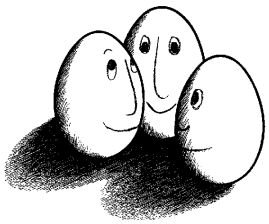
Diplomarbeit

Maschinelle Lernverfahren für kollaboratives Tagging

Andreas Kaspari



Diplomarbeit
am Fachbereich Informatik
der Universität Dortmund

14.09.2007

Betreuer:

Prof. Dr. Katharina Morik
Dipl.-Inform. Michael Wurst

Danksagung

Ein herzliches Dankeschön an alle, die mir bei der Erstellung der vorliegenden Arbeit helfend zur Seite gestanden haben. Mein besonderer Dank gilt meinen beiden Betreuern Prof. Dr. Katharina Morik und Dipl.-Inform. Michael Wurst.

Inhaltsverzeichnis

Abbildungsverzeichnis

Notation

$A, B, C, ...$ – werden zur Bezeichnung von Mengen atomarer Elemente (auch einfache Mengen genannt) oder Relationen zwischen einfachen Mengen verwendet.

$a, b, c, ...$ – werden zur Bezeichnung einzelner Elemente einfacher Mengen verwendet.

$\mathsf{A}, \mathsf{B}, \mathsf{C}, ...$ – bezeichnen Mengenfamilien, also Mengen von Mengen.

$\mathsf{M}_{|i}$ – bezeichnet eine Teilmenge, die alle i-elementigen Elemente einer Mengenfamilie M enthält. Sei zum Beispiel M eine Menge und $\mathsf{M} \subseteq \mathcal{P}(M)$ eine Mengenfamilie. Dann wird mit $\mathsf{M}_{|i}$ die Menge $\{M' \in \mathsf{M} \mid i = |M'|\}$ für ein $i \in \mathbb{N}$ bezeichnet.

$\mathfrak{A}, \mathfrak{B}, \mathfrak{C}, ...$ – werden zur Bezeichnung von Mengenfamilien von Mengenfamilien verwendet.

Allgemein: Bezeichner können auch zusammengesetzt werden. So ist beispielsweise ABC ein Bezeichner für eine einfache Menge und ABC ein Bezeichner für eine Mengenfamilie.

Kapitel 1

Einleitung

Tagging-Systeme sind Web-Anwendungen, die es ihren Benutzern ermöglichen, Ressourcen mit Tags zu annotieren. Ressourcen sind beispielsweise Web-Links, Fotos oder Publikationen und Tags sind textuelle Bezeichner, die von den Benutzern frei gewählt werden. Das Taggen von Ressourcen ist ein Mittel zur Strukturierung von Information nach den persönlichen Vorstellungen und Erfordernissen des Benutzers. Eine grundliegende Eigenschaft von Tagging-Systemen ist der freie Zugang zu den Einträgen jedes Benutzers. Auf diese Weise wird neben der Möglichkeit der persönlichen Strukturierung von Ressourcen zum einen ein Wissensaustausch zwischen den Benutzern ermöglicht und zum anderen die Strukturierung als kollaborativer Prozess gefördert.

Tagging-Systeme, wie Del.icio.us oder Flickr, erlauben ihren Benutzern nur eine sehr eingeschränkte Navigation durch den Datenbestand, da die Beziehungen zwischen den Tags weitestgehend verborgen bleiben. Bestehende Verfahren zur Verbesserung der Zugänglichkeit wählen Tags aus, die für den Datenbestand repräsentativ sind, und ordnen diese nach Themengebieten in Cluster an. Dabei kommen sowohl klassische Clustering-Verfahren, wie beispielsweise KMeans, als auch Verfahren aus dem Bereich des Clusterings von Graphen zum Einsatz. Die hohe Dimensionalität der Eingabedaten führt zu einer schlechten Qualität der erzeugten Clusterings. Darüberhinaus führt die Dimensionalität kombiniert mit dem Volumen der zu verarbeitenden Daten zu einem schlechten Skalierungsverhalten. Weiterhin erhalten die erzeugten Cluster in der Regel keine Beschreibung, was eine spätere Interpretation durch den Benutzer erschwert. Es sind häufig Parametereinstellungen notwendig, die die Qualität des Ergebnisses stark beeinflussen und eine genaue Kenntnis des Verfahrens und der Eigenschaften der verwendeten Eingabedaten erfordern.

Frequent Itemset-basierte Ansätze, wie das Clustering-Verfahren Fihc von Ester et al. [12], wurden mit dem Ziel entwickelt, die zuvor erwähnten Probleme zu lösen. Obwohl es ursprünglich für das Clustering von Textdokumenten entwickelt wurde, lässt sich Fihc aufgrund der Ähnlichkeit der beiden Problemdomänen auch zum Clustern von Tags einsetzen. Fihc zeichnet sich durch eine hohe Cluster-Qualität und gute Skalierbarkeit aus und muss darüberhinaus nicht parametrisiert werden. Es wird eine baumartige Anordnung von Tags erzeugt, die sich sehr gut als Basis für die Navigation durch den Datenbestand eines Tagging-Systems eignet.

Bei Erzeugung der Lösung werden durch das Verfahren implizit mehrere Kriterien

optimiert. Zum einen kommt eine Ressource nur in einem Cluster vor, wodurch die Überlappung der Cluster minimiert wird. Zum anderen kommt jede Ressource aus dem Datenbestand in einem Cluster vor; die Vollständigkeit wird also maximiert. Diese Kriterien sind dem Benutzer jedoch unbekannt, weshalb es für ihn schwierig ist, die erzeugte Lösung zu interpretieren.

Zudem wird nur eine einzige Lösung erzeugt und der Benutzer hat keine Möglichkeit in das Verfahren einzugreifen, um seine persönlichen Anforderungen umzusetzen. Er möchte eine hierarchische Navigationsstruktur für den Datenbestand des Tagging-Systems erhalten, die zum einen übersichtlich ist und zum anderen den Datenbestand möglichst vollständig abdeckt. Darüberhinaus soll die erzeugte Struktur möglichst tief sein, um eine feine Steuerung der Navigation zu erlauben. Es ist offensichtlich, dass seine Anforderungen an die Struktur in Konkurrenz stehen. So sinkt beispielsweise die Übersichtlichkeit, wenn die Abdeckung steigt.

Im Rahmen dieser Arbeit wird ein Verfahren zur Erzeugung individueller, hierarchischer Navigationsstrukturen für Tagging-Systeme entwickelt. Da die Anforderungen an Navigationsstrukturen inhärent widersprüchlich sind, wird das Finden dieser Strukturen dabei als multikriterielles Suchproblem formuliert. Die zur Steuerung der Suche verwendeten Kriterien werden explizit dargestellt und es wird eine Menge alternativer Lösungen erzeugt, aus welcher der Benutzer die für ihn passendste auswählen kann. Durch die explizite Darstellung der Optimierungskriterien kann er Rückschlüsse auf die Eigenschaften der alternativen Lösungen ziehen.

Eine statische Navigationsstruktur für ein Tagging-System ist nach einiger Zeit nicht mehr aktuell, da der Datenbestand von den Benutzern ständig erweitert wird. Daher wird das Verfahren so erweitert, dass Strukturen für den erweiterten Datenbestand erzeugt werden können, die ähnlich zu einer vom Benutzer zu einem früheren Zeitpunkt ausgewählten Struktur sind.

Schließlich wird das Verfahren erprobt, indem Navigationsstrukturen für ein reales Tagging-System erzeugt werden. Dazu kommt ein Datensatz des Tagging-Systems BIBSONOMY [21] zum Einsatz.

Gliederung dieser Arbeit

Kapitel 2 erläutert, was Tagging-Systeme sind, warum sie sich gut als Mittel zur Organisation von Web-Ressourcen eignen und warum ihre Betrachtung interessant ist. Es stellt die Bedienkonzepte aktueller Tagging-Systeme und die vorhandenen Schwächen vor. Als Zielsetzung für diese Arbeit wird die Entwicklung eines Verfahrens zur Beseitigung dieser Schwächen definiert und es werden alle an dieses Verfahren gestellten Anforderungen ausgearbeitet.

Kapitel 3 stellt alle Methoden vor, die als Inspiration für das in der Arbeit entwickelte Verfahren dienten oder aber direkt als Bestandteil des Verfahrens verwendet

werden.

Kapitel 4 führt den Formalismus zur Beschreibung von hierarchischen Navigationsstrukturen für Tagging-Systeme ein und enthält eine detaillierte Beschreibung des Verfahrens zur Erzeugung dieser Strukturen.

Kapitel 5 befasst sich mit der experimentellen Erprobung des Verfahrens auf den Daten eines realen Tagging-Systems. Die Eigenschaften des verwendeten Datensatzes werden ebenfalls betrachtet.

Kapitel 6 fasst die Funktionsweise des entwickelten Verfahrens noch einmal zusammen und schließt diese Arbeit mit einer Betrachtung der erzielten Ergebnisse und einem Ausblick ab.

Kapitel 2

Tagging-Systeme

Tagging-Systeme, Anwendungen aus dem Bereich des Web 2.0, bilden die Grundlage dieser Arbeit. Dieses Kapitel befasst sich zunächst mit der Entstehungsgeschichte des Begriffs Web 2.0 und definiert anschließend, was man unter Tagging-Systemen versteht und warum ihre Betrachtung interessant ist. Anschließend werden aktuelle Tagging-Systeme und ihr Bedienkonzept betrachtet, wobei der Fokus besonders auf dem für diese Arbeit zentralen Aspekt des Browsings liegt. Anschließend werden die Probleme des aktuellen Bedienkonzepts diskutiert. Es folgt eine Beschreibung der Strategie zur Beseitigung dieser Probleme, welche dann in den folgenden Kapiteln ausgearbeitet und erprobt wird.

2.1 Was sind Tagging-Systeme?

Der Begriff *Web 2.0* wurde ursprünglich im Jahre 2004 vom O'Reilly Verlag erfunden [31]. Aus dem *Dot-com*-Crash im Jahre 2001 schlossen viele Menschen, dass das Web, nachdem es übermäßig hochgespielt worden war, stetig an Bedeutung verlieren würde. Tatsächlich entstanden aber regelmäßig interessante neue Webseiten und Webanwendungen. Mit diesen Entwicklungen sollte sich eine neue Konferenzreihe des Verlags beschäftigen - nur ein Name fehlte. Dale Dougherty, Mitbegründer von O'Reilly, schlug den Namen *Web 2.0* vor, um die Neuentwicklungen im Web seit dem Crash zu beschreiben - die *Web 2.0 Conference* war geboren.

In den folgenden Jahren wurde der Begriff weltweit aufgegriffen, auch wenn man sich über seine genaue Bedeutung bis heute nicht einig ist. Man beschreibt damit in der Regel eine neue Generation von Webanwendungen, die den Benutzer nicht mehr nur als Konsument, sondern auch als Produzent von Informationen ansehen - Wikis, Blogs, Social-Networking-Systeme und Tagging-Systeme sind einige Beispiele.

Ein Kritikpunkt am Begriff *Web 2.0* ist, dass zur Implementierung dieser Systeme keine grundsätzlich neuen Technologien entwickelt wurden, obwohl das Suffix *2.0* dies suggeriert. Tatsächlich sind die technologischen Grundlagen so alt wie das Web selbst. Neu ist lediglich die Idee, sie in einer Weise zu neuen Systemen zusammenzusetzen, die es den Benutzern ermöglicht aktiv mitzuwirken und Inhalte zu produzieren. Begünstigt wurde die Entwicklung dieser Systeme sicher auch durch den ständig wachsenden Anteil der Bevölkerung, der über einen schnellen Zugang zum Internet verfügt.

Im Rahmen dieser Arbeit werden die sogenannten Tagging-Systeme betrachtet. Ihre grundsätzliche Idee ist es, dass jeder Benutzer Ressourcen wie Hyperlinks oder Bilder mit beliebigen Begriffen, Tags genannt, annotieren kann. Ein Tag ist eine beliebige Zeichenkette. Die Tätigkeit des Zuweisens von Tags zu einer Ressource nennt man *Tagging*. Die entstandenen Daten werden online und öffentlich zugänglich im Web gespeichert.

Alle Tagging-Systeme stimmen in dieser grundsätzlichen Idee überein, unterscheiden sich aber in den Details. Marlow et al. [27] entwickelten eine Taxonomie zur Kategorisierung der verschiedenen Systeme. Hier sollen nur die wichtigsten der von ihnen aufgeführten Unterscheidungsmerkmale erwähnt werden:

Typ der Ressource: Beispiele für verschiedene Typen von Ressourcen sind URIs [4], Musik, Fotos und Publikationen.

Quelle der Ressource: Woher stammen die Ressourcen? Gehören sie jemandem? Werden sie vom System vorgegeben oder von den Benutzern in dasselbe importiert?

Tagging Rechte: Wer darf was taggen? In einigen Systemen dürfen die Benutzer alles, in anderen nur die eigenen Ressourcen taggen.

In dieser Diplomarbeit werden Tagging-Systeme eines bestimmten Typs behandelt. Erstens sind alle Ressourcen URIs, was keine große Einschränkung ist, da sich viele der anderen Ressource-Typen als URI darstellen lassen. Zweitens werden Ressourcen nicht vorgegeben und gehören auch niemandem und drittens darf jeder Benutzer alle Ressourcen taggen.

Im Folgenden sind einige aktuelle Tagging-Systeme aufgeführt. Die vollständige Liste findet sich in [27].

- Del.icio.us (`http://del.icio.us/`): Tagging von Web-Ressourcen.
- Bibsonomy (`http://bibsonomy.org/`) und CiteULike (`http://citeulike.org/`): Tagging von Web-Ressourcen und Publikationen
- Last.fm (`http://last.fm/`): Tagging von Künstlern, Alben und Songs (Musik)
- Flickr (`http://flickr.com/`) und YouTube (`http://youtube.com/`): Tagging der eigenen Fotos bzw. Videos
- Yahoo! Podcasts (`http://podcasts.yahoo.com/`): Tagging von Podcasts
- Technorati (`http://technorati.com/`): Tagging der eigenen Blog-Postings
- Upcoming (`http://upcoming.org/`): Tagging von Veranstaltungen (Konzerte, Ausstellungen,...)

U	T	R
User 1	Tag 1	Res 2
User 1	Tag 2	Res 3
User 2	Tag 3	Res 3
User 4	Tag 3	Res 3

Abbildung 2.1: Beispiel für eine Folksonomy. Es gibt vier Benutzer, drei Tags und drei Ressourcen. Die Tabelle stellt die Relation Y dar, die alle Taggings enthält.

Die Artikel [14, 16, 17, 28] bieten weiterführende Informationen zum Thema.

Um den Datenbestand eines Tagging-Systems formal zu beschreiben, wird eine Folksonomy verwendet. Das Wort Folksonomy ist ein Neologismus der Wörter *Folk* und *Taxonomy*, der Thomas Vander Wal [40] zugeschrieben wird. Die folgende, formale Definition stammt von Hotho et al. [34].

Definition 2.1. (Folksonomy)
Eine Folksonomy ist ein Tupel $\mathbb{F} := (U, T, R, Y)$.

- U, T und R sind endliche Mengen, die Benutzer, Tags und Ressourcen repräsentieren.

- Y ist eine Relation zwischen diesen Mengen, d.h. $Y \subseteq U \times T \times R$. Ein Tupel $(u, t, r) \in Y$ bedeutet, dass der Benutzer u der Ressource r das Tag t zugewiesen hat.

Hotho et al. fügen dem Tupel noch ein weiteres Element hinzu, welches benutzerdefinierte Beziehungen zwischen Tags modelliert. In dieser Arbeit werden aber nur die in der obigen Definition spezifizierten Elemente verwendet.

Ein Beispiel für eine Folksonomy ist in Abbildung 2.1 zu finden. Eine Folksonomy kann als tripartiter Graph dargestellt werden. Abbildung 2.2 zeigt den Graphen zu dem Beispiel.

Im Kontext einer Folksonomy gibt es mehrere Begriffe, die im Folgenden häufig verwendet werden:

Definition 2.2. (Tagging)
Gegeben sei eine Folksonomy $\mathbb{F} = (U, T, R, Y)$. Ein Tupel $(u, t, r) \in U \times T \times R$ bezeichnet man als ein Tagging.

Der Begriff *Tagging* beschreibt also die Zuweisung eines einzelnen Tags zu einer Ressource durch einen Benutzer.

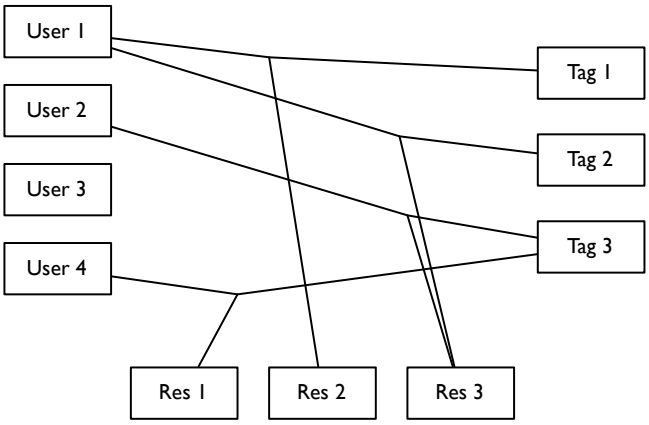


Abbildung 2.2: Die Folksonomy aus Abbildung 2.1 dargestellt in Form eines tripartiten Graphen. Jede Kante des Graphen setzt jeweils einen Benutzer, ein Tag und eine Ressource in Beziehung.

Definition 2.3. (Tagset)
Gegeben sei eine Folksonomy $\mathbb{F} := (U, T, R, Y)$. Eine Menge $T' \subseteq T$ wird als Tagset bezeichnet.

Wenn davon die Rede ist, dass ein Benutzer einer Ressource ein Tagset zugewiesen hat, ist damit gemeint, dass er der Ressource jedes Tag aus dem Tagset zugewiesen hat.

2.2 Warum Tagging-Systeme?

Ein Benutzer eines Tagging-Systems annotiert Ressourcen mit Tags, d.h. er zeichnet Daten mit zusätzlichen Daten aus, mit dem Ziel sie zu einem späteren Zeitpunkt leichter wiederfinden zu können.

Daten, die andere Daten beschreiben, nennt man *Metadaten*. Nach [32] unterscheidet man drei verschiedene Arten von Metadaten - strukturelle, administrative und deskriptive. Strukturelle Metadaten setzen Ressourcen untereinander in Relation, gruppieren beispielsweise mehrere Kapitel zu einem Buch. Administrative Metadaten dienen dem Management der Ressourcen, beispielsweise Zugriffsrechte oder Erstellungsdaten. Deskriptive Metadaten beschreiben Eigenschaften und Inhalt, mit dem Ziel die Wiederauffindbarkeit der Ressourcen zu verbessern.

Demnach sind Tags deskriptive Metadaten. Im Folgenden wird der Begriff Metadaten synonym zu *deskriptive Metadaten* verwendet.

Id	Category
000	Computer science, information and general works
100	Philosophy and psychology
200	Religion
300	Social sciences
400	Language
500	Science
600	Technology
700	Arts and recreation
800	Literatur
900	History and geography

Abbildung 2.3: Die Hauptkategorien des Dewey Decimal System, einem System zur Organisation des Inhalts von Bibliotheken.

2.2.1 Klassische Informationsstrukturierung

Schon vor den Zeiten des Web existierte die Notwendigkeit, Informationen zu organisieren. Bücher stellten die Hauptinformationsquelle dar. Es entstand mit den Bibliothekswissenschaften ein Wissenschaftszweig, der sich ausschließlich mit der Katalogisierung von Büchern beschäftigt. Bücher müssen innerhalb einer Bibliothek sowohl thematisch als auch räumlich angeordnet werden. Beispiele für Klassifikationsschemata sind das *Dewey Decimal Classification System* (DDC) [11], welches von öffentlichen Bibliotheken verwendet wird, und das *Library of Congress Classification Scheme* (LCC), das von den meisten amerikanischen Bibliotheken im Bereich von Forschung und Lehre eingesetzt wird. Die Hauptkategorien des DDC sind in Abbildung 2.3 dargestellt.

Kategorisierungsschemata sind baumförmig, d.h. ein Buch wird einem Oberthema und immer spezieller werdenden Unterthemen (einem Pfad von Themen) zugeordnet. Die Themen, die einem Buch zugeordnet werden, sollen einerseits den Inhalt des Buches möglichst genau charakterisieren, aber andererseits allgemein genug sein, um neben diesem auch noch für andere Bücher verwendet werden zu können. Die Themen einer Ebene sollen möglichst orthogonal zueinander sein, damit sich jedem Buch ein eindeutiger Themenpfad zuordnen lässt. Diese Forderung ist nicht allein Produkt einer platonischen Weltsicht, sondern eine Notwendigkeit, die sich aufgrund der einfachen Tatsache ergibt, dass ein Buch immer nur auf einem Regalbrett gleichzeitig stehen kann. Das speziellste Thema eines Buches muss also einem Ort innerhalb der Bibliothek zugeordnet werden. Verwandte Themen sollen zudem möglichst nah beieinander liegen. Abbildung 2.4 zeigt ein einfaches Beispiel für eine solche Anordnung.

Der Inhalt einer Bibliothek ist nicht statisch. Es müssen vielmehr ständig neue

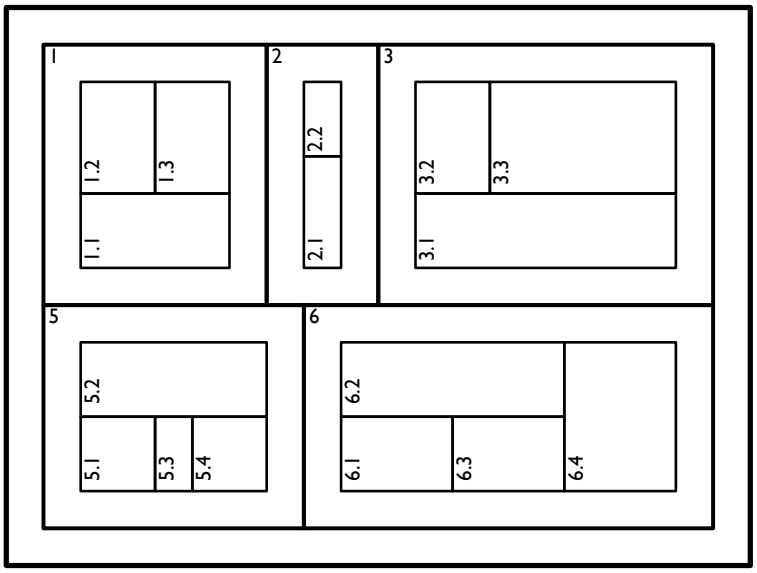


Abbildung 2.4: Räumliche Anordnung der Medien in einer Bibliothek. Die Zahlen 1 - 6 bezeichnen die Hauptkategorien. Darunter ist jeweils eine Ebene von Unterkategorien angeordnet.

Medien einsortiert werden. Das Katalogisierungsschema muss also nicht nur für die zum Erstellungszeitpunkt vorhandenen, sondern auch für alle in Zukunft erscheinenden Bücher gut geeignet sein.

Die Erstellung eines Katalogisierungsschemas stellt hohe Anforderungen an den Erstellenden. Ebenso erfordert seine Anwendung in der Praxis ein hohes Maß an Fachkenntnis. Es hat sich gezeigt, dass perfekte Katalogisierungsschemata kaum erreichbar sind.

2.2.2 Strukturierung von Web-Ressourcen

Lassen sich die Methoden der Bibliothekswissenschaften auf den Inhalt des Web anwenden? Können Web-Ressourcen in einem allumfassenden Katalog organisiert werden? Um diese Frage zu beantworten, müssen die Unterschiede zwischen Bibliotheken und dem Web betrachtet werden.

Der offensichtlichste Unterschied zwischen den beiden Anwendungsfeldern ist die Größe der zu organisierenden Information. Das Web enthält wesentlich mehr Ressourcen als eine Bibliothek. Eine Untersuchung von Gulli et al. [15] aus dem Jahr 2005 beziffert die Anzahl der Webseiten auf über 11 Milliarden. Im Gegensatz zu einer Bibliothek ist das Web dezentral organisiert. Neue Web-Ressourcen werden an vielen Orten gleichzeitig und ohne Kontrolle eingefügt. Für neue Medien in einer Bibliothek geschieht dies zentralisiert. Ein Buch in einer Bibliothek ist nicht mehr veränderbar.

Dagegen können Web-Ressourcen zu jeder Zeit verändert werden, ohne dass dies zentral bekannt gemacht wird. Es gibt sehr viele verschiedene Typen von Web-Ressourcen gegenüber wenigen Medientypen in einer Bibliothek. Der Typ einer Web-Ressource ist infolge der einheitlichen Darstellung als URI schwierig zu bestimmen.

Nach Betrachtung dieser Unterschiede ist offensichlich, dass die Katalogisierung des Web ähnlich einer Bibliothek nicht erreichbar ist. Die Vielfalt der Inhalte allein macht die Erstellung eines Katalogisierungsschemas immens schwierig; durch die Dezentralität und Dynamik des Webs ist die anschließende Wartung eines Katalogs unmöglich.

Im Folgenden werden drei weniger umfassende Methoden zur Strukturierung von Web-Ressourcen betrachtet. Als erstes werden Web-Verzeichnisse als ein Versuch vorgestellt, einen zentralisierten, durch eine Redaktion erstellten und gewarteten Katalog für einen ausgewählten Teil des Web anzubieten. Es folgt der Ansatz des Semantic Web, der die Autoren der Web-Ressourcen zur Strukturierung heranzieht. Beide Ansätze haben das Problem, dass der eigentliche Nutzer der Information am Strukturierungsprozess nicht beteiligt ist. Tagging-Systeme, der dritte vorgestellte Ansatz, stellen den Nutzer der Ressource in den Mittelpunkt und ermöglichen es ihm, die Strukturierung nach eigenen Vorstellungen vorzunehmen.

Web-Verzeichnisse

Unter dem Begriff Web-Verzeichnis versteht man ein redaktionell geführtes Verzeichnis von Links zu Web-Ressourcen. Es ist gemäß eines Katalogisierungsschemas organisiert, welches dem einer Bibliothek stark ähnelt. Bücher müssen räumlich angeordnet werden und daher genau einer Kategorie zugewiesen werden. Diese Anforderung besteht für Links nicht. Ungeachtet dessen wird ein Link in einem Web-Verzeichnis ebenso in nur einen Themenpfad einsortiert.

Prominente Beispiele für Web-Verzeichnisse sind das *Open Directory Project*[1] (siehe auch Abbildung 2.5) und das *Yahoo! Web Directory*[2].

Web-Verzeichnisse geben dem Benutzer durch das zugrunde liegende Katalogisierungsschema eine Sichtweise auf die Ressourcen vor. Im Folgenden wird diesbezüglich auch von einer *Sicht der Welt* gesprochen. Die Namen der einzelnen Kategorien werden genauso festgelegt wie die Granularität der Aufteilung und die hierarchische Anordnung. Die Sichtweise des Benutzers unterscheidet sich häufig stark von dieser Vorgabe. Er würde Kategorien anders benennen, würde sie je nach Interessantheit gröber oder feiner unterteilen, würde ihnen andere Ressourcen zuordnen und sie anders in Beziehung zueinander setzen.

Selbst wenn ein Benutzer sich auf die vorgegebene Sichtweise einstellen kann, ist er mit einem weiteren Problem von Web-Verzeichnissen konfrontiert, nämlich mit der

[1] `http://dmoz.org/`
[2] `http://dir.yahoo.com/`

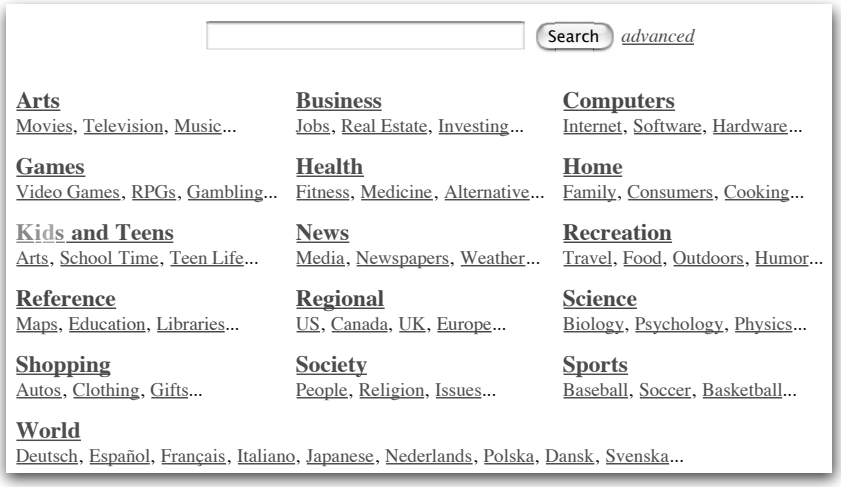


Abbildung 2.5: Die Webseite des Open Directory

Inkonsistenz der Einsortierung der Ressourcen. So ist beispielsweise beim *Open Directory Project* die Webseite des Informatik-Fachbereichs der Universität Dortmund in eine Kategorie[3] einsortiert, während die Webseiten sehr vieler anderer Informatik-Fachbereiche in einer anderen Kategorie[4] zu finden sind.

Die Redaktion eines Web-Verzeichnisses muss sowohl neu aufzunehmenden Ressourcen eine Kategorie zuweisen als auch bestehende Einträge überprüfen. Die Änderung der Kategorie einer Ressource kann notwendig werden, wenn sich ihr Inhalt ändert. Aufgrund der Größe eines Web-Verzeichnisses und der Schnelllebigkeit des Webs ist das Erfüllen dieser Aufgabe sehr schwierig.

Web-Verzeichnisse eignen sich nicht gut zur Strukturierung von Web-Ressourcen. Die Katalogisierungsschemata sind sehr kompliziert und verhindern, da sie eine Sicht der Welt vorgeben, einen effektiven Zugang des Benutzers zu den Ressourcen. Die Kategorisierung ist häufig inkonsistent, da auch die Redaktion nicht den Überblick behalten kann. Der Inhalt ist veraltet, da das Web zu groß und schnelllebig ist.

Eine ausführlichere Darstellung und Kritik zum Thema ist in [37] zu finden.

Semantic Web

Die Idee des Semantic Web ist es, Web-Ressourcen mit Elementen eines einheitlichen Vokabulars von Metadaten auszuzeichnen. So sollen Web-Inhalte durch Software-

[3] `Reference⇒Education⇒Colleges&Universities⇒Europe⇒German ⇒North-Rhine-Westphalia⇒University of Dortmund`

[4] `Computers⇒Computer Science⇒Academic Departments⇒Europe⇒Germany`

Agenten verarbeitbar gemacht werden. Berners-Lee et al. beschreiben in [5] beispielhaft die Planung von Arztbesuchen durch einen Software-Agenten. Dieser löst seine Aufgabe durch Verarbeitung spezieller Metadaten, mit denen die abgefragten Webseiten ausgezeichnet sind. Ein Arzt zeichnet seine Webseite beispielsweise mit Daten bezüglich *Ort* und *freien Terminen* aus. Es finden sich auch Webseiten, die *Bewertungen der Ärzte* durch ihre Patienten enthalten. Auf Basis dieser Daten gelingt es dem Software-Agenten, einen Besuch bei einem guten Arzt aus der näheren Umgebung zu einem passenden Zeitpunkt zu planen.

Ebenso wie ein Web-Verzeichnis gibt auch das Semantic Web eine Sicht der Welt vor. Für den Autor ist das Vokabular zur Beschreibung von Web-Ressourcen festgelegt, für den Nutzer die Art, in der Anfragen an den persönlichen Agenten gestellt werden müssen. Die Skalierungsprobleme der Web-Verzeichnisse, die sich aufgrund der Größe und Dynamik des Webs ergeben, werden dadurch gelöst, dass die Autoren, die ja die Urheber jeder Änderung oder Erweiterung sind, selbst für die Aktualisierung der Metainformationen verantwortlich sind. Wiederum aber ist der Nutzer vom Strukturierungsprozess ausgeschlossen.

Damit das Semantic Web funktioniert, muss sich ein großer Anteil der Autoren von Web-Inhalten beteiligen. Bisher ergibt sich aber noch kein direkter Nutzen, sondern nur zusätzlicher Aufwand. Die kritische Masse ist noch nicht erreicht. Berners-Lee et al. halten die Umsetzung aber für erreichbar [36], auch wenn dies bisher noch nicht geschehen ist.

Tagging-Systeme

Tagging-Systeme sind Systeme zur kollaborativen Kategorisierung von Ressourcen. Die Kategorien werden nicht vorgegeben, sondern von den Benutzern durch Vergabe von Tags erstellt. Ein Benutzer kann sein eigenes Vokabular verwenden und Kategorien nach seinen eigenen Vorstellungen zuweisen. Eine Ressource kann einer Themenkategorie (*software,politics,news,art*) ebenso zugeordnet werden, wie beispielsweise einer Kategorie (*toread,wishlist,myown*), die der Selbstorganisation des Benutzers dient. Allgemeine Kategorien sind möglich, aber auch spezielle. Eine Ressource kann mehr als einer Kategorie zugewiesen werden und Kategorien sind nicht hierarchisch angeordnet, sondern stehen nebeneinander in einem großen Namensraum.

Die Strukturierung von Informationen durch Tags ist nicht neu, sondern unter dem Begriff *keywording* schon lange bekannt. Ihr Vorteil ist die einfache Durchführbarkeit für den Benutzer, da keine komplizierten Kategorisierungsschemata gelernt und angewendet werden müssen. Neu ist die Verbindung der Tag-Vergabe und der Tag-Nutzung in einem engen Zyklus. Der Benutzer eines Tagging-Systems erhält ein direktes Feedback, sobald er eine Ressource getaggt hat. Er kann betrachten, welche Tags die Ressource von anderen Benutzern des Systems erhalten hat, oder welche anderen Ressourcen mit den von ihm verwendeten Tags ausgezeichnet wurden. Er hat also nicht

nur den direkten Nutzen, eine Ressource zu einem späteren Zeitpunkt wiederfinden zu können, sondern er kann auf das Wissen der anderen Benutzer Zugriff nehmen, um weitere interessante Ressourcen zu finden.

Da die Generierung von Metadaten, im Gegensatz zu Web-Verzeichnissen, nicht in der Verantwortung weniger Personen liegt, passen sich Tagging-Systeme gut an die Dynamik des Webs an. Es werden ständig neue Kategorien notwendig, andere Kategorien verschwinden. Ein statisches Kategorisierungsschema kann dem nicht gerecht werden, wohingegen in einem Tagging-System die Benutzer einfach ein passendes neues Tag erstellen oder ein anderes Tag nicht weiter verwenden.

Verglichen mit dem Ansatz des Semantic Web sind nicht die Autoren der Web-Ressourcen, sondern die Nutzer die Ersteller der Strukturierung. Diese Struktur entspricht daher den Vorstellungen und Zielen der Nutzer und nicht denen der Autoren.

Die Benutzer eines Tagging-Systems sind nahezu unbeschränkt in ihren Möglichkeiten zur Strukturierung. Dennoch bildet sich innerhalb einer Menge von Benutzern sehr häufig ein Konsens über die Verwendung von Tags und deren Zuordnung zu Ressourcen. Die Menge von Tags wächst also nicht unendlich, sondern es werden bereits vorhandene Tags weiterverwendet. Während bei Web-Verzeichnissen eine große Menge von Ressourcen dazu führt, dass das statische Kategorisierungsvokabular nicht mehr ausreicht und angepasst werden muss, führt ein starkes Wachstum der Ressourcen in einem Tagging-System gerade dazu, dass sich Gruppen von Benutzern mit einem ähnlichen Vokabular bilden. Es etablieren sich also mehrere koexistierende Sichten der Welt. Eine solche Sicht der Welt ist dynamisch in dem Sinne, dass sich das Vokabular ständig weiter anpasst. Auch halten sich Benutzer nicht strikt daran, sondern verwenden, falls gewünscht oder notwendig, eigene Tags.

Tagging-Systeme liefern zu jedem Tagging einer Ressource die Information, welche und wieviele Benutzer das Tag wann zugewiesen haben. Das ist für den Benutzer eine wertvolle Information, da er so die Bedeutsamkeit des Taggings und die Interessantheit der getaggten Ressource besser beurteilen kann [35]. Beispielsweise kann ein Tagging zu *news* uninteressant sein, wenn es schon vor einigen Wochen ins System eingefügt wurde. Weiterhin ist es für den Benutzer so möglich, die Taggings eines anderen Benutzers oder einer Gruppe von Benutzern zu verfolgen, wenn er die Erfahrung gemacht hat, dass diese meistens für ihn von Interesse waren.

Für die gezielte Informationssuche eignen sich Suchmaschinen am besten. Sucht man nicht eine Web-Ressource, sondern viele Ressourcen der gleichen Art - ist also nicht Suche, sondern Browsing das Ziel - so verwendet man besser ein Web-Verzeichnis oder ein Tagging-System. Web-Verzeichnisse sind für diesen Verwendungszweck aber deshalb ungeeignet, da der Benutzer eine vorgegebene Sicht der Welt annehmen muss. Tagging-Systeme dagegen eignen sich sehr gut zum Browsing, da sie diese Anforderung nicht stellen. Der Benutzer kann die Ressourcen betrachten, die andere Benutzer den von ihm verwendeten Tags zugeordnet haben. Dies funktioniert deshalb gut, da sich, wie zuvor beschrieben, Gruppen von Benutzern bilden.

Abbildung 2.6: Ausschnitt der Tag-Cloud der beliebtesten Tags auf DEL.ICIO.US

2.3 Navigation in aktuellen Tagging-Systemen

Einer der Gründe für die Beliebtheit von Tagging-Systemen ist die einfache Bedienbarkeit über einen Web-Browser. Im Folgenden wird der typische Ablauf der Bedienung am Beispiel des Tagging-Systems DEL.ICIO.US erläutert. Die Bedienung dieses Systems ist prototypisch für die meisten anderen Tagging-Systeme.

Eine Tag-Cloud ist eine Visualisierungsform für eine Menge von Tags. Es werden die n am häufigsten verwendeten Tags alphabetisch sortiert als ein Fließtext hintereinander geschrieben. Die Größe jedes einzelnen Tags ist dabei proportional zur Häufigkeit seiner Verwendung durch die Benutzer. In Abbildung 2.6 ist die Tag-Cloud der beliebtesten Tags von DEL.ICIO.US dargestellt.

Tag-Clouds sind häufig der Ausgangspunkt einer Navigation durch den Datenbestand. Der Benutzer wählt ein Tag aus, welches ihn besonders interessiert. In unserem Beispiel ist dies das Tag *photography*. Durch einen Klick auf den Link gelangt er zu einer neuen Seite, die in Abbildung 2.7 dargestellt ist. Die Seite enthält eine chronologisch absteigend sortierte Liste aller Ressourcen, die mit *photography* getaggt wurden. Nachdem er einige der Ressourcen betrachtet hat, entschließt sich der Benutzer im *Related Tags*-Bereich auf der rechten Seite ein neues Tag auszuwählen. Dort werden Tags aufgelistet, die einer Ressource häufig zusätzlich zu *photography* zugewiesen wurden. Er interessiert sich zwar für Ressourcen zum Thema *photography*, strebt aber etwas mehr in Richtung „Kunst“. Also wählt er das *art* Tag aus und gelangt durch einen Klick zur Übersichtsseite für dieses Tag. Nach diesem Schema lässt er sich durch den Datenbestand treiben und betrachtet das für ihn interessante Themengebiet aus verschiedenen Blickwinkeln.

