Analyse der Koautorenschaft zur Konsolidierung ambiger Autorennamen – Überblick und Bewertung eines neuen Verfahren

Diplomarbeit

im Studiengang `Informatik`
angefertigt am Fachbereich Informatik
der Technischen Universität Darmstadt

von

`Christian Seufert`

Darmstadt, Juli 2006

Betreuer: Prof. Dr. Johannes Fürnkranz

Ehrenwörtliche Erklärung

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

Rüsselsheim, 31.07.2006

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

1.1. Einführung und Motivation

Data Mining

Mit dem Anstieg der enormen Datenmengen, die in heutigen Unternehmen, Forschungsprojekten oder im Internet entstehen, besteht zunehmend die Notwendigkeit, Techniken zur Analyse und möglicherweise zur Interpretation solcher Daten zu entwickeln. Durch die Analyse soll das in den Daten implizit vorhandene Wissen extrahiert werden, das unter anderem zur Entscheidungsunterstützung in den unterschiedlichsten Prozessen verwendet werden kann. Die traditionellen Methoden der Wissensextraktion stützen sich auf eine manuelle Datenanalyse, die jedoch in Anbetracht des immensen Datenvolumens nicht weiter realisierbar ist. Data Mining, ebenfalls unter dem Namen Knowledge Discovery in Databases (KDD) bekannt, bezeichnet den automatischen Prozess der Identifikation gültiger, neuer, potenziell nützlicher Muster in großen Datenmengen. Obwohl die Begriffe Data Mining und KDD häufig synonym füreinander verwendet werden, bezieht sich KDD auf den gesamten Prozess der Wissensextraktion, wohingegen der Begriff Data Mining nur einen einzelnen, obgleich wesentlichen Teilaspekt dieses Prozesses abdeckt [FPSSU96]. Entsprechend dieser Sichtweise ist unter dem Begriff Data Mining die Anwendung spezifischer Algorithmen zu verstehen, die auf bekannten Techniken des Maschinellen Lernens, der Mustererkennung und der Statistik beruht, um Muster aus Daten zu extrahieren.

Datenbereinigung

Die Analyse von Daten setzt voraus, dass diese in einem bereinigten Zustand vorliegen, in der inkorrekte, unvollständige oder redundante Daten entdeckt und entsprechend behandelt wurden. Die Datenbereinigung, als Vorverarbeitungsphase im KDD-Prozess, stellt somit einen entscheidenden Faktor dar, um die Qualität der Daten und folglich die Ergebnisse des Minings zu verbessern. Insbesondere bei großen Realwelt-Datenbeständen nimmt die Datenbereinigung eine wichtige Rolle ein, da diese Daten eine stärkere Anfälligkeit für Dateninkonsistenzen aufweisen. Ein grundlegendes Problem in der Datenbereinigung ist die Erkennung und Eliminierung von Duplikaten in einer Objektmenge. Dieses Problem entsteht überwiegend in der Informationsintegration, d.h. bei der Zusammenführung mehrerer Datenbestände zu einer kohärenten Datenbank. Auf Grund der unterschiedlichen Repräsentation von Objekten in verschiedenen Datenbanken, fehlender konsistenter Identifizierer und der daraus resultierenden Ambiguität, ist die Erkennung von Duplikaten essentiell, um eine akkurate Analyse der

Daten durchführen zu können. Prinzipiell kann zwischen zwei verschiedenen Formen von Ambiguität unterschieden werden: Zum einen kann ein Objekt in verschiedenen Repräsentationen in der Datenmenge auftreten, und zum anderen kann die Repräsentation genau eines Objektes auf die Beschreibung mehrerer Objekte zutreffen.

Als Beispiel betrachten wir die Formen der Ambiguität bei Autorennamen wissenschaftlicher Publikationen. Die erste Form von Namensambiguität tritt auf, wenn ein Autor unter verschiedenen Namen aufgeführt wird. Zum Beispiel könnte der Autor „John A. Smith“ in unterschiedlichen Publikationen unter verschiedenen Namen erscheinen, wie zum Beispiel „John Smith“, „J. A. Smith“ oder auch in einer falsch geschriebenen Namensvariante „Jon Smith“. Die zweite Form von Namensambiguität besteht darin, dass mehrere Autoren unter demselben Namen publizieren. Zum Beispiel könnte sich der Name „J. Smith“ sowohl auf den Autor „John A. Smith“ als auch auf den Autor „John B. Smith“ beziehen. Die Anwendung von Analysetechniken, wie beispielsweise die Bestimmung des Prestiges von Autoren, kann auf Datenmengen, in denen diese Ambiguitäten nicht korrekt aufgelöst wurden zu fehlerhaften Resultaten führen.

Objektkonsolidierung

Diese Arbeit befasst sich mit der Objektkonsolidierung (engl.: object consolidation), einer wichtigen Aufgabe in der Datenbereinigung, die in einem engen Zusammenhang mit der Erkennung und Eliminierung von Duplikaten steht. Ziel der Objektkonsolidierung ist, alle Referenzen bzw. Duplikate einer Entität in einer Datenmenge korrekt zu gruppieren, d.h. es wird eine Partitionierung der Referenzen in Gruppen angestrebt, so dass einerseits alle Referenzen innerhalb einer Gruppe und andererseits keine zwei Referenzen aus zwei verschiedenen Gruppen dieselbe Entität repräsentieren. Wie wir gleich sehen werden kann die Objektkonsolidierung alternativ als Clustering Problem formuliert werden. Das Clustering von Daten ist eine klassische Aufgabe der Datenanalyse, in der die Daten in Gruppen ähnlicher Objekte partitioniert werden. Jede Gruppe oder auch jeder Cluster besteht aus Objekten, die zueinander ähnlich sind und unähnlich zu Objekten anderer Gruppen. Konventionelle Clustering Algorithmen verwenden dabei die Attributinformationen, um Objekte in Gruppen zusammenzufassen, unter der Annahme, dass zwei Objekte einer Gruppe angehören, falls ihre Attributwerte ähnlich sind. Der Schlüssel zum Erfolg eines Clustering Algorithmus hängt dabei von dem verwendeten Ähnlichkeitsmaß ab. Hinsichtlich der Objektkonsolidierung bezieht sich nun das Clustering Problem auf die Partitionierung der Referenzen einer Entität in denselben Cluster, wobei das Ähnlichkeitsmaß auf die Disambiguierung von Referenzen abzielt. Es werden also Duplikaten höhere Ähnlichkeitswerte zugewiesen als Nicht-Duplikaten. Da die Qualität des Clusterings zu einem wesentlichen Anteil von dem verwendeten Ähnlichkeitsmaß abhängig ist und dieses wiederum von der zur Verfügung stehenden Attributinformationen abhängt, kann es vorteilhaft sein, weitere Kontextinformationen, die eventuell in der Datenmenge implizit vorhanden sind, zu berücksichtigen. Einen solchen Kontext finden wir in Datenmengen vor, in der die Objekte (durch Relationen) in Beziehung zueinander stehen. Das in den Relationen zusätzliche vorhandene Wissen kann zusammen mit den Attributinformationen in einer verbesserten Genauigkeit

