzer sich im Übertragungsgebiet aufhält. In diesem Gebiet verfügt das Endgerät durch die WLAN-Anbindung an die Infostation auch über eine direkte Internetanbindung und muss hier somit nicht auf den Hoardcache zurückgreifen.

3. Nach Verlassen des Übertragungsgebiets besitzt der Benutzer keine Internetanbindung mehr und ist nun auf die vorab übertragenen Webseiten angewiesen.

2.1.2 Bisheriges Verfahren

Wie bereits angesprochen, speichert sich jede Infostation die aus empfangenen Logdateien gewonnenen Informationen über Webseitenaufrufe in einer Graphstruktur, dem so genannten *Informationsgraphen*. In diesem Graph steht jeder Knoten für eine Webseite. Die gerichteten Kanten sind gewichtet und geben jeweils die Wahrscheinlichkeit wieder, mit der die entsprechenden Webseiten der Start- und Endknoten direkt aufeinander folgend angefordert werden. Existiere zum Beispiel zwischen Knoten a und b eine Kante von a nach b mit Gewicht $0,3$. Dann wird ein Benutzer in 30% der Fälle nach dem Aufruf der Webseite von a , direkt die Seite von b anfordern. Nach jedem Eingang einer neuen Logdatei wird der Graph aktualisiert.

Die Auswahl der vorab zu übertragenden Webseiten wird periodisch mittels einer Traversierung nach dem Breiten- oder Tiefensuchprinzip, wie in [26] beschrieben, durchgeführt. Für dieses Verfahren gilt, dass die Nachfolger jedes Knotens in der absteigenden Reihenfolge der Gewichte der ausgehenden Kanten des Knotens besucht werden. Das bedeutet, dass von einem Knoten aus zunächst der direkte Nachfolger besucht wird, zu dem die Kante mit dem höchsten Gewicht existiert und so weiter. Wird ein Knoten bei der Traversierung angefasst, wird die entsprechende Webseite sofort an die *Vorabübertragungsliste* angehängt.

2.1.3 Erwartete Vorteile durch Clusterbildung

Zwar lässt sich durch die Tatsache, dass die Kanten im Informationsgraphen gewichtet sind sicherstellen, dass bei der Traversierung, wie unter Abschnitt 2.1.2 beschrieben, immer zunächst die höchstgewichtete Kante abgeschritten wird. Diese Berücksichtigung ist aber auch schon der einzige Indikator für die Beziehung zwischen Webseiten, der im vorgestellten Tiefen- und Breitensuchverfahren beachtet wird. Tauchen zum Beispiel häufig bestimmte Folgen von Webseitenaufrufen auf oder zeigen Benutzer an verschiedenen Knoten bzw. Webseiten ein bestimmtes Verhalten, so kann dies mit diesem Verfahren jedoch nicht genauer analysiert werden.

Ziel der Clusterbildung ist es nun, die Zusammengehörigkeit von Webseiten auf Basis einer zu erarbeitenden Semantik festzustellen, die Dinge wie Folgen von Webseitenaufrufen oder Benutzerverhalten bei bestimmten Seiten berücksichtigt. Mengen semantisch zusammengehörender Webseiten werden dann zu Clustern zusammengefasst. Anschließend soll bei der Erstellung der Vorabübertragungsliste für jeden Cluster gelten, dass er entweder komplett in die Liste aufgenommen wird oder gar nicht. Der Vorteil dieser Vorgehensweise ist, dass weniger Speicherplatz auf dem Endgerät durch einzelne, verstreute Seiten verschwendet wird. Die Semantik ist demnach so zu wählen, dass, wenn der Benutzer eine Seite des Clusters

anfordert, er wahrscheinlich auch die anderen Seiten des Clusters besuchen möchte. So soll erreicht werden, dass der Benutzer seltener auf nicht verfügbare Webseiten stößt.

Wie Cluster effizient aufgebaut werden können, welcher Semantik sie folgen und auf welcher Basis anschließend die Entscheidung für bestimmte Cluster zur Übernahme in die Vorübertragungsliste gefällt wird, wird in den folgenden Kapiteln erarbeitet.

2.2 Datenstrukturen

Wie bereits in Abschnitt 2.1 angeschnitten, verwaltet jede Infostation die Folgen von Webseitenaufrufen in einem Informationsgraphen, der durch die Analyse von Logdateien aufgebaut wird. Im folgenden wird die Struktur der Logdateien und des daraus aufgebauten Informationsgraphen formal definiert. Diese Definitionen stammen von Susanne Bürklen und wurden bereits schriftlich niedergelegt aber noch nicht veröffentlicht. Die für diese Arbeit benötigten Teile werden nachfolgend erläutert.

2.2.1 Aufbau der Logdateien

Die auf einem mobilen Endgerät erstellten Logdateien liefern den Infostations Informationen über die Webseitenaufrufe, die ein Benutzer im Hoarding-Gebiet in der Vergangenheit getätigt hat. Damit Infostations diese Daten verarbeiten können, müssen sie in einer definierten Struktur vorliegen.

Sämtliche Webseitenaufrufe eines Benutzers werden in einer Logdatei festgehalten, die wiederum aus einer Folge von Logeinträgen besteht.

Definition 1 *Ein Logeintrag ist ein Tupel $l = (ID, loc, time, size, action)$ wobei gilt:*

- *$l.ID$ ist die URL der aufgerufenen Webseite.*
- *$l.loc$ ist der geographische Ort, an dem der Aufruf getätigt wurde. Welche Art geographischer Koordinaten hierzu verwendet wird, ist für diese Arbeit unerheblich.*
- *$l.time$ ist der Zeitpunkt, zu dem der Aufruf erfolgte.*
- *$l.size$ ist die Größe der angeforderten Webseite in Bytes.*
- *$l.action$ ist die Art, auf welche die Webseite angefordert wurde. Mögliche Werte sind: $l.action \in \{jump, link, revisit\}$. “jump” bedeutet, dass der Benutzer die Seite direkt angefordert hat (z.B. durch Eingeben der URL oder die Auswahl eines Lesezeichens). “link” steht dafür, dass die Webseite durch das Klicken auf einen Link aufgerufen wurde. “revisit” bedeutet, dass die Seite innerhalb der Sitzung bereits einmal besucht wurde und nun erneut besucht wird (z.B. durch Betätigen des Zurück-Buttons im Browser).*

Mit der Definition eines Logeintrags kann nun auch die Struktur einer Logdatei formal festgelegt werden.

Definition 2 Eine *Logdatei* \mathcal{L} ist eine geordnete Folge von Logeinträgen

$$\mathcal{L} = \{l_1, \dots, l_i, l_{i+1}, \dots, l_n\}$$

für die gilt: $l_i < l_{i+1}$ mit $l_i < l_{i+1} \iff l_i.time < l_{i+1}.time$.

Eine Logdatei kann weiter in *Sitzungen* (engl. *Sessions*) unterteilt werden. Als eine Sitzung wird das Durchforsten des Internets ohne Unterbrechung bezeichnet. Das bedeutet, dass der Benutzer die Anwendung während einer Sitzung weder beendet noch eine unverhältnismäßig lange Pause zwischen zwei Seitenaufrufen einlegt, die darauf schließen lässt, dass sich der Benutzer in der Zwischenzeit etwas anderem gewidmet hat. Jede Logdatei enthält eine nach oben offene Zahl an Sitzungen, jedoch mindestens eine.

Wie genau die Erkennung von Sitzungsstart- und -endpunkten definiert ist, ist für diese Arbeit nicht weiter von Belang. Die Gewissheit, dass diese Erkennung existiert, soll hier genügen.

2.2.2 Ableitbare Attribute für Logdateien

Aus den Definitionen 1 und 2 lassen sich weitere Attribute für die Einträge von Logdateien ableiten. Diese zusätzlichen Attribute werden grundsätzlich deshalb definiert, um aus dem Benutzerverhalten so viele Beziehungen zwischen Webseiten als möglich ableiten zu können.

Besuchsdauer von Webseiten

Die Besuchsdauer einer Webseite kann als Indikator dafür dienen, für wie interessant der Benutzer diese Seite hält. Je länger ein Benutzer eine Seite besucht hat, umso mehr interessante Inhalte wird er vermutlich dort entdeckt haben. Für jeden Logeintrag, außer für Sitzungsendeinträge, lässt sich die Besuchsdauer einer Webseite aus der Differenz der Besuchszeitpunkte des Eintrags selbst und des folgenden Eintrags berechnen.

Definition 3 Seien l_i und l_{i+1} zwei direkt aufeinander folgende Logeinträge und l_i kein Sitzungsendeintrag, dann gilt für die Besuchsdauer von l_i :

$$l_i.period = l_{i+1}.time - l_i.time$$

Inhaltseigenschaft

In [5] und [24] wird festgestellt, dass Benutzer Webseiten auf zwei verschiedene Arten besuchen. Entweder liest sich der Benutzer eine Seite wegen des Inhaltes durch oder er verwendet sie, um zu einer anderen Seite zu navigieren, die wiederum zu einer dieser beiden Arten gehört. Diese Unterscheidung kann auf Beziehungen zwischen Webseiten hindeuten, wenn zum Beispiel eine gewisse Seite häufig der Navigation zu einer anderen Seite dient.

Aus der Besuchsdauer wird erschlossen, welche Webseiten der Benutzer als inhaltlich interessant betrachtet hat und welche Seiten zu Navigationszwecken nur kurz besucht wurden. Mit Hilfe eines Grenzwerts für die Besuchsdauer wird zwischen Logeinträgen von *Inhaltsseiten* und *Navigationsseiten* unterschieden. Eine Seite, die länger als die Grenzbesuchsdauer

betrachtet wurde, gilt als Inhaltsseite, während alle anderen Seiten als Navigationsseiten gelten. Da sich die Lese- und Klickgeschwindigkeit von Benutzer zu Benutzer unterscheiden, lässt sich für den Grenzwert kein fixer Wert festlegen. Stattdessen wird der Grenzwert mit Hilfe der Bildung des Mittelwerts $m_{\mathcal{L}}$ über alle Besuchsdauern der Logdatei gebildet. Welche Art der Mittelwertbildung letztendlich zu verwenden ist, wird anschließend diskutiert.

Für die Inhaltseigenschaft wird ein weiteres Attribut $l.type$ für Logeinträge eingeführt, das die Werte *main* für Inhaltsseiten und *transit* für Navigationsseiten annehmen kann.

Definition 4 Für den Typ eines Logeintrags l gilt:

$$l.type = main \iff l.period * mainStatMultiplier > m_{\mathcal{L}}$$

Der Parameter *mainStatMultiplier* dient zur manuellen Justierung der Typunterteilung.

Arithmetisches Mittel Bei der Verwendung des arithmetischen Mittels [21] beeinflussen Einträge mit extrem hohen Besuchsdauern, die so genannten Ausreißer, den Mittelwert $m_{\mathcal{L}}$ zu stark. Dies führte bei empirischen Versuchen dazu, dass durch hohe Schwankungen im Mittelwert der *mainStatMultiplier* je nach Logdatei sehr unterschiedliche Werte annehmen musste, um eine sinnvolle Typunterteilung zu liefern.

Winsorisiertes Mittel Einen besseren Ansatz bildet das winsorisierte Mittel [11]. Hier wird zwar auch letztendlich ein arithmetischer Mittelwert gebildet, jedoch werden Ausreißer außen vor gelassen. Hierzu müssen die Besuchsdauerwerte zunächst sortiert werden. Anschließend wird an beiden Enden der sortierten Liste ein prozentualer Anteil ignoriert, um Ausreißereinflüssen vorzubeugen. Der Ansatz bringt aber zwei Probleme mit sich. Zunächst müssen die Besuchszeiten einer Logdatei sortiert werden, was zusätzlichen Aufwand bedeutet. Zweitens ist nicht universell festlegbar, wie groß die abzuschneidenden Anteile zu wählen sind.

Geometrisches Mittel Das geometrische Mittel [21] benötigt keine Sortierung und keine Bestimmung prozentualer Anteile. Durch die Multiplikation der Besuchsdauerwerte und die anschließende Wurzelbildung beim geometrischen Mittel wird erreicht, dass Ausreißer bei der Mittelbildung nicht zu stark gewichtet werden. Das geometrische Mittel löst also das Problem der Ausreißerbehandlung und wird daher in dieser Arbeit verwendet.

Zum besseren Verständnis des *mainStatMultiplier* sei kurz festgehalten, dass für den Wert 0 kein Eintrag Typ *main* besitzen wird, da das Produkt in Def. 4 immer den Wert 0 liefert und dadurch nie größer als das geometrische Mittel werden kann. Für den Fall ∞ wird hingegen jeder Eintragstyp auf *main* gesetzt. Der Wert 0,5 bedeutet, dass die Besuchszeit eines Knotens mehr als doppelt so groß sein muss wie das geometrische Mittel, damit der Eintrag als *main* angesehen wird. Unter Verwendung des geometrischen Mittels ließ sich empirisch ein guter Wert für diesen Parameter festlegen. Die besten Ergebnisse beim empirischen Vergleich mehrerer Parameterwerte mit zahlreichen Logdateien lieferte der Wert *mainStatMultiplier* = **0,75**.

In Abb. 2.2 ist ein Beispiel für eine Logdatei, die eine Sitzung enthält, dargestellt. Sieht man sich den dritten Eintrag an, so erfährt man, dass Seite *D* an den geographischen Koordinaten (5, 0; 7, 0) per Link von Seite *B* aus angefordert wurde. Neben der Größe der Seite sind auch die beiden abgeleiteten Attribute dargestellt. Mit Hilfe des Zugriffszeitpunkts auf *E* kann die Besuchsdauer von 25 Sekunden berechnet werden. Die Berechnung der Inhaltseigenschaft mit Def. 4 ergibt Typ *main*, da $25 * 0,75 > 6,82$. Im Gegensatz dazu wird Seite *E* nur 5 Sekunden besucht, was vermuten lässt und durch die Berechnung bestätigt wird, dass es sich hierbei um eine Navigationsseite handelt. Auch wird *E* zwar direkt nach *D* angefordert, jedoch nicht über einen Link. Der Benutzer kann die Adresse also eingegeben oder ein Lesezeichen ausgewählt haben.

ID	loc	time	action	size	period	type
A	(1,0;2,0)	2005-10-20 / 12:00:00	jump	5512	3	transit
B	(1,0;2,0)	2005-10-20 / 12:00:03	link	5432	2	transit
D	(5,0;7,0)	2005-10-20 / 12:00:05	link	5510	25	main
E	(1,0;2,0)	2005-10-20 / 12:00:30	jump	4524	5	transit
F	(1,0;2,0)	2005-10-20 / 12:00:35	link	7649	3	transit
T	(1,0;2,0)	2005-10-20 / 12:00:38	jump	8665	37	main
F	(1,0;2,0)	2005-10-20 / 12:01:15	revisit	7649	7	transit
J	(1,0;2,0)	2005-10-20 / 12:01:22	link	5293	8	transit
I	(1,0;2,0)	2005-10-20 / 12:01:30	jump	3200	---	

Sitzungsende

geometrisches Mittel:	6,82
mainStatMultiplrier	0,75

Abbildung 2.2: Beispiel - Logdatei

2.2.3 Semantik und Struktur des Informationsgraphen

Jede Infostation verwaltet ihren eigenen Informationsgraphen, welchen sie aus den Informationen, die in den eingehenden Logdateien enthalten sind, aufbaut. Hierzu wird jede Logdatei in die enthaltenen Sitzungen unterteilt.

Der Informationsgraph ist ein gewichteter und gerichteter Graph, der zudem knoten- und kantenbeschriftet ist. Jeder Knoten des Informationsgraphen repräsentiert dabei eine Webseite, die im Einzugsbereich der Infostation angefordert wurde. Eine Kante zwischen zwei Knoten bedeutet, dass die beiden entsprechenden Webseiten innerhalb einer Sitzung direkt aufeinander folgend angefordert wurden.

Definition 5 Ein *Informationsgraph* ist ein Tupel $IG = (V, E, f, g, B, z)$ wobei gilt:

- $IG.V$ ist die Menge an Knoten des Graphen.
- $IG.E \subseteq \{(u, v) \mid u, v \in V \wedge u \neq v\}$ ist die Menge an Kanten.
- $IG.f$ ist die zeitliche Glättungsfunktion. Sie sorgt dafür, dass die Bedeutung von Logeinträgen mit zunehmendem Alter nachlässt.
- $IG.g$ ist der Wurzelknoten des Graphen. Er kann als Einstiegspunkt in den Graphen betrachtet werden. Von ihm aus sind alle Sitzungsstartknoten direkt mit einer Kante verbunden.
- $IG.B \subseteq V$ ist die Menge an Knoten, welche Webseiten repräsentieren, die von Benutzern als Startpunkt einer Sitzung benutzt wurden. Diese Knoten werden **Sitzungsstartknoten** genannt. Für jeden Knoten $u \in V$ gilt: $(g, u) \in E \iff u \in B$.
- $IG.z$ ist der akkumulierte Knoten. Eine Kante zu z können Knoten aus zwei Gründen besitzen:
 1. Repräsentiert ein Logeintrag das Ende einer Sitzung, so wird der entsprechende Knoten im Graphen mit z verbunden.
 2. Wird von einem Logeintrag l aus der nächste Eintrag l' per "revisit" angefordert, so wird ebenfalls der Knoten, der die Webseite aus l repräsentiert, mit z verbunden. Dies wird so gehandhabt, da durch das Wiederbesuchen eines bereits bekannten Knotens eine Folge von Seitenaufrufen als unterbrochen angesehen wird.

Die Verwendung der Glättungsfunktion f kann sehr einfach mittels eines Beispiels begründet werden. Angenommen, eine Webseite, die zu einem Sportereignis Informationen bietet, wird im Vorfeld der Veranstaltung von sehr vielen Benutzern besucht. Nachdem das Ereignis vorüber ist, wird diese Seite aber nur noch sehr viel seltener aufgerufen. Würden Logeinträge ihre Bedeutung im Informationsgraphen nun immerfort bewahren, so würde diese Seite eventuell eine sehr lange Zeit noch als sehr populär eingestuft werden, obwohl sie das eigentlich längst nicht mehr ist. Ohne eine zeitliche Glättung würden es neue Webseiten sehr schwer haben, als populär eingestuft zu werden, da ältere Seiten eventuell über einen langen Zeitraum bereits Logeinträge angesammelt haben könnten.

Wie diese Glättungsfunktion genau definiert ist, ist für diese Arbeit jedoch unerheblich. Es reicht aus zu wissen, dass die Wichtigkeit von Logeinträgen bezüglich der Graphaktualisierung anhand ihrer Zugehörigkeit zu Zeiträumen berechnet wird. Ab einem gewissen Alter werden Einträge dann komplett ignoriert. Im folgenden wird diese Berücksichtigung des Alters von Einträgen *zeitliche Glättung* genannt.

Attribute von Knoten und Kanten

Definition 6 Ein Knoten des Informationsgraphen ($v \in V$) ist ein Tupel $v = (URL, n, size, mainStat)$, wobei gilt:

- $v.URL$ repräsentiert die URL der Webseite.
- $v.n$ ist die Zahl der Anfragen von Seite v , die mit Hilfe von f zeitlich geglättet wurde.
- $v.size$ steht für die Größe der Webseite in Bytes.
- $v.mainStat$ ist die Zahl der Inhaltsseiteneinstufungen, die mit Hilfe von f zeitlich geglättet wurde.

Definition 7 Eine Kante des Informationsgraphen $e = (u, v) \in E$ ist ein Tupel $e = (numUpdates, w)$, für das gilt:

- $e.numUpdates$ ist die Zahl der Aktualisierungen, die mit Hilfe von f geglättet wurde.
- $e.w$ ist das Kantengewicht und gibt die Wahrscheinlichkeit wieder, mit der ein Besucher nach u direkt den Knoten v besucht. Formal bedeutet dies:

$$e.w = \frac{e.numUpdates}{\sum_{e_i \in E \text{ die von } u \text{ ausgehen}} e_i.numUpdates}$$

Für alle Knoten außer z gilt somit: Die Summe der Gewichte aller ausgehenden Kanten, inklusive der möglichen Kante zu z , eines Knotens ergibt sich zu 1.

In Abb. 2.3 ist ein Beispiel gegeben, wie ein Informationsgraph aussehen kann. Hier wurden sowohl die Knoten mit einer symbolischen URL versehen als auch die Kantengewichte eingetragen. Die Einträge der Logdatei aus Abb. 2.2 wurden in diesem Graphen berücksichtigt. Zur Veranschaulichung der Bedeutung der Kantengewichte sei erwähnt, dass zum Beispiel das Gewicht der Kante (H, D) aussagt, dass 75% der Benutzer, die H besuchen, danach direkt Knoten D anfordern.

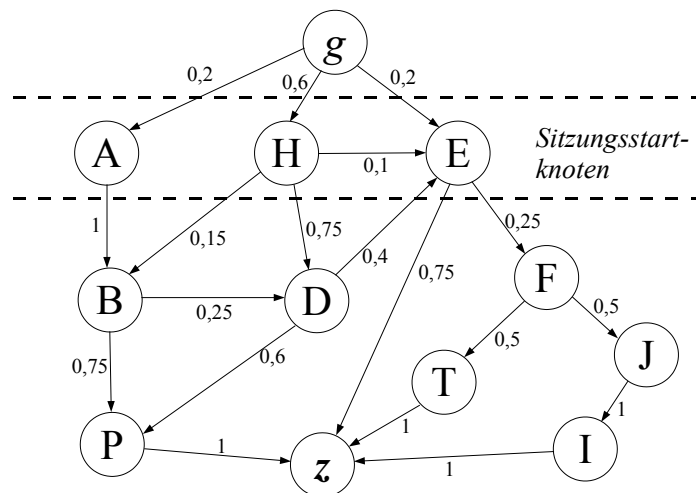


Abbildung 2.3: Beispiel - Informationsgraph

2.3 Anforderungen an Clusterverfahren

Nachdem die zu Grunde liegenden Datenstrukturen erläutert wurden, wird nun auf die Anforderungen eingegangen, die das Clusterverfahren erfüllen muss.

Aus dem vorigen Abschnitt lässt sich leicht folgern, dass der Clusteralgorithmus auf einer Graphstruktur arbeiten muss. Wie die Definition des Informationsgraphen zeigt, handelt es sich aufgrund der speziellen Knoten g und z auch um eine spezielle Art eines Graphen. Der Graph ist zusammenhängend, gerichtet und gewichtet.

Wie sich leicht nachvollziehen lässt, kann der Informationsgraph beliebig groß werden. Der Umfang hängt allein von der Menge der repräsentierten Logdateien und der damit verbundenen Webseiten ab. Es ist zu erwarten, dass bei großem Verkehrsaufkommen an Daten im Bereich einer Infostation der Graph sehr schnell aus mehreren Tausend Logdateien zusammengestellt wird.

Weiter wird die zeitliche Glättung der Graphinformationen periodisch durchgeführt. Für diese Glättung muss der komplette Graph traversiert werden, da die Eigenschaften aller Knoten und Kanten geglättet werden müssen. Aus Effizienzgründen ist es daher sinnvoll, bei dieser Traversierung die Clusterbildung vorzunehmen. Allgemein gilt für das Clusterverfahren: Je häufiger es durchgeführt wird, umso aktueller und besser wird das Ergebnis ausfallen.

Durch die nach oben unbeschränkte Größe des Graphen und die erwünschte, möglichst häufige Durchführung der Clusterbildung muss das Clusterverfahren sehr schnell sein. Das Ziel ist es, ein Verfahren zu entwickeln, das linear in der Anzahl der Knoten und Kanten des Informationsgraphen ist ($O(|V| + |E|)$).

Als nächstes muss berücksichtigt werden, dass die Zusammengehörigkeit zwischen Webseiten untersucht wird, ohne deren Inhalt zu kennen. Korrelationen zwischen Seiten werden ausschließlich über Eigenschaften des Informationsgraphen abgeleitet.

Zuletzt muss noch gefordert werden, dass das Verfahren es ermöglichen muss, dass sich Cluster gegenseitig überlappen können. Es ist zu einschränkend davon auszugehen, dass eine Webseite nur einem Cluster zugehören kann, wie Abb. 2.4 zeigt. In diesem Beispiel geht jeder Benutzer, der a oder b besucht hat, in jedem Fall zu Knoten c weiter. Diese Kanten deuten auf eine starke Zusammengehörigkeit zwischen a und c bzw. b und c hin. Da zwischen den Knoten a und b aber keine direkte Verbindung besteht, ist es sinnvoll, dass c mit seinen beiden Vorgängern jeweils separate Cluster a, c und b, c bildet.

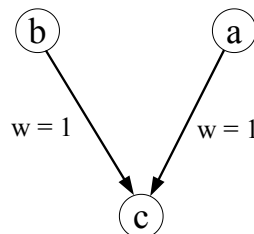


Abbildung 2.4: Überlappung von Clustern

Die Anforderungen an das Clusterverfahren lassen sich wie folgt zusammenfassen:

- Das Verfahren muss auf einen gerichteten, gewichteten Graphen anwendbar sein.
- Der Aufwand muss linear in der Anzahl der Knoten und Kanten sein ($O(|V| + |E|)$).
- Die Clusterbildung muss ohne Wissen über Seiteninhalte erfolgen.
- Es muss eine Überlappung der Cluster möglich sein.

2.4 Taxonomie existierender Clusterverfahren

Bevor konkret auf einzelne verwandte Arbeiten eingegangen wird, folgt in diesem Abschnitt ein Gesamtüberblick über verschiedene Arten von Clusterverfahren.

Durchforstet man die Literatur, so findet man unzählige verschiedene Clusteransätze zu den unterschiedlichsten Problemstellungen. Dies liegt daran, dass es kein ultimatives Vorgehen für alle Arten von Clusterproblemen gibt. Durch die Vielzahl der existierenden Clusterverfahren lassen sich diese auch nach vielen verschiedenen Kriterien klassifizieren. In [13] und [14] werden Verfahren unter anderem nach drei großen Kriterien unterschieden:

Format der Eingabedaten Die Form, in der die Eingabedaten vorliegen und auf die das Clusterverfahren aufbauen muss, ist ein wichtiger Punkt. Sind die Daten numerischer oder kategorischer Natur? Sind Daten mehrdimensional durch Vektoren beschrieben oder liegen spezielle Datenstrukturen vor, wie ein Graph im Fall dieser Arbeit?

Art des Clusteraufbaus In der Art wie Cluster aufgebaut werden, also im Algorithmus selbst, können sich Clusteransätze auf viele Arten unterscheiden. Am häufigsten findet man in der Literatur eine Aufteilung in partitionierende und hierarchische Verfahren.

Format der Ausgabe Wie bei der Eingabe auch, kann die erwünschte Form der Ausgabe sich von Fall zu Fall sehr unterscheiden. Zum Beispiel erstellen die meisten existierenden Verfahren disjunkte Cluster, wobei es auch Problemstellungen gibt, wie in dieser Diplomarbeit, die sich überlappende Cluster verlangen.

Abb. 2.5, entnommen aus [13], zeigt eine mögliche Taxonomie von Clusterverfahren. Jain et al. weisen aber sogleich auch darauf hin, dass es eine Vielzahl weiterer Unterscheidungskriterien gibt. Nachfolgend sollen einige aufgezählt werden.

- Bereits aus der Theorie ist bekannt, dass das Finden einer optimalen Clusterunterteilung abhängig von der Zielfunktion zumeist ein NP-vollständiges Problem darstellt [8]. Von daher muss sich der Entwickler eines Clusterverfahrens bereits im Vorfeld Gedanken darüber machen, wie er Cluster möglichst effizient bilden kann. Dadurch ergeben sich teils erhebliche Unterschiede im Aufwand der Verfahren.
- Clusteransätze lassen sich ebenso in agglomerative und unterteilende Verfahren aufgliedern. Während beim agglomerativen Verfahren die Cluster, von einelementigen Clustern

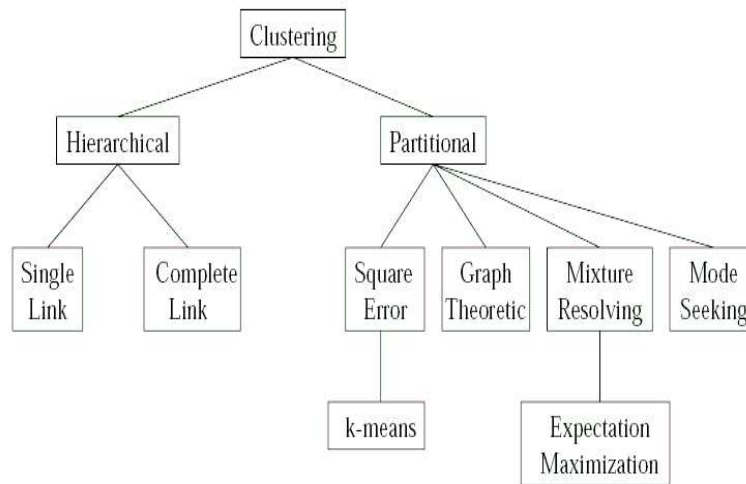


Abbildung 2.5: Taxonomie (Jain et al.)

ausgehend, zusammengefügt werden, wird beim unterteilenden Ansatz von einem großen Cluster ausgegangen, der sukzessiv in kleinere Cluster aufgeteilt wird. Die Clusterzusammenfügungen bzw. -unterteilungen werden so lange durchgeführt, bis eine Abbruchbedingung erfüllt wurde.

- Es lässt sich zwischen der Clusterbildung von relationalen Daten und der von Objektdaten unterscheiden. Objektdaten sind häufig Datensätze, die aus Eigenschaftsvektoren bestehen. Jedes Objekt besitzt einen solchen Vektor, für welchen nach einer Vorschrift ein Funktionswert berechnet wird. Anhand dieses Ergebnisses können ähnliche Elemente bestimmt und in Clustern zusammengefügt werden. Im relationalen Fall hingegen, wie er im Graphen vorherrscht, werden jeweils die paarweisen Beziehungen zwischen Elementen betrachtet (zum Beispiel durch Kanten verbundene Knoten).

2.4.1 Partitionierende und Hierarchische Verfahren

Wie bereits erwähnt, ist die Unterscheidung zwischen hierarchischen und partitionierenden Verfahren in der Literatur sehr verbreitet und beispielsweise auch in [25] enthalten. Im folgenden sollen die Vorgehensweisen dieser Verfahren anhand von Beispielen kurz erklärt werden.

Ein *partitionierender Ansatz* ist zum Beispiel das sehr bekannte *k-means* Verfahren [10]. Eine Menge an Daten wird in eine fixe Zahl k disjunkter Cluster unterteilt. Ein Element der Datenmenge ist also nach Ablauf des Algorithmus Element genau eines Clusters. Zu Beginn des grundlegenden Ablaufs von *k-means* werden k zufällige Clusterzentren gewählt. Danach

wird jedes Element dem Cluster zugeteilt, zu dessen Zentrum es am nächsten liegt. Wurde dieser Schritt durchgeführt, wird das Clusterzentrum auf Basis der nun enthaltenen Elemente für jeden Cluster neu berechnet. Anschließend wird für jedes Element überprüft, ob es nun näher zum Zentrum eines anderen Clusters liegt, als zum Zentrum des Clusters, zu dem es momentan gehört. Ist dies der Fall, so wandert das Element in den anderen Cluster, was natürlich wiederum Zentrumswerte verändert. Diese Schritte müssen so lange durchgeführt werden, bis alle Elemente im Cluster des nächstgelegenen Zentrumswerts liegen.

In Abb. 2.6, ebenfalls aus [13], ist ein Beispiel für das Ergebnis eines *hierarchischen Verfahrens* gegeben, das auf die Elemente $A \dots G$ angewendet wurde. Es kann mit Hilfe eines Baumes oder, wie im Beispiel, eines Dendrogramms visualisiert werden. Auf der obersten Ebene steht ein Cluster, der alle Elemente enthält. Die Blätter stellen die einelementigen Cluster dar. Die Ähnlichkeitsachse im Bild zeigt, dass man immer kleiner werdende Cluster erhält, wenn man engere Beziehungen zwischen den Clusterelementen fordert. Diese Hierarchien von Clustern können entweder agglomerativ oder unterteilend erstellt werden. Es lässt sich aus der Grafik bereits vermuten, dass hierarchische Verfahren nicht mit linearem Aufwand auskommen, da in jedem Hierarchieschritt der entsprechende Cluster zur Unterteilung bzw. die zu vereinigenden Cluster bestimmt werden müssen. Hierarchische Verfahren sind bekannt dafür, dass sie mindestens quadratischen Aufwand in der Anzahl der Elemente erfordern und daher sind solche Verfahren für unser Problem ungeeignet.

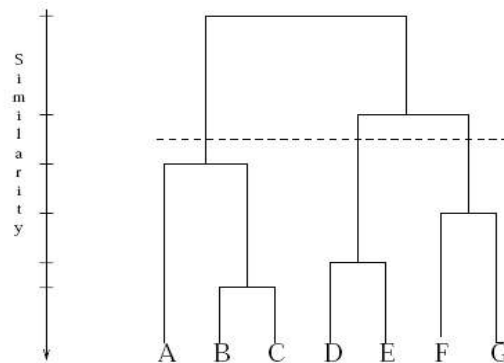


Abbildung 2.6: Hierarchisches Verfahren - Ergebnis (Jain et al.)

2.4.2 Clusterbildung in Graphen

Auch wenn die Clusterbildung in Graphen, an der im Rahmen dieser Arbeit besonderes Interesse besteht, bereits eine Spezialisierung des Clusterproblems darstellt, so gibt es in diesem Bereich immer noch sehr viele unterschiedliche Ansätze, die sich teils erheblich unterscheiden. Das kann beispielsweise an Eigenschaften des Graphen (gerichtet, gewichtet), an der Art der zu produzierenden Cluster oder am Aufwand liegen. Da für unser Problem überlappende Cluster und ein linearer Aufwand gewünscht werden, fällt bereits die große Mehrheit aller

Verfahren durch das Raster, denn die allermeisten Verfahren liefern ausschließlich disjunkte Cluster. Auch beim Aufwand liegen die meisten Verfahren mindestens im quadratischen Bereich in der Anzahl der Knoten.

Als Beispiel soll der unterteilende Algorithmus von Zahn dienen [29]. In ihm wird für einen kantenbeschrifteten Graphen zunächst der minimale Spannbaum berechnet. Danach werden sukzessive die am schwächsten bewerteten Kanten aus dem Spannbaum entfernt, so dass Cluster entstehen. Das Entfernen der Kanten wird fortgesetzt bis entweder genug Cluster entstanden sind oder keine als schwach eingestuften Kanten mehr existieren. Man sieht allein am Ergebnis, dass diese Vorgehensweise nicht den Anforderungen an das Verfahren für diese Arbeit entspricht. Es werden keine überlappenden Cluster produziert.

2.5 Verwandte Arbeiten

Nach dem Allgemeinüberblick werden in diesem Abschnitt verschiedene Arbeiten vorgestellt, deren Problemstellungen entweder mit der dieser Arbeit verwandt sind, die Lösungen ebenfalls durch Clusterbildung suchen oder beides zugleich. Die Nachforschung über diese Arbeiten erfolgte, um eventuell auf für diese Diplomarbeit wiederverwendbare Ansätze oder zumindest Teilaspekte zu stoßen. Es wird daher zu jeder Arbeit erläutert, warum der dort gewählte Ansatz nicht auf dieses Problem anwendbar ist. Die meisten dieser Veröffentlichungen stammen aus dem Bereich der Informationsgewinnung (engl. *Information retrieval*) und verwenden Clusterbildung, um Benutzern verschiedene zusammengehörige Gruppen von Dokumenten anbieten zu können.

Genauer wird letztenendes auf das SEER Projekt [16] eingegangen. Das liegt zum einen an der sehr ähnlichen Problemstellung, als auch an der Tatsache, dass einzelne Ideen und Begriffe für diese Diplomarbeit übernommen werden konnten.

Semantische Gemeinsamkeiten von Suchmaschinenanfragen auf Basis temporaler Korrelation In [4] wird ein Verfahren vorgestellt, wie semantische Ähnlichkeiten zwischen Suchmaschinenanfragen auf Basis temporaler Übereinstimmungen festgestellt werden können. Zwei Anfragen werden dann als semantisch verwandt angesehen, wenn sich die Häufigkeit ihres Auftretens über einen Zeitraum ähnlich verhält. Der Inhalt der Anfragen ist, entsprechend dem Inhalt der Webseiten im Informationsgraphen, hierbei nicht bekannt. Das Ziel dabei ist es, Benutzern, die bei Suchanfragen nicht das gewünschte Ergebnis erhalten, alternative, semantisch ähnliche Suchanfragen anzubieten, die ein Erhalten des gewünschten Resultats wahrscheinlich machen.