Was leisten Tagging-Systeme momentan? Durch die Verwendung von Tag-Clouds zur Darstellung der Tags kann der Benutzer schnell beliebte von unbeliebten Tags

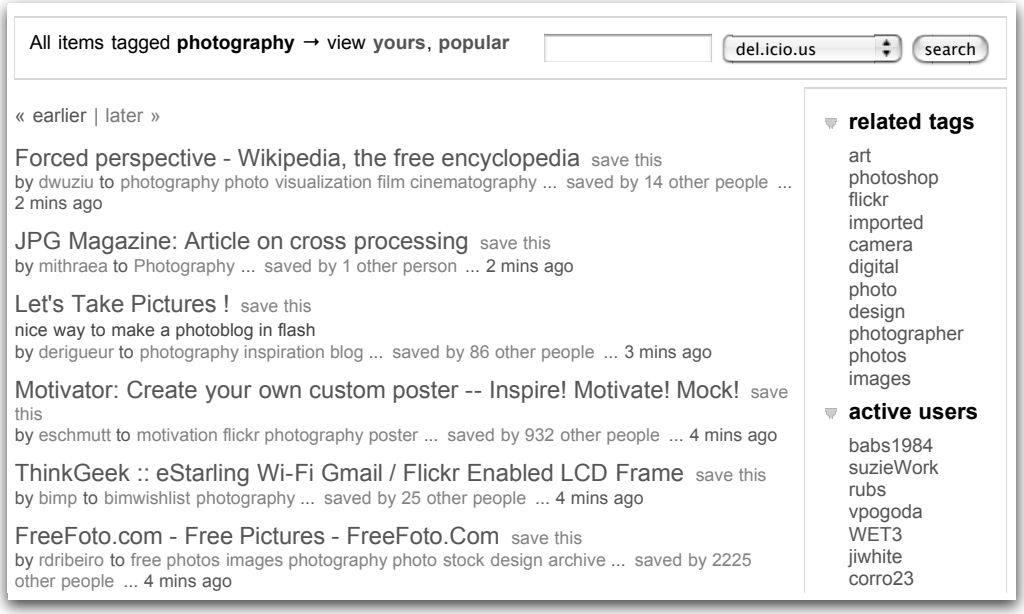


Abbildung 2.7: Seite aller Ressourcen zum Tag *photography* auf DEL.ICIO.US

unterscheiden. Durch die Darstellung verwandter Tags kann er zu Tags mit ähnlichen Ressourcen springen. Insgesamt ist durch das *Alles ist ein Link*-Konzept eine sehr flüssige und intuitive Navigation möglich.

Die Extension eines Tags, d.h. die Menge der Ressourcen, die mit diesem Tag ausgezeichnet wurden, lässt sich jeweils nur einzeln betrachten. Der Benutzer springt von Extension zu Extension (siehe Abbildung 2.8a und 2.8b), ohne eine Information darüber zu erhalten, wie die Extensionen in Beziehung zueinander stehen. In unserem Beispiel wollte der Benutzer Ressourcen zum Tag *photography* betrachten, welche sich aber auch mit dem Thema *art* befassen. Es wäre für ihn also nützlich gewesen, hätte er nur die Ressourcen betrachten können, die mit beiden Tags gleichzeitig ausgezeichnet sind. Das Tagging-System ermöglichte es ihm aber nur, die beiden Tags unabhängig voneinander zu betrachten.

Betrachtet man zwei Tag-Extensionen, so sind verschiedene Beziehungen möglich. Diese sind in Abbildung 2.9 dargestellt. Entweder sind die Extensionen (a) disjunkt, (b) sie überlappen sich teilweise oder (c) die eine Extension ist Teilmenge der anderen. Im Fall der Related Tags kann (a) nicht gelten, sondern nur (b) oder (c).

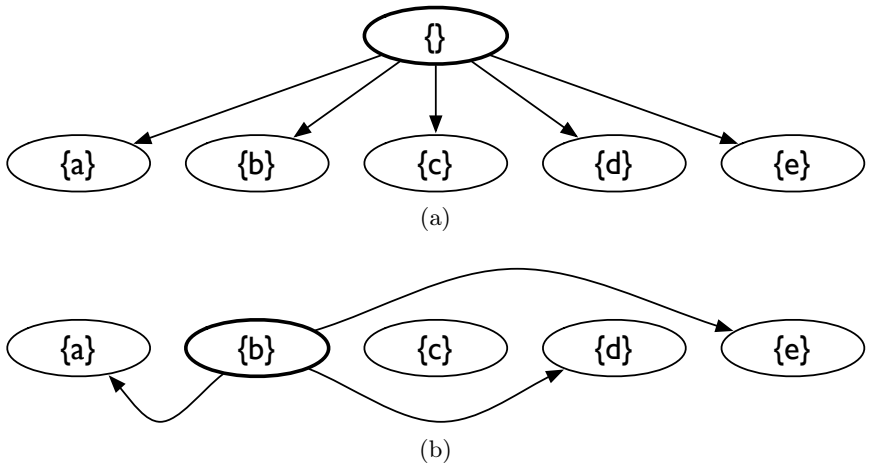


Abbildung 2.8: Navigationsmöglichkeiten in heutigen Tagging-Systemen:
(a) Ein Benutzer beginnt das Browsing durch Betrachten aller Tags des Tagging-Systems. Selten verwendete Tags werden von Tagging-Systemen in dieser Sicht ausgeblendet.
(b) In $\{b\}$ angekommen, setzt der Benutzer seine Navigation über die *Related Tags*-Links fort.

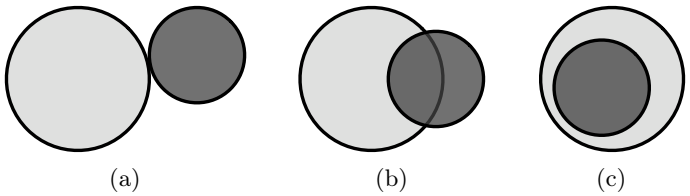


Abbildung 2.9: Die möglichen Beziehungen zweier Tag-Extensionen. Die Extensionen können (a) disjunkt sein, (b) sich teilweise überlappen und (c) eine Extension kann die andere Extension enthalten.

2.4 Tag-Clustering und Visualisierung

Das Fehlen einer expliziten Darstellung der Beziehungen zwischen Tag-Extensionen macht es dem Benutzer zum einen schwierig, einen Überblick über den Datenbestand eines Tagging-Systems zu gewinnen und zum anderen verhindert es ein zielgerichtetes Browsing. Auch wird eine Tag-Cloud meistens von einer Menge sehr ähnlicher Tags dominiert, die den Datenbestand schlecht charakterisieren [3]. Es wurden mehrere Verfahren entwickelt, um diese Probleme zu lösen.

Hassan-Montero et al. [19] verwenden eine Bewertungsfunktion ähnlich zu TF/IDF, um aus der Gesamtmenge aller Tags die Tags mit großer Unterscheidungskraft auszuwählen. Anschließend clustern sie die ausgewählten Tags mit dem Verfahren Bisecting KMeans unter Verwendung der Jaccard Similarity als Metrik.

Schmitz et al. [34] stellen mehrere Möglichkeiten zur Projektion einer Folksonomy auf eine für einen FIM-Algorithmus geeignete Eingabeform vor. Auf Basis der bestimmten Frequent Tagsets werden Assoziationsregeln erstellt und als Graph visualisiert. Auf diese Weise werden Beziehungen zwischen Tags, aber auch zwischen Benutzern oder Ressourcen graphisch dargestellt.

Begelman et al. [3] definieren mit der Menge der Tags als Knoten einen gewichteten Graph. Das Gewicht einer Kante zwischen einem Tag t_i und einem Tag t_j ist die Häufigkeit ihrer gemeinsamen Verwendung. Es wird ein Kriterium definiert, welches für je zwei Tags angibt, ob die Kante tatsächlich in den Graphen aufgenommen wird. Auf den erstellten Graph wird das Graph-Clustering-Verfahren Spectral Bisection angewandt und so die Menge der Tags geclustert.

Kaser et al. [24] stellen ein Verfahren zur Verbesserung der geometrischen Anordnung der Tags in der Tag-Cloud vor. Ähnliche Tags sollen in der Tag-Cloud nah beeinander liegen. Ebenso wie bei Begelman et al. wird ein gewichteter Graph aufgebaut. Der Graph wird an der Stelle seines minimalen Schnitts zerteilt. Die Knoten des ersten Teilgraphen werden links, die des zweiten Teilgraphen rechts angeordnet. Nun zerteilt man entsprechend beide Teilgraphen und ordnet die Knoten der Teilgraphen oben bzw. unten an. Dieser Vorgang wird rekursiv solange durchgeführt, bis die Graphen vollständig zerlegt sind. Für jeden Knoten bzw. jedes Tag hat sich so eine Position in der Tag-Cloud ergeben. Beim abschließenden Zeichnen der Tag-Cloud ist die Größe der Darstellung eines Tags wie üblich proportional zu seiner Häufigkeit.

Alle vorgestellten Verfahren machen die Beziehungen zwischen den Tags transparenter, tun dies jedoch, indem sie implizite, dem Benutzer nicht bekannte Kriterien optimieren. Dies führt dazu, dass der Benutzer das erzeugte Ergebnis schwer interpretieren kann. Ebenso hat er keine Möglichkeit, den Erzeugungsprozess zu beeinflussen und seine individuellen Vorstellungen umzusetzen, da stets nur eine einzelne Lösung erzeugt wird. Die innerhalb der Verfahren häufig verwendeten Heuristiken kann man als versteckte Parameter interpretieren. An dieser Stelle bietet sich dem Benutzer dann doch die Möglichkeit, den Erzeugungsprozess zu beeinflussen. Die Erforschung mögli-

cher alternativer Lösungen muss dann allerdings als mühsamer und unintuitiver *Trial and Error*-Prozess betrieben werden.

Es soll daher ein neues Verfahren entwickelt werden, welches, ebenso wie die soeben vorgestellten Verfahren, die inhärent im Datenbestand eines Tagging-Systems enthaltene Information über die Beziehung der Tag-Extensionen explizit macht, dieses Ziel aber auf eine Weise erreicht, die es dem Benutzer gestattet zum einen zu verstehen, unter Verwendung welcher Kriterien das Ergebnis erzeugt wurde, und zum anderen Einfluss zu nehmen, um seine individuellen Vorstellungen umzusetzen. Dabei soll verstärkt Wert auf eine hierarchische Anordnung von Tags gelegt werden, da dies für die Navigation, wie im nachfolgenden Abschnitt beschrieben, sehr nützlich ist. Existierende Verfahren bleiben sehr nahe am Konzept der Tag-Cloud und erzeugen nur sehr eingeschränkte Hierarchien. Tags werden in der Regel gruppiert, was als Baum der Tiefe 1 interpretiert werden kann.

2.5 Erweiterte Navigationsmöglichkeiten durch hierarchische Strukturen

Um die Beziehungen zwischen den Extensionen verschiedener Tags darzustellen, betrachtet man zusätzlich Tagsets. Diese lassen sich auf natürliche Weise hierarchisch anordnen. Jedem Tagset werden alle Ressourcen zugewiesen, die damit getaggt wurden. Eine Ressource, die Tagset $\{a, b\}$ zugewiesen ist, muss damit automatisch auch $\{a\}$ und $\{b\}$ zugewiesen sein. Ein Beispiel für die entstehende Struktur ist in Abbildung 2.10 zu sehen.

Der Benutzer startet in Tagset $\{\}$ und sieht die Links zu den Tags $\{a\}$, $\{b\}$ und $\{c\}$. Wählt er nun das Tag $\{a\}$ aus, gelangt er zur Übersichtsseite für dieses Tag und kann die entsprechenden Ressourcen betrachten. An diesem Punkt bietet ihm das klassische Tagging-System durch *Related Tags* die Möglichkeit zum Tag $\{b\}$ zu springen. Die erweiterte Variante ermöglicht es ihm nun zusätzlich, alle Ressourcen zu betrachten, die mit a und b getaggt sind. Der Benutzer entscheidet sich für das Tagset $\{a, b\}$ und springt über den Link dorthin. Hier kann er entweder zusätzlich das Tag c wählen, gelangt also zu $\{a, b, c\}$, oder eines der beiden momentan betrachteten Tags wieder aus seiner Betrachtung entfernen und so entweder zu $\{a\}$ oder zu $\{b\}$ navigieren.

Durch die Verwendung einer Menge hierarchisch angeordneter Tagset als Navigationsstruktur gewinnt der Benutzer einen Freiheitsgrad in der Wahl seiner Richtung hinzu. Während er in der klassischen Variante eines Tagging-Systems nur *horizontal* von Tag zu Tag springen kann (siehe Abbildungen 2.8a und 2.8b), bietet sich ihm so zusätzlich die Möglichkeit, *vertikal* die nächst größeren oder kleineren Tagsets zu erreichen.

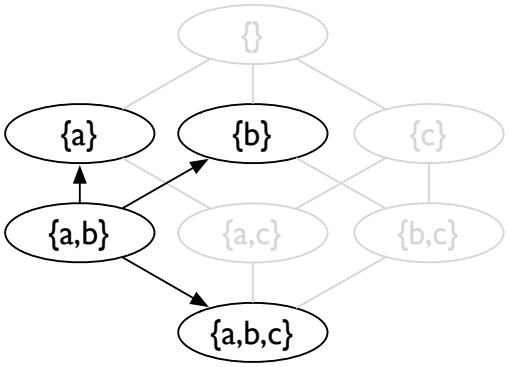


Abbildung 2.10: Tagsets als hierarchische Navigationsstruktur. Der Benutzer besucht gerade die Seite zu $\{a, b\}$. Die Pfeile kennzeichnen die Alternativen für seine nächsten Navigationsschritt.

2.6 Ziel der Arbeit

Die hierarchische Anordnung von Tagsets vermittelt dem Benutzer, wie im letzten Abschnitt beschrieben, Informationen über die Beziehungen zwischen den Tags, die für ein zielgerichtetes Browsing durch den Datenbestand sehr wertvoll sind. Eine triviale Möglichkeit zur Erstellung einer solchen Struktur ist die hierarchische Anordnung aller vorhandenen Tagsets und die anschließende Zuordnung der Ressourcen entsprechend der vergebenen Tags. Die resultierende Struktur ist aber mit großer Wahrscheinlichkeit unbenutzbar, da typische Tagging-Systeme mehrere tausend Tags enthalten. Da zusätzlich noch alle verwendeten Tagsets betrachtet werden, ergibt sich eine sehr komplizierte Struktur.

Ziel der Arbeit soll es sein, Methoden zu entwickeln, die eine Erstellung übersichtlicher, auf die Bedürfnisse des Benutzers angepasster Navigationsstrukturen ermöglichen. Dazu ist es zunächst interessant zu betrachten, was eine gute Navigationsstruktur ausmacht. Der Benutzer möchte möglichst viele Ressourcen aus dem Datenbestand des Tagging-Systems finden können. Dadurch minimiert er die Wahrscheinlichkeit, eine für ihn interessante Ressource zu verpassen, weil sie nicht in der Struktur enthalten ist. Wenn Ressourcen nicht in der Struktur repräsentiert werden, dann sollen es eher die unbeliebten - das sind Ressourcen, die von wenigen Benutzern getaggt wurden - als die beliebten sein.

Gleichzeitig möchte er, egal welches Tagset er gerade besucht, eine möglichst übersichtliche Anzahl von Wahlmöglichkeiten für seinen nächsten Navigationsschritt erhalten (schmale Struktur). Um den Bereich der gesuchten Ressourcen so präzise wie möglich eingrenzen zu können, möchte er möglichst viele Tags gleichzeitig in seine

Betrachtung einbeziehen können (tiefe Struktur).

Es stellt sich also die Aufgabe, eine Auswahl von Tagsets zu finden, die diese Eigenschaften haben. Es ist einfach, eine einzelne Eigenschaft zu garantieren. Um möglichst viele Ressourcen abzudecken, müssen alle Tags ausgewählt werden. Eine schmale Struktur erhält man, wenn man gegenteilig handelt und sehr wenige Tagsets auswählt. Eine Struktur zu finden, die alle Eigenschaften in sich vereint, ist hingegen wesentlich schwieriger. Fordert man viele Ressourcen, so führt das zur Auswahl von mehr Tagsets. Dies widerspricht aber dem Ziel der Übersichtlichkeit. Große Übersichtlichkeit wiederum widerspricht der Forderung nach Vollständigkeit.

Es kann keine Auswahl von Tagsets geben, die alle Eigenschaften in perfekter Weise erfüllt, da das bessere Erfüllen einer Eigenschaft zum schlechteren Erfüllen einer anderen führt. Um trotzdem ein einzelnes Ergebnis zu erhalten, müssen den Eigenschaften Gewichtungen zugeordnet werden, mit denen der Benutzer dokumentieren kann, dass es beispielsweise weniger wichtig ist, dass alle Ressourcen abgedeckt sind, wenn dafür die Struktur einfach bleibt. Eine solche Zuordnung von Gewichten ist aber in der Praxis schwierig, da unbekannt ist, wie genau sich deren Änderungen auf die erzeugte Struktur auswirken. Auch lässt sich eine empfundene Wichtigkeit schwierig als Wert auf einer Skala ausdrücken.

Anstatt durch Gewichtung ein Einzelergebnis zu erzwingen, erstellt man eine Menge von Alternativen - von übersichtlich und unvollständig bis unübersichtlich und vollständig. Der Benutzer hat dann die Möglichkeit, nach seinen Wünschen eine Alternative auszuwählen, ohne eine Gewichtung der Eigenschaften vorgeben zu müssen. Möchte er anschließend von seiner aktuellen Auswahl ausgehend versuchen, eine Struktur zu finden, deren Übersichtlichkeit ähnlich, deren Abdeckung aber größer ist, so durchsucht er die Alternativenmenge in diese Richtung. Auf diese Weise kann der Benutzer, wie in Abschnitt 2.4 gefordert, Einfluss nehmen und ein Ergebnis nach seinen Vorstellungen auswählen. Durch die explizite Darstellung der einzelnen Kriterien kann er die vom Verfahren erzeugten Lösungen interpretieren und die Auswirkung der Änderung des Wertes eines Kriteriums nachvollziehen. Die Erzeugung einer Menge alternativer Navigationsstrukturen lässt sich als multikriterielles Optimierungsproblem formulieren. Die Grundlagen der multikriteriellen Optimierung werden in Kapitel 3.5 vorgestellt. In Kapitel 4 folgt die konkrete Formulierung des Problems.

Die bisher genannten Anforderungen sollen die Grundversion des zu entwickelnden Verfahrens bilden. Weiterhin soll der sich ständig ändernde Datenbestand eines Tagging-Systems berücksichtigt werden. Diese Anforderung soll als Erweiterung der Grundversion des Verfahrens umgesetzt werden.

Alle Ziele der Arbeit im Überblick sind auf der folgenden Seite zusammengestellt.

Ziele der Arbeit:

1. Formale Definition eines Modells der hierarchischen Navigationsstrukturen für Tagging-Systeme
2. Definition des Findens von Mengen alternativer Navigationsstrukturen als Suchproblem
3. Definition von Bewertungsfunktionen für Navigationsstrukturen, die die Richtung der Suche steuern. Es müssen Funktionen für Abdeckung, Übersichtlichkeit und Tiefe formuliert werden.
4. Empirische Analyse der Beziehungen zwischen diesen Bewertungsfunktionen
5. Anpassung eines multikriteriellen Optimierungsverfahren zur Lösung des Suchproblems
6. Erweiterung des Verfahrens auf zeitlich veränderliche Daten (Inkrementelles Verfahren)
7. Anwendung des Verfahrens auf den Datenbestand eines Tagging-Systems und Auswertung der Ergebnisse

Kapitel 3

Grundlagen

Dieses Kapitel stellt alle Verfahren vor, die entweder als Quelle der Inspiration oder als Bestandteil zu dem in dieser Arbeit entwickelten Verfahren beigetragen haben. In Abschnitt 3.1 wird zunächst eine allgemeine Einführung in übliche Verfahren zum hierarchischen Clustering von Textdokumenten gegeben. Es folgt in Abschnitt 3.2 eine Einführung in die Grundbegriffe des Frequent Itemset Mining (FIM). Als konkreter Algorithmus zur Lösung eines FIM-Problems wird Fp-Growth *vorgestellt. Dieser Algorithmus wird in Kapitel 4.2 als Teil des Verfahrens eingesetzt, um eine Menge häufiger Tagsets auszuwählen. In Abschnitt 3.3 wird ein Frequent Itemset-basiertes Clustering-Verfahren vorgestellt, welches bereits einigen Erfordernissen des Clusterings von Tags gerecht wird.*

Danach schließt sich in Abschnitt 3.4 ein Überblick über die grundliegende Funktionsweise eines genetischen Algorithmus (GA) an. Ein spezieller GA ist die zentrale Komponente für die Suche nach Navigationsstrukturen. Die Auswahl von geeigneten Navigationsstrukturen wird in Kapitel 4.4 als ein multikriterielles Optimierungsproblem formuliert. Einen Überblick über die Grundbegriffe der multikriteriellen Optimierung bietet der Abschnitt 3.5.

3.1 Hierarchisches Clustering von Textdokumenten

Wie im letzten Kapitel definiert, sollen Navigationsstrukturen für Tagging-Systeme erstellt werden, die dem Benutzer die in einem Tagging-System gespeicherten Ressourcen in strukturierter Weise zugänglich machen. Zur Exploration großer, unstrukturierter Mengen werden häufig Clustering-Verfahren eingesetzt. Einen Teilbereich dieser Verfahren bilden die Verfahren zum Clustering von Textdokumenten, deren Aufgabe es ist, Textdokumente derart in Cluster zu gruppieren, dass Dokumente innerhalb eines Clusters sehr ähnlich zueinander sind, während Dokumente verschiedener Cluster sich möglichst stark voneinander unterscheiden. Dabei werden zwei Dokumente als ähnlich angesehen, wenn sie ein ähnliches Vokabular benutzen. Es handelt sich um ein Verfahren aus dem Bereich des unüberwachten Lernens. Dies bedeutet, dass außer den Dokumenten selbst keine weitere externe Information vorhanden ist. Die Cluster werden in einer hierarchischen Struktur angeordnet, wobei eine Elter-Kind-Beziehung zwischen zwei Clustern bedeutet, dass das Kind-Cluster ein Teilgebiet des Themenge-

biets des Elter-Clusters ist.

Die Aufgabenstellung des Clusterns von Ressourcen auf der Basis von Tags lässt sich auf natürliche Weise in die Problemdomäne des Clusterings von Textdokumenten abbilden, indem man Ressourcen als Dokumente und Tags als darin vorkommende Wörter betrachtet. Im Folgenden soll eine kurze Einführung in den Bereich des Clusterings gegeben werden, um anschließend zu überprüfen, inwieweit die üblichen Verfahren bei geeigneter Umformulierung der Problemstellung tatsächlich einsetzbar sind. Eine detaillierte Einführung in den Bereich der Clustering-Verfahren bieten auch Jain et al. [23].

3.1.1 Darstellung der Dokumente

Das *Vector Space Model* (oder auch *Term Vector Model*) wurde von Salton [33] entwickelt und wird in den Bereichen *Information Filtering* und *Information Retrieval*, vor allem aber auch im Bereich des Clusterings eingesetzt. Jedes Textdokument wird als Vektor in einem mehrdimensionalen, linearen Raum dargestellt, in dem für jedes Element einer vordefinierten Menge von Wörtern (auch *Terms* genannt) eine Dimension existiert.

In einer vereinfachten Version des *Vector Space Model*, dem sogenannten *Term Count Model*, enthält ein Vektor (auch Wortvektor genannt) die absoluten Häufigkeiten, mit denen jedes der vordefinierten Wörter in dem jeweils repräsentierten Textdokument vorkommt. Um Vektoren zu erhalten, die sich gut zur Verwendung in einem Clustering-Verfahren eignen, wird im *Vector Space Model* für jedes Wort nicht nur seine *lokale* Häufigkeit in einem einzelnen Textdokument, sondern zusätzlich seine *globale* Häufigkeit in allen Dokumenten betrachtet. Jede Stelle eines Wortvektors wird daher mit einer Gewichtung versehen:

Sei $\vec{d}$ ein Wortvektor und $\vec{d_i}$ die i-te Stelle des Vektors. Dann gilt:

$$\vec{d_i} = tf_i \cdot log(\frac{n}{df_i}) \tag{3.1}$$

Hierbei ist tf_i die Anzahl der Vorkommen des Wortes i im entsprechenden Textdokument, df_i die Anzahl der Textdokumente, die das Wort i enthalten, und n die Gesamtanzahl der Textdokumente. Der Gewichtsfaktor $log(\frac{n}{df_i})$ wird als *Inverse Document Frequency* (IDF) bezeichnet und sorgt dafür, dass Wörter, die in sehr vielen Dokumenten vorkommen und daher schlecht dazu geeignet sind, Dokumente zu trennen, eine geringe Gewichtung erhalten. Dadurch wird gewährleistet, dass Wörter dieser Art den Clustering-Prozess nicht behindern.

Die Ähnlichkeit zweier Dokumente wird häufig wie folgt definiert:

$$\cos(\vec{d'}, \vec{d''}) = \frac{(\vec{d'} \cdot \vec{d''})}{\|\vec{d'}\|\|\vec{d''}\|} \tag{3.2}$$

wobei $\cdot$ das Skalarprodukt und $\|\vec{d}\|$ die Länge des Vektors $\vec{d}$ ist. Die durchgeführte Normierung der Längen der Vektoren ermöglicht das Vergleichen von Vektoren mit stark unterschiedlichen Worthäufigkeiten. Solche Vektoren treten häufig dann auf, wenn die betrachteten Textdokumente eine sehr unterschiedliche Länge haben. Man spricht bei diesem Ähnlichkeitsmaß auch vom *Cosinus-Maß*.

Im Folgenden werden klassische Clustering-Verfahren vorgestellt, die für den Bereich des Clusterings von Textdokumenten anwendbar sind. Eine ausführliche Evaluation der Vor- und Nachteile dieser Verfahren speziell bei Anwendung in diesem Bereich liefern Steinbach et al. [38].

3.1.2 Hierarchische Verfahren

Hierarchische Verfahren teilen sich in agglomerative und divisive Verfahren. Gegeben seien n verschiedene Dokumente, dann startet ein agglomeratives Clustering-Verfahren mit einer Menge von n einelementigen Clustern. Man bestimmt das ähnlichste Paar von Clustern und verschmilzt es zu einem zweielementigen Cluster. Diesen Schritt führt man solange aus, bis nur noch ein großes, alle Dokumente enthaltendes Cluster vorhanden ist. In der erstellten Cluster-Hierarchie ist jedes n-elementige Cluster der Elter derjenigen zwei $n-1$-elementigen Cluster, aus denen es durch Verschmelzung entstanden ist. Verschiedene Varianten dieses Verfahrens unterscheiden sich im Wesentlichen in der Definition des Cluster-Ähnlichkeitsmaßes.

Ein divisives Clustering-Verfahren startet mit einem Cluster, welches alle n Dokumente enthält. In jedem Schritt wird eines der Cluster ausgewählt und in zwei Cluster zerteilt. Dies wird solange wiederholt, bis ein Abbruchkriterium erfüllt ist oder nur noch einelementige Cluster vorhanden sind. Varianten dieses Verfahrens unterscheiden sich in der Auswahl des Clusters, welches in einem Schritt zerteilt wird und darin, wie es zerteilt wird.

3.1.3 Partitionierende Verfahren

Partitionierende Verfahren erstellen eine Menge von Clustern, die eine Partition der Menge aller Dokumente bildet. Das Verfahren KMeans erstellt eine Menge von k Clustern, wobei der Parameter k vom Anwender vor Start des Verfahrens festgelegt wird. Ein Cluster-Centroid ist ein Vektor im Raum der Worthäufigkeiten. Als erstes werden Centroide $C_1, ..., C_k$ zufällig im Raum positioniert. Anschließend ordnet man alle Dokumente jeweils dem passendsten Centroid zu. Die Ähnlichkeit eines Dokuments zu einem Centroid wird, genauso wie die Ähnlichkeit zweier Dokumente, über das Cosinus-Maß bestimmt. Zu jedem Centroid entsteht auf diese Weise eine Menge S_i von Dokumenten. Nun werden alle Centroide C_i neu bestimmt.

$$C_i = \frac{1}{|S_i|} \sum_{\vec{d} \in S_i} d \tag{3.3}$$

Dokumentzuordnung und Centroid-Neubestimmung werden solange wiederholt, bis sich die Centroide nicht mehr verändern. Die aktuellen Mengen S_i werden als Ergebnis-Cluster ausgegeben.

KMeans wurde zu einem hierarchischen Verfahren, Bisecting KMeans genannt, erweitert. Es beginnt wie ein divisives Verfahren mit einem Cluster, das alle Dokumente enthält. Dieses Cluster wird durch Anwendung von KMeans (mit $k = 2$) in zwei Cluster zerteilt. Dieser Schritt wird n-mal durchgeführt. Aus den entstanden n alternativen Splits wird derjenige mit der höchsten Gesamtgüte übernommen. Die Güte eines Clusters ist definiert als:

$$\frac{1}{|S|^2} \sum_{\substack{\vec{d} \in S \\ \vec{d}' \in S}} \cos(\vec{d}', \vec{d}) \tag{3.4}$$

Es wird solange ein Cluster nach obiger Strategie ausgewählt und zerteilt, bis eine Hierarchie von k Clustern erstellt worden ist.

3.1.4 Eignung der Verfahren zum Tag-Clustering

Wie bereits zu Beginn dieses Kapitels erläutert, lässt sich die Aufgabe des Clusterings von Ressourcen anhand von Tags so umformulieren, dass ein Clustering-Verfahren für Textdokumente zur Lösung verwendet werden kann.

Die vorgestellten Verfahren sind aus verschiedenen Gründen (siehe auch Ester et al. [12]) für das Clustering von Textdokumenten ungeeignet. Die durch das Verfahren zu verarbeitenden Räume sind sehr hochdimensional und es ist sehr schwierig, Cluster in einem solchen Raum zu finden, da die Cluster häufig nur in Teilräumen existieren. Die erzeugten Cluster haben keine Beschreibung, was ihre Interpretation für den Benutzer erschwert oder sogar verhindert. Desweiteren ist es notwendig, Parametereinstellungen vorzunehmen. Die Auswirkung der Änderung eines Parameters ist vom Benutzer in der Regel schwierig vorherzusehen. Häufig hängt die Güte des Ergebnisses sehr stark davon ab, welche Parametereinstellungen verwendet wurden. Versucht der Benutzer durch verschiedene Parametereinstellungen, eine für ihn geeignete Lösung zu finden, sind mühsame Versuche in einem *Trial and Error*-Prozess notwendig. Schließlich skalieren die Verfahren mit steigender Problemgröße schlecht, was gerade im Bereich des Clusterings von Textdokumenten ein Problem ist, da hier in der Regel sehr viele, hochdimensionale Wortvektoren verarbeitet werden müssen.

Die Probleme, die klassische Clustering-Verfahren im Bereich des Clusterings von Dokumenten aufweisen, treten beim Clustering von Tags verstärkt auf. Tagging-Daten weisen häufig eine noch höhere Dimensionalität auf und gleichzeitig kommen viele Tags nur in sehr wenigen Ressourcen vor; die Wortvektoren in der umformulierten Problemstellung sind also sehr spärlich besetzt. Die in Kapitel 2.4 bereits vorgestellten Clustering-Verfahren für Tagging-Daten leiden ebenfalls an diesen Problemen und

wurden aus diesen Gründen von ihren Autoren jeweils nur auf relativ kleine Datensätze angewandt.

Aus den gegebenen Gründen wurde eine neue Klasse von Clustering-Verfahren speziell für den Bereich des Clusterings von Textdokumenten entwickelt. Diese Verfahren verwenden ein Frequent-Itemset-Mining-Verfahren zur Vorverarbeitung der Eingabedaten. In Kapitel 3.3 wird das Verfahren von Ester et al. [12] vorstellt und untersucht, ob es sich zum Clustering von Tags einsetzen lässt. Zuvor sollen jedoch die Grundlagen des Frequent-Itemset-Mining vorgestellt werden.

3.2 Frequent Itemset Mining

Im Bereich des Frequent Itemset Mining (FIM) beschäftigt man sich mit dem Finden wiederkehrender Muster in Datenbanken. Dieses Problem entstand mit der großflächigen Einführung von Barcode-Scannern in Supermärkten. Dies erschloss den Betreibern der Supermärkte eine riesige Datenquelle. Für jeden Einkauf (Transaktion) konnte nun abgespeichert werden, welche Gegenstände gemeinsam gekauft wurden. Um als Erheber von diesen riesigen Datenbeständen zu profitieren, mussten spezielle Methoden entwickelt werden. Diese mussten aufgrund der Problemgröße sehr effizient sein. Die so gewonnenen Daten erwiesen sich als sehr nützlich zur Erforschung des Kaufverhaltens der Kunden und daher werden FIM-Verfahren seitdem mit großem Erfolg zur Preiskalkulation, zur Gestaltung von Geschäften und Katalogen, und in vielen anderen Bereichen genutzt.

3.2.1 Itemsets

Gegeben sei eine Menge $I = \{i_1, \ldots, i_n\}$. Jedes $i_j \in I$ bezeichnet einen Gegenstand (*Item*). Im Folgenden wird die englische Ausdrucksweise verwendet. Jede Teilmenge $I' \subseteq I$ von Items heißt *Itemset*. Wenn $|I'| = k$, so wird I' auch als *k-Itemset* bezeichnet.

Eine *Transaktion* über I ist ein Tupel (tid, I'), wobei tid ein eindeutiger Transaktionsidentifikator ist. Wenn für eine Transaktion (tid, I') und ein Itemset I'' gilt, dass $I'' \subseteq I'$, so spricht man davon, dass die Transaktion das Itemset I'' enthält. Eine *Transaktionsdatenbank* D über I ist eine Menge von Transaktionen über I.

Die Abdeckung (*Cover*) eines Itemsets M ist definiert als die Menge der Identifikatoren der Transaktionen, die das Itemset enthalten:

$$\text{cover}(M, \mathsf{D}) := \{tid \mid (tid, I') \in \mathsf{D}, M \subseteq I'\} \tag{3.5}$$

Die absolute Häufigkeit (*Support*) eines Itemsets M ist definiert als die Größe seiner Abdeckung.

$$\text{supp}(M, \mathsf{D}) := |\,\text{cover}(M, \mathsf{D})| \tag{3.6}$$

Die relative Häufigkeit (*Frequency*) eines Itemsets M ist die Wahrscheinlichkeit, dass M in einer Transaktion aus D enthalten ist.

$$\text{freq}(M, \mathsf{D}) := P(M) = \frac{\text{supp}(M, \mathsf{D})}{|\mathsf{D}|} \tag{3.7}$$

Ist die Transaktionsdatenbank bekannt, so schreibt man alle drei Funktionen statt nach dem Schema $f(M, \mathsf{D})$ einfach als $f(M)$.

Ein Itemset wird als *Frequent Itemset* bezeichnet, wenn seine absolute Häufigkeit nicht unterhalb einer definierten unteren Schranke σ_{abs} liegt. Für σ_{abs} gilt: $0 \leq \sigma_{abs} \leq |\mathsf{D}|$. Eine untere Schranke für die relative Häufigkeit wird durch σ_{rel} bezeichnet. Es ist $\sigma_{abs} = \lceil \sigma_{rel} \cdot |\mathsf{D}| \rceil$. Im Folgenden wird ausschließlich die absolute Häufigkeit verwendet und mit dem Symbol σ bezeichnet.

Satz 3.1. (Monotonie der Häufigkeit von Mengen)
Sei D eine Transaktionsdatenbank über I. Seien X und Y zwei Mengen mit $X, Y \subseteq I$. Dann gilt

$$X \subseteq Y \Rightarrow \text{supp}(X) \geq \text{supp}(Y) \tag{3.8}$$

und daraus folgt

$$\text{supp}(Y) \geq \sigma \Rightarrow \text{supp}(X) \geq \sigma \tag{3.9}$$

$$\text{supp}(X) < \sigma \Rightarrow \text{supp}(Y) < \sigma \tag{3.10}$$

Definition 3.2. (Menge der Frequent Itemsets)
Sei D eine Transaktionsdatenbank über einer Menge von Items I und σ eine untere Schranke für die absolute Häufigkeit von Itemsets. Die Familie F von Frequent Itemsets unter Verwendung von σ ist definiert als

$$\mathsf{F}(\mathsf{D}, \sigma) := \{M \subseteq I \mid \text{supp}(M, \mathsf{D}) \geq \sigma\} \tag{3.11}$$

Definition 3.3. (Frequent Itemset Mining)
Gegeben seien eine Transaktionsdatenbank D über einer Menge I und eine untere Schranke σ für die absolute Häufigkeit. Aufgabe des Frequent Itemset Mining ist das Finden der Mengenfamilie $\mathsf{F}(\mathsf{D}, \sigma)$ aller Frequent Itemsets. Zusätzlich sollen zu allen Frequent Itemsets M ihre Häufigkeiten $\text{supp}(M, \mathsf{D})$ bestimmt werden.

3.2.2 Assoziationsregeln

Sobald die Menge F der Frequent Itemsets bestimmt ist, können daraus Assoziationsregeln konstruiert werden. Eine Assoziationsregel ist ein Ausdruck der Form $X \Rightarrow Y$,

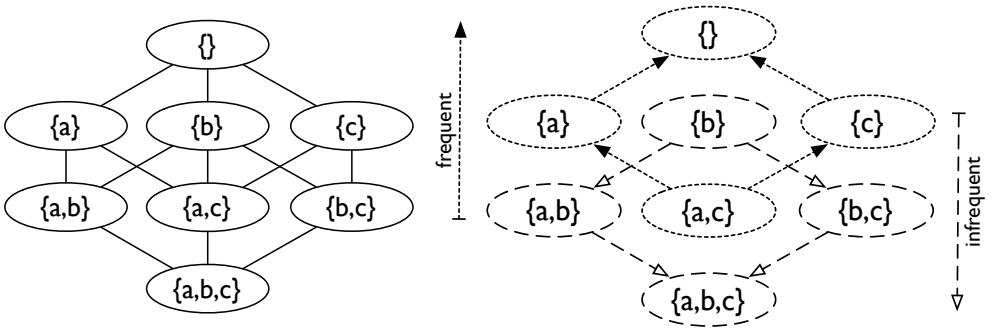


Abbildung 3.1: Der Suchraum eines Algorithmus zum Finden aller Frequent Itemsets hat die Form des Verbands, der durch die Potenzmenge $\mathcal{P}(I)$ der Menge aller Items und die Teilmengen-Relation gegeben ist. Es müssen aber nicht alle $2^{|I|}$ Elemente betrachtet werden, da einerseits alle Teilmengen einer häufigen Menge ebenfalls häufig sind und andererseits aus der Seltenheit einer Menge auf die Seltenheit aller ihrer Übermengen geschlossen werden kann. Die Menge $\{a, c\}$ wird als *positive Grenze*, die Menge $\{b\}$ als *negative Grenze* bezeichnet.

wobei X und Y Itemsets sind und $X \cap Y = \emptyset$ gilt. Eine Regel dieser Form sagt aus, dass eine Transaktion, die X beinhaltet auch Y enthält. Die Häufigkeit einer Regel entspricht der Häufigkeit von $X \cup Y$. Als häufige Regeln werden diejenigen Regeln bezeichnet, deren Häufigkeit nicht unterhalb einer unteren Schranke σ_{abs} (bzw. σ_{rel}) liegt.

Die Konfidenz (*Confidence*) einer Regel ist definiert als die bedingte Wahrscheinlichkeit, dass Transaktionen, die X beinhalten, auch Y enthalten.

$$\text{conf}(X \Rightarrow Y, \mathsf{D}) := P(X \mid Y) = \frac{\text{supp}(X \cup Y, \mathsf{D})}{\text{supp}(X, \mathsf{D})} \tag{3.12}$$

Eine Regel heißt sicher (*confident*), wenn ihre Konfidenz nicht unterhalb einer unteren Schranke λ liegt, wobei $0 \leq \lambda \leq 1$ ist.