der Objektkonsolidierung resultieren. Während unser Ansatz der Objektkonsolidierung domänenunabhängig ist, werden wir zur Illustrierung die Konsolidierung mehrdeutiger Autorennamen in Autorennetzwerken wissenschaftlicher Publikationen betrachten. In Autorennetzwerken liegt mit der Kollaboration zwischen Autor und Koautor eine zusätzliche Kontextinformation in Form der Koautorenschaft vor. Eine zunächst nahe liegende Idee, diese Information in den Konsolidierungsprozess einzubeziehen, besteht darin, die Koautorenschaft als zusätzliches Attribut für einen Autor aufzunehmen. Wollen wir nun bestimmen, ob zwei Autorenreferenzen dieselbe Entität repräsentieren und angenommen den Fall, beide Referenzen besitzen dieselben Koautoren, sollten wir diese Information in dem verwendeten Ähnlichkeitsmaß positiv berücksichtigen. Ein direkter Vergleich der Koautoren greift jedoch zu kurz, da zwei Autorenreferenzen, die denselben Autor repräsentieren, keine gemeinsamen Koautoren haben müssen, jedoch über indirekte Relationen eng miteinander in Beziehung stehen können. Um den Grad der Beziehung zwischen Objekten im Netzwerk quantifizieren zu können, sind Verfahren zu entwickeln, die darauf ausgerichtet sind, nicht nur die direkten Beziehungen, sondern gerade auch die indirekten Beziehungsmuster zu berücksichtigen. Wir wollen in dieser Arbeit untersuchen, wie das in den Beziehungsstrukturen implizit vorhandene Wissen, genutzt werden kann, um eine Verbesserung im Ergebnis der Objektkonsolidierung zu erzielen. Unser Ansatz basiert dabei auf folgender Hypothese: Wenn zwei Referenzen sich auf dieselbe Entität beziehen, ist es wahrscheinlich, dass diese beiden Referenzen über weitere explizite sowie implizite Beziehungen miteinander verbunden sind. Der Fokus auf die relationalen Strukturen in Netzwerken erfordert eine Reihe von Methoden und analytischen Konzepten, die sich von den traditionellen Methoden der Datenanalyse unterscheiden.

Soziale Netzwerkanalyse

Die zurzeit umfassendsten Modelle zur Analyse von Netzwerken liefert eine wissenschaftliche Disziplin, die in den letzten Jahren unter dem Begriff *soziale Netzwerkanalyse* (SNA) verstärkt Interesse auf sich gezogen hat, unter anderem im Bereich Data Mining, wo dieses Thema erst kürzlich aufgetreten ist. Dabei richtet die *soziale Netzwerkanalyse* den analytischen Blick auf die Beziehungen zwischen Entitäten, in der Terminologie der SNA auch Akteure genannt, und untersucht Strukturen und Muster solcher Beziehungen. In dieser Arbeit stellen wir zwei typische Probleme der Sozialen Netzwerkanalyse vor, die für die Entwicklung unseres Konsolidierungsalgorithmus relevant sind: Bestimmung der Akteurszentralität und Identifizierung von Community Struktur.

Die Zentralität eines Akteurs beschreibt dessen strategische Position im Netzwerk, seine Wichtigkeit und seinen Grad an Einfluss im Netzwerk. Der Grundansatz der Netzwerkanalyse ist dabei, nicht nur die direkten Beziehungen der Akteure, sondern gerade auch die indirekten Beziehungsmuster zu berücksichtigen, um die Einbettung der Akteure zu evaluieren. Ein Beispiel für Zentralität findet sich in der derzeit populären Suchmaschine Google wieder, dessen Erfolg zu einem gewissen Anteil auf den verwendeten Algorithmus zur Bestimmung der Relevanz von Webseiten zurückzuführen ist. Der nach einem seiner Erfinder, Lawrence Page, benannte PageRank Algorithmus [PBMW98]

verwertet dabei Informationen, die implizit in der Hyperlink-Struktur des Webs vorhanden ist, um die Relevanz von Webseiten zu bestimmen. Durch die Kombination eines Prestigemaßes mit traditionellen Methoden des Informations-Retrieval ist es möglich, die *Precision*[1] von Suchresultaten deutlich zu steigern, indem unter allen Treffern einer Anfrage diejenigen Dokumente mit hoher Autorität möglichst weit vorne im Ranking platziert werden. Im Kontext unserer Arbeit steht dabei weniger für die absolute Zentralität im Mittelpunkt, sondern wir richten unser Augenmerk auf die relative Zentralität, d.h. wir interessieren uns für die relative Wichtigkeit von Akteuren bezüglich eines speziellen Akteurs.

Das zweite für diese Arbeit relevante Problem ist die Identifizierung von Community Struktur in einem sozialen Netzwerk. Community oder auch Modulare Struktur ist eine in vielen Netzwerken vorzufindende Eigenschaft, die sich auf eine natürliche Aufteilung von Netzwerkknoten in Gruppen bezieht, innerhalb derer die Verbindungen dicht sind, aber die Gruppen untereinander nur spärlich verbunden sind. Die Identifizierung und Analyse von Community Struktur in komplexen Netzwerken ist eine wichtige Aufgabe in Bereichen, wie der Soziologie, Informatik, und, etwas konkreter, in der Entwicklung effizienter Suchmaschinen im Web. Um an das Beispiel der Relevanzbewertung von Webseiten anzuknüpfen, betrachten wir den Nutzen von Web Communities zur Visualisierung von Suchergebnissen. Häufig besteht bei Suchanfragen das Problem, dass der Kontext nicht eindeutig aus einem oder mehreren Suchbegriffen hervorgeht. Dadurch beziehen sich viele der zurückgelieferten Seiten auf eine andere Bedeutung, was wiederum die *Precision* der Ergebnisse reduziert. Um dieses Problem teilweise zu entgehen, könnten die Dokumente bei der Visualisierung des Suchergebnisses ihren Communities bzw. Kategorien zugeordnet werden, so dass der Benutzer diese Ambiguität auflösen und die von ihm angedachte Kategorie auswählen kann. Dieses Beispiel stellt einen direkten Bezug zu unserer Arbeit her, bei der wir die Identifizierung von Communities zur Disambiguierung von Autorennamen einsetzen wollen.