Aufgrund der Menge an Anfragen, die heutzutage an Suchmaschinen gestellt werden, wird hier spezielles Augenmerk auf den Aufwand gerichtet. Im Gegensatz zu den verwendeten Datenstrukturen dieser Arbeit, dient aber kein Graph als Basisstruktur. Die Suchanfragen bilden eine Punktmenge und die semantisch verwandten Anfragen werden mit einer *nearest neighbor*-Methode bestimmt, um nicht jede Anfrage mit jeder anderen vergleichen zu müssen und so den Aufwand niedrig halten zu können. Ähnliche Anfragen werden hierbei auch nicht in Clustern verpackt, sondern für jede eingehende Anfrage in Echtzeit neu bestimmt.

Clusterhierarchien bei der Informationsgewinnung aus Hypertext [6] setzt ein hierarchisches Clusterverfahren ein, um semantisch ähnliche Hypertextdokumente zusammenzufassen. Somit können bei Benutzeranfragen schneller Dokumente gesuchten Inhaltes geliefert werden. Im Vergleich zum Problem dieser Diplomarbeit, liegt der Fokus hier auf der schnelleren Bearbeitung der Informationsanfragen mit Hilfe der Cluster und nicht auf der schnellen Clustererstellung selbst. Die Cluster werden agglomerativ hierarchisch erstellt. Dies geschieht einerseits auf Basis von zu Links zwischen Dokumenten gehörenden Schlüsselworten, andererseits werden auch hierarchische Strukturen in den Dokumenten selbst berücksichtigt. Dies ermöglicht es dem Benutzer, ähnliche Dokumente entweder anhand ihrer Verlinkung zu finden oder durch den semantisch ähnlichen Inhalt einzelner Teile der Dokumente. Die Ergebnisse werden in einem Browser visualisiert, um dem Benutzer ein Durchforsten der Clusterstruktur zu ermöglichen.

Es gibt hier gleich mehrere Punkte, die den Anforderungen unseres Problems widersprechen. Der in Abschnitt 2.4.1 bereits erwähnte quadratische Aufwand für die hierarchische Erstellung der Cluster wird in Kauf genommen, da nur der Zugriff auf die Inhalte effizient vollzogen werden soll. Ebenso werden die Dokumente auf Basis ihres Inhaltes und der Art ihrer Verlinkung zusammengefasst (engl. *content-aware*), während im Informationsgraphen nichts über den Inhalt der Webseiten bekannt ist.

Zusammenfassung mehrerer Dokumente: Verbesserung der Clusterbildung bei interaktiver Informationsgewinnung In [19] wurde ein Ansatz entwickelt, der dem Benutzer die Rückgabewerte von Suchanfragen in übersichtlicher Form präsentiert. Hierbei werden inhaltlich ähnliche Dokumente nicht nur in Cluster zusammengefasst, sondern es werden zusätzlich die Gemeinsamkeiten und Unterschiede der im Cluster enthaltenen Dokumente beschrieben. Die Clusterbildung, auf der in dieser Arbeit nicht das Hauptaugenmerk liegt, erfolgt auf Basis der Algorithmen *k-means* und *bisecting k-means* [27].

Auch für diese Arbeit ist die inhaltliche Analyse der Dokumente essentiell. Hinzu kommt die Verwendung von partitionierenden Clusteralgorithmen, die keine überlappenden Cluster liefern. Somit lässt sich aus dieser Arbeit keinerlei Nutzen für das Clusterproblem des Informationsgraphen ziehen.

Clusterbasierte Informationsgewinnung unter Verwendung von Sprachmodellen In [18] wird eine Menge von Dokumenten zunächst basierend auf deren inhaltlichen Sprachkonstrukten einem Clusterverfahren unterzogen. Benutzer auf der Suche nach einem bestimmten Inhalt können entsprechende Dokumente mit Hilfe einer Anfrage auffinden, die aus einer Folge von Worten besteht. Bei der Verarbeitung solch einer Anfrage werden die einzelnen Worte herausgegriffen und für jeden Cluster eine Wahrscheinlichkeit errechnet, mit der dieses Wort zu diesem Cluster passt.

Wie bei vielen anderen Arbeiten auch, spielt hier die inhaltliche Untersuchung von Dokumenten eine zentrale Rolle. Es lässt sich nur wiederholen, dass im Informationsgraphen keine Informationen über Webseiteninhalte bekannt sind und solche Unterscheidungskriterien hier nicht verwendet werden können. Weiter kommt hinzu, dass neben dem bekannten *k-means*

Algorithmus auf hierarchisch agglomerative Verfahren zurückgegriffen wird und somit auch die in dieser Diplomarbeit geforderte Aufwandsgrenze nicht eingehalten wird.

Auswertung von Dokumentclustermethoden zur interaktiven Informationsgewinnung Leuski bespricht in [17] verschiedene Clusterverfahren, die Dokumente gruppieren, die auf eine Suchanfrage hin als Resultat zurückgeliefert werden. Damit wird dem Benutzer eine übersichtlichere Darstellung des Ergebnisses geliefert. Dies stellt eine Verbesserung zur klassischen Ranglistenanzeige der Treffer dar, die in vielen Suchmaschinen verwendet wird. Die Ähnlichkeit der Dokumente wird anhand eines Vektorraummodells gemessen. Die Dimension des Vektorraums ist gleich der Anzahl der Worte im verwendeten Vokabular. Für jedes Dokument wird das Gewicht jedes möglichen Ausdrucks im Vektor repräsentiert. Um zwei Dokumente miteinander auf Ähnlichkeit hin vergleichen zu können, wird der zwischen den entsprechenden Vektoren aufgespannte Winkel betrachtet. Auf Basis dieses Modells werden dann die Cluster durch sechs verschiedene Clusteralgorithmen aufgebaut und die Resultate miteinander verglichen.

Neben der Clusterentscheidung auf Basis des Inhaltes der Dokumente werden aufgrund der Annahme, dass hinreichend wenige Dokumente miteinander verglichen werden müssen, Clusterverfahren mit quadratischem Aufwand eingesetzt. Dies führt dazu, dass keines der in dieser Arbeit vorgestellten Verfahren für diese Diplomarbeit in Betracht kommt.

SimFusion Die Verwendung einer *Unified Relationship Matrix* zur Feststellung von Beziehungen zwischen heterogenen Datenobjekten ist Inhalt von [28]. Es wird definiert, wie Objekte verschiedener so genannter Datenräume auf Korrelationen hin untersucht werden können. Damit ist zum Beispiel gemeint, dass man vermeiden will, dass die Beziehung zwischen Dokumenten lediglich durch deren Verlinkung bestimmt wird. Neben den Dokumenten bilden auch Worte einen Datenraum, womit Beziehungen zwischen Dokumenten auch mittels Worthäufigkeiten erkannt werden können. Weitere berücksichtigte Datenräume sind Suchanfragen und Menschen. Jedes einzelne Datenobjekt jedes Datenraums wird dann durch einen Eintrag in der Matrix repräsentiert.

Wie die Verwendung einer Matrix jedoch bereits vermuten lässt, besitzt dieses Verfahren quadratische Komplexität. Im schlechtesten Fall ist der Aufwand sogar kubisch in der Anzahl der gesamten Datenobjekte.

Agglomerative Clusterbildung eines Logs von Suchmaschinenanfragen In [1] wird ähnlich wie bei den Infostations ein Graph aus Logdateien aufgebaut. Bei den Einträgen der Logs handelt es sich jedoch um Paare von Suchanfragen und der dazu vom Benutzer ausgewählten Webseite. Daraus wird ein bipartiter Graph aufgebaut. Auf einer Seite befinden sich sämtliche Knoten für Suchanfragen, während auf der anderen Seite die Knoten liegen, welche die Webseiten repräsentieren. Ziel ist es, sowohl Cluster ähnlicher Suchanfragen als auch Cluster von Webseiten auf Basis dieses Graphen zu erstellen. Dies geschieht folgendermaßen:

Besuchen Benutzer nach dem Absetzen einer identischen Suchanfrage zweierlei Seiten, so

wird zwischen diesen eine Ähnlichkeit angenommen. Analog wird die Ähnlichkeit von Suchanfragen festgestellt. Das Maß der Ähnlichkeit zwischen Webseiten- bzw. Anfragenknoten wird dann mit Hilfe des folgenden Bruchs ermittelt:

$$\frac{\text{Größe der Schnittmenge der unmittelbaren Knotennachbarn}}{\text{Größe der Vereinigungsmenge der unmittelbaren Knotennachbarn}}$$

Dieser Ansatz arbeitet, entsprechend den Anforderungen dieser Diplomarbeit, ohne Wissen über den Inhalt von Seiten und Anfragen. Zusammenhänge werden lediglich durch Kanten erkannt. Da der Graph allerdings bipartit ist, handelt es sich beim verwendeten Clusterverfahren um ein sehr spezielles, das allerdings trotz seiner agglomerativen Eigenschaft lediglich linearen Aufwand in der Zahl der Knoten benötigt. Weiter kommt hinzu, dass keine überlappenden Cluster entstehen können. So lassen sich aus diesem sowohl einfachen als auch interessanten Verfahren keine Schlussfolgerungen für das Problem dieser Diplomarbeit ziehen.

Datenpartitionierung für Client-Server-Datenbankbetrieb ohne Netzanbindung

Phatak und Badrinath benutzen in [23] ein ähnliches Systemmodell wie diese Diplomarbeit, das ebenfalls auf die Verwendung von Infostations aufbaut. Benutzer können benötigte Daten von einem zentralen Datenbankserver vorab übertragen lassen, um in Zeiträumen ohne Netzanbindung auf diese Daten zugreifen und sie gegebenenfalls auch ändern zu können. Beim Betreten des Einzugsbereichs der Infostation werden dem Benutzer sowohl die benötigten Daten vorab übertragen als auch die vorgenommenen Änderungen dem Server mitgeteilt. Weiter wird in [23], wie auch hier, von einem Lokationsbezug der benötigten Daten ausgegangen.

Im Gegensatz zur Vorabübertragung in dieser Diplomarbeit muss der Benutzer, auch wenn die Auswahl der Daten bezogen auf den Ort automatisch erfolgen kann, manuell durch Eingabe so genannter *qualifiers* bestimmen, welche Art der Daten er im entkoppelten Betrieb brauchen wird. Die Auswahl der Daten erfolgt also nicht ohne Zutun des Benutzers. Auch werden keine dauerhaften Cluster erstellt, sondern es wird die benötigte Datenpartition für jedes Endgerät neu ermittelt.

2.5.1 SEER

Kuenning und Popek nehmen sich in [16] einem Problem an, das der Aufgabenstellung dieser Diplomarbeit sehr ähnlich ist. Es wird ein Verfahren gesucht, das es Benutzern mobiler Endgeräte ermöglicht, im lokalen Betrieb mit entfernt gespeicherten Dateien, die sonst nur über eine Netzanbindung erreichbar sind, arbeiten zu können. Dazu werden diese Dateien auf dem Endgerät gespeichert. Die Auswahl der vorab zu übertragenden Dateien wird wie hier aus der Analyse des Nutzerverhaltens in der Vergangenheit getroffen und erfolgt völlig automatisch ohne jegliche Interaktion mit dem Benutzer. Hierzu wird ein neues Maß eingeführt, die so genannte *semantische Distanz* zwischen einzelnen Dateien. Dieser Begriff wird für das Maß an Korrelation zwischen Webseiten für diese Diplomarbeit übernommen. Kuenning und Popek gehen davon aus, dass sich das zukünftige Nutzerverhalten durch Zugriffe auf Projekte bestimmen lässt, statt durch Zugriffe auf einzelne Dateien. Die Dateien, die zusammen

ein Projekt bilden, können durch ihre semantische Distanz bestimmt, in einem Cluster zusammengefasst und gemeinsam übertragen werden. Wie auch in dieser Arbeit werden Cluster dann entweder komplett oder gar nicht übertragen.

Ein Beobachterprozess protokolliert sämtliche Art von Zugriffen auf Dateien und übergibt diese Ergebnisse, die den Logdateien des Clusterproblems des Informationsgraphen entsprechen, einem Korrelator. Dieser berechnet die semantische Distanz zwischen Dateien, auf deren Basis ein Clusteralgorithmus dann jede Datei einem oder mehreren Projekten hinzufügt. Dadurch erfüllt dieses Verfahren auch die in dieser Diplomarbeit geforderte mögliche Überlappung von Clustern.

Die semantische Distanz wird ohne Wissen über Dateiinhalte durch temporale Übereinstimmungen der Dateizugriffe berechnet. Vereinfacht gesagt bedeutet dies, dass die semantische Distanz zweier Dateien umso geringer ist, je kürzer die Abstände zwischen den Zugriffen auf diese Dateien sind. Hinzu kommen noch weitere Faktoren, wie beispielsweise die Verzeichniszugehörigkeit, auf die aber nicht näher eingegangen werden soll.

Um nun nicht die Distanzen zwischen allen möglichen Paaren von Dateien berechnen zu müssen und so quadratischen Aufwand zu vermeiden, wird eine Heuristik eingesetzt, die jede Datei nur mit ihren n nächsten Nachbarn vergleicht.

Der Clusteralgorithmus geht agglomerativ vor, beginnt also mit jeder Datei als einelementigem Cluster. Die Entscheidung, wie mit Paaren von Dateien verfahren werden soll, hängt davon ab, wie viele nächste Nachbarn sie gemeinsam haben. Auch hier wird durch eine Heuristik vermieden, dass jede Datei mit jeder anderen verglichen werden muss. Der Algorithmus teilt sich nun in zwei Phasen auf:

1. Für alle untersuchten Paare von Dateien (a, b) , die mindestens k_n nächste Nachbarn gemeinsam haben, gilt: Die Cluster, zu denen a oder b gehören, werden zu einem verschmolzen.
2. Für alle untersuchten Paare von Dateien (a, b) , die weniger als k_n aber mindestens k_f nächste Nachbarn gemein haben, gilt: a wird zu den Clustern, die b enthalten, zugefügt. Analog wird b all denen Clustern hinzugefügt, die a enthalten. So entsteht die Überlappung.

Die Auswahl der zu übertragenden Cluster erfolgt nach einem LRU-Verfahren (engl. *last recently used*). Das bedeutet, dass der Cluster zuerst übertragen wird, der die Datei enthält, auf die zuletzt zugegriffen wurde. Entsprechend wird mit der zweitletzten Datei umgegangen und so weiter. In der Praxis wird das LRU-Prinzip häufig beim *Caching* angewendet.

Dieses Verfahren liefert hervorragende Ergebnisse. Nicht nur werden fast ausschließlich solche Dateien ausgewählt, die der Benutzer auch tatsächlich benutzt. Die Anzahl der Fehlermeldungen, die der Nutzer wegen fehlender Dateien erhält, ist sehr gering. Auch fehlte im Test bei keinem Benutzer eine Datei, ohne die das Gerät für ihn unbenutzbar werden würde. Der Zeitraum bis zur ersten Referenzierung einer nicht übertragenen Datei bewegt sich im Schnitt im Bereich mehrerer Stunden. Wobei festgehalten werden muss, dass häufig entweder kurz nach dem Arbeitsbeginn bereits eine nicht vorhandene Datei bemängelt werden muss oder aber während der kompletten Arbeitsphase alle benötigten Dateien vorhanden sind.

In [15] machen Kuenning et al. bei der Analyse der exzellenten SEER-Ergebnisse eine überraschende Entdeckung. Um die Verbesserung durch das Hinzufügen des Clusterverfahrens besser einschätzen zu können, wird das in [16] zum Vergleich herangezogene LRU-Verfahren ohne Clusterbildung mit der gleichen Heuristik verbunden, die auch im Clusterverfahren zum Einsatz kommt. Das Ergebnis ist, dass fünf von sechs Benutzern mit diesem neuen Vergleichsverfahren die besten Ergebnisse erzielen. Kuenning et al. kommen also zu dem Schluss, dass die Clusterbildung unnötig ist und das Verfahren allein durch die Modifikation des LRU-Verfahrens deutlich verbessert werden kann.

Trotz der sehr ähnlichen Problemstellung lassen sich keine konkreten Vorgehensweisen in diese Diplomarbeit übernehmen. Das liegt vor allem daran, dass die Berechnung der semantischen Distanz zwischen Dateien, wie sie in SEER vorgenommen wird, so nicht auf Webseiten übertragbar ist. Auch arbeitet das Clusterverfahren, das ja zudem laut [15] keine Verbesserung des Ergebnisses bewirkt, nicht auf einer Graphstruktur, sondern auf einer Menge von Dateien mit Hilfe einer Art von Adjazenzlisten für die nächsten Nachbarn.

Kapitel 3

Untersuchte Clusteransätze

Nachdem festgestellt wurde, warum existierende Verfahren nicht auf das Problem dieser Arbeit anwendbar sind, wird in diesem Kapitel auf die Entwicklung eigener Clusteransätze eingegangen. Es wird erläutert, welche Ansätze untersucht werden und warum sie schlussendlich nicht verwendet werden. Das in Kapitel 4 erläuterte, letztendlich angewendete Verfahren entsteht aus diesen Vorarbeiten und durch Erkenntnisse, welche mit Hilfe der folgenden Ansätze gewonnen werden.

3.1 Allgemeines zur Vorgehensweise

Bei der Erarbeitung der folgenden Ansätze wird mit einem intuitiven Verfahren begonnen, welches sich leicht von den bekannten Suchverfahren für Graphen ableiten lässt. Bei der Entwicklung des intuitiven Ansatzes wird, wie auch bei allen Weiterentwicklungen, auf bestimmte Kriterien geachtet, die für ein gutes Clusterverfahren grundlegend sind. Neben dem Aufwand gilt es, die Semantik der Cluster für einen jeden Ansatz festzulegen. Außerdem muss eine Bewertung der Cluster ermöglicht werden, um sie nach deren Erstellung vergleichen zu können. Es sollen am Ende schließlich nur die Cluster in der Vorabübertragungsliste enthalten sein, die dem Benutzer den größten Nutzen bieten. Bei der Erarbeitung eines Clusteransatzes ist also auf die folgenden drei Punkte zu achten:

Aufwand Wie bereits in Abschnitt 2.3 begründet wurde, ist es wichtig, dass der Algorithmus sehr effizient arbeitet. Da der Informationsgraph beliebig groß werden kann und die Cluster in regelmäßigen Zeitabständen neu erstellt werden müssen, ist ein zu hoher Aufwand nicht vertretbar. Das Ziel liegt darin, linear in der Anzahl der Knoten und Kanten des Graphen zu bleiben ($O(|V| + |E|)$).

Semantik Es muss eine passende Clustersemantik für das Problem dieser Arbeit gefunden werden, die aussagt was der Cluster darstellt und warum ausgerechnet die enthaltenen Knoten einen Cluster bilden sollen.

Bewertbarkeit Die Cluster müssen bewertbar sein, um sie vergleichen zu können. Welche Kriterien hierzu herangezogen werden können, wird im folgenden Abschnitt vorgestellt.

3.1.1 Kriterien zur Clusterbewertung

Zur Bewertung der Cluster können, unabhängig vom verfolgten Clusteransatz, zwei Eigenschaften herangezogen werden. Neben der Clustergröße eignet sich auch die Wahrscheinlichkeit des Clusters als ein Kriterium zur Bewertung. Die Wahrscheinlichkeit des Clusters, im folgenden *inneres Gewicht* genannt, kann je nach Semantik verschieden definiert werden.

Inneres Gewicht Wie eben erwähnt, lässt sich die Bestimmung des inneren Gewichts eines Clusters nicht pauschal festlegen. Es können jedoch ansatzunabhängige Aussagen darüber getroffen werden, welche Eigenschaften des Informationsgraphen hierzu verwendet werden können.

Eine dieser Eigenschaften ist das Kantengewicht. Es sagt aus, mit welcher Wahrscheinlichkeit ein Besucher, der sich an dem Knoten befindet, von welchem die Kante ausgeht, als nächstes den Endknoten dieser Kante besucht. Über die Gewichtung der Kanten lässt sich also feststellen, welche Knoten des Graphen häufiger und welche seltener besucht werden. Bei der Bewertung eines Clusters müssen die Kantengewichte der im Cluster enthaltenen Kanten berücksichtigt werden. Je größer diese Gewichte sind, desto wahrscheinlicher bewegt sich der Benutzer innerhalb des Clusters. Da der Informationsgraph mit g einen festen Startknoten besitzt, sind aber auch die Gewichte der Kanten wichtig, über welche der Cluster im Graph überhaupt erreichbar wird. Ob und wie noch weitere Kantengewichte berücksichtigt werden müssen, hängt dann vom jeweiligen konkreten Verfahren und dessen Semantik ab.

Auch Knoteneigenschaften können bei der Berechnung des inneren Gewichts eine Rolle spielen. Im Informationsgraph besitzen die Knoten jedoch keine Eigenschaft, die pauschal Berücksichtigung im inneren Gewicht verlangen könnte. Auch hier gilt, dass eine Einbeziehung in die Berechnung, abhängig von der Semantik des konkreten Verfahrens, Sinn machen kann.

Clustergröße Das zweite Kriterium, das bei der Clusterbewertung mit einbezogen werden muss, ist die Größe des Clusters. Da auf dem Endgerät des Benutzers nur eine bestimmte Menge an Speicher zur Verfügung steht, ist es von Vorteil, wenn Cluster möglichst klein sind. Dies liegt in der Annahme begründet, dass mehrere kleine Cluster ein breiteres, besser ausgewähltes Spektrum an Webseiten des Graphen abdecken als ein großer Cluster, der auf einen gewissen Teil des Graphen beschränkt ist. Jeder Cluster muss also die Summe seiner Webseitengrößen wissen.

3.2 Intuitive Ansätze

Wie bereits in Abschnitt 2.3 angesprochen, besitzt der zu Grunde liegende Graph einige Eigenschaften, die spezielle Anforderungen an das Clusterverfahren stellen.

Der gerichtete Graph ist nicht notwendigerweise azyklisch. Zudem besitzt er einen fixen Startknoten g , der für das Clusterverfahren als Einstiegspunkt dienen wird. Außerdem kann der Graph, je nach der Menge an Benutzerdaten, auf der er aufbaut, beliebig groß werden. Dies erfordert, dass das zu verwendende Verfahren ausreichend schnell ist. Nicht zuletzt soll

es möglich sein, dass sich die erstellten Cluster überlappen können.

Als Basisverfahren für das Erstellen der Cluster kommen existierende Verfahren zur Traversierung von Graphen in Betracht. Die verbreiteten Verfahren zur Tiefen- bzw. Breitensuche [26] in Graphen sind nicht nur sehr schnell und einfach. Sie ermöglichen es auch, jeden vom Einstiegsknoten des Suchalgorithmus aus erreichbaren Knoten im Graphen zu besuchen. Als Einstiegspunkt für diese Verfahren bietet sich der Startknoten g an, weil von ihm aus alle Knoten des Graphen erreichbar sind. Da diese Suchverfahren linearen Aufwand in der Anzahl an Knoten und Kanten besitzen ($O(|V| + |E|)$), scheinen sie als Basis des Clusterverfahrens zunächst sehr gut geeignet.

3.2.1 Clustersemantik

Bevor ein Algorithmus aufgestellt werden kann, muss zunächst festgelegt werden, auf welcher semantischen Grundlage die Cluster aufgebaut werden sollen. Da im folgenden Ansatz das Hauptaugenmerk dem Aufwand gilt, wird eine sehr einfache Clustersemantik benutzt. Die Clusterbildung baut ausschließlich auf die Betrachtung von Knotenpaaren auf, die durch eine Kante direkt verbunden sind. Es wird davon ausgegangen, dass eine semantische Distanz zwischen solchen Knotenpaaren definiert wurde. Wie diese genau aussieht, ist hier nicht von Belang. Ist nun die semantische Distanz eines Knotenpaares, das direkt durch eine Kante miteinander verbunden ist, kleiner einem Grenzwert s_{th} , so soll die verbindende Kante als *semantisch stark* bezeichnet werden, ansonsten als *semantisch schwach*. Nach Ablauf des Algorithmus soll für alle semantisch starken Kanten gelten: Der Knoten, an dem die Kante eingeht, ist in all den Clustern enthalten, zu denen der Knoten gehört, von welchem die Kante ausgeht. In Abb. 3.1 ist links ein Teilgraph gegeben, der diese Semantik erfüllt. Sowohl b als auch c sind in dem Cluster enthalten, zu dem auch a gehört. Wie zu sehen ist, gehört b noch zu einem anderen Cluster, was erlaubt ist. Im rechten Teil ist ein Beispiel gegeben, in dem diese Semantik verletzt ist. Hier gehört c nicht zu den Clustern, in denen a liegt, obwohl die beiden Knoten durch eine semantisch starke Kante verbunden sind.

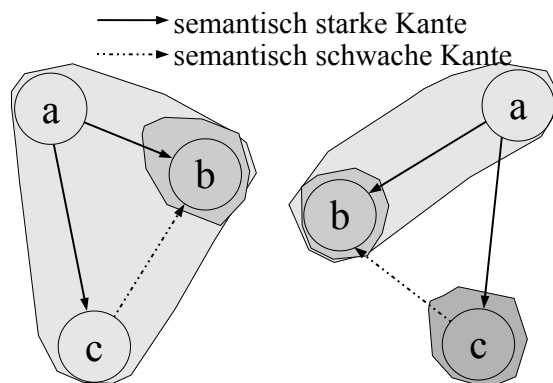


Abbildung 3.1: Clustersemantik - Beispiel

Formal niedergeschrieben bedeutet dies:

Definition 8 Sei V die Menge der Knoten und E die Menge der Kanten des Informationsgraphen. Weiter sei M_C die Menge aller erstellten Cluster nach Ablauf des Algorithmus. Ferner sei $s(x, y)$ die semantische Distanz zwischen zwei Knoten x, y mit $(x, y) \in E$ und s_{th} ein Grenzwert für die semantische Distanz. Dann soll nach Ablauf des Algorithmus gelten:

$$\forall x, y : x, y \in V \wedge (x, y) \in E \text{ gilt: } s(x, y) < s_{th} \implies (\forall C \in M_C : x \in C \implies y \in C)$$

Diese Semantik unterstellt eine Zusammengehörigkeit zwischen einer Webseite x und allen Nachfolgerwebseiten, zu deren Knoten eine semantisch starke Kante von x aus existiert. Da alle diese Nachfolger wegen x in mindestens einem Cluster gemeinsam enthalten sind, wird gleichzeitig auch eine Zusammengehörigkeit zwischen diesen Nachfolgerseiten angenommen. Mit dieser Semantik wird nun ein erster einfacher Algorithmus aufgestellt.

3.2.2 Tiefenansatz

Auf Basis der Tiefensuche werden nun die Cluster mit Hilfe des rekursiven Abstiegs erstellt. In Algorithmus 1 ist die genaue Vorgehensweise in Pseudocode aufgezeigt:

- Beginne beim Startknoten g und gehe alle Kanten rekursiv nach dem Tiefensuchprinzip ab. Für jeden Knoten x steht $N(x)$ für die Menge an Nachfolgeknoten von x .
- Ist die semantische Distanz zweier benachbarter Knoten geringer einem Grenzwert, dann gehören die beiden zusammen in einen Cluster.

Wie das Ergebnis dieses Ansatzes aussehen kann, ist in Abb. 3.2 beispielhaft dargestellt. Als semantisch starke Kanten werden diejenigen Kanten bezeichnet, deren semantische Distanz unter einem Grenzwert s_{th} liegen. Alle anderen Kanten gelten als semantisch schwach. Weiter werden die Nachfolger eines Knotens in der festen, absteigenden Reihenfolge der zugehörigen Kantengewichte besucht. In Abb. 3.2 wird davon ausgegangen, dass für alle Knoten die Gewichte der ausgehenden Kanten von links nach rechts abnehmen. Es wird also immer zuerst der Nachfolger besucht, der am weitesten links eingezeichnet ist. Eine andere Besuchsreihenfolge kann ein anderes Ergebnis liefern.

Zu beachten ist, dass kein Cluster existiert, aus dem eine semantisch starke Kante herausführt, denn dies würde eine Verletzung der Clustersemantik darstellen. An Knoten f lässt sich gut die Bedingung aus Def. 8 kontrollieren. Er gehört zu all den Clustern, zu welchen auch seine direkten Vorgänger zählen, mit denen er per semantisch starker Kante verbunden ist.

Aufwand: Im Vergleich zur regulären Tiefensuche, bei der jeder besuchte Knoten markiert wird und somit sichergestellt ist, dass er nicht mehrmals besucht wird, können hier Knoten sehr viel häufiger besucht werden. Eine Markierung ist nicht möglich, da hier nicht nur nach einem Knoten gesucht wird. Knoten können bei mehreren Besuchen immer wieder neuen Clustern hinzugefügt werden.

Algorithmus 1 Einfacher Tiefenansatz

```

for all  $n \in N(g)$  do //schaue jeden Sitzungsstartknoten an
  if  $\forall C \in M_C : n \notin C$  then //n in keinem Cluster
    erstelleCluster( $n$ ); //Erstelle neuen Cluster mit  $n$  als Element
    clusterNachfolger( $n$ );
  end if
end for

```

```

function clusterNachfolger (Knoten  $x$ ):

```

```

for all  $n \in N(x)$  do
  if  $\forall C \in M_C : x \in C \implies n \in C$  then //n bereits in allen Clustern, die x enthalten
    überspringeSchleifendurchlauf;
  else
    if  $s(x, n) < s_{th}$  then
      fügeZuClusternHinzu( $n, x$ ); //füge  $n$  allen Clustern hinzu, die  $x$  enthalten
      clusterNachfolger( $n$ );
    else if  $s(x, n) \geq s_{th} \wedge \forall C \in M_C : n \notin C$  then //n in keinem Cluster
      erstelleCluster( $n$ );
      clusterNachfolger( $n$ );
    else //Distanz zu groß und  $n$  bereits in einem Cluster
      überspringeSchleifendurchlauf;
    end if
  end if
end for

```

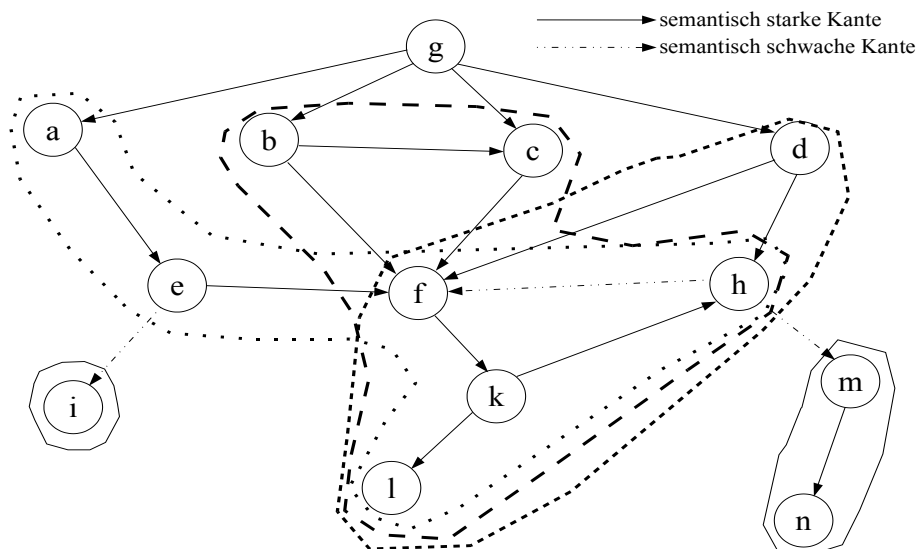


Abbildung 3.2: einfacher Tiefenansatz - Beispiel

Die rekursive Funktion *clusterNachfolger* wird genau dann aufgerufen, wenn ein neuer Cluster erstellt wird oder ein Knoten einem Cluster hinzugefügt wird. Es wird ein Szenario für den schlechtesten Fall (engl. *worst case*) erstellt, das den Aufwand abhängig der Anzahl der Knoten $|V|$ darstellt.

Der schlechteste Fall lässt sich wie in Abb. 3.3 gezeigt konstruieren, wenn man davon ausgeht, dass die Nachfolger von g in aufsteigender Reihenfolge ihrer Nummern besucht werden. Alle eingezeichneten Kanten sind semantisch stark. Hier erstellt jeder Knoten einen eigenen, neuen Cluster, dem dann alle Knoten links von ihm hinzugefügt werden. Dadurch ergibt sich für die Menge an Aufrufen der rekursiven Funktion A_{cN} :

$$A_{cN} = \sum_{i=1}^{|V|-1} i = \frac{(|V|-1)|V|}{2} \text{ Aufrufe}$$

Es ergibt sich also ein unerwünschter, quadratischer Aufwand in der Anzahl der Knoten im schlechtesten Fall ($O(|V|^2)$).

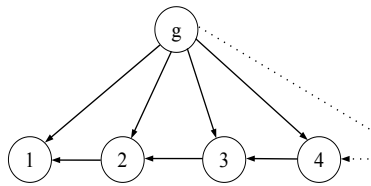


Abbildung 3.3: schlechtester Fall - Tiefenansatz

3.2.3 Aufwandsverminderung

Um den quadratischen Aufwand umgehen zu können, wird der Algorithmus so erweitert, dass wie bei der Tiefensuche jeder Knoten und jede Kante höchstens einmal besucht wird.

Damit dies möglich werden kann, muss an jedem Knoten bekannt sein, welche Knoten von ihm aus erreichbar sind und im selben Cluster wie der betrachtete Knoten liegen. Wird der Knoten also zum ersten Mal besucht, wird der Algorithmus, wie in Alg. 1 beschrieben, weiter den Graphen hinabsteigen und den oder die Cluster weiter aufbauen. Beim Zurückgehen aus der Rekursion gibt jedoch jeder Knoten an seinen Vorgänger, von dem er aufgerufen wurde, die Liste der zum Cluster gehörenden Knoten zurück und merkt sich diese. Wird der Knoten nun weitere Male besucht, kann jedesmal einfach diese Knotenliste zurückgegeben werden. Der Aufrufer weiß dann, welche Knoten zu seinem Cluster gehören und man erspart sich weitere rekursive Abstiege. Die Verwendung einer Knotenliste pro Knoten erhöht zwar den Speicheraufwand, jedoch wird die Anzahl der rekursiven Aufrufe linear in der Anzahl der Knoten. Im Beispiel in Abb. 3.4 sind die Knotenlisten eingetragen. Der Knoten h baut sich bei seinem ersten Aufruf durch k eine Knotenliste auf. Der Knoten t ist nicht in dieser Liste

enthalten, da dieser nicht in den Cluster zusammen mit h gehört. Wird h durch einen Aufruf von p aus erneut besucht, so wird diesmal einfach die Knotenliste zurückgegeben und p weiß, dass die Knoten h, u, w in den Cluster gehören.

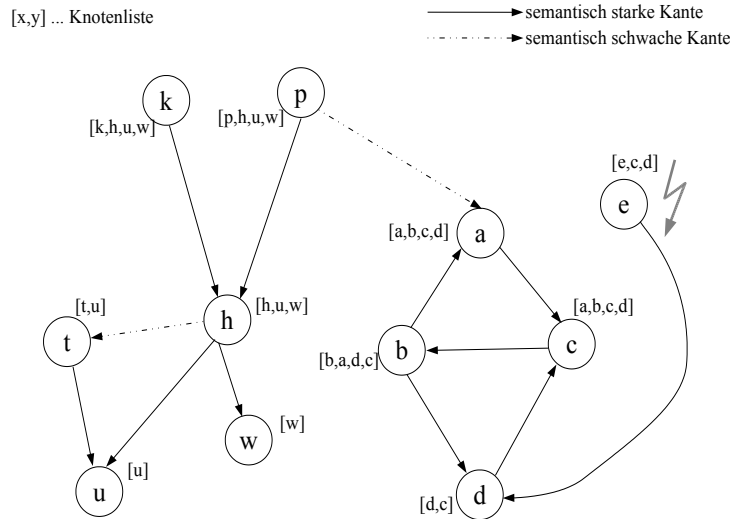


Abbildung 3.4: Verwendung einer Knotenliste

Das Zyklenproblem Bisher wurde außer Acht gelassen, dass der Graph auch Zyklen enthalten kann, was zu Problemen mit der Verwendung der Knotenliste führt. Zwar wurde der Aufwand reduziert, jedoch können Zyklen im Graph inkonsistente Knotenlisten verursachen. In Abb. 3.4 ist dieses Problem anhand von Knoten d veranschaulicht. Durch den Zyklus (b, c, d) , der ausschließlich aus semantisch starken Kanten besteht, müssten sämtliche Knoten des Zyklus identische Knotenlisten besitzen. Hierzu ist es erforderlich, eine Zyklererkennung durchzuführen. Dies könnte zum Beispiel so geschehen, dass sämtliche Knoten, die sich gerade in einem *clusterNachfolger*-Aufruf befinden, eine Markierung erhalten. So könnte d bei seinem Aufruf durch b erkennen, dass bei c ein Zyklus geschlossen wird. Zusätzlich zur Knotenliste müsste d dann eine Zyklusliste zurückgeben, die alle Knoten enthält, die bislang zum erkannten Zyklus gehören (c, d) . Beim Zurückgehen aus der Rekursion erkennt dann b , weil er eine Zyklusliste erhält, dass er Teil eines Zyklus ist und gibt die erweiterte Zyklusliste (b, c, d) zurück. Daraufhin erhält Knoten c diese Liste und erkennt, dass er selbst enthalten ist. Folglich wurde ein Zyklus erkannt und über die Knotenlisten sämtlicher Zykelselemente wird eine gemeinsame Vereinigungsliste für alle Elemente gebildet.