Definition 3.4. (Menge häufiger und sicherer Assoziationsregeln)
Sei D eine Transaktionsdatenbank über einer Menge von Items I, σ eine untere Schranke für die absolute Häufigkeit und λ eine untere Schranke für die Konfidenz. Dann ist die Menge der häufigen und sicheren Assoziationsregeln definiert als

$$\begin{aligned} \mathsf{R}(\mathsf{D}, \sigma, \lambda) := \{X \Rightarrow Y \mid & X, Y \subseteq I, \\ & X \cap Y = \emptyset, \\ & X \cup Y \in \mathsf{F}(\mathsf{D}, \sigma), \\ & \text{conf}(X \Rightarrow Y, \mathsf{D}) \geq \lambda\} \end{aligned} \tag{3.13}$$

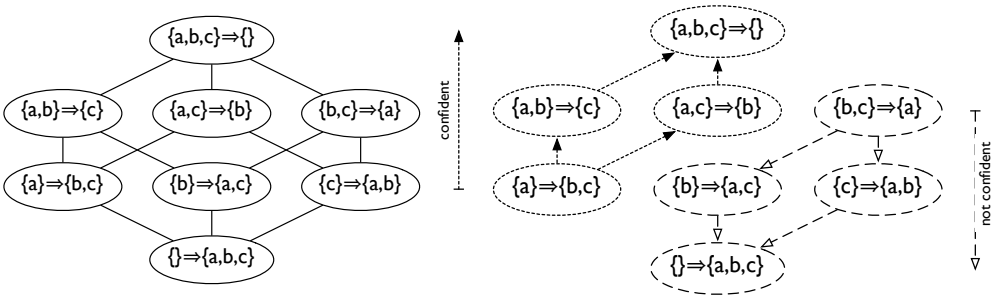


Abbildung 3.2: Die Menge aller Assoziationsregeln $X \Rightarrow Y$ zu einem Itemset I' mit $X \cup Y = I'$ und die Teilmengenrelation bilden einen Verband mit $2^{|I'|}$ Elementen. Der Suchraum für einen Algorithmus zur Bestimmung von R hat eine Größe von $3^{|I|}$.

Definition 3.5. (Association Rule Mining)
Gegeben sei eine Transaktionsdatenbank D über einer Menge I, eine untere Schranke für die absolute Häufigkeit σ und eine untere Schranke λ für die Konfidenz. Aufgabe des Association Rule Mining ist das Finden der Menge $\mathsf{R}(\mathsf{D}, \sigma, \lambda)$ aller häufigen und sicheren Assoziationsregeln. Zusätzlich soll zu jeder Regel ihre Häufigkeit und Konfidenz bestimmt werden.

Satz 3.6. (Monotonie der Konfidenz von Assoziationsregeln)
Seien X, Y und Z Mengen mit $X, Y, Z \subseteq I$ und $X \cap Y = \emptyset$. Dann gilt

$$\text{conf}(X \setminus Z \Rightarrow Y \cup Z) \leq \text{conf}(X \Rightarrow Y) \tag{3.14}$$

und daraus folgt

$$X \Rightarrow Y \geq \lambda \Rightarrow (X \cup Z \Rightarrow Y \setminus Z) \geq \lambda \tag{3.15}$$
$$X \Rightarrow Y < \lambda \Rightarrow (X \setminus Z \Rightarrow Y \cup Z) < \lambda \tag{3.16}$$

Einen sehr viel ausführlicheren Überblick über den Forschungsbereich des Frequent Itemset Mining bzw. des Association Rule Mining bietet ein Artikel von Goethals [13].

3.2.3 FP-Growth

Der erste Algorithmus zur Bestimmung von Frequent Itemsets und der zugehörigen Assoziationsregeln ist der Algorithmus AIS von Agrawal et al. [1]. Der Algorithmus wurde kurz darauf verbessert [2], indem die Monotonieeigenschaften der Häufigkeit von Mengen (Satz 3.1) und der Konfidenz von Regeln (Satz 3.6) ausgenutzt wurden. Diese verbesserte Version wurde APRIORI genannt.

Im Rahmen dieser Diplomarbeit wird der Algorithmus FP-GROWTH von Han et al. [18] verwendet.

FP-GROWTH behandelt Transaktionen als Wörter über einem Alphabet I, in denen jeder Buchstabe nur einmal vorkommen kann. Die Aufgabe des Findens von Frequent Itemsets entspricht, in dieser Formulierung des Problems, dem Finden häufiger Muster (*frequent patterns*), d.h. Teilmengen des Alphabets, in diesen Wörtern. Ein Muster der Kardinalität $k \in \mathbb{N}$ wird im Folgenden k-Muster genannt. Die Transaktionsdatenbank wird in diesem Kontext Wortbasis B genannt.

In einem ersten Schritt wird die globale Häufigkeit jedes Buchstabens ermittelt, indem nacheinander alle Wörter aus der Wortbasis betrachtet werden. Mit einer vorgegebenen unteren Schranke für die absolute Häufigkeit σ werden aus dieser Menge die häufigen Buchstaben ausgewählt.

$$\Phi = \{\phi_1, ..., \phi_n\} = \{i \in I \mid \text{supp}(i) \geq \sigma\} \subseteq I \tag{3.17}$$

ϕ_1 entspricht dem häufigsten, ϕ_n dem seltensten Buchstaben.

Im zweiten Schritt werden wiederum alle Wörter nacheinander betrachtet. Nun werden aus jedem Wort alle seltenen Buchstaben entfernt und die verbleibenden Buchstaben absteigend nach ihrer Häufigkeit sortiert, um es anschließend in einer FPTree genannten Baumstruktur zu speichern. Für das erste Wort $(b_1, ..., b_k)$ mit $b_i \in \Phi$, wird für jeden Buchstaben ein Baumknoten angelegt, der diesen Buchstaben und einen Häufigkeitswert von 1 enthält. Der Knoten des ersten Buchstabens b_1 wird unter einen speziell ausgezeichneten Wurzelknoten gehängt. Die Knoten für zwei im Wort benachbarte Buchstaben b_i und b_{i+1} werden gegenseitig miteinander verbunden. Jedes weitere Wort wird, abhängig von den bereits eingefügten Wörtern, auf eine von drei verschiedenen Weisen in den Baum eingefügt.

1. Für ein Wort $(b'_1, ...b'_l)$, welches ein Präfix eines bereits eingefügten Wortes $(b'_1, ..., b'_l, b'_{l+1}, ...)$ ist, werden keine neuen Knoten angelegt. Es ist bereits ein passender Pfad im Baum vorhanden. Der Häufigkeitswert in jedem Knoten, der sich auf diesem Pfad befindet, wird um den Wert 1 erhöht.

2. Für ein Wort, welches kein Präfix mit einem bereits eingefügten Wort teilt, wird nach dem gleichen Schema wie beim ersten Wort ein vollständig neuer Pfad angelegt.

3. Soll ein Wort $(b'_1, ..., b'_l, b'_{l+1}, ...)$ eingefügt werden, welches ein gemeinsames Präfix $(b'_1, ..., b'_l)$ mit einem bereits eingefügten Wort $(b'_1, ..., b'_l, b''_1, ...)$ hat, wird zunächst für das Präfix $(b'_1, ..., b'_l)$ nach dem gleichen Schema wie unter Punkt 1 der entsprechende Pfad im Baum modifiziert. Anschließend wird für alle dem Präfix nachfolgenden Buchstaben $(b'_{l+1}, ...)$, im gleichen Schema wie unter Punkt 2, ein

unverarbeitet	gefiltert und sortiert
f,c,a,d,g,i,m,p	*f,c,a,m,p*
a,b,c,f,l,m,o	*f,c,a,b,m*
b,f,h,j,o	*f,b*
b,c,k,s,p	*c,b,p*
a,f,c,e,l,p,m,n	*f,c,a,m,p*

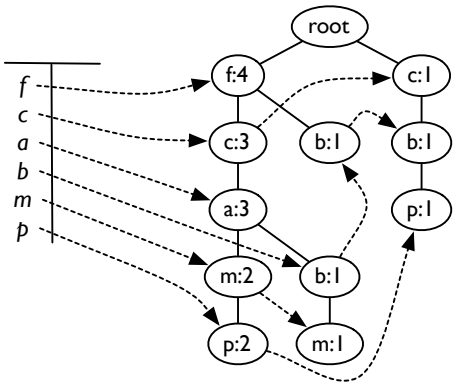
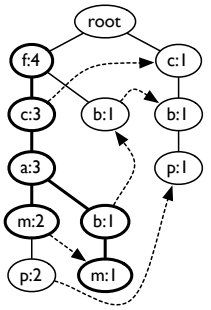


Abbildung 3.3: Ein Beispiel für einen FPTree. Es wird eine minimale Häufigkeit von $\sigma = 3$ verwendet.

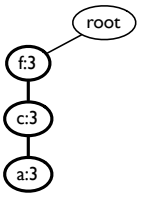
neuer Pfad angelegt. Im Unterschied zu Punkt 2 wird der Knoten zum Buchstaben b'_{l+1} des neuen Pfads mit dem letzten Knoten des zuvor betrachteten und modifizierten Pfad und nicht mit dem Wurzelknoten verbunden.

Um den Präfixbaum schnell traversieren zu können, wird eine Referenztabelle erstellt, die zu jedem Buchstaben ϕ_i, in absteigender Reihenfolge nach ihrer Häufigkeit sortiert, einen Zeiger auf den ersten Knoten für diesen Buchstaben enthält. In jedem Knoten selbst ist ein Zeiger auf einen weiteren Knoten mit dem selben Buchstaben wie dem eigenen gespeichert. Auf diese Weise werden alle Knoten für einen bestimmten Buchstaben in Listenform organisiert. Während der Konstruktion des Präfixbaums wird ein neu angelegter Knoten an die seinem Buchstaben entsprechende Liste angehängt.

Die Menge Φ der häufigen Buchstaben wird nun beginnend mit dem seltensten Buchstaben ϕ_n durchlaufen. Jeder Buchstabe ϕ_i entspricht einem 1-Muster und wird direkt ausgegeben. Anschließend betrachtet man die zugehörige Liste aller Knoten, die in der Referenztabelle abgelegt ist. Man verfolgt von diesen Endknoten aus alle Pfade aufwärts zum Wurzelknoten. Für jeden Pfad notiert man das Wort, welches sich aus den in den Knoten gespeicherten Buchstaben ergibt, wobei man mit den Knoten oberhalb der Endknoten beginnt. Man erstellt aus diesen Wörtern eine neue Wortbasis. Wie oft jedes einzelne Wort in diese Basis eingefügt wird, richtet sich nach dem Häufigkeitswert, der im jeweiligen Startknoten gespeichert ist. Die Intuition hinter diesem mehrfachen Einfügen ist, dass ein Endknoten mit Häufigkeitswert x nur entstanden sein kann, wenn der Pfad von der Wurzel zu diesem Knoten beim Einfügen von genau x Wörtern benutzt wurde. Genau diese Wörter möchte man im Folgenden rekursiven Schritt weiter untersuchen.



(a) Ursprünglicher FPTree



(b) Conditional FPTree

unverarbeitet	gefiltert und sortiert
f,c,a	f,c,a
f,c,a	f,c,a
f,c,a,b	f,c,a

(c) Eingeschränkte Wortbasis

Abbildung 3.4: Für den Buchstaben m werden alle Pfade zur Wurzel verfolgt und aus den enthaltenen Wörtern eine neue Wortbasis erstellt, die nur Wörter enthält, die auf m enden. Daraus wird dann ein Conditional-FPTree erzeugt.

Die so erstellte Wortbasis B' enthält nur Wörter, die der Bedingung genügen, dass sie auf den Buchstaben ϕ_i enden. Wortbasen auf Rekursionstiefe k fordern ein Wortsuffix aus k Buchstaben. Wörter mit dem Suffix (b, ϕ_i) sind nur dann häufig, wenn der Buchstabe b ebenfalls die Mindesthäufigkeit σ hat. Jeder Buchstabe b, dessen Häufigkeit unterhalb dieser Schranke liegt, wird aus allen Wörtern entfernt. Es entsteht eine neue Menge $\Phi' \subset \Phi$ häufiger Buchstaben für das Teilproblem. Die Buchstaben in allen Wörtern werden wieder absteigend gemäß der Häufigkeiten geordnet. Anschließend wird nach dem bekannten Verfahren ein FPTree erstellt, ein sogenannter Conditional-FPTree. Das Wort *Conditional* soll andeuten, dass eine Bedingung an die Wörter in der Wortbasis gestellt wird. Beginnend mit dem seltensten Buchstaben wird die Menge Φ' durchlaufen. Da ϕ'_j häufig ist und zusätzlich die Bedingung erfüllt ist, dass ϕ_i häufig ist, wird das 2-Muster $\{\phi'_j, \phi_i\}$ ausgegeben. Anschließend wird ein Rekursionsschritt durchgeführt, d.h. es wird eine Wortbasis B'' erstellt, die nur Wörter mit dem Suffix (ϕ'_j, ϕ_i) enthält. Daraus wird wiederum ein Conditional-FPTree für eine Wortbasis erstellt. Die Rekursion endet, sobald die rekursiv erzeugte Wortbasis keine häufigen Buchstaben mehr enthält.

Der Divide-and-Conquer Ansatz von Fp-Growth partitioniert den Gesamtsuchraum aller Muster rekursiv. Dabei wird die Monotonie-Eigenschaft aus Satz 3.1 ausge-

nutzt. Für Wort-Suffixe, die nicht häufig sind, wird keine Rekursion durchgeführt und der entsprechende Teil des Suchraums im weiteren Verlauf ignoriert.

3.3 Frequent Itemset-basiertes Clustering von Textdokumenten

In [12] argumentieren Ester et al., dass die in Abschnitt 3.1 vorgestellten Verfahren nicht gut auf die speziellen Anforderungen des Clusterings von Textdokumenten angepasst sind.

Hohe Dimensionalität: Wortvektoren haben in der Regel eine hohe Dimensionalität. Cluster können häufig nicht im voll-dimensionalen Raum, sondern nur in Unterräumen gefunden werden.

Skalierbarkeit: Es muss eine sehr große Anzahl von Dokumenten verarbeitet werden. Die meisten Verfahren sind nicht in diesem Maße skalierbar.

Hohe Genauigkeit: Die Ähnlichkeit der Dokumente in einem Cluster sollte hoch, die Ähnlichkeit der Cluster untereinander gering sein.

Verständlichkeit: Das Resultat sollte dem Benutzer verständlich sein. Dazu müssen die erstellten Cluster mit aussagekräftigen Beschreibungen versehen werden. Auf diese Weise wird der Benutzer beim interaktiven Browsing in der Dokumenthierarchie unterstützt.

Parametrisierung: Es sollte vom Benutzer des Verfahren nicht verlangt werden, vor dem Start Parametereinstellungen (wie z.B. die Clusteranzahl k bei KMeans) vornehmen zu müssen, die die Güte des Resultats beeinflussen.

Wang et al. [41] führen ein neues Kriterium zum Clustering von Transaktionen auf der Basis von Frequent Itemsets (siehe Kapitel 3.2) ein. Transaktionen sollen derart in Cluster angeordnet werden, dass innerhalb eines Clusters gleiche Frequent Items häufig auftauchen, während verschiedene Cluster möglichst unterschiedliche Frequent Items aufweisen.

Basierend auf dieser Arbeit wird von Ester et al. das auf Frequent Itemsets basierende Verfahren Fihc vorgestellt, das die speziellen Anforderungen des Clusterings von Dokumenten erfüllt. Der Ablauf des Verfahrens ist wie folgt:

Konstruktion der initialen Cluster: Für jedes Frequent Itemset wird ein initiales Cluster erstellt. Diesem Cluster werden alle Dokumente zugeordnet, die dieses Itemset enthalten. Das Frequent Itemset dient als Label für dieses Cluster. Die auf diese Weise erstellten Cluster überlappen sich. Daher wird nun für jedes Dokument doc_j mittels

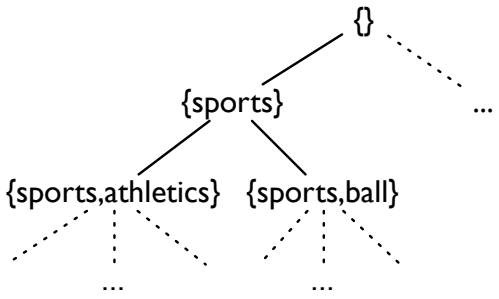


Abbildung 3.5: Ein von FIHC erzeugtes Beispiel-Clustering

einer Funktion *score* bestimmt, welches das passendste Cluster C_i ist. Es verbleibt nur in diesem Cluster und wird aus allen anderen entfernt. Nach diesem Schritt ist jedes Dokument in exakt einem Cluster enthalten.

Aufbau des Cluster-Baums: Jedes Cluster hat in diesem Baum genau ein Elter-Cluster. Ein Cluster mit einem k-Itemset als Label befindet sich auf Ebene k im Baum. Der Baum wird von unten nach oben aufgebaut. Für ein Cluster C auf Ebene k mit einem Frequent Itemset X wird, unter allen Clustern auf Ebene $k-1$ mit einem Frequent Itemset $Y \subset X$, das beste Elter-Cluster ausgesucht. Dazu verschmilzt man alle Dokumente in C zu einem einzigen Dokument und verwendet die Funktion *score*, wie schon bei der Bestimmung der besten initialen Cluster für ein einzelnes Dokument.

Pruning des Cluster-Baums: Der konstruierte Baum kann sehr breit und sehr tief sein und ist deshalb ungeeignet zum Browsing durch einen Benutzer. Daher werden im Baum zwei Geschwister-Cluster verschmolzen, wenn sie sehr ähnlich zueinander sind. Ebenso werden Kind-Cluster entfernt, die zu ähnlich zu ihrem Elter-Cluster sind. Die im Kind-Cluster enthaltenden Dokumente werden dem Elter-Cluster zugeordnet.

FIHC erfüllt die gestellten Anforderungen an ein Verfahren im Anwendungsbereich des Clusterings von Dokumenten. Die Verwendung von Frequent Itemsets führt zu einer Reduktion der Dimensionalität auf ein handhabbares Maß. Die gute Skalierbarkeit für große Datenmengen wird in [12] experimentell belegt. Die erzeugte Hierarchie von Frequent Itemsets ist für den Benutzer verständlich und eignet sich sehr gut für den Anwendungsfall des Browsings (siehe Abbildung 3.5). Desweiteren ist für das Verfahren keine Parametrisierung notwendig.

Wie bereits in Kapitel 3.1 beschrieben, lassen sich die in einem Tagging-System ge-

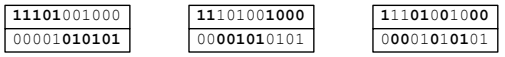
speicherten Ressourcen als Textdokumente und Tags als Wörter in diesen Dokumenten interpretieren. Ein Verfahren zum Clustering von Textdokumenten kann auf diesem Wege zum Clustering von Ressourcen anhand von Tags verwendet werden. Daraus folgt, dass das in diesem Abschnitt vorgestellte Clustering-Verfahren FIHC ebenfalls anwendbar ist. Für Tagging-Daten können somit trotz ihrer hohen Dimensionalität verständliche Cluster effizient erstellt werden, ohne dass der Benutzer eine Parametrisierung des Verfahrens vornehmen muss.

In Kapitel 2.6 wird als Ziel für diese Arbeit die Erzeugung von Navigationsstrukturen definiert, die an die Anforderungen des Benutzers angepasst sind. Aufgrund der Widersprüchlichkeit dieser Anforderungen soll eine Menge alternativer Strukturen generiert werden, aus welcher der Benutzer seine Auswahl trifft. Die Verwendung von FIHC erzeugt aber nur eine einzelne Lösung. Die durch das Verfahren optimierten Eigenschaften dieser Lösung sind dem Benutzer nicht ersichtlich, was eine Interpretation sehr schwierig macht. Der als Teil des Verfahrens durchgeführte Pruning-Schritt kann als versteckter Parameter angesehen werden, welcher steuert, wann Cluster zusammengefasst werden. Für den Benutzer bietet die Anpassung dieses Parameters die einzige Möglichkeit, den Ablauf des Verfahrens zu beeinflussen. Eine Exploration des Lösungsraums durch sukzessive Änderung von Parametereinstellungen ist aber, wie schon in Abschnitt 3.1.4 erwähnt, sehr mühsam und unintuitiv.

In Kapitel 4 wird ein neuer Ansatz zum Clustering von Tagging-Daten vorgestellt. Die Optimierungskriterien werden explizit dargestellt, was eine bessere Interpretation der Ergebnisse ermöglicht. Es werden mehrere alternative Lösungen generiert und diese auf eine Weise visualisiert, die es dem Benutzer erlaubt, die Auswirkungen der Änderung des Wertes eines Kriteriums zu verstehen. Die Alternativenmenge bietet dem Benutzer zudem die Möglichkeit, den Lösungsraum zu erforschen und die für ihn individuell passendste Lösung auszuwählen. Der Ansatz wird als multikriterielles Optimierungsproblem formuliert, da dies die natürliche Umsetzung der gestellten Anforderungen ermöglicht. Zur Lösung multikriterieller Optimierungsprobleme werden in der Regel evolutionäre Verfahren eingesetzt. Daher folgt in Kapitel 3.4 zunächst eine Einführung in die Grundlagen der genetischen Algorithmen. Es schließt sich in Kapitel 3.5 eine Darstellung der Grundlagen der multikriteriellen Optimierung an. Schließlich wird mit NSGA-II [9] das konkrete Optimierungsverfahren vorgestellt, welches im Rahmen dieser Arbeit verwendet wird.

3.4 Genetische Algorithmen

Genetische Algorithmen (*GAs*) sind Suchverfahren, deren Suchraum aus Bitvektoren konstanter Länge besteht. Eine grundlegende Einführung bieten Deb [7] und Mitchell [30]. Es folgt eine kompakte Darstellung der wichtigsten Begriffe: Im Suchraum soll derjenige Bitstring gefunden werden, dessen durch eine vorgegebene Funktion berech-



(a) 1-point (b) 2-point (c) Uniform

Abbildung 3.6: Verschiedene Crossover-Arten.

nete Bewertung optimal ist. GAs sind inspiriert von biologischen Evolutionsprozessen. Lebewesen, deren individuelle Eigenschaften ihre Chancen zum Überleben und zur Fortpflanzung verbessern, geben diese Eigenschaften an ihre Nachkommen weiter. Über viele Generationen verbreiten sich diese Eigenschaften, während weniger erfolgversprechende Eigenschaften seltener werden oder ganz verschwinden. Dieser Prozess führt zu einer Anpassung an die Lebensbedingungen. GAs verwenden diese Prinzipien in vereinfachter Weise zur Steuerung des Suchprozesses.

Ein Individuum ist ein Vektor $\vec{i} \in \{0,1\}^l$, wobei $l \in \mathbb{N}$ die Länge des Vektors bezeichnet. Eine Population ist eine Menge $P \subseteq \mathcal{P}(\{0,1\}^l)$ von Individuen, wobei $m = |P|$ die Größe der Population ist. Die Fitness eines Individuums wird durch eine vorgegebene Funktion fit, die sogenannte Fitnessfunktion, berechnet. Es gilt $fit : \{0,1\}^l \to \mathbb{R}$. Unter genetischen Operatoren versteht man Funktionen, die auf eines oder mehrere Individuen angewandt werden, um neue Individuen zu erzeugen. Man unterscheidet zwischen Selektions-, Mutations- und Crossover-Operatoren. Sie werden verwendet, um im Ablauf des Algorithmus, beginnend mit einer initialen Population, sequentiell aufeinanderfolgende Populationen, auch Generationen genannt, zu erzeugen.

Selektion: Ein Selektionsoperator wählt aus einer Population probabilistisch ein Individuum aus. *Roulette-Wheel-Selection* wählt jedes Individuum mit einer Wahrscheinlichkeit aus, die proportional zu seiner eigenen Fitness und invers-proportional zur Fitness aller anderen Individuen der Population ist. *Tournament-Selection* wählt zwei Individuen zufällig aus, deren Fitnesswerte miteinander verglichen werden. Einer vorher festgelegten Wahrscheinlichkeit entsprechend, wird das bessere oder das schlechtere Individuum selektiert. *Ranking-Selection* sortiert die Individuen zunächst nach ihrer Fitness. Anschließend wird ein Individuum mit einer Wahrscheinlichkeit proportional zu seinem Rang in der sortierten Liste anstatt zu seiner Fitness selektiert.

Crossover: Eine Crossover-Operator erzeugt aus zwei Individuen einer Population zwei neue Individuen. In Abbildung 3.6 sind verschiedene Varianten dieses Operators dargestellt. Die hervorgehobenen Teile der beiden Bitstrings bilden das erste der neuen

Individuen, die anderen Teile das zweite.

Mutation: Mutationsoperatoren erzeugen aus einem Individuum ein neues. *Point mutation* invertiert ein zufällig gewähltes Bit. Es gibt andere Varianten, die mehrere Bits invertieren.

Definition 3.7. (Genetische Suche)
Im Suchraum $\mathcal{P}(\{0,1\}^l)$ der Bitvektoren der Länge $l \in \mathbb{N}$, finde einen Vektor $\vec{i}_*$ derart, dass $\vec{i}_* = \text{argmax}_x\, fit(x)$ (bei Maximierung der Fitness) bzw. $\vec{i}_* = \text{argmin}_x\, fit(x)$ (bei Minimierung der Fitness).

3.4.1 Problemtransformation und Parameterwahl

Vor Verwendung eines GA sind mehrere Vorbereitungsschritte notwendig. Es muss eine Relation zwischen dem Raum U des eigentlich zu lösenden Problems und dem Raum $B = \mathcal{P}(\{0,1\}^l)$ von Bitstrings konstanter Länge konstruiert werden. Passend dazu muss eine Fitnessfunktion definiert werden, die ein Individuum aus B zurück nach U abbildet und es dann bewertet. In Einzelfällen kann die Berechnung ohne Abbildungsschritt direkt auf B arbeiten. Es muss ein Abbruchkriterium definiert werden. Üblicherweise definiert man entweder eine zu erreichende Fitnessschranke oder eine maximale Anzahl von Generationen. Weiterhin muss festgelegt werden, welcher spezielle Selektions-,Mutations- und Crossover-Operator verwendet wird. Schließlich müssen eine Populationsgröße $m \in \mathbb{N}$, eine Crossover-Rate $q \in [0,1]$, die den Anteil der Population festlegt, der pro Generation durch Crossover ersetzt wird, und eine Mutationsrate $s \in [0,1]$, die definiert, wie groß der Anteil der Population ist, auf den der Mutationsoperator angewendet wird, spezifiziert werden.

3.4.2 Ablauf eines genetischen Algorithmus

Es folgt ein prototypischer Ablauf eines GA.

- Initiale Population erstellen: $P \leftarrow$ Generiere zufällig m Individuen.
- Auswerten der Fitness: Für jedes $\vec{i} \in P$, berechne $fit(\vec{i})$.
- Solange das Abbruchkriterium nicht erfüllt ist:
  1. Selektion: Wähle mit dem Selektionsoperator $(1-q) \cdot m$ Individuen aus P und füge sie unverändert zu P_+ hinzu.
  2. Crossover: Selektiere $\frac{q \cdot m}{2}$ Paare von Individuen aus P. Erzeuge durch Anwendung des Crossover-Operators zwei neue Individuen und füge sie zu P_+ hinzu.

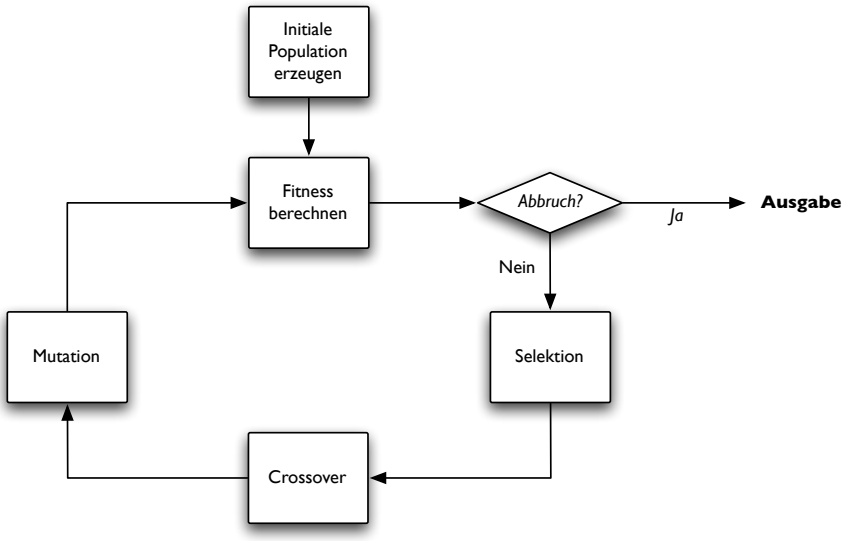


Abbildung 3.7: Prototypischer Ablauf eines genetischen Algorithmus.

3. Mutation: Wähle uniform verteilt $s \cdot m$ aus P_+ Individuen. Wende den Mutationsoperator auf jedes gewählte Individuum an.
4. Ersetze die alte Population: $P = P_+$
5. Berechne die Fitnesswerte: Für jedes $\vec{i} \in P$, berechne $fit(\vec{i})$.

- Gib das Individuum aus P mit dem höchsten (bzw. geringstem) Fitnesswert zurück.

Praktische Probleme genetischer Algorithmen
GAs leiden an einem Problem, das unter dem Begriff *Crowding* bekannt ist. Damit wird beschrieben, dass sich Individuen mit hoher Fitness schneller reproduzieren und die Diversität der Population absinkt [30], was den Suchfortschritt stark bremst. Ein Ansatz zur Lösung dieses Problems ist die Wahl eines Selektionsoperators, der mit einer bestimmten Wahrscheinlichkeit auch schlechte Individuen selektiert. Ein anderer ist es, die Fitness eines Individuums herabzusetzen, wenn es viele zu diesem ähnliche Individuen in der Population gibt.

3.5 Multikriterielle Optimierung

Viele Planungs- und Entscheidungsprobleme in der realen Welt setzen die gleichzeitige Optimierung mehrerer Kriterien voraus. Beim Betrieb einer Produktionsanlage bei-

spielsweise sollen gleichzeitig die Wartungskosten minimiert und die Produktionsgeschwindigkeit maximiert werden. Soll die Produktionsgeschwindigkeit erhöht werden, so verursacht dies einen Anstieg der Wartungskosten.

Multikriterielle Optimierung befasst sich mit der Formulierung und Lösung von Problemen dieser Art. Während einkriterielle Optimierungsverfahren ein Optimum finden, welches ein Kriterium bestmöglich erfüllt, ist es bei multikriteriellen Optimierungsproblemen im Allgemeinen nicht möglich, ein solches Optimum zu finden, da die verschiedenen Kriterien in der Regel zueinander in Konflikt stehen. Stattdessen wird eine Menge von Lösungen mit möglichst geringem Konflikt zwischen den Kriterien bestimmt. Ohne Beschränkung der Allgemeinheit werden im Folgenden Maximierungsprobleme betrachtet.

Einen Überblick über die Grundbegriffe der multikriteriellen Optimierung bietet Deb [8]. Im Folgenden werden die wichtigsten Begriffe und Definitionen zusammengefasst:

Definition 3.8. (Multikriterielle Bewertungsfunktion)
Sei ein Optimierungsproblem mit n Kriterien gegeben. Als multikriterielle Bewertungsfunktion wird eine Vektorfunktion $\vec{f} : S \to \mathbb{R}^n$ bezeichnet, die ein Element $x \in S$ auf seine Bewertungen bezüglich der n Kriterien abbildet.

Definition 3.9. (Pareto-Dominanz)
Gegeben seien zwei Lösungsvektoren $\vec{u}, \vec{v} \in \mathbb{R}^n$, die mit einer multikriteriellen Bewertungsfunktion für zwei Elemente aus S bestimmt wurden. Ein Lösungsvektor $\vec{u}$ pareto-dominiert einen anderen Lösungsvektor $\vec{v}$ (kurz $\vec{u} \succeq \vec{v}$), wenn gilt:

$$\forall i \in \{1, \ldots, n\} : u_i \geq v_i \tag{3.18a}$$

sowie

$$\exists i \in \{1, \ldots, n\} : u_i > v_i \tag{3.18b}$$

Ein Lösungsvektor heißt nicht-dominiert, wenn er von keinem anderen Lösungsvektor pareto-dominiert wird.

Definition 3.10. (Pareto-optimale Menge)
Für ein multikriterielles Optimierungsproblem ist die pareto-optimale Menge gegeben als

$$Q^* := \{x \in S \mid \nexists y \in S : \vec{f}(y) \succeq \vec{f}(x)\} \tag{3.19}$$

d.h. Q^* enthält nur nicht-dominierte Lösungsvektoren.

Definition 3.11. (Multikriterielle Optimierung)
Bestimme eine Teilmenge L der Menge Q^* aller pareto-optimalen Lösungen, sodass die Elemente in L möglichst gleichmäßig über Q^* verteilt sind (siehe Abbildung 3.8)

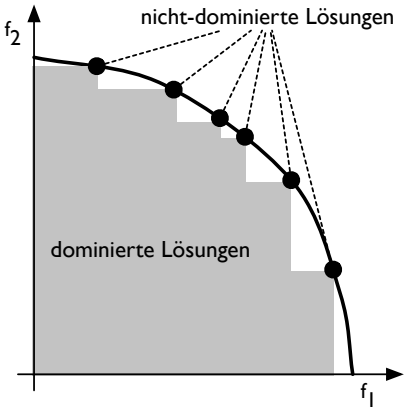


Abbildung 3.8: Für ein Beispielproblem ist hier die pareto-optimale Menge Q^* als schwarze Linie dargestellt. Diese wird auch Pareto-Front genannt. Die Punkte auf der Linie stellen die Elemente der Lösungsmenge L dar. L soll auf eine Weise bestimmt werden, dass die Punkte möglichst gleichmäßig über die Pareto-Front verteilt sind.

Die Lösungen in L sind derart, dass sie sich bezüglich eines Kriterium nur dann weiter verbessern können, wenn sie sich bezüglich eines anderen verschlechtern.

Es ist es oft sehr schwierig, eine Aussage darüber zu treffen, welches Kriterium mit welcher Priorität erfüllt werden soll. Selbst wenn dies für einen speziellen Anwendungsfall möglich ist, muss bei Änderungen der Anforderungen die Gewichtung angepasst werden. In vielen Fällen ist es darüberhinaus sogar wünschenswert, dass das Ergebnis nicht eine einzelne Lösung ist, sondern der Anwender die Wahl zwischen mehreren alternativen Lösungen hat.

Deb stellt in [8] einige klassische Verfahren zur Lösung multikriterieller Optimierungsprobleme vor, die im Folgenden kurz dargestellt werden:

3.5.1 Lösung durch Gewichtung der Kriterien

Die einfachste Möglichkeit ein multikriterielles Optimierungsproblem zu lösen, ist es, das Problem zunächst in ein einkriterielles Problem zu transformieren, indem man die einzelnen Kriterien gewichtet. Auf diese Weise ergibt sich folgende Formulierung für das Problem:

$$\max \sum_{i=1}^{n} w_i \cdot f_i(x) \text{ u.d.N. } x \in S \tag{3.20}$$

wobei $0 \leq w_i \leq 1$ und $\sum_{i=1}^{n} w_i = 1$. Mit n wird wiederum die Anzahl der Kriterien bezeichnet und f_i bezeichnet die i-te Stelle der multikriteriellen Bewertungsfunktion $\vec{f}$. Im Allgemeinen ergibt sich eine pareto-optimale Lösung [8].

Eine weitere Möglichkeit ist die Verwendung einer Distanzfunktion. Dazu müssen die jeweiligen Optima $\bar{y}_i$ der einzelnen Kriterien im Vorhinein bekannt sein. Danach lässt sich folgendes einkriterielle Optimierungsproblem formulieren:

$$\max(\sum_{i=1}^{n} |f_i(x) - \bar{y}_i|^k)^{1/k} \tag{3.21}$$

Der Wert $k \in \mathbb{N}$ gibt die Ordnungszahl der verwendeten Metrik an. So bezeichnet beispielsweise $k = 2$ die euklidische Metrik.

Alle diese Methoden lösen das multikriterielle Problem, indem sie es in ein einkriterielles Problem transformieren. Die errechnete Lösung ist im Allgemeinen pareto-optimal, jedoch muss der Anwender eine genaue Kenntnis der Problemdomäne haben, um die Verfahren korrekt zu parametrisieren. Die Lösung hängt sehr stark von der Wahl dieser Parameter ab. Sind die Zielfunktionen nicht deterministisch, wird eine Parameterwahl umso schwieriger. Eine Lösung mehrkriterieller Probleme auf diese Weise ist nicht sinnvoll [8].

Stattdessen ist es sinnvoller, dem Anwender eine Anzahl alternativer Lösungen zu präsentieren und ihm die Auswahl zu überlassen. Zur Implementierung eines solchen Verfahrens bieten sich GAs (siehe Kapitel 3.4) an, da diese auf einer Population von Individuen arbeiten. Jedes Individuum stellt eine alternative Lösung dar. Das Problem des Findens einer Menge pareto-optimaler Lösungen lässt sich auf natürliche Weise als GA formulieren. Verfahren dieser Art sind unter dem Begriff *Evolutionäre Mehrziel-Optimier-Algorithmen* (EMOA) bekannt. Eine grundlegende Übersicht zu EMOA bieten Deb [10] oder Coello Coello et al. [25].

Im Rahmen der Diplomarbeit wird der Algorithmus NSGA-II von Deb et al. [9] verwendet, dessen Funktionsweise im Folgenden vorgestellt wird. Alternative Verfahren sind SPEA2 von Zitzler [42] oder SMS-EMOA von Beume et al. [6].

3.5.2 NSGA-II

NSGA-II verwendet einen GA zur Bestimmung der Lösungsmenge $L \subseteq Q^*$ nicht-dominierter Elemente aus S. Wie in Abschnitt 3.4 beschrieben, muss zur Anwendung eines GA eine Abbildung vom Raum B der Bitvektoren in den Raum U des Problems gefunden werden. In diesem Kontext ist $U = S$. Wir nehmen im Folgenden an, dass eine solche Abbildung vom Benutzer des Algorithmus definiert wird. Weiterhin sei eine multikriterielle Bewertungsfunktion $\vec{f}$ vorgegeben. Dann kann jedem Bitvektor durch $\vec{f}$ ein Punkt im $\mathbb{R}^n$ zugeordnet werden, welcher anschließend in einen skalaren Fitnesswert transformiert werden muss. Alle Abbildungsschritte zusammengenommen bilden die Fitnessfunktion $fit : B \to \mathbb{R}$ (vgl. Abschnitt 3.4).

Um die Fitnesswerte für alle Bitvektoren einer Population zu berechnen, werden die Vektoren zunächst nach S abgebildet. Dort wird unter Verwendung von $\vec{f}$ die Bewertung bezüglich der n Kriterien vorgenommen; es ergibt sich für jeden Bitvektor

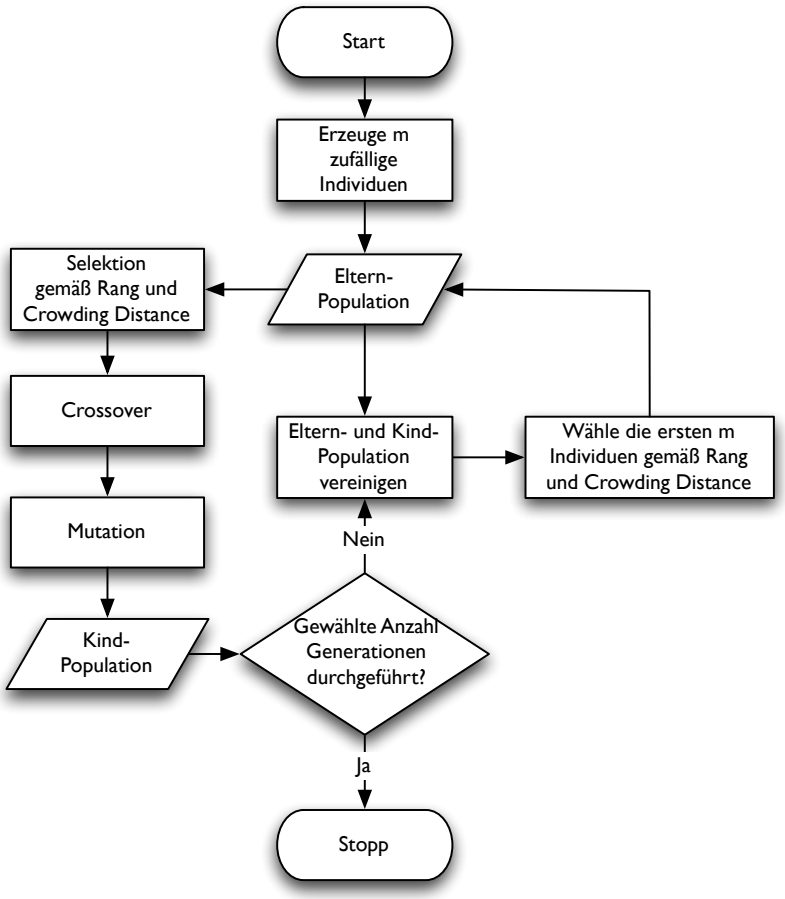


Abbildung 3.9: Ablauf des NSGA-II Algorithmus

ein Punkt im $\mathbb{R}^n$. Auf dieser Grundlage werden alle nicht-dominierten Individuen, die erste Front, ausgewählt. Jedes bekommt einen Fitnesswert von 1 (auch Rang genannt) zugewiesen. Damit wird gewährleistet, dass jedes Individuum das gleiche Reproduktionspotential hat. Dieses Verfahren wird auf die restliche Population erneut angewandt. Diesmal erhalten alle nicht-dominierten Individuen einen Fitnesswert von 2. Auf diese Weise werden alle Individuen nach Fronten klassifiziert und bekommen einen entsprechenden Fitnesswert. Es entsteht eine Rangfolge unter den Individuen. Ein Fitnesswert von 1 entspricht der besten Fitness, die einem Individuum zugeordnet werden kann. Die Fitness soll also minimiert werden.