Link Mining: Mining in sozialen Netzwerken

Die beiden oben betrachteten Probleme der SNA, Zentralität und Identifizierung von Community Struktur, können in gewisser Weise als eine Art Wissensextraktion in einer vernetzten Umgebung verstanden werden. Trotz dieser Gemeinsamkeit mit Data Mining, dass eine bestimmte Form von Wissen aus den Daten extrahiert wird, unterscheiden sich die Techniken der SNA und die des Data Minings in einem wesentlichen Punkt. Während der Fokus der *sozialen Netzwerkanalyse* ausschließlich auf die relationalen Strukturen zwischen Entitäten gerichtet ist, basieren die traditionellen Data Mining Techniken auf der Annahme, dass die unterliegende Datenmenge aus unabhängigen Instanzen zusammengesetzt ist. Das in vernetzten Strukturen implizit vorhandene Wissen kann jedoch nützlich sein, um klassisches Verfahren des Data Minings, wie beispielsweise Klassifizierung und Clustering, zu verbessern.

[1] *Precision* ist ein Maß zur Bestimmung der Güte eines Suchergebnisses im Information Retrieval und bezieht sich auf die Genauigkeit eines Suchergebnisses. Sie ist definiert als der Anteil der gefundenen relevanten Dokumente von allen bei einer Suche gefundenen Dokumenten.

Mit Link Mining ist neuerdings ein Forschungsgebiet in Erscheinung getreten, das in der Schnittmenge der *sozialen Netzwerkanalyse*, Web Mining und Relationalen Lernen liegt. Link Mining ist im weitesten Sinne eine Instanz von multi-relationalem Data Mining; mit dem Begriff Link Mining soll jedoch die Betonung auf die Relationen gelegt werden. In der Arbeit [Get03] untersucht die Autorin, Lise Getoor, welche klassischen Data Mining Aufgaben sich auf vernetzte Domänen übertragen lassen und nennt neue Aufgaben und Probleme, die mit der Einführung von Links einhergehen. Zu den wichtigen Link Mining Aufgaben, die aus den klassischen Verfahren des Data Minings hervorgehen, zählt unter anderem die Link-basierte Klassifikation und die Link-basierte Cluster Analyse.

Obwohl einige Aspekte und Begrifflichkeiten aus Link Mining einen wesentlichen Bestandteil dieser Arbeit betreffen, sei an dieser Stelle darauf hingewiesen, dass das Hauptaugenmerk vielmehr auf den Konzepten und Techniken der *sozialen Netzwerkanalyse* liegt. Aus diesem Grund wird auf das Thema Link Mining nicht weiter eingegangen und nur in Einzelfällen darauf Bezug genommen.

1.2. Ziel dieser Arbeit

Die Idee für unseren Konsolidierungsalgorithmus geht auf die Arbeit [CKM05] von Chen et al. zurück. Dazu betrachten wir die zugrunde liegende Datenmenge als Graphen, in welchem die Knoten den aufzulösenden Referenzen entsprechen und die Kanten mit den Beziehungen zwischen Referenzen korrespondieren. Wir bezeichnen diesen Graphen als Referenzgraphen. Ziel ist es, einen bereinigten Entitätengraphen zu rekonstruieren, bei dem alle Referenzen, die dieselbe Entität referieren, zu einem Knoten konsolidiert sind. Im Falle eines Autorennetzwerkes ergibt sich der Referenzgraph aus den Autorenreferenzen und der Koautorenschaft, wobei erstere den Knoten und letztere den Kanten in diesem Graphen entspricht. Zur Konsolidierung der Autorenreferenzen setzen wir dabei Techniken aus den Bereichen Maschinelles Lernen und *soziale Netwerkanalyse* ein, wobei im Hinblick auf die Koautorenschaft insbesondere die relationalen Merkmale der Datenmenge im Mittelpunkt der Analyse stehen. Der Ablauf des Konsolidierungsalgorithmus kann kurz wie folgt beschrieben werden: Zunächst verwendet das Verfahren die Attributinformationen, um die Referenzen zu identifizieren, die auf Grund ihrer Ähnlichkeit potentiell dieselbe Entität repräsentieren. Anschließend werden die einander ähnlichen Referenzen paarweise analysiert, um die relative Zentralität zwischen diesen Referenzen im Graphen zu quantifizieren. Das Verfahren identifiziert dann Communities zur weiteren Disambiguierung der Autorenreferenzen.

1.3. Gliederung

Die vorliegende Arbeit unterteilt sich folgendermaßen: Kapitel 2 dient zunächst einmal dazu den Leser in die Theorie der *sozialen Netzwerkanalyse* heranzuführen, wobei insbesondere mit der Zentralität und der Identifizierung von Community Struktur zwei

wesentliche Aspekte dieser Theorie und zugleich zentraler Bestandteil dieser Arbeit im Detail dargestellt werden sollen. Der auf die Arbeit von Chen et al. zurückzuführende Konsolidierungsalgorithmus wird dann in Kapitel 3 – dem Hauptteil unserer Arbeit – vorgestellt. Zu Beginn dieses Kapitels wird auf die allgemeine Problematik von Duplikaten in digitalen Bibliotheken eingegangen und im Anschluß daran das Problem der Objektkonsolidierung formal eingeführt. Der Rest des Hauptteils widmet sich der Beschreibung der von Chen et al. vorgeschlagenen Verfahren mit den unseren, der *sozialen Netzwerkanalyse* entspringenden Verfahren. Das Kapitel schließt mit der Einführung von Metriken ab, die wir im Experimentteil unserer Arbeit – das in Kapitel 4 beschrieben wird – dazu verwenden wollen, die in dem Konsolidierungsframework eingesetzten Verfahren zu evaluieren. Zum Schluß werden in Kapitel 5 die gewonnenen Erkenntnisse noch einmal zusammengefasst und einige Anreize für zukünftige Arbeiten geliefert, die sich eventuell mit dieser Thematik auseinandersetzen.

2. Aspekte der sozialen Netzwerkanalyse

2.1. Soziale Netzwerke

Die Theorie sozialer Netzwerke umfasst eine Reihe von Konzepten und Begrifflichkeiten, die im Folgenden kurz dargestellt werden. Für eine detaillierte Einführung in diese Theorie sei der interessierte Leser auf [WF94] verwiesen.