Diese Zyklererkennung an einem Knoten darf aber erst vollzogen werden, nachdem alle rekursiven Aufrufe seiner Nachfolger abgearbeitet sind. Werden an einem Knoten mehrere Zyklen erkannt, so müssen die Knotenlisten aller Elemente aller Zyklen vereinigt werden.

Es zeigt sich, dass das Zyklenproblem die Handhabung der Knotenlisten deutlich erschwert. Vom einfachen Tiefenansatz ausgehend, der ohne die Knotenlisten einen zu hohen

Aufwand besitzt, zeigt sich hier, dass der Aufwand nicht ohne eine erhebliche Verkomplizierung des Algorithmus verringert werden kann.

3.2.4 Clusterbewertung

Nachdem die Semantik sowie die Erstellung der Cluster beschrieben wurde, wird nun untersucht, wie die gebildeten Cluster bewertet werden können. Hierzu werden die in Abschnitt 3.1.1 vorgestellten Kriterien verwendet. Während sich die Größe des Clusters sehr einfach bestimmen lässt, liegt das Problem in der Bestimmung des inneren Gewichts.

Es wurde bereits erwähnt, dass zur Bestimmung des inneren Gewichts unter anderem die Gewichte der Kanten, die im Cluster enthalten sind, miteinander verrechnet werden müssen. Dies stellt sich bei der möglichen Form der hier erstellten Cluster als schwierig heraus. In Abb. 3.5 sind zwei Cluster dargestellt. Die Struktur des linken Clusters erlaubt eine Berechnung beispielsweise nach dem folgenden Schema: $0,3(0,35 + 0,3 + 0,25) + 0,2 = 0,47$. Dieser Wert sagt aus, dass der Benutzer von a aus in 47% der Fälle innerhalb des Clusters zu einem der Endknoten c, d oder e geht. Endknoten bedeutet hier, dass von diesen Knoten aus keine Kante zu einem anderen Knoten des Clusters existiert.

Der Cluster auf der rechten Seite beinhaltet einen Zyklus. Dies verwehrt eine einfache Verrechnung der Kantengewichte miteinander, da der Zyklus theoretisch unendlich oft abgesritten werden kann, bevor der Benutzer eventuell mit k fortsetzt. Zwar lässt sich die Wahrscheinlichkeit des unendlichen Abschreitens in diesem Fall mittels Konvergenz der geometrischen Reihe bestimmen [20]. Für die Praxis hat dieser Wert aber natürlich keine Aussagekraft. Wie dieser Cluster letztendlich zu bewerten ist, wird nicht weiter ausgearbeitet. Als Erkenntnis wird jedoch gewonnen, dass Zyklen in Clustern bei zukünftigen Ansätzen besser vermieden werden sollten. Neben dem Zyklusproblem der Knotenlisten erweisen sie sich nun zusätzlich für die Bewertung von Clustern als hinderlich.

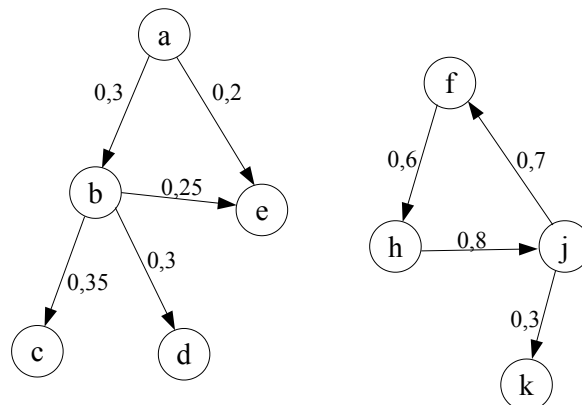


Abbildung 3.5: Clusterbewertung - Problem

3.2.5 Vor- und Nachteile

Der Vorteil des einfachen Tiefenansatzes ohne Aufwandsverbesserung ist seine Einfachheit. Es wurde jedoch gezeigt, dass der Aufwand im schlechtesten Fall quadratisch ist. Deshalb wurde dieser durch die Verwendung der Knotenlisten verbessert. Dies führte aber, vornehmlich durch das Zyklenproblem, zu einer deutlichen Verkomplizierung des Verfahrens. Weiter zeigte sich, dass Zyklen die Clusterbewertung deutlich erschweren.

Ein Blick auf die erzeugten Cluster zeigt noch weitere Probleme auf, welche noch nicht angesprochen wurden.

Mit der Clustersemantik der eben vorgestellten Ansätze können, bei entsprechend vorhandenen Kanten, Cluster gebildet werden, die eine Vielzahl an Verzweigungen enthalten. Dies liegt darin begründet, weil Cluster von Sitzungsstartknoten ausgehend aufgebaut werden und ein Cluster sämtliche Pfade von semantisch starken Kanten enthält, die ausgehend vom entsprechenden Sitzungsstartknoten existieren. Diese Verzweigungen führen dazu, dass Paare von Knoten zu einem Cluster gehören können, zwischen denen kein Pfad im Graphen existiert. Im Beispiel aus Abb. 3.5 gilt dies zum Beispiel für die Knoten c und e . Die Nichtexistenz eines Pfades bedeutet nach der Semantik der Kanten des Graphen aus Abschnitt 2.2.3, dass kein Benutzer je die beiden betrachteten Knoten nacheinander angefordert hat, ohne zwischendurch entweder die Sitzung zu beenden oder zu einer bereits besuchten Seite zurückzukehren. Durch diese Verzweigungen entstehen in der Regel sehr große Cluster. In Abschnitt 3.1.1 wurde jedoch bereits erörtert, warum große Cluster unerwünscht sind.

Verzweigungen sind daher in Clustern unerwünscht.

3.2.6 Breitenansatz

Anstatt der Tiefensuche kann natürlich ebenso die Breitensuche als Basis eines Ansatzes genommen werden. Statt eines rekursiven Verfahrens muss hier eine Warteschlange nach dem FIFO-Prinzip (engl. *first in first out*) benutzt werden, damit der Breitendurchlauf gewährleistet wird. Das Verwenden einer solchen Warteschlange macht ein solches Verfahren jedoch unbrauchbar für das Clusterproblem. Es führt dazu, dass Cluster nicht nacheinander aufgebaut werden, sondern gleichzeitig. Nach dem Startknoten g werden zunächst sämtliche Sitzungsstartknoten besucht. Es sind zu diesem Zeitpunkt zum Beispiel keinerlei Informationen über Verbindungen zwischen diesen Knoten bekannt. Das bedeutet, dass eventuell später über eine Zusammenlegung von aufeinandertreffenden Clustern entschieden werden muss.

Durch das parallele Aufbauen der Cluster muss außerdem jederzeit festgestellt werden können, welcher Cluster im Moment eigentlich bearbeitet wird. Es zeigt sich also, dass im Vergleich zum Tiefenansatz bereits ein erster Ansatz sehr viel komplizierter ausfallen würde.

Zusätzlich wurde in Abschnitt 3.2.5 bereits begründet, warum Verzweigungen in Clustern unerwünscht sind. Dadurch, dass beim Breitendurchlauf aber alle Nachfolger eines Knotens direkt aufeinander folgend besucht werden, eignet sich dieses Verfahren vornehmlich für das Erstellen eben solcher Cluster.

Aus obigen Gründen wird auf die Erarbeitung eines Breitenansatzes verzichtet und das Hauptaugenmerk auf die Entwicklung rekursiver Ansätze gelegt.

3.3 Aufwärtsansatz

Nachdem in Abschnitt 3.2.5 festgestellt wurde, dass Verzweigungen in Clustern unerwünscht sind, wird für die Weiterentwicklung des Tiefenansatzes angestrebt, dass Cluster Pfade ohne Verzweigungen repräsentieren. Wie sich dies in einen Algorithmus einbauen lässt und welche neuen Probleme dabei auftauchen, wird in diesem Abschnitt erläutert.

3.3.1 Clustersemantik

Zur Bildung der Semantik wird die Forderung dahingehend erweitert, dass die Cluster Pfade enthalten sollen, die Folgen von Webseiten repräsentieren, die von Benutzern häufig angefordert werden. Eine solche, häufig aufgerufene Webseitenfolge zeichnet sich wiederum durch einen Pfad semantisch starker Kanten im Informationsgraphen aus. Diese Pfade können an einem Sitzungsstartknoten oder nach einer semantisch schwachen Kante beginnen. Ein Cluster darf nur an Knoten enden, welche entweder gar keinen semantisch starken Nachfolger besitzen oder nur solche, die einen Zyklus mit dem abgeschrittenen Pfad bilden würden. Formal festgehalten hat ein Cluster die folgende Form:

Definition 9 Sei E die Menge aller Kanten im Informationsgraphen und weiter $x_0 \dots x_n$ eine Folge von Knoten. Weiter sei B die Menge aller Sitzungsstartknoten. Ferner sei $s(x, y)$ die semantische Distanz zweier Knoten mit $(x, y) \in E$ und s_{th} ein Grenzwert für die semantische Distanz. Für einen Cluster $C = \{x_0, \dots, x_n\}$ muss folgendes gelten:

$$\begin{aligned} & ((x_i, x_{i+1}) \in E \wedge s(x_i, x_{i+1}) < s_{th}) \quad \text{für } 0 \leq i < n \\ & \wedge (x_0 \in B \vee \exists (y, x_0) \in E : s(y, x_0) \geq s_{th}) \\ & \wedge \neg \exists (x_n, q) \in E : (s(x_n, q) < s_{th} \wedge q \notin C) \end{aligned}$$

3.3.2 Einfache Version

Diese einfache Version ist eine Weiterentwicklung des ersten Tiefenansatzes und liefert die in Def. 9 festgelegte Pfadform der Cluster. Da die Cluster hier nicht direkt beim rekursiven Abstieg erstellt werden, sondern erst beim Zurückgehen aus der Rekursion, wird dieses Verfahren *Aufwärtsansatz* genannt. In Algorithmus 2 ist die einfache Art dieses Ansatzes in Pseudocode aufgezeigt.

Durch die Rekursion wird ein Pfad des Graphen immer weiter abgeschritten, so lange eine von zwei Bedingungen erfüllt ist. Die Rekursion wird fortgesetzt, wenn der nächste Nachfolger durch eine semantisch starke Kante verbunden ist, die keinen Zyklus bildet, oder der Nachfolger durch eine semantisch schwache Kante verbunden ist und noch nicht besucht wurde. Ein Cluster wird eröffnet und in die Clusterliste eingetragen, wenn ein Knoten nur semantisch schwache ausgehende Kanten besitzt oder alle ausgehenden, semantisch starken Kanten Zyklen schließen. Diese Clusterliste wird an den aufrufenden Vorgänger zurückgegeben.

In Abb. 3.6, die das Ergebnis des Algorithmus an einem Beispiel zeigt, wird bei Knoten p ein Cluster eröffnet, da p keinen semantisch stark verbundenen Nachfolger besitzt. Daraufhin

bekommt Knoten l die Clusterliste zurück und fügt sich selbst zum Cluster hinzu. Knoten j erhält drei Clusterlisten zurück aus seinen Aufrufen. Die Liste von a wird jedoch ignoriert, da die Kante zu a semantisch schwach ist. Zu den beiden Clustern, die j von l und q erhält, fügt sich j hinzu und gibt die beiden in seiner Liste an k zurück. Wenn c von a aus aufgerufen wird, erkennt c den Zyklus der Kante zu a und bricht den Abstieg in dieser Richtung ab. Ebenso wird c von q aus nicht aufgerufen, wenn man annimmt, dass c bereits besucht wurde. Nach diesem einfachen Prinzip entstehen die Cluster wie abgebildet.

Algorithmus 2 Aufwärtsansatz

```

for all  $n \in N(g)$  do
  clusterNachfolger( $n$ );
end for

function clusterNachfolger(Knoten  $x$ ) return Clusterliste {
   $x$ .markiere();//zur Zyklenerkennung
   $x$ .wurdeBesucht(true);
  Clusterliste  $l$  = new Clusterliste();
  for all  $n \in N(x)$  do
    if  $s(x, n) < s_{th} \wedge n$ .istMarkiert() == false then //starke Kante + kein Zyklus
       $l$ .fügeHinzu(clusterNachfolger( $n$ ));
    else if  $s(x, n) \geq s_{th} \wedge n$ .besucht() == false then
      clusterNachfolger( $n$ ); //Rückgabewert unwichtig
    end if
  end for
  if  $l$ .istLeer() then
     $l$ .fügeHinzu(erstelleCluster( $x$ )); //kein starker Nachfolger, also neuer Cluster
  else
    for all  $c \in l$  do
       $c$ .fügeHinzu( $x$ );
    end for
  end if
   $x$ .entferneMarkierung();
  return  $l$ ;
}

```

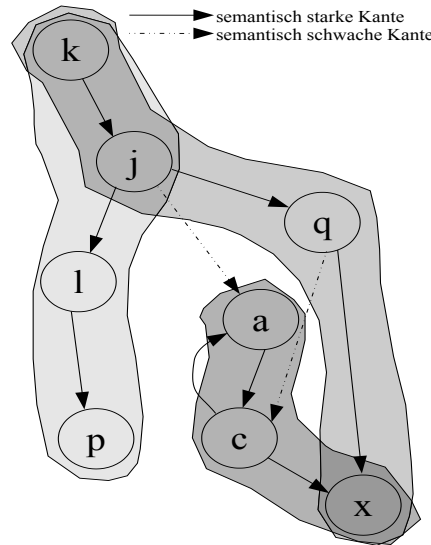


Abbildung 3.6: Aufwärtsansatz - Beispiel

3.3.3 Bewertung der Cluster

Aus der Pfadsemantik ergibt sich auch ein Vorteil für die Bewertung der Cluster. Wie in diesem Abschnitt gezeigt wird, lassen sich Pfade repräsentierende Cluster einfach bewerten und die Probleme aus Abschnitt 3.2.4 können umgangen werden.

Die Clustergröße lässt sich wieder durch einfaches Aufsummieren der Größen der enthaltenen Webseiten berechnen.

Mit dem inneren Gewicht wird hier die Wahrscheinlichkeit wiedergegeben, mit der ein Benutzer zum betrachteten Cluster im Graphen gelangt, diesen abschreitet und seinen Abstieg genau an jener Stelle beendet, an der auch der Cluster endet. Im Beispiel aus Abb. 3.7 beträgt die Wahrscheinlichkeit, dass der Benutzer auf dem betrachteten Pfad zum Cluster gelangt: $P_{Pfad} = 0,1 * 0,003 = 0,0003$. Weiter ist die Wahrscheinlichkeit, dass der Pfad im Cluster abgeschritten wird: $P_{Cluster} = 0,8 * 0,45 = 0,36$. Zuletzt ist die Wahrscheinlichkeit, dass der Benutzer seinen Abstieg am letzten Knoten des Clusters beendet, die Summe aller Gewichte von Zyklus- und Endkanten (hier also: $P_{Ende} = 0,3 + 0,05 = 0,35$). Die Verfolgung jeder anderen, im Beispiel nicht eingezeichneten, Art von Kante von d aus, würde eine Fortsetzung des Abstiegs im Graphen bedeuten. Die Gesamtwahrscheinlichkeit für das innere Gewicht P_i ergibt sich dann aus dem Produkt der einzelnen Wahrscheinlichkeiten:

$$P_i = P_{Pfad} * P_{Cluster} * P_{Ende} = 0,0003 * 0,36 * 0,35 = 0,0000378$$

Es wird deutlich, warum die Berechnung des inneren Gewichts für die Cluster des Aufwärtsansatzes einfach ist. Da die Cluster Pfade repräsentieren und keine Verzweigungen enthalten können, lassen sich $P_{Cluster}$ und P_{Ende} durch simples Aufmultiplizieren bzw. Aufsummieren von Kantengewichten berechnen.

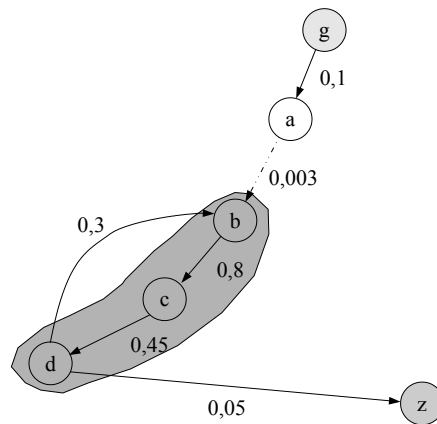


Abbildung 3.7: Clusterbewertung - Inneres Gewicht

3.3.4 Aufwandsverminderung

Wie bereits beim Tiefenansatz, besitzt das einfache Verfahren eine höhere Komplexität als das angestrebte $O(|V| + |E|)$. Mit dem Beispielgraphen aus Abb. 3.3, der hier noch nicht einmal den schlechtesten Fall darstellt, lässt sich auch für diesen Algorithmus ein möglicher quadratischer Aufwand nachweisen. Im folgenden werden drei Ansätze untersucht, wie der Aufwand verringert werden könnte.

1. Verwendung von Knotenlisten

Wie auch beim Tiefenansatz wird versucht, den Aufwand durch das Verwenden von Knotenlisten einzuschränken. Jedoch verkompliziert auch hier die Verwendung dieser Listen den Algorithmus erheblich. Neben der wieder durchzuführenden Zyklenerkennung ergeben sich noch weitere Punkte, die Berücksichtigung verlangen. Die Knotenlisten sind hier nicht unbedingt eindimensional, da die Nachfolger eines Knotens nicht notwendigerweise alle im gleichen Cluster enthalten sein müssen (siehe Nachfolger von Knoten j in Abb. 3.6). In Abb. 3.8 ist beispielhaft dargestellt, wie die Knotenlisten an den einzelnen Knoten des Zyklus aussehen, wenn dieser zunächst von b ausgehend abgeschritten wird. Wird der Zyklus später erneut angetroffen, diesmal von c ausgehend, so wird deutlich, dass die Knotenliste so nicht verwendbar ist. Das Problem bei diesem Ansatz ist, dass die resultierenden Cluster verschieden aussehen, abhängig vom Einstiegspunkt in den Zyklus. In diesem Beispiel werden von b ausgehend zwei Cluster (b, d, e, a und b, d, e, f) erstellt. Von c ausgehend würde nach Algorithmus 2 lediglich ein Cluster (c, f, d, e, a) erzeugt. Die Abhängigkeit vom Einstiegspunkt in den Zyklus erschwert eine Synchronisierung der Knotenlisten der Zyklenelemente so erheblich, dass dieser Ansatz nicht weiter verfolgt wird. Da die Verwendung von Knotenlisten nicht in Frage kommt, erfordert dies die Erarbeitung anderer Ansätze, die nachfolgend erläutert werden.

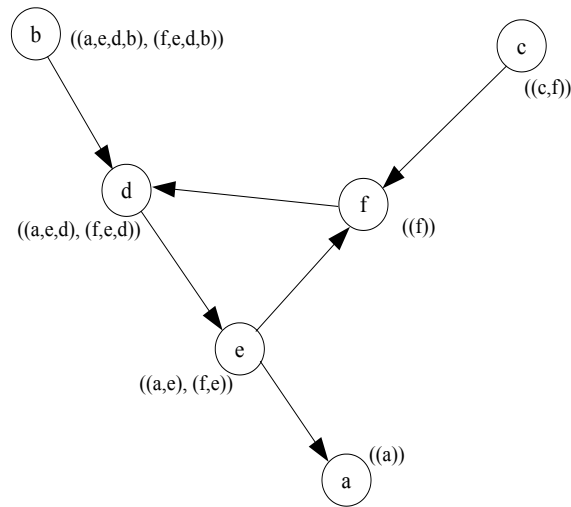


Abbildung 3.8: Aufwärtsansatz - Zyklusproblem

2. Vernachlässigung unwahrscheinlicher Cluster

Ein weiterer Ansatz, der bei den Clusteransätzen bisher nicht berücksichtigt wurde, offenbart sich in Abschnitt 3.1.1. Es dürfte aufgefallen sein, dass das innere Gewicht des Beispielclusters aus Abb. 3.7 sehr klein ausfällt und dies vor allem dem geringen Gewicht der semantisch schwachen Kante (a, b) zu verdanken ist. Es wurden zwar bislang noch keine semantischen Distanzen definiert, doch es ist offensichtlich, dass das Kantengewicht dabei eine wichtige Rolle spielen wird. Es darf also davon ausgegangen werden, dass semantisch schwache Kanten auch ein eher geringes Kantengewicht besitzen. Bei den bisherigen Ansätzen gehen Cluster entweder von einem Sitzungsstartknoten aus oder beginnen an einem Knoten, der auf eine semantisch schwache Kante folgt (wie b im Beispiel). Es stellt sich nun die Frage, ob Cluster, die nicht von einem Sitzungsstartknoten ausgehen, überhaupt aufgebaut werden sollen. Da auf dem Pfad, auf dem sie erreicht werden, zumindest eine semantisch schwache Kante liegen muss, werden diese Cluster ein geringes inneres Gewicht besitzen. Es werden daher zukünftig nur noch Cluster von Sitzungsstartknoten ausgehend aufgebaut.

Das Vernachlässigen der Cluster mit einem geringen inneren Gewicht bringt zweierlei Vorteile mit sich:

- Der Aufwand des Verfahrens wird kleiner, da die Tiefe des rekursiven Abstiegs nun durch die semantisch schwachen Kanten beschränkt wird.
- Es müssen keine Abhängigkeiten zwischen den Clustern berücksichtigt werden. Würde der Cluster wie in Abb. 3.7 aufgebaut, so müsste bei der Vorabübertragung die Abhängigkeit des Clusters (b, c, d) von Knoten a berücksichtigt werden. Es müsste sichergestellt werden, dass vor dem Cluster (b, c, d) ein Cluster übertragen wird, der den

Knoten a enthält. Begründet wird dies durch die Definition des Informationsgraphen, die für dieses Beispiel aussagt, dass es keinen Benutzer gibt, der zu den Knoten des Clusters (b, c, d) gelangt, ohne vorher a besucht zu haben. Dies würde jedoch einen weiteren Graphen verlangen, der die Abhängigkeiten zwischen den Clustern charakterisiert. Dies wiederum würde das Verfahren zur Clusterauswahl für die Vorabübertragung erheblich verkomplizieren und den Gesamtaufwand erhöhen.

3. Verwendung von Grenzwerten

Um den Aufwand weiter einschränken zu können, werden zwei Grenzwerte eingeführt. Diese sollen verhindern, dass Cluster zu groß bzw. die inneren Gewichte zu klein werden.

Das bedeutet, dass beim rekursiven Abstieg die aufmultiplizierten Kantengewichte, also das Gewicht des abgeschrittenen Pfades mitgegeben werden müssen. Beim Unterschreiten eines Grenzwerts w_{pth} wird dann auf jeden Fall der Abstieg beendet. So können zu unwahrscheinliche Cluster vermieden werden.

Ebenso sind zu große Cluster, die eventuell nicht in den Speicher des Endgeräts passen, sinnlos und brauchen nicht erstellt zu werden. Hierfür wird der Grenzwert γ_{th} verwendet. Beim rekursiven Abstieg muss also auch die Summe der Seitengrößen des bisherigen Pfades bekannt sein.

Aufwand

Die Verwendung von w_{pth} als Grenzpfadgewicht ermöglicht eine einfache Beschreibung des Aufwands für den schlechtesten Fall. Im Informationsgraphen gilt nach Definition 7 für jeden Knoten außer z , dass sich die Summe aller ausgehenden Kantengewichte zu 1 ergibt. Hierdurch ergibt sich, dass die Summe der Gewichte aller möglichen Pfade von g zu z wiederum 1 ergibt, wie in Abb. 3.9 beispielhaft dargestellt ist. Nach der Definition des Informationsgraphen

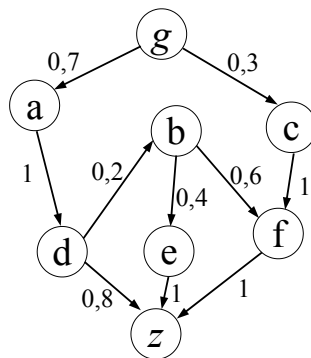


Abbildung 3.9: Summe aller Pfadgewichte - Beispiel

besitzt nur z keine ausgehenden Kanten und ist von jedem Knoten aus erreichbar. Ebenso existiert von g aus zu jedem Knoten ein Pfad. Im Beispiel aus Abb. 3.9 sind die vier Pfade (g, a, d, z) , (g, a, d, b, e, z) , (g, a, d, b, f, z) und (g, c, f, z) enthalten. Außerdem ist die Summe

aller ausgehenden Kanten für jeden Knoten außer z wie gefordert 1. Summiert man nun die vier Pfadgewichte, so ergibt sich: $0,56 + 0,056 + 0,084 + 0,3 = 1$.

Im Algorithmus wird durch den rekursiven Abstieg kein Pfad mehr als einmal abgegangen. Ebenso ist kein Pfad in einem anderen komplett enthalten. Da die abgestiegenen Pfade alle unterschiedlich sind und jeweils ein Pfadgewicht größer w_{pth} besitzen, kann es also nicht mehr als $\frac{1}{w_{pth}}$ verschiedene solche Pfade geben. Da der Algorithmus keine Zyklen abschreitet, kann jeder Pfad höchstens die Länge $|V|$ besitzen. Damit berechnet sich der durch rekursive Aufrufe entstehende Aufwand für den schlechtesten Fall, mit einem $w_{pth} \in]0..1]$, zu: $\frac{1}{w_{pth}} * |V|$. Dies ist zwar linear in $|V|$, jedoch bestimmt das gewählte w_{pth} natürlich den konstanten Faktor. Es ist also abzusehen, dass die Wahl des w_{pth} großen Einfluss auf die Laufzeit des Verfahrens haben wird.

3.3.5 Detaillierter Aufwärtsansatz

In diesem Abschnitt wird der Aufwärtsansatz insofern erweitert, dass die Cluster sich die für ihre Bewertung wichtigen Attribute des inneren Gewichts und der Clustergröße merken. Außerdem werden die gewonnenen Erkenntnisse aus Abschnitt 3.3.4 eingebaut, indem der rekursive Abstieg entlang semantisch schwacher Kanten nicht mehr vollzogen wird und die Grenzwerte für Pfadgewicht und Clustergröße eingebaut werden. Neben den direkten Folgerungen der letzten Abschnitte ergibt sich noch ein weiterer Punkt, der ebenfalls in diesen Ansatz eingebaut wird:

Bis jetzt ist laut Definition 9 die Bedingung für ein Clusterende genau dann erfüllt, wenn es keinen weiteren Bedarf für einen rekursiven Abstieg mehr gibt. Es ist jedoch sinnvoller die Cluster an den Stellen enden zu lassen, an denen viele Benutzer ihren Abstieg im Graphen abbrechen. Deshalb wird die Definition für ein Clusterende erweitert.

Besitzt ein Knoten sowohl eine semantisch starke Kante zu einem Nachfolger, als auch ein P_{Ende} größer einem Grenzwert $w_{end_{th}}$, so lasse einen Cluster an diesem Knoten enden und führe einen weiteren entlang der semantisch starken Kante fort.

Bei genauerer Betrachtung von Algorithmus 3 fällt auf, dass der Name Aufwärtsansatz eigentlich nicht länger zutreffend ist. Die Cluster werden nicht mehr beim Zurückgehen aus der Rekursion aufgebaut, sondern beim letzten Knoten, der erkennt, dass ein Cluster gebildet werden muss. Dies liegt daran, dass hier die Knotenliste direkt beim rekursiven Abstieg, mitsamt den Größen- und Gewichtsinformationen, mitgegeben wird. In Abbildung 3.10 ist links eine Momentaufnahme während des Ablaufs des Algorithmus gezeigt. Auf der rechten Seite sind die fertigen Cluster des Beispiels zu sehen, deren innere Gewichte in der darunterliegenden Tabelle dokumentiert sind.

Algorithmus 3 Detaillierter Aufwärtsansatz

```

for all  $n \in N(g)$  do
    clusterNachfolger( $n$ , (), 0, ( $g, n$ ). $w$ );
end for

function clusterNachfolger(Knoten  $x$ , Knotenliste  $l$ , int  $groesse$ , float  $innGew$ ){
 $x$ .markiere();//zur Zyklenerkennung
 $l$ .fügeHinzu( $x$ );
 $groesse$  +=  $x$ .groesse();
boolean clusterFortgesetzt = false;
float endkantengewicht = 0;
for all  $n \in N(x)$  do
    if  $s(x, n) < s_{th} \wedge n$ .nichtMarkiert()  $\wedge n \neq z \wedge$ 
 $groesse + n$ .groesse()  $< \gamma_{th} \wedge innGew * (x, n).w > w_{pth}$  then //Fortsetzungsbedingungen
        clusterNachfolger( $n$ ,  $l$ ,  $groesse$ ,  $innGew * (x, n).w$ );
        clusterFortgesetzt = true;//es existiert ein starker Nachfolger
    else if  $n$ .istMarkiert()  $\vee n = z$  then //Endkante
        endkantengewicht += ( $x, n$ ). $w$ ;
    end if
end for
if endkantengewicht  $> w_{end_{th}} \vee clusterFortgesetzt = false$  then
    erstelleCluster( $l$ ,  $groesse$ ,  $innGew * endkantengewicht$ );
end if
 $x$ .entferneMarkierung();
}
    
```

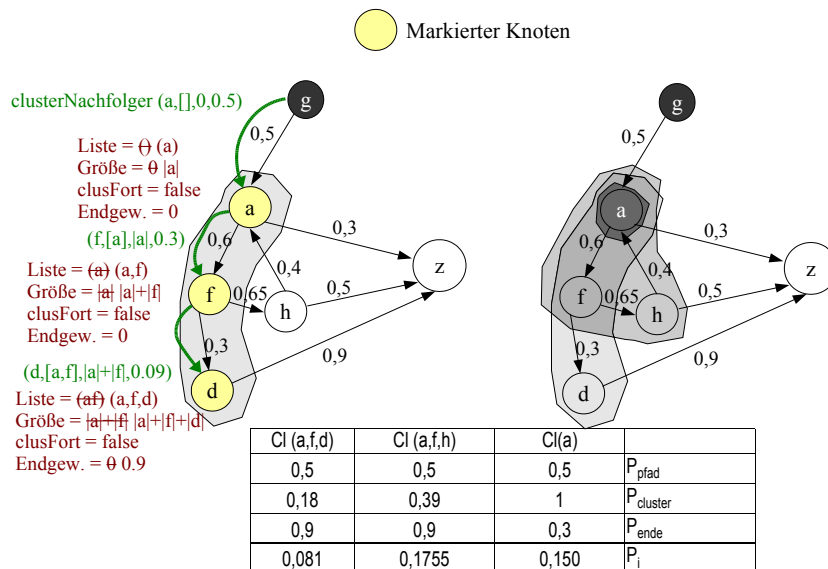


Abbildung 3.10: Detaillierter Aufwärtsansatz - Beispiel

3.3.6 Vor- und Nachteile

Der in diesem Abschnitt behandelte Ansatz liefert gut bewertbare Cluster. Im Vergleich zum Tiefenansatz fallen die Cluster durch ihre Pfadeneigenschaft auch kleiner aus und folgen einer detaillierteren Semantik. Es bleibt aber die Frage offen, wie diese Semantik der Cluster noch verfeinert werden kann und muss. So wurde bisher weder die Inhaltseigenschaft von Webseiten berücksichtigt noch wurden semantische Distanzen definiert. Beide Punkte können der Verfeinerung der Semantik dienen, die bisher lediglich aussagt, dass Knoten entlang eines Pfades semantisch starker Kanten zu einem Cluster zusammengefasst werden. Die Cluster können theoretisch komplette Sitzungen bzw. lange Folgen von Webseitenaufrufen bis zum Wiederbesuch einer Seite enthalten und damit immer noch sehr groß werden.

Der Aufwand des Verfahrens kann durch die Verwendung von Knotenlisten in Abschnitt 3.3.4 nicht auf einfache Art reduziert werden. Es muss weiter in Kauf genommen werden, dass Knoten mehrmals besucht werden können. Die Verwendung des Grenzwertes w_{pth} für das Pfadgewicht erlaubt jedoch eine Einschränkung der rekursiven Aufrufe auf linearen Aufwand $\frac{1}{w_{pth}} * |V|$. Auch wenn der konstante Faktor sehr groß werden kann, so ist er doch von einem Administrator frei wählbar.

Trotz der Nachteile bietet der detaillierte Ansatz jedoch eine gute Grundlage, an die das letztendliche Verfahren angelehnt werden kann.

Kapitel 4

Clusterverfahren

Nach der Vorstellung der ersten Ansätze und deren Weiterentwicklungen wird in diesem Kapitel auf das letztendlich verwendete Clusterverfahren eingegangen. Ziel ist es, zunächst die Semantik der Cluster zu verfeinern. Neben dem Algorithmus zum Aufbau der Cluster wird zudem detailliert auf die Bewertung selbiger eingegangen.

4.1 Semantik der Cluster

Nachdem verschiedene Arten der Clusterbildung inklusive Semantiken untersucht wurden, wird nun betrachtet, welche Semantik die zu erstellenden Cluster des letztendlich angewendeten Verfahrens besitzen sollen. Die relativ grobe Semantik des Aufwärtsansatzes lässt zu, dass möglicherweise komplette Sitzungen eines Benutzers zu einem Cluster zusammengefasst werden können. Dies kann wiederum in großen Clustern resultieren, die nicht erwünscht sind. Mit Hilfe einer Knoteneigenschaft des Informationsgraphen, die bisher noch nicht weiter verwendet wurde, wird eine feinere Clustersemantik festgelegt und die Struktur der Cluster definiert.

Internetverhalten von Benutzern Ein in [5] und [24] festgestelltes, häufiges Verhalten beim Durchforsten des Internets ist es, dass der Benutzer auf der Suche nach bestimmten Informationen eine Folge von Webseiten besucht, bis er schließlich zur gewünschten Seite gelangt. Solche Folgen von Webseiten, welche die Suche von Benutzern repräsentieren, werden folgend als *Suchpfad* bezeichnet. Wird nun jedoch mit dem bisher verwendeten Tiefen- und Breitensuchverfahren nicht die gesamte Folge der Webseiten in den Speicher des Endgeräts übertragen, sondern nur ein Teil davon, so wird der Benutzer seine Suche nicht abschließen können. An irgendeinem Punkt auf dem Weg zum Ziel wird eine benötigte Seite nicht zur Verfügung stehen, so dass der Benutzer die gesuchte Seite, zumindest auf diesem Weg, nicht finden kann. Dies ist natürlich kein zufriedenstellendes Ergebnis für den Benutzer. Sinnvoller ist es in diesem Fall also, entweder die ganze Folge von Webseiten, die den Suchpfad bildet, zu übertragen oder gar keine davon, damit kein Speicherplatz verschwendet wird. Um dieses Ziel zu erreichen, wird das hier vorgestellte Clusterverfahren eingesetzt.

Die Semantik eines Clusters ist es also, die Knoten eines möglichen Suchpfades im Informationsgraphen zu gruppieren. Ein Cluster ist aufgrund dieser Pfadsemantik ein Teilgraph

des Informationsgraphen, der keine Verzweigungen enthält. Bereits in Abschnitt 3.2.5 wurde festgestellt, dass Verzweigungen enthaltende Cluster nicht sinnvoll sind.

4.2 Clusterstruktur

Bevor die Struktur der Cluster, welche auf die eben beschriebene Semantik aufbaut, definiert werden kann, muss zunächst etwas Vorarbeit geleistet werden. Zuerst wird die Inhaltseigenschaft von Knoten beleuchtet. Aus dieser lässt sich anschließend eine Struktur für Suchpfade festlegen. Mit dieser gegebenen Suchpfadstruktur lässt sich dann auch die Clusterstruktur endgültig definieren.

4.2.1 Inhaltseigenschaft von Knoten

Um zu definieren wie ein Suchpfad im Graphen aussieht, ist es notwendig, einen Blick auf eine Knoteneigenschaft zu werfen, die sich aus der Definition 6 des Informationsgraphen ableiten lässt. Jeder Knoten im Informationsgraphen besitzt eine so genannte *Inhaltswahrscheinlichkeit*. Diese Wahrscheinlichkeit ist gleich dem Anteil der Besucher der Seite, welcher die vom Knoten repräsentierte Seite als Inhaltsseite angesehen hat. Im Informationsgraphen besitzt laut Def. 6 jeder Knoten ein Attribut *mainStat*, dessen Wert aussagt, wie häufig die entsprechende Webseite von Besuchern als Inhaltsseite eingestuft wurde.