Um die Diversität in der Population zu erhalten und damit den Problemen des Crowding (siehe Kapitel 3.4.2) vorzubeugen, muss die Selektionswahrscheinlichkeit

für Individuen der gleichen Front gesteuert werden. Man definiert hierfür den Begriff der *Crowding-Distance*. Man schätzt die Größe eines Quaders um das Individuum ab, welcher gerade so groß ist, dass er keine anderen Individuen enthält. Auf Basis dieses Maßes wird ein Vergleichsoperator $\geq_n$ definiert. Dieser wird benutzt, um den zur Selektion verwendeten binären Tournament-Selection-Operator zu steuern. Ein Individuum $\vec{i}_1$ wird einem Individuum $\vec{i}_2$ vorgezogen, wenn sein Fitnesswert kleiner ist, oder wenn beide Fitnesswerte gleich sind, die Individuen also in der gleichen Front liegen, $\vec{i}_1$ aber eine größere Crowding-Distance aufweist, also vereinzelter im Raum steht.

Der Ablauf des Algorithmus ist nun wie folgt: Als Erstes wird eine zufällige Population P'_0 erzeugt. Diese wird, wie zuvor beschrieben, bezüglich Nicht-Dominiertheit sortiert. Durch Anwendung von binärer Tournament-Selection, Rekombination und Mutation wird eine Kind-Population P''_0 der Größe m erzeugt. Für die Schritte $t \geq 0$ wird das Verfahren wie folgt fortgesetzt:

Zunächst wird eine kombinierte Population $P_t = P'_t \cup P''_t$ gebildet. Es gilt $|P_t| = 2m$. Anschließend wird diese Menge bezüglich Nicht-Dominiertheit sortiert. In aufsteigender Reihenfolge fügt man nun die Fronten zu einer neuen Eltern-Population P'_{t+1} hinzu, solange eine Größe von m Elementen nicht überschritten wird. Die besten bisher gefundenen Individuen, d.h. alle Individuen mit einem Rang von 1 bezüglich Nicht-Dominiertheit, werden daher in jedem Fall übernommen. Um P'_{t+1} anschließend bis auf eine Elementzahl von m aufzufüllen, sortiert man die zuletzt betrachtete (und nicht mehr eingefügte) Front bezüglich $\geq_n$. Anschließend wählt man die ersten $m - |P'_{t+1}|$ Individuen und fügt sie zu P'_{t+1} hinzu. Aus P'_{t+1} wird nun per Selektion, Mutation und Crossover die neue Kind-Population P''_{t+1} gebildet. Dieser Vorgang wird solange wiederholt, bis die vom Benutzer vorgegebene Anzahl durchzuführender Generationen erreicht wurde. Die aktuelle Eltern-Population P_t wird dann als Ergebnis zurückgeliefert.

Kapitel 4

Multikriterielles Tagset-Clustering

Der Inhalt dieses Kapitels ist gemäß der in Kapitel 2.6 vorgestellten Liste der Ziele der Arbeit strukturiert. Als Erstes wird ein formales Modell für hierarchische Navigationsstrukturen ausgearbeitet (Ziel 1) und die Suche nach geeigneten Navigationsstrukturen als multikriterielles Suchproblem formuliert (Ziel 2). Es folgt in Abschnitt 4.4 die Anpassung eines multikriteriellen Optimierungsverfahren für die vorliegende Aufgabe (Ziel 5). In Abschnitt 4.5 schließt sich die Formulierung der Bewertungsfunktionen (Ziel 3) an. Abschließend wird das Verfahren auf zeitlich veränderliche Folksonomies erweitert (Ziel 6).

4.1 Tagset-Clusterings

Satz 4.1. Gegeben sei eine Folksonomy $\mathbb{F} := (U, T, R, Y)$. Die Menge aller Tagsets $\mathcal{P}(T)$ ist eine Halbordnung bezüglich der Mengeninklusion $\subseteq$, da die Relation $\subseteq$ reflexiv, antisymmetrisch und transitiv ist.

Das kleinste Element der Ordnung ist das leere Tagset $\emptyset$, das größte Element ist das Tagset T. Die Halbordnung der Tagsets wird im Folgenden *Tagset-Halbordnung* $(\mathcal{P}(T), \subseteq)$ genannt.

Satz 4.2. Gegeben sei eine Folksonomy $\mathbb{F} := (U, T, R, Y)$. Die darauf definierte Tagset-Halbordnung $(\mathcal{P}(T), \subseteq)$ bildet mit dem Mengenschnitt $\cap$ und der Mengenvereinigung $\cup$ einen vollständigen Verband. Dies bedeutet, dass zu jeder Teilmenge von Tagsets $\mathsf{M} \subseteq \mathcal{P}(T)$ ein Infimum und ein Supremum existiert.

$$\inf(\mathsf{M}) = \bigcap_{M \in \mathsf{M}} M \qquad \sup(\mathsf{M}) = \bigcup_{M \in \mathsf{M}} M \tag{4.1}$$

Dieser Verband wird im Folgenden als *Tagset-Verband* $\langle \mathcal{P}(T), \cap, \cup \rangle$ bezeichnet.

Der Tagset-Verband bildet nun die Grundlage der Definition einer Navigationsstruktur, einem sogenannten Tagset-Clustering.

Definition 4.3. (Tagset-Clustering)
Gegeben sei eine Folksonomy $\mathbb{F} := (U, T, R, Y)$. Ein Tagset-Clustering ordnet eine

Menge von Tagsets als Cluster in hierarchischer Weise an. Jedes Cluster kann eine Menge von Ressourcen aus R enthalten. Jedes Tagset-Clustering besteht aus den folgenden Elementen:

- $\mathsf{C} \subseteq \mathcal{P}(T)$ ist eine endliche, nichtleere Menge von Tagsets. Jedes Tagset repräsentiert ein Cluster. C wird als *Clustermenge* des Clusterings bezeichnet.

- R ist eine endliche, nichtleere Menge, die alle Ressourcen repräsentiert, die im Clustering vorhanden sind. R ist durch die Folksonomy $\mathbb{F}$ vorgegeben.

- Die Relation $\prec$: $\mathcal{P}(T) \times \mathcal{P}(T)$ definiert die hierarchische Anordnung aller Cluster aus C. Für alle Cluster $C, D \in \mathsf{C}$ gilt:

$$C \prec D \Leftrightarrow (C \subset D) \wedge (|C| = |D| - 1) \tag{4.2}$$

- Die Funktion $\phi : R \times \mathcal{P}(T) \rightarrow \mathbb{N}$ definiert die Zugehörigkeiten von Ressourcen zu Clustern. Für jede Ressource $r \in R$ und jedes Cluster $C \in \mathcal{P}(T)$ gilt:

$$\phi(r, C) = |\{u \in U \mid \forall t \in C : (u, t, r) \in Y\}| \tag{4.3}$$

Da $\prec$ und ϕ auf der gesamten Menge $\mathcal{P}(T)$ definiert sind, können sie ohne Anpassung zur Beschreibung jedes Tagset-Clusterings verwendet werden.

Verschiedene Tagset-Clusterings zu einer Folksonomy unterscheiden sich in der Auswahl der Menge C. Dabei darf C nicht beliebig aus $\mathcal{P}(T)$ ausgewählt werden, sondern muss die folgenden Bedingungen erfüllen:

Definition 4.4. (Bedingungen für die Clustermenge)
Eine Clustermenge $\mathsf{C} \subseteq \mathcal{P}(T)$ muss folgenden Bedingungen genügen:

$$\emptyset \in \mathsf{C} \tag{4.4a}$$
$$\forall D \in \mathsf{C} \text{ mit } D \neq \emptyset : \exists C : C \prec D \tag{4.4b}$$
$$\forall C \in \mathsf{C} : \exists r \in R : \phi(r, C) > 0 \tag{4.4c}$$

Bedingung (4.4a) fordert, dass die leere Menge als kleinstes Element in jedem Clustering vorhanden sein muss. Bedingung (4.4b) stellt sicher, dass das Cluster für die leere Menge durch $\prec$ transitiv mit jedem anderen Cluster in Beziehung steht. Dies führt dazu, dass das Clustering nur aus einer einzigen Zusammenhangskomponente besteht. Bedingung (4.4c) sorgt dafür, dass jedem Cluster durch ϕ mindestens eine Ressource aus R zugewiesen ist.

Ausgehend vom Tagset-Verband $\langle \mathcal{P}(T), \cap, \cup \rangle$ bezüglich einer Folksonomy, macht ein Tagset-Clustering einige Veränderungen. Zunächst wird die betrachtete Tagset-Menge von $\mathcal{P}(T)$ auf eine Auswahl $\mathsf{C} \subseteq \mathcal{P}(T)$ eingeschränkt, wobei C die Eigenschaften (4.4a - 4.4c) haben muss. Dadurch geht die Verbandseigenschaft verloren, da für jede Teilmenge von Clustern aus C zwar ein Infimum, aber nicht in jedem Fall ein Supremum gefunden werden kann.

Die Clustermenge erfüllt zusammen mit der Mengeninklusion $\subseteq$ nur noch die Bedingungen für einen unteren Halbverband.

Korollar 4.5. Gegeben sei ein Tagset-Clustering mit einer Clustermenge $\mathsf{C} \in \mathcal{P}(T)$. Die Clustermenge C bildet einen unteren Halbverband $\langle \mathsf{C}, \cap \rangle$, da die folgende Bedingung erfüllt ist:

$$\forall C, D \in \mathsf{C} : C \cap D \in \mathsf{C} \tag{4.5}$$

Die Gültigkeit dieser Bedingung folgt direkt aus den Eigenschaften 4.4a und 4.4b für die Clustermenge.

Als Nächstes wird die Relation $\subseteq\colon \mathcal{P}(T) \times \mathcal{P}(T)$ auf die Teilrelation $\prec$ eingeschränkt. Diese Einschränkung geschieht aber auf eine solche Weise, dass die erzeugte Relation $\prec$ immer noch die gesamte Information von $\subseteq$ enthält.

Korollar 4.6. Der transitive Abschluss der Relation $\prec\colon \mathcal{P}(T) \times \mathcal{P}(T)$, die die einzelnen Cluster eines Tagset-Clusterings hierarchisch anordnet, bildet genau die Mengeninklusionsrelation $\subseteq\colon \mathcal{P}(T) \times \mathcal{P}(T)$.

$$\text{trans}(\prec) = \bigcup_{n \geq 1} \prec^n = \prec^1 \cup \prec^2 \cup \prec^3 \ldots = \subseteq \tag{4.6}$$

Schließlich reichert die Funktion ϕ die reine Ordnungsstruktur auf den Tagsets mit zusätzlicher Information über die Tag-Zuweisungen der Benutzer an. Diese Information wird durch die Relation Y aus der Folksonomy geliefert.

Wie bereits erwähnt, unterscheiden sich die verschiedenen Tagset-Clusterings zu einer Folksonomy in ihrer Auswahl der Clustermenge. Die Gesamtheit aller gültigen Clustermengen stellt den Suchraum dar, in dem nach geeigneten Navigationsstrukturen gesucht wird.

Definition 4.7. (Suchraum)
Gegeben sei eine Folksonomy $\mathbb{F} = (U, T, R, Y)$. Die Menge $\mathfrak{C}$ aller gültigen Clustermengen ist definiert als

$$\mathfrak{C} = \{\mathsf{C} \in \mathcal{P}(T) \mid \mathsf{C} \text{ erfüllt die Bedingungen (4.4a - 4.4c)}\} \tag{4.7}$$

Es gilt $\mathfrak{C} \subseteq \mathcal{P}(\mathcal{P}(T))$ und $\mathfrak{C}$ ist eine endliche Menge.

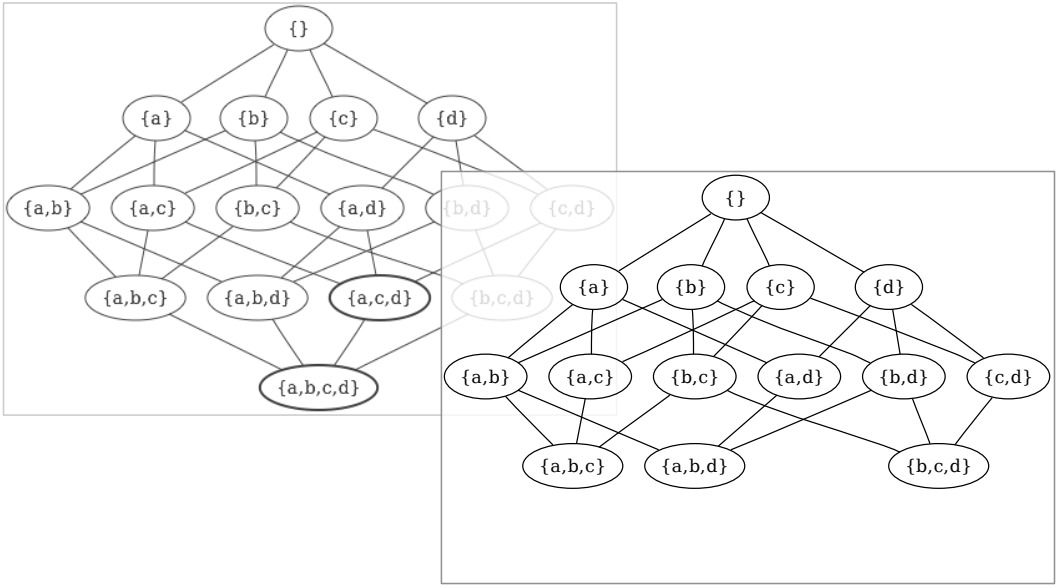


Abbildung 4.1: Auf Grundlage des Tagset-Verbands (hinten) zu einer Folksonomy wird das TS-Clustering (vorne) erstellt. Die Tagsets $\{a, c, d\}$ und $\{a, b, c, d\}$ werden nicht benutzt und fallen daher weg.

Der Suchraum $\mathfrak{C}$ enthält als kleinstes Element die Menge $\{\emptyset\}$ und als größtes Element die Clustermenge TS des sogenannten vollständigen Tagset-Clusterings.

Definition 4.8. (Vollständiges Tagset-Clustering)
Gegeben sei eine Folksonomy $\mathbb{F} := (U, T, R, Y)$. Die Menge $\mathsf{TS} \subseteq \mathcal{P}(T)$ ist definiert als

$$\mathsf{TS} = \{C \in \mathcal{P}(T) \mid \exists (u, r) \in U \times R : \forall t \in C : (u, t, r) \in Y\} \tag{4.8}$$

Das Tagset-Clustering, welches die Menge TS als Clustermenge verwendet, wird als vollständiges Tagset-Clustering bezeichnet. Man schreibt als Abkürzung auch TS-Clustering.

Die Menge TS ist die größtmögliche Auswahl von Tagsets aus $\mathcal{P}(T)$, für die gilt, dass jedes Tagset mindestens einmal verwendet wurde, d.h. dass es mindestens einer Ressource $r \in R$ von mindestens einem Benutzer $u \in U$ zugewiesen wurde.

4.2 Frequent-Tagset-Clusterings

Der erste Schritt der Suche nach einer Navigationsstruktur ist die Einschränkung der Tagset-Menge TS gemäß der Häufigkeit, mit der die einzelnen Tagsets von den Benut-

zern des Tagging-Systems verwendet wurden. Man legt einen Mindesthäufigkeitswert σ zugrunde und wählt eine Teilmenge FTS aus der Menge TS aus, in der alle seltenen Tagsets nicht mehr vorkommen.

Die Auswahl von Tagsets nach ihrer Häufigkeit ist inspiriert vom Frequent-Itemset-Mining, welches in Kapitel 3.2 vorgestellt wurde. FIM betrachtet zwei Konzepte: Transaktionen und Items. Eine Folksonomy modelliert den Datenbestand des Tagging-Systems aber auf Grundlage von drei Konzepten: Benutzer, Tags und Ressourcen. Um ein FIM-Verfahren zur Auswahl der Tagsets heranziehen zu können, muss zuerst eine Projektion gefunden werden.

Definition 4.9. (Häufigkeiten von Tagsets)
Es gibt verschiedene Möglichkeiten den Begriff der Häufigkeit von Tagsets zu definieren. Sei T' ein Tagset:

Die Funktion $\text{supp}_U : \mathcal{P}(T) \to \mathbb{N}_0$ bestimmt die Häufigkeit als die Anzahl der Benutzer aus U, die jedes Tag aus T' mindestens einmal verwendet haben:

$$\text{supp}_U(T') = |\{u \in U \mid \forall t \in T' : \exists r \in R : (u, t, r) \in Y\}| \tag{4.9}$$

Die Funktion $\text{supp}_R : \mathcal{P}(T) \to \mathbb{N}_0$ bestimmt die Häufigkeit als die Anzahl Ressourcen aus R, denen jedes Tag aus T' mindestens einmal zugewiesen wurde:

$$\text{supp}_R(T') = |\{r \in R \mid \forall t \in T' : \exists u \in U : (u, t, r) \in Y\}| \tag{4.10}$$

Die Funktion $\text{supp}_{U\times R} : \mathcal{P}(T) \to \mathbb{N}_0$ bestimmt die Häufigkeit als die Anzahl Tupel (u, r) von Benutzern und Ressourcen, die alle Tags aus T' aufweisen:

$$\text{supp}_{U\times R}(T') = |\{(u, r) \in U \times R \mid \forall t \in T' : (u, t, r) \in Y\}| \tag{4.11}$$

Eine Menge T' heißt *Frequent Tagset*, wenn ihre Häufigkeit größer oder gleich einem Wert $\sigma \in \mathbb{N}$ ist. Wir definieren folgende Mengen von Frequent Tagsets:

$$\mathsf{FTS}_U = \{T' \in \mathsf{TS} \mid \text{supp}_U(T') \geq \sigma\} \tag{4.12a}$$
$$\mathsf{FTS}_R = \{T' \in \mathsf{TS} \mid \text{supp}_R(T') \geq \sigma\} \tag{4.12b}$$
$$\mathsf{FTS}_{U\times R} = \{T' \in \mathsf{TS} \mid \text{supp}_{U\times R}(T') \geq \sigma\} \tag{4.12c}$$

Durch eine Projektion von dreidimensionalen Daten auf zweidimensionale Daten geht zwangsläufig Information verloren. Daher ist bei der Auswahl des Häufigkeitsbegriffs Vorsicht geboten.

Bestimmt man die Häufigkeit mit der Funktion $\text{supp}_{U\times R}$, werden unter Umständen Tagsets als häufig bestimmt, die von einer Minderheit der Benutzer auf eine sehr große Anzahl von Ressourcen vergeben wurden.

Um zu verhindern, dass einzelne Benutzer einen zu großen Einfluss auf die Häufigkeit eines Tagsets haben, kann stattdessen eine der Funktionen supp_U und supp_R verwenden. Allerdings geht hier die Information verloren, wievielen Ressourcen ein Benutzer ein Tagset zugewiesen hat bzw. von wievielen Benutzern eine Ressource mit einem bestimmten Tagset ausgezeichnet wurde. Ist einer der drei Häufigkeitsbegriffe ausgewählt, kann die erste Einschränkung des TS-Clusterings definiert werden.

Definition 4.10. (Frequent-Tagset-Clustering)
Gegeben sei eine Folksonomy $\mathbb{F} := (U, T, R, Y)$, eine Mindesthäufigkeit $\sigma \in \mathbb{N}$ und eine gemäß dieses Wertes bestimmte Menge $\mathsf{FTS} \subseteq \mathsf{TS}$ von Frequent Tagsets mit $\mathsf{FTS} \in \{\mathsf{FTS}_U, \mathsf{FTS}_R, \mathsf{FTS}_{U\times R}\}$.

Ein Tagset-Clustering, welches die Menge FTS als Clustermenge verwendet, wird als Frequent-Tagset-Clustering bezeichnet. Alternativ wird auch FTS-Clustering geschrieben.

Korollar 4.11. Gegeben sei eine Folksonomy $\mathbb{F}$ und eine gemäß eines Supports σ erzeugte Clustermenge FTS von Frequent Tagsets. Dann erfüllt FTS die folgende Eigenschaft:

Für jedes Tagset $C \in \mathsf{FTS}$ gilt:

$$C \in \mathsf{FTS} \Rightarrow \{D \mid D \subseteq C\} \subseteq \mathsf{FTS} \tag{4.13}$$

Korollar 4.11 besagt, dass für jedes Tagset in FTS auch alle seine Teilmengen enthalten sind. Die Gültigkeit dieser Aussage folgt direkt aus der Monotonie-Eigenschaft des Supports (siehe Satz 3.1), gemäß der, wenn ein Tagset häufig ist, auch alle seine Teilmengen häufig sein müssen.

Um die Menge FTS zur Bildung eines Tagset-Clustering verwenden zu können, muss sichergestellt werden, dass die für Clustermengen geforderten Bedingungen erfüllt sind.

Satz 4.12. Gegeben sei eine Folksonomy $\mathbb{F}$. Jede Menge FTS von Frequent Tagsets, die gemäß eines Supports σ erzeugt wurde, erfüllt die Bedingungen (4.4a - 4.4c), bildet also eine gültige Clustermenge.

Beweis. Die Vorbedingung, dass alle Tagsets aus FTS häufig sind, führt dazu, dass die geforderten Bedingungen erfüllt sind:

- Bedingung (4.4a) gilt, da das leere Tagset $\emptyset$ immer häufig ist.
- Bedingung (4.4b) folgt direkt aus Korollar 4.11.
- Bedingung (4.4c) gilt, da Tagsets nur dann als häufig ermittelt werden, wenn zumindest eine Ressource existiert, der sie zugewiesen wurden.

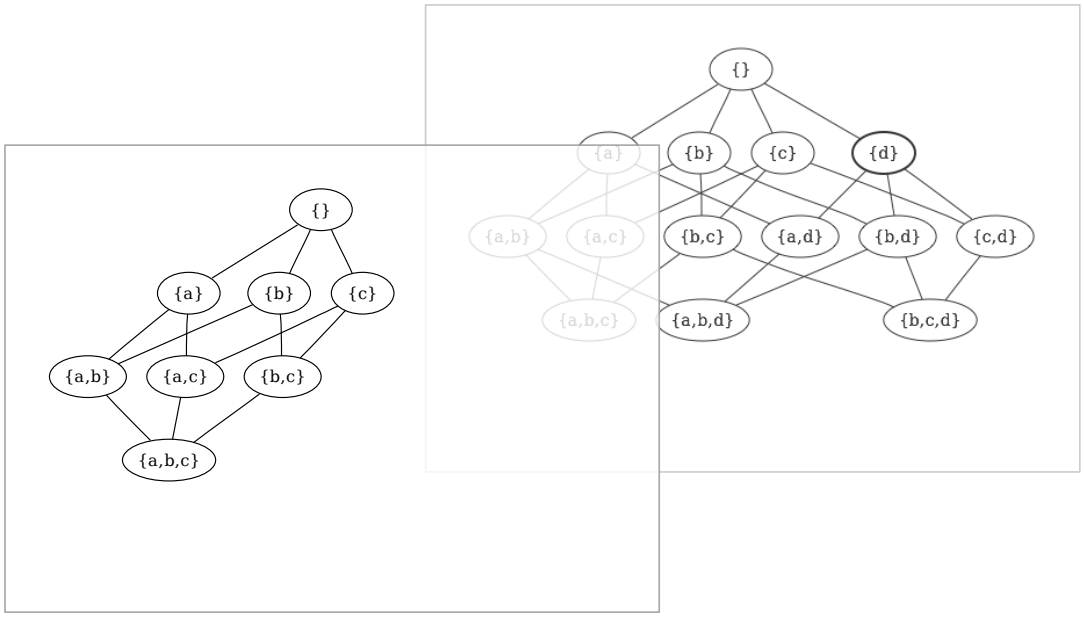


Abbildung 4.2: Aus einem TS-Clustering (hinten) ergibt sich ein FTS-Clustering (vorne), in dessen Clustermenge das Tagset $\{d\}$ wegen eines zu geringen Supports nicht mehr enthalten ist. Wegen der Monotonie der Häufigkeit von Tagsets fallen auch alle Tagsets im darunterliegenden Teilbaum weg.

Tagset-Support als Auswahlstrategie für Clustermengen

Um den Suchraum aller Clustermengen zu durchsuchen, muss eine Strategie gefunden werden, nach der einzelne Kandidaten erzeugt werden. Der erste Ansatz ist die sukzessive Wahl von Supportwerten $\sigma \in \mathbb{N}$ und die Erzeugung der entsprechenden FTS-Clusterings.

Es wird nicht der gesamte Suchraum $\mathfrak{C}$ betrachtet, sondern nur ein Teil $\mathfrak{C}_{freq}$ davon.

$$\mathfrak{C}_{freq} = \{\mathsf{C} \mid \text{alle } C \in \mathsf{C} \text{ häufig und } \mathsf{C} \text{ erfüllt Gleichung (4.13)}\} \subseteq \mathfrak{C} \tag{4.14}$$

In Abbildung 4.2 wird dargestellt, welche Veränderungen der Struktur eines Clusterings durch eine Auswahl nach dem Support verursacht werden. Da für jeden Kandidaten Gleichung (4.13) erfüllt sein muss, verursacht eine Veränderung des Supports immer das Hinzufügen bzw. Entfernen ganzer Teilbäume von Tagsets. Der Suchraum ist sehr eingeschränkt, die Suche also sehr grob.

Daher wird der Tagset-Support nicht als alleinige Auswahlstrategie verwendet, sondern sie bildet nur die erste Stufe in einem zweistufigen Prozess. Es wird eine Voraus-

wahl für ein Clustering getroffen, dessen sämtliche Teilstrukturen in der zweiten Stufe unter Verwendung einer anderen Strategie betrachtet werden.

4.3 Filtered-Frequent-Tagset-Clusterings

Auf Grundlage des in der ersten Stufe der Suche vorausgewählten FTS-Clusterings werden nun in der zweiten Stufe Teilstrukturen dieses Clusterings als mögliche Kandidaten für eine Navigationsstruktur betrachtet. Im Gegensatz zur Betrachtung von Clusterings auf Basis verschiedener Supportwerte, werden nun sehr viel mehr Möglichkeiten in die Betrachtung einbezogen. Jeder Kandidat muss lediglich die Eigenschaften (4.4a - 4.4c) erfüllen.

Definition 4.13. (Filtered-Frequent-Tagset-Clustering)
Gegeben sei eine Folksonomy $\mathbb{F}$ und ein FTS-Clustering mit einer Clustermenge FTS, das für einen Support von $\sigma \in \mathbb{N}$ bestimmt wurde.

Sei $\mathsf{FFTS} \subseteq \mathsf{FTS}$ eine Auswahl von Tagsets aus der Menge $\mathsf{FTS} \subseteq \mathcal{P}(T)$. Für FFTS seien die Eigenschaften (4.4a - 4.4c) erfüllt.

Dann erhält man ein sogenanntes Filtered-Frequent-Tagset-Clustering, kurz FFTS-Clustering.

Aus einer beliebigen Teilmenge FTS' von FTS wird eine gültige Clustermenge FFTS erzeugt, indem alle nicht durch $\prec$ mit dem $\emptyset$-Tagset in Relation stehenden Tagsets aus FTS' entfernt werden.

Auswahl der Clustermenge durch Filtern

Die Wahl des FTS-Clusterings in der ersten Stufe der Suche beschränkt den Suchraum, da im Weiteren nur Teilclusterings dieses Clusterings betrachtet werden. Der Support sollte daher so gering wie technisch möglich gewählt werden, um die Betrachtung eines möglichst großen Teils des Gesamtsuchraum $\mathfrak{C}$ zu ermöglichen. Die Forderung nach Beschränkung auf Strukturen mit der Eigenschaft (4.13), wie sie in Schritt 1 bestand, ist hier aufgehoben. Der verbleibende Teilraum des Suchraums wird also vollständig betrachtet.

$$\mathfrak{C}_\sigma = \{\mathsf{C} \mid \mathsf{C} \subseteq \mathsf{FTS} \text{ und } \mathsf{C} \text{ erfüllt Bedingungen (4.4a - 4.4c)}\} \tag{4.15}$$

Die Menge FTS wurde für einen Support von $\sigma \in \mathbb{N}$ bestimmt.

Wie in Abbildung 4.3 zu sehen ist, sind durch Filterung sehr viel feinere Anpassungen der Struktur des Clusterings möglich. Das Hinzufügen und Entfernen einzelner Tagsets, im Gegensatz zu ganzen Teilbäumen in Schritt 1, ist jetzt möglich.

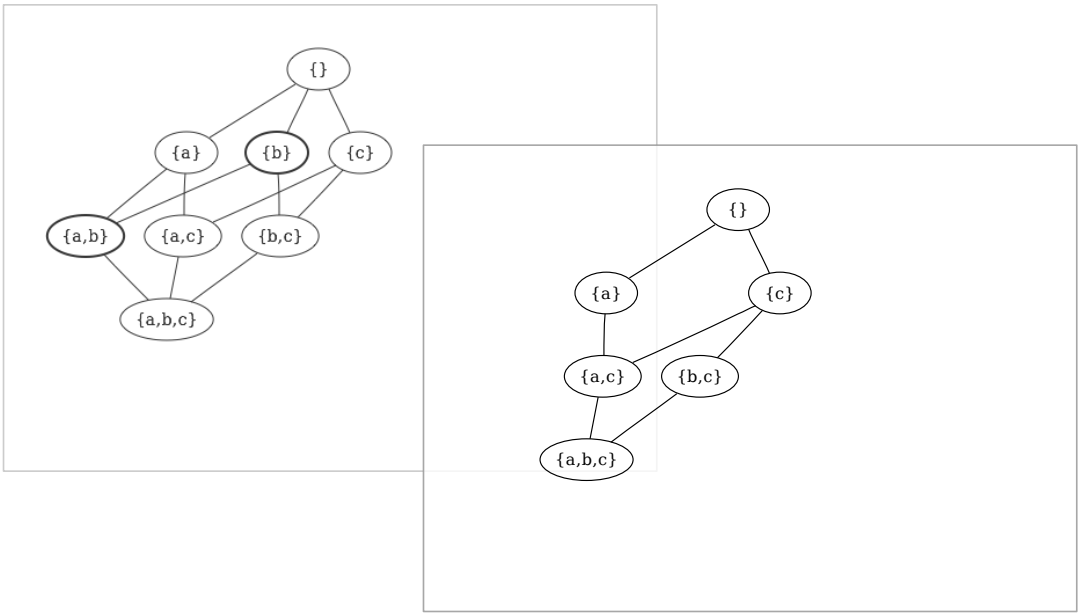


Abbildung 4.3: Aus einem FTS-Clustering (hinten) ergibt sich durch Deselektion einiger Tagsets ein FFTS-Clustering (vorne).

4.4 Auswahl der Clustermenge als multikriterielles Suchproblem

Wie in Kapitel 2 beschrieben, stellt der Benutzer verschiedene konkurrierende Anforderungen an eine Navigationsstruktur. Das Verfahren soll daher eine Menge alternativer Clusterings erzeugen. Einige Bestandteile des Verfahrens wurden in den letzten Abschnitten vorgestellt. Es fehlt die formale Definition der Bewertung von Clusterings, auf dessen Grundlage in der zweiten Stufe der Suche die Auswahl der Menge alternativer Clusterings vorgenommen wird.

Definition 4.14. (Fitnessfunktion)
Gegeben sei eine Folksonomy $\mathbb{F} = (U, T, R, Y)$. Eine Fitnessfunktion ist eine Funktion der Signatur $\mathfrak{C} \rightarrow \mathbb{R}$, die jeder gültigen Clustermenge $\mathsf{C} \in \mathfrak{C}$ eine reellwertige Bewertung zuordnet.

Im letzten Kapitel wurde der Begriff „Fitnessfunktion“ im Kontext der Genetischen Algorithmen (siehe Kapitel 3.4) als Funktion zur Abbildung von Bitvektoren auf skalare Fitnesswerte definiert. Im Folgenden bezeichnet er jedoch immer eine Funktion

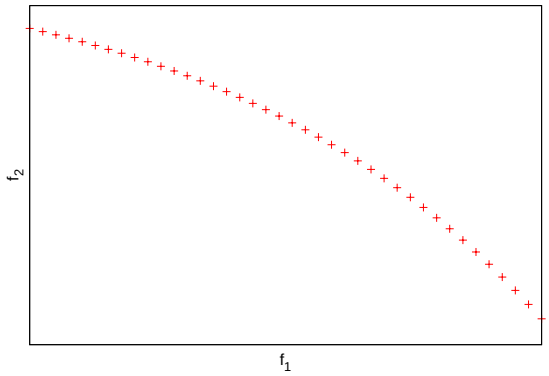


Abbildung 4.4: Beispielhafte Pareto-Front aller nicht-dominierten FFTS-Clusterings. f_1 und f_2 sind beliebige, konkurrierende Fitnessfunktionen.

zur Bewertung von Clustermengen. Ebenso bezieht sich der Begriff „Individuum“ nicht mehr auf einen Bitvektor, sondern auf eine Clustermenge.

Da mehrere, widersprüchliche Anforderungen existieren, die gleichzeitig möglichst gut erfüllt werden sollen, reicht eine einzelne Fitnessfunktion nicht aus. Theoretisch muss für jede Anforderung eine Funktion definiert werden. Da aber das Ergebnis des Verfahrens eine Alternativenmenge ist und das Betrachten der verschiedenen Alternativen für den Benutzer verständlich bleiben soll, werden die Anforderungen auf eine solche Weise zusammengefasst, dass sie mit nur zwei Funktionen beschrieben werden können.

Die vorliegende Aufgabenstellung wird im Folgenden als ein multikriterielles Optimierungsproblem aufgefasst. Die Grundbegriffe der multikriteriellen Optimierung wurden in Kapitel 3.5 eingeführt.

Definition 4.15. (Pareto-optimale Clustermengen)
Gegeben sei eine Folksonomy $\mathbb{F} = (U, T, R, Y)$. Das FTS-Clustering wurde für einen Support $\sigma \in \mathbb{N}$ bestimmt. Der Lösungsraum für die Clustermengen ist also $\mathfrak{C}_\sigma$. Die Funktionen f_1 und f_2 seien zwei Fitnessfunktionen für Clustermengen, welche zu einer Vektorfunktion $\vec{f}$ (vgl. Definition 3.8) zusammengefasst werden.

Die Menge $\mathfrak{C}^*_\sigma \subseteq \mathfrak{C}_\sigma$ enthält die bezüglich $\vec{f}$ pareto-optimalen Clustermengen.

$$\mathfrak{C}^*_\sigma = \{\mathsf{C} \in \mathfrak{C}_\sigma \mid \nexists \mathsf{D} \in \mathfrak{C}_\sigma : \vec{f}(\mathsf{D}) \succeq \vec{f}(\mathsf{C})\} \tag{4.16}$$

Die Relation $\succeq$ drückt die Pareto-Dominanz einer Clustermenge gegenüber einer anderen aus (vgl. Definition 3.9). $\mathfrak{C}^*_\sigma$ wird im Folgenden als Pareto-Front bezeichnet.

Abbildung 4.4 zeigt die Menge aller pareto-optimalen Lösungen anhand eines Beispielproblems. Die verschiedenen Clusterings sind gemäß der berechneten Bewertun-

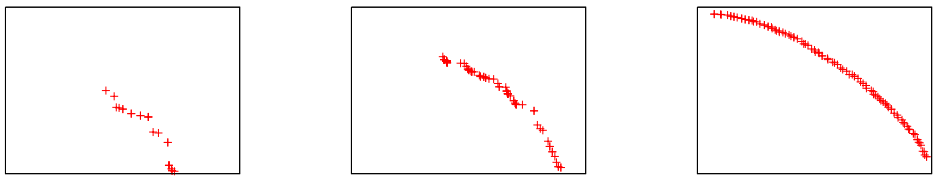


Abbildung 4.5: Ein Beispiel für die Approximation der Pareto-Front durch NSGA-II. Die Abbildungen zeigen die Menge der nicht-dominierten Individuen nach 5, 20 und 300 Generationen.

gen für zwei beliebige Fitnessfunktionen f_1 und f_2 als Punkte im $\mathbb{R}^2$ dargestellt. Das Clustering, welches durch den Punkt links oben repräsentiert wird, hat die maximale Bewertung bezüglich des Ziels f_2 und aufgrund der Konkurrenz der Funktionen die minimale Bewertung bezüglich des Ziels f_1. Je besser, von diesem Punkt ausgehend, ein Clustering das Ziel f_1 erfüllt, umso schlechter erfüllt es das Ziel f_2. Der Punkt rechts unten schließlich erfüllt das Ziel f_1 am besten und demzufolge das Ziel f_2 am schlechtesten.

Die Menge $\mathfrak{C}_\sigma^*$ enthält alle nicht-dominierten FFTS-Clusterings. In der Regel ist $\mathfrak{C}_\sigma^*$ unbekannt und kann durch ein Lösungsverfahren nur approximiert werden. Die approximierte Menge soll keine dominierten Elemente enthalten und ihre Elemente sollen zum einen möglichst nahe an der Pareto-Front $\mathfrak{C}_\sigma^*$ liegen und zum anderen möglichst gleichmäßig im Raum verteilt sein. Der Benutzer bekommt diese Menge als Alternativenmenge zurückgeliefert.

In dieser Arbeit wird zur Approximation von $\mathfrak{C}_\sigma^*$ das evolutionäre Verfahren NSGA-II von Deb et al. [9] verwendet, das in Kapitel 3.5.2 beschrieben wurde. In Abbildung 4.5 ist die schrittweise Approximation der Pareto-Front dargestellt, wie sie von NSGA-II durchgeführt wird. Bei dem Verfahren handelt es sich um einen GA. Eine Einführung in die Arbeitsweise von GAs wurde in Kapitel 3.4 gegeben. Intern arbeitet NSGA-II, wie für einen GA üblich, mit Bitvektoren. Im Ablauf erzeugt das Verfahren Kandidaten für Clustermengen in Form von Bitvektoren. Um diese Clustermengen gemäß der zwei Fitnessfunktionen bewerten zu können, ist eine Abbildung von Bitvektoren auf Clustermengen notwendig.

Basis der Abbildung bildet das FTS-Clustering, welches in Schritt 1 der Suche vorausgewählt wurde. Die Länge der Bitvektordarstellung ist $|\mathsf{FTS}| - 1$. In Abbildung 4.6 ist die Repräsentation eines FFTS-Clusterings als Bitvektor dargestellt. Jede 1 oder 0 im Vektor bedeutet, dass das entsprechende Tagset in der Clustermenge FFTS vorhanden bzw. nicht vorhanden ist. Das Tagset $\emptyset$ wird im Vektor nicht repräsentiert, da es aufgrund der Anforderungen, die an eine Clustermenge gestellt werden, niemals ausgeblendet werden darf.