Wie bereits in der Einleitung angeführt wurde, befasst sich die *soziale Netzwerkanalyse* im Kern mit den Beziehungen zwischen sozialen Entitäten und den Implikationen dieser Beziehungen. Die sozialen Entitäten werden als Akteure bezeichnet. Akteure sind diskrete Individuen oder kollektive Einheiten, die untereinander durch soziale Verbindungen, z.B. die Kollaboration zwischen Autor und Koautor, verknüpft sind. Die elementare Eigenschaft einer Verbindung ist, dass eine Beziehung zwischen einem Paar von Akteuren etabliert wird. Die Menge aller Akteure, die demselben Typ angehören, wird unter dem Begriff der Gruppe zusammengefasst. Eine Gruppe besteht somit aus einer endlichen Menge von Akteuren, die aus konzeptuellen Gründen zusammengehören und die Gegenstand der Analyse ist. Die Menge aller Verbindungen eines bestimmten Typs zwischen Akteuren einer Gruppe definiert eine Relation. So beschreibt beispielsweise die Menge aller Verbindungen zwischen Autoren und Koautoren die Relation der Koautorenschaft. Nachdem nun die elementaren Begriffe Akteur, Gruppe und Relation erläutert sind, können wir eine explizitere Definition eines sozialen Netzwerkes angeben: Ein soziales Netzwerk besteht aus einer endlichen Menge von Akteuren und einer auf diese definierte Relation.

In Anlehnung an die klassische Graphentheorie wird ein soziales Netzwerk als Graph modelliert, der aus einer Menge von Knoten und einer Menge von Kanten besteht. In diesem Modell repräsentieren die Knoten die Akteure und die Kanten entsprechen den Verbindungen zwischen den Akteuren. Die Graphentheorie hat sich diesbezüglich als nützliches Instrument erwiesen, da zum einen soziale strukturelle Eigenschaften durch ein einheitliches Vokabular beschrieben werden können und zum anderen bietet sie die mathematische Operationen, um viele dieser Eigenschaften zu quantifizieren.

2.2. Notation

An dieser Stelle führen wir noch einige Grundbegriffe und Definitionen aus der Graphentheorie ein, die in dieser Arbeit verwendet werden. Ein ungerichteter Graph $G = (V, E)$

besteht aus einer endlichen Menge V von Knoten und einer endlichen Menge $E \subseteq V^{(2)}$ von Kanten. Dabei bezeichnet $V^{(2)}$ die Menge der ungeordneten Paare von (nicht notwendigerweise verschiedenen) Elementen aus V. Ist $e = (u, v)$ für eine Kante $e \in E$ dann nennen wir u, v Endknoten von e. Wir sagen, u und v sind inzident zu e beziehungsweise liegen auf e, die Kante e verbindet u und v, und die Knoten u und v sind Nachbarn beziehungsweise adjazent. Neben ungerichteten Graphen werden auch gerichtete Graphen betrachtet. Ein gerichteter Graph oder Digraph $D = (V, A)$ besteht aus einer endlichen Menge von Knoten und einer endlichen Menge $A \subseteq V \times V$ von Bögen, wobei jeder Bogen $a \in A$ ein geordnetes Paar von Knoten ist, also $a = (u, v)$ für $u, v \in V$. Die Adjazenzmatrix $A = (a_{ij})$ eines Graphen $G = (V, E)$ ist eine $n \times n$ Matrix mit Elementen:

$$a_{ij} = \begin{cases} 1 & \text{falls } i \text{ und } j \text{ verbunden sind,} \\ 0 & \text{andernfalls.} \end{cases} \tag{2.1}$$

Die obige Definition gilt sowohl für ungerichtete als auch für gerichtete Graphen. Im Falle eines ungerichteten Graphen ist die Adjazenzmatrix symmetrisch, d.h. $a_{ij} = a_{ji} \quad \forall i, j \in \{1, ..., n\}$. Im Falle von gewichteten Graphen mit einer auf den Kanten bzw. Bogen definierten Gewichtsfunktion $w : E \to \mathbb{R}_0^+$ repräsentiert a_{ij} das Gewicht w_{ij} der Kanten zwischen Knoten i und j. In einem ungerichteten Graphen ist der Grad $d(i) = \sum_j a_{ij}$ eines Knotens i die Anzahl seiner adjazenten Nachbarn. In gerichteten Graphen wird zusätzlich zwischen Eingangsgrad und Ausgangsgrad eines Knotens unterschieden: Der Eingangsgrad eines Knotens i ist dann definiert als $d_{in}(i) = \sum_j a_{ji}$ und der Ausgangsgrad als $d_{out}(i) = \sum_j a_{ij}$.

2.3. Zentralität und Prestige

Ein wichtiges Strukturmerkmal der Akteure in einem Netzwerk ist der Grad ihrer Prominenz. Dieses Merkmal stellt ein netzwerkanalytisches Konzept dar, das den Grad der Eingebundenheit eines Akteurs in einer Sozialstruktur reflektiert und somit so etwas wie dessen Einfluss im Netzwerk beschreibt. Die Gleichsetzung der Prominenz eines Akteurs mit dem Grad der Eingebundenheit des Akteurs im Netzwerk geht auf Knoke und Burt (1983) zurück. Hubbel (1965) und Friedkin (1991) fügten dem hinzu, dass nicht nur die direkten Beziehungen der Akteure, sondern gerade auch die indirekten Beziehungsmuster und Pfade über „Mittelsleute“ zu berücksichtigen sind, um die Prominenz der Akteure zu evaluieren. Um zwischen ungerichteten und gerichteten Relationen differenzieren zu können, haben Knoke und Burt zusätzlich zwei Ausprägungen von Prominenz eingeführt: Zentralität und Prestige. Bei der Betrachtung von ungerichteten Beziehungen wird die Zentralität herangezogen, in der die Rolle des Akteurs als Empfänger oder als Initiator der Beziehung irrelevant ist. Die Prominenz eines Akteurs bezieht sich ausschließlich darauf, in welchem Maße der Akteur in den Beziehungen involviert ist. Dementsprechend definieren wir einen zentralen Akteur, als einen, der an vielen Interaktionen teilnimmt

und somit stark im Netzwerk eingebunden ist. Der Begriff Prestige stellt eine Verfeinerung des Zentralitätskonzepts dar, in welcher zwischen initiierenden und empfangenden Verbindungen unterschieden wird. Wir definieren einen prestigeträchtigen Akteur als einen Empfänger vieler gerichteter Verbindungen, d.h. das Prestige eines Akteurs wächst mit der Zunahme seines Eingangsgrades, jedoch nicht notwendigerweise mit dem seines Ausgangsgrades. Somit ist das Konzept hinter Prestige nur im Zusammenhang mit gerichteten Relationen anwendbar, da sich der Eingangsgrad eines Akteurs nur in diesen vom Ausgangsgrad unterscheiden lässt.