Definition 10 Sei V die Menge an Knoten des Informationsgraphen und $x \in V$ ein Knoten daraus. Dann ist die *Inhaltswahrscheinlichkeit* von x wie folgt definiert:

$$p_{\text{main}}(x) = \frac{x.\text{mainStat}}{x.n}$$

Analog zum hergeleiteten Inhaltstyp für Logeinträge aus Abschnitt 2.2.2 wird ein solches Attribut auch für die Knoten des Informationsgraphen eingeführt. Jeder Knoten erhält ein Attribut *type*, das die Werte *transit* für eine Navigationsseite und *main* für eine Inhaltsseite annehmen kann. Eine Seite soll genau dann als Inhaltsseite gelten, wenn ihre Inhaltswahrscheinlichkeit einen Schwellwert $p_{\text{main}_{th}}$ überschreitet.

Definition 11 Sei $p_{\text{main}}(x)$ die Inhaltswahrscheinlichkeit für Knoten $x \in V$. Dann gilt:

$$x.\text{type} = \text{main} \iff p_{\text{main}}(x) > p_{\text{main}_{th}}$$

4.2.2 Aufbau eines Suchpfades

Nachdem nun der Inhaltstyp auch für die Knoten eingeführt wurde, kann formal definiert werden, wie die Struktur eines Suchpfades im Graphen aussieht. Ein Suchpfad ist eine beliebig lange Folge von Navigationsseiten, gefolgt von mindestens einer Inhaltsseite. Da die gesuchten Inhalte einer Suche nicht unbedingt nur auf eine Inhaltsseite beschränkt sein müssen, ist es sinnvoll, eine Folge mehrerer Inhaltsseiten als Abschluss eines Suchpfades zu erlauben.

Für den Beginn eines Suchpfades soll gelten, dass ein solcher Pfad entweder bei einem Sitzungsstartknoten oder nach einem Inhaltsknoten beginnen kann. Damit wird verhindert,

dass ein Suchpfad bei einer Navigationsseite beginnt, die als Vorgänger im Graph selbst nur Navigationsseiten besitzt und damit mindestens das zweite Element eines Suchpfades sein muss. Formal sieht dies folgendermaßen aus:

Definition 12 Sei $transit^*$ ein beliebig langer Pfad von Navigationsseiten (auch leerer Pfad möglich) und $main^+$ ein nicht leerer Pfad von Inhaltsseiten.

Ein **Suchpfad** S ist ein Pfad der folgenden Form:

$$S = transit^* main^+, \text{ wenn zusätzlich gilt:}$$

Ist der Pfad von Navigationsseiten $transit^*$ nicht leer, so muss eine Kante von der letzten Navigationsseite zur ersten Inhaltsseite des Pfades $main^+$ existieren.

Sei weiter x der erste Knoten des Suchpfades S , dann muss gelten:

$$\exists y \in V : (y, x) \in E \wedge (y.type = main \vee y = g)$$

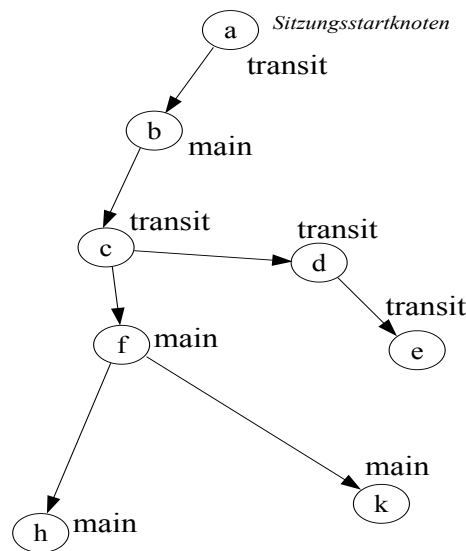


Abbildung 4.1: Suchpfade - Beispiel

In Abb. 4.1 ist ein Graph gegeben, der Suchpfade nach obiger Definition enthält. Die Pfade (a, b) , (c, f) , (c, f, h) und (c, f, k) sind offensichtlich Suchpfade. Nicht vergessen werden darf jedoch, dass nach Definition 12 auch die einelementigen Pfade (h) und (k) Suchpfade darstellen. Dies ist sinnvoll, da nicht unbedingt davon ausgegangen werden kann, dass sowohl h als auch k in inhaltlichem Zusammenhang zu f stehen. Der Pfad (c, d, e) hingegen stellt keinen Suchpfad dar, da er nicht mit einer Inhaltsseite endet.

4.2.3 Festlegung der Clusterstruktur

Nachdem festgelegt wurde wie Suchpfade aufgebaut sind, gilt es nun die Struktur der Cluster zu definieren.

Beginn eines Clusters: Ein Suchpfad kann laut Definition 12 entweder nach einem Inhaltsknoten oder an einem Sitzungsstartknoten beginnen. Begännen die Cluster an denselben Knoten, so würde dies das Problem mit sich bringen, dass es Abhängigkeiten zwischen Clustern geben könnte. In Abb. 4.2 ist links zu sehen, dass der Cluster (f, h) nur von Cluster (a, b, c) aus erreichbar ist. Es würde sich hiermit eine Abhängigkeit ergeben. Da die Berücksichtigung von Clusterabhängigkeiten, wie in Abschnitt 3.3.4 bereits festgestellt, die Erstellung der Liste zur Vorabübertragung verkomplizieren würde, wird ein anderer Ansatz verfolgt.

Es werden ausschließlich Cluster von Sitzungsstartknoten aus aufgebaut. In Abb. 4.2 sei a ein Sitzungsstartknoten. Auf der rechten Seite gibt es nun keine Abhängigkeiten mehr bzw. wird die Abhängigkeit indirekt modelliert, indem alle Vorgängerknoten des eigentlichen Clusters auch aufgenommen werden. Ein Cluster repräsentiert damit also nicht mehr nur einen Suchpfad S , sondern zusätzlich noch die Suchpfade, die auf dem beschrittenen Weg zu S liegen. Damit muss beim Erstellen der Vorabübertragungsliste nicht mehr auf Abhängigkeiten geachtet werden. Es ist sichergestellt, dass, wenn ein Cluster in die Liste übernommen wird, dieser auch erreichbar ist.

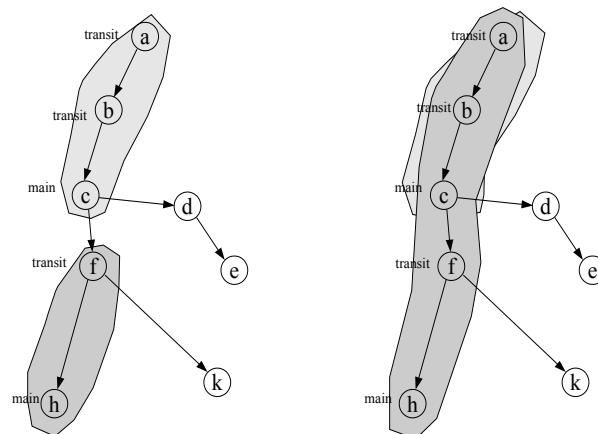


Abbildung 4.2: Wo beginnt ein Cluster?

Ende eines Clusters: Nach Definition 12 endet ein Cluster immer an einem Inhaltsknoten. Es muss jedoch nicht zwangsläufig so sein, dass sich die gewünschte Information auf genau einer Webseite befindet. Die Inhalte können sich durchaus auf mehr als eine Seite verteilen, was in der Definition ebenfalls berücksichtigt wurde. Abb. 4.3 zeigt links einen Pfad, in dem zwei Inhaltsknoten (f und h) direkt aufeinander folgen. Ein Suchpfad könnte also sowohl bei f als auch bei h enden. Daher wird für jedes mögliche Pfadende ein eigener Cluster erstellt. Diese Methode hat in Verbindung mit dem ausschließlichen Beginn der Cluster bei Sitzungsstartknoten den Vorteil, dass die resultierenden Cluster in beiden Fällen identisch

aussehen, egal ob h inhaltlich zu f gehört oder nicht. Würde h einen eigenen, einelementigen Suchpfad bilden, so würden trotzdem sämtliche Vorgängerknoten des abgeschrittenen Pfades mit in den Cluster übernommen werden und somit das gleiche Ergebnis resultieren. Wenn wie im rechten Beispiel von Abb. 4.3 mehrere Inhaltsknoten auf f folgen, so werden analog entsprechend weitere Cluster gebildet.

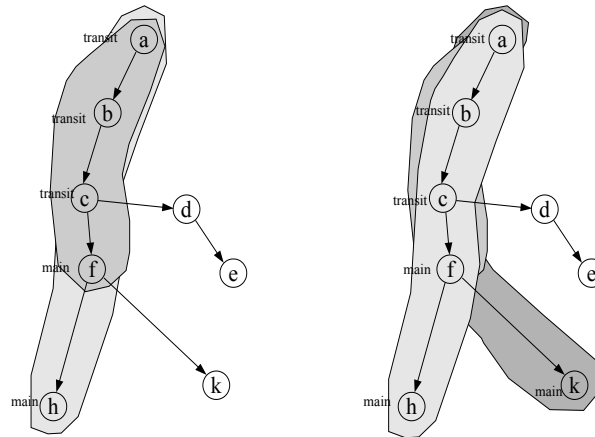


Abbildung 4.3: Wo endet ein Cluster?

Formale Beschreibung: Nachdem nun ausführlich beschrieben wurde wie die Cluster aufgebaut sein sollen, wird die Struktur formal festgehalten.

Definition 13 Ein **Cluster** ist ein Tupel $C = (V_C, E_C, x_0, x_n, pw, size)$, für das gilt:

- V_C ist die Menge an Knoten, die im Cluster enthalten ist und einen Pfad bildet. Es gilt $V_C \subseteq V$ und es existiert eine Reihenfolge x_0, \dots, x_n der Knoten (mit $x_i \in V_C$), die den Pfad wiedergibt.
- $E_C = \{(x_i, x_{i+1}) \mid x_i, x_{i+1} \in V_C \wedge (x_i, x_{i+1}) \in E \wedge 0 \leq i < n\} \cup \{(g, x_0)\}$ ist die Menge an Kanten, die den Pfad des Clusters bildet.
- x_0 ist der Sitzungsstartknoten des Clusters ($x_0 \in B$). Dies ist der einzige Knoten in V_C , für den gilt: $(g, x_0) \in E_C$.
- x_n ist der letzte Inhaltsknoten des Clusters und damit das Ende eines Suchpfades ($x_n.type = main$). Dies ist der einzige Knoten in V_C ohne ausgehende Kante in E_C .
- pw ist das Pfadgewicht des Clusters. Es berechnet sich aus dem Produkt aller Kantengewichte des Clusters. Es gilt:

$$C.pw = \prod_{e \in E_C} e.w$$

- *size ist die Größe des Clusters . Es gilt:*

$$C.size = \sum_{x \in V_C} x.size$$

4.3 Bewertung der Cluster

In Abschnitt 3.1.1 wurden die Kriterien erläutert, die bei der Bewertung von Clustern berücksichtigt werden müssen. Für den detaillierten Aufwärtsansatz wurde in Abschnitt 3.3.3 bereits beispielhaft geschildert, wie die Werte für Clustergröße und inneres Gewicht berechnet werden können. Wie die Größe und vor allem das innere Gewicht für die hier gegebenen Cluster berechnet werden sollen, ist Inhalt dieses Abschnitts. Allein mit der Festlegung wie diese Werte zu bestimmen sind, können die Cluster jedoch noch nicht bewertet werden. Es muss weiter definiert werden, wie diese Werte zueinander zu gewichten sind. Letztendlich soll, als Ergebnis der Bewertung, für jeden Cluster ein einzelner Zahlenwert existieren, der zur Sortierung der Cluster herangezogen werden kann.

4.3.1 Größe und inneres Gewicht

Die Clustergröße wird als Summe über die Größen aller im Cluster enthaltenen Webseiten berechnet. In Def. 13 ist dieser Wert bereits im Tupel enthalten.

Auf das innere Gewicht eines Clusters haben zwei Eigenschaften Einfluß. Das **Pfadgewicht** aus Def. 13 sagt aus, mit welcher Wahrscheinlichkeit ein Benutzer den kompletten Cluster abschreitet. Zusätzlich zum Pfadgewicht ist jedoch noch die **Inhaltswahrscheinlichkeit** des Endknotens ausschlaggebend. Dieser Wert gibt an, mit welcher Wahrscheinlichkeit der Benutzer, der den Pfad des Clusters abgeht, am letzten Knoten auch seinen gesuchten Inhalt findet. Angenommen, man baut einen Cluster auf, der ein sehr hohes Pfadgewicht besitzt, aber gleichzeitig auch eine sehr geringe Inhaltswahrscheinlichkeit hat. Dann gehen zwar sehr viele Benutzer den Pfad des Clusters entlang. Die meisten von ihnen finden aber den gesuchten Inhalt nicht am letzten Inhaltsknoten des Clusters.

Das Beispiel in der nächsten Tabelle demonstriert, dass bei identischem Pfadgewicht natürlich der Cluster gewählt werden sollte, dessen Endseite von mehr Benutzern als Inhaltsseite angesehen wird.

	Cluster 1	Cluster 2
Pfadgewicht	0,11	0,11
Inhaltswahrscheinlichkeit	0,8	0,3

Um Pfadgewicht und Inhaltswahrscheinlichkeit zueinander gewichten zu können, werden die beiden Parameter α und β eingeführt.

Definition 14 *Das innere Gewicht p_i von Cluster C wird folgendermaßen berechnet:*

$$p_i(C) = C.pw^\alpha * p_{main}(C.x_n)^\beta$$

4.3.2 Aufstellen der Bewertungsfunktion

Nachdem nun festgelegt wurde, wie die Clustergröße und das innere Gewicht zu berechnen sind, wird nun auf die Verrechnung dieser beiden Werte eingegangen.

Offensichtlich gilt, dass ein Cluster natürlich umso höher zu bewerten ist, je größer sein inneres Gewicht ist. Weiter ist ein Cluster umso höher zu bewerten, je kleiner er ist. Dadurch wird ein sparsamer Umgang mit dem Platz in der Vorabübertragungsliste gewährt. Im unteren Beispiel ist also Cluster 3 vor Cluster 2 zu wählen, da Cluster 3 ein größeres inneres Gewicht besitzt. Cluster 1 ist vor Cluster 2 zu wählen, da Cluster 1 weniger Platz in der Vorabübertragungsliste belegt.

	Cluster 1	Cluster 2	Cluster 3
Größe	40	60	60
Inneres Gewicht	0,2	0,2	0,25

In diesem Beispiel drängt sich die Frage auf, wie der Vergleich zwischen Cluster 1 und 3 ausfällt. Zwar besitzt Cluster 3 das höhere innere Gewicht im Vergleich zu Cluster 1. Er belegt gleichzeitig aber auch mehr Platz in der Vorabübertragungsliste. Die Bewertung erfolgt letztendlich dadurch, indem man das Verhältnis von innerem Gewicht zur Größe des Clusters als eine Art Steigung betrachtet. Ein Cluster verbessert durch sein inneres Gewicht einerseits die Qualität der Vorabübertragungsliste, braucht andererseits aber gleichzeitig Platz in der Liste auf.

Steigungsansatz Der verwendete Bewertungsansatz sortiert die Cluster nach einem einfachen Prinzip. Vergleicht man zwei Cluster miteinander, so ist immer der Cluster vorzuziehen, welcher der Vorabübertragungsliste den größten Qualitätszuwachs durch sein inneres Gewicht pro Größeneinheit bringt.

Definition 15 Für jeden Cluster C gilt die **Bewertungsfunktion** b :

$$b(C) = \frac{p_i(C)}{C.size}$$

Es erfolgt hier eine indirekte Gewichtung der beiden Quotienten. Durch höhere bzw. niedrigere Wahl von α und β in Def. 14 beeinflusst das innere Gewicht die Bewertung nach dem Steigungsansatz mehr bzw. weniger. In der folgenden Tabelle ist in einem Beispiel dargestellt, wie durch Verdopplung von α und β das innere Gewicht bei der Bewertung stärker berücksichtigt wird. Cluster 2 wird durch die Verdopplung der beiden Parameter höher als Cluster 1 bewertet. Da in diesem Beispiel $\alpha = \beta$ gilt, kann die Multiplikation von Pfadgewicht und Inhaltswahrscheinlichkeit in Zeile 1 vor der Potenzierung erfolgen.

	Cluster 1	Cluster 2
$C.pw * p_{main}(C.x_n)$	0,25	0,5
$C.size$	5	10
$p_i(C)$ für $\alpha = \beta = 1$	0,25	0,5
$b(C)$	0,05	0,05
$p_i(C)$ für $\alpha = \beta = 2$	0,0625	0,25
$b(C)$	0,0125	0,025

In Abb. 4.4 ist eine Visualisierung des Steigungsansatzes anhand eines Beispiels zu sehen. Die Cluster seien hier allesamt disjunkt zueinander, haben also keinen gemeinsamen Knoten.

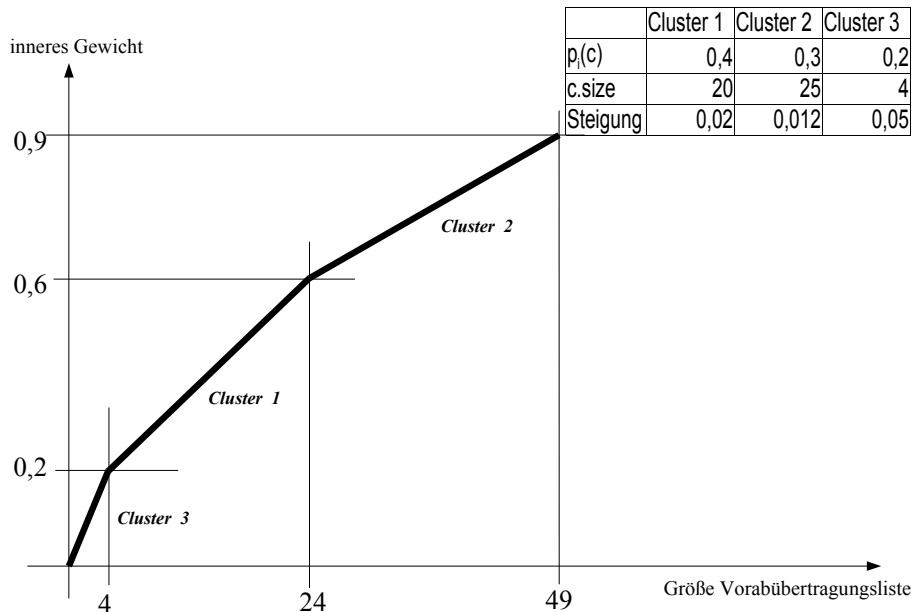


Abbildung 4.4: Visualisierung Steigungsansatz

4.4 Entwicklung des Algorithmus

Da genau wie in Abschnitt 3.3.1 die Pfadeigenschaft der Cluster vorgeschrieben ist, wird der Algorithmus auf den detaillierten Aufwärtsansatz aus Abschnitt 3.3.5 aufbauend entwickelt. Bevor der Algorithmus jedoch in seiner endgültigen Form festgehalten werden kann, müssen zunächst einige Punkte geklärt werden. Es wird darauf eingegangen, wie der Aufwand des Verfahrens in Grenzen gehalten werden kann. Anschließend werden mögliche semantische Distanzen definiert, um die Einhaltung der Struktur der Cluster sicherzustellen. Sind diese Fragen beantwortet, kann der Algorithmus selbst bestimmt werden.

4.4.1 Eingrenzung des Aufwands

Es wurde wiederholt angesprochen, dass die Effizienz des Algorithmus ein kritisches Kriterium darstellt, da der Clusteralgorithmus in periodischen Zeitabständen durchgeführt werden muss. Es gilt für das Clusterverfahren, dass je kürzer die Abstände zwischen den Abläufen sind, umso besser repräsentieren die Ergebnisse jederzeit das aktuelle Benutzerverhalten.

Mit dem Wissen über die Semantik und Struktur eines Clusters wird folgend gezeigt, wieviele Cluster im schlechtesten Fall für einen Informationsgraphen gebildet werden. Für

jeden möglichen Pfad, der zu einem Inhaltsknoten führt, kann ein eigener Cluster erzeugt werden. Der schlechteste Fall stellt sich dar, wenn jeder Knoten des Informationsgraphen, mit Ausnahme von g und z , ein Inhaltsknoten und Sitzungsstartknoten zugleich ist und zusätzlich jeder Inhaltsknoten mit jedem anderen Inhaltsknoten bidirektional verbunden ist.

Es sei $N = V \setminus \{g, z\}$ (damit $|N| = |V| - 2$). In der folgenden Tabelle ist aufgezeigt, wieviele Suchpfade jeweils für eine gegebene Pfadlänge existieren.

Suchpfadlänge	Anzahl der Suchpfade
N	$N * (N - 1) * \dots * 3 * 2 * 1 = \frac{N!}{(N-N)!} = \frac{N!}{0!}$
$N - 1$	$N * (N - 1) * \dots * 3 * 2 = \frac{N!}{(N-(N-1))!} = \frac{N!}{1!}$
$N - 2$	$N * (N - 1) * \dots * 3 = \frac{N!}{(N-(N-2))!} = \frac{N!}{2!}$
\vdots	
2	$N * (N - 1) = \frac{N!}{(N-2)!}$
1	$N = \frac{N!}{(N-1)!}$

Es ergibt sich eine Gesamtsumme von $N! * \sum_{i=0}^{N-1} \frac{1}{i!}$. Aus der Mathematik [20] ist bekannt, dass der Wert der Summe für größer werdendes $N - 1$ sehr schnell gegen die Eulersche Zahl $e (\approx 2,718)$ konvergiert. Es existieren also etwa $e * N!$ verschiedene Suchpfade in diesem Informationsgraphen. Für jeden dieser Pfade kann ein eigener Cluster erstellt werden. Es zeigt sich, dass der Aufwand für die Clusterbildung in diesem Graphen absolut untragbar ist.

Im folgenden werden nun Möglichkeiten vorgestellt, wie der Aufwand des Algorithmus in Grenzen gehalten werden kann. Da eine Aufwandsverminderung bedeutet, dass weniger Cluster erzeugt werden, gilt es darauf zu achten, dass von der Verminderung der Clusterzahl stets die Cluster betroffen sind, die am niedrigsten zu bewerten sind.

Kantengewicht: Eine wichtige Eigenschaft ist das Kantengewicht. Jede Kante besitzt ein Gewicht w mit $0 < w \leq 1$. Ein kleines Gewicht von beispielsweise 0,01 sagt aus, dass lediglich einer von hundert Benutzern die beiden zur Kante gehörenden Knoten direkt aufeinander folgend besucht hat. Um den Algorithmus effizienter zu gestalten, wird die Rekursion entlang von Kanten mit einem zu geringen Gewicht nicht fortgeführt. Hierzu wird ein Grenzwert w_{th} eingeführt.

Definition 16 Sei M_C die Menge aller erzeugten Cluster, dann gilt:

$$\forall C \in M_C : \forall e \in C.E_C : e.w > w_{th}$$

Es bleibt die Frage zu klären, wie w_{th} zu wählen ist. Im folgenden Abschnitt über semantische Distanzen wird dieser Grenzwert in die Definition der Distanzen eingebaut und entscheidet somit sinnvollerweise mit darüber, ob eine Kante als semantisch stark oder schwach gilt. Zur Erinnerung sei erwähnt, dass in Abschnitt 3.2.1 eingeführt wurde, dass eine Kante als semantisch stark gilt, wenn die semantische Distanz des durch die Kante verbundenen Knotenpaares kleiner einem Grenzwert s_{th} ist. Andernfalls ist die Kante semantisch schwach.

Pfadgewicht: Ein ähnliches Entscheidungskriterium für den Clusteraufbau ist es, nicht jedes Kantengewicht einzeln zu betrachten, sondern an jedem Knoten anhand der bislang aufmultiplizierten Kantengewichte zu entscheiden, ob der abgeschriftene Pfad noch wahrscheinlich genug ist. Hierzu wird der Grenzwert w_{pth} des detaillierten Aufwärtsansatzes übernommen, der dort genau diese Funktion erfüllt. Durch diesen Grenzwert wird die Komplexität des detaillierten Aufwärtsansatzes gleichzeitig auf linearen Aufwand in der Anzahl der Knoten $\frac{1}{w_{pth}} * |V|$ beschränkt, wie in Abschnitt 3.3.4 nachzulesen ist. Nach der Beschreibung des Clusteralgorithmus wird untersucht, ob diese Aufwandsbeschränkung auch hier gilt.

4.4.2 Semantische Distanzen

Die semantische Distanz zwischen zwei direkt miteinander verbundenen Knoten ist ein Maß für die Zusammengehörigkeit jener Knoten. Sie wird im Algorithmus benutzt, um an einem Knoten entscheiden zu können, wie mit seinen Nachfolgern verfahren werden soll.

Im vorigen Abschnitt wurde bereits angesprochen, dass das Kantengewicht hier mit einbezogen werden muss. Die semantische Distanz wird im folgenden davon abhängig gemacht, ob die betrachtete Kante den Grenzwert des Kantengewichts w_{th} überschreitet oder nicht. Weiter werden die Inhaltseigenschaften benachbarter Knoten berücksichtigt, um entscheiden zu können, an welchen Stellen Cluster erzeugt werden müssen. Die semantische Distanz wird folgendermaßen definiert:

Definition 17 Sei V die Knotenmenge und E die Kantenmenge des Informationsgraphen. Seien $x, y \in V$, $y \neq z$ und $(x, y) \in E$, dann gilt für die **semantische Distanz** $s(x, y)$:

$$s(x, y) = \begin{cases} 0 & \text{wenn } (x, y).w > w_{th} \wedge (x.type = transit \vee x = g) \\ 0,5 & \text{wenn } (x, y).w > w_{th} \wedge x.type = main \\ 1 & \text{wenn } (x, y).w \leq w_{th} \end{cases}$$

Diese Distanz hat im Algorithmus folgende Bedeutung:

$s(x, y) = 0$ Dies bedeutet, dass das Gewicht der betrachteten Kante bedeutend genug ist. Der Cluster wird mit y fortgesetzt, da x entweder g oder ein Knoten vom Typ *transit* ist.

$s(x, y) = 0,5$ Auch hier ist das Gewicht der verbindenden Kante bedeutend genug. x ist jedoch vom Typ *main* und damit endet bei x ein Cluster C . Da die Kante jedoch das starke Gewicht besitzt, wird bei y ein Cluster weitergeführt.

$s(x, y) = 1$ Das Gewicht der betrachteten Kante ist nicht bedeutend genug. Das wiederum heißt, dass in Richtung y die Traversierung nicht fortgesetzt wird. Der Knotentyp ist hier nicht von Belang.

Es bleibt die Frage offen, wie w_{th} zu wählen ist. In diesem Punkt liegt auch die Variabilität dieser Definition. Folgende Möglichkeiten werden untersucht:

- Wähle w_{th} statisch. Dies ist der einfachste Ansatz, der sich jedoch als ungeeignet erweisen kann. Im Beispielgraphen aus Abb. 4.5 ist ein durchaus zu erwartendes Szenario

dargestellt. Wenn Benutzer sehr viele verschiedene Sitzungsstartseiten wählen, entsteht dadurch eine große Anzahl an schwach gewichteten Kanten von g aus. Wird nun ein statischer Wert zu hoch angesetzt, würde im Extremfall gar kein Cluster aufgebaut werden. Dadurch, dass von g eventuell eindeutig am meisten Kanten ausgehen, wird der Grenzwert nicht auf diese Kanten angewendet ($w_{gth} = 0$). Da die vorhandenen Topologieinformationen über den Rest des Informationsgraphen keine Anhaltspunkte für die Wahl eines guten Wertes liefern, werden die Evaluierungsergebnisse Aufschluss über gute Werte geben.

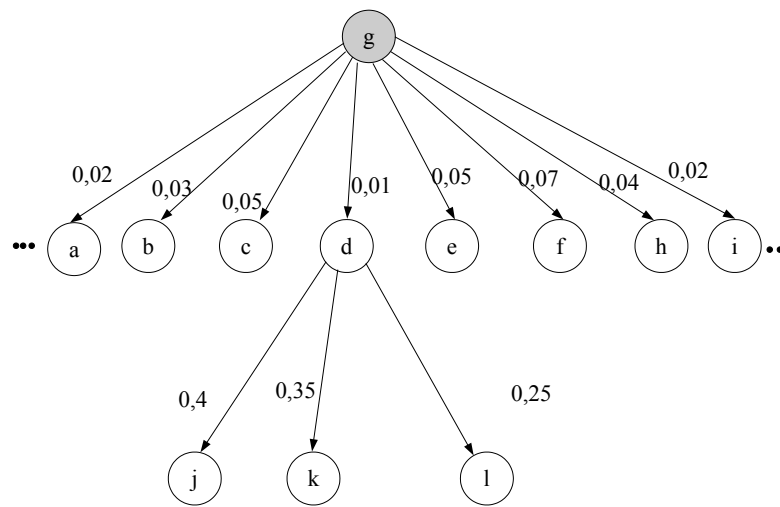


Abbildung 4.5: Wahl von w_{th}

- Wähle w_{th} für jeden Knoten individuell anhand der Verteilung seiner ausgehenden Kantengewichte.

Definition 18 Sei e_{max} die stärkste ausgehende Kante von Knoten x . Dann gilt für jede von x ausgehende Kante e :

$$e.w > w_{th} \iff \frac{e.w}{e_{max}.w} > t \text{ mit } t \in [0, 1]$$

Hier ist t der zu variierende Wert. Auch die Wahl von t wird durch Evaluierungsläufe erfolgen. Es bleibt zu entscheiden, ob dieser dynamische Grenzwert auch auf die Kanten, welche von g ausgehen, angewendet wird. Dies wiederum ist abhängig von der Verteilung der Kantengewichte. Hat eine Kante von g aus ein deutlich höheres Gewicht als die meisten anderen, so werden unter Umständen zu wenige Cluster aufgebaut, da die Traversierung entsprechend schnell beendet ist. Sind die Gewichte hingegen gleichmäßig verteilt, so werden genügend Nachfolger von g besucht. Um die Erstellung einer ausreichenden Zahl an Clustern sicherzustellen, wird w_{th} nicht auf Kanten angewendet, die von

g ausgehen. Bei der Evaluierung wird ein Vergleich zwischen den beiden Möglichkeiten geliefert.

4.4.3 Verwendung einer Warteschlange

Wie im vorigen Abschnitt bereits angeschnitten, kann die alleinige Verwendung von Clustern zum Füllen der Vorabübertragungsliste zur Folge haben, dass zu einem Zeitpunkt keine Cluster mehr zur Verfügung stehen, obwohl die Liste noch nicht gefüllt ist. Dieser Fall kann eintreten, wenn zu wenig Cluster aufgebaut wurden oder die Vorabübertragungsliste sehr groß gewählt wird.

Ebenso ist es möglich, dass die Liste mit Clustern fast vollständig gefüllt wird und kein weiterer Cluster mehr in den verbleibenden Speicherplatz passt. Um diesen Platz jedoch nicht leer zu lassen, kann man eventuell noch einzelne Webseiten in die Liste aufnehmen.

Bei der Traversierung des Graphen zur Erstellung der Cluster wird es vorkommen, dass Knoten besucht werden, die in keinen Cluster aufgenommen werden. Anstatt diese Knoten nun zu ignorieren, bietet es sich an, sie stattdessen einer Warteschlange anzufügen. Sollte die Vorabübertragungsliste dann mit den gegebenen Clustern nicht zu füllen sein oder passt kein Cluster mehr in den Restplatz der Liste, so können einzelne Webseiten nach und nach vom Kopf dieser Warteschlange hinzugefügt werden.

Um bei der Traversierung der Abhängigkeit zwischen Knoten im Graphen gerecht zu werden, die durch die Besuchsreihenfolgen der Benutzer entsteht, ist die verwendete Warteschlange vom Typ FIFO (engl. *first in first out*). Werden zum Beispiel zwei Knoten a und b in die Warteschlange aufgenommen, wobei b im Graphen nur über Pfade erreichbar ist, die über a gehen, so muss a vor b in die Warteschlange aufgenommen werden.

4.4.4 Algorithmus

Es wurde in den vorangehenden Abschnitten sowohl darauf eingegangen, wie der Aufwand des Verfahrens beschränkt werden kann, als auch, wie die semantischen Distanzen für die Erfüllung der Strukturanforderungen der Cluster eingesetzt werden. Weiter ist bekannt, welche Informationen für jeden Cluster gespeichert werden müssen, um ihn anschließend bewerten zu können. Mit diesem Wissen wird Algorithmus 4 erstellt, der auf dem detaillierten Aufwärtsansatz basiert.