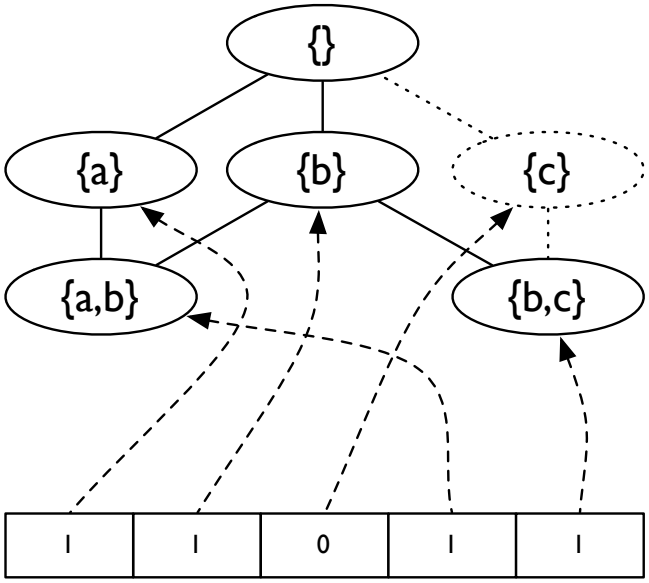


Abbildung 4.6: Darstellung eines FFTS-Clusterings als Bitvektor. Alle Tagsets mit Ausnahme des ∅-Tagsets werden durch eine Stelle im Bitvektor repräsentiert.

Ablauf des Verfahrens: Mit einer Folksonomy als Eingabe wird in der ersten Stufe der Suche für einen möglichst geringen Support $\sigma \in \mathbb{N}$ ein FTS-Clustering erstellt. Dieses Clustering besteht in der Darstellung als Bitvektor ausschließlich aus Einsen. Es werden zwei Fitnessfunktionen definiert und das Verfahren NSGA-II angewandt. Während NSGA-II arbeitet, werden sukzessiv Bitvektoren generiert, die durch die Abbildungsfunktion auf FFTS-Clusterings abgebildet und mit den Fitnessfunktionen ausgewertet werden. Es ist möglich, dass Bitvektoren erzeugt werden, die auf eine ungültige Clustermenge abgebildet werden. In diesem Fall wird die Clustermenge repariert, indem alle Cluster, die nicht mit dem ∅-Cluster verbunden sind, aus der Menge entfernt werden. Die Abbildung von Bitvektoren auf Clustermengen ist demnach surjektiv. Nach einer vorgegebenen Anzahl von Generationen bricht das Verfahren ab und alle nicht-dominierten Individuen der aktuellen Generation werden ausgegeben. Die Ausgabe wird dem Benutzer in Form eines zweidimensionalen Koordinatensystems präsentiert, in das jedes Individuum gemäß seiner beiden Bewertungen eingezeichnet ist. Jedes Individuum kann vom Benutzer in seiner Darstellung als Tagset-Clustering betrachtet werden.

4.5 Bewertungsfunktionen für Tagset-Clusterings

Ester et al. stellen in [12] ein Verfahren zum hierarchischen Clustern von Textdokumenten vor, welches bereits in Kapitel 3.3 beschrieben wurde.

Nach Bestimmung der Worthäufigkeiten (*term frequencies*) in den Texten wird unter Verwendung eines FIM-Verfahrens eine Menge von *Frequent Termsets* bestimmt. Ein Termset ist eine Menge von Wörtern, die mit einer bestimmten Mindesthäufigkeit in den Texten vorkommen. Die Termsets werden hierarchisch in einer baumförmigen Clusterstruktur angeordnet. Jedes Termset ist das Label für genau ein Cluster und hat genau einen Elter-Cluster. Jedes Textdokument wird dem passendsten Cluster zugeordnet. Sehr ähnliche Cluster werden verschmolzen.

Ziel des Verfahrens ist es, eine einfache und verständliche Clusterstruktur zu erstellen. Dies ist eines der Ziele, das in Kapitel 2.6 für Navigationsstrukturen definiert wurde.

Die durch das Verfahren erzeugte Clusterstruktur hat zum einen die Eigenschaft, dass die Anordnung der Cluster baumförmig ist, und zum anderen, dass keine Überlappung zwischen den Clustern existiert, eine Ressource also nur in einem Cluster vorkommt. Die Menge der Ressourcen wird auf der obersten Ebene sowie in jedem Teilbaum partitioniert.

Auf dieser Grundlage wird im folgenden Abschnitt 4.5.1 die erste Fitnessfunktion für das multikriterielle Optimierungsproblem formuliert. Im Abschnitt 4.5.2 folgt dann die zweite Fitnessfunktion.

Zuvor ist die Definition einiger allgemeiner Hilfsfunktionen notwendig, die im Folgenden verwendet werden. Die Funktion $\text{res} : \mathcal{P}(\mathcal{P}(T)) \rightarrow \mathbb{R}$ berechnet für eine Menge M von Clustern die Menge der insgesamt enthaltenen Ressourcen $r \in R$:

$$\text{res}(\mathsf{M}) = \bigcup_{M \in \mathsf{M}} \{r \mid \phi(r, M) > 0\} \tag{4.17}$$

Die Funktion $\text{depth} : \mathfrak{C} \rightarrow \mathbb{N}$ berechnet die Tiefe eines Tagset-Clusterings:

$$\text{depth}(\mathsf{C}) = \max_{C \in \mathsf{C}} |C| \tag{4.18}$$

4.5.1 Bewertung der Überlappung der Cluster (Overlap)

Das Clustering soll eine einfache Struktur haben. Wir definieren als erste Fitnessfunktion den Overlap, um die Optimierung in diese Richtung zu lenken:

Definition 4.16. (Simple Overlap)
Gegeben sei ein Tagset-Clustering mit einer Clustermenge C zu einer Folksonomy $\mathbb{F} = (U, R, T, Y)$.

Dann ist die Fitnessfunktion overlap : $\mathfrak{C} \rightarrow \mathbb{R}$ definiert als:

$$\text{overlap}(\mathsf{C}) = \frac{1}{\text{depth}(\mathsf{C})} \cdot \sum_{i=1}^{\text{depth}(\mathsf{C})} \frac{|\text{res}(\mathsf{C}_{|i})|}{\sum\limits_{C \in \mathsf{C}_{|i}} |\text{res}(\{C\})|} \tag{4.19}$$

Ist $\text{depth}(\mathsf{C}) = 0$, so gilt $\text{overlap}(\mathsf{C}) = 1$.

Die Funktion overlap misst die durchschnittliche Überlappung der Ressourcenmengen der Cluster einer Ebene. Idealerweise kommt jede Ressource aus R pro Ebene nur in einem Cluster vor. In diesem Fall liefert die Funktion den Wert 1. Je mehr ein Clustering von diesem Idealfall abweicht, umso mehr wandert seine Bewertung in Richtung 0.

Korollar 4.17. Gegeben seien zwei Clusterings mit den Clustermengen C_1 und C_2. Es ist $\mathsf{C}_1 \subset \mathsf{C}_2$. Dann gilt:

$$\text{overlap}(\mathsf{C}_1) \geq \text{overlap}(\mathsf{C}_2) \tag{4.20}$$

Das bedeutet, dass ein Clustering, welches neben allen Clustern aus C_1 noch zusätzliche Cluster enthält, nur einen größeren Overlap haben kann (d.h. die Funktion overlap liefert einen kleineren Wert). Ein zusätzliches Cluster kann niemals zum Absinken des Overlaps führen.

Für jede Ressource ist bekannt, wieviele Benutzer sie getaggt haben. Diese Information lässt sich dazu verwenden, um eine gewichtete Version des Overlap zu definieren:

Definition 4.18. (Weighted Overlap)
Gegeben sei ein Tagset-Clustering mit einer Clustermenge $\mathsf{C} \in \mathfrak{C}$. Das Gewicht einer Ressource $r \in R$ sei definiert als die Anzahl von Benutzern, die diese Ressource getaggt haben. Also wird eine Funktion $w : R \rightarrow \mathbb{N}_0$ definiert als $w(r) = \phi(r, \emptyset)$.

Dann ist die Fitnessfunktion w-overlap : $\mathfrak{C} \rightarrow \mathbb{R}$ definiert als:

$$\text{w-overlap}(\mathsf{C}) = \frac{1}{\text{depth}(\mathsf{C})} \cdot \sum_{i=1}^{\text{depth}(\mathsf{C})} \frac{\sum\limits_{r \in \text{res}(\mathsf{C}_{|i})} w(r)}{\sum\limits_{\substack{C \in \mathsf{C}_{|i} \\ r \in \text{res}(\{C\})}} w(r)} \tag{4.21}$$

Ist $\text{depth}(\mathsf{C}) = 0$, so gilt $\text{w-overlap}(\mathsf{C}) = 1$.

Die Funktion w-overlap verfolgt die gleiche Idee wie die Funktion overlap. Während Letztere jede Resource aus R gleichwertig behandelt, wird hier die Beliebtheit einer Ressource miteinbezogen. Die Bewertung wird für eine Ressource, die in vielen Clustern auf einer Ebene vorkommt, um einen Wert proportional zur Beliebtheit der Ressource reduziert.

Analog zu Korollar (4.17) der ungewichteten Variante des Overlap, ergibt sich die folgende Eigenschaft für die gewichtete Variante:

Korollar 4.19. Gegeben seien zwei Clusterings mit den Clustermengen C_1 und C_2. Es ist $\mathsf{C}_1 \subset \mathsf{C}_2$. Dann gilt:

$$\text{w-overlap}(\mathsf{C}_1) \geq \text{w-overlap}(\mathsf{C}_2) \tag{4.22}$$

Die ungewichtete Variante des Overlap ist ein Spezialfall der gewichteten Variante, bei der das Gewicht auf den Wert 1 festgelegt ist. Daraus ergibt sich für die beiden Funktionen die folgende Eigenschaft:

Korollar 4.20. Gegeben seien zwei Clusterings mit den Clustermengen C_1 und C_2. Es ist $\mathsf{C}_1 \subset \mathsf{C}_2$. Dann gilt:

$$\text{overlap}(\mathsf{C}_1) > \text{overlap}(\mathsf{C}_2) \Leftrightarrow \text{w-overlap}(\mathsf{C}_1) > \text{w-overlap}(\mathsf{C}_2) \tag{4.23}$$

4.5.2 Bewertung des Abdeckungsgrads (Coverage)

Nachdem wir im letzten Abschnitt die einfache Struktur der Clusterings betrachtet haben, wird nun eine zweite Fitnessfunktion definiert, die den Suchprozess in Richtung möglichst vollständiger Clusterings lenkt.

Definition 4.21. (Simple Coverage)
Gegeben sei ein Tagset-Clustering mit einer Clustermenge $\mathsf{C} \in \mathfrak{C}$ zu einer Folksonomy $\mathbb{F} = (U, R, T, Y)$. Dann ist die Funktion $\text{coverage} : \mathfrak{C} \rightarrow [0, 1]$ wie folgt definiert:

$$\text{coverage}(\mathsf{C}) = \frac{|\text{res}(\mathsf{C}_{|1})|}{|R|} \tag{4.24}$$

In jedem Tagset-Clustering sind im $\emptyset$-Cluster alle Ressourcen aus R enthalten. Da ein Tagset-Clustering als Navigationsstruktur genutzt werden soll, ist es interessant zu bestimmen, wieviele Ressourcen noch in anderen Clustern vorhanden sind. Die Funktion coverage berechnet diesen Wert und bezieht ihn auf die Gesamtanzahl der Ressourcen. Je mehr Ressourcen nicht nur im $\emptyset$-Cluster vorhanden sind, umso größer ist der von coverage errechnete Wert.

Angenommen R enthalte zwei Ressourcen r_1 und r_2. r_1 sei mit a getaggt und r_2 mit b. Für ein Clustering mit der Clustermenge $\{\emptyset, \{a\}, \{b\}\}$ liefert die Funktion coverage dann die Bewertung $\frac{2}{2} = 1$, während Clusterings mit den Clustermengen $\{\emptyset, \{a\}\}$ oder $\{\emptyset, \{b\}\}$ jeweils nur eine Bewertung von $\frac{1}{2}$ erhalten.

Korollar 4.22. Gegeben seien zwei Clusterings mit den Clustermengen C_1 und C_2. Wenn $\mathsf{C}_1 \subset \mathsf{C}_2$, dann gilt:

$$\text{coverage}(\mathsf{C}_1) \leq \text{coverage}(\mathsf{C}_2) \tag{4.25}$$

Die Gültigkeit dieses Korollars ist einleuchtend, wenn man sich überlegt, dass durch ein zusätzliches Cluster keine Ressourcen wegfallen können und die Bezugsgröße $|R|$ konstant bleibt.

Korollar 4.23. Gegeben seien zwei Clusterings mit den Clustermengen C_1 und C_2. Es ist $\mathsf{C}_1 \subset \mathsf{C}_2$. Dann gilt :

$$\text{coverage}(\mathsf{C}_1) < \text{coverage}(\mathsf{C}_2) \Leftrightarrow \exists C \in \mathsf{C}_2 \setminus \mathsf{C}_1 : \text{depth}(C) = 1 \tag{4.26}$$

Kommen nur Cluster hinzu, deren Tiefe größer als 1 ist, so bleibt der Wert von coverage konstant. Nur zusätzliche Cluster auf Tiefe 1 führen zu zusätzlicher Abdeckung. Diese Eigenschaft folgt direkt aus der Definition der Fitnessfunktion.

Genauso wie beim Overlap ergibt sich eine gewichtete Version des Coverage-Kriteriums:

Definition 4.24. (Weighted Coverage)
Gegeben sei ein Tagset-Clustering mit einer Clustermenge C zu einer Folksonomy $\mathbb{F} = (U, R, T, Y)$. Das Gewicht einer Ressource $r \in R$ sei definiert als die Anzahl von Benutzern, die diese Ressource getaggt haben. Also wird eine Funktion $w : R \to \mathbb{N}_0$ definiert als $w(r) = \phi(r, \emptyset)$.

Dann ist die Funktion w-coverage : $\mathfrak{C} \to [0, 1]$ definiert als:

$$\text{w-coverage}(\mathsf{C}) = \frac{\sum\limits_{r \in \text{res}(\mathsf{C}_{|1})} w(r)}{\sum\limits_{r \in R} w(r)} \tag{4.27}$$

Die Funktion w-coverage folgt der gleichen Idee, wie die Funktion coverage, bezieht aber zusätzlich die Beliebtheit der Ressourcen ein. Die Beliebtheit einer Ressource ist definiert als die Anzahl von Benutzern, die sie getaggt haben. Sind Ressourcen nur im $\emptyset$-Cluster und nicht in den darunter liegenden Clustern vorhanden, so sinkt der von w-coverage gelieferte Wert proportional zu ihrer Beliebtheit ab.

Wir greifen unser Beispiel noch einmal auf. Sei r_1 diesmal 5-mal mit a getaggt, r_2 aber nur 3-mal mit b. Dann erhält wiederum das Clustering mit der Clustermenge $\{\emptyset, \{a\}, \{b\}\}$ die Bewertung von 1. Die beiden anderen Clusterings werden aber nicht mehr gleich gut bewertet. Da r_1 beliebter ist als r_2, erhält das Clustering mit der Clustermenge $\{\emptyset, \{a\}\}$ mit einem Wert von $\frac{5}{5+3} = \frac{5}{8}$ eine bessere Bewertung als das mit der Clustermenge $\{\emptyset, \{b\}\}$, welches nur einen Wert von $\frac{3}{5+3} = \frac{3}{8}$ erhält.

Analog zur ungewichteten Variante des Coverage-Kriteriums ergeben sich folgende Korollare für die gewichtete Variante:

Korollar 4.25. Gegeben seien zwei Clusterings mit den Clustermengen C_1 und C_2. Es ist $\mathsf{C}_1 \subset \mathsf{C}_2$. Dann gilt:

$$\text{w-coverage}(\mathsf{C}_1) \leq \text{w-coverage}(\mathsf{C}_2) \tag{4.28}$$

Korollar 4.26. Gegeben seien zwei Clusterings mit den Clustermengen C_1 und C_2. Es ist $\mathsf{C}_1 \subset \mathsf{C}_2$. Dann gilt :

$$\text{w-coverage}(\mathsf{C}_1) < \text{w-coverage}(\mathsf{C}_2) \Leftrightarrow \exists C \in \mathsf{C}_2 \setminus \mathsf{C}_1 : \text{depth}(C) = 1 \tag{4.29}$$

4.5.3 Ergebnisse: Overlap versus Coverage

Die meisten klassischen Clustering-Verfahren optimieren implizit die Kriterien Overlap und Coverage, da sie in der Regel eine Partitionierung (Overlap) einer Menge erzeugen, ohne Elemente wegzulassen (Coverage). Mit dem Ziel diese Vorgehensweise auf den Datenbestand eines Tagging-Systems anzuwenden, wurden in den vergangenen beiden Abschnitten diese traditionellen Kriterien für die gegebene Problemdomäne formal definiert. Im Folgenden wird mit Vorgriff auf die empirischen Ergebnisse erläutert, inwieweit die in Kapitel 2.6 formulierten Eigenschaften für gute Clusterings unter Verwendung dieser Kriterien im Suchverfahren erfüllbar sind. Die empirische Untersuchung folgt in Kapitel 5.3.

Die Verwendung von *Weighted Overlap* und *Weighted Coverage* als Optimierungskriterien erzeugt eine Menge alternativer Clustermengen mit folgenden Eigenschaften:

Clustermengen, die das Overlap-Kriterium sehr gut erfüllen, sind schmal und nähern sich, soweit dies auf Grundlage der Daten möglich ist, einer Partionierung der Menge der Ressourcen an. Da sich die Cluster kaum überlappen, der Datenbestand einer Folksonomy aber im Allgemeinen sehr heterogen ist, ist ein relativ geringer Anteil der Ressourcen abgedeckt.

Clustermengen, die das Coverage-Kriterium sehr gut erfüllen, sind sehr breit und enthalten einen großen Anteil der Ressourcen. Dafür überlappen sich die Cluster stärker.

Die auf den Clustermengen definierten Tagset-Clusterings sind sehr flach. Das ist dadurch zu erklären, dass es beide Kriterien nicht erfordern, dass tiefe Clusterings erzeugt werden. Sowohl eine sehr große Abdeckung als auch ein sehr geringer Overlap lassen sich jeweils mit einem Clustering der Tiefe 1 erreichen. Wenn das zugrunde liegende FTS-Clustering sehr stark verbunden ist, werden die erzeugten Clusterings tiefer, da es unwahrscheinlicher wird, dass tiefe Cluster deselektiert werden. Der auftretende Effekt wird in Kapitel 5.3.1 ausführlicher beschrieben.

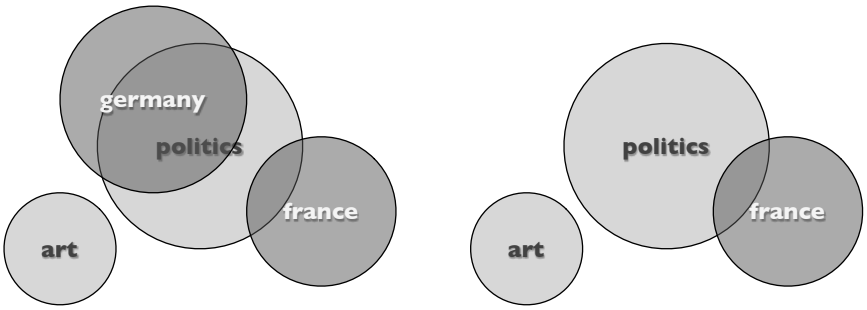


Abbildung 4.7: Auswirkungen der Optimierung des Overlap. Die Tag-Aspekte *Länder* und *Themen* können nicht gleichzeitig existieren. Die Heterogenität des Datenbestands wird zerstört.

Es werden also Mengen alternativer Clustermengen erzeugt, die zum Teil schmal und unvollständig und zum Teil breit und vollständig sind. Die geforderte Eigenschaft der Tiefe wird, je nach Stärke der Verbundenheit des zugrunde liegenden FTS-Clusterings, mehr oder weniger gut erreicht. Allerdings wird die Tiefe nicht explizit erzwungen und kann ohne genaue Kenntnis des verwendeten Datensatzes nicht im Voraus garantiert werden.

Es lässt sich noch ein weiterer Effekt beobachten. Der Datenbestand eines Tagging-Systems ist im Allgemeinen sehr heterogen. Jeder Benutzer kann seine Ressourcen gemäß seiner eigenen Sicht der Welt taggen. Nach einiger Zeit bilden sich Gruppen von Benutzern, die auf ähnliche Weise taggen. Diese, in einem Tagging-System koexistierenden, Sichten der Welt sind ein Grund für Heterogenität der Daten. Weiterhin kann ein Benutzer seine Ressourcen nach unterschiedlichen Aspekten taggen. Beispielsweise kann er auf den gleichen Ressourcen parallel Tags für die geographische Position und Tags für das Thema verwenden. Das führt ebenfalls zu heterogenen Daten.

Die Minimierung des Overlap zerstört die Heterogenität des Datenbestands. Ein Beispiel findet sich in Abbildung 4.7. Einige Benutzer haben Ressourcen nach dem Thema getaggt (art, politics), andere Benutzer haben Ländernamen als Tags verwendet (germany, france). Eine Minimierung des Overlap führt dazu, dass beispielsweise das Tag *germany* wegfällt, weil es sich sehr stark mit dem Tag *politics* überlappt.

Es ist ein großer Wert von Tagging-Systemen, dass verschiedene Meinungen im Datenbestand koexistieren können. Tagging-Systeme sind vor allem deshalb so beliebt, weil sie dem Benutzer eine freie Strukturierung erlauben. Die Heterogenität des Datenbestands sollte in einem Tagset-Clustering erhalten bleiben.

Es müssen neue Optimierungskriterien gefunden werden, die zu Clusterings führen, die nicht nur möglichst schmal und vollständig sind, sondern auch tief, wobei die Tiefe nicht als Nebeneffekt der Eigenschaften des Datensatzes entstehen, sondern explizit

optimiert werden soll. Außerdem soll die Heterogenität der Originaldaten im Clustering möglichst bewahrt bleiben.

In den folgenden beiden Abschnitten werden zwei neue Kriterien vorgestellt, die die zusätzlichen Eigenschaften in der erzeugten Alternativenmenge gewährleisten sollen.

4.5.4 Bewertung der Anzahl der Kinder eines Clusters (Childcount)

Wie wir im letzten Abschnitt gesehen haben, wird durch die Verwendung des Overlaps als Optimierungskriterium die Heterogenität innerhalb der erzeugten Tagset-Clusterings zerstört.

Es wird ein alternatives Kriterium benötigt, das die Heterogenität der Daten erhält und gleichzeitig zu einfachen Strukturen führt. Betrachten wir aber noch einmal genau, was *einfache Struktur* hier eigentlich bedeutet. Eine global einfache Struktur ist nicht realisierbar ohne die Heterogenität der Daten negativ zu beeinflussen. Sie ist aber auch gar nicht notwendig, da man ja eigentlich nur erreichen will, dass der Benutzer eine übersichtliche Anzahl an Alternativen für seinen nächsten Browsing-Schritt erhält. Es zählt demnach nicht die globale, sondern die lokale Einfachheit.

Wir definieren also eine neue Fitnessfunktion, die den Overlap als erstes Kriterium des Optimierungsproblems ersetzt:

Definition 4.27. (Childcount)
Gegeben sei ein Tagset-Clustering mit einer Clustermenge C zu einer Folksonomy $\mathbb{F} = (U, R, T, Y)$.

Sei die Funktion $\text{succ} : \mathsf{C} \to \mathcal{P}(\mathsf{C})$ definiert als

$$\text{succ}(C) = \{D \in \mathsf{C} \mid C \prec D\} \tag{4.30}$$

und damit die Menge $\mathsf{C}' \subseteq \mathsf{C}$ (innere Knoten bzw. Cluster) als

$$\mathsf{C}' = \{C \in \mathsf{C} \mid |\text{succ}(C)| > 0\} \tag{4.31}$$

Dann sind die folgenden Fitnessfunktionen definiert:

$$\text{childcount}_{avg}(\mathsf{C}) = \frac{1}{|\mathsf{C}'|} \sum_{C \in \mathsf{C}'} |\text{succ}(C)| \tag{4.32a}$$

$$\text{childcount}_{max}(\mathsf{C}) = \max_{C \in \mathsf{C}'} |\text{succ}(C)| \tag{4.32b}$$

$$\text{childcount}_{var}(\mathsf{C}) = \frac{1}{|\mathsf{C}'|} \sum_{C \in \mathsf{C}'} (|\text{succ}(C)| - \text{childcount}_{avg}(\mathsf{C}))^2 \tag{4.32c}$$

Jedes Cluster $C \in \mathsf{C}$ steht durch die Relation $\prec$ mit einer Menge von Clustern $\{D \in \mathsf{C} \mid C \prec D\}$ in Beziehung. Die Funktion childcount bestimmt für alle Cluster in der Clustermenge die Kardinalität dieser Menge und berechnet daraus das arithmetische Mittel, das Maximum bzw. die Varianz.

Korollar 4.28. Gegeben seien zwei Clusterings mit den Clustermengen C_1 und C_2. Es ist $\mathsf{C}_1 \subset \mathsf{C}_2$. Dann gilt :

$$\text{childcount}_{max}(\mathsf{C}_1) \leq \text{childcount}_{max}(\mathsf{C}_2) \tag{4.33}$$

Man kann sich leicht überlegen, dass die Eigenschaft für die beiden Funktionen childcount_{avg} und childcount_{var} im Allgemeinen nicht gilt.

Wie zu Anfang dieses Abschnitts erläutert, sind wir an Navigationsstrukturen interessiert, die *lokal einfach* sind. Die Anzahl der Auswahlmöglichkeiten des Benutzers für seinen nächsten Navigationsschritt soll überschaubar gehalten werden. Es interessiert demnach die maximale Anzahl Kind-Cluster eines Clusters, nicht die durchschnittliche Anzahl und auch nicht die Varianz. Wir wählen als Fitnessfunktion die Funktion $\text{childcount}_{max}(\mathsf{C})$ aus.

4.5.5 Bewertung der Nähe zum Original (Completeness)

Wie wir in Abschnitt 4.5.3 gesehen haben, lenkt *Weighted Coverage* die Suche im Lösungsraum in Richtung vollständige Clusterings. Die gefundenen Clusterings waren aber sehr flach, was den in Kapitel 2.6 geforderten Eigenschaften für eine Navigationsstruktur widerspricht.

Es wird eine neue Fitnessfunktion benötigt, die die Suche in Richtung der vollständigen und gleichzeitig tiefen Clusterings im Suchraum lenkt. Die maximale Tiefe eines Clusterings ist durch die Taggings aus der Folksonomy vorgebenen. Wenn keine Ressource existiert, der mehr als vier verschiedene Tags zugewiesen wurde, können das FTS-Clustering und alle Teilstrukturen davon höchstens die Tiefe 4 haben. Wir greifen das in Abschnitt 4.5.4 angesprochene Ziel der Erhaltung der Heterogenität der Daten noch einmal auf. Dieses wird durch das Childcount-Kriterium nicht optimiert, lässt sich aber nun an dieser Stelle zusammen mit Tiefe und Abdeckung zu einem neuen Kriterium kombinieren.

Die Idee ist, dass man versucht ein Clustering zu erzeugen, welches so nahe wie möglich am zugrunde liegenden FTS-Clustering ist. Tiefe, Abdeckung und Heterogenität steigen je mehr sich ein erzeugtes Clustering dem FTS-Clustering nähert.

Es folgt die Definition des Completeness-Kriteriums für Tagset-Clusterings, welches das Coverage-Kriterium als zweites Optimierungskriterium ersetzt.

Definition 4.29. (Completeness)
Gegeben sei eine Folksonomy $\mathbb{F}$ und zwei Tagset-Clusterings mit den Clustermengen C bzw. C_{ref}. Es gelte $\mathsf{C} \subset \mathsf{C}_{ref}$. Dann ist die Funktion completeness : $\mathfrak{C} \times \mathfrak{C} \rightarrow \mathbb{R}$ definiert als:

$$\text{completeness}(\mathsf{C}, \mathsf{C}_{ref}) = \frac{\sum_{M \in \mathsf{C}} (\sum_{r \in R} \phi(r, M))}{\sum_{M \in \mathsf{C}_{ref}} (\sum_{r \in R} \phi(r, M))} \tag{4.34}$$

Durch das Referenz-Clustering wird für jede Ressource $r \in R$ eine maximale Punktzahl vorgeben, die errechnet wird, indem man die Anzahlen ihres Vorkommens in allen Clustern $C \in \mathsf{C}_{ref}$ aufsummiert. Summiert man die Punktzahlen aller Ressourcen auf, so ergibt sich eine Gesamtpunktzahl. Nun wird auf gleiche Weise die Punktzahl für das Clustering mit der Clustermenge C bestimmt. Die Funktion completeness errechnet einen Wert zwischen 0 und 1. Eine 1 wird erreicht, wenn die Clustermengen beider Clusterings übereinstimmen, also $\mathsf{C} = \mathsf{C}_{ref}$ gilt.

Anschaulich misst Completeness ausgehend vom Referenz-Clustering, inwieweit die Aussagen, die die Benutzer durch ihre Taggings gemacht haben, im eingeschränkten Clustering erhalten geblieben sind. Für die Verwendung als Optimierungskriterium wird als Referenz-Clustering das in Schritt 1 des Verfahrens (siehe Abschnitt 4.3) bestimmte FTS-Clustering gewählt.

Korollar 4.30. Gegeben seien zwei Clusterings mit den Clustermengen C_1 und C_2. Es ist $\mathsf{C}_1 \subset \mathsf{C}_2$. Dann gilt:

$$\text{completeness}(\mathsf{C}_1) < \text{completeness}(\mathsf{C}_2) \tag{4.35}$$

Das Korollar besagt, dass wenn sich die Clustermenge vergrößert, die Completeness ansteigen muss. Dies folgt direkt aus der Funktionsdefinition.

4.5.6 Ergebnisse: Completeness versus Childcount

Bevor wir in diesem Abschnitt die Ergebnisse der Verwendung der neuen Kriterien betrachten, soll kurz wiederholt werden, welche Ziele es zu erreichen gilt. Die erzeugte Alternativenmenge soll Clusterings enthalten, die schmal und tief sind, die möglichst viele Ressourcen enthalten, und die einen möglichst großen Anteil der Heterogenität der Tagging-Daten aufweisen.

Unter Verwendung der ersten Kriterien, Overlap und Coverage, wurden schmale Clusterings mit vielen enthaltenen Ressourcen erzeugt. Tiefe Clusterings entstanden, wenn überhaupt, nur als ein Nebeneffekt individueller Eigenschaften des Datensatzes. Die Heterogenität der Tagging-Daten ging zu großen Teilen verloren. In den letzten beiden Abschnitten wurden daher zwei neue Kriterien, Childcount und Completeness, formuliert. Childcount ersetzt das Kriterium Overlap, Completeness das Kriterium Coverage.

Die Anwendung der neuen Kriterien führt zu den folgenden Ergebnissen, wobei die folgenden Aussagen wiederum als Vorgriff auf die, in Kapitel 5.3 folgenden, empirischen Ergebnisse zu verstehen sind. Clusterings mit einem geringen Wert für Childcount sind schmal und enthalten wenig Ressourcen, während Clusterings mit einem hohen Wert für Completeness sehr viele Ressourcen enthalten, dafür aber von komplexerer Struktur sind.

Das Completeness-Kriterium führt im Optimierungsprozess dazu, dass die Alternativenmenge durchgängig aus tiefen Clusterings besteht. Dieser Effekt ist dadurch zu erklären, dass tiefe Clusterings im Gegensatz zu flachen Clusterings eine höhere Completeness aufweisen. Zusammen mit dem Childcount als konkurrierendes Kriterium ergibt sich so eine Menge von Clusterings mit vielen Abstufungen: von *schmal und tief* zu *breit und tief*.

Clusterings mit hoher Completeness sind dem zugrunde liegenden FTS-Clustering sehr ähnlich und enthalten deshalb einen großen Anteil der Heterogenität der Originaldaten. Clusterings, die das Childcount-Kriterium gut erfüllen und daher eine geringere Completeness haben, enthalten einen kleineren Anteil der Heterogenität. Im Vergleich zu den Clusterings, die unter Verwendung des Overlaps als Kriterium erzeugt wurden, bleibt die Heterogenität insgesamt wesentlich besser erhalten.

Die beiden Kriterien Childcount und Completeness erzeugen Alternativenmengen, die allen definierten Anforderungen entsprechen. Die Definition der Basisverfahrens ist somit vollständig. Eine detaillierte Untersuchung der Anwendung sowohl der alten, als auch der neuen Kriterien auf den Datenbestand eines realen Tagging-Systems, folgt in Kapitel 5.3.

4.5.7 Weitere Bewertungsfunktionen

Im Folgenden werden einige zusätzliche Bewertungsfunktionen definiert, die in Kapitel 5 zur Untersuchung von Clusterings verwendet werden.

Definition 4.31. (Tiefe eines Clusterings)
Gegeben sei ein Tagset-Clustering mit einer Clustermenge C zu einer Folksonomy $\mathbb{F} = (U, R, T, Y)$.

Sei die Menge $\mathsf{C}' \subseteq \mathsf{C}$ definiert als

$$\mathsf{C}' = \{C \in \mathsf{C} \mid \nexists D : C \prec D\} \tag{4.36}$$

Dann sind die folgenden Funktionen definiert:

$$\text{depth}_{avg}(\mathsf{C}') = \frac{1}{|\mathsf{C}'|} \sum_{C \in \mathsf{C}'} |C| \tag{4.37}$$

$$\text{depth}_{max}(\mathsf{C}') = \max_{C \in \mathsf{C}'} |C| \tag{4.38}$$

$$\text{depth}_{var}(\mathsf{C}') = \frac{1}{|\mathsf{C}'|} \sum_{C \in \mathsf{C}'} (|C| - \text{depth}_{avg}(\mathsf{C}')) \tag{4.39}$$

Zur Bestimmung der Tiefe eines Tagset-Clusterings werden alle Blätter betrachtet. Die Tiefe eines Blattes ist aufgrund der Konstruktionsvorschriften für Tagset-Clusterings gleich der Kardinalität des Tagsets. Die Funktion depth_{avg} bestimmt das

arithmetische Mittel der Tiefen aller Blätter, depth_{max} bestimmt das Blatt mit der maximalen Tiefe und depth_{var} gibt die Varianz der Tiefen aller Blätter an.

Korollar 4.32. Gegeben seien zwei Clusterings mit den Clustermengen C_1 und C_2. Es ist $\mathsf{C}_1 \subset \mathsf{C}_2$. Dann gilt:

$$\text{depth}_{max}(\mathsf{C}_1) \leq \text{depth}_{max}(\mathsf{C}_2) \tag{4.40}$$

Man kann sich leicht überlegen, dass eine solche Aussage bezüglich depth_{avg} und depth_{var} nicht möglich ist.

Definition 4.33. (Baumartigkeit des Clusterings)
Gegeben sei ein Tagset-Clustering mit einer Clustermenge $\mathsf{C} \in \mathfrak{C}$ zu einer Folksonomy $\mathbb{F}$.

Dann ist die Funktion $\text{treeness} : \mathfrak{C} \to \mathbb{R}$ definiert als

$$\text{treeness}(\mathsf{C}) = \frac{|\mathsf{C}|}{|\{(C, D) \in \mathsf{C} \times \mathsf{C} \mid C \prec D\}|} \tag{4.41}$$

Diese Funktion misst die „Baumartigkeit“ eines Tagset-Clusterings, indem sie die Anzahl der Cluster auf die Anzahl der Verbindungen zwischen den Clustern bezieht. Sie liefert ihren maximalen Wert für ein Clustering, das die Form eines minimalen Spannbaums hat. Für Clusterings, deren Clustermengen so ausgewählt sind, dass durch $\prec$ mehr Cluster in Beziehung stehen, sinkt der von treeness gelieferte Wert entsprechend in Richtung 0.

Betrachtet man als Beispiel zwei Clusterings, das erste mit einer Clustermenge $\{\{\}, \{a\}, \{a, b\}\}$ und das zweite mit einer Clustermenge $\{\{\}, \{a\}, \{b\}, \{a, b\}\}$, dann ergeben sich die Bewertungen $\frac{3}{2}$ und $\frac{4}{4}$. Das erste Clustering wird besser bewertet, da es die Form eines minimalen Spannbaums aufweist.

Satz 4.34. Gegeben seien zwei Clusterings mit den Clustermengen C_1 und C_2. Es ist $\mathsf{C}_1 \subset \mathsf{C}_2$. Dann gilt:

$$\text{treeness}(\mathsf{C}_1) \geq \text{treeness}(\mathsf{C}_2) \tag{4.42}$$

Beweis. Sei $\mathsf{C}_2' = \mathsf{C}_2 \setminus \mathsf{C}_1$. Für jedes Element $C \in \mathsf{C}_2'$ gilt:

- Wenn C ein Blatt ist, dann ergibt sich ein neuer Knoten und eine neue Kante. Der berechnete treeness-Wert bleibt konstant.

- Wenn C ein innerer Knoten ist, dann entstehen ein neuer Knoten und mindestens zwei Kanten. Dies führt zu einer Verringerung des treeness-Wertes.

Der treeness-Wert bleibt also nur dann gleich, wenn alle neuen Cluster in C_2' Blätter sind. Ansonsten sinkt er ab.

Definition 4.35. (Gesamtanzahl der Cluster)
Gegeben sei ein Tagset-Clustering mit einer Clustermenge $\mathsf{C} \in \mathfrak{C}$ zu einer Folksonomy $\mathbb{F}$. Dann ist die Funktion clustercount : $\mathfrak{C} \to \mathbb{R}$ definiert als

$$\text{clustercount}(\mathsf{C}) = |\mathsf{C}| \tag{4.43}$$

Die Funktion clustercount gibt die Kardinalität der Clustermenge eines Clusterings an. Der gelieferte Wert ist nach oben beschränkt durch die Kardinalität der Clustermenge TS des vollständigen Tagset-Clusterings. Der minimale Wert ist 1, da das leere Tagset $\emptyset$ in jeder Clustermenge vorhanden sein muss.

Definition 4.36. (Verhältnis der Blattanzahl zur Gesamtanzahl der Cluster)
Gegeben sei ein Tagset-Clustering mit einer Clustermenge $\mathsf{C} \in \mathfrak{C}$ zu einer Folksonomy $\mathbb{F}$. Dann ist die Funktion leafratio : $\mathfrak{C} \to [0, 1]$ definiert als

$$\text{leafratio}(\mathsf{C}) = \frac{1}{|\mathsf{C}|} \cdot |\{C \in \mathsf{C} \mid \nexists D : C \prec D\}| \tag{4.44}$$

Die Funktion leafratio bestimmt das Verhältnis der Anzahl der Blätter zur Gesamtanzahl der Cluster. Der errechnete Wert liegt im reellwertigen Intervall $[0, 1]$. leafratio liefert einen Wert nahe 1, wenn das Clustering größtenteils aus Blättern besteht, also sehr breit und flach ist. Für ein Clustering mit einem geringeren Anteil von Blättern liefert leafratio einen entsprechend geringeren Wert. Ein solches Clustering hat mehr innere Knoten und ist daher schmaler und tiefer.

Definition 4.37. (Durchschnittliche Anzahl der Einsortierungspfade pro Ressource)
Gegeben sei ein Tagset-Clustering mit Clustermenge $\mathsf{C} \in \mathfrak{C}$ zu einer Folksonomy $\mathbb{F}$.
Sei eine Funktion clusters : $R \to \mathcal{P}(\mathsf{C})$ definiert als

$$\text{clusters}(r) = \{C \in \mathsf{C} \mid \phi(r, C) > 0\} \tag{4.45}$$

und eine Funktion leafs : $R \to \mathcal{P}(\mathsf{C})$ als

$$\text{leafs}(r) = \{C \in \mathsf{C} \mid \nexists D \in \text{clusters}(r) : C \prec D\} \tag{4.46}$$

Die Funktion paths : $R \to \mathcal{P}(\mathsf{C})^*$ bildet eine Ressource auf die Menge aller Pfade ab, über die es in das Clustering einsortiert ist.

$$\text{paths}(r) = \{(C_1, ..., C_n) \mid C_1 = \emptyset \wedge C_{i-1} \prec C_i \wedge C_n \in \text{leafs}(r)\} \tag{4.47}$$

Es ergibt sich die Bewertungsfunktion directness : $\mathfrak{C} \to \mathbb{R}$ als:

$$\text{directness}(\mathsf{C}) = \frac{1}{|R|} \sum_{r \in R} |\text{paths}(r)| \tag{4.48}$$

Die Funktion directness bestimmt die „Geradlinigkeit“ eines Clusterings. Ein Clustering ist sehr geradlinig, wenn jede Ressource, ähnlich wie ein Buch in einem Bibliothekskatalog, auf genau einem Pfad $\emptyset \prec C_1 \prec C_2 ...$ einsortiert ist. In diesem Extremfall liefert die Funktion einen Wert nahe 1. Liegt ein Clustering vor, in dem Ressourcen auf vielen verschiedenen Pfaden einsortiert sind, was aufgrund der gegensätzlichen Sichtweisen der Benutzer eines Taggingsystems häufig der Fall sein kann, steigt der Wert entsprechend an.