2.3.1. Zentralität

Degree Zentralität

Das einfachste Maß zur Charakterisierung der Zentralität eines Akteurs ist die Anzahl seiner direkten Nachbarn, das aus graphentheoretischer Sicht dem Grad des Knotens entspricht. Zentral nach diesem Konzept ist der Akteur, der am aktivsten ist, in dem Sinne, dass dieser in vielen direkten Beziehungen mit anderen Akteuren involviert ist. Es sei anzumerken, dass sich die *Degree* Zentralität nur auf die direkten bzw. adjazenten Verbindungen eines Akteurs beschränkt. Wie bereits jedoch weiter oben angemerkt wurde sollte sich die Prominenz eines Akteurs aber nicht nur aus seinen direkten Verbindungen erschließen, sondern auch und vielmehr aus seinen indirekten Verbindungen.

$$C_D(v) = d(v) \tag{2.2}$$

Betweenness Zentralität

Die Interaktionen zwischen zwei nichtadjazenten Akteuren kann in gewissem Sinne abhängig von den anderen Akteuren in der Menge sein, wobei insbesondere den Akteuren, die auf den Pfaden zwischen den beiden liegen, eine besondere Rolle zuteil wird. Diese Akteure haben potentiell mehr Kontrolle über die Interaktionen zwischen den nichtadjazenten Akteuren. Zentral nach dieser Idee ist der Akteur, der zwischen (engl.: *between*) vielen Akteurspaaren im Netzwerk auf deren kürzesten Verbindungen positioniert ist. Daher wird dieses Maß als *Betweenness* Zentralität bezeichnet. Ein zentraler Akteur nach *Betweenness* verbindet also viele Akteure im Netzwerk und kann deshalb viele Aktivitäten im Netzwerk kontrollieren. Obwohl die strategische Bedeutung der Positionierung von Akteuren auf den geodesischen Pfaden früh von Bavelas (1948) erkannt wurde, gelang es zunächst nicht diese Art von Zentralität zu quantifizieren. Es hat mehr als zwanzig Jahre gedauert bis eine Berechnung dieser Zentralität von Anthonisse (1971) und Freeman (1977) entwickelt wurde. Die Grundidee zur Berechnung dieses Maßes besteht zunächst darin, alle geodesischen Pfade zwischen den Akteuren u und w zu bestimmen und dann auszuwerten, wie häufig ein Akteur v in diesen geodesischen Pfaden vorkommt. Diese Prozedur ist für alle verschiedenen Indizes v, u, w durchzuführen. Der Index zur *Betweenness* Zentralität eines Knotens v berechnet sich dann folgendermaßen:

$$C_B(v) = \sum_{u \neq w \neq v} g_{uw}(v)/g_{uw} \tag{2.3}$$

wobei $g_{uw}(v)$ die Anzahl bezeichnet, wie häufig der Knoten v in den geodesischen Pfaden zwischen den Knotenpaaren u und w vorkommt. Wenn mehr als ein Pfad zwischen den Knoten u und w existiert, dann beträgt die Wahrscheinlichkeit, dass ein bestimmter geodesischer Pfad ausgewählt wird $1/g_{uw}$, wobei g_{uw} die Anzahl der geodesischen Pfade ist, die zwei Akteure miteinander verbinden. Demnach wird die Wahrscheinlichkeit, dass ein bestimmter Knoten v in der Kommunikation zwischen den beiden Knoten involviert ist mit $g_{uw}(v)/g_{uw}$ abgeschätzt. Die Akteurs *Betweenness* berechnet sich dann – wie aus Gleichung (2.3) ersichtlich – aus der Summe dieser geschätzten Wahrscheinlichkeiten aller Akteurspaare, die nicht den Akteur v enthalten. Dabei wird das Minimum des *Betweenness* Index bei 0 angenommen, wenn der Akteur auf keiner der geodesischen Pfade liegt. Umgekehrt, wenn der Akteur auf allen geodesischen Pfade liegt, wird das Maximum bei $(n-1)(n-2)/2$ angenommen, wobei mit letzterem Ausdruck die Anzahl der Akteurspaare mit Ausnahme des Akteurs v beschrieben wird. Um diesen Index von der Knotenanzahl unabhängig zu machen, hat Freeman eine normalisierte Variante von Gleichung (2.3) eingeführt:

$$C'_B(v) = \frac{C_B(v)}{(n-1)(n-2)/2} \tag{2.4}$$

Durch diese Normalisierung liegt der Wertebereich dieses Index im Intervall $[0, 1]$, wodurch die Akteure miteinander verglichen werden können, sowie sich auch unterschiedliche Netzwerke bezüglich dieses Maßes besser vergleichen lassen.

Closeness Zentralität

Die *Closeness* Zentralität basiert auf dem Konzept der Kürzesten Wege, bei der ein Akteur zentral ist, falls dieser über viele kurze Verbindungen zu allen anderen Akteuren im Netzwerk verfügt. Die Idee hinter dieser Zentralität stützt sich auf die Kommunikationsfähigkeit, d.h. ein zentraler Akteur sollte schnell mit anderen Akteuren kommunizieren können, d.h. die Zentralität steht in negativer Korrelation zur Distanz. Die Kommunikationsfähigkeit eines Akteurs ist umso besser, je näher dieser zu den anderen Akteuren steht; die *Geodesischen* oder Kürzeste Wege, die zentrale Akteure mit anderen Akteuren verbindet müssen also so kurz wie möglich sein.