Bei der Verwendung der Warteschlange in Alg. 4 fällt auf, dass es eine globale und für jeden Knoten eine lokale Warteschlange gibt. Dies ist notwendig, um die Abhängigkeit zwischen Knoten bezüglich der Besuchsreihenfolge zu berücksichtigen. Wird ein Knoten im Ablauf des Algorithmus besucht, so werden alle seine Nachfolger mit einer semantischen Distanz von 1 in die lokale Warteschlange aufgenommen. Diese Elemente dürfen nicht sofort in die globale Warteschlange eingetragen werden, da sie Nachfolger des betrachteten Knotens sind. Wird für den betrachteten Knoten nach dem Besuch aller seiner Nachfolger festgestellt, dass er ebenfalls in die globale Warteschlange gehört, muss er vor seinen Nachfolgern eingetragen werden. Dieser Fall tritt ein, wenn der Knoten nach dem Besuch aller seiner Nachfolger noch in keinem Cluster enthalten ist und gleichzeitig selbst kein Inhaltsknoten ist. So werden Abhängigkeiten

Algorithmus 4 Clusteralgorithmus

```

1: for all  $n \in N(g)$  do
2:   if  $semDist((g, n)) == 0$  then //siehe Abs. 4.4.2
3:     clusterNachfolger( $n, ()$ , 0,  $(g, n).w$ );
4:   end if
5: end for

function clusterNachfolger(Knoten  $x$ , Knotenliste  $l$ , int  $groesse$ , float
   $pfadgewicht$ ){
6:  $x.markiere()$ ; //zur Zyklenerkennung
7:  $l.f\u00fcgeHinzu(x)$ ;
8:  $lokaleWarteschlange = new$  Warteschlange(); //FIFO Warteschlange
9:  $groesse += x.groesse()$ ;
10: for all  $n \in N(x)$  do //schaue alle Nachfolger an
11:   if  $\neg n.istMarkiert() \wedge n \neq z \wedge pfadgewicht * (x, n).w > w_{pth}$  then
12:     if  $semDist((x, n)) < 1$  then
13:       clusterNachfolger( $n, l, groesse, pfadgewicht * (x, n).w$ );
14:     else // $semDist((x, n)) == 1$ 
15:        $lokaleWarteschlange.f\u00fcgeHinzu(n)$ ;
16:     end if
17:   end if
18: end for
19:  $x.entferne$  Markierung(); //alle Nachfolger besucht
20: if  $x.type == main$  then //Inhaltsknoten
21:    $erstelleCluster(x, l, groesse, pfadgewicht)$ ;
22: end if
23: if  $x$  ist in keinem Cluster then
24:    $globaleWarteschlange.f\u00fcgeHinzu(x)$ ;
25: end if
26:  $globaleWarteschlange.f\u00fcgeHinzu(lokaleWarteschlange)$ ; }

function erstelleCluster(Knoten  $endk$ , Knotenliste  $l$ , int  $groesse$ , float  $pfadgew$ ){
27: Cluster  $c = new$  Cluster( $l$ );
28:  $c.setzeEndknoten(endk)$ ;
29:  $c.setzeGroesse(groesse)$ ;
30:  $c.setzePfadgewicht(pfadgew)$ ;
31:  $f\u00fcgeClusterZuClusterlisteHinzu(c)$ ; }

// $a$  ist der Ausgangsknoten der Kante  $e$ 
function semDist(Kante  $e$ ) return float{
32: Knoten  $startknoten = e.startknoten()$ ; //Knoten, von dem  $e$  ausgeht
33: float  $w_{th} = bestimmeGrenzwert(startknoten)$ ; //siehe Abs. 4.4.2
34: if  $e.w > w_{th} \wedge (startknoten.type == transit \vee startknoten = g)$  then
35:   return 0;
36: else if  $e.w > w_{th} \wedge startknoten.type == main$  then
37:   return 0,5;
38: else
39:   return 1;
40: end if }

```

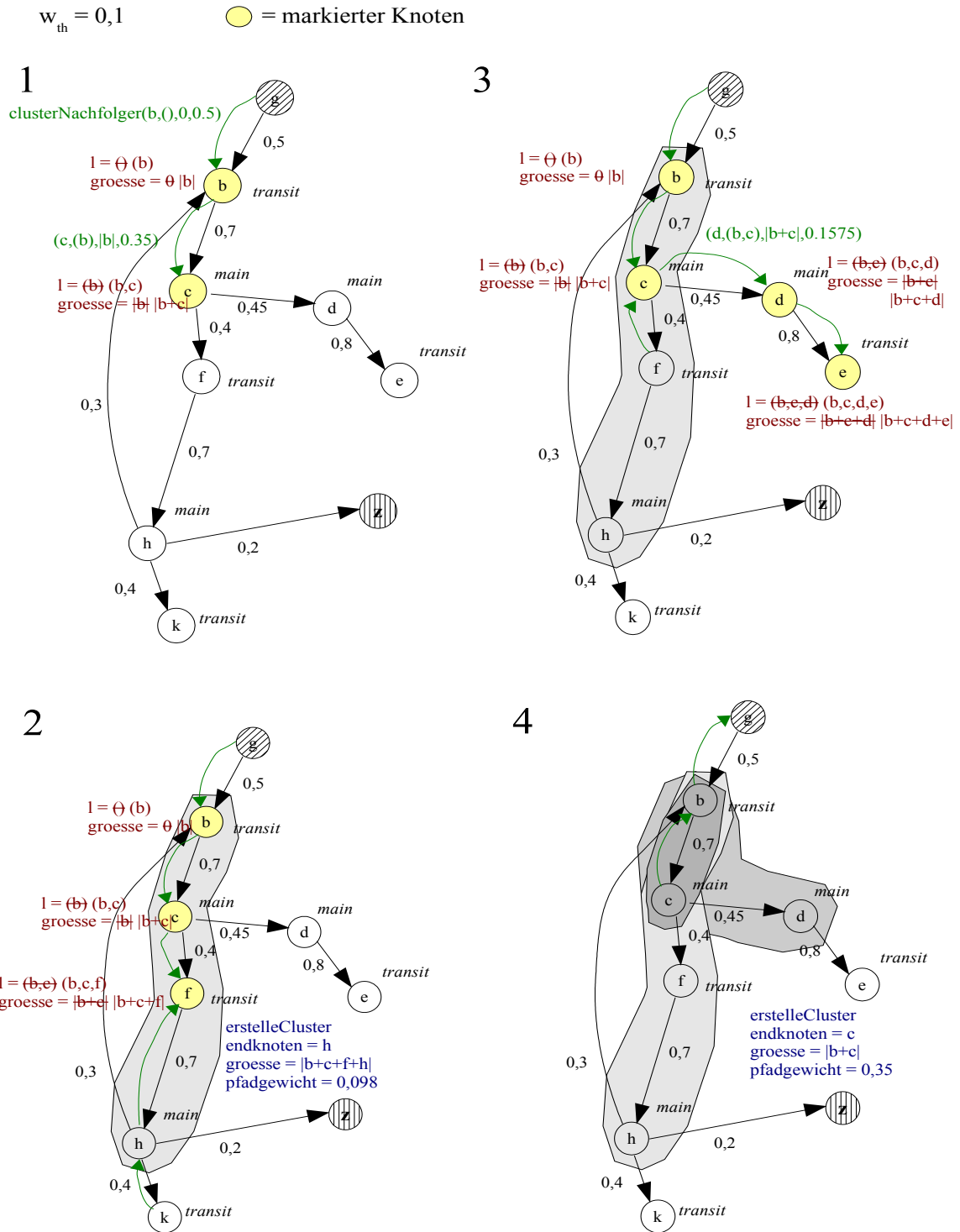


Abbildung 4.6: Ablauf des Clusteralgorithmus

bezüglich der Besuchsreihenfolge zwischen den Knoten berücksichtigt. Es ist zu beachten, dass Knoten mehrfach in diese Warteschlange aufgenommen werden können. Es ist ebenso möglich, dass ein aufgenommener Knoten später im Ablauf des Algorithmus durchaus noch einem Cluster hinzugefügt werden kann. Beim Füllen der Vorabübertragungsliste mit Knoten aus der globalen Warteschlange ist dies zu beachten.

Ein Ablauf des Algorithmus anhand eines Ausschnitts aus einem Beispielgraphen lässt sich in Abb. 4.6 verfolgen. In Teilbild 1 wird der rekursive Abstieg entlang der Knoten b, c dargestellt. Hier ist sichtbar, wie die Knotenliste, das Pfadgewicht und die Summe der Größen propagiert werden. In Teil 2 befindet sich der Algorithmus bereits im Rückgang aus der Rekursion. An Knoten h wird aufgrund der Inhaltseigenschaft ein Cluster mit den in der Abbildung eingetragenen Attributen erstellt. Knoten k wird in die Warteschlange aufgenommen, da er vor dem Rückgang aus der Rekursion nicht Teil eines Clusters wurde. Da c mit d einen weiteren Nachfolger hat, wird die Rekursion in Teil 3 in dieser Richtung fortgesetzt. Auch hier entsteht bei d ein Cluster, wie in Teil 4 zu sehen ist, während e der Warteschlange angefügt wird. Nachdem alle Nachfolger von c besucht wurden, wird für den Inhaltsknoten c selbst ein Cluster erstellt. Danach wird aus der Rekursion bis zu g zurückgegangen und der Algorithmus ist fertig.

Um ein besseres Verständnis zu ermöglichen, wird die Bedeutung einzelner Zeilen in Algorithmus 4 erläutert:

Zeile 1 und 10: Für jeden Knoten x liefert der Ausdruck $N(x)$ die Menge aller direkten Nachfolgeknoten von x .

Zeile 6: Eine Zyklenerkennung ist nötig, um Endlosrekursionen zu vermeiden. Zyklen müssen nicht weiterverfolgt werden, da vom Zyklusnoten aus bereits die Nachfolger besucht werden.

Zeile 11: Die Einschränkung des Aufwands mit Hilfe des Grenzwertes für das Pfadgewicht aus Abschnitt 4.4.1 ist hier enthalten.

Zeile 15: Füllen der lokalen Warteschlange mit Nachfolgeknoten, die aufgrund einer zu hohen semantischen Distanz nicht für eine Clusterbildung mit dem aktuellen Knoten in Frage kommen.

Zeile 23: Füge aktuellen Knoten zur globalen Warteschlange hinzu, wenn bis jetzt kein Cluster aufgebaut wurde, der ihn enthält.

Zeile 28: Ein Cluster muss sich den Inhaltsknoten merken, an dem er erstellt wurde. Dies ist für die Clusterbewertung notwendig (siehe Abschnitt 4.3.1).

Zeile 33: Wie w_{th} gewählt werden kann, ist in Abschnitt 4.4.2 festgelegt. Die verschiedenen Ansätze daraus sind durch die Funktion *bestimmeGrenzwert* zu realisieren.

Aufwand Da die Verwendung von $w_{p_{th}}$ vom detaillierten Aufwärtsansatz aus Abschnitt 3.3.5 übernommen wurde und auch hier keine Zyklen verfolgt werden, ergibt sich hier der identische Aufwand für den schlechtesten Fall von $\frac{1}{w_{p_{th}}} * |V|$ rekursiven Aufrufen. Es wird sich bei der Evaluierung zeigen, inwiefern bei der Wahl dieses Parameters Kompromisse eingegangen werden müssen. Bei der Wahl eines großen Wertes wird zwar die Laufzeit reduziert, jedoch werden eventuell brauchbare Cluster nicht aufgebaut. Umgekehrt verhält es sich bei der Wahl eines kleinen Wertes. Es werden zwar mehr Cluster aufgebaut, gleichzeitig wird aber auch die Laufzeit steigen.

Kapitel 5

Erstellung und Bewertung der Vorabübertragungsliste

In diesem Kapitel werden mögliche Verfahren zur Erstellung der Vorabübertragungsliste vorgestellt. Das Clusterverfahren aus Kapitel 4 liefert eine Menge bewerteter, unsortierter Cluster. Die hier eingeführten Verfahren sollen unter Zuhilfenahme von Sortierungen sicherstellen, dass in jedem Fall nur die höchstbewerteten Cluster in die Vorabübertragungsliste übernommen werden. Nachfolgend wird darauf eingegangen, wie diese erstellte Liste bewertet werden kann.

5.1 Erstellung der Vorabübertragungsliste

Nachdem die Cluster erstellt und bewertet wurden, ist die Vorabübertragungsliste aufzubauen. Zunächst wird begründet, warum dies nicht einfach durch eine einmalige Sortierung der Cluster geschehen kann. Anschließend werden Verfahren vorgestellt, die in jedem Schritt schnell und zuverlässig immer die besten Cluster in die Vorabübertragungsliste einfügen.

5.1.1 Problematik der Clusterüberlappung

Der einfachste Weg die Vorabübertragungsliste aufzubauen wäre es, die Cluster nach vollzogener Bewertung zu sortieren und in dieser Reihenfolge der Liste zuzufügen, bis diese gefüllt ist. Dieser Ansatz ist jedoch nicht sinnvoll. Der Grund liegt darin, dass die Cluster nicht disjunkt sind, wie für das Beispiel in Abb. 4.4 noch angenommen.

Durch die möglichen Überlappungen von Clustern können sich die einzelnen Bewertungen nach dem Steigungsansatz im Laufe des Füllens der Liste verändern. Die Bewertungsänderung eines Clusters C wird genau dann erforderlich, wenn Knoten, die C enthält, der Vorabübertragungsliste hinzugefügt werden, ohne dass C selbst hinzugefügt wird. In die Bewertung von C geht dann nicht seine Gesamtgröße ein, sondern nur die Summe der Größen der Seiten, die noch nicht in die Vorabübertragungsliste aufgenommen wurden. Dies lässt sich anhand des Beispiels aus Abb. 5.1 begründen. Hier ändert sich die Bewertung für Cluster 2, nachdem Cluster 1 in die Liste aufgenommen wurde. Durch das Aufnehmen von Cluster 1 in die Liste

müssen die Knoten a und b bei einer eventuellen Aufnahme von Cluster 2 nicht mehr der Liste hinzugefügt werden. Da somit bei der Größenberechnung von Cluster 2 nur noch c und d zu berücksichtigen sind, sich das innere Gewicht des Clusters aber natürlich nicht ändert, bedeutet dies nach dem Steigungsansatz aus Abschnitt 4.3.2, dass Cluster 2 nach dem Hinzufügen von Cluster 1 höher zu bewerten ist als vorher.

Diese Veränderung der Bewertung von Cluster 2 zieht im Beispiel wiederum eine Veränderung in der Reihenfolge der Cluster nach sich. War ursprünglich Cluster 3 höher als Cluster 2 bewertet, so hat sich diese Reihenfolge nach dem Hinzufügen von Cluster 1 geändert. Das bedeutet, dass also nach jeder Aufnahme eines Clusters in die Liste, sich die Ordnung der restlichen Cluster ändern kann.

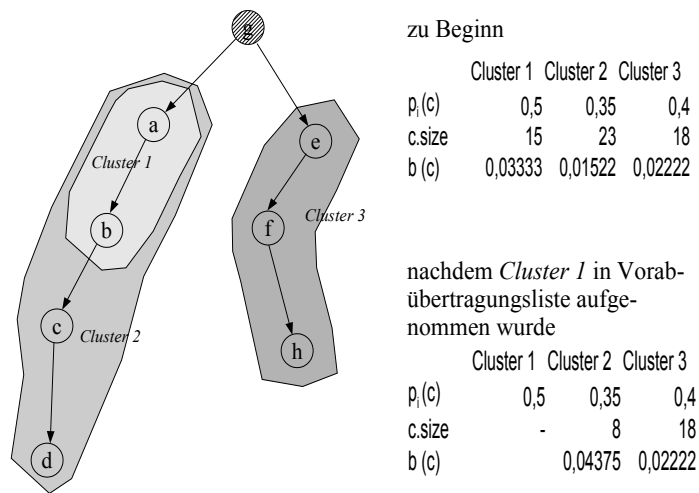


Abbildung 5.1: Beispiel für Bewertungsänderung von Clustern

Formal ausgedrückt bedeutet dies:

Sei M_C die Menge aller Cluster, die nicht in der Vorabübertragungsliste sind.

$\forall C \in M_C$ sei $C.V_C$ die Menge der im Cluster C enthaltenen Knoten.

Sei Cluster C_1 der Cluster mit der höchsten Bewertung, dann:

- Füge C_1 der Vorabübertragungsliste hinzu und entferne Cluster C_1 aus M_C .
- $\forall C \in M_C \implies C.V_C = C.V_C \setminus (C.V_C \cap C_1.V_C)$

Nachdem die Cluster neu bewertet wurden, nimmt man erneut den Cluster mit der höchsten Bewertung und der Ablauf beginnt von vorn. Dies wird wiederholt, bis die Liste entweder gefüllt ist oder alle Cluster aufgenommen wurden.

5.1.2 Optimierung des Aufwands

Es wird nun auf zwei Probleme eingegangen, die sich aus dem vorigen Abschnitt ergeben und die den Aufwand für die Erstellung der Vorabübertragungsliste in die Höhe treiben können.

Auffinden der neu zu bewertenden Cluster

In Abschnitt 5.1.1 wurde ausführlich auf die Notwendigkeit der wiederholten Bewertung von Clustern eingegangen. Es wurde festgestellt, dass nach dem Hinzufügen eines Clusters zur Vorabübertragungsliste eventuell einige Cluster neu bewertet werden müssen. Es stellt sich nun die Frage, wie diese Cluster ausgemacht werden können. Es ist nicht effizient, jedes Mal wenn ein Cluster in die Liste aufgenommen wird, alle anderen Cluster auf eine nicht leere Schnittmenge mit diesem Cluster hin zu überprüfen. Angenommen, es seien n Cluster gegeben. Für den schlechtesten Fall, dass alle Cluster in die Liste passen, hat diese Vorgehensweise quadratischen Aufwand. Das liegt daran, dass bei n Durchläufen jedes Mal alle verbleibenden Cluster auf eine Bewertungsänderung hin überprüft werden müssen. Da die Anzahl der zu überprüfenden Cluster mit jedem Durchlauf um 1 reduziert wird, errechnet sich die Gesamtzahl der Überprüfungen wie folgt:

$$\sum_{i=0}^{n-1} i = \frac{(n-1) * n}{2}$$

Da die Anzahl der Cluster im angewendeten Verfahren durchaus die Anzahl der Knoten im Graphen übersteigen kann, ist solch ein Aufwand untragbar. Zur Lösung des Problems wird der Informationsgraph erweitert. Jeder Knoten merkt sich, zusätzlich zu seinen sonstigen Attributen, welchen Clustern er zugehört. So müssen, wenn die Knoten eines Clusters der Vorabübertragungsliste zugefügt werden, die neu zu bewertenden Cluster nicht erst aufwändig gesucht werden. Bei jedem Knoten, der hinzugefügt wird, können so mit konstantem Aufwand die Cluster bestimmt werden, deren Bewertung sich ändert. Im Beispiel aus Abb. 5.1 merken sich die Knoten a und b ihre Zugehörigkeit zu den Clustern 1 und 2.

Weiter erhält jeder Knoten ein weiteres Attribut, das aussagt, ob er bereits in die Vorabübertragungsliste aufgenommen wurde oder nicht. Somit muss nicht bei jedem Knoten, der in die Liste aufgenommen werden soll, überprüft werden, ob er bereits in der Liste enthalten ist.

Auffinden des höchstbewerteten Clusters

Um eine möglichst gute Vorabübertragungsliste zu erstellen, besteht die Notwendigkeit, dass jedes Mal wenn ein Cluster der Vorabübertragungsliste hinzugefügt werden soll, es sich hierbei um den höchstbewerteten Cluster handelt. Durch die möglichen Bewertungsänderungen muss der nächste höchstbewertete Cluster jedoch nach jedem Hinzufügen eines Clusters zur Liste von neuem gesucht werden.

Wieder angenommen, es gäbe n Cluster. Das Auffinden des besten Clusters in einer unsortierten Menge erfordert linearen Aufwand und damit erhält man in jedem Fall einen quadratischen Gesamtaufwand, wenn alle n Cluster in die Liste aufgenommen werden. Wie jedoch

bereits festgestellt wurde, ist quadratischer Aufwand, aufgrund der möglichen Vielzahl an Clustern, nicht erwünscht.

Ziel der folgend vorgestellten Verfahren muss es also sein, die Vorabübertragungsliste mit geringerem als quadratischem Aufwand zu erzeugen. Dazu wird eine Sortierung der Cluster eingesetzt. Der Vorteil dieser Verfahren wird sein, dass sie für den durchschnittlichen Fall eine komplette Neusortierung der Cluster vermeiden können, nachdem sich Bewertungen geändert haben. Deshalb werden diese Verfahren im Durchschnittsfall einen geringeren Aufwand als den quadratischen besitzen.

5.1.3 Effiziente Verfahren

Wie bereits angedeutet wurde, ist es das Ziel der hier entwickelten Verfahren, die Vorabübertragungsliste möglichst schnell zu erstellen. Eine weitere Anforderung soll sein, stets sicherzustellen, dass jeweils immer der bestmögliche Cluster als nächstes in die Liste aufgenommen wird. Hierzu werden zwei Verfahren beleuchtet, die gute Laufzeiten versprechen.

Bubble Sort-Verfahren

Eine Möglichkeit ist es, die Clusterliste nur einmal zu Beginn in $O(n \cdot \log(n))$ zu sortieren. Anschließend werden dann, nach jedem Hinzufügen eines Clusters zur Vorabübertragungsliste, sämtliche Cluster mit veränderter Bewertung mittels einer Bubble Sort-Variation neu eingeordnet. Eine zuverlässige Aufwandsabschätzung für den Durchschnittsfall kann aufgrund mangelnder Topologieinformationen jedoch nicht erfolgen. Zwar ist bekannt, dass Bubble Sort [26] im schlechtesten Fall Aufwand $O(n^2)$ besitzt, jedoch spricht eine Eigenschaft der Clusterliste dafür, dass die Sortierung wesentlich schneller vollzogen werden kann.

Clusterbewertungen können sich nur vergrößern, da sich die Summe der Größen von Seiten, die nicht in der Liste enthalten sind, nur verkleinern kann. Das bedeutet, dass die "Blasen" nur in eine Richtung aufsteigen werden, was eine deutliche Verbesserung des Aufwands mit sich bringt. Nimmt man nun an, dass m Cluster in die Vorabübertragungsliste passen und sich nach jedem Hinzufügen eines Clusters die Bewertung von k Clustern ändert, die dann durchschnittlich um t Schritte in der Liste aufsteigen. So liegt der Aufwand in $O(m * k * t)$. Hier ergibt sich für den schlechtesten Fall zwar gar ein kubischer Aufwand $O(n^3)$. Für den Durchschnittsfall ist jedoch zu erwarten, dass der Aufwand geringer als quadratisch ausfällt. Angenommen, von den existierenden n Clustern werden alle in die Vorabübertragungsliste übernommen. Nach jedem Hinzufügen ändern $\log(n)$ Cluster ihre Bewertung, worauf sie im Schnitt $\log(n)$ Plätze in der Clusterliste aufsteigen. Dann ergibt sich bereits ein Aufwand, der in $O(n \log^2(n))$ liegt. Der Aufwand für den schlechtesten Fall erhöht sich durch die Sortierung also auf $O(n^3)$. Im Durchschnittsfall wird aber eine Verbesserung des Aufwands erwartet.

In Abb. 5.2 ist der Ablauf des Verfahrens beispielhaft dargestellt. Die Reihenfolge, mit der die markierten Cluster abgearbeitet werden, ist wichtig. Es muss zunächst der geänderte und deshalb markierte Cluster neu einsortiert werden, der vor der Änderung die höchste Bewertung aller markierten Cluster hatte. Im Beispiel kann dies anhand von Cluster 6 und 3 demonstriert werden. Würde man Cluster 6 zuerst in der Liste aufsteigen lassen, so würde er

sich hinter Cluster 3 einordnen. Nach dem Aufsteigen von Cluster 3 ergäbe dies die falsche Reihenfolge 3, 2, 6, . . . , da Cluster 6 nicht erneut einsortiert wird. Deshalb ist es wichtig, dass man die markierten Cluster in der Reihenfolge ihrer alten Bewertung aufsteigen lässt. Im Beispiel würde man also mit Cluster 3 beginnen, wodurch sich nach dem Aufsteigen von Cluster 6 das korrekte Ergebnis 3, 6, 2, . . . ergibt.

Cl. 1	Cl. 2	Cl. 3	Cl. 4	Cl. 5	Cl. 6	Cl. 7
0,7	0,56	0,5	0,4	0,37	0,32	0,23

Füge Cl. 1 der Vorabübertragungsliste hinzu und markiere alle Cluster, deren Bewertung sich geändert hat.

	Cl. 2	Cl. 3 *	Cl. 4	Cl. 5	Cl. 6 *	Cl. 7
	0,56	0,72	0,4	0,37	0,6	0,23

Beginne beim stärksten, markierten Cluster und lasse ihn aufsteigen (danach Markierung aufheben). Dann der zweitstärkste...

	Cl. 2	Cl. 3 *	Cl. 4	Cl. 5	Cl. 6 *	Cl. 7
	0,56	0,72	0,4	0,37	0,6	0,23

Abbildung 5.2: Ablauf des Bubble Sort-Verfahrens

Heap-Verfahren

Da beim Aufbau der Vorabübertragungsliste zu jeder Zeit immer nur der beste Cluster bekannt sein muss, bietet sich die Verwendung eines *Heaps* für die Sortierung der Cluster an. Es wird ein Heap verwendet, wie er in [26] definiert ist. Der Aufwand für den Aufbau des Heaps mit n Elementen, in diesem Fall also n Cluster, liegt laut [26] in $O(n)$. Dazu werden die Aktionen *Löschen*, *Downheap* und *Upheap* für dieses Verfahren benötigt. Beim *Löschen* liefert der Heap das stärkste Element zurück. Die anderen beiden Aktionen lassen ein Element im Heap aufsteigen bzw. absinken, bis die Heap-Eigenschaft wiederhergestellt wurde. Diese Aktionen haben allesamt einen Aufwand in $O(\log(n))$. Es wird wieder angenommen, dass m Cluster in die Vorabübertragungsliste passen und nach jedem Hinzufügen eines Clusters zur Liste, sich die Bewertung für durchschnittlich k Cluster ändert. Für jeden Cluster, der in die Vorabübertragungsliste aufgenommen wird, muss eine Löschoption mit Aufwand $\log(n)$ erfolgen. Danach werden die k neu bewerteten Cluster jeweils ebenfalls mit $\log(n)$ Aufwand neu einsortiert. Damit ergibt sich der Gesamtaufwand zu: $O(m * \log(n) * k * \log(n)) = O(m * k * \log^2(n))$. Für den schlechtesten Fall, dass alle Cluster in die Liste passen und sich in jedem Schritt alle Clusterbewertungen ändern, wird der Aufwand quadratisch. Das lässt sich aber sowieso mit keinem Ansatz umgehen. Auch hier kann aufgrund mangelnder Informationen über die

Topologie des Informationsgraphen leider keine Abschätzung für ein durchschnittliches k getroffen werden. Geht man von der Annahme aus, dass sich in jedem Schritt die Bewertung für etwa $\log(n)$ Cluster ändert, dann hätte man den quadratischen Aufwand umgangen. Ob diese Annahme für den Durchschnittsfall Gültigkeit besitzt, wird bei der Evaluierung überprüft.

Der Aufbau der Liste läuft so ab, dass zu Beginn jeder Runde der Cluster an der Spitze des Heaps in die Liste übernommen wird. Daraufhin wird, wie bei einem Heap üblich, das letzte Element an die Spitze gesetzt und in den Baum, mittels *Downheap*-Operation, neu einsortiert. Die Cluster, die ihre Bewertung geändert haben, steigen per *Upheap*-Operation im Heap auf. Danach kann die nächste Runde beginnen. Dies wird so lange fortgesetzt, bis entweder alle Cluster aufgenommen wurden oder die Vorabübertragungsliste voll ist.

Es ist aus der Praxis bekannt, dass das Heap-Verfahren effizienter arbeitet als Bubble Sort. Im Evaluierungskapitel wird ein Vergleich der Laufzeiten dieser beiden Verfahren geliefert. Neben der Überprüfung, ob das Heap-Verfahren hier im Durchschnittsfall schneller ist, wird auch untersucht, ob der Fall des quadratischen Aufwands eintritt.

5.1.4 Inkrementeller Ansatz

Eine weitere Möglichkeit, die sich mit den beiden vorigen Ansätzen kombinieren lässt, ist es, die Vorabübertragungsliste nicht für jede Anfrage eines Endgeräts erneut aufzubauen. Stattdessen wird nur eine Liste verwendet, die je nach Bedarf nach und nach aufgebaut wird. Das bedeutet, dass es zwei Möglichkeiten gibt, eine Listenanfrage bestimmter Größe zu bearbeiten. Ist die Vorabübertragungsliste bereits mindestens bis zur verlangten Größe aufgebaut, so wird einfach der vordere Teil der Liste mit entsprechender Größe zurückgeliefert. Ist die aufgebaute Liste hingegen noch nicht groß genug, so wird sie erweitert, bis die Listengröße der Anfrage erfüllt werden kann.

In Abb. 5.3 ist an einem Beispiel ein Nachteil dieses Verfahrens aufgezeigt. Im obigen Teil des Bildes ist eine fast gefüllte Vorabübertragungsliste dargestellt (grauer Bereich). Der restliche Platz in der Liste kann nun für die drei verbleibenden, in der Mitte eingezeichneten Cluster optimal genutzt werden. Es ist offensichtlich, dass der höchstbewertete Cluster nicht mehr in die Liste passt. Anstatt einen Cluster nur teilweise in die Liste zu übernehmen, wird dieser Cluster ignoriert und es wird mit dem nächsten Cluster fortgefahren. In diesem Fall passen der zweite und dritte Cluster noch in die Liste und füllen diese beinahe ganz auf. Wenn kein Cluster mehr in die Liste passt, können anschließend noch Knoten aus der globalen Warteschlange entnommen werden.

Im unten dargestellten inkrementellen Ansatz geht dies nicht, weil hier nur eine Liste für alle Anfragen erstellt wird. Es ist zum Zeitpunkt der Listenerstellung nicht bekannt, ob vielleicht später eine Anfrage für eine noch größere Vorabübertragungsliste kommt, in die der erste Cluster passt. Daher kann man hier nicht einfach den verbleibenden Rest der Liste mit anderen Clustern oder gar Knoten aus der Warteschlange füllen. Dies hätte zur Folge, dass die Cluster nicht in der gewünschten Reihenfolge in der Liste enthalten sind, wenn eine Anfrage für eine größere Liste kommt. Den ersten Cluster nur teilweise in die Liste aufzunehmen widerspricht hingegen der Bedingung, dass Cluster entweder komplett oder gar nicht in die Liste aufgenommen werden.

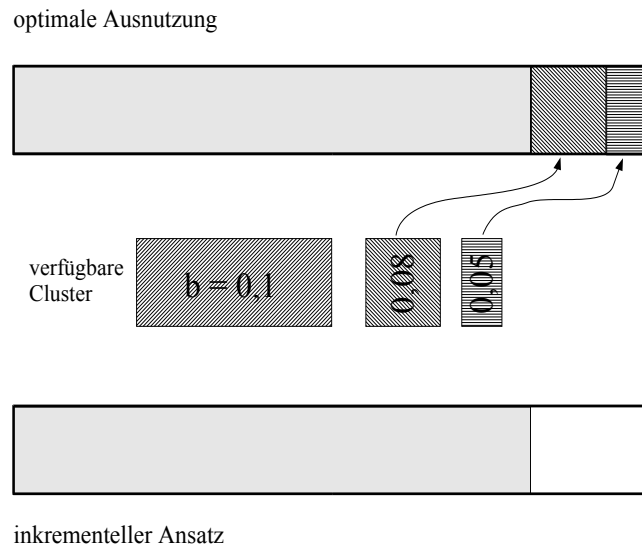


Abbildung 5.3: Inkrementeller Ansatz - Nachteil

Um die Optimierung mit dem zweiten und dritten Cluster dennoch zu ermöglichen, muss man sich die Stelle speichern, ab der die Cluster nicht mehr in der richtigen Reihenfolge hinzugefügt werden. Bei späteren Anfragen für größere Listen müssen die letzten, zur optimalen Platzausnutzung verwendeten Cluster und Knoten zunächst wieder entfernt werden, bevor die Liste weiter aufgebaut werden kann. Bei der Fortsetzung des Listenaufbaus muss dann aber sichergestellt sein, dass die wieder entfernten Cluster und Knoten wieder an richtiger Stelle in der Speicherstruktur der Cluster bzw. Warteschlange eingefügt wurden. Da beim Hinzufügen von Clustern Bewertungsänderungen an anderen Clustern ausgelöst werden können, gestaltet sich das Entfernen von Clustern aus der Liste als nicht einfach, da dann eventuell auch Bewertungsänderungen rückgängig gemacht werden müssen.

Selbst wenn die Optimierung durch die Stellenspeicherung verwendet wird, lässt sich für dieses Verfahren ein weiterer Nachteil demonstrieren. Angenommen, die Vorabübertragungsliste wurde mit dem inkrementellen Ansatz bis zu einer Größe von 1 MB aufgebaut. Es folgt nun aber eine Anfrage für eine nur 500 KB große Liste. Es kann durch das Abschneiden der vorderen Hälfte der 1 MB-Liste passieren, dass sich am Ende dieser 500 KB-Liste nur Teile eines Clusters befinden. Dies verletzt die Bedingung, dass Cluster ganz oder gar nicht in die Liste aufgenommen werden.

Beim inkrementellen Ansatz kann man also nicht generell davon ausgehen, dass die zurückgelieferte Vorabübertragungsliste optimal aufgebaut wurde, so wie es der Fall ist, wenn für jede Anfrage eine eigene Liste gebildet wird. Eine Analyse der Laufzeiten für die Listenerstellung dürfte sich daher als interessant erweisen. Kann die Erstellung einer einzelnen Liste schnell genug erfolgen, besteht keine Notwendigkeit den inkrementellen Ansatz anzuwenden.

5.2 Bewertung der Vorabübertragungsliste

Nachdem nun mit den verschiedenen Ansätzen Vorabübertragungslisten aufgebaut werden können, ist es erwünscht, die Ergebnisse zu bewerten, um sie vergleichen zu können. Analog zur Bewertung der Ergebnisse von Suchmaschinenanfragen, werden in dieser Diplomarbeit zwei Bewertungsarten verwendet. Die *Precision*-Metrik, welche die Bewertung der Vorabübertragungsliste aus Sicht der Infostation vornimmt, wird in diesem Abschnitt festgelegt. Bewertungsarten des *Recall*-Types, welche die Ergebnisse aus Sicht des Benutzers bewerten, sind Teil des folgenden Kapitels.

Für die Bewertung aus Sicht der Infostation gilt, dass die Vorabübertragungsliste umso höher zu bewerten ist, je besser die Liste aus den vorhandenen Clustern aufgebaut wurde. Dazu wird eine Kostenfunktion k abhängig der Vorabübertragungsliste H definiert, deren Wert für Vergleiche dieser Art herangezogen werden kann. Diese Funktion liefert zudem das Optimierungskriterium für sämtliche Ansätze, da es gilt, $k(H)$ zu maximieren.

Ein Ziel bei der Erstellung der Vorabübertragungsliste ist es, mit den enthaltenen Clustern möglichst viele wahrscheinliche Pfade des Informationsgraphen abzudecken. Deshalb muss das innere Gewicht der Cluster aus Definition 14 in die Bewertung mit eingehen.

Weiter ist es erwünscht, dass neben dem Abdecken der Pfade, möglichst viele Inhaltsknoten des Informationsgraphen in die Vorabübertragungsliste aufgenommen werden. Es lässt sich argumentieren, dass dieser Punkt für die Liste irrelevant ist und allein durch das Abdecken der wahrscheinlichsten Pfade das Ziel bereits erreicht ist. Als Gegenargument sind in Abb. 5.4 drei Cluster dargestellt. Cluster 1 und 2 besitzen den Inhaltsknoten d , während h in Cluster 3 den Inhaltsknoten darstellt. Sei nun in der Liste Platz für genau zwei Cluster, wären wegen der höheren inneren Gewichte, Cluster 1 und 2 die beste Wahl. Hier wird jedoch nicht berücksichtigt, wie sich Benutzer verhalten werden, wenn ein Pfad nicht zur Verfügung steht. Findet ein Benutzer eine gesuchte Seite nicht auf einem bestimmten Pfad, so wird davon ausgegangen, dass er es wahrscheinlich auf einem anderen Weg versucht. Von daher ist es erwünschenswert, dass Cluster 1 und 3 in die Liste aufgenommen werden, da diese beiden Cluster beide Inhaltsknoten abdecken, bei größtmöglichem inneren Gewicht. Benutzer, die versuchen den Pfad aus Cluster 2 abzuschreiten, können bei Nichtverfügbarkeit von p immer noch den Pfad aus Cluster 1 abgehen. Um diese Möglichkeit des Ausweichens auf andere Pfade zu berücksichtigen, wirkt die Anzahl der Inhaltsknoten auf die Kostenfunktion mit ein.

Definition 19 Für jede Vorabübertragungsliste H ist die Kostenfunktion k wie folgt definiert:

Sei M_C die Menge aller Cluster und $p_i(M_C)$ die Summe der inneren Gewichte aller Cluster. Sei m_i die Anzahl der Inhaltsseiten in der Vorabübertragungsliste, die zugleich in mindestens einem Cluster enthalten sind.

Sei m_{ges} die Anzahl der Inhaltsseiten, die in mindestens einem Cluster enthalten sind.

$$k(H) = \left(\frac{\sum_{C \in H} p_i(C)}{p_i(M_C)} \right)^\alpha * \left(\frac{m_i}{m_{ges}} \right)^\beta \quad \text{mit } \alpha, \beta \geq 0$$

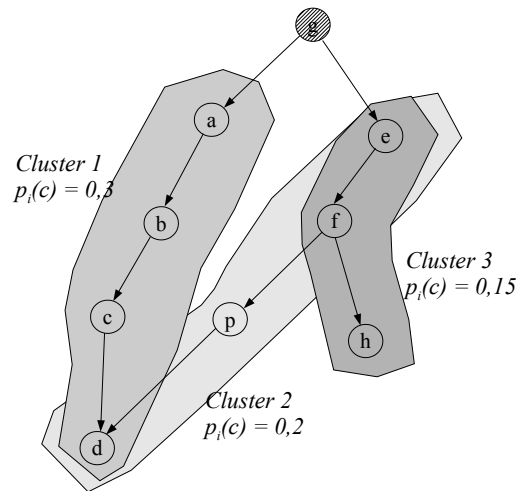


Abbildung 5.4: Berücksichtigung der Anzahl der Inhaltsknoten

Im Optimalfall ergeben sich die Werte beider Brüche jeweils zu 1 und damit $k(H) = 1$. Dann wären sämtliche Cluster in die Vorabübertragungsliste aufgenommen und damit auch alle Inhaltsseiten, die in mindestens einem Cluster enthalten sind, abgedeckt. Die Parameter α und β können zur Gewichtung der Kriterien benutzt werden.

Wie bereits erwähnt, trifft diese Funktion eine Aussage darüber, wie gut die Vorabübertragungsliste aus den gegebenen Clustern aufgebaut wurde. Sie sagt jedoch nichts darüber aus, wie gut die Cluster auf Basis des gegebenen Informationsgraphen aufgebaut wurden. Die Funktion kann also dazu dienen, verschiedene Ansätze zur Clusterbewertung zu vergleichen, wie zum Beispiel den Steigungsansatz aus Definition 15. Sollten verschiedene Ansätze zur Erstellung der Vorabübertragungsliste untersucht werden, die unterschiedliche Ergebnisse liefern, so kann diese Funktion ebenfalls herangezogen werden. Da Heap- und Bubble Sort-Verfahren aus Abschnitt 5.1.3 jedoch das gleiche Ergebnis liefern, wird auch der Wert dieser Funktion identisch ausfallen. Deshalb ist diese Funktion eher für den Fall geeignet, um eventuell in Zukunft erarbeitete Ansätze mit den hier vorgestellten Verfahren zu vergleichen.

Kapitel 6

Evaluierung

In diesem Kapitel wird zuerst kurz auf das verwendete Modell eingegangen, mit dessen Hilfe das Verhalten von Benutzern simuliert werden kann. Anschließend werden Metriken eingeführt, die eine Bewertung von Verfahren zur Erstellung einer Vorabübertragungsliste erlauben. Nachfolgend wird der Aufbau des zur Evaluierung herangezogenen Testgraphen festgehalten. Abschließend werden mit Hilfe der Metriken die Ergebnisse verschiedener Parametereinstellungen verglichen. Weiter erfolgt ein Vergleich des Clusterverfahrens mit den bisher eingesetzten Verfahren sowie eine Interpretation der Ergebnisse.