Definition 4.38. (Anzahl der Ressourcen pro Cluster)
Gegeben sei ein Tagset-Clustering mit einer Clustermenge $\mathsf{C} \in \mathfrak{C}$ zu einer Folksonomy $\mathbb{F}$.

Die Funktion $\text{rescount}_{avg} : \mathcal{P}(\mathsf{C}) \rightarrow \mathbb{R}$ berechnet für eine Menge von Clustern die durchschnittliche Anzahl der Ressourcen pro Cluster:

$$\text{rescount}_{avg}(\mathsf{C}') = \frac{1}{|\mathsf{C}'|} \sum_{C \in \mathsf{C}'} |\text{res}(\{C\})| \tag{4.49}$$

Die Funktion $\text{rescount}_{var} : \mathcal{P}(\mathsf{C}) \rightarrow \mathbb{R}$ berechnet für eine Menge von Clustern die Varianz der Anzahl der Ressourcen pro Cluster:

$$\text{rescount}_{var}(\mathsf{C}') = \frac{1}{|\mathsf{C}'|} \sum_{C \in \mathsf{C}'} (|\text{res}(\{C\})| - \text{rescount}_{avg}(\mathsf{C}'))^2 \tag{4.50}$$

Dann ist die Bewertungsfunktion $\text{uniformity} : \mathfrak{C} \rightarrow \mathbb{R}$ definiert als:

$$\text{uniformity}(\mathsf{C}) = \frac{1}{\text{depth}(\mathsf{C})} \sum_{i=1}^{\text{depth}(\mathsf{C})} \text{rescount}_{var}(\mathsf{C}_{|i}) \tag{4.51}$$

Die Funktion uniformity bestimmt den Mittelwert der Varianz der Ressourcenanzahl der Cluster jeder Ebenen eines Clusterings. Ein Clustering, dessen Cluster, ebenenweise betrachtet, ähnliche Ressourcenanzahlen aufweisen, erhält eine geringe Bewertung. Weichen die Anzahlen stärker voneinander ab, so steigt der errechnete Wert.

4.6 Inkrementelle Erweiterung

Das bisher vorgestellte Verfahren nimmt eine Folksonomy als eine Datenquelle an, die sich nicht ändert. Tatsächlich wird die Folksonomy eines Tagging-Systems von den Benutzern aber ständig modifiziert und erweitert.

Die Definition einer Folksonomy (siehe Definition 2.1) wird im Folgenden so erweitert, dass sie den Zeitpunkt der Momentaufnahme des Datenbestands beinhaltet.

Definition 4.39. (Folksonomy zu einem Zeitpunkt)
Sei $t \in \mathbb{R}$ ein Zeitpunkt. Wir definieren eine Folksonomy zu einem Zeitpunkt t als:

$$\mathbb{F}_t = (U_t, T_t, R_t, Y_t) \tag{4.52}$$

Dabei repräsentieren U_t, T_t, R_t und Y_t Momentaufnahmen der Mengen der Benutzer, Tags, Ressourcen und Taggings zu einem Zeitpunkt t.

Angenommen ein Benutzer verwendet eine Implementierung des Verfahrens und erzeugt zum Zeitpunkt t eine Menge von Clusterings auf Basis der Folksonomy $\mathbb{F}_t$. Zu einem späteren Zeitpunkt $t+s$ enthält die Folksonomy $\mathbb{F}_{t+s}$ neue Ressourcen und neue Taggings.

Der Benutzer möchte natürlich, dass seine Clusterings möglichst aktuelle Tagging-Daten widerspiegeln. Die bisher geforderten Eigenschaften für eine gute Navigationsstruktur (siehe Kapitel 2.6) werden also um eine neue Forderung ergänzt: der Abstand des aktuellen Zeitpunkts $t + s$ und des Zeitpunkts t, auf den sich die Menge der Clusterings bezieht, soll möglichst klein sein.

Der Benutzer habe nun ein Clustering ausgewählt, das seinen Anforderungen gut entspricht. Um die Veränderungen in der Folksonomy berücksichtigen zu können, muss er das Verfahren erneut anwenden. Es ergibt sich eine neue Menge alternativer Clusterings für die Folksonomy $\mathbb{F}_{t+s}$.

Der Benutzer möchte aus der neuen Front nun ein Clustering auswählen, welches seinen persönlichen Anforderungen an Vollständigkeit und Einfachheit in ähnlicher Weise entspricht, wie das zuvor gewählte Clustering. Daher wählt er ein Clustering, dessen relative Position in der Front ähnlich der Position des Clusterings in der vorherigen Front ist. Das so gewählte Clustering enthält je nach Umfang der Änderungen an der Folksonomy, die sich seit dem Zeitpunkt t ergeben haben, viele neue Tagsets. Es ist auch wahrscheinlich, dass bisher vorhandene Tagsets weggefallen sind. Für den Benutzer ist es wünschenswert die Struktur, an die er sich gewöhnt hat, trotz der neuen Daten in der Folksonomy, weiterzuverwenden. Wir definieren ein neues Kriterium, welches die strukturelle Ähnlichkeit zweier Clusterings misst:

Definition 4.40. (Similarity)
Gegeben seien zwei Clusterings mit den Clustermengen C und C_{ref}. Es gelte $\mathsf{C} \subseteq \mathsf{C}_{ref}$ und eine Menge $\mathsf{C}' \subseteq \mathsf{C}$ sei definiert als $\mathsf{C}' = \mathsf{C}_{ref} \setminus \mathsf{C}$. C' enthält alle Cluster, die in der Clustermenge C in Vergleich zu C_{ref} weggefallen sind.

Dann ist eine Funktion $\mathrm{sim} : \mathfrak{C} \times \mathfrak{C} \to \mathbb{R}$ definiert als:

$$\mathrm{sim}(\mathsf{C}, \mathsf{C}_{ref}) = \sum_{C \in \mathsf{C}'} |\{D \in \mathsf{C}' \mid C \prec^* D\}| \tag{4.53}$$

wobei $\prec^*$ der transitive Abschluss über $\prec$ ist.

Die Funktion sim bewertet die strukturelle Ähnlichkeit zweier Clusterings. Den Ausgangspunkt bildet das Referenz-Clustering. Das Clustering mit der Clustermenge C entsteht aus dem Referenz-Clustering, indem Cluster gelöscht und neue Cluster hinzugefügt werden. Neu hinzugefügte Cluster haben keinen Einfluss auf den zurückgelieferten Wert der Funktion. Jedes gelöschte Cluster jedoch erhöht ihn. Dabei verursachen gelöschte Blätter eine Erhöhung von 1. Cluster mit Kind-Clustern erhöhen den Wert proportional zur Anzahl der durch $\prec$ transitiv von dort aus erreichbaren Cluster. Die Funktion sim ist nicht-linear, da für einen weggefallenen Teilbaum von Clustern sowohl die Wurzel als auch alle Kinder jeweils einen Strafwert proportional zur Anzahl ihrer Nachfolger verursachen.

Wenn $\mathsf{C} = \mathsf{C}_{ref}$ gilt, so sind die Clusterings maximal ähnlich und sim liefert den Wert 0. Je mehr Cluster aus C_{ref} gelöscht wurden, umso höher ist der gelieferte Wert.

Unter Vorgabe eines Referenz-Clustering kann das Kriterium sim, welches im Folgenden auch als Similarity-Kriterium bezeichnet wird, dazu verwendet werden, die Suche im Raum der Tagset-Clusterings in Richtung strukturell ähnlicher Clusterings zu lenken.

Die Grundversion des Verfahrens verwendet zwei Optimierungskriterien, da dies eine für den Benutzer verständliche Auswahl eines Clusterings aus der Ergebnismenge ermöglicht. Um die Ähnlichkeit zu einem zuvor ausgewählten Clustering zu berücksichtigen, muss das Verfahren um ein drittes Optimierungskriterium erweitert werden. Es werden nun also gleichzeitig die Kriterien childcount_{max}, completeness und sim verwendet.

Die Auswahl eines Clusterings aus einer dreidimensionalen Pareto-Front ist recht unintuitiv. Daher wird der Auswahlprozess zweistufig gestaltet. Im ersten Schritt wählt der Benutzer aus einer Projektion der Pareto-Menge auf die Dimensionen *Childcount* und *Completeness* den ungefähren Wert für *Childcount* aus, den das in Zeitpunkt t gewählte Clustering aufwies. Dadurch definiert er eine Projektionsebene, anhand der die Paretomenge auf die Dimensionen *Completeness* und *Similarity* projiziert wird. Unter der Voraussetzung, dass die drei Kriterien in Konkurrenz zueinander stehen, sollte dieser Schnitt eine gewöhnliche zweidimensionale Pareto-Front enthalten.

In dieser Front kann der Benutzer dann intuitiv diejenigen Clusterings betrachten, deren *Einfachheit* (d.h. *Childcount*) das in Zeitpunkt t bereits ausgewählte Maß hat. Durch die explizite Darstellung des Similarity-Kriteriums kann er erforschen, inwieweit ein Clustering der erweiterten Folksonomy möglich ist, welches große strukturelle Ähnlichkeit zu seinem gewohnten Clustering hat.

Es gibt alternative Möglichkeiten zur Definition der inkrementellen Erweiterung: Zum einen könnte im ersten Schritt statt eines Wertes für *Childcount* ein Wert für *Completeness* ausgewählt werden. Der Grad der Einfachheit der Struktur hat aber eine größere Wichtigkeit für den Benutzer, wenn das letztliche Ziel quasi eine inkrementelle Anpassung einer bestehenden Struktur auf neue Daten ist. Die Ähnlichkeit zum FTS-Clustering spielt eine geringere Rolle. Außerdem kann sich das FTS-Clustering durch

die zusätzlichen Daten auch radikal verändert haben, was das Übertragen des Wertes vom Zeitpunkt t auf den Zeitpunkt $t + s$ erschwert.

Zum anderen wäre es möglich, die ersten beiden Kriterien als gewichtete Summe zu einem einzigen Kriterium zu verschmelzen und das Similarity-Kriterium als zweites Kriterium zu verwenden. Die Wahl der richtigen Gewichtung ist allerdings sehr schwierig und eine schlechte Wahl kann die Suchrichtung stark verändern. Vor allem würde aber auf diese Weise gegen eines der in Kapitel 2.6 definierten Ziele des Verfahrens verstoßen. Die verwendeten Kriterien wären nicht mehr explizit ersichtlich und der Benutzer könnte das Ergebnis nur schwer interpretieren.

In Kapitel 5.4 wird der Ansatz der Erweiterung zu einem dreidimensionalen Optimierungsverfahren anhand des Datenbestands eines Tagging-Systems untersucht.

Kapitel 5

Experimente

Dieses Kapitel beschreibt die Anwendung des Verfahrens auf den Datenbestand eines realen Tagging-Systems. Das Verfahren wurde als Plugin für Yale *[29], einer Plattform für Methoden des maschinellen Lernens, implementiert. Im folgenden Abschnitt werden die Eigenschaften des Datensatzes beschrieben. Es folgt eine Untersuchung der Beziehungen der Bewertungsfunktionen für Tagset-Clustering (Ziel 4). Dann schließt sich die Evaluation der Fitnessfunktionen (Ziel 7) an, bevor abschließend die inkrementelle Erweiterung des Verfahrens erprobt wird.*

5.1 Der Bibsonomy-Datensatz

Bibsonomy [21] ist ein an der Universität Kassel entwickeltes Tagging-System für Web-Ressourcen und Publikationen. Der verwendete Datensatz ist eine Ende April 2007 angefertigte Kopie des Datenbestands dieses Systems.

Auf Basis dieses Datensatzes ist eine Folksonomy $\mathbb{F} = (U, T, R, Y)$ definiert. Es gilt:

$$|U| \approx 780$$

$$|R| \approx 59000$$

$$|T| \approx 25000$$

$$|Y| \approx 330000$$

Die Menge R der Ressourcen enthält ausschließlich Web-Ressourcen. Die Publikationen und zugehörige Taggings werden nicht betrachtet. Es gilt weiterhin:

$$|\{(u, r) \mid \exists (u, t, r) \in Y\}| \approx 66000$$

5.1.1 Tagnutzung

Zunächst soll das Verhalten der Benutzer beim Tagging untersucht werden. Hierbei ist es interessant festzustellen, inwieweit Benutzer vorhandene Tags weiterverwenden oder neue Tags zum System hinzufügen. Für jedes Tag des Datensatzes werden drei

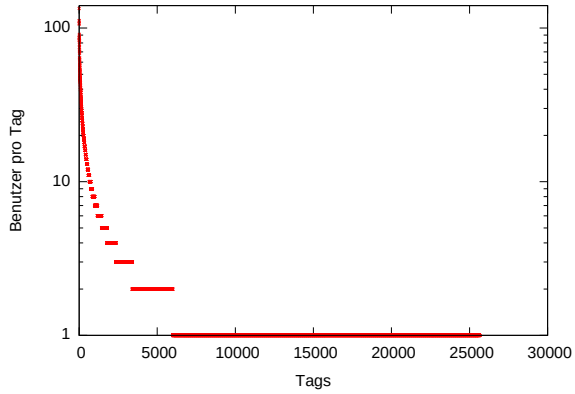


Abbildung 5.1: Anzahl der Benutzer pro Tag im Bibsonomy-Datensatz

Betrachtungen angestellt: die Anzahl der Benutzer aus U, die das Tag verwendet haben, die Anzahl der Ressourcen aus R, für das ein Tag verwendet wurde, und die Anzahl der Taggings $(u, t, r) \in Y$, die das Tag enthalten. Die Ergebnisse sind in den Abbildungen 5.1, 5.2 und 5.3 dargestellt.

Wie in Abbildung 5.1 zu sehen, wird ein Großteil der Tags (ca. 67%) von jeweils nur einem Benutzer verwendet. Abbildung 5.2 zeigt, dass etwa die Hälfte der Tags nur einer Ressource zugewiesen wurden. Diese Abbildung ist außerdem nahezu identisch zu Abbildung 5.3; demnach wird die Hälfte der Tags nur in jeweils einem Tagging verwendet. Diese Ähnlichkeit der Abbildungen ist damit zu erklären, dass die durchschnittliche Anzahl gemeinsamer Ressourcen der Benutzer relativ gering ist (siehe Abbildung 5.4). Die Eigenschaft von Tagging-Systemen, dass relativ wenige Tags sehr häufig und sehr viele Tags selten verwendet werden, wird auch als *Long Tail*-Effekt bezeichnet. Entsprechende Betrachtungen über die Nutzung von Tags in Tagging-Systemen wurden von Marlow et al. [27] für FLICKR und von Golder et al. [14] für DEL.ICIO.US in ausführlicherer Form angestellt. Auch Hammond et al. [16, 17], Mathes [28] und Shirky [37] beschäftigen sich mit diesem Thema.

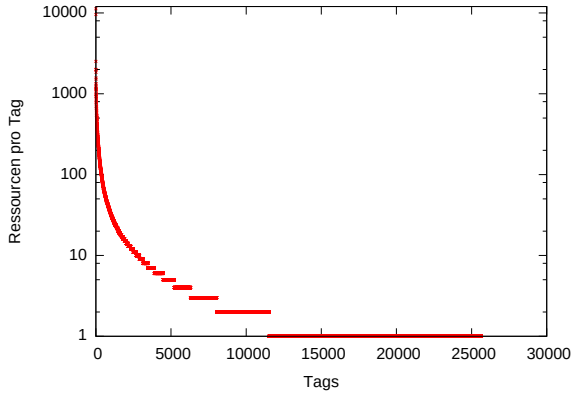


Abbildung 5.2: Anzahl der Ressourcen pro Tag im Bibsonomy-Datensatz

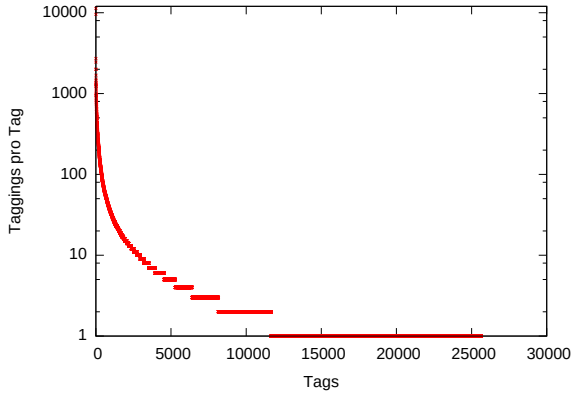


Abbildung 5.3: Anzahl der Taggings pro Tag im Bibsonomy-Datensatz

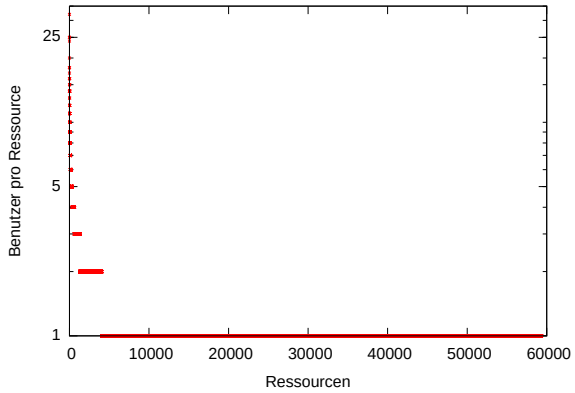
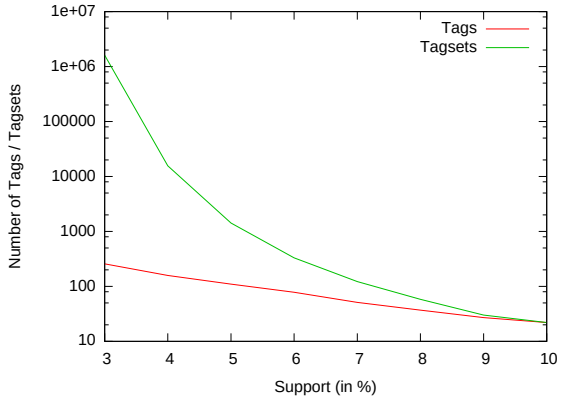


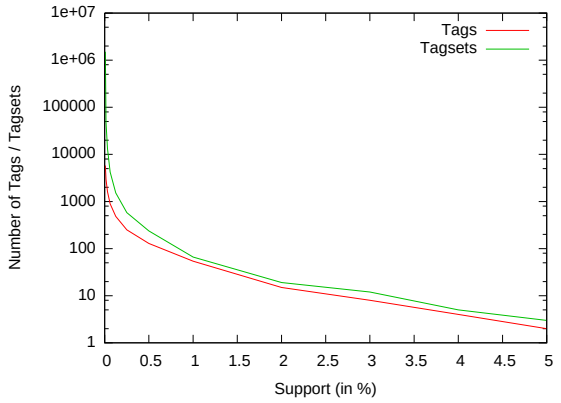
Abbildung 5.4: Anzahl der Benutzer pro Ressource im Bibsonomy-Datensatz

5.1.2 Häufigkeiten der Tagsets

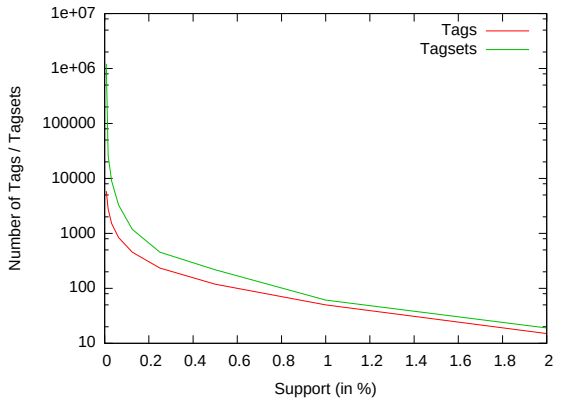
In Abschnitt 4.9 wurden drei verschiedene Möglichkeiten zur Bestimmung der Häufigkeit von Tagsets eingeführt. Für den Datensatz wurde zu jeder dieser Möglichkeiten unter Verwendung verschiedener Supportwerte die Anzahl der Frequent Tags und der Frequent Tagsets bestimmt. Die Ergebnisse sind in den Abbildungen (5.5a - 5.5c) dargestellt.



(a)



(b)



(c)

Abbildung 5.5: Anzahlen der Tags und Tagsets im Bibsonomy-Datensatz bei Verwendung verschiedener Supportwerte und der drei Häufigkeitsbegriffe: (a) Häufigkeit bestimmt auf U, (b) Häufigkeit bestimmt auf R, (c) Häufigkeit bestimmt auf $U \times R$

5.2 Beziehungen der Bewertungsfunktionen

Im letzten Kapitel wurden verschiedene Bewertungsfunktionen für Tagset-Clusterings definiert. Einige der Funktionen werden als Fitnessfunktionen direkt im Suchverfahren eingesetzt, andere sind formale Definitionen interessanter Eigenschaften von Clusterings.

Zwei Bewertungsfunktionen können auf verschiedene Weise in Beziehung zueinander stehen. Es ist möglich, dass (i) der Wert von Funktion g sinkt, wenn der Wert von Funktion f steigt, dass (ii) der Wert von Funktion g steigt, wenn der Wert von Funktion f steigt, oder dass (iii) die Funktionswerte in keiner Beziehung zueinander stehen.

Einige Eigenschaften der Bewertungsfunktionen wurden bereits in Kapitel 4 als Korollare der Definitionen eingeführt. Zusätzlich zu diesen analytischen Betrachtungen schließt sich nun eine empirische Untersuchung an.

Zur empirischen Untersuchung wurden für den Bibsonomy-Datensatz zufällige Clusterings erzeugt, diese mit den Bewertungsfunktionen ausgewertet und anschließend die Korellation der Meßwerte untersucht. Stellt man eine negative Korellation zwischen zwei Funktionen fest, so ist es wahrscheinlich, dass diese Funktionen als Fitnessfunktionen in einem multikriteriellen Optimierungsprozess eingesetzt werden können. Eine positive Korellation zwischen einer verwendeten Fitnessfunktion und einer anderen Bewertungsfunktion bedeutet, dass die durch die Bewertungsfunktion formalisierte Eigenschaft gleichzeitig mit optimiert wird.

Auf Basis des Bibsonomy-Datensatzes wurde unter Verwendung der Funktion supp_U eine Menge von Frequent Tagsets bestimmt. Der Support wurde so gewählt, dass ein Tagset von 35 der 780 Benutzer verwendet worden sein muss, damit es als häufig angesehen wird. Das resultierende FTS-Clustering hatte ca. 4000 Cluster. Es wurden zufällig 5000 FFTS-Clusterings für dieses FTS-Clustering erzeugt und mit den Bewertungsfunktionen ausgewertet. Die Ergebnisse sind in den Abbildungen 5.6, 5.7 und 5.8 dargestellt.

Eine Erhöhung des Childcount wird in der Regel durch neue Kind-Cluster des $\emptyset$-Clusters verursacht. Je mehr sich die Anzahl der Kinder des $\emptyset$-Clusters der durch das FTS-Clustering vorgegebenen oberen Schranke annähert, umso ähnlicher wird das Clustering dem FTS-Clustering. Daher steigt der Wert für completeness (siehe Abbildung 5.6c). Gemäß Gleichung 4.26 wird eine Erhöhung des Wertes für w-coverage durch neue Kind-Cluster des $\emptyset$-Clusters verursacht. Der Wert für w-coverage steigt daher ebenfalls (siehe Abbildung 5.6d).

Eine Erhöhung von childcount_{mean} führt zu einer Erhöhung der Gesamtanzahl der Cluster (siehe Abbildung 5.6j). Je mehr Cluster vorhanden sind, umso ähnlicher wird

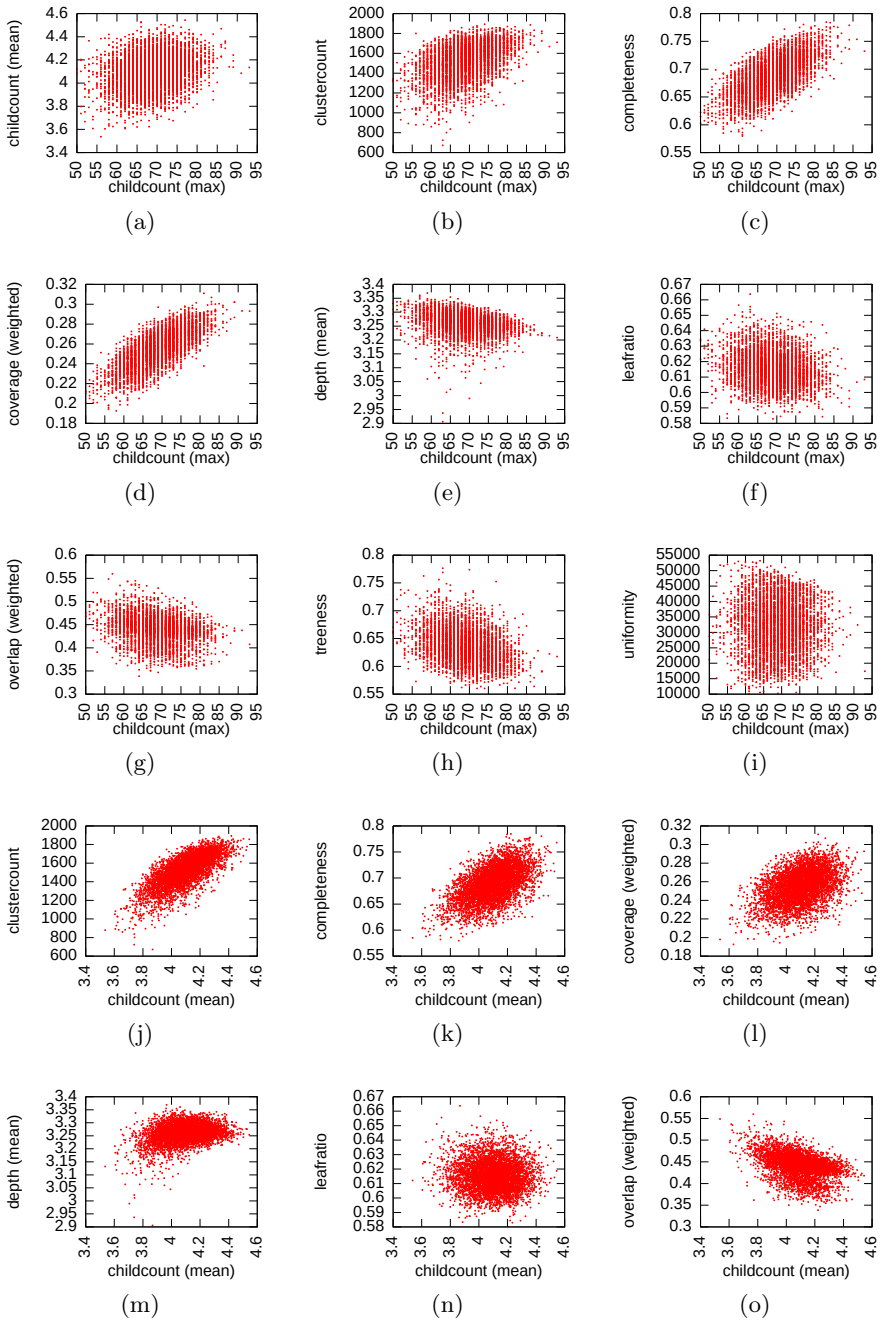


Abbildung 5.6: Beziehungen der Bewertungsfunktionen für zufällige FFTS-Clusterings des Bibsonomy-Datensatzes (1/3)

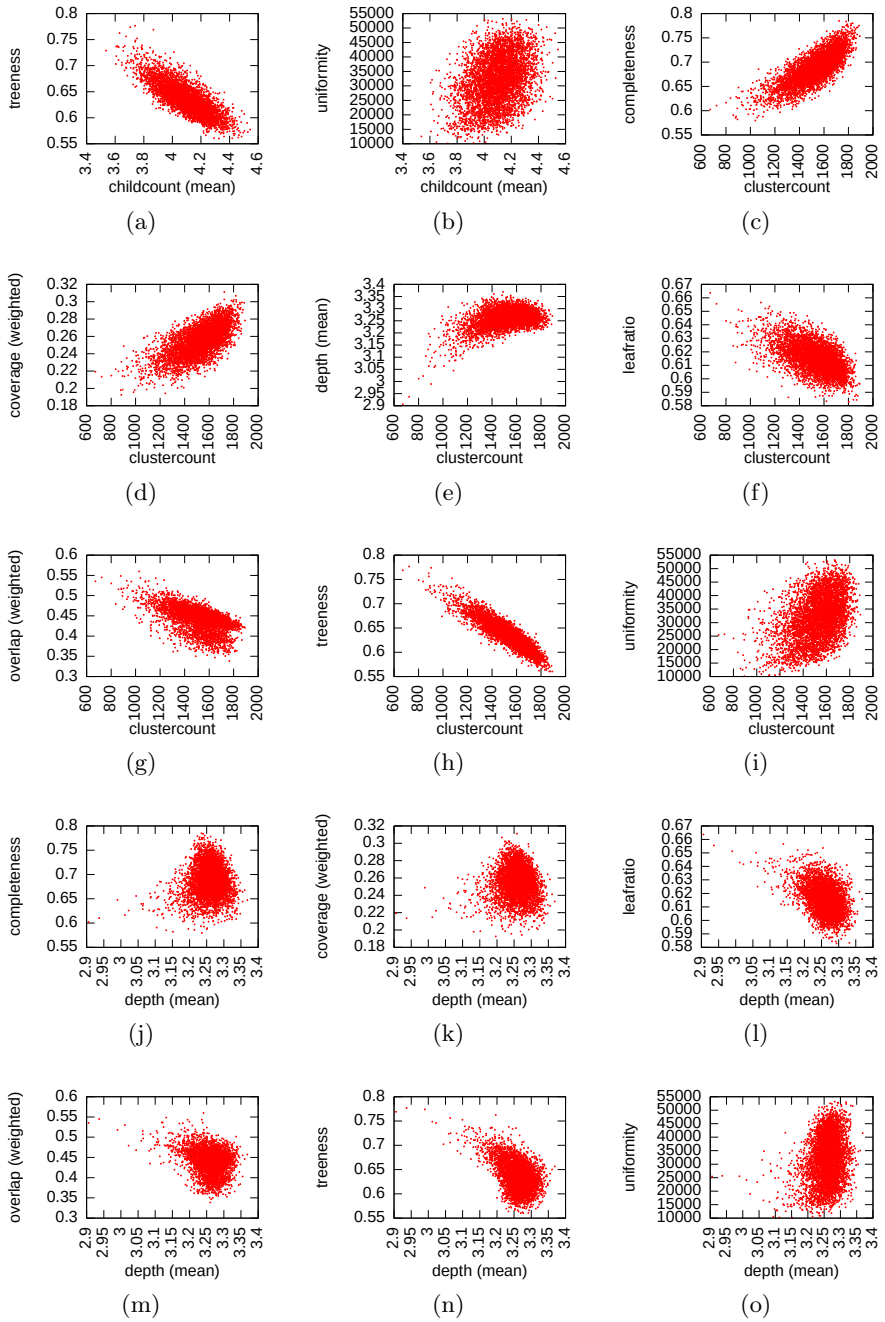


Abbildung 5.7: Beziehungen der Bewertungsfunktionen für zufällige FFTS-Clusterings des Bibsonomy-Datensatzes (2/3)

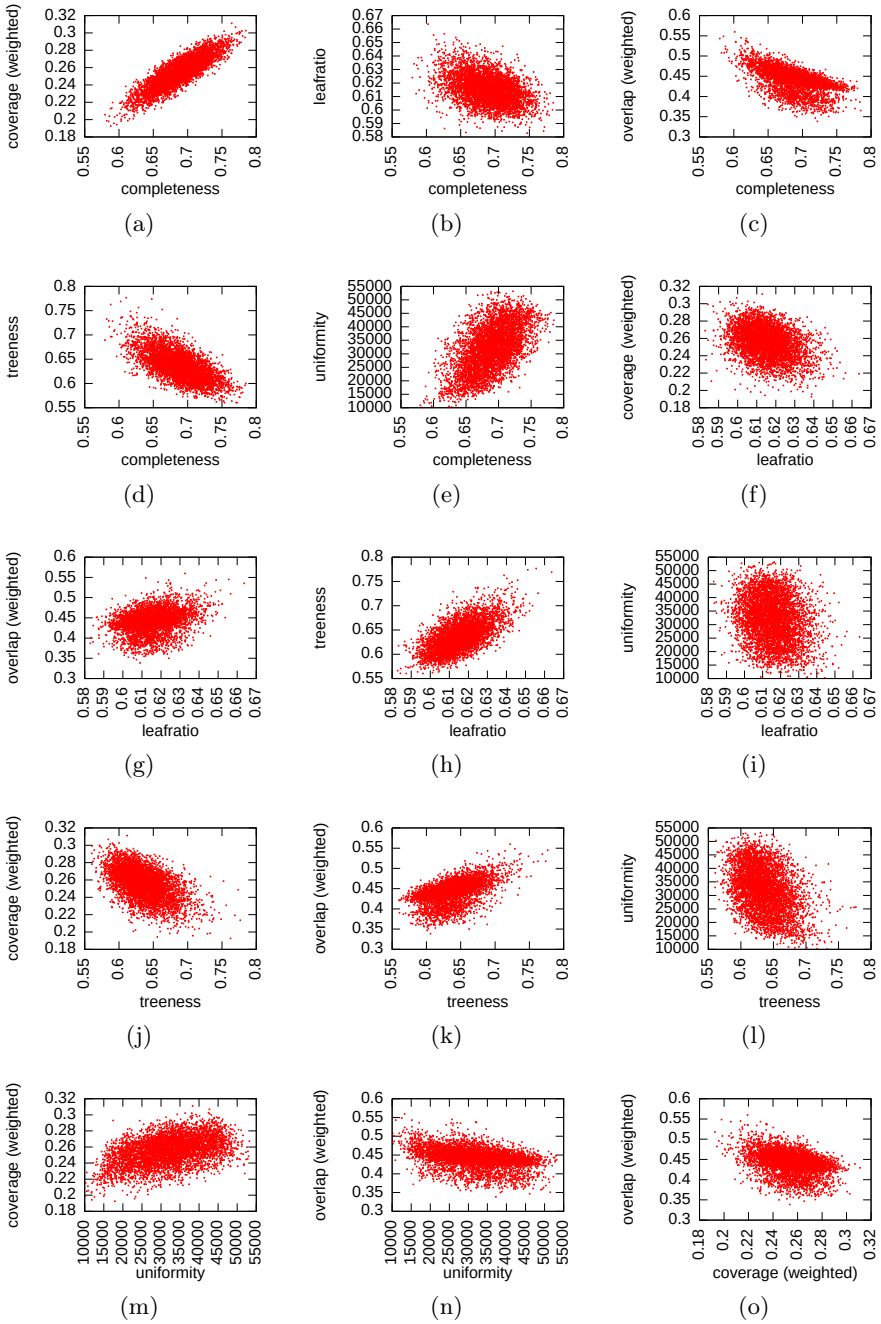


Abbildung 5.8: Beziehungen der Bewertungsfunktionen für zufällige FFTS-Clusterings des Bibsonomy-Datensatzes (3/3)

	childcount$_{max}$	childcount$_{mean}$	clustercount	depth$_{mean}$	completeness	leafratio	treeness	uniformity	w-coverage	w-overlap
childcount$_{max}$	+	0	0	0	+	0	0	0	+	0
childcount$_{mean}$		+	+	0	+	0	−	0	+	−
clustercount			+	0	+	−	−	0	+	0
depth$_{mean}$				+	0	0	0	0	0	0
completeness					+	0	−	0	+	−
leafratio						+	+	0	0	0
treeness							+	0	0	+
uniformity								+	0	0
w-coverage									+	−
w-overlap										+

Abbildung 5.9: Beziehungen der Bewertungsfunktionen für Clusterings. Ein + kennzeichnet eine positive Korellation und ein – eine negative Korellation. Gibt es keine erkennbare Beziehung zwischen zwei Funktionen, so ist dies durch eine 0 gekennzeichnet.

das Clustering dem FTS-Clustering und der Wert für completeness steigt (siehe Abbildung 5.7c). Da das FTS-Clustering viele Querverbindungen[1] hat und nicht sehr baumförmig ist, weist ein Clustering mit einem höheren Wert für completeness einen verminderten Wert für treeness auf (siehe Abbildung 5.8d). Aufgrund der Erhöhung der Gesamtanzahl der Cluster haben die Clusterings ebenfalls einen höheren Wert für w-coverage (siehe Abbildung 5.7d). Eine Erhöhung der Baumförmigkeit führt zu einer geringeren Überlappung der Cluster und einem Ansteigen des Werts von w-overlap (siehe Abbildung 5.8k). Das ist dadurch zu erklären, dass die Überlappung der Cluster sinkt, wenn es weniger Querverbindungen gibt.

Clusterings mit einer höheren Anzahl an Blättern sind baumförmiger (siehe Abbildung 5.8h), da auf Ebene 1 keine Querverbindungen vorhanden sein können. Clusterings mit einem höheren Wert für w-coverage haben in der Regel einen geringeren Wert für w-overlap (siehe Abbildung 5.8o), da zusätzliche Abdeckung durch neue Cluster verursacht wird und sich jedes zusätzliche Cluster potentiell mit einigen der vorhandenen Clustern überlappt.

Gemäß der untersuchten Daten stehen die Funktionen w-coverage und w-overlap sowie completeness und childcount$_{max}$ (multipliziert mit -1) in Konkurrenz (siehe Abbildungen 5.8o und 5.6c). Konkurrenz der Zielfunktionen ist eine Grundvoraussetzung für

[1] Die zweite und alle weiteren Kanten, mit der ein Cluster mit Clustern der vorherigen Ebene verbunden ist, werden als Querverbindungen bezeichnet.

das Gelingen einer multikriterielle Optimierung. Auch lassen sich Beziehungen der ersten und den zweiten Fitnessfunktionen erkennen. Eine Optimierung von completeness sollte w-coverage mit optimieren (siehe Abbildung 5.8a). completeness steht mit dem Vorgänger-Kriterium des childcount$_{max}$, der Funktion w-overlap, in Konkurrenz (siehe Abbildung 5.8c). Ebenso steht childcount$_{max}$ mit dem Vorgänger-Kriterium von completeness, der Funktion w-coverage, in Konkurrenz (siehe Abbildung 5.6d). Bei einer Optimierung des childcount$_{max}$ wird w-overlap in der Regel nicht mit optimiert (siehe Abbildung 5.6g).

Außerdem zeigt sich, dass depth$_{mean}$ mit keiner der anderen Bewertungsfunktionen korelliert ist. Wir werden im folgenden Abschnitt sehen, wie sich die Tiefe der Clusterings verhält, wenn man gemäß zweier konkreter Fitnessfunktionen optimiert, anstatt nur zufällige Clusterings zu betrachten.

In Abbildung 5.9 werden die Ergebnisse der Untersuchung der Beziehungen zwischen den Bewertungsfunktionen zusammengefasst.

5.3 Evaluation der Fitnessfunktionen

Zur Erprobung der Fitnessfunktionen *Coverage / Overlap* und *Childcount / Completeness* wurde der in Abschnitt 5.1 beschriebene Datensatz des Bibsonomy-Systems verwendet. Für die Experimente wurden die Supportwerte $3,9\%$, $4,5\%$ sowie $5,1\%$ verwendet, wobei der Support über die Funktion supp$_U$ (siehe Definition 4.9) definiert ist.

Der Suchraum (siehe Gleichung 4.15) besteht aus allen gültigen Teil-Clusterings eines FTS-Clusterings. Die Ergebnisse einer Auswertung der wichtigsten Bewertungsfunktionen (unter anderem der Anzahl der Cluster) auf diesem FTS-Clustering sind in Abbildung 5.10 dargestellt.