Diese Art von Zentralität, die die Nähe zu anderen Akteuren im Netzwerk reflektiert, geht unter anderem auf die vorgeschlagenen Zentralitätsmaße von Bavelas (1950), Sabidussi (1966) und Freeman (1979) zurück. Das einfachste Maß nach Freeman ist die Zentralität von Sabidussi, bei der die *Closeness* als Funktion der *geodesischen* Distanzen gemessen wird. Nach Sabidussi berechnet sich der Index zur *Closeness* Zentralität folgendermaßen:

$$C_C(v) = \frac{1}{\sum_u dist(v,u)} \tag{2.5}$$

wobei $dist(v,u)$ die kürzeste Entfernung von v nach u bezeichnet. Dieser Index ist in dieser einfachen Variante abhängig von der Anzahl der Knoten und folglich für einen Vergleich zwischen Netzwerken unterschiedlicher Größe nicht geeignet. Beauchamp (1965) hat diesen zu einem standardisierten Index erweitert, um eine normalisierten Wert zwischen 0 und 1 zu erhalten:

$$C'_C(v) = (n-1)C_C(v) \tag{2.6}$$

Zum Schluss sei noch anzumerken, dass sich mit diesem Zentralitäts-Index im Gegensatz zur *Degree* Zentralität, die Eingebundenheit eines Akteurs nicht nur über die Zahl seiner direkten Verbindungen erschließt, sondern auch über seine indirekten Verbindungen.

2.3.2. Prestige

Die bisherigen Methoden zur Quantifizierung der Prominenz von Akteuren bezogen sich auf ungerichtete Graphen, in denen nicht zwischen empfangenden und initiierenden Verbindungen unterschieden wurde. Mit gerichteten Beziehungen rücken nun die empfangenden Beziehungen in den Blickpunkt der Betrachtung, durch die das Prestige eines Akteurs – als Verfeinerung des Zentralitätskonzeptes – zum Ausdruck gebracht wird. Ein prestigeträchtiger Akteur genießt im Allgemeinen eine starke Popularität, wenn er ein Empfänger vieler gerichteter Verbindungen ist. Bei den einfachen Prestigemaßen, die – analog zu den Zentralitätsindizes – auf dem Eingangsgrad oder den Distanzen zwischen Akteuren basieren, wird die Prominenz der individuelle Akteure, die die Beziehung initiieren, per se nicht mit einbezogen. Die Kombination der *Degree* Zentralität mit dem Prestigewert der involvierten benachbarten Akteure, führt zu einer neuen Definition von Prestige, die auf nachfolgender Idee beruht: Rein Intuitiv sind Akteure deren unmittelbare Nachbarschaft aus prestigeträchtigen Akteuren besteht, ebenso als prominent anzusehen. Umgekehrt, falls die Nachbarschaft eines Akteurs nur periphere oder marginal wichtige Akteure enthält, so sollte das Prestige dieses Akteurs ebenso niedrig sein. Diese neue Idee, dass das Prestige der Nachbarschaft für die Bestimmung des Prestiges eines Akteurs relevant ist, wurde erstmals von Seeley (1949) realisiert, der auch die rekursive Natur von Prestige in einem sozialen Netzwerk erkannte:

...we are involved in an infinite regress: [an actor's status] is a function of the status of those who choose him; and their [status] is a function of those who choose them, and so ad infinitum.

In der Literatur wird dieses Maß als Status bzw. Rank bezeichnet, bei der ein Akteur als prestigeträchtig bezüglich seines Status gilt, wenn dieser relativ zu der Akteurmenge

einen hohen Statuswert aufweist. Der Rest dieses Abschnitts befasst sich mit der Herleitung einer Lösung dieser rekursiven Definition von Prestige. Diese führt direkt zu dem aus der Linearen Algebra bekannten Problem der Eigenvektor Berechnung, einem Vorläufer der Eigenvektor Methoden, auf die die Ranking-Algorithmen PageRank und HITS zurückzuführen sind.

Die einfachste Möglichkeit, eine Lösung für diesen „unendlichen Regress“ zu präsentieren, besteht zunächst einmal darin, mit jedem Akteur v einen positiven reellen Wert p_v zu assoziieren. Nach der Theorie, die sich hinter Prestige als Status verbirgt, ist der Status eines Akteurs eine lineare Funktion der Akteure, die zu diesem eine Beziehung initiieren. Wir erhalten somit folgende Linearkombination, die den Status eines Akteurs v misst:

$$p_v = \sum_{(u,v)\in E} p_u \quad \forall v \in V \tag{2.7}$$

Mathematisch betrachtet liegt in (2.7) ein lineares Gleichungssystem mit $n = |V|$ Gleichungen und n Unbekannten vor. Falls wir die Adjazenzmatrix A des Graphen verwenden, kann dieses Gleichungssystem in Matrixnotation umformuliert werden:

$$p = A^T p \tag{2.8}$$

aufgrund folgender Beziehung:

$$p_v = \sum_{(u,v)\in E} p_u = \sum_u A_{uv} p_u \tag{2.9}$$

Der Vektor p aus Gleichungssystem (2.8) ist der gesuchte Statusvektor, dessen Elemente mit den Statuswerten der Akteure korrespondieren. Zur Bestimmung des Statusvektors p bedarf es einiger mathematischer Grundlagen der Linearen Algebra, die wir im Folgenden kurz darstellen wollen.

Definition 2.1 (Eigenwert, Eigenvektor): Eine Zahl $\lambda \in \mathbb{R}$ heißt Eigenwert von A, wenn es einen Vektor $0 \neq v$ gibt, so dass

$$Av = \lambda v \tag{2.10}$$

gilt und jeder Vektor $0 \neq v$, der diese Gleichung erfüllt, heißt Eigenvektor von A zum Eigenwert λ.

Definition 2.2 (Stochastische Matrix): Eine $n \times n$ Matrix $A = (a_{ij})$ wird stochastische Matrix genannt, falls $a_{ij} \geq 0 \quad \forall i, j = 1, .., n$ und $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i = 1, .., n$ gilt.

Die Gleichung (2.8) ist somit identisch mit dem Eigensystem aus (2.10), in welcher der Statusvektor p ein Eigenvektor der Matrix A^T zum Eigenwert $\lambda = 1$ ist:

$$A^T p = 1p \tag{2.11}$$

Eine Lösung für das System (2.11) besteht somit darin einen solchen Eigenwert für die Matrix A^T zu erzwingen, indem die Zeilen der Matrix A normalisiert werden, so dass sich diese zu 1 aufsummieren. Die Matrix muss also die Stochastizitäts-Bedingung aus Definition (2.2) erfüllen. Diese Bedingung an die Adjazenzmatrix stellt keinen Abbruch an die ursprüngliche Statusbetrachtung dar, sondern hat eine naturgemäße Interpretation: Bei einem Akteur, der viele Beziehungen zu seinen Nachbarn initiiert, sollte der Status auf die einzelnen Verbindungen aufgeteilt werden, so dass diese nicht so stark gewichtet werden. Um eine Lösung für das Gleichungssystem (2.11) zu garantieren sind neben der Stochastizitäts-Bedingung der Matrix A noch weitere Bedingungen an die Matrix zu stellen; andernfalls hat das Gleichungssystem keine endliche Lösung, wie dies erstmal von Katz (1953) festgestellt wurde. In der Tat haben sich viele Autoren mit diesem Problem beschäftigt und die Lösungen können dahingehend kategorisiert werden, welche Bedingungen an die Matrix des Eigensystems gestellt werden. Wir wollen an dieser Stelle nicht weiter auf die Problematik eingehen, sondern uns erst im nachfolgenden Abschnitt bei der Statusbetrachtung im Web mit genau diesem Problem im Detail beschäftigen.