6.1 Verwendetes Simulationsmodell - User Centric Walk

Für die Erstellung des Informationsgraphen und die Bewertung des Verfahrens werden Logdateien herangezogen, die mit Hilfe des *User Centric Walk* (UCW) aus [2] erstellt wurden. UCW liefert eine Modellierung des Zugriffsverhaltens auf Webseiten von Benutzern inklusive Besuchszeiten. Die verfügbaren Webseiten und Beziehungen untereinander, wie zum Beispiel Links, werden mit einem Graphen modelliert.

Gegen die Verwendung von realen Logdateien spricht, dass die verfügbaren Dateien große Mengen an unnötigen Informationen enthalten und zudem nicht alle Daten liefern, die den Logdefinitionen zufolge benötigt werden. Weiter ist die Verwendung realer Dateien auch aus datenschutzrechtlichen Gründen nicht unproblematisch. Die per UCW erstellten Logdateien hingegen benutzen symbolische Webseiten und können exakt das benötigte Logformat liefern.

Ein weiterer Vorteil der Simulation ist, dass die Besuchsdauer einer Sitzungsendseite bestimmt werden kann, während aus realen Logs diese Information nicht ausgelesen werden kann, wie in Abschnitt 2.2.2 nachzulesen ist.

6.2 Metriken zur Verfahrensbewertung

Wie in Abschnitt 5.2 bereits angekündigt, werden nun Metriken behandelt, die Verfahren zur Vorabübertragung von Webseiten aus Sicht des Benutzers bewerten. Die Bewertung erfolgt unter Verwendung der vom Verfahren gelieferten Vorabübertragungsliste.

Zur Bewertung anhand der Liste werden Mengen von Logdateien herangezogen. Prinzipiell wird die Liste anhand einer Logdatei dann umso höher bewertet, je mehr Webseiten der Logeinträge in der Vorabübertragungsliste aufzufinden sind. Der Anteil an vorhandenen Logeinträgen in der Vorabübertragungsliste wird fortan als *Vollständigkeit* bezeichnet. Wie diese Logdateien genau interpretiert werden sollen und wie das Vorhandensein von Internetseiten einzelner Logeinträge in der Vorabübertragungsliste im Detail zu bewerten ist, wird in den folgenden Abschnitten behandelt.

6.2.1 Interpretation der Logdateien zur Evaluierung

Die Logdateien, die zur Bewertung herangezogen werden sollen, können auf zwei Arten interpretiert werden.

Die erste Möglichkeit besteht darin, den mittels [2] simulierten Benutzer ausschließlich mit der gelieferten Vorabübertragungsliste arbeiten zu lassen. Die daraus entstehenden Logdateien werden im folgenden als *entkoppelt* bezeichnet, da der simulierte Benutzer über keine Netzverbindung verfügt.

Ein weiterer Ansatz besteht darin, dass der simulierte Benutzer Logdateien beim normalen Betrieb mit Netzanbindung aufbaut, ohne Beachtung der Vorabübertragungsliste. Anschließend ist zu vergleichen, welche Seiten auch bei ausschließlicher Verwendung der Vorabübertragungsliste zugreifbar gewesen wären.

Entkoppelter Betrieb

Der Benutzer besitzt keine Netzanbindung und verwendet ausschließlich die Vorabübertragungsliste, die für ihn erstellt wurde. Es soll nun erläutert werden, wie gut sich eine entsprechende Logdatei zur Bewertung eignet.

Aus solch einem Log lässt sich offensichtlich leicht bestimmen, wieviele Seiten der Benutzer gefunden hat. Der Nachteil ist jedoch, dass sich nicht mit Sicherheit sagen lässt, welche und wieviele weitere Seiten der Benutzer eigentlich besuchen wollte.

In Abb. 6.1 ist links ein Beispiel für das Verhalten eines Benutzers im entkoppelten Betrieb angegeben. In diesem Beispiel sind die Seiten E und T nicht verfügbar, weil sie nicht in der Vorabübertragungsliste enthalten sind. Die Schwierigkeit dieses Ansatzes besteht darin, herauszufinden, wieviele Seiten der Benutzer aufgrund des Fehlens von E und T nicht auffinden konnte. Die Logdatei bietet keine Informationen darüber, wie sich das Verhalten des Benutzers verändert, wenn er auf eine nicht vorhandene Seite trifft. Woher will man zum Beispiel wissen, ob der Inhalt, welchen sich der Besucher eventuell von Seite E versprach, nicht vielleicht in W enthalten ist und durch das Gehen über H und K doch gefunden wurde.

Es wird deutlich, dass beim Antreffen einer nicht verfügbaren Seite im Log über die eigentliche Intention und das weitere Vorgehen des Benutzers nur spekuliert werden kann. Dadurch wird dieser Ansatz unbrauchbar.

Seite	Inhalt	in Liste?	Action	Seite	Inhalt	Action
A	transit	JA	jump	A	transit	jump
B	transit	JA	link	B	transit	link
C	transit	JA	link	C	transit	link
D	main	JA	link	D	main	link
E	-	NEIN	jump	E	transit	jump
H	transit	JA	jump	K	transit	link
K	transit	JA	link	W	main	link
W	main	JA	link	T	transit	link
T	-	NEIN	link	J	main	link
Sitzungsende				Sitzungsende		
entkoppelter Betrieb				Betrieb mit Netzanbindung		

Abbildung 6.1: Arten von Logdateien - Beispiel

Betrieb mit Netzanbindung

Der Benutzer durchforstet das Internet zunächst mit einer Netzanbindung, ohne auf die Vorabübertragungsliste zu achten. Nun kann mit Hilfe der Logdatei festgestellt werden, welche Seiten dem Benutzer unter Verwendung der Vorabübertragungsliste zur Verfügung gestanden hätten und welche nicht. In Abb. 6.1 ist rechts das Benutzerverhalten im Beispielfall mit Netzanbindung gegeben.

Der Nachteil dieses Ansatzes besteht darin, dass im Log nicht wiedergegeben wird, wie sich der Benutzer bei Nichtauffindbarkeit einer Seite verhalten hätte. Auf diese Art gebildete Logdateien haben im Vergleich zum vorigen Ansatz jedoch den großen Vorteil, dass neben der Anzahl gesuchter, in der Liste vorhandener Webseiten zusätzlich auch die Anzahl der insgesamt gesuchten Webseiten bekannt ist. Aus diesem Grund wird diese Art von Logdatei bei den folgenden Bewertungsansätzen verwendet.

6.2.2 Herleitung der Metriken

Mit der Festlegung der zu verwendenden Logdateienart werden nun einige Ansätze zur Bewertung des Gesamtverfahrens zur Erstellung der Vorabübertragungsliste beleuchtet. Ausgehend von einer sehr einfachen Metrik werden unter Einbeziehung zusätzlicher Kriterien zur Benutzerzufriedenheit weitere Metriken definiert.

Berücksichtigung aller Logeinträge

Als Ausgangspunkt wird eine sehr einfache Herangehensweise an das Ziel der Vorabübertragung gewählt: Je mehr Seiten einer Logdatei in der Vorabübertragungsliste enthalten sind, desto zufriedener ist der Benutzer. Formal lässt sich diese allgemeine Vollständigkeit der Liste wie folgt definieren:

Definition 20 Sei H die Vorabübertragungsliste, L die zu verwendende Logdatei, $|L|$ die Anzahl an Einträgen und l ein Eintrag in L nach Def. 1. Dann gilt für die **Seiten-Vollständigkeit** $A_{Alle}(L, H)$:

$$A_{Alle}(L, H) = \frac{|\{l | l \in L \wedge l.ID \in H\}|}{|L|}$$

Seite	Inhalt	Action
A	transit	jump
B	transit	link
C	transit	link
D	main	link
E	transit	jump
H	transit	jump
K	transit	link
P	transit	link
W	main	link
F	transit	jump
T	transit	link
F	transit	revisit
J	transit	link
I	main	link

Sitzungsende

 = Seite in Vorabübertragungsliste

Abbildung 6.2: Bewertung mittels Logdatei - Beispiel

Für das Beispiellog aus Abb. 6.2 ergibt sich eine Bewertung $A_{Alle} = \frac{7}{14} = 0.5$.

Dieses Verfahren ist zwar sehr einfach, hat aber den Nachteil, dass die Wichtigkeit sämtlicher Einträge als äquivalent angenommen wird. Analog zur Inhaltseigenschaft von Knoten im Informationsgraphen, haben auch Logeinträge aufgrund dieser Eigenschaft unterschiedliche Bedeutung für den Benutzer.

Bewertung mittels Inhaltsseiten

Mit Hilfe der Inhaltseigenschaft der Logeinträge soll nun der Bewertungsansatz so verbessert werden, dass er den im vorigen Abschnitt erwähnten Nachteil nicht mehr enthält. Damit die Metrik weiter einfach gehalten wird, erfolgt die Berechnung ähnlich zum vorigen Ansatz. Es wird nun davon ausgegangen, dass der Benutzer umso zufriedener ist, desto mehr Logeinträge mit Inhaltseigenschaft *main* in der Vorabübertragungsliste enthalten sind.

Definition 21 Sei wieder H die Vorabübertragungsliste, L die zu verwendende Logdatei und l ein Eintrag in L . Dann gilt für die **Inhaltsseiten-Vollständigkeit** $A_{Inhalt}(L, h)$:

$$A_{Inhalt}(L, H) = \frac{|\{l | l \in L \wedge l.ID \in H \wedge l.type = main\}|}{|\{l | l \in L \wedge l.type = main\}|}$$

Für das Beispiel aus Abb. 6.2 resultiert dies mit $A_{Inhalt} = \frac{3}{3} = 1$ in einer deutlichen Veränderung der Bewertung.

Es ergibt sich jedoch auch für diesen Ansatz ein Nachteil, der sich sehr schön am Beispiellog veranschaulichen lässt. Eintrag D ist hier in der Vorabübertragungsliste vorhanden. Es ist aber erkennbar, dass D durch eine Verfolgung von Verlinkungen über die Seiten A, B, C , die allesamt nicht in der Liste zu finden sind, besucht wurde. Es kann nun nicht einfach angenommen werden, dass der Benutzer die Seite D ohne Zuhilfenahme der vorhergehenden Seiten gefunden hätte. Analog zur Semantik der Cluster, lassen sich auch in Logdateien Suchpfade wie A, B, C, D im Beispiel ausmachen. Für eine bessere Bewertung muss also neben der Inhaltseigenschaft eines Eintrags auch berücksichtigt werden, wie dieser Eintrag überhaupt erreicht wurde. Dies ist Inhalt des folgenden Ansatzes.

Bewertung anhand von Logsuchpfaden

Um eine Bewertung anhand von *Logsuchpfaden* zu ermöglichen, muss zunächst definiert werden, wie ein Suchpfad in einer Logdatei zu erkennen ist.

Suchpfade in Logdateien Die Folge von Einträgen einer Logdatei lässt sich in einer Baumstruktur darstellen. Abb. 6.3 zeigt den Baum für den hinteren Teil des Beispiellogs aus Abb. 6.2. Wie auch bei der Suchpfaderkennung im Informationsgraphen, gilt eine Folge

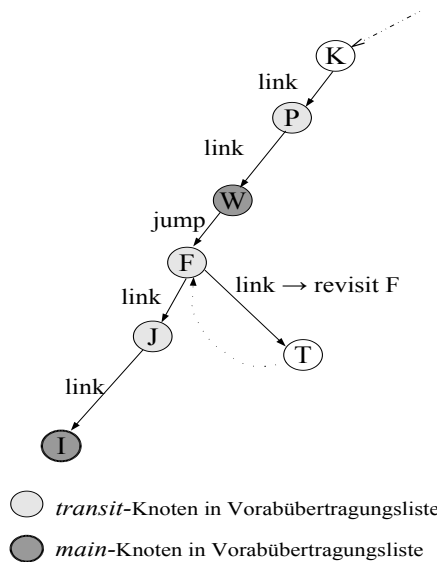


Abbildung 6.3: Logsuchpfad - Beispiel

von Navigationsseiten, gefolgt von mindestens einer Inhaltsseite als Logsuchpfad. Im Beispiel beginnt der Benutzer, nach dem Besuch von Inhaltsknoten W , bei F eine neue Suche, die in I ihren Abschluss findet. Von F aus wird zwar zunächst T besucht. Der Benutzer geht aber sofort wieder zu F zurück, um über J zu I zu finden.

In diesem Beispiel zeichnet sich eine weitere Eigenschaft ab, die für einen Logsuchpfad erfüllt sein muss. Die Seiten entlang des Logsuchpfades im Baum müssen alle miteinander verlinkt sein. Da ein *jump*, wie hier zwischen *W* und *F*, bedeutet, dass *F*, auch ohne über *W* zu gehen, gefunden worden wäre, ist diese Bedingung sinnvoll. Für die Definition eines Logsuchpfades ergibt sich somit folgendes:

Definition 22 Ein **Logsuchpfad** S_L ist ein Pfad von Logeinträgen $(l_1 \dots l_n)$ im die Logdatei repräsentierenden Baum, für den die folgenden Bedingungen gelten:

1. $l_1.action = jump$
2. $l_i.action = link$ ($2 \leq i \leq n$)
3. l_n ist ein Inhaltsknoten

Im Beispielbaum bilden also F, J, I einen Logsuchpfad, wohingegen F, T keinen bilden, da T keine Inhaltsseite ist. Würde auf I eine weitere Inhaltsseite X folgen, die per *link* mit I verbunden ist, dann würde auch F, J, I, X einen Logsuchpfad bilden. Ein Logsuchpfad muss jedoch nicht notwendigerweise Navigationsseiten enthalten. Würde eben erwähntes X per *jump* von I aus angefordert, dann bildet X als Inhaltsseite einen eigenen, einelementigen Logsuchpfad.

Dieses Beispiel zeigt ebenfalls, wie es gilt, mit *revisit*-Einträgen umzugehen. T gehört durch das Zurückgehen zu F nicht zum Pfad F, J, I , was sich auch sehr schön in der Baumstruktur zeigt.

Bewertung Mit der Definition des Logsuchpfades kann nun die Bewertung mittels dieser Pfade definiert werden.

Definition 23 Sei wieder H die Vorabübertragungsliste, L die zu verwendende Logdatei und l ein Eintrag in L . Weiter sei $S_L(l)$ der zu l gehörende Logsuchpfad. Dann gilt für die **Suchpfad-Vollständigkeit** $A_{Pfad}(L, H)$:

$$A_{Pfad} = \frac{|\{l \mid l \in L \wedge l.type = main \wedge \forall i \in S_L(l) : i.ID \in H\}|}{|\{l \mid l \in L \wedge l.type = main\}|}$$

Wie angesprochen, wird in dieser Definition verlangt, dass ein Inhaltsknoten nur dann als in der Vorabübertragungsliste vorhanden gilt, wenn sein kompletter Logsuchpfad ebenfalls enthalten ist. Für das Beispiellog ergibt sich $A_{Pfad} = \frac{1}{3}$, da K, P, W und A, B, C, D nicht komplett in der Vorabübertragungsliste enthalten sind.

Im Vergleich zur Bewertung aus Def. 21, in der die Logsuchpfade überhaupt keine Rolle spielen, gilt hier das andere Extrem. Das Problem auch bei diesem Ansatz ist es, dass er ein zu erwartendes Benutzerverhalten vernachlässigt. Keiner der Ansätze berücksichtigt, dass der Benutzer eventuell über eine andere Suchfolge von Seiten zum Ziel findet, wenn der Logsuchpfad zu einem gesuchten Inhalt nicht in der Vorabübertragungsliste vorhanden ist. Während Def. 21 aussagt, dass der Benutzer in jedem Fall zum Inhaltsknoten findet, wenn dieser in der Vorabübertragungsliste ist, so geht Def. 23 davon aus, dass der Inhalt nur dann

gefunden wird, wenn genau der gewünschte Pfad in der Liste enthalten ist. Da von beiden Aussagen nicht unbedingt ausgegangen werden kann, liegt die tatsächliche Vollständigkeit irgendwo dazwischen.

Da jedoch keine stichhaltige Aussage darüber getroffen werden kann, wie wahrscheinlich ein Benutzer zu einem Inhaltsknoten findet, wenn nur Teile des Logsuchpfades in der Liste enthalten sind, kann keine bessere Bewertung erfolgen. Dieses Problem ergibt sich als Folge des in Abschnitt 6.2.1 erwähnten Nachteils dieser Logdateienart.

Zusammenfassung

Alle Metriken zeichnen sich dadurch aus, dass 1 den optimalen Wert für die Bewertung bildet, während 0 den schlechtesten Wert darstellt. Ebenso zeigt sich am Beispiellog, wie unterschiedlich die einzelnen Ergebnisse ausfallen können. Als Bewertung für Verfahren zur Erstellung von Vorabübertragungslisten geht Def. 20 zu vereinfachend von der Gleichbedeutung aller Einträge aus. Die beiden darauf folgenden Metriken liefern hier Werte, welche die Zufriedenheit der Benutzer besser wiedergeben. Das Verhalten von Benutzern, das durch die Logdateien nicht aufgezeigt werden kann, wird aber auch hier nicht berücksichtigt. Damit gemeint ist das Benutzerverhalten beim Antreffen von nicht verfügbaren Seiten.

6.3 Evaluierungsaufbau

Mit der Festlegung der Bewertungsmetriken kann nun die Evaluierung des Clusterverfahrens in Angriff genommen werden. In diesem Abschnitt wird beschrieben, aus welchen Logdateien der Informationsgraph aufgebaut wird und mit wie vielen Logdateien die Bewertung vorgenommen wird. Vorweg sei erwähnt, dass eine Vielzahl an Evaluierungsläufen mit verschiedenen Informationsgraphen und Logdateien durchgeführt wurde. Die Ergebnisse ähnelten sich jedoch für alle Läufe so erheblich, dass hier lediglich die Ergebnisse eines ausgewählten Laufes repräsentativ für alle vorgestellt werden.

Allgemeines	
maximale Webseitengröße	2 MB
Anzahl Webseiten insgesamt	10000
Informationsgraph	
Anzahl Logdateien für Graphaufbau	15000
Anzahl Knoten	9589
Anzahl Sitzungsstartknoten	5629
Anzahl Kanten	47014
Gesamtgröße aller enthaltenen Webseiten	174,93 MB
Bewertung	
Anzahl Logdateien	5000

Tabelle 6.1: Evaluierungslauf - Eigenschaften

In den per UCW erstellten Logdateien wurden insgesamt 10000 verschiedene symbolische Webseiten referenziert. Der Informationsgraph kann also nicht mehr als 10002 Knoten besitzen (g , z und ein Knoten pro Webseite). Der Maximalwert für die Webseitengröße wurde auf 2 MB festgelegt. Für den Graphaufbau und die Bewertung wurden insgesamt 20000 Logdateien erstellt. Davon dienen 15000 dem Aufbau und 5000 der Verfahrensbewertung. In Tab. 6.1 sind die grundlegenden Eigenschaften aufgezählt, auf die der dargestellte Evaluierungslauf aufbaut. Unter anderem lässt sich ablesen, dass, um den gesamten Graphen vorab übertragen zu können, ca. 175 MB an Speicherplatz auf dem Endgerät benötigt werden. Der Graph eignet sich also dazu, um feststellen zu können, wie gut die verschiedenen Verfahren die Webseiten auswählen, um einen Speicher mit dem Bruchteil dieser Größe zu füllen.

Zur Evaluierung wurde ein Rechner mit 3 GHz-Prozessor und 1 GB Arbeitsspeicher verwendet.

6.4 Auswertung

Nachdem die Struktur des Versuchsgraphen vorgestellt wurde, erfolgt in diesem Abschnitt die Darstellung der Ergebnisse des repräsentativen Evaluierungslaufs. Zunächst werden die verschiedenen Parametereinstellungen für das Clusterverfahren untersucht. Beim Test verschiedener Werte für einen bestimmten Parameter waren die Werte der anderen Parameter jeweils auf die besten Werte gesetzt. Nachdem daraus die beste Parameterkombination gewonnen wurde, kann ein Vergleich mit dem bisherigen Tiefen- und Breitensuchverfahren erfolgen.

Für sämtliche Vergleiche gilt, dass diese für verschiedene Größen von Vorabübertragungslisten durchgeführt wurden. Da sich auf mobilen Endgeräten die Größe des verfügbaren Speicherplatzes erheblich unterscheiden kann, ist es beispielsweise nicht erwünscht, eine Parametereinstellung als die beste zu erklären, die nur für eine gewisse Größe der Vorabübertragungsliste das beste Ergebnis liefert. Deshalb wird ein breites Spektrum an Listengrößen untersucht und das Verfahren ausgewählt, das über jenes Spektrum hinweg gesehen die besten Ergebnisse liefert. Da einerseits mobile Endgeräte häufig sparsam mit ihrem wenigen verfügbaren Speicher umgehen müssen und andererseits die Zeit begrenzt ist, die für die letztendliche Vorabübertragung zur Verfügung steht, sind die Ergebnisse für kleinere Listen hier von vorrangiger Bedeutung. Es wird daher besonders auf die Ergebnisse für Listen im Größenbereich von 50 KB bis 20 MB geachtet.

Die Bewertungsachsen in den folgenden Abbildungen sind mit der jeweils angewendeten Art der Vollständigkeit beschriftet, die in Prozent angegeben wird. Ein Suchpfad-Vollständigkeit von 10 % bedeutet zum Beispiel, dass die Benutzer im Durchschnitt 10 % ihrer Logsuchpfade mit Hilfe der entsprechenden Vorabübertragungsliste abschreiten konnten. Analog gilt dies für die Seiten- und die Inhaltsseiten-Vollständigkeit.

6.4.1 Bestimmung der besten Parameterwerte

Nun folgend werden für die verschiedenen zu variierenden Parameter des Clusterverfahrens die besten Werte ermittelt. Vorweg ist zu erwähnen, dass die Parameter ebenfalls auf Abhängigkeiten untereinander hin untersucht wurden, jedoch keine festgestellt werden konnten.

Laufzeitvergleich: Bubble Sort- und Heap-Verfahren

In Abschnitt 5.1.3 über effiziente Verfahren zur Erstellung der Vorabübertragungsliste wurden diese beiden Methoden vorgestellt. Da dort ebenfalls festgehalten wurde, dass sich diese Verfahren nicht im gelieferten Ergebnis unterscheiden, wird hier lediglich ein Vergleich der Laufzeiten dargestellt.

Es wird untersucht, wie sich die Laufzeit, abhängig von der Größe der zu erstellenden Liste, verhält. Ebenso werden die Ergebnisse mit verschiedenen großen zugrundeliegenden Clustermengen verglichen.

Ergebnis In Abb 6.4 wird wie erwartet klar ersichtlich, dass Vorabübertragungslisten jeglicher Länge deutlich schneller mit dem Heap-Verfahren aufgebaut werden können. Es wurden hier Listen verschiedener Größen jedes Mal aus derselben Clustermenge aufgebaut.

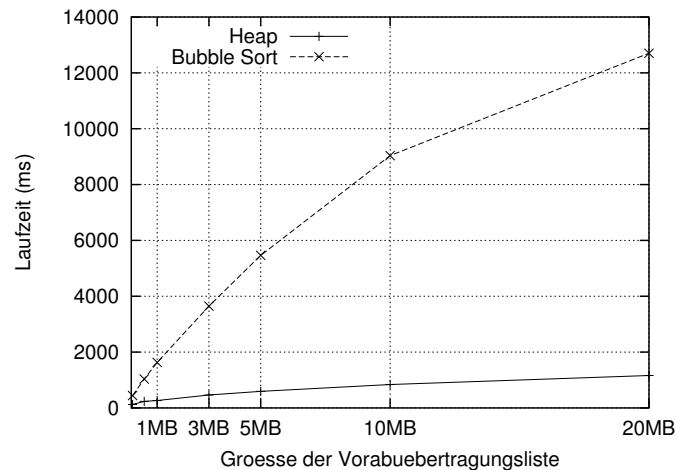


Abbildung 6.4: Laufzeitvergleich: Heap- und Bubble Sort-Verfahren

Ein Laufzeitvergleich des Heap-Verfahrens für verschieden große Clustermengen ist in Tabelle 6.2 wiedergegeben. Hier lässt sich durch Vergleich der Laufzeiten der Zeilen 1, 3 und 5 erkennen, dass eine Verdopplung der Mengengröße bei weitem nicht zu einer quadrierten Laufzeit führt, wie es im schlechtesten Fall möglich wäre.

Größe der Clustermenge	gemittelte Laufzeit (in Millisekunden)
843	8,79
1196	14,57
1714	25,15
1897	36,74
3464	96,32

Tabelle 6.2: Laufzeit des Heap-Verfahrens für verschiedene Clustermengen

Grenzwert der Inhaltswahrscheinlichkeit

Im folgenden wird der Parameter untersucht, der den Grenzwert für die Inhaltswahrscheinlichkeit $p_{main_{th}}$ aus Definition 10 bestimmt. Liegt die Inhaltswahrscheinlichkeit eines Knotens im Informationsgraphen über dem Grenzwert, so wird die entsprechende Webseite als Inhaltsseite, ansonsten als Navigationsseite angesehen.

Je kleiner $p_{main_{th}}$ gewählt wird, desto mehr Seiten werden als Inhaltsseiten betrachtet und umso mehr Cluster werden aufgebaut. Wenn mehr Cluster aufgebaut werden, kann die Wahl der besten Cluster für die Vorabübertragungsliste aus einer größeren Clustermenge getroffen werden. Die resultierende Liste sollte also bei kleiner werdendem Grenzwert nicht schlechter, sondern höchstens besser werden können. Von daher ist zu erwarten, dass ein Grenzwert von 0 das beste Ergebnis liefert.

Da bei kleiner werdendem Grenzwert jedoch mehr Cluster erstellt werden, ist zu erwarten, dass sich die Laufzeit des Verfahrens erhöht. Nicht nur werden mehr Cluster erzeugt, auch muss die größere Anzahl an Clustern für die Listenerstellung im Heap wiederholt sortiert werden. Die Aufwandserhöhung ist also wesentlich davon abhängig, wie schnell die Liste aus den Clustern erstellt werden kann, was wiederum von der Schnelligkeit des Heap-Verfahrens abhängt.

Ergebnis Bei der Verwendung von 0 wird für jede Seite, die mindestens einmal als Inhaltsseite angesehen wurde, ein Cluster erstellt. Das bedeutet, dass also auch sehr viele Seiten, die in den allermeisten Fällen als Navigationsseiten benutzt werden, eigene Cluster erhalten. In Tabelle 6.3 wird deutlich, dass vor allem für 0,5 deutlich weniger Cluster aufgebaut werden.

Grenzwert für Inhaltswahrscheinlichkeit	Anteil Inhaltsseiten im Graph	Anzahl Cluster
0.0	80,95 %	23497
0.1	80,60 %	23407
0.2	75,61 %	22139
0.3	67,01 %	19416
0.5	34,47 %	9166

Tabelle 6.3: Anteil an Inhaltsseiten

In Abb. 6.5 sind die Bewertungsergebnisse für diesen Parameter dargestellt. Es zeigt sich, dass die Werte von 0 bis 0,3 bei der Inhaltsseiten-Vollständigkeit (Grafik rechts oben) kaum nennenswerte Unterschiede aufweisen. Lediglich 0,5 fällt für größere Listen etwas ab. Die Ähnlichkeit der Ergebnisse deutet darauf hin, dass die Inhaltsseiten der meisten Cluster, welche in die Vorabübertragungsliste übernommen werden, eine Inhaltswahrscheinlichkeit größer als 50 % besitzen. Da etwa ein Drittel aller Knoten im Graphen eine Inhaltswahrscheinlichkeit von über 50 % besitzt und diese somit, von der Gesamtgröße des Graphen ausgehend, eine Gesamtgröße von etwa 60 MB haben dürften, gehen in diesem Bereich der Listengrößen auch noch nicht die Cluster aus.

Interessant wird es, wenn man die Seiten- und Inhaltsseiten-Vollständigkeit gemeinsam be-

trachtet. Vor allem für 0, 5 ergeben sich erheblich schlechtere Werte bei der Seiten-Vollständigkeit. Der große Unterschied muss sich aus der unterschiedlichen Auswahl der Navigationsseiten ergeben, da die Inhaltsseiten-Vollständigkeit ja fast identisch für alle Parameterwerte ausfällt. Das bedeutet, dass vor allem für 0, 5 eine andere Auswahl der Cluster getroffen wird, als bei den übrigen Werten. Es werden zwar für alle Parameterwerte in etwa die gleichen Inhaltsseiten in die Liste aufgenommen, jedoch unterscheiden sich die Suchpfade zu diesen Seiten.

Umso überraschender präsentiert sich dadurch das Ergebnis der Suchpfad-Vollständigkeit. Obwohl unterschiedliche Cluster ausgewählt werden, so liefern doch alle Parameterwerte ähnliche Ergebnisse. Der Wert 0, 5 fällt hier, wie auch bei der Inhaltsseiten-Vollständigkeit, nur wenig ab. In Kombination mit der Seiten-Vollständigkeit gewinnt man dadurch eine Erkenntnis. Die deutlich höheren Werte der Seiten-Vollständigkeit bei 0 bis 0, 3 bedeuten in Kombination mit den fast identischen Kurven für die Suchpfad-Vollständigkeit, dass Benutzer für diese kleineren Parameterwerte häufiger nur Teile von Logsuchpfaden in der Vorabübertragungsliste vorfinden.

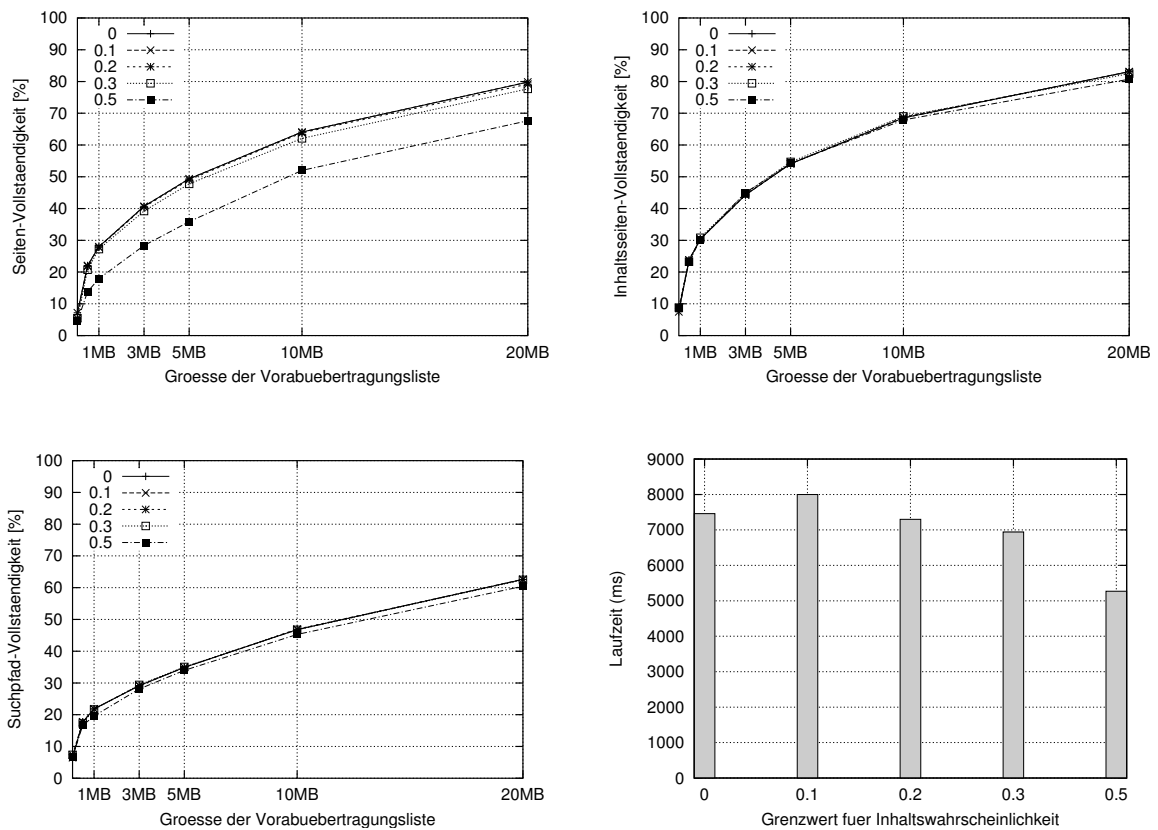


Abbildung 6.5: Ergebnisse: Inhaltswahrscheinlichkeit

Die Werte der Laufzeitmessung werden durch die Mittelung der einzelnen Laufzeiten für die verschiedenen Listengrößen ermittelt. Die Laufzeit wird vom Start des Clusterverfahrens bis zum Zurückliefern der fertigen Liste gemessen. Die Messung, die aufgrund der Maschinen-

auslastung immer gewissen Schwankungen unterworfen ist, zeigt kaum größere Unterschiede mit Ausnahme von 0,5. Durch die weitaus weniger erstellten Cluster lässt sich das Verfahren also etwas beschleunigen, jedoch wird dies durch etwas schwächere Ergebnisse erkauft. Die Einsparung der Laufzeit erfolgt zu einem Großteil dadurch, dass weniger Cluster erstellt werden. Da weniger Cluster sortiert werden müssen, werden auch die Listen schneller erstellt. Die Listen für den Wert 0,5 benötigen für ihre Erstellung im Schnitt, im Vergleich zum Wert 0, etwa $\frac{1}{3}$ der Zeit. Die absoluten Werte fallen aber in beiden Fällen niedrig aus. Zum Beispiel dauert die Erstellung der 5 MB-Liste ca. 200 ms bzw. 600 ms. Abhängig von der Anzahl der Listenanfragen muss entschieden werden, wie lang die Erstellung der Liste dauern darf. Steht genügend Zeit zur Verfügung, so sollte 0 als Parameterwert verwendet werden.

Nutzen der Inhaltswahrscheinlichkeit Mit der Erkenntnis, dass der Parameter für 0 die besten Werte liefert, stellt sich die Frage, ob die Unterscheidung nach Inhalts- und Navigationsseiten überhaupt sinnvoll ist. Immerhin werden mehr als 80 % der Seiten mindestens einmal als Inhaltsseiten eingestuft. In Abb. 6.6 ist der Vergleich der Ergebnisse mit und ohne Berücksichtigung der Inhaltseigenschaft dargestellt. Das zum Vergleich herangezogene Verfahren ignoriert die Inhaltswahrscheinlichkeit und erstellt bei jedem angetroffenen Knoten einen Cluster. Das bedeutet, dass für jeden Knoten und jeden Pfad, der zu diesem Knoten abgeschrieben wird, jeweils ein Cluster erzeugt wird. Die Anzahl der rekursiven Aufrufe ist also gleich der Clusteranzahl.

Wie der Abbildung entnommen werden kann, bringt die Berücksichtigung der Inhaltswahrscheinlichkeit eine erkennbare Ergebnisverbesserung mit sich. Von daher ist die Berücksichtigung dieser Knoteneigenschaft gerechtfertigt.

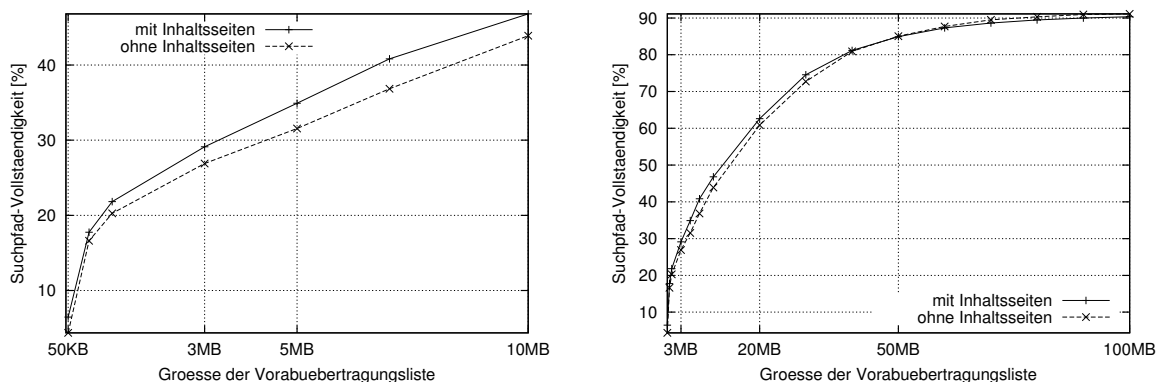


Abbildung 6.6: Ignorieren der Inhaltswahrscheinlichkeit

Minimales Kantengewicht - Statische Semantische Distanz

Wie bei der Definition der statischen semantischen Distanz in Abschnitt 4.4.2 festgelegt, wird anhand eines Grenzwerts für Kantengewichte entschieden, ob die Traversierung des Graphen

entlang einer Kante fortgesetzt wird. Ist das Kantengewicht zu gering, wird die semantische Distanz gleich 1 und der Clusteralgorithmus setzt entlang dieser Kante nicht fort.