Support	childcount$_{max}$	childcount$_{mean}$	clustercount	depth$_{mean}$	completeness	leafratio	treeness	uniformity	w-coverage	w-overlap
5,1%	116	6,093	1425	2,792	1,0	0,579	0.390	42768	0,345	0,286
4,5%	138	6,239	4057	3,293	1,0	0,514	0.330	32344	0,357	0,331
3,9%	172	6,363	23525	4,258	1,0	0,387	0.256	21603	0,370	0,360

Abbildung 5.10: Ergebnisse der Auswertung der initialen FTS-Clusterings mit den wichtigsten Bewertungsfunktionen

NSGA-II wurde mit den genetischen Operatoren (siehe auch Kapitel 3.4) verwendet, die in [9] vorgeschlagen wurden.

Uniform mutation: Jedes Bit eines Individuum wird mit einer Wahrscheinlichkeit p_{mut} invertiert.

Uniform crossover: Mit einer Wahrscheinlichkeit p_{cross} findet *Crossover* zwischen zwei Individuen statt. Jedes Bit wird dann mit 50% Wahrscheinlichkeit zwischen den Individuen ausgetauscht.

Tournament selection: Das Gewinner-Individuum eines Tournaments wird immer in die neue Population übernommen.

Die initiale Population wird zufällig erzeugt. In jedem Individuum wird jede Stelle mit einer Wahrscheinlichkeit p_{init} mit einer 1 initialisiert. Die Population enthält n_{pop} Individuen. Die Länge eines Individuums wird mit n_{ind} bezeichnet. Der Algorithmus berechnet n_{gen} Generationen. Es wurde folgende Parametrisierung verwendet:

$$n_{gen} = 300 \qquad p_{init} = 0,5$$
$$n_{pop} = 256 \qquad p_{cross} = 0,5$$
$$p_{mut} = \frac{1}{n_{ind}}$$

Nach diesem Aufbau werden die Experimente nun zunächst für die Fitnessfunktionen *Coverage / Overlap* und danach für *Childcount / Completeness* durchgeführt.

5.3.1 Overlap versus Coverage

Abbildung 5.11 zeigt die Pareto-Fronten für die verschiedenen Supportwerte, wenn w-overlap und w-coverage als Fitnessfunktionen verwendet werden.

Je geringer der gewählte Supportwert, umso mehr Cluster existieren im zugrunde liegenden FTS-Clustering insgesamt, und umso schwieriger wird es, ein Clustering mit einem guten Wert für den Overlap zu erzeugen. Je mehr Cluster es gibt, umso größer ist die Wahrscheinlichkeit für Überlappungen zwischen den Clustern. Während für einen Support von $5,1\%$ ein Wert von über $0,85$ erreicht wird, hat das beste Individuum bei $3,9\%$ nur noch einen Wert von etwa $0,75$. Eine untere Schranke für den Wert ist durch das FTS-Clustering gegeben (siehe Abbildung 5.10). Individuen mit einem geringen Wert für den Overlap sind weit von dieser Schranke entfernt.

Je kleiner der Supportwert, umso größer ist der maximal erreichbare Wert für Coverage. Durch das FTS-Clustering ist eine obere Schranke vorgegeben. Diese Schranke wird von einigen Individuen jeder Front erreicht.

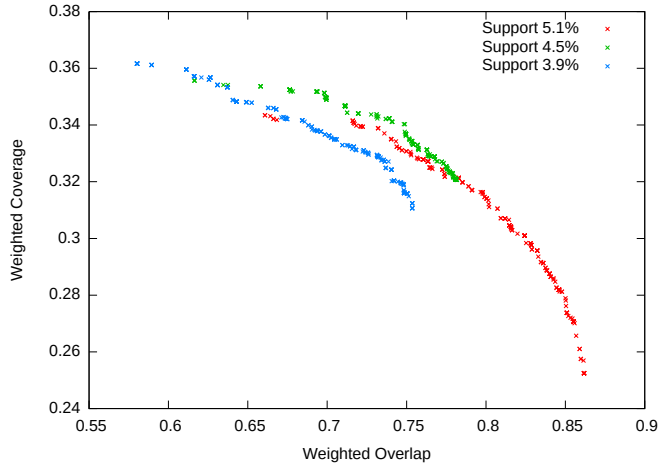


Abbildung 5.11: Pareto-Fronten für Weighted Overlap vs. Weighted Coverage bei verschiedenen Werten für den Support.

Bei einem kleinen Supportwert gibt es insgesamt mehr überlappende Cluster und auch auf Ebene 1 mehr Cluster. Eine Erhöhung des Wertes für Coverage entsteht durch zusätzliche Cluster auf Ebene 1 (siehe Korollar 4.26). Umgekehrt fällt bei Entfernen eines Clusters der Wert für Coverage stärker ab, wenn dieses Cluster eine geringe Überlappung mit anderen Clustern hat. Bei einem geringen Supportwert sind solche Cluster sehr selten und daher ist es schwierig, einen geringen Wert für Coverage zu erreichen. Aufgrunddessen liegen die Individuen für einen Support von $3,9\%$ alle relativ nahe an der oberen Schranke, wohingegen bei 5.1% ein viel größerer Wertebereich abgedeckt wird.

Abbildung 5.12 zeigt die Anzahl der Cluster aller Clusterings für die verschiedenen Supportwerte. Je mehr Cluster im jeweils zugrunde liegenden FTS-Clustering vorhanden sind, umso mehr Cluster werden durchschnittlich in den Individuen selektiert.

Je besser ein Clustering das Overlap-Kriterium erfüllt, umso mehr ist zu erwarten, dass es sich strukturell in Richtung eines minimalen Spannbaums entwickelt. Abbildung 5.13 zeigt für jedes Clustering seinen Wert für Overlap gegenüber dem Wert für Treeness. Tatsächlich erhalten Clusterings mit einem besseren Wert für Overlap einen besseren Wert für Treeness. Je höher der Supportwert, umso höher ist der maximal erreichte Wert für Treeness. Das FTS-Clustering weist für hohe Supportwerte wesentlich weniger Querverbindungen zwischen den Clustern auf als für niedrige Supportwerte. Da die Relation der Cluster untereinander für jedes Clustering konstant ist, gelingt es für niedrige Supportwerte weniger gut, die Querverbindungen zwischen den Clustern

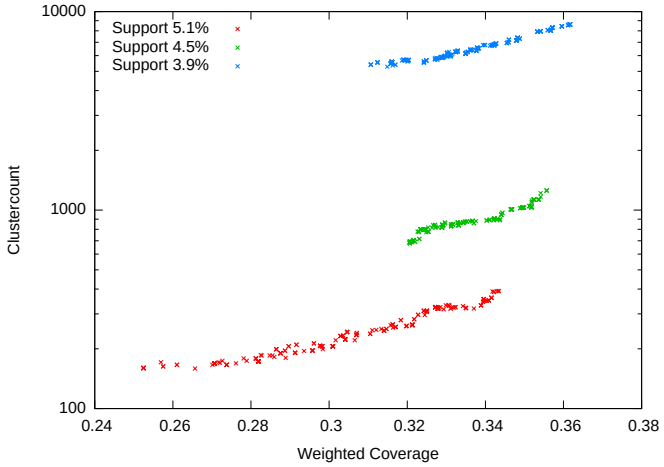


Abbildung 5.12: Anzahl der Cluster in den Clusterings für die verschiedenen Supportwerte bei Optimierung nach Overlap und Coverage

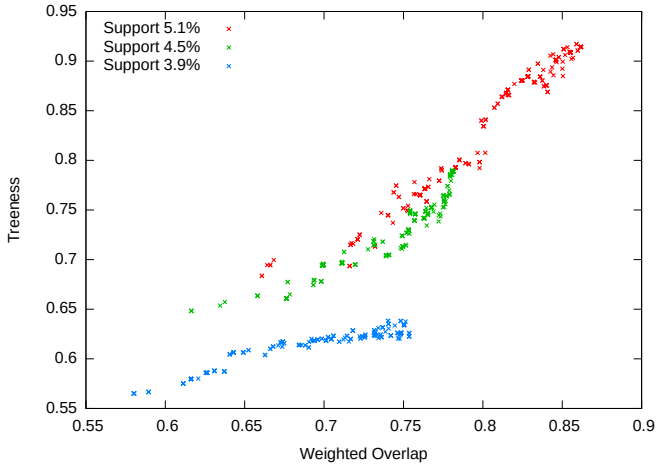


Abbildung 5.13: Overlap der Clusterings gegenüber ihrer Baumartigkeit (*treeness*) bei Optimierung nach Overlap und Coverage

zu vermeiden.

Abbildung 5.14 zeigt den Wert der Clusterings für Coverage gegenüber der durchschnittlichen Tiefe ihrer Blätter (siehe Definition 4.31). Je geringer der Supportwert, umso mehr Tags enthält das größte Tagset. Für das FTS-Clustering liegt dieser Wert für einen Support von 3.9% bei 7, für 4.5% bei 5 und für 5.1% bei 4. Diese Werte sind die oberen Schranken für die erreichbare durchschnittliche Tiefe. Für alle Fron-

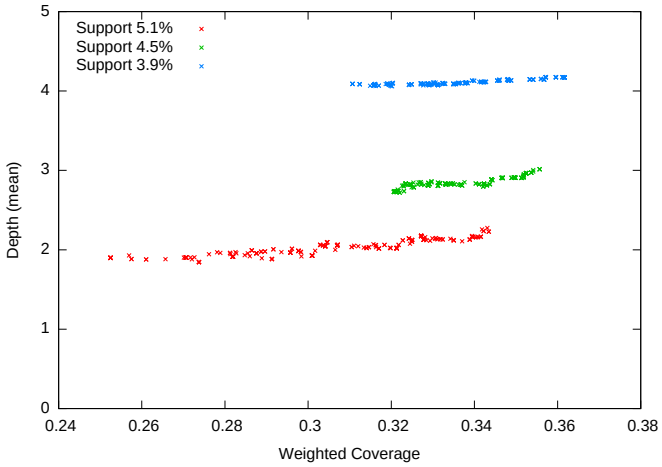


Abbildung 5.14: Coverage der Clusterings gegenüber der durchschnittlichen Tiefe ihrer Blätter bei Optimierung nach Overlap und Coverage

ten gilt, dass die erreichte durchschnittliche Tiefe für alle Clusterings nahezu gleich ist. Für einen geringeren Supportwert ergibt sich eine höhere Tiefe. Das zugrunde liegende FTS-Clustering weist dann mehr Querverbindungen zwischen den Clustern auf. Ein tiefes Cluster ist somit über viele verschiedene Wege erreichbar und es wird unwahrscheinlicher, dass es wegfällt, da dazu alle Wege durch Deselektion anderer Cluster unterbrochen werden müssten. Dies führt dazu, dass ohne explizite Optimierung durch die Fitnessfunktionen insgesamt eine höhere durchschnittliche Tiefe entsteht.

Ein Clustering, welches gute Bewertungen von Overlap und Coverage erhält, muss nicht tief sein. Da ein höherer Wert für Coverage durch Cluster auf der Ebene 1 verursacht wird, kann schon mit einem Clustering der Tiefe 1 der maximal mögliche Wert erreicht werden. Tiefe Clusterings haben einen potentiell höheren Overlap. Wenn zwei Cluster ein gemeinsames Kind-Cluster haben, so überlappen sie sich. Je größer die Überlappung, umso mehr Ressourcen enthält das Kind-Cluster. Coverage hat also keinen Einfluss und für Overlap würden flache Clusterings bevorzugt. Wie aber im vorherigen Abschnitt erläutert, wird es mit geringerem Supportwert unwahrscheinlicher, dass alle Cluster von größerer Tiefe deselektiert werden.

In Abbildung 5.15 sind verschiedene Clusterings für einen Supportwert 7% als Graphen dargestellt. Das Individuum links oben hat einen sehr guten Wert für Overlap. Wie im letzten Abschnitt beschrieben, sind größtenteils Cluster der Ebene 1 selektiert und es gibt keine gemeinsamen Kind-Cluster. Man kann erkennen, dass diese Selektionsstrategie auch für alle anderen Individuen angewandt wurde.

Coverage soll im nächsten Abschnitt durch das Kriterium Completeness ersetzt wer-

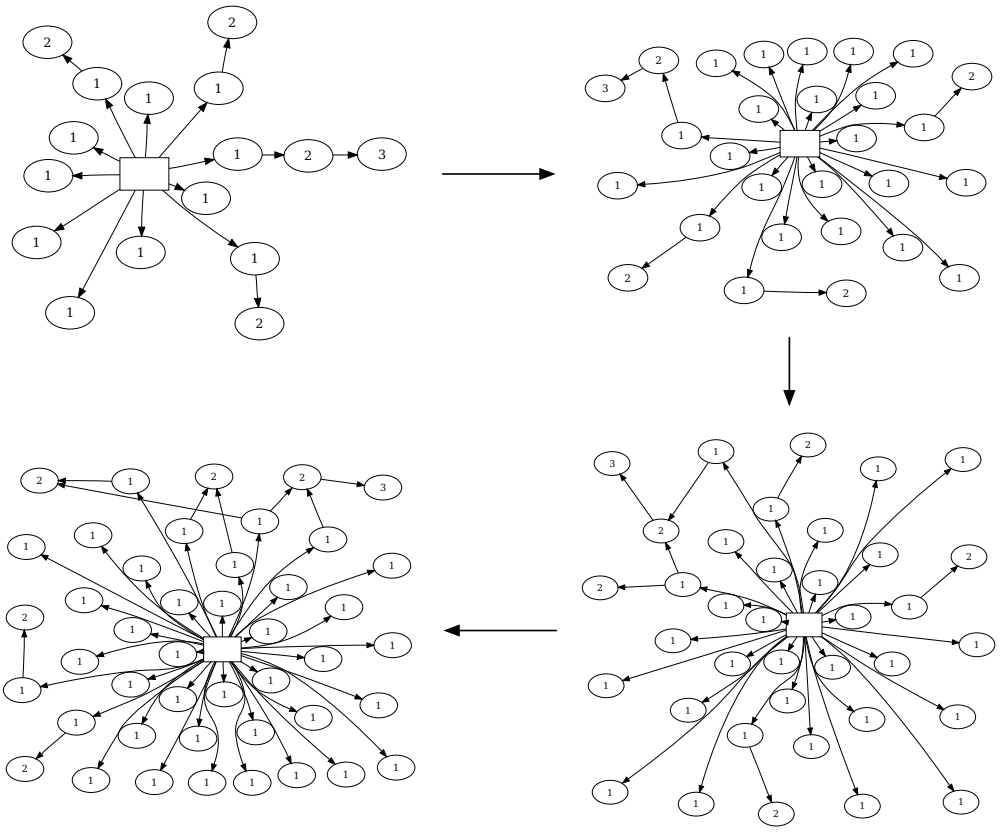


Abbildung 5.15: Einige Individuen auf der Pareto-Front bei Optimierung nach Overlap und Coverage: Von geringem Overlap (links oben) zu hoher Coverage (links unten). Die Zahl in den Clustern gibt jeweils ihre Tiefe im Clustering an.

den. Es ist daher interessant zu betrachten, wie diese beiden Kriterien miteinander in Beziehung stehen. Abbildung 5.16 zeigt die Werte der Individuen für Coverage gegenüber den Werten für Completeness, wenn Overlap und Coverage als Fitnessfunktionen verwendet werden. Die andere Richtung wird später betrachtet.

Overlap soll im Folgenden durch das Kriterium Childcount ersetzt werden. Abbildung 5.17 zeigt die Werte der Individuen für Overlap gegenüber der Werte für Childcount. Die alten und neuen Kriterien stehen offensichtlich miteinander in Beziehung. Es wurde im Rahmen der Untersuchung von zufällig erzeugten Clusterings in Kapitel

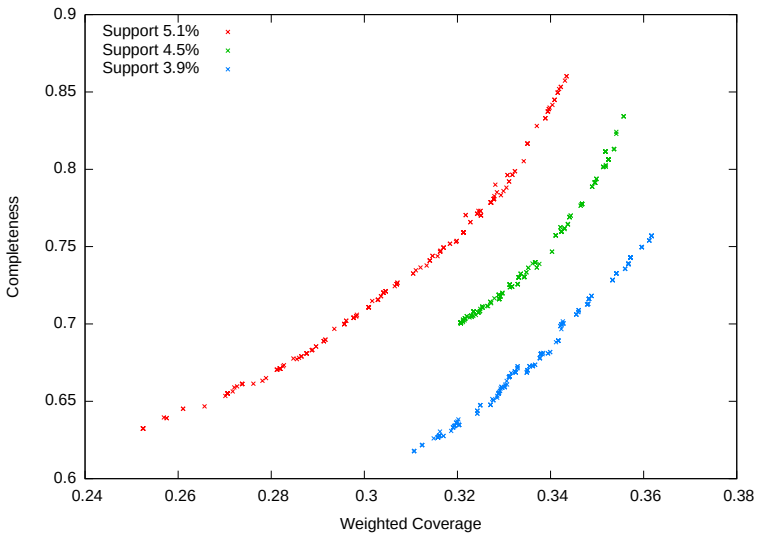


Abbildung 5.16: Coverage gegenüber Completeness bei Optimierung nach Overlap und Coverage

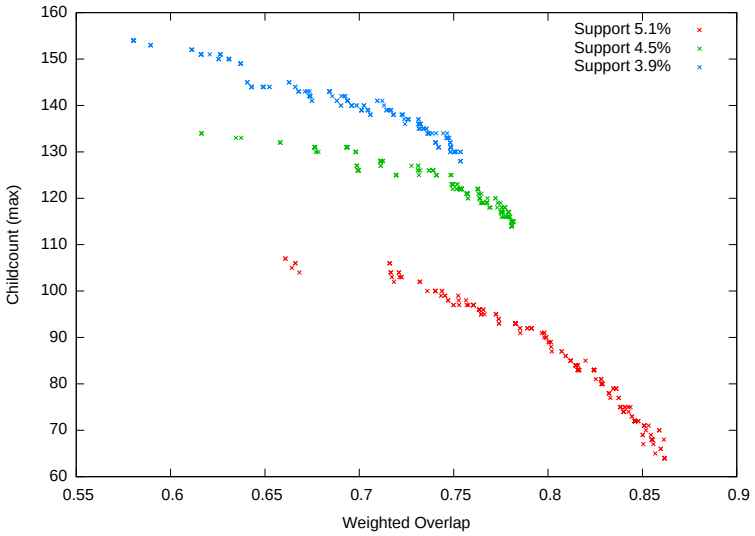


Abbildung 5.17: Overlap gegenüber Childcount bei Optimierung nach Overlap und Coverage

5.2 bereits angemerkt, dass dies zu erwarten ist. Zusätzliche Cluster auf Ebene 1 führen zu einem höheren Wert für Coverage und sie erhöhen den Wert für Completeness ebenfalls. Je mehr Kind-Cluster vorhanden sind, umso wahrscheinlicher ist eine Überlappung der Cluster. Für einen höheren Wert für Childcount sinkt daher der Wert für

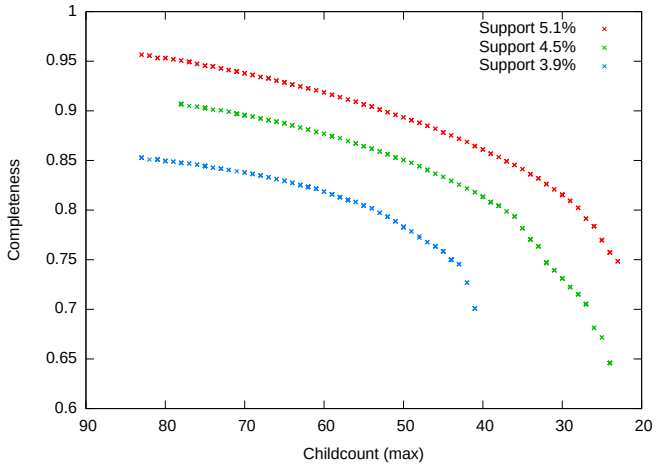


Abbildung 5.18: Pareto-Fronten für Childcount vs. Completeness bei verschiedenen Werten für den Support

Overlap.

5.3.2 Childcount versus Completeness

Abbildung 5.18 zeigt die Pareto-Fronten für die verschiedenen Supportwerte, wenn childcount$_{max}$ und completeness als Fitnessfunktionen verwendet werden. Je geringer der Supportwert ist, umso größer ist der minimale Wert eines Individuums für Childcount und umso größer ist der maximale Wert eines Individuums für Completeness.

Die durchschnittlichen Tiefen der Blätter der erzeugten Clusterings sind in Abbildung 5.19 dargestellt. Die Abbildung zeigt die Completeness aller Individuen gegenüber der durchschnittlichen Tiefe aller jeweils enthaltenen Blätter.

Im Gegensatz zur Optimierung mit Overlap und Coverage wird die Tiefe der Clusterings nun explizit mit optimiert. Da durch Completeness die Ähnlichkeit zum zugrunde liegenden FTS-Clustering gemessen wird, führt eine Optimierung dieses Kriteriums zur Selektion von Clusterings, die eine ähnliche Tiefe haben. Je besser Completeness erfüllt wird, umso tiefere Cluster werden innerhalb eines Clusterings selektiert. Die maximal erreichbare durchschnittliche Tiefe ist wiederum durch das FTS-Clustering vorgegeben. Für einen Support von 3.9% liegt sie bei 7, für 4.5% bei 5 und für 5.1% bei 4. Ebenso wie zuvor ist die Tiefe der Clusterings für alle Individuen einer Front sehr ähnlich. Das liegt daran, dass hohe Tiefe und geringer Childcount keine konträren Ziele sind, sondern gleichzeitig realisiert werden können. Für einen hohen Supportwert sind die Clusterings wesentlich tiefer als dies unter Verwendung der vorherigen Optimie-

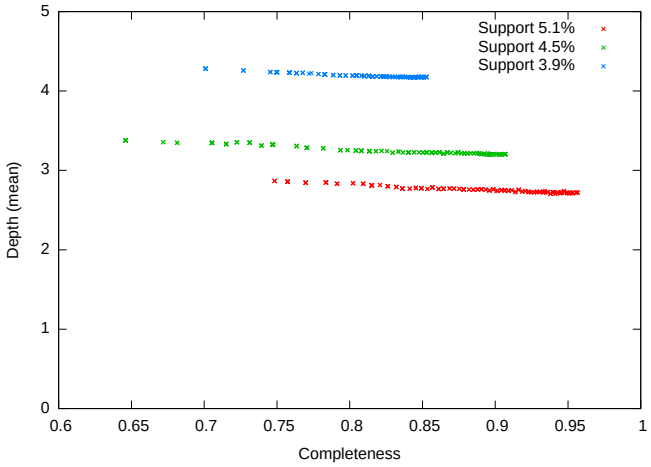


Abbildung 5.19: Completeness der Clusterings gegenüber der durchschnittlichen Tiefe ihrer Blätter bei Optimierung von Childcount und Completeness

rungskriterien der Fall war. Für einen Support von 5.1% liegt die Tiefe nun bei etwa 3, wobei sie vorher bei etwa 2 lag. Für höhere Supportwerte nimmt dieser Vorsprung stetig ab, da sich dann die im Kontext der vorherigen Fitnessfunktionen beschriebenen Effekte eines stark verbundenen FTS-Clusterings auswirken.

Es ist interessant die Beziehungen zwischen den alten und den neuen Kriterien zu betrachten. Es wurde zuvor schon dargestellt, wie sich die Werte der jeweils neuen Kriterien verhalten, wenn gemäß der alten Kriterien optimiert wird. Nun soll die Gegenrichtung betrachtet werden, d.h. wir betrachten die alten Kriterien, während wir die Optimierung gemäß der neuen Kriterien durchführen. Abbildung 5.20 zeigt die Werte für Childcount gegenüber den Werten für Overlap und Abbildung 5.21 zeigt die Werte für Completeness gegenüber den Werten für Coverage.

Individuen mit einem geringen Wert für den Childcount haben weniger sich überlappende Cluster. Der Wert für Overlap liegt also höher. Je größer die maximale Anzahl Kind-Cluster ist, umso größer ist die Wahrscheinlichkeit für Überlappungen der Cluster und umso geringer ist der Wert für Overlap.

Wird Completeness optimiert, so wird Coverage ebenfalls optimiert, da alle Cluster auf Ebene 1, welche für eine Erhöhung des Werts für Coverage sorgen, bei Selektion auch für einen höheren Wert für Completeness sorgen.

In Abbildung 5.22 sind wiederum Darstellungen einiger Individuen als Graphen zu finden. Das Experiment wurde mit einem Supportwert von 7% durchgeführt. Links oben ist das Individuum mit dem geringsten Wert für Childcount dargestellt. Es wurden pro Cluster wenig Kind-Cluster selektiert. Da die Tiefe der Clusterings mit opti-

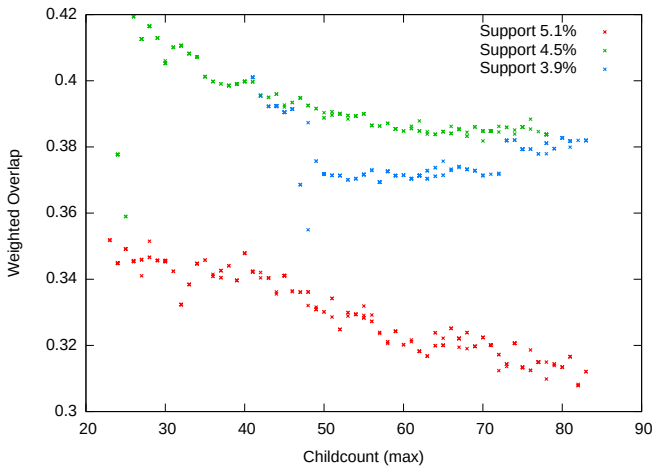


Abbildung 5.20: Childcount gegenüber Overlap bei Optimierung nach Completeness und Childcount

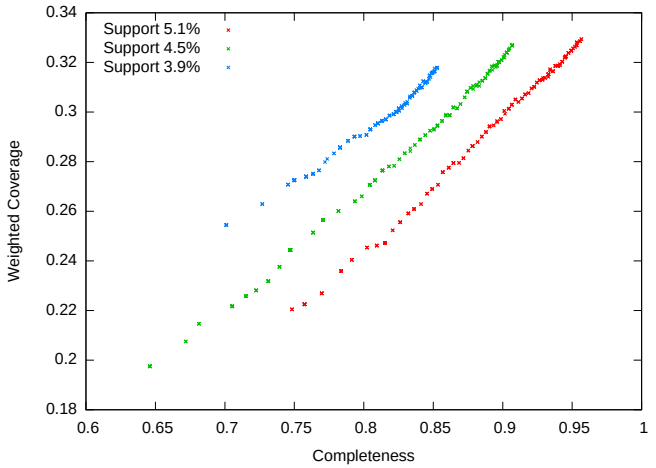


Abbildung 5.21: Completeness gegenüber Coverage bei Optimierung nach Completeness und Childcount

miert wird, wiederholt sich dies auch in den Kind-Clustern. Es entsteht ein Clustering von einfacher Struktur mit einer hohen durchschnittlichen Tiefe. Diese Selektionsstrategie kommt auch bei den anderen Individuen zur Anwendung, nur werden zunehmend mehr Kind-Cluster selektiert, um einen höheren Wert für Completeness zu erreichen.

Abbildung 5.23 zeigt einen Vergleich des jeweils strukturell einfachsten Individuums bei Optimierung gemäß der alten und neuen Kriterien. In Kapitel 5.2 wurden die

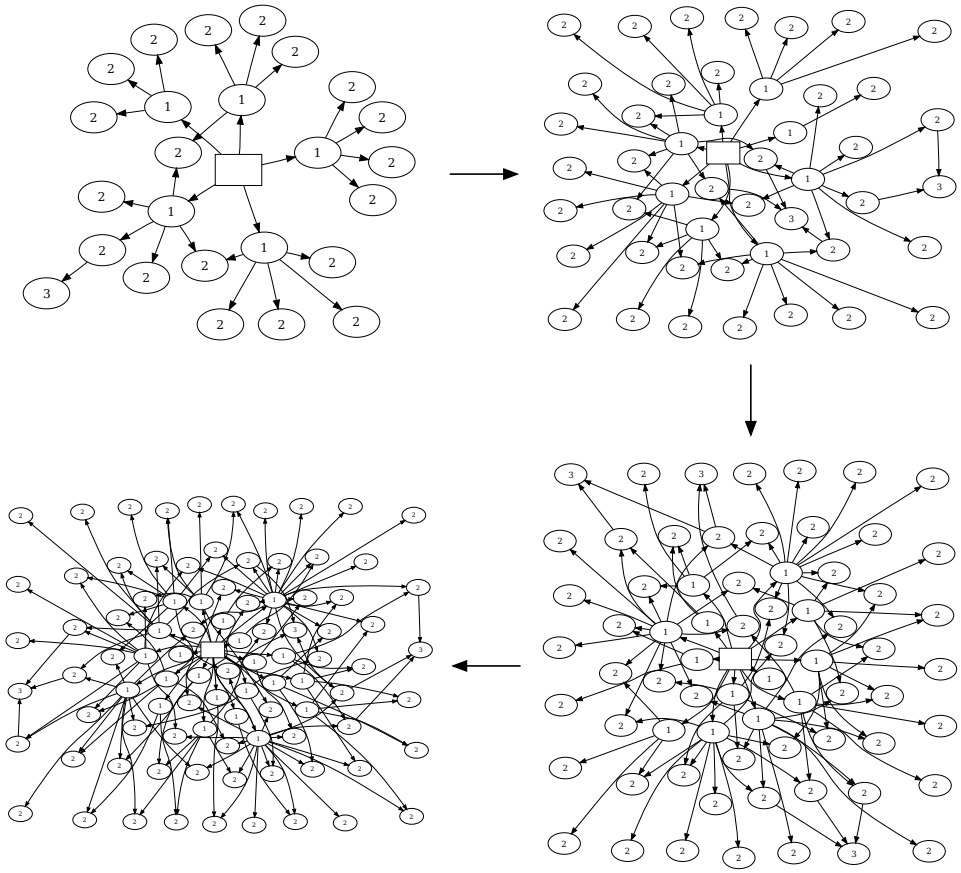


Abbildung 5.22: Einige Individuen der Pareto-Front bei Optimierung nach Childcount und Completeness: Von geringem Childcount (links oben) zu großer Completeness (links unten)

Beziehungen zwischen den interessantesten Bewertungsfunktionen unter Verwendung zufällig erzeugter Clusterings untersucht. Die dazu erstellten Plots wurden zusätzlich auch für die Ergebnismenge von Clusterings bei Anwendung des Verfahrens mit den Zielfunktionen Overlap und Coverage sowie Childcount und Completeness angefertigt und sind im Anhang A zu finden.

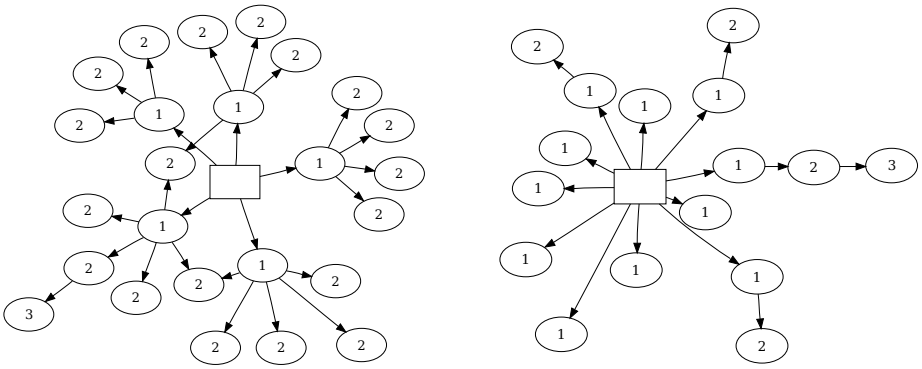


Abbildung 5.23: Das einfachste Individuum für Completeness vs. Childcount (links) und Coverage vs. Overlap (rechts) im Vergleich

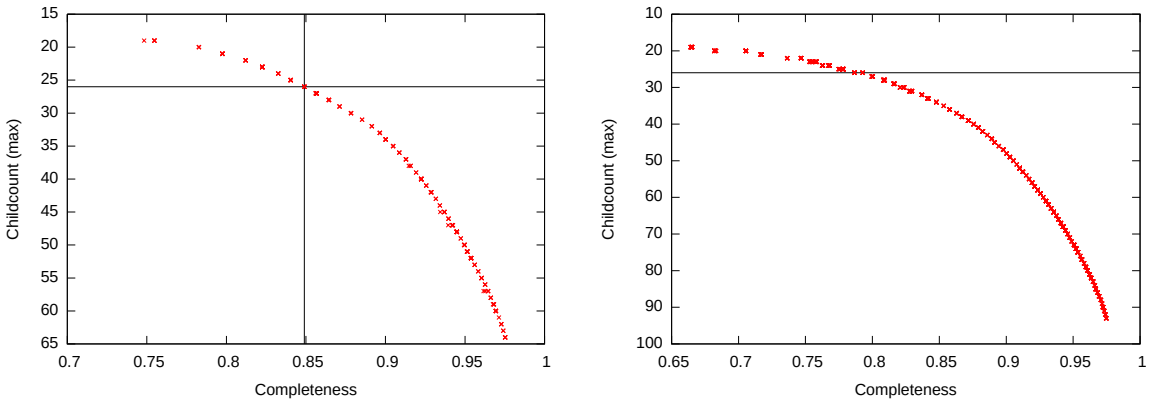
5.4 Inkrementelle Erweiterung

Zur Erprobung der inkrementellen Erweiterung des Verfahrens wurden aus der Folksonomy zum Bibsonomy-Datensatz Folksonomies für zwei aufeinanderfolgende Zeitpunkte t und $t + s$ erstellt (siehe auch Definition 4.39). Für die Folksonomy $\mathbb{F}_t$ wurde aus Y die Teilmenge Y_t aller Taggings von September 1995 bis Mai 2006 ausgewählt. Die Mengen U, R und T wurden entsprechend der Tupel aus Y_t auf U_t, R_t und T_t eingeschränkt. Als Folksonomy für den Zeitpunkt $t + s$ wurde die gesamte Folksonomy $\mathbb{F}$ verwendet, d.h. es gilt $\mathbb{F}_{t+s} = \mathbb{F}$. Y_{t+s} enthält also alle Taggings von September 1995 bis April 2007. Die Kardinalitäten der Menge der beiden Folksonomies sind in Abbildung 5.24 dargestellt.

Die Parametrisierung des Verfahrens entsprach der zuvor schon verwendeten. Zur Bestimmung des Supports der Tagsets wurde wiederum die Funktion supp_U verwendet. Der minimale Support für ein Tagset wurde für $\mathbb{F}_t$ auf ca. 10% festgelegt. Nach Anwendung des Verfahrens auf die Folksonomy $\mathbb{F}_t$ unter Verwendung der Fitnessfunktionen Childcount und Completeness wurde aus der Pareto-Front ein Referenz-Clustering

	$\lvert U\rvert$	$\lvert R\rvert$	$\lvert T\rvert$	$\lvert Y\rvert$
$\mathbb{F}_t$	240	35000	11300	144000
$\mathbb{F}_{t+s}$	780	59000	25000	333000

Abbildung 5.24: Kardinalitäten aller Mengen der Folksonomy für die Zeitpunkte t und $t + s$



(a) (b)

Abbildung 5.25: Selektion eines Referenz-Clusterings aus der Pareto-Front zur Folksonomy $\mathbb{F}_t$ (a) und Definition der Schnittebene zur Folksonomy $\mathbb{F}_{t+s}$ (b)

ausgewählt (siehe Abbildung 5.25a).

Für die Folksonomy $\mathbb{F}_{t+s}$ wurde unter Verwendung der Kriterien Childcount, Completeness und Similarity eine neue Menge von Clusterings erzeugt, wobei zur Berechnung der Similarity das zuvor ausgewählte Clustering als Referenz verwendet wurde. Um eine gute Abdeckung der Pareto-Front trotz der zusätzlichen Dimension zu ermöglichen, wurde die Größe der Population auf 1536 Individuen erhöht. Der minimale Support wurde auf 5% festgelegt. Alle weiteren Parameter des Verfahrens blieben unverändert.

Wie in Kapitel 4.6 beschrieben, wählt der Benutzer nach der erneuten Anwendung des Verfahrens die Individuen nicht direkt aus der dreidimensionalen Pareto-Front (siehe Abbildung 5.26) aus, sondern er definiert zunächst eine Schnittebene (siehe Abbildung 5.25b). Das Referenz-Clustering hatte einen Wert von 26 für Childcount. Der entsprechende Schnitt ist in Abbildung 5.27b dargestellt. Die Abbildungen 5.27a und 5.27c zeigen zwei weitere Schnitte für die Werte 20 und 30. Keiner der gezeigten Schnitte enthält eine zweidimensionale Pareto-Front und aus der Form der dreidimensionalen Front kann geschlossen werden, dass dies auch für keinen anderen Schnitt der Fall ist.

Das Kriterium Similarity steht weder mit Childcount noch mit Completeness in Konkurrenz. Individuen mit einem Wert größer 26 für den Childcount werden durch das Verfahren so selektiert, dass stets möglichst viele Cluster des Referenz-Clusterings vorhanden sind. Dies gelingt für alle Individuen gleichermaßen gut und daher ist der Wert für Similarity sehr konstant. Fällt der Wert aber unter 26, so fehlt der „Platz“

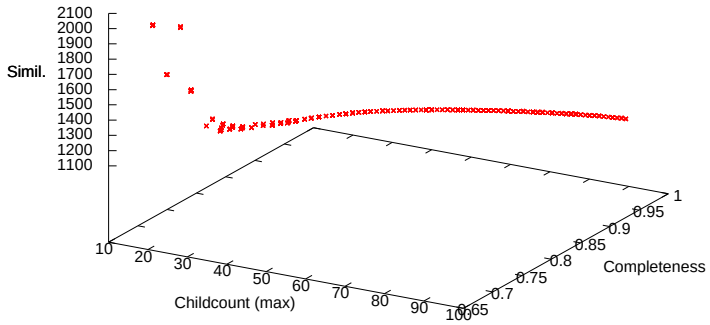
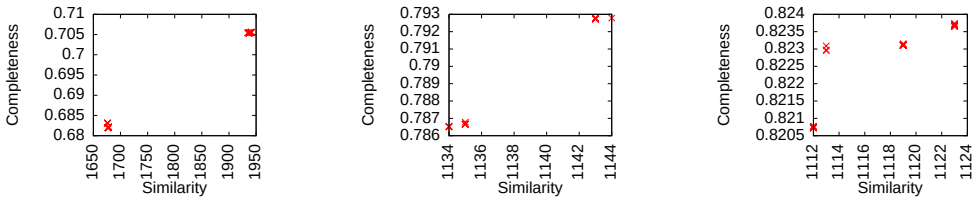


Abbildung 5.26: Dreidimensionale Ansicht der Pareto-Front zur Folksonomy $\mathbb{F}_{t+s}$ für verschiedene Werte für Childcount

für viele der Cluster und der Wert für Similarity steigt entsprechend an. Der Effekt des dritten Kriteriums insgesamt ist, dass die *zweidimensionale* Pareto-Front im dreidimensionalen Raum quasi in Richtung geringer Werte für Similarity gezogen wird.

Der in Kapitel 4.6 beschriebene Anwendungsfall der Erforschung des Suchraums nach zu einem Referenz-Clustering ähnlichen Clusterings ist nicht durchführbar, da die verwendeten drei Kriterien nicht ausreichend konkurrieren.



(a) (b) (c)

Abbildung 5.27: Drei zweidimensionale Schnitte der Pareto-Front zur Folksonomy $\mathbb{F}_{t+s}$ für verschiedene Werte für Childcount: (a) 20, (b) 26, (c) 30

Kapitel 6

Zusammenfassung

Tagging-Systeme ermöglichen dem Benutzer die Strukturierung von Ressourcen durch Annotation von Tags. Während die Redaktion eines Web-Verzeichnisses oder der Autor eines Semantic Web-Angebots die Strukturierung der Ressourcen für den Benutzer vorgibt, ermöglichen Tagging-Systeme ihm eine freie Strukturierung nach den eigenen Vorstellungen. Tagging-Systeme sind einfach zu bedienen, skalierbar und ihr Datenbestand ist in der Regel sehr aktuell.