2.3.3. Status im Web: PageRank und HITS

Das Ranking ist eine integrale Komponente eines jeden Information Retrieval Systems, um die als Antwort auf eine Suchanfrage gefundenen Dokumente hinsichtlich ihrer Relevanz zu sortieren. Rankingverfahren spielen insbesondere bei Web-Suchmaschinen eine kritische Rolle, da diese in der Regel wegen der enormen Menge an indizierten Dokumenten sehr große Treffermengen zurückgeben. Auf Grund der Subjektivität, die mit dem Begriff der Relevanz einhergeht, sind jedoch nicht alle zurückgegebenen Dokumente für den Benutzer interessant. Da der Benutzer jedoch nicht die Zeit und Geduld aufbringen kann, um alle Treffer bezüglich deren subjektiven Relevanz auszuwerten, sind neben der Relevanz noch andere Faktoren für das Ranking zu berücksichtigen. Kleinberg [Kle99], Page und Brin [PBMW98] haben in ihren Arbeiten erkannt, dass Webbenutzer nicht nur an relevanten Seiten, sondern auch gleichzeitig an autoritativen Seiten interessiert sind, d.h. zuverlässigen Quellen mit korrekten Informationen und einer starken Präsenz im Web. Diesbezüglich besteht die Aufgabe eines Ranking-Verfahrens darin, aus einer großen Sammlung von relevanten Seiten eine kleinere Untermenge von autoritativen Seiten zu identifizieren, um diese möglichst weit vorne im Ranking zu platzieren. Dazu ist es jedoch zunächst notwendig den Begriff einer autoritativen Seite bzw. die Autorität einer Seite im Kontext einer Suchanfrage näher zu spezifizieren. Wie wir jedoch gleich sehen werden ist der Begriff der Autorität eine Variante des Prestiges aus dem vorherigen Abschnitt.

Zu diesem Zweck wollen wir den gerichteten Hyperlink-Graphen $D = (V, A)$ des *World Wide Webs* (WWW) als soziales Netzwerk auffassen, wobei die Webseiten den Knoten und die Hyperlinks den gerichteten Kanten entsprechen. Da die komplexe Hyperlink-Struktur des Webs keiner zentralen Koordination oder redaktionellen Kontrolle unterliegt, sondern sich vielmehr organisch entwickelt, ist eine globale Organisation nicht vorhanden. Dennoch enthält die Hyperlink-Struktur reichhaltige Informationen, die für diverse Aufgaben des Information Retrieval genutzt werden können, vorausgesetzt natürlich, dass wir über die notwendigen Techniken zur Extraktion dieses Wissens verfügen. Chakrabarti et al. haben das Potential der Hyperlinks erkannt, um diese als Zusatzinformation zur Klassifikation von Webseiten zu verwenden [Cha02]. Die bedeutendste Errungenschaft jedoch gelang durch die Erkenntnis, dass Hyperlinks eine beträchtliche Menge an latenten Beurteilungen enthalten und dass mit genau dieser Form von Beurteilungen Aussagen über die Autorität von Webdokumenten getroffen werden kann. Rein intuitiv können wir uns den Hyperlink-Graphen als Netzwerk von Empfehlungen vorstellen, in der stark verlinkte Seiten von höherem Prestige bzw. höherer Autorität sind als Dokumente, die wenig oder gar nicht verlinkt sind. Ein Link wird also – wie bei Zitationen wissenschaftlicher Publikationen auch – als Empfehlung für das Dokument angesehen, auf das verwiesen wird. Um nun diese Idee für das Ranking von Suchergebnissen einer Suchmaschine aufzugreifen, muss eine Ranking-Funktion diese latenten Informationen extrahieren und ein Ranking produzieren, das die relative Autorität von Webseiten reflektiert. Ein erstes simples Ranking-Verfahren könnte darin bestehen, die von der Suchmaschine zurückgegebenen Seiten hinsichtlich deren Eingangsgrades – d.h. der Anzahl der auf diese verweisenden Seiten – zu sortieren. Aus mehreren Gründen erweist sich jedoch ein lokales Maß, wie der Eingangsgrad, als kein guter Indikator für die Autorität einer Seite. Unter anderem besteht das Problem, dass dadurch universell populäre Seiten wie beispielsweise die Homepage von *Microsoft*, in ihrer Autorität zu hoch eingestuft werden.

Mit den Arbeiten von Kleinberg und Brin und Page wurden die Grundlagen für das Link-basierte Ranking geschaffen. Beide Arbeiten gehen dabei von obiger Annahme aus, dass Hyperlinks eine beträchtliche Menge an verborgenen Beurteilungen enthalten. Im *PageRank* Algorithmus von Brin und Page wird dazu jeder Webseite unabhängig von einer gestellten Suchanfrage ein Prestigemaß zugeordnet. Im Gegensatz dazu wird im Verfahren von Kleinberg, dem HITS-Algorithmus (*Hypertext Induced Topic Search*), die Suchanfrage mit einbezogen, um zunächst einen Untergraphen des Webs zu identifizieren. In diesem fokussierten Untergraphen werden dann zwei Arten von Knoten identifiziert, *Hubs* und *Autoritäten*. Obwohl zwischen diesen Algorithmen technische Unterschiede bestehen, basieren beide auf rekursiven Strukturen. Das Prestige eines Knotens hängt vom Prestige der anderen Knoten ab, und ein guter *Hub* hängt davon ab, wie gut die von ihm zitierten Nachbarknoten als *Autoritäten* agieren und umgekehrt.