Dieser Parameter ist primär dazu gedacht, den Aufwand des Algorithmus zu beschränken. Wird der Grenzwert für das minimale Kantengewicht sehr groß gewählt, so wird der Algorithmus zwar sehr schnell fertig sein, jedoch eventuell auch nur sehr wenige Cluster aufbauen. Bei der Wahl von 0 wird jede Kante weiterverfolgt, was dazu führt, dass die durch diesen Parameter beeinflussbare, maximale Anzahl an Clustern aufgebaut wird. Wie beim vorigen Parameter also auch, wird erwartet, dass das Ergebnis sich bei Verkleinerung des Grenzwertes nur verbessern kann, da die zur Auswahl stehende Clustermenge sich mit kleinerem Grenzwert vergrößert. Da sich damit aber gleichzeitig die Laufzeit erhöht, muss darauf geachtet werden, ob hier eventuell ein Kompromiss eingegangen werden muss. Die Erhöhung der Laufzeit ergibt sich hier nicht nur aus den beiden Faktoren, die noch bei der Inhaltswahrscheinlichkeit galten. Neben der größeren Menge an zu erstellenden und sortierenden Clustern steigt hier ebenfalls die Menge an rekursiven Abstiegen.

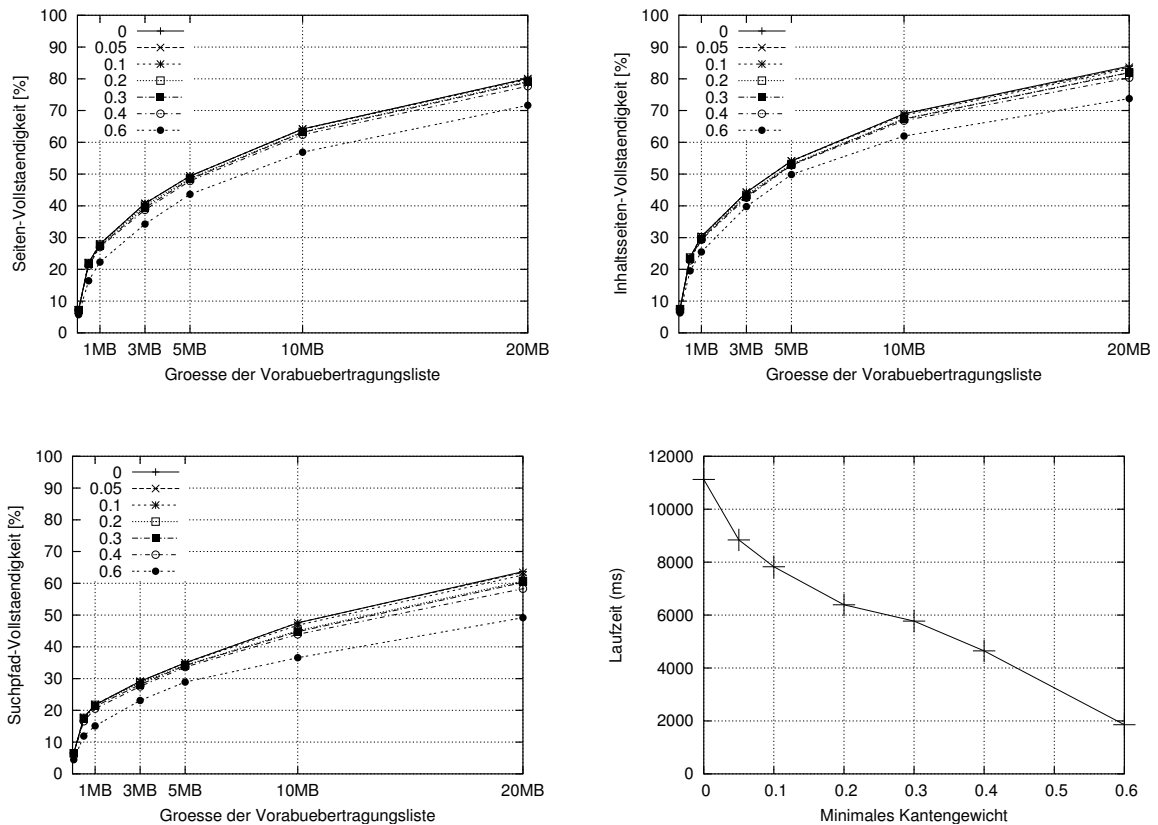


Abbildung 6.7: Ergebnisse: Minimales Kantengewicht bei statischer semantischer Distanz

Ergebnis In Abb. 6.7 sind die wenig überraschenden Ergebnisse für diesen Parameter dargestellt. Es zeigt sich, dass bei kleiner werdendem minimalem Kantengewicht die Ergebnisse

immer besser werden. Was in der Grafik schlecht zu sehen ist, aus der zugehörigen Wertetabelle aber gut zu entnehmen war, ist, dass sich die Ergebnisse für die Werte von 0 bis 0,1 so gut wie nicht unterscheiden. Erst bei 0,2 lassen sich bei der Suchpfad- und Inhaltsseiten-Vollständigkeit Verluste im Bereich von einem Prozent beobachten. In Verbindung mit der Laufzeitmessung eignet sich 0,1 als bester Wert, da er im Vergleich zu den kleineren Werten so gut wie die gleichen Ergebnisse liefert und dabei doch merklich schneller ist.

Dass die Laufzeitunterschiede hier so deutlich ausfallen, wurde aufgrund der angesprochenen Faktoren ebenfalls erwartet. Die Entwicklung dieser Werte ist Tabelle 6.4 zu entnehmen.

Minimales Kantengewicht	Rekursive Aufrufe	Anzahl Cluster
0.0	33160	31264
0.05	27727	26024
0.1	25091	23497
0.2	21394	19945
0.3	19501	18144
0.4	15842	14595
0.6	7918	6771

Tabelle 6.4: Minimales Kantengewicht - Laufzeitfaktoren (Statische sem. Distanz)

Minimales Kantengewicht - Dynamische Semantische Distanz

In Folge von Definition 18 wurde die Frage aufgeworfen, ob der dynamische Grenzwert auch auf die von g ausgehenden Kanten angewendet werden soll. In sämtlichen getesteten Graphen ergaben sich sehr schlechte Ergebnisse bei der Verwendung des Grenzwerts für diese Kanten. Da meist nur ein geringer Anteil der massenhaft vorhandenen g -Kanten vom Algorithmus berücksichtigt wurde, war dieser sehr schnell fertig und es wurde eine zu kleine Menge an Clustern aufgebaut. Dies bestätigt die Festlegung, dass die Bedingung des minimalen Kantengewichts, wie bei der statischen Variante auch, nicht für von g ausgehende Kanten gilt.

Der Einsatzzweck dieses Parameters ist gleich dem, des minimalen Kantengewichts für die statische semantische Distanz. Deswegen wird auch hier, neben der Verbesserung des Ergebnisses durch kleinere Grenzwerte, auf die Laufzeiterhöhung geachtet.

Ergebnis Die Ergebnisse dieser Parameterevaluierung können Abb. 6.8 entnommen werden. Die Inhaltsseiten-Vollständigkeits zeigen, bis auf für den Wert 1, nur gering unterschiedliche Ergebnisse. Bei der Suchpfad-Vollständigkeit lassen sich ein wenig deutlichere Unterschiede ausmachen. Die Parameterwerte 0 und 0,1 liefern hier die besten Ergebnisse, die fast gleichauf liegen. Vor allem bei den größeren Listen werden die Unterschiede zu den größeren Parameterwerten ab 0,3 deutlicher.

Die Laufzeiten, die in Abb. 6.9 grafisch dargestellt sind, verhalten sich ebenso wie erwartet. Auffällig ist, dass selbst für sehr hohe Grenzwerte wie 0,8 immer noch sehr viele

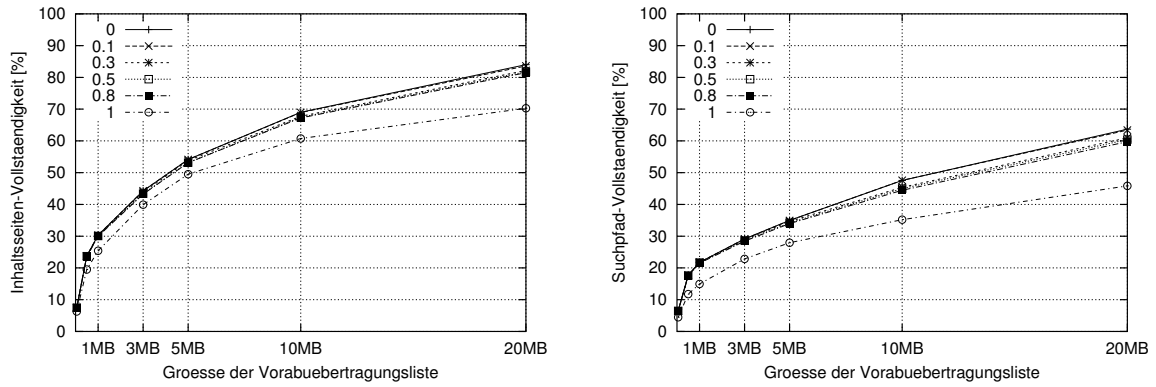


Abbildung 6.8: Ergebnisse: Minimales Kantengewicht bei dynamischer semantischer Distanz

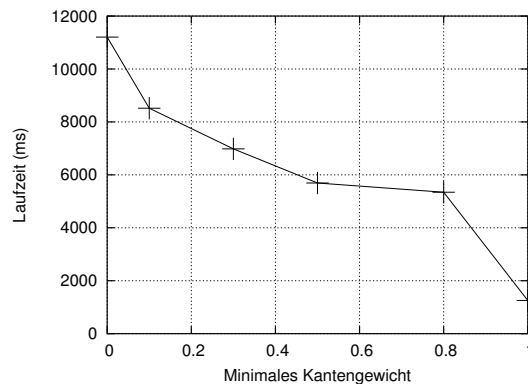


Abbildung 6.9: Laufzeitvergleich: Minimales Kantengewicht (Dynamische sem. Distanz)

Minimales Kantengewicht	Rekursive Aufrufe	Anzahl Cluster
0.0	33160	31264
0.1	27589	25890
0.3	23457	21915
0.5	20055	18688
0.8	18776	17430
1.0	5629	4588

Tabelle 6.5: Minimales Kantengewicht - Laufzeitfaktoren (Dynamische sem. Distanz)

rekursive Abstiege erfolgen, wie Tabelle 6.5 entnommen werden kann. Dies spricht dafür, dass die Gewichte ausgehender Kanten eines Knotens ziemlich gleichmäßig verteilt sind.

Durch den Laufzeitvorteil gegenüber 0 ergibt sich 0,1 als der beste und damit als Standard auszuwählende Wert.

Vergleich: Statische und Dynamische Semantische Distanz

Nachdem jeweils die besten Werte für den Grenzwert des minimalen Kantengewichts für beide Arten von Distanzen ermittelt wurden, werden beide Ansätze miteinander verglichen. Auf Basis der Bewertungen und Laufzeiten wird dann der bessere Ansatz ausgewählt.

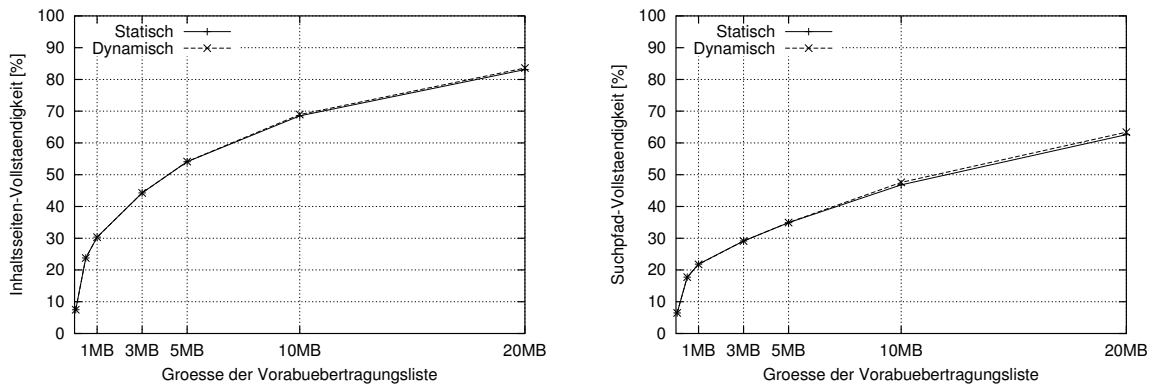


Abbildung 6.10: Bewertungsvergleich: Statische und Dynamische semantische Distanz

Ergebnis Die Ergebnisse der Bewertungen für beide Arten der semantischen Distanz, die jeweils 0,1 als besten Wert für das minimale Kantengewicht benutzen, sind in Abb. 6.10 dargestellt. Für beide Bewertungsarten lässt sich eine große Übereinstimmung feststellen. Unterschiede zwischen diesen beiden Ansätzen zeichnen sich zumeist erst in der zweiten Stelle hinter dem Komma ab. Für die kleineren Vorabübertragungslisten zwischen 1 MB und 5 MB ist die statische Variante leicht besser. Bei den größeren Listen hingegen ist die dynamische Variante ganz leicht besser. Da die statische Variante für die gewählten Parameterwerte etwas schneller arbeitet, wie Abb. 6.11 entnommen werden kann, wurde dieses Verfahren für alle anderen Evaluierungsläufe gewählt. Es wurden auch Tests anderer Parameter unter Verwendung der dynamischen Distanz unternommen, um eventuell doch auftauchende Unterschiede zwischen den Ergebnissen der Distanzen feststellen zu können. Man gewann jedoch auch dort lediglich die hier geschilderte Erkenntnis, dass diese beiden Ansätze nahezu identische Resultate liefern.

Die große Ähnlichkeit der Ergebnisse lässt sich zu einem Großteil dadurch begründen, dass beide Distanzen den Grenzwert nicht auf Kanten von g aus anwenden. Da der Informationsgraph aber sehr viele Sitzungsstartknoten besitzt, laufen viele Schritte der beiden Verfahren identisch ab. Die erhofften Verbesserungen, die durch das dynamische Anpassen

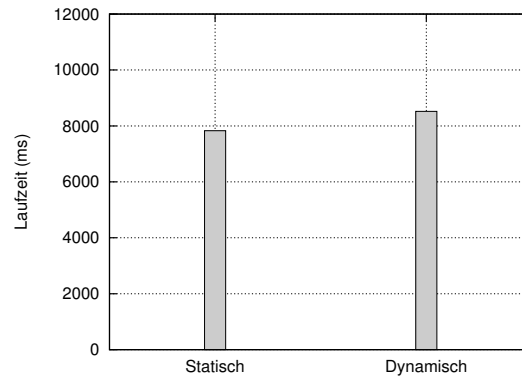


Abbildung 6.11: Laufzeitvergleich: Statische und Dynamische semantische Distanz

des minimalen Kantengewichts je Knoten erzielt werden sollten, sind also nicht eingetreten.

Minimales Pfadgewicht

Wie in Abschnitt 4.4.1 erläutert, dient das minimale Pfadgewicht ebenfalls dazu, den Aufwand des Verfahrens zu verringern. Dieser Parameter legt den Grenzwert des Pfadgewichts fest, unterhalb dessen von g aus abgeschrittene Pfade nicht weiter verfolgt werden. Kommt der Algorithmus also über einen bestimmten Pfad zu einem Knoten, so werden nur die Kanten weiter abgeschritten, die das Gesamtpfadgewicht nicht unter den Grenzwert drücken.

Von diesem Parameter wird der deutlichste Einfluss auf die Laufzeit erwartet, da er den konstanten Faktor in der Aufwandsabschätzung darstellt. Je größer dieser Wert, der sinnvollerweise in $[0..1]$ liegen muss, gewählt wird, desto schneller ist der Algorithmus fertig. Jedoch werden dann eventuell zu wenige Cluster aufgebaut. Je kleiner der Wert gewählt wird, umso mehr Cluster werden aufgebaut, mit dem Nebeneffekt, dass sich die Laufzeit erhöht. Bei der Festlegung dieses Wertes ist besonders auf den Kompromiss zwischen Qualität der Clustermenge und Laufzeit zu achten.

Ergebnis In Abb. 6.12 ist das Bewertungsergebnis der Evaluierung dieses Parameters zu finden. Wie zu erwarten, ergeben sich Ergebnisverbesserungen durch die Verringerung des Grenzwertes. Bei sehr kleinen Listengrößen sind sämtliche Werte noch gleichauf, da die Cluster mit den besten Pfadgewichten in allen Fällen aufgebaut werden und dieses Gewicht eine wichtige Rolle bei der Bewertung der Cluster spielt. Je größer dann aber die Listen werden, desto weiter auseinander bewegen sich die Kurven, da bei größeren Grenzwerten nach und nach die eigentlich auszuwählenden Cluster nicht mehr aufgebaut werden. Dieses Verhalten lässt sich für alle Werte beobachten, außer für den Bereich von 10^{-7} bis 10^{-5} . Hier unterscheiden sich die Ergebnisse kaum. Es zeigt sich jedoch, dass für Listengrößen, die hier nicht dargestellt werden, auch diese Werte auseinander laufen. Jedoch öffnet sich die Schere der Kurven hier nur sehr wenig und merkbliche Unterschiede ergeben sich erst für Listen nahe der 150 MB-Grenze.

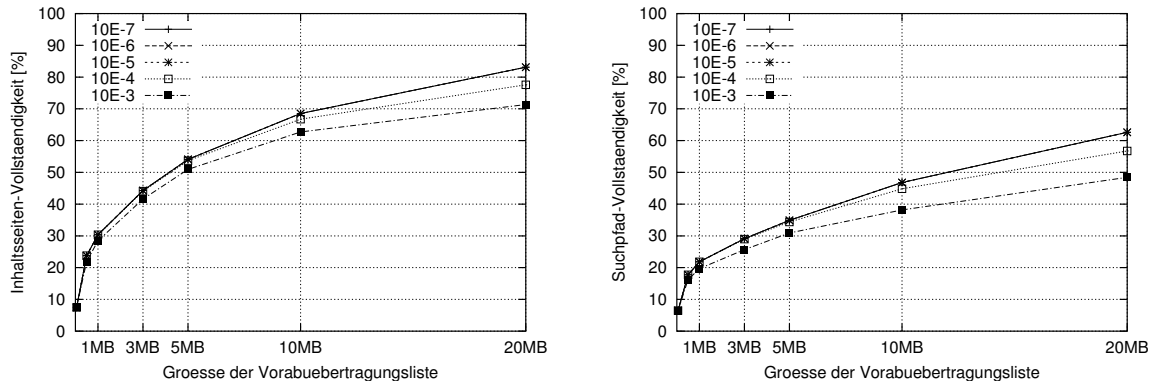


Abbildung 6.12: Bewertungen: Minimales Pfadgewicht

Bei der Analyse des Laufzeitenschaubilds in Abb. 6.13 muss beachtet werden, dass die Achsen logarithmisch skaliert sind. Während beim kleinsten getesteten Wert der Algorithmus fast 50 Sekunden läuft, braucht er bei 10^{-5} lediglich knapp 8 Sekunden, wobei sich das Ergebnis nicht wesentlich unterscheidet. Wie erwartet, hat dieser Parameter also einen erheblichen Einfluss auf die Laufzeit des Algorithmus. Dies wird auch durch die Darstellung der Laufzeitfaktoren in Tabelle 6.6 deutlich.

Im Verhältnis vom Ergebnis zur Laufzeit stellte sich der Wert 10^{-5} bei sämtlichen getesteten Graphen als der beste heraus. Sollte eine noch schnellere Traversierung des Graphen gewünscht werden, kann der Parameterwert erhöht werden. Die gelieferten Ergebnisse werden dann jedoch mit zunehmender Größe der Vorabübertragungslisten schlechter werden.

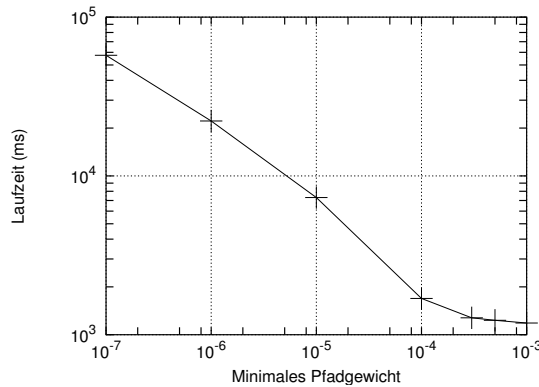


Abbildung 6.13: Laufzeitverhalten: Minimales Pfadgewicht

Minimales Pfadgewicht	Rekursive Aufrufe	Anzahl Cluster
10^{-7}	136960	133415
10^{-6}	63880	61570
10^{-5}	25091	23497
10^{-4}	7249	6196
10^{-3}	5686	4645

Tabelle 6.6: Minimales Pfadgewicht - Laufzeitfaktoren

Gewichtung der Faktoren der Clusterbewertung

Bei der Clusterbewertung nach dem Steigungsansatz spielt das innere Gewicht des Clusters eine entscheidende Rolle. Dieser Wert wiederum setzt sich aus dem Pfadgewicht des Clusters und der Inhaltswahrscheinlichkeit des letzten Inhaltsknotens zusammen. In Definition 14 wird die Gewichtung der beiden Werte zueinander mit Hilfe von α und β ermöglicht. Hierbei steht α für die Gewichtung des Pfadgewichts, während mit β die Inhaltswahrscheinlichkeit gewichtet wird. Wie die optimale Gewichtung dieser beiden Werte zueinander aussieht, wird folgend aufgezeigt.

Es sei ergänzend erwähnt, dass noch andere als die hier dargestellten Kombinationen von α und β untersucht wurden. Da diese jedoch keine nennenswert anderen Ergebnisse lieferten, wurde aus Gründen der Übersichtlichkeit auf deren graphische Berücksichtigung verzichtet.

Ist einer der beiden Parameterwerte 0, so bedeutet dies, dass der entsprechende Faktor im inneren Gewicht nicht berücksichtigt wird. Wie sich das Ignorieren einzelner Faktoren auswirkt dürfte interessant zu beobachten sein. Erhofft wird jedoch, damit Definition 14 gerechtfertigt wird, dass die Berücksichtigung beider Faktoren zugleich eine Verbesserung des letztendlichen Ergebnisses bedeutet.

Da es sich hierbei lediglich um verschiedene Arten der Clusterbewertung handelt, ist eine Analyse der Laufzeiten nicht notwendig. Die Clustermenge für alle hier vorgenommenen Evaluierungsläufe ist identisch. Unterschiede in den Ergebnissen entspringen der unterschiedlichen Reihenfolge, in welcher die Cluster in die Vorabübertragungsliste eingefügt werden. Es wird erwartet, dass sich die beste Bewertung und damit Sortierung der Cluster in den Ergebnissen für kleinere Listen zeigt. Wenn die Listen sehr groß werden und einen Großteil der Cluster enthalten, sollten sich die verschiedenen Kurven dann wieder enger aneinander fügen, bis sie schließlich, beim Enthalten aller Cluster, alle denselben Wert liefern. Um dieses Zusammenlaufen der Kurven beobachten und überprüfen zu können, wird der Maximalwert der Listengrößen auf 100 MB gesetzt.

Ergebnis Die Ergebnisse der Inhaltsseiten-Vollständigkeit in Abb. 6.14 heben die Bedeutung von β am deutlichsten heraus. Dies ist einleuchtend, da hier ja anhand der Knoteneigenschaft bewertet wird, die eben durch β gewichtet wird. Dennoch zeigen die großen Abweichungen für $\alpha = 0$, dass der Pfad zum Inhaltsknoten nicht ignoriert werden darf. Die besten Ergebnisse ergeben sich, wenn entweder beide Werte auf 1 gesetzt sind oder wenn α auf 0,75

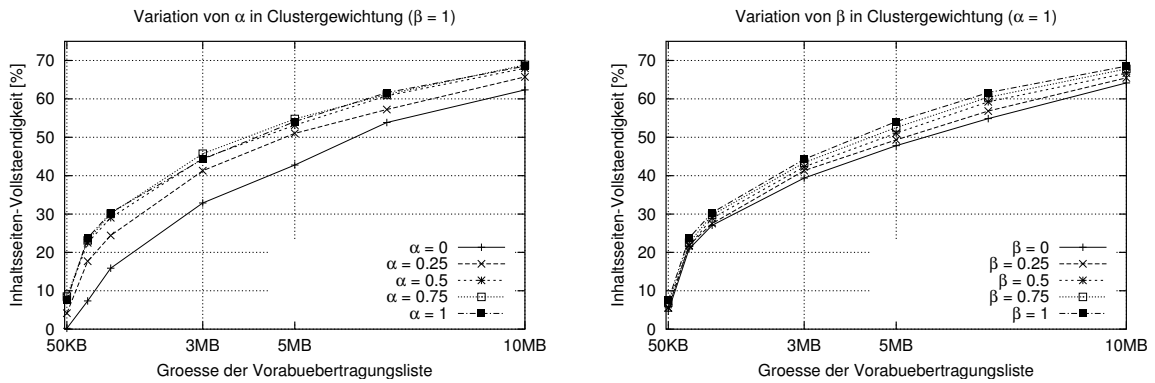


Abbildung 6.14: Inhaltsseiten-Vollständigkeit für variierende α und β

abgewertet wird.

Die Suchpfad-Vollständigkeit in Abb. 6.15, bei der sowohl Pfadgewicht als auch Inhaltsseigenschaft einbezogen werden, zeigt die größeren Abweichungen bei der Variation von α . Die Variation der Gewichtung der Wahrscheinlichkeit, mit welcher der Benutzer den Pfad zur Inhaltsseite abschreitet, beeinflusst das Ergebnis deutlich stärker als die Gewichtungsänderung der Inhaltswahrscheinlichkeit. Das Ignorieren von α führt hier im Bereich kleiner Listen gar zu einer Kurve mit wachsender Steigung, was bedeutet, dass schlechtere Cluster vor eigentlich besseren ausgewählt werden.

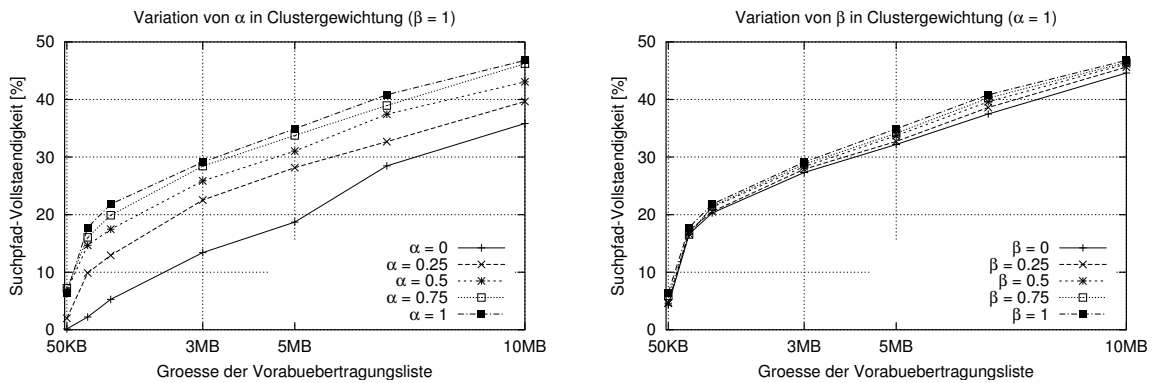


Abbildung 6.15: Suchpfad-Vollständigkeit für variierende α und β

Es ergibt sich bei der Analyse der Wertetabelle, dass die besten Werte mit Ausnahme der 50 KB-Liste stets für $\alpha = \beta = 1$ erzielt werden. So ergibt sich, trotz unterschiedlicher Abweichungen beim Verändern der Parameter, dass eine Gleichgewichtung der beiden Faktoren für die Clusterbewertung das Optimum darstellt.

In Abb. 6.16 wird anhand der Suchpfad-Vollständigkeit verglichen, wie das Ergebnis aus-

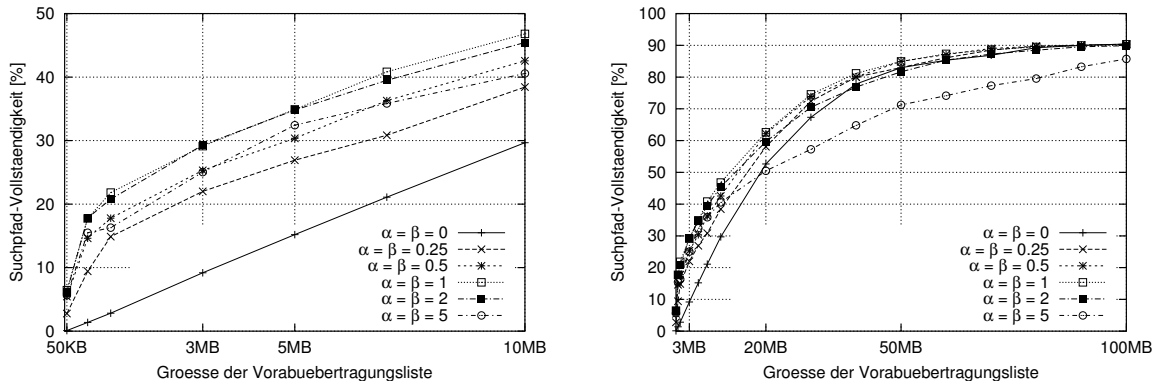


Abbildung 6.16: Indirekte Gewichtung von innerem Gewicht zur Clustergröße

sieht, wenn die absoluten Werte von α und β variiert werden. In Abschnitt 4.3.2 ist nachzulesen, dass diese beiden Parameter indirekt das innere Gewicht zur Clustergröße gewichten. Je größer die beiden Werte sind, desto stärker wird das innere Gewicht bewertet. Sind beide Werte 0, so wird nur die Clustergröße zur Bewertung herangezogen. Es wurde bereits festgestellt, dass die Wahl eines identischen Wertes für α und β die besten Ergebnisse liefert. Wie dieser Wert allerdings zu wählen ist, ist noch nicht bekannt. Abb. 6.16 kann entnommen werden, dass für den Wert 1 die besten Ergebnisse geliefert werden. Je weiter der Parameterwert von 1 entfernt ist, desto schlechter werden die Ergebnisse des Clusterverfahrens. Die schlechtesten Ergebnisse für kleine Listen werden erzielt, wenn das innere Gewicht ignoriert wird. Für große Listen wird das Ergebnis schlechter, wenn der Wert sehr groß gewählt wird und damit die Clustergröße nicht genug berücksichtigt wird.

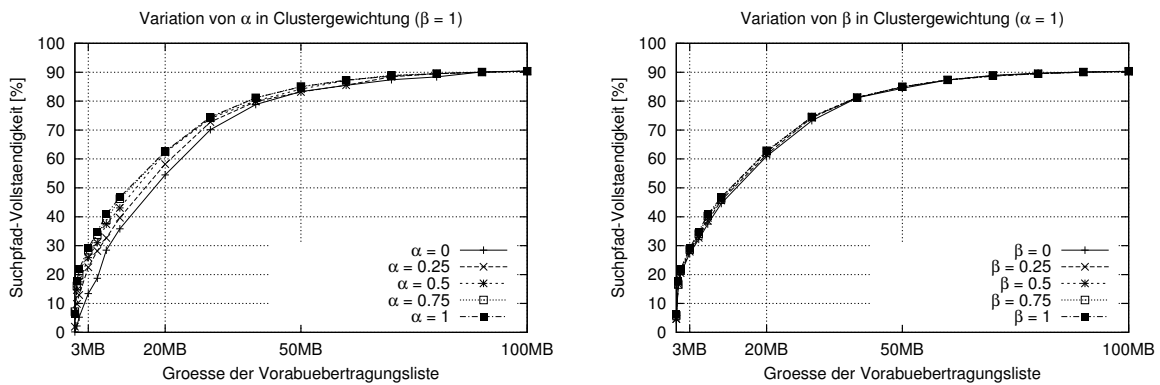


Abbildung 6.17: Verhalten bei großen Vorabübertragungslisten

Aus den gewonnenen Erkenntnissen lässt sich folgern, dass der Algorithmus auf Änderungen an der Gewichtung des Pfadgewichts deutlich empfindlicher reagiert als auf Änderungen

an β . Die besten Werte liefern Inhaltsseiten- und Suchpfad-Vollständigkeit bei Gleichgewichtung der Parameter ($\alpha = \beta = 1$).

Das erwartete Zusammenlaufen der verschiedenen Kurven lässt sich in Abb. 6.17 gut verfolgen.

6.4.2 Vergleich mit bisherigem Verfahren

Nachdem nun die besten Parameterwerte ermittelt sind, wird in diesem Abschnitt untersucht, inwiefern das Clusterverfahren eine Verbesserung zum existierenden Tiefen- und Breitensuchverfahren darstellt. Hierzu wird zunächst kurz bestimmt, welche Parametereinstellungen mit dem existierenden Verfahren die besten Werte liefern. Danach kann der direkte Vergleich erfolgen.

Ergebnisse des bisherigen Verfahrens

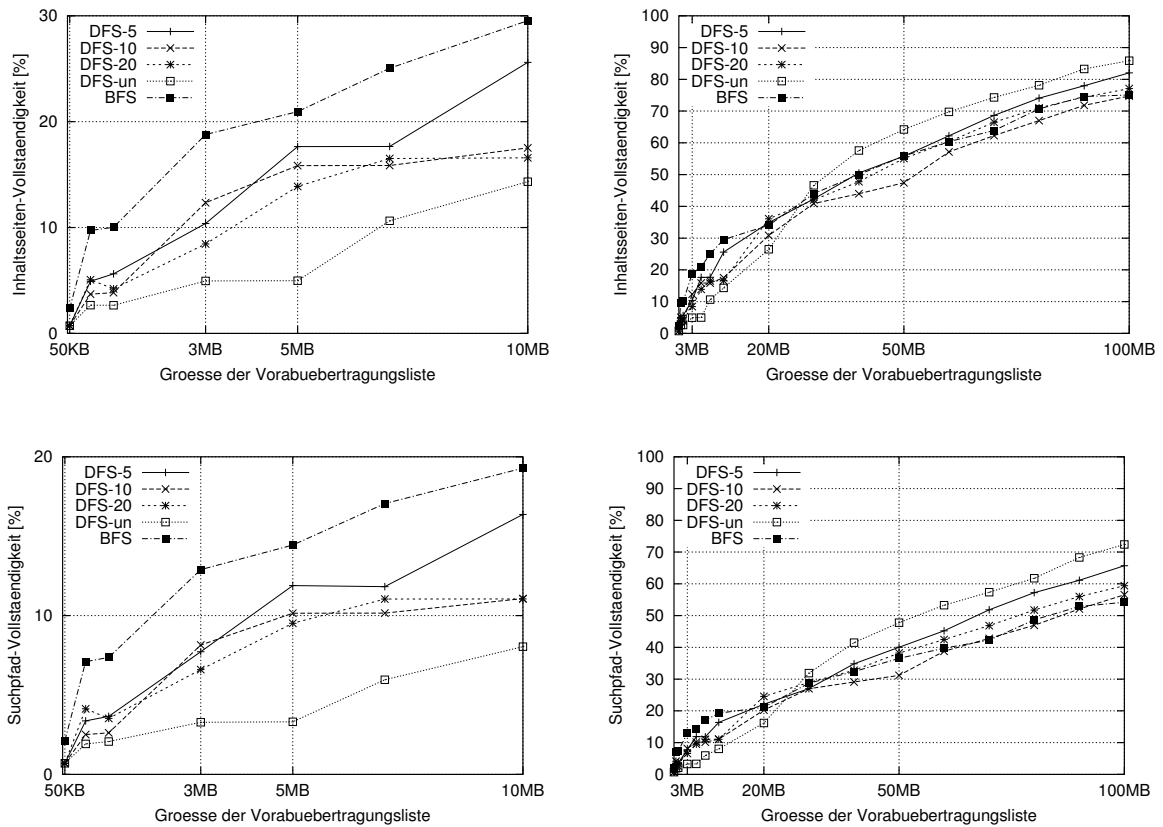


Abbildung 6.18: Existierendes Breiten- und Tiefensuchverfahren - Ergebnis

Während für den Breitensuchansatz (BFS) keine Parameter gesetzt werden, kann beim Tiefensuchansatz (DFS) eine maximale Abstiegstiefe der Rekursion angegeben werden. Der

natürliche Zahlenwert x legt dann fest, dass keine Pfade mit einer Länge größer x abgeschritten werden.