Die heute vorhandenen Möglichkeiten des Benutzers zur Navigation im Datenbestand eines Tagging-Systems sind sehr eingeschränkt; die Beziehungen zwischen den Tags bleiben weitestgehend unbekannt und ein zielgerichtetes Browsing ist nicht möglich. Vorhandene Ansätze zur Verbesserung der Navigation bestehen in der Regel darin, eine informativere Tag-Cloud zu erstellen. Dazu wurden von Hassan-Montero et al. [19] ein klassisches Clustering-Verfahren, von Begelman et al. [3] und Kaser et al. [24] Verfahren aus dem Bereich des Clusterings von Graphen eingesetzt. Diese Verfahren zeigen ein schlechtes Skalierungsverhalten, da sie für die zu verarbeitende, große Menge hochdimensionaler Daten nicht geeignet sind. Für Eingaben dieser Art eignen sich Frequent Itemset-basierte Clustering-Verfahren sehr viel besser. Das von Ester et al. [12] ursprünglich für den Bereich des Clusterings von Textdokumenten entwickelte Verfahren Fihc lässt sich aufgrund der Nähe zur Problemdomäne des Clusterings von Tags für diese Anwendung anpassen. Zum einen werden dadurch die Skalierungsprobleme der anderen Verfahren gelöst, zum anderen ergibt sich, anstatt einer leicht verbesserten Tag-Cloud, eine hierarchische Clusterstruktur, die dem Benutzer sehr vielfältige Möglichkeiten zur Navigation durch die Tagging-Daten bietet. Allen Ansätzen ist allerdings gemein, dass nur eine einzelne Lösung erzeugt wird und dass der Benutzer den Erzeugungsprozess nicht beeinflussen kann; persönliche Anforderungen an die Clusterstruktur - für die spätere effektive Verwendbarkeit sehr wichtig - sind nicht umsetzbar. Während der Erzeugung der Lösung werden von den Verfahren implizite Kriterien optimiert; da diese dem Benutzer nicht bekannt sind, wird eine Interpretation der Lösung erschwert oder sogar verhindert.

Im Rahmen dieser Arbeit wurde ein Verfahren zur Erzeugung individueller, hierarchischer Navigationsstrukturen für Tagging-Systeme entwickelt. Aufgrund der Nutzung eines FIM-Verfahrens zur Vorverarbeitung der Eingabedaten, weist es ein sehr gutes Skalierungsverhalten bei Erhöhung der Datenmenge und Dimensionalität auf.

Die bei der Erzeugung optimierten Eigenschaften der Navigationsstrukturen werden als Kriterien in einem multikriteriellen Suchproblem explizit dargestellt. Angesichts der Konkurrenz dieser Eigenschaften wird anstatt einer Lösung eine Menge alternativer Lösungen erzeugt. In Kombination mit der expliziten Darstellung der Kriterien kann der Benutzer eine Lösung auswählen, die seinen persönlichen Anforderungen am besten entspricht. Die Auswirkung der Änderung des Wertes eines Kriteriums ist ihm dabei intuitiv verständlich.

Zum Teil bieten bestehende Verfahren dem Benutzer durch versteckte Parameter die Möglichkeit, die erzeugte Lösung zu beeinflussen; beim Verfahren FIHC kann die beim Pruning des Cluster-Baums verwendete Heuristik als ein solcher Parameter interpretiert werden. Die Exploration des Lösungsraums durch Parameteranpassungen ist sehr mühsam und unintuitiv, da es nur schwer möglich ist, einen Zusammenhang zwischen der Änderung eines Parameters und der Auswirkung auf die erzeugte Lösung herzustellen. Im Gegensatz dazu bietet der hier vorgestellte multikriterielle Ansatz, durch alternative Lösungsvorschläge und die explizite Darstellung der Eigenschaften der angebotenen Alternativen, eine sehr einfache und intuitive Möglichkeit, den Lösungsraum zu erforschen.

Um die Auswahl einer Lösungsalternative für den Benutzer so einfach wie möglich zu gestalten, wurden zwei Optimierungskriterien verwendet. Im Rahmen dieser Arbeit wurden zwei Sätze von Kriterien vorgestellt. Die ersten beiden Kriterien waren Overlap und Coverage, welche zur Erzeugung von Navigationsstrukturen führen, die zum einen die Menge der Ressourcen möglichst partitionieren und zum anderen viele Ressourcen enthalten. Die im Datenbestand eines Tagging-Systems gleichberechtigt existierenden, verschiedenen Meinungen der Benutzer sorgen für eine große Überlappung der Tag-Extensionen. Wird der Overlap optimiert, geht ein Großteil dieser Information verloren. Die Tiefe der Navigationsstrukturen, welche sehr hilfreich für ein zielgerichtetes Browsing ist, wird nicht mit optimiert.

Aus diesem Grunde wurde ein zweiter Satz von Kriterien, Childcount und Completeness, entwickelt. Die Optimierung der Completeness führt zu Strukturen, die erstens viele der Ressourcen aus den Tagging-Daten enthalten und zweitens sehr tief sind. Die gleichzeitige Minimierung des Childcount sorgt für Strukturen, die dem Benutzer beim Browsing eine übersichtliche Anzahl von Navigationsmöglichkeiten bieten. Somit wird das Ziel der Erzeugung einfacher, vollständiger und tiefer Strukturen erreicht, ohne dabei die Heterogenität der Tagging-Daten zu verlieren.

Neben diesen vier Kriterien wurden weitere interessante Eigenschaften der Navigationsstrukturen, wie beispielsweise die strukturelle Ähnlichkeit zu einem Baum oder das Verhältnis von Blättern zu inneren Knoten, als formale Bewertungsfunktionen formuliert. Die Beziehungen dieser Funktionen und damit der verschiedenen Eigenschaften wurden anschließend empirisch untersucht.

Das entwickelte Verfahren wird im folgenden Abschnitt noch einmal im Überblick dargestellt. Es folgt ein Ausblick auf vorhandene Möglichkeiten, das Verfahren zu

erweitern oder es für andere Problemdomänen einzusetzen.

6.1 Multikriterielles Tagset-Clustering

Mit dem Tagset-Clustering (siehe Definition 4.3) wird ein formales Modell für eine hierarchische Navigationsstruktur auf Basis von Tagsets definiert. Ein Tagset-Clustering ist eine Einschränkung des Tagset-Verbands (siehe Gleichung 4.1). Zum einen ist die Relation zwischen den Tagsets von der Teilmengen-Relation auf eine Relation $\prec$ eingeschränkt, die nur Mengen in Relation setzt, deren Kardinalität sich um den Wert 1 unterscheiden. Der transitive Abschluss von $\prec$ ergibt wieder die Teilmengen-Relation. Zum anderen wird nur eine Teilmenge der Menge aller Tagsets betrachtet. Diese Teilmenge wird als Clustermenge bezeichnet. An die Clustermenge werden einige Anforderungen gestellt: Das leere Tagset muss immer enthalten sein und die Tagsets müssen zusammen mit $\prec$ eine Zusammenhangskomponente bilden.

Neben in Relation zueinander gesetzten Tagsets enthält ein Tagset-Clustering Information über die Zugehörigkeit von Ressourcen zu Tagsets, die durch die zugrunde liegende Folksonomy vorgegeben wird. Verschiedene Tagset-Clusterings zu einer Folksonomy unterscheiden sich in der Wahl der Clustermenge. Das größtmögliche Clustering ist das vollständige Tagset-Clustering (siehe Definition 4.8), das kleinstmögliche Clustering enthält nur das leere Tagset. Diese beiden Extrema und alle gültigen Clustermengen dazwischen bilden den Suchraum (siehe Definition 4.7), der bei der Suche nach geeigneten Navigationsstrukturen betrachtet wird.

Die Häufigkeit eines Tagsets kann auf Benutzern, Ressourcen oder Taggings definiert werden (siehe Definition 4.9). Ein Tagset-Clustering mit einer gemäß eines Supportwerts bestimmten Clustermenge von Frequent Tagsets heißt Frequent-Tagset-Clustering (siehe Definition 4.10). Eine Strategie zur Erforschung des Suchraums ist das iterative Betrachten von FTS-Clusterings für verschiedene Supportwerte. Es wird dabei nur ein Teil des Suchraums (siehe Gleichung 4.14) betrachtet; die Suche ist sehr grob. Das Suchverfahren wird daher in zwei Schritte aufgeteilt.

Im ersten Schritt des Verfahrens wird ein FTS-Clustering mit einem möglichst geringen Support erstellt. Im zweiten Schritt werden dann alle gültigen Teilstrukturen dieses FTS-Clusterings betrachtet (siehe Gleichung 4.15). Eine gültige Teilstruktur ist jedes Tagset-Clustering, dessen Clustermenge eine gültige Teilmenge der Clustermenge des FTS-Clusterings ist. Ein solches Clustering wird als Filtered-Frequent-Tagset-Clustering (siehe Definition 4.13) bezeichnet.

Ergebnis des zweiten Schritts des Verfahrens ist eine Menge alternativer Tagset-Clusterings. Der Prozess der Auswahl der geeigneten FFTS-Clusterings wird als multikriterielles Suchproblem formuliert. Zur Steuerung der Suche werden zwei Fitnessfunktionen (siehe Definition 4.14) verwendet. Mit dem Verfahren NSGA-II (siehe Kapitel 3.5.2) wird die Alternativenmenge bestimmt. Diese wird dem Benutzer als zweidimen-

sionales Koordinatensystem präsentiert, in dem jede alternative Lösung gemäß ihrer beiden Fitnesswerte eingetragen ist. Dadurch ist eine intuitive Erforschung des Lösungsraums möglich, da die Änderungen der Fitnesswerte und ihre Auswirkungen auf das erzeugte Clustering explizit dargestellt werden.

Es werden verschiedene Bewertungsfunktionen für Clusterings eingeführt. Darunter sind Funktionen zur Bestimmung der Überlappung der Cluster, der Gesamttiefe des Clusterings, des Anteils durch das Clustering abgedeckter Ressourcen, der Ähnlichkeit des Clusterings zu einem minimalen Spannbaum.

Schließlich wird eine Erweiterung des Verfahrens auf einen zeitlich veränderlichen Datenbestand definiert. Dazu wird eine dritte Fitnessfunktion verwendet, die die strukturelle Ähnlichkeit zu einem Referenz-Clustering misst (siehe Definition 4.40). Der Benutzer kann so den Lösungsraum in Richtung zum Referenz-Clustering strukturell sehr ähnlicher Clusterings erforschen. Strukturell ähnliche Clusterings sind für ihn interessant, da er gewohnte Strukturierungen weiterverwenden kann und sich nicht bei jeder Neuanwendung des Verfahrens an völlig neue Strukturierungen anpassen muss. Der Prozess der Auswahl eines Clusterings durch den Benutzer wird zweistufig gestaltet. Anstatt einer unintuitiven Auswahl aus einer dreidimensionalen Pareto-Front, trifft der Benutzer zwei Auswahlen aus jeweils zweidimensionalen Fronten.

6.2 Ausblick

Das in dieser Arbeit entwickelte Verfahren wurde zwar zur Lösung eines konkreten Problems eingesetzt, ist aber darüberhinaus als ein Framework zu verstehen, da es viele Möglichkeiten für Anpassungen bietet. Die Basis dieses Frameworks bilden der vorgestellte Formalismus zur Beschreibung hierarchischer Navigationsstrukturen für Tagging-Systeme und das verwendete multikriterielle Optimierungsverfahren.

Bei den in Definition 4.9 vorgestellten Möglichkeiten zur Definition der Häufigkeit eines Tagsets handelt sich jeweils um eine Projektion aus dem Datenbestand einer Folksonomy in die Begriffswelt der FIM-Algorithmen. Neben weiteren Möglichkeiten der Definition der Häufigkeit eines Tagset bietet sich so auch die Möglichkeit, beispielsweise Benutzer anstatt von Tags zu betrachten und die in einem Tagging-System vorhandenen Benutzergruppen zu untersuchen. Die Arbeit von Schmitz et al. [34] ist diesbezüglich sehr interessant.

Je nach Anwendungsfall ist die Verwendung unterschiedlicher Optimierungskriterien sinnvoll. Während sich die Kriterien Childcount und Completeness für die Erzeugung von Navigationsstrukturen als besonders geeignet herausgestellt haben, kann sich eine Optimierung nach Overlap und Coverage beim automatischen Erstellen von Ontologien als sinnvoll erweisen. Das Verfahren lässt sich darüberhinaus um beliebige, neue Optimierungskriterien erweitern.

Schließlich ist auch die Anwendung des Verfahrens in einer anderen Problemdomäne

möglich. Wie in Kapitel 3.1 erläutert wurde, ist das Clustering von Tags dem Clustering von Textdokumenten sehr ähnlich. Daher ließe sich das Verfahren ohne größere Anpassungen auch für das multikriterielle Clustering von Textdokumenten einsetzen.

Literaturverzeichnis

[1] Agrawal, Rakesh, Tomasz Imielinski und Arun N. Swami: *Mining Association Rules between Sets of Items in Large Databases.* In: *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Seiten 207–216, 1993.

[2] Agrawal, Rakesh und Ramakrishnan Srikant: *Fast Algorithms for Mining Association Rules.* In: *Proceedings of the 20th International Conference on Very Large Data Bases*, Seiten 487–499, 1994.

[3] Begelman, Grigory, Philipp Keller und Frank Smadja: *Automated Tag Clustering: Improving search and exploration in the tag space.* In: *Collaborative Web Tagging Workshop at WWW2006*, 2006.

[4] Berners-Lee, Tim, Roy Fielding und Larry Masinter: *Uniform Resource Identifiers (URI): Generic Syntax [RFC3986].* Internet RFCs, 2005.

[5] Berners-Lee, Tim, James Hendler und Ora Lassila: *The Semantic Web.* Scientific American, 284(5):34–43, 2001.

[6] Beume, Nicola, Boris Naujoks und Michael Emmerich: *SMS-EMOA: Multiobjective selection based on dominated hypervolume.* European Journal of Operational Research, 181(3):1653–1669, 2007.

[7] Deb, Kalyanmoy: *Genetic Algorithm in Search and Optimization: The Technique and Applications.* `http://www.iitk.ac.in/kangal/papers/isinew.ps.gz`, 1998.

[8] Deb, Kalyanmoy: *Evolutionary Algorithms for Multi-Criterion Optimization in Engineering Design.* In: *Evolutionary Algorithms in Engineering and Computer Science*, Seiten 135–161. John Wiley & Sons, 1999.

[9] Deb, Kalyanmoy, Samir Agrawal, Amrit Pratab und T. Meyarivan: *A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization:* Nsga-II. In: *Proceedings of the 6th Parallel Problem Solving from Nature Conference*, Seiten 849–858, 2000.

[10] Deb, Kalyanmoy und Deb Kalyanmoy: *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, 2001.

[11] Dewey, Melvil: *A Classification and Subject Index for Cataloguing and Arranging the Books and Pamphlets of a Library*. http://www.gutenberg.org/files/12513/, 1876.

[12] Fung, Benjamin C. M., Ke Wang und Martin Ester: *Hierarchical Document Clustering using Frequent Itemsets*. In: *Proceedings of the 3rd SIAM International Conference on Data Mining*, 2003.

[13] Goethals, Bart: *Survey on Frequent Pattern Mining*. `http://www.adrem.ua.ac.be/bibrem/pubs/fpm_survey.pdf`, 2003.

[14] Golder, Scott und Bernando A. Huberman: *Usage Patterns of Collaborative Tagging Systems*. Journal of Information Science, 32(2):198–208, 2006.

[15] Gulli, Antonio und Alessio Signorini: *The Indexable Web is More than 11.5 Billion Pages*. In: *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, Seiten 902–903, 2005.

[16] Hammond, Tony, Timo Hannay, Ben Lund und Joanna Scott: *Social Bookmarking Tools (I): A General Review*. D-Lib Magazine, 11(4), 2005.

[17] Hammond, Tony, Timo Hannay, Ben Lund und Joanna Scott: *Social Bookmarking Tools (II): A Case Study - Connotea*. D-Lib Magazine, 11(4), 2005.

[18] Han, Jiawei, Jian Pei und Yiwen Yin: *Mining frequent patterns without candidate generation*. In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, Seiten 1–12, 2000.

[19] Hassan-Montero, Yusef und Victor Herrero-Solana: *Improving Tag-Clouds as Visual Information Retrieval Interfaces*. In: *Proceedings of the 2006 International Conference on Multidisciplinary Information Sciences and Technologies*, 2006.

[20] Hillmann, Diane: *Using Dublin Core*. http://dublincore.org/documents/usageguide/, 2005.

[21] Hotho, Andreas, Robert Jäschke, Christoph Schmitz und Gerd Stumme: Bibsonomy*: A Social Bookmark and Publication Sharing System*. In: *Proceedings of the 1th Conceptual Structures Tool Interoperability Workshop at the 14th International Conference on Conceptual Structures*, Seiten 87–102, 2006.

[22] Hotho, Andreas, Robert Jäschke, Christoph Schmitz und Gerd Stumme: *Information Retrieval in Folksonomies: Search and Ranking.* In: *Proceedings of the 3rd European Semantic Web Conference*, Seiten 411–426, 2006.

[23] Jain, Anil K., M. N. Murty und Patrick J. Flynn: *Data clustering: a review.* ACM Computing Surveys, 31(3):264–323, 1999.

[24] Kaser, Owen und Daniel Lemire: *Tag-Cloud Drawing: Algorithms for Cloud Visualization.* In: *Tagging and Metadata for Social Information Organization Workshop at WWW2007*, 2007.

[25] Lamont, Gary B. und David A. Van Veldhuizen: *Evolutionary Algorithms for Solving Multi-Objective Problems.* Kluwer Academic Publishers, 2002.

[26] Lewis, Daniel: *What is web 2.0?* Crossroads, 13(1):3–3, 2006.

[27] Marlow, Cameron, Mor Naaman, Danah Boyd und Marc Davis: *Position Paper, Tagging, Taxonomy, Flickr, Article, ToRead.* In: *Collaborative Web Tagging Workshop at WWW2006*, 2006.

[28] Mathes, Adam: *Folksonomies - Cooperative Classification and Communication Through Shared Metadata.* `http://www.adammathes.com/academic/computer-mediated-communication/folksonomies.html`, 2004.

[29] Mierswa, Ingo, Michael Wurst, Ralf Klinkenberg, Martin Scholz und Timm Euler: *YALE: rapid prototyping for complex data mining tasks.* In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, Seiten 935–940, 2006.

[30] Mitchell, Thomas: *Machine Learning*, Kapitel Genetic Algorithms, Seiten 249–262. McGraw-Hill Education, 1997.

[31] O'Reilly, Tim: *What is Web 2.0. Design Patterns and Business Models for the Next Generation of Software.* `http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html`, 2005.

[32] Organization, National Information Standards: *Understanding Metadata.* Technischer Bericht, National Information Standards Organization, 2004.

[33] Salton, Gerard und Michael J. McGill: *Introduction to Modern Information Retrieval.* McGraw-Hill, 1986.

[34] Schmitz, Christoph, Andreas Hotho, Robert Jäschke und Gerd Stumme: *Mining Association Rules in Folksonomies.* In: *Proceedings of the 2006 IFCS Conference*, Seiten 261–270, 2006.

[35] Setten, Mark van, Rogier Brussee, Harry van Vliet, Luit Gazendam, Ynze van Houten und Mettina Veenstra: *On the Importance of Who Tagged What.* In: *Workshop on the Social Navigation and Community based Adaptation Technologies*, 2006.

[36] Shadbolt, Nigel, Tim Berners-Lee und Wendy Hall: *The Semantic Web Revisited.* IEEE Intelligent Systems, 21(3):96–101, 2006.

[37] Shirky, Clay: *Ontology is Overrated: Categories, Links, and Tags.* `http://www.shirky.com/writings/ontology_overrated.html`, 2005.

[38] Steinbach, Michael, George Karypis und Vipin Kumar: *A Comparison of Document Clustering Techniques.* In: *KDD Workshop on Text Mining*, 2000.

[39] Tonkin, Emma und Marieke Guy: *Folksonomies: Tidying Up Tags?* D-Lib Magazine, 12(1), 2006.

[40] Wal, Thomas Vander: *Folksonomy Coinage and Definition.* `http://vanderwal.net/folksonomy.html`, 2007.

[41] Wang, Ke, Chu Xu und Bing Liu: *Clustering transactions using large items.* In: *Proceedings of the 8th International Conference on Information and Knowledge Management*, Seiten 483–490, 1999.

[42] Zitzler, E., M. Laumanns und L. Thiele: *SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization.* In: *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems*, Seiten 95–100, 2002.

Anhang A

Weitere Statistiken

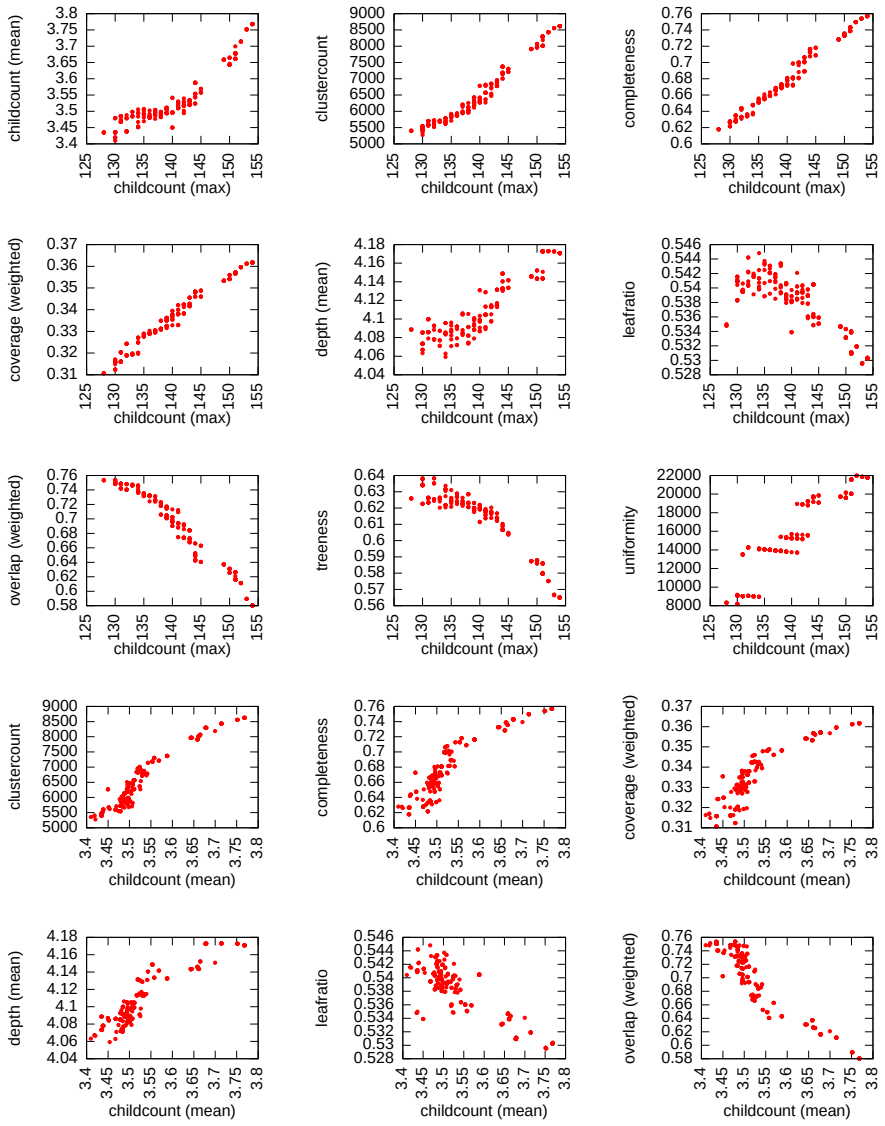


Abbildung A.1: Beziehungen der Bewertungsfunktionen bei Optimierung nach Overlap und Coverage (1/3)

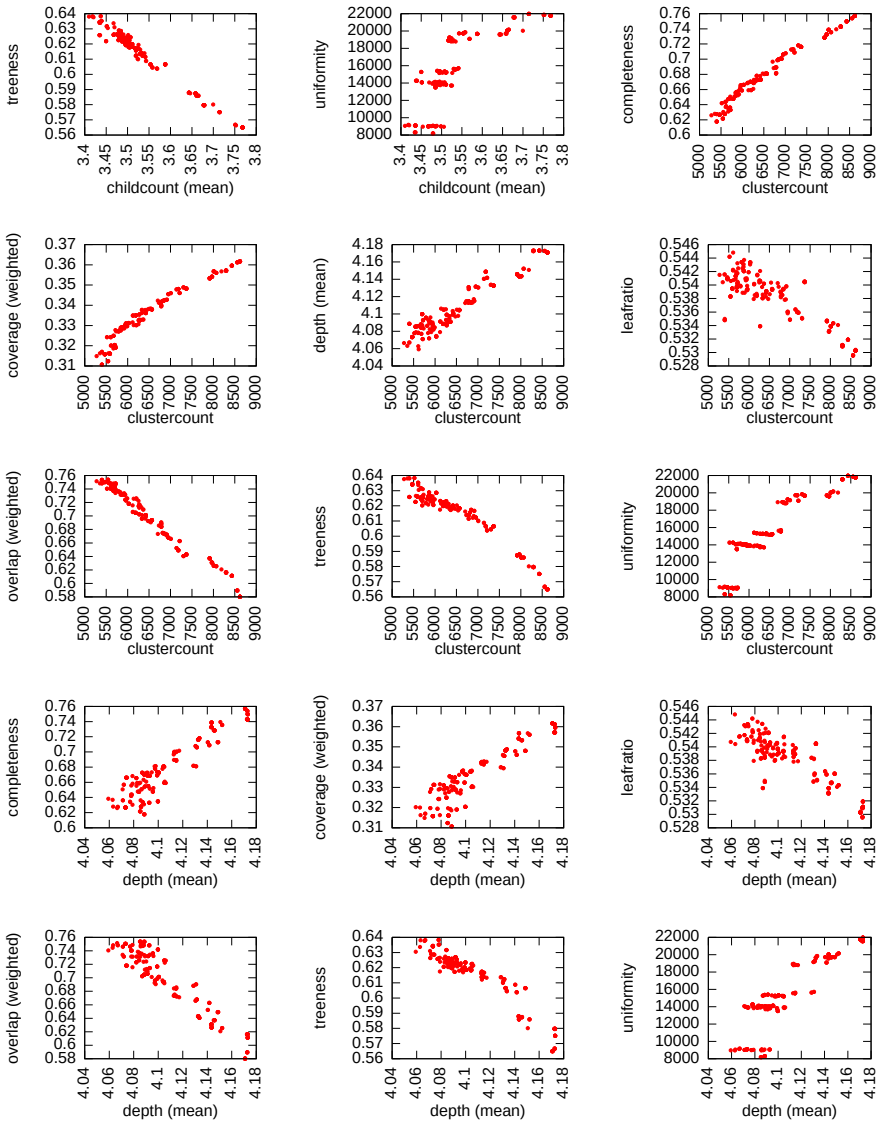


Abbildung A.2: Beziehungen der Bewertungsfunktionen bei Optimierung nach Overlap und Coverage (2/3)

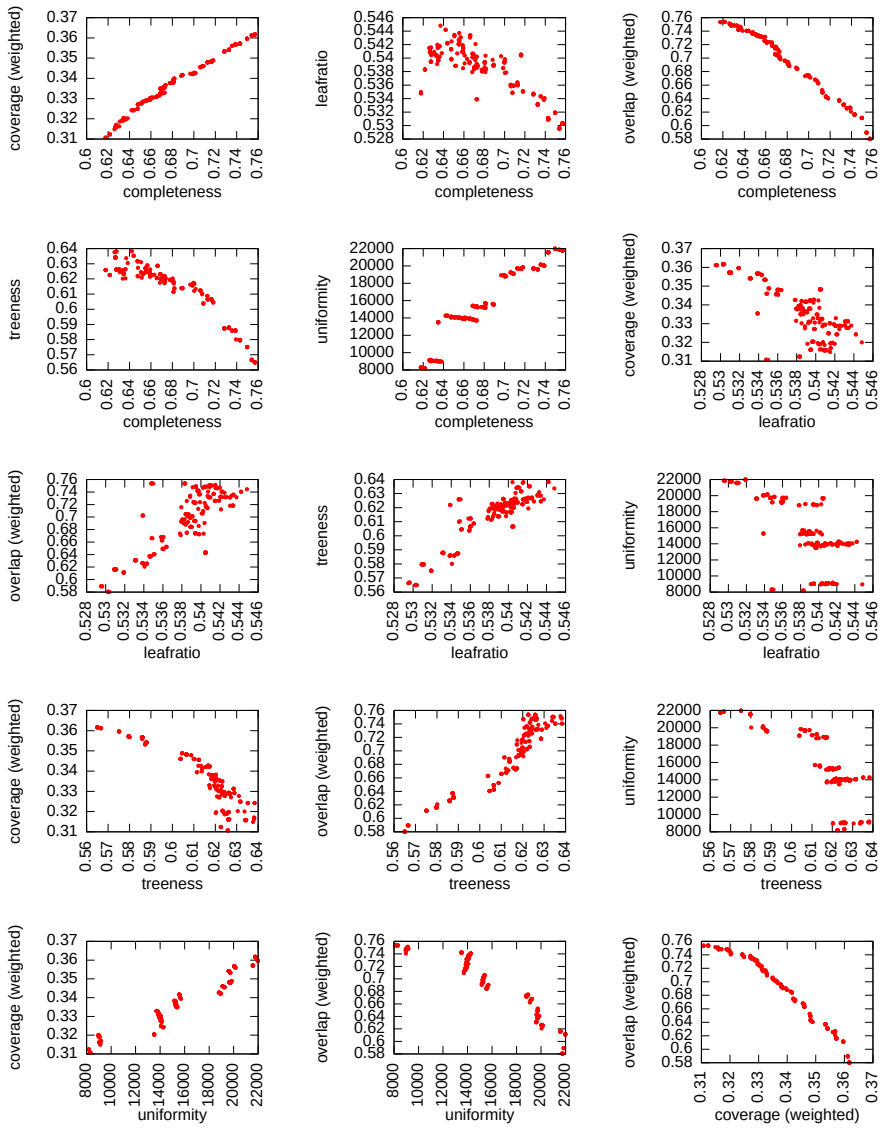


Abbildung A.3: Beziehungen der Bewertungsfunktionen bei Optimierung nach Overlap und Coverage (3/3)

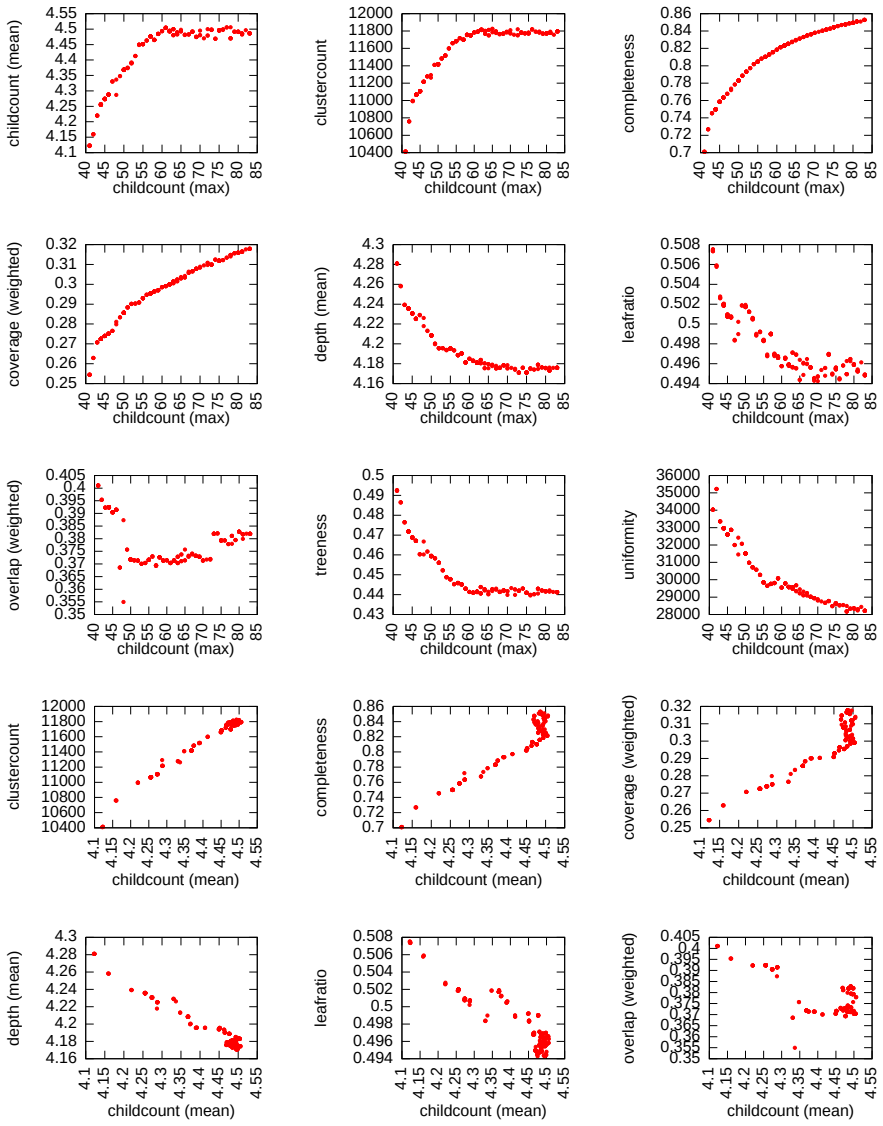


Abbildung A.4: Beziehungen der Bewertungsfunktionen bei Optimierung nach Childcount und Completeness (1/3)

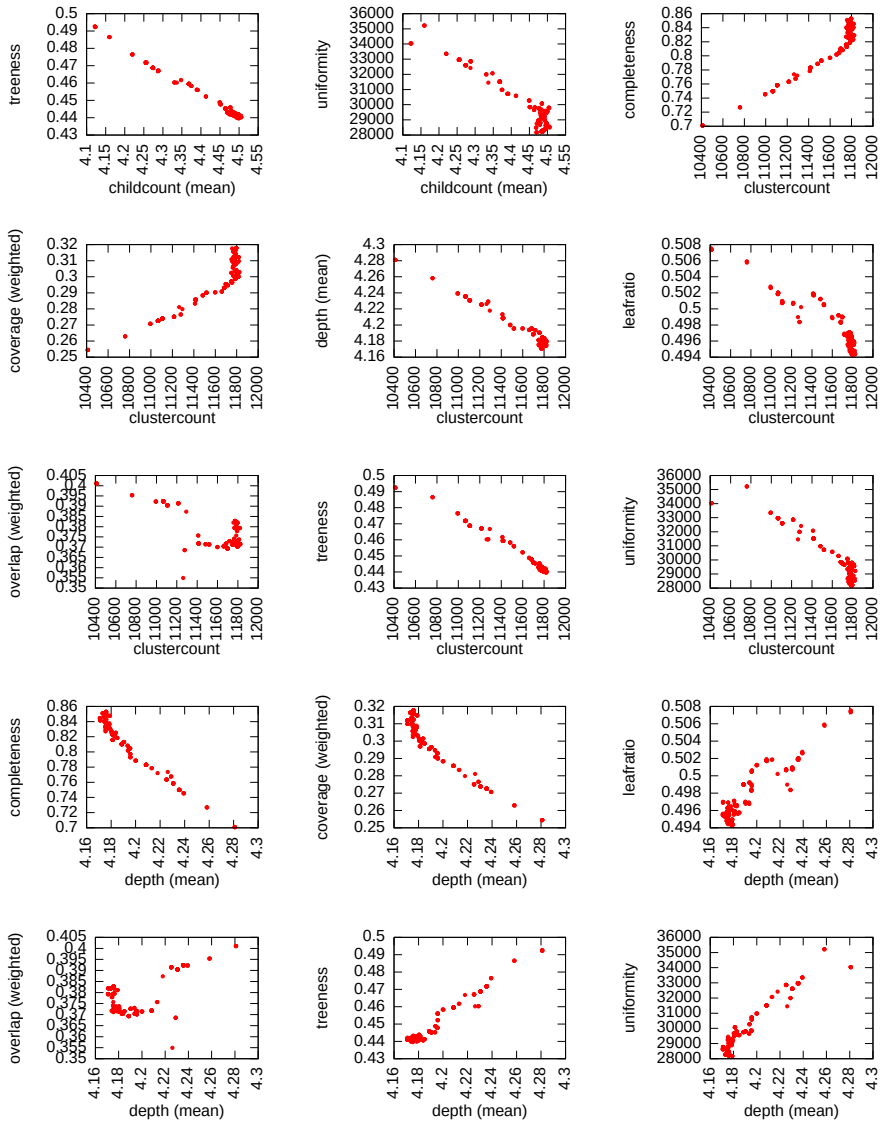


Abbildung A.5: Beziehungen der Bewertungsfunktionen bei Optimierung nach Childcount und Completeness (2/3)

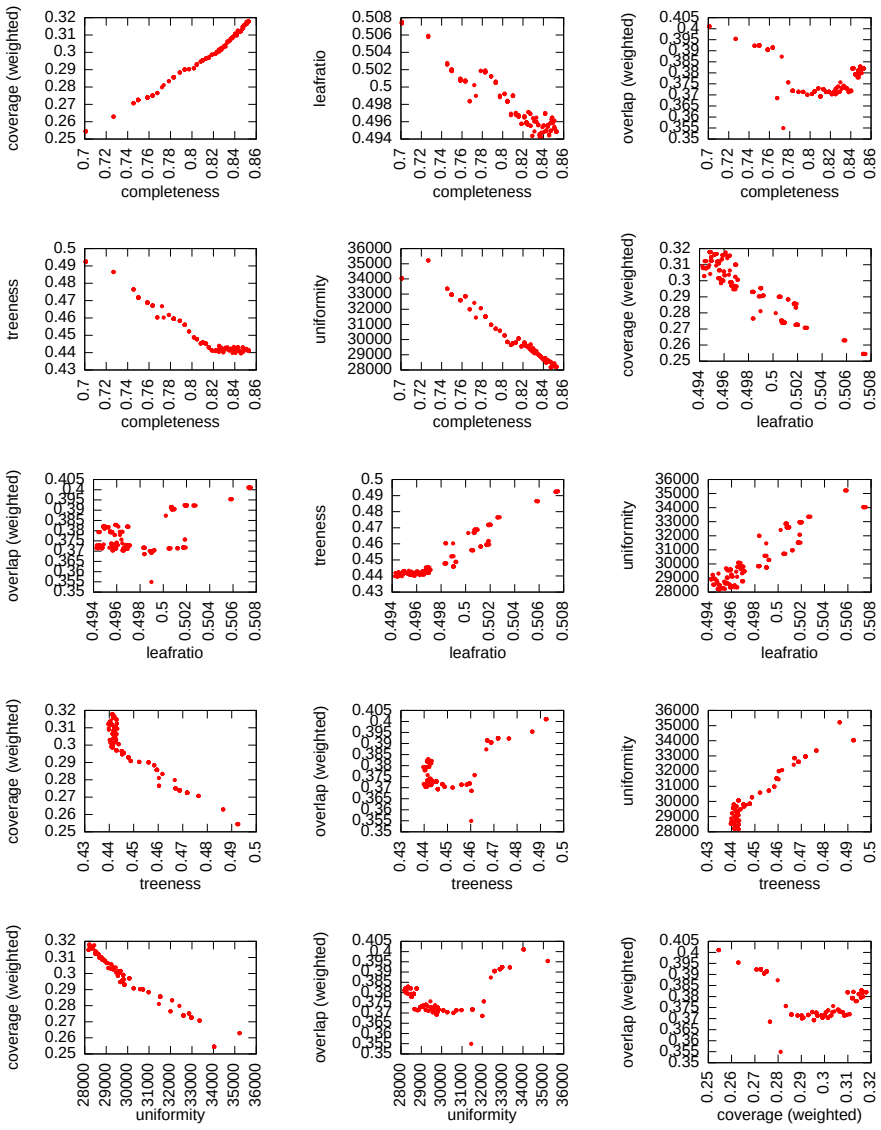


Abbildung A.6: Beziehungen der Bewertungsfunktionen bei Optimierung nach Childcount und Completeness (3/3)

Index