Die Ergebnisse in Abb. 6.18 zeigen, dass für die unteren Listenbereiche der Breitenansatz am besten arbeitet. Bei größeren Listen liefert dann der Tiefensuchansatz die besten Ergebnisse, dessen Abstiegstiefe unbegrenzt ist (DFS-un). Die Ergebnisse lassen sich sehr einfach erklären. Der Breitenansatz, aber auch die stärker beschränkten Tiefenansätze besuchen zu Beginn der Traversierung mehr Sitzungsstartknoten als der unbeschränkte Tiefenansatz. Erst bei der durch größere Listen ermöglichten längeren Laufzeit des unbeschränkten Tiefenverfahrens besucht dieses immer mehr Sitzungsstartknoten. Da dieses Verfahren auch sehr tiefe Pfade abgeht, werden die Verfahren irgendwann überholt, die den Graphen nur sehr "flach" abgehen und keine tiefen Pfade verfolgen.

Vergleich

Der Vergleich des Clusterverfahrens erfolgt nach den gesehenen Ergebnissen mit dem Breiten- und dem unbeschränkten Tiefensuchansatz.

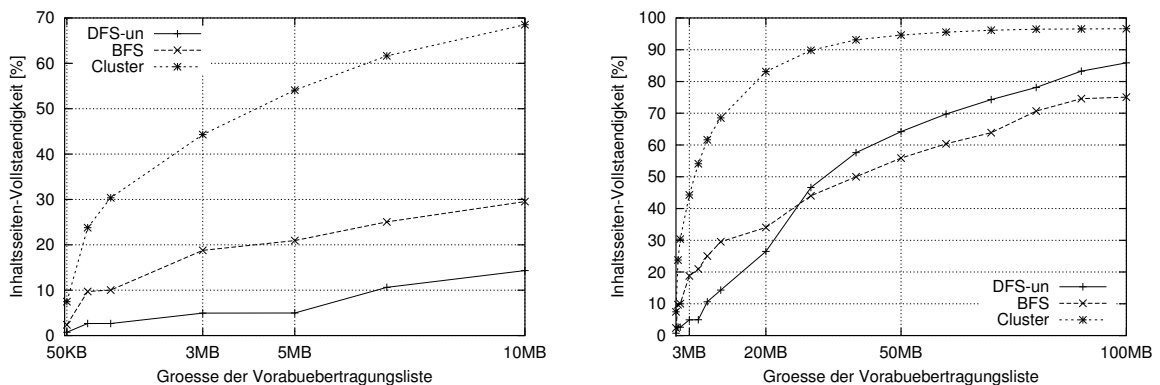


Abbildung 6.19: Verfahrensvergleich anhand Inhaltsseiten-Vollständigkeit

Die Ergebnisse in Abb. 6.19 und Abb. 6.20 zeigen ein deutlich besseres Ergebnis beim Clusterverfahren für sämtliche Größen von Vorabübertragungslisten. Zum Beispiel zeigt die Inhaltsseiten-Vollständigkeit bei der 1 MB-Liste eine Verbesserung, im Vergleich zum Breitenansatz, von ca. 10 % auf etwa 30 % Vollständigkeit, also eine Verdreifachung der Zufriedenheit der Benutzer. Aber auch für alle anderen Listen bis zu 30 MB Größe liefert das Clusterverfahren Werte für die Inhaltsseiten-Vollständigkeit, die die Werte der Vergleichsverfahren um mehr als das Doppelte übertreffen.

Bei der Suchpfad-Vollständigkeit ergibt sich ein ganz ähnliches Bild. Zwar liegen die absoluten Werte logischerweise unter denen der Inhaltsseiten-Vollständigkeit, die Verhältnisse der Ergebnisse zwischen den verschiedenen Verfahren ähneln sich jedoch stark. Auch hier liefert das Clusterverfahren im Listenbereich bis 30 MB Ergebnisse, die eine Verdopplung bis Verdreifachung der Suchpfad-Vollständigkeit bedeuten.

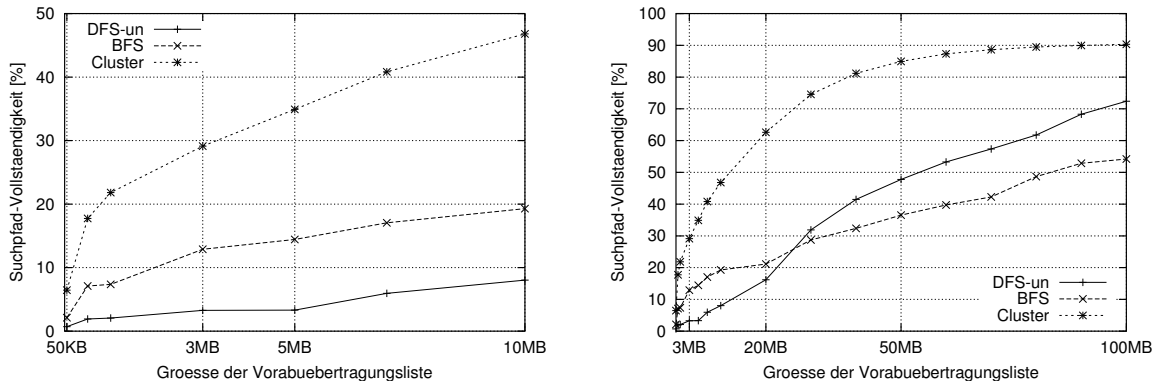


Abbildung 6.20: Verfahrensvergleich anhand Suchpfad-Vollständigkeit

Auch die Betrachtung der absoluten Werte zeigt das sehr gute Ergebnis. Bereits mit einer 10 MB großen Vorabübertragungsliste können Benutzer fast 50 % ihrer Logsuchpfade genau so abschreiten, wie es beabsichtigt ist, ohne auf eine nicht vorhandene Seite zu treffen. Weiter sind in einer solchen Liste fast 70 % der angeforderten Inhaltsseiten enthalten. Bedenkt man, dass mit der 10 MB-Liste nicht einmal 6 % der Gesamtgröße des Graphen übertragen werden, so sind diese Ergebnisse umso höher einzuschätzen.

6.4.3 Fazit

Das Clusterverfahren liefert sehr gute Werte bei einer hinreichend kurzen Laufzeit. Nicht nur die Verbesserung der Bewertungsergebnisse im Vergleich zu Tiefen- und Breitenansatz spricht für dieses Verfahren. Auch die Tatsache, dass die Steigung der Clusterkurven in den Abbildungen 6.19 und 6.20 stetig abnimmt, zeigt, dass die Cluster in der richtigen Reihenfolge in die Vorabübertragungsliste übernommen wurden. Plötzliche Steigungsanstiege, wie sie bei den anderen beiden Kurven vereinzelt zu beobachten sind, bedeuten, dass Seiten in der falschen Reihenfolge ihrer Wichtigkeit für den Benutzer aufgenommen werden.

Die erhebliche Verbesserung der Ergebnisse ist zwei Faktoren zu verdanken. Als einer dieser Faktoren sorgt die Berücksichtigung des Pfadgewichts und der Clustergröße für eine Verbesserung des Ergebnisses. Das Ergebnis des Clusterverfahrens, das ausschließlich diese beiden Eigenschaften berücksichtigt, ist in Abb. 6.21 nochmals dargestellt. In diesem Fall wird in jedem Rekursionsschritt für den gegenwärtig besuchten Knoten ein Cluster aufgebaut. Man sieht hier bereits die deutliche Verbesserung der Suchpfad-Vollständigkeit gegenüber dem Breitensuchverfahren (BFS). Wird nun als zweiter Faktor noch die Inhaltseigenschaft berücksichtigt und werden die Cluster den Suchpfaden entsprechend aufgebaut, so verbessert sich das Ergebnis nochmals.

Aus der Abbildung lässt sich auch folgern, dass die Verbesserungen aus der Berücksichtigung der Inhaltseigenschaft verhältnismäßig gering ausfallen. Das Ergebnis des Verfahrens, das die Knotenauswahl nur auf Größe und Pfadgewicht der Cluster basierend trifft,

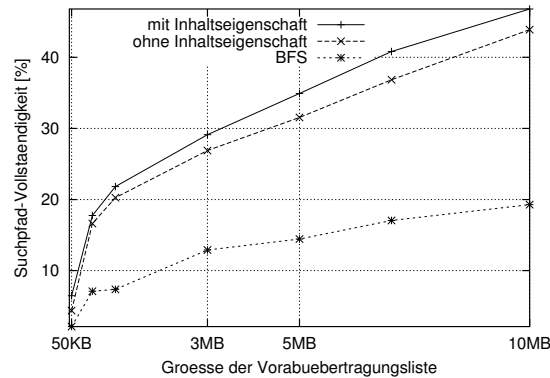


Abbildung 6.21: Einfluss der Faktoren auf Ergebnisverbesserung

zeigt, dass diese beiden Eigenschaften der Hauptgrund für die Verbesserungen sind. Am Beispiel der 5MB-Liste lässt sich dies gut ausmachen. Im Vergleich zum Breitensuchansatz bringt das Clusterverfahren, das die Inhaltseigenschaft ignoriert, eine Erhöhung der Suchpfad-Vollständigkeit von etwa 17%. Wird die Inhaltseigenschaft berücksichtigt und werden so die Cluster nach Suchpfaden aufgebaut, bringt dies lediglich eine weitere Verbesserung von etwa 3%.

Der große Vorteil des Clusterverfahrens liegt also darin, dass die Knoten nicht sofort bei der Traversierung des Graphen in die Vorabübertragungsliste aufgenommen werden, sondern zunächst mit Hilfe der Cluster bewertet werden. Ein anschließender Vergleich der Knoten sorgt für die Sortierung.

Abschließend lässt sich festhalten, dass das Clusterverfahren vor allem deswegen deutlich bessere Ergebnisse als das existierende Tiefen- und Breitensuchverfahren liefert, weil die Knoten nicht sofort in die Vorabübertragungsliste übernommen werden, sondern erst nach einem Kriterium, hier dem Steigungsansatz aus Def. 15, sortiert werden. Die Berücksichtigung der Inhaltswahrscheinlichkeit und der damit verbundene Aufbau von Suchpfaden sorgt für eine weitere Verbesserung des Verfahrens. Diese fällt jedoch, im Vergleich zum Vorteil durch die Knotensortierung, weitaus geringer aus.

Kapitel 7

Zusammenfassung

Nach der Evaluierung des Clusterverfahrens werden abschließend in diesem Kapitel die gewonnenen Erkenntnisse dieser Arbeit zusammengestellt. Zuletzt wird ein Ausblick gegeben, wie das Clusterverfahren in Zukunft vielleicht noch zu verbessern ist. Es wird außerdem auf die zukünftige Rolle des Hoarding eingegangen und was dies für das Clusterverfahren bedeutet.

7.1 Erkenntnisse dieser Diplomarbeit

Ausgehend von einem gegebenen, sehr einfach gehaltenen Tiefen- und Breitensuchverfahren zur Durchführung einer Vorabübertragung von Webseiten wurde in dieser Arbeit gezeigt, dass durch Clusterbildung, -bewertung und die anschließende Sortierung deutlich bessere Ergebnisse erzielt werden können. Ein besseres Ergebnis bedeutet in diesem Zusammenhang, dass Benutzer mit der Qualität der erstellten Vorabübertragungsliste zufriedener sind.

Anfangen von einem sehr einfachen Clusteransatz, der auf die rekursive Tiefensuche aufbaut, wurde nach und nach das letztendlich angewendete Verfahren erarbeitet. Hierbei wurde festgestellt, dass Cluster im Informationsgraphen, welche Verzweigungen oder Zyklen enthalten, unerwünscht sind. Diese führen zu Problemen mit der Semantik und der Bewertung der Cluster. Pfade hingegen repräsentieren Folgen von Seitenaufrufen. In Verbindung mit dem Verhalten von Benutzern auf der Suche nach Informationen, wird die Zusammengehörigkeit von Knoten durch die Suchpfadsemantik ermittelt. Durch die Repräsentation von Pfaden erweist sich auch die Bewertung der Cluster als einfach.

Die Ermittlung der Zufriedenheit der Benutzer mit den Ergebnissen des Clusterverfahrens wird durch die erarbeiteten Bewertungsmetriken ermöglicht. Hierbei wurden vor allem die Inhaltsseiten- und Suchpfad-Vollständigkeit definiert, welche gemeinsam eine gute Aussagekraft über die Benutzerzufriedenheit besitzen. Hierbei wird davon ausgegangen, dass der Benutzer umso zufriedener ist, je mehr gesuchte Informationen er findet.

Die Evaluierung mit Hilfe des UCW-Modells [2] ergab, dass das verwendete Clusterverfahren um ein Vielfaches bessere Werte liefert als das bisherige Tiefen- und Breitensuchverfahren.

Es wurde jedoch auch deutlich, dass die Bedeutung der Suchpfade geringer ausfällt als erhofft. Die Ergebnisverbesserung entsteht vor allem dadurch, dass durch die Clusterbildung die einzelnen Webseiten nicht sofort beim Antreffen durch die Traversierung in die Vorabübertragungsliste übernommen werden, wie es beim bisherigen Verfahren ist. Es wird stattdessen zuerst eine Bewertung und ein Vergleich der Knoten bzw. Cluster vorgenommen. Durch die damit verbundene Sortierung ergeben sich die deutlich besseren Vorabübertragungslisten.

7.2 Ausblick

Sollte in Zukunft die Struktur des Informationsgraphen erweitert werden, lässt sich das hier vorgestellte Verfahren vielleicht weiter verbessern. So ist zum Beispiel denkbar, dass zukünftig im Informationsgraphen nicht nur die Adressen der Webseiten gespeichert werden, sondern auch Informationen über den eigentlichen Inhalt der Seiten hinterlegt werden. Dies würde die Entwicklung neuer Clustersemantiken erlauben, die eventuell ein noch besseres Clusterverfahren ermöglichen.

Wie gezeigt wurde, kann mit dem in dieser Arbeit vorgestellten Clusterverfahren aus der Analyse vergangenen Benutzerverhaltens eine gute Vorhersage über zukünftig angeforderte Webseiten getroffen werden. Zukünftig werden solche Hoarding-Verfahren, zu denen das hier vorgestellte Clusterverfahren zählt, mit Sicherheit solange Anwendung finden, bis eine flächendeckende Versorgung mit Funknetzen gewährleistet ist. Es ist abzusehen, dass in der Zukunft irgendwann eine solche Versorgung erreicht sein wird. Angesichts der heutigen Verfügbarkeit dieser Funknetze und der gegenwärtigen Ausbreitung werden bis zu diesem Zeitpunkt aber mit Sicherheit noch einige Jahre vergehen. Bis dahin wird das Hoarding weiterhin die beste Möglichkeit darstellen, auf Endgeräten ohne Netzanbindung mit großen, entfernt gespeicherten Datenmengen zu arbeiten.

Durch die fortwährende Entwicklung hin zu immer kleineren und dabei mehr Kapazität bietenden Speicherchips ist zu erwarten, dass sich der verfügbare Speicherplatz auf mobilen Endgeräten erhöhen wird. Ebenfalls ist zu erwarten, dass die Funkverbindungen sich hin zu immer größeren Bandbreiten entwickeln werden und so mehr Daten pro Zeiteinheit versendet werden können. Das bedeutet gleichzeitig, dass die am häufigsten verwendeten Größen von Vorabübertragungslisten vermutlich ebenfalls steigen werden. Das Clusterverfahren hat sich in den Vergleichen mit dem bisherigen Tiefen- und Breitensuchverfahren auch für größere Listen als sehr gut erwiesen. Durch die ebenfalls nur gering höhere Laufzeit, die dann ebenfalls durch schnellere Prozessoren aufgefangen werden kann, wird sich dieses Clusterverfahren auch dann noch sinnvoll einsetzen lassen.

Anhang A

Implementierung

Folgend wird erläutert, wie die in der Programmiersprache Java erfolgte Implementierung vorgenommen wurde. Neben einem Klassendiagramm werden ebenfalls die einzelnen Schnittstellendefinitionen vorgestellt.

A.1 Struktur der Klassen

Die Implementierung erfolgte unter Eclipse [7]. Hier existierte für das Hoarding bereits ein eigenes Projekt. Dies enthielt bis zu dieser Diplomarbeit die Implementierungen des Informationsgraphen, des Tiefen- und Breitensuchverfahrens sowie weitere dazugehörige Funktionalitäten. Das Clusterverfahren wurde durch das Hinzufügen eines eigenen Pakets eingebaut.

In Abb. A.1 sind sämtliche Schnittstellen des Pakets in den Kästchen aufgeführt. Neben den zugehörigen Implementierungsklassen, die jeweils neben oder unterhalb der Kästchen aufgeführt sind, sind auch die Zusammenhänge der Klassen untereinander dargestellt. Die einzige Ausnahme ist das Kästchen für den Clusteralgorithmus. *ClusterAlgorithm* ist keine Schnittstelle sondern eine abstrakte Klasse, welche die existierende abstrakte Klasse *TraversalAlgorithm* erweitert. Die Erweiterung besteht darin, dass der Clusteralgorithmus in zwei Schritten ablaufen muss und nicht, wie beim bisherigen Verfahren, ein Schritt ausreichend ist. Diese beiden Schritte sind zunächst der Aufbau der Cluster und anschließend die Erstellung der Vorübertragungsliste. Die konkrete Implementierung des in dieser Arbeit vorgestellten Clusteralgorithmus aus Algorithmus 4 ist in *DepthFirstClusterAlgorithm* zu finden.

Die im ersten Schritt erstellten Cluster werden in Objekten gespeichert, welche *Cluster* implementieren. Die Menge der erstellten Cluster wird in einem *ClusterContainer* gespeichert. Sämtliche Cluster benutzen für ihre Gewichtung ein *ClusterWeighting*. Der vorgestellte Steigungsansatz wurde in *SlopeClusterWeighting* implementiert.

Nach der Erstellung des *ClusterContainer* wird die Vorübertragungsliste erstellt. Hierzu wird eine Implementierung von *HoardListAlgorithm* verwendet. Die zu erstellende Liste muss eine Instanz der bereits existierenden Klasse *HoardList* sein.

Die abschließende Evaluierung benutzt dann die erstellte Vorübertragungsliste, um entweder die Liste selbst (*HoardListAssessment*) oder das Clusterverfahren ganzheitlich (*TraversalAssessment*) zu bewerten.

Durch die Verwendung von Schnittstellen und die klare Strukturierung ist es leicht, später neue Ansätze zu integrieren. Sei es nun ein neuer Algorithmus, eine andere Bewertungsmethode der Cluster oder etwas anders. Durch das Implementieren der entsprechenden Schnittstelle kann ein neuer Ansatz schnell implementiert werden. Um bei der Auswahl der zu verwendenden Implementierung möglichst flexibel zu sein, werden die zu ladenden, die Schnittstellen implementierenden Klassen erst zur Laufzeit bestimmt. Die Auswahl der Implementierungsklasse legt der Programmierer in einer Textdatei fest. In einer Factory-Klasse (*Clustering-Factory*), die hier nicht extra aufgeführt wurde, erfolgt dann die Instanziierung mit Hilfe des Java-ClassLoader.

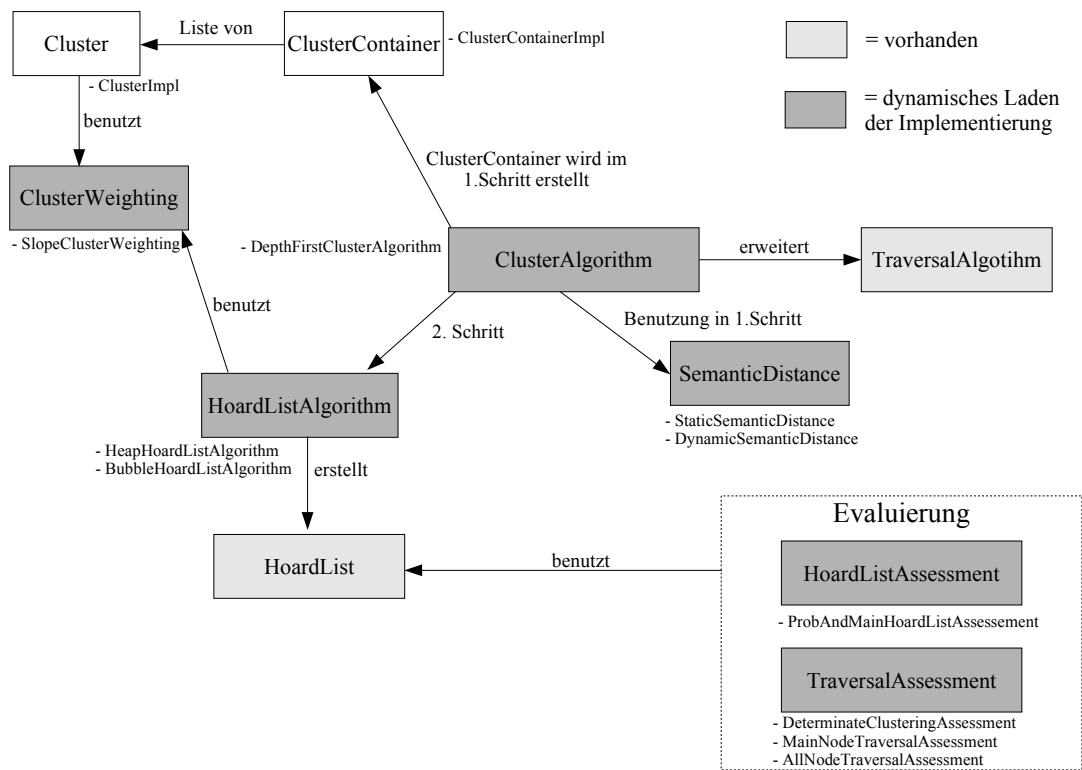


Abbildung A.1: Klassendiagramm

A.2 Beschreibung der Schnittstellen

Nachdem nun ein Gesamtüberblick über die Klassenstruktur gegeben wurde, erfolgt nun eine kurze Beschreibung der Schnittstellen selbst. Diese Beschreibung ist ebenfalls in den Kommentaren des Quellcodes zu finden.

Cluster

- **getId()** liefert die eindeutige ID des Clusters.
- **getVertices()** liefert eine Liste der im Cluster enthaltenen Knoten.
- **setVerticesAndPathWeight(LinkedList vertices, double pathWeight)** löscht den alten Inhalt des Clusters und nimmt die in der Liste übergebenen Knoten. Der letzte Knoten der Liste wird als der letzte Inhaltsknoten erwartet. Das Pfadgewicht muss dabei ebenfalls neu gesetzt werden.
- **getSize()** liefert die Summe aller Webseitengrößen, die durch die Knoten repräsentiert werden.
- **getPathWeight()** liefert das Pfadgewicht für den Cluster.
- **getMainProb()** liefert die Inhaltswahrscheinlichkeit des letzten Knotens des Clusters.
- **getBuildUpSize()** wird beim Aufbau der Vorabübertragungsliste benötigt. Diese Methode liefert die Summe der Webseitengrößen im Cluster, die noch nicht in die Liste aufgenommen wurden.
- **computeBuildUpSize()** muss aufgerufen werden, wenn ein Knoten des Clusters in die Liste aufgenommen wird, ohne dass der Cluster selbst aufgenommen wird. Hier wird auch gleichzeitig die Neubewertung des Clusters vorgenommen.
- **setInHoardList(boolean inHoardList)** setzt eine Variable, die aussagt, ob der Cluster in die Vorabübertragungsliste aufgenommen wurde.
- **isInHoardList()** liefert den Wert eben angesprochener Variable.
- **getWeight()** liefert das Ergebnis der Clusterbewertung.
- **getHoardListBuildIndex()** liefert den Index des Clusters in der Datenstruktur, die zum Aufbau der Vorabübertragungsliste verwendet wird. Bei der Verwendung des Heap-Verfahrens ist dies zum Beispiel der Index im Heap. Dies ist nötig, da der Cluster bei Bewertungsänderungen neu sortiert werden muss und dazu muss man wissen, an welcher Stelle im Heap er sich derzeit befindet.
- **setHoardListBuildIndex(int index)** ist die entsprechende Methode zum Setzen des Indexes.

ClusterContainer

- **getAllClusters()** liefert die Menge an Clustern zurück.
- **getCluster(int id)** liefert den Cluster mit der entsprechenden ID.
- **addCluster(Cluster cl)** fügt dem Container einen Cluster hinzu.

- **getNonClusterVertexQueue()** liefert die Warteschlange mit den Knoten, die bei der Traversierung eventuell keinem Cluster hinzugefügt wurden.
- **appendToNonClusterVertexQueue(InfoGraphVertex vertex)** dient zum Aufbau der Liste durch Anfügen von Knoten ans Ende der Warteschlange.
- **clear()** leert die Clustermenge und die Warteschlange.

ClusterWeighting

- **weightCluster(Cluster c)** bewertet einen Cluster.

SemanticDistance

- **getSemanticDistance(Relation rel)** liefert die semantische Distanz zweier Knoten, die durch die Kante *rel* direkt miteinander verbunden sind.
- **emptyEdgeWeightMap()** wird für die dynamische Variante benötigt. Damit nicht jedes Mal beim Antreffen eines Knotens das größte Kantengewicht aller seiner ausgehenden Kanten berechnet werden muss, wird dieses nach der ersten Berechnung in einer Hashmap gespeichert. Vor dem erneuten Traversieren des Graphen muss diese Hashmap aber wieder geleert werden, da sich Kantengewichte verändert haben können.

ClusterAlgorithm

Da es sich hier nicht um eine Schnittstelle sondern um eine abstrakte Klasse handelt, wird hier die abstrakte Methode vorgestellt.

- **clusterGraph(InfoGraph ig, GlobalValues gv)** liefert einen ClusterContainer als Ergebnis der Traversierung des Informationsgraphen. Verwendete, manuell zu setzende Parameter sind in *gv* enthalten.

HoardListAlgorithm

- **createNewDynamicSizeHoardList(ClusterContainer con, long size)** liefert eine neue Vorabübertragungsliste mit der angegebenen Größe nach dem inkrementellen Ansatz aus Abschnitt 5.1.4. Die erstellte Liste wird intern gespeichert.
- **createNewFixedSizeHoardList(ClusterContainer con, long size)** liefert eine neue Vorabübertragungsliste mit einer festen Größe. Hier wird der inkrementelle Ansatz also nicht angewendet. Auch dieses Ergebnis wird intern gespeichert.
- **continueHoardList(long size)** baut eine inkrementell erstellte Liste weiter auf. Funktioniert nicht, wenn die zuletzt erstellte Liste fester Größe war.
- **getCurrentHoardList()** liefert die gegenwärtig intern gespeicherte Liste.
- **getHoardListSize()** liefert die Größe der intern gespeicherten Liste.

- **isPresentHoardListFixed()** sagt aus, ob die interne Liste mit fester Größe aufgebaut wurde oder inkrementell.

HoardListAssessment

- **assessHoardList(HoardList hl, ClusterContainer clusCon)** bewertet die Vorabübertragungsliste auf Basis der zur Verfügung stehenden Cluster.

TraversalAssessment

- **assessTraversalAlgorithm(HoardList hl, LogFile log)** bewertet das Ergebnis eines Verfahrens zur Erstellung der Vorabübertragungsliste anhand eines Benutzerlogs.
- **assessTraversalAlgorithm(HoardList hl, LogFileContainer logCon)** bewertet das Ergebnis eines Verfahrens zur Erstellung der Vorabübertragungsliste anhand einer Menge an Benutzerlogs. Das Ergebnis ist der Durchschnitt der einzelnen Bewertungen.

Anhang B

Begriffslexikon

- **Hoarding** bezeichnet in dieser Arbeit die Vorabübertragung von Webseiten. Die Entscheidung über die zu übertragenden Webseiten wird auf Basis der Analyse vergangenen Benutzerverhaltens getroffen. Hoarding wird benutzt, um es einem mobilen Benutzer zu ermöglichen, Webseiten auch dann abrufen zu können, wenn gerade keine Netzverbindung zur Verfügung steht. Die Herausforderung besteht darin, zu bestimmen, welche Webseiten voraussichtlich in Zukunft angefordert werden.
- **Hoardcache** ist der Speicherplatz auf dem mobilen Endgerät, der für vorab übertragene Daten verwendet werden kann. Die Größe des Hoardcache kann für das Hoarding-Verfahren sehr relevant sein.
- **Informationsgraph** ist der Graph, der das Internetverhalten sämtlicher Benutzer, die ihre Logdateien als Eingabe zur Verfügung stellen, repräsentiert. Der Informationsgraph ist ein gerichteter, gewichteter, knoten- und kantenbeschrifteter Graph. Jeder Knoten repräsentiert eine Webseite. Jede Kante bedeutet, dass deren jeweiliger Beginn- und Endknoten direkt aufeinander folgend angefordert wurden. Eine entsprechende Kantengewichtung gibt die Häufigkeit bzw. Wahrscheinlichkeit der Anforderungsfolge wieder. Der Graph besitzt einen Initialknoten g , der eine Kante zu jedem Knoten besitzt, der mindestens einmal als Startseite für eine Sitzung gedient hat. Von dieser Startseite aus werden Kanten und Knoten, entsprechend der Sitzung, erzeugt oder aktualisiert. Genauere Definitionen sind Abschnitt 2.2.3 zu entnehmen.
- **Logdateien** protokollieren das Zugriffsverhalten eines Benutzers auf Webseiten. In den Logdateien werden verschiedene Daten, wie Besuchszeit, Größe und Adresse zu sämtlichen angeforderten Webseiten gespeichert. Logdateien bzw. die daraus generierten Sitzungen dienen als Eingabe für den Informationsgraphen.
- **Semantische Distanz** ist ein Maß für die Zusammengehörigkeit von Paaren von Webseiten. Das Maß der semantischen Distanz wird für die Clusterbildung benötigt. Idealerweise sollten Webseiten, die eine niedrige semantische Distanz aufweisen, auch im selben Cluster liegen. Die Erarbeitung von Ansätzen für eine geeignete Definition solcher semantischen Distanzen ist Inhalt dieser Diplomarbeit. Dabei werden Distanzen

nur zwischen Paaren von Knoten definiert, die direkt durch eine Kante verbunden sind.

- **Semantische Zusammengehörigkeit** ist das Gegenteil zur semantischen Distanz. Eine hohe semantische Distanz bedeutet eine niedrige semantische Zusammengehörigkeit und vice versa.
- **Semantisch starke Kante** ist eine Kante, für deren jeweilige Endknoten gilt, dass die semantische Distanz zwischen diesen Knoten unter einem Grenzwert s_{th} liegt.
- **Semantisch schwache Kante** ist das Gegenteil einer semantisch starken Kante.
- **Sitzung** ist eine zusammenhängende Folge von Webseitenaufrufen eines Benutzers, die in den Logdateien durch eine Folge von Einträgen wiedergegeben wird. Die Einträge einer Sitzung zeichnen sich dadurch aus, dass der Benutzer zwischen den einzelnen Aktivitäten die Anwendung nicht unterbrochen oder sich zwischendurch mit etwas anderem beschäftigt hat.
- **Vorabübertragungsliste** ist die Liste der vorab zu übertragenden Webseiten. Ziel dieser Diplomarbeit ist es, diese Liste möglichst mit den Webseiten zu füllen, die später auch vom Benutzer angefordert werden.
- **Zeitliche Glättung** bedeutet im Zusammenhang mit dem Informationsgraphen, dass Logeinträge mit zunehmendem Alter weniger Berücksichtigung im Graphen erfahren. Nach einer gewissen Zeitspanne werden sie komplett ignoriert. Dies ist nötig, damit der Informationsgraph immer möglichst nahe am aktuellen Benutzerverhalten orientiert ist.

Literaturverzeichnis

- [1] Doug Beeferman and Adam Berger. Agglomerative clustering of a search engine query log. In *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 407–416, New York, NY, USA, 2000. ACM Press.
- [2] Susanne Bürklen, Pedro José Marrón, Serena Fritsch, and Kurt Rothermel. User centric walk: An integrated approach for modeling the browsing behavior of users on the web. In *Proceedings of the 38th IEEE Annual Simulation Symposium (ANSS'05), San Diego, CA, USA, 2005*.
- [3] Susanne Bürklen, Pedro José Marrón, and Kurt Rothermel. An enhanced hoarding approach based on graph analysis. In *Proceedings of the IEEE International Conference on Mobile Data Management*, pages 358–369, 2004.
- [4] Steve Chien and Nicole Immorlica. Semantic similarity between search engine queries using temporal correlation. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 2–11, New York, NY, USA, 2005. ACM Press.
- [5] R. Cooley, B. Mobasher, and J. Srivastava. Grouping web page references into transactions for mining world wide web browsing patterns. In *KDEX '97: Proceedings of the 1997 IEEE Knowledge and Data Engineering Exchange Workshop*, page 2, Washington, DC, USA, 1997. IEEE Computer Society.
- [6] D. B. Crouch, C. J. Crouch, and G. Andreas. The use of cluster hierarchies in hypertext information retrieval. In *HYPERTEXT '89: Proceedings of the second annual ACM conference on Hypertext*, pages 225–237, New York, NY, USA, 1989. ACM Press.
- [7] <http://www.eclipse.org/>, 17.11.2005.
- [8] Michale R. Garey and David S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- [9] D. J. Goodman, J. Borrás, N.B. Mandayam, and R. D. Yates. Infostations: a new system model for data and messaging services. *Vehicular Technology Conference, 1997 IEEE 47th*, 2:969–973, 1997.
- [10] John A. Hartigan. *Clustering Algorithms*. John Wiley and Sons, 1975.

-
- [11] David C. Hoaglin, John Tukey, and Frederick Mosteller. *Understanding robust and exploratory data analysis*. Wiley Interscience, 2000.
- [12] <http://grouper.ieee.org/groups/802/11/>, 27.10.2005.
- [13] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, 1999.
- [14] Anil K. Jain, Alexander Topchy, Martin H. C. Law, and Joachim M. Buhmann. Landscape of clustering algorithms. In *ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 1*, pages 260–263, Washington, DC, USA, 2004. IEEE Computer Society.
- [15] Geoffrey H. Kuenning, Wilkie Ma, Peter Reiher, and Gerald J. Popek. Simplifying automated hoarding methods. In *Proceedings of the 5th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems*, pages 15–21. ACM Press, 2002.
- [16] Geoffrey H. Kuenning and Gerald J. Popek. Automated hoarding for mobile computers. In *Proceedings of the sixteenth ACM symposium on Operating systems principles*, pages 264–275. ACM Press, 1997.
- [17] Anton Leuski. Evaluating document clustering for interactive information retrieval. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 33–40, New York, NY, USA, 2001. ACM Press.
- [18] Xiaoyong Liu and W. Bruce Croft. Cluster-based retrieval using language models. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 186–193, New York, NY, USA, 2004. ACM Press.
- [19] Manuel J. Mana López, Manuel De Buenaga, and José M. Gómez-Hidalgo. Multidocument summarization: An added value to clustering in interactive retrieval. *ACM Trans. Inf. Syst.*, 22(2):215–241, 2004.
- [20] Gerhard Merziger and Thomas Wirth. *Repetitorium der höheren Mathematik*. Binomi, fourth edition, 1999.
- [21] Werner Neubauer, Egon Bellgardt, and Andreas Behr. *Statistische Methoden*. Verlag Vahlen, second edition, 2002.
- [22] <http://www.nexus.uni-stuttgart.de>, 26.10.2005.
- [23] Shirish Hemant Phatak and B. R. Badrinath. Data partitioning for disconnected client server databases. In *MobiDe '99: Proceedings of the 1st ACM international workshop on Data engineering for wireless and mobile access*, pages 102–109, New York, NY, USA, 1999. ACM Press.

-
- [24] D. Pierrakos, G. Paliouras, C. Papatheodorou, and C.D. Spyropoulos. Web usage mining as a tool for personalization: A survey. *User Modeling and User-Adapted Interaction Journal*, 13(4):311–372, 2003.
- [25] Edie Rasmussen. Clustering algorithms. *Information retrieval: data structures and algorithms*, pages 419–442, 1992.
- [26] Robert Sedgewick. *Algorithms in C++, Parts 1-4: Fundamentals, Data Structure, Sorting, Searching*. Addison-Wesley, third edition, 1998.
- [27] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. *KDD Workshop on Text Mining, 2000.*, 2000.
- [28] Wensi Xi, Edward A. Fox, Weiguo Fan, Benyu Zhang, Zheng Chen, Jun Yan, and Dong Zhuang. Simfusion: measuring similarity using unified relationship matrix. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 130–137, New York, NY, USA, 2005. ACM Press.
- [29] C.T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, 20:68–86, 1971.