PageRank

PageRank wurde in [PBMW98] eingeführt und ist das von der Internetsuchmaschine *Google* eingesetzte Verfahren zur Bewertung von Webseiten mit dem Ziel der Relevanz-

beurteilung. Die Bewertung der Webseiten basiert auf dem *Random Surfer Modell*, einem stochastischen Prozess zur Simulation eines Websurfers im Hyperlink-Graphen. Die Idee dabei ist, einen unendlich langen, zufälligen Pfad eines Websurfers im Hyperlink-Graphen zu betrachten und die Wahrscheinlichkeit für den Besuch einer Seite als wichtigen Indikator für das Ranking herzunehmen. Ausgehend von diesem Modell ergeben sich mehrere Faktoren, die Einfluss auf die Autoritätsbewertung haben. Je mehr eingehende Links eine Seite hat, desto höher ihre Autorität. Darüber hinaus ist die Autorität einer Seite proportional zur Summe der Autoritätsbewertungen ihrer Vorgängerseiten und umgekehrt proportional zum Ausgangsgrad ihrer Nachbarn. Letzteres ist eine unmittelbare Konsequenz des *Random Surfer Modells*, da die Wahrscheinlichkeit, welchen Link der „Zufällige Surfer“ auswählt, um auf die nächste Seite zu gelangen, allein von der Anzahl der ausgehenden Links der Seite abhängt. Bei Vorgängern mit vielen ausgehenden Kanten wird somit das Autoritätsmaß nur anteilsmäßig auf die Nachbarseiten umgelegt. Wir wollen zunächst damit beginnen diese intuitive Beschreibung des *Random Surfers* mit einem einfachen Modell formal zu erfassen:

$$r_j = \sum_{i \to j} \frac{r_i}{d_{out}(i)} \tag{2.12}$$

wobei r_j den *PageRank* der Seite j bezeichnet. Mit dieser rekursiven Definition erhalten wir im Prinzip die Eigenvektorzentralität zur Bestimmung des Prestiges eines Akteurs in einem *Sozialen Netzwerk*, die wir im letzten Abschnitt eingeführt haben. Auf die gleiche Weise wie bei der Behandlung der Eigenvektorzentralität lässt sich die Gleichung äquivalent in Matrixschreibweise notieren. Diese erhalten wir wie folgt: Sei dazu $A = (a_{ij})$ die Adjazenzmatrix des Hyperlink-Graphen wie in 2.1 definiert. Des Weiteren definieren wir eine Übergangsmatrix $P = (p_{ij})_{i,j \in V}$ folgendermaßen:

$$p_{ij} = \begin{cases} a_{ij}/d_{out}(i) & \text{falls } d_{out}(i) > 0, \\ 0 & \text{andernfalls.} \end{cases} \tag{2.13}$$

Falls wir mit $r = (r_i)$ den Vektor bezeichnen, dessen Elemente mit dem PageRank der Knoten bzw. Webseiten des Hyperlink-Graphen korrespondieren, dann erhalten wir folgende Matrixnotation zur Berechnung des *PageRanks*.

$$r = P^T r \tag{2.14}$$

Mit dieser naiven Definition des *PageRanks* sind jedoch eine Reihe von Problemen verbunden, die wir bereits im Zusammenhang mit der Eigenvektorzentralität angedeutet haben und auf die wir nun im Detail eingehen werden. Zu diesem Zweck werden wir *PageRank* mit Hilfe von *Markov-Ketten* formal definieren und auf die theoretischen Grundlagen für die Existenz sowie der Konvergenz der Potenziteration – einer numerischen Methode zur Bestimmung des *PageRank* Vektors – eingehen.

Betrachten wir das Verhalten eines *Random Surfers* im Hyperlink-Graphen etwas genauer. Dazu nehmen wir an, dass der *Random Surfer* auf einer zufällig gewählten Seite zu surfen beginnt. Wenn sich der *Surfer* zum Zeitpunkt t auf einer Seite i befindet, dann befindet er sich zum nächsten Zeitpunkt $t+1$ auf einer zufällig gewählten Nachfolgerseite j von i. Die Wahrscheinlichkeit der nächsten zu besuchenden Webseite hängt dabei – wie bereits oben erklärt – alleine vom Ausgangsgrad der aktuell besuchten Seite ab. Beobachten wir nun das Verhalten des *Random Surfers* über einen längeren Zeitraum hinweg, so erhalten wir mit den besuchten Seiten einen Pfad im Hyperlink-Graphen. Dieser Pfad des *Random Surfers* kann als *Markov-Kette* interpretiert werden – einem stochastischen Prozess, in dem eine zufällige Sequenz von Zuständen im Verlaufe der Zeit betrachtet wird, bei der die Wahrscheinlichkeitsverteilung des nächsten Zustands allein vom aktuellen Zustand abhängt.

Definition 2.3 (Markov-Kette): Eine (einfache) *Markov-Kette* ist eine Folge von Zufallsvariablen X_0, X_1, $X_2, ...$ über einer (endlichen) Menge S, so dass die Wahrscheinlichkeit für $X_{t+1} = x$ nur vom Wert von X_t und t abhängt, d.h.

$$P(X_{t+1} = x \,|X_t = x_t, ..., X_0 = x_0) = P(X_{t+1} = x \,|X_t = x_t) = p^t_{x_t,x}$$

wobei $p^t_{x,y}$ die Übergangswahrscheinlichkeit $P(X_{t+1} = y \,|X_t = x)$ angibt. Eine *Markov-Kette* heißt homogen, falls die Übergangswahrscheinlichkeiten $p^t_{x,y}$ nicht von t abhängen.

Wir werden uns im Zusammenhang mit *PageRank* nur mit einfachen, homogenen *Markov-Ketten* beschäftigen. Die obige Definition besagt, dass der Zustand X_{t+1} nur vom vorherigen Zustand X_t abhängt, jedoch unabhängig von einer bestimmten Historie ist, wie der Prozess zum Zustand X_t gelangt ist. Dies wird als *Markov*-Eigenschaft bezeichnet. Es sei angemerkt, dass die *Markov*-Eigenschaft nicht bedeutet, dass X_{t+1} unabhängig der vorhergehenden Zuständen ist. Sie bedeutet lediglich, dass eine Abhängigkeit von X_{t+1} auf die vergangenen Zustände im Zustand X_t enthalten ist. Die *Markov*-Eigenschaft impliziert, dass die *Markov-Kette* eindeutig durch eine $|S| \times |S|$ Übergangsmatrix $\mathrm{M} = (m_{ij})$ (engl. transition matrix) beschrieben werden kann:

$$\mathrm{M} = (p_{x,y})_{x,y \in S} \tag{2.15}$$

Die Matrix (2.15) ist eine stochastische Matrix (siehe Definition (2.2)), deren Einträge die Wahrscheinlichkeiten der Übergänge zwischen den Zuständen angeben. Die Darstellung der *Markov-Kette* in Form der Übergangsmatrix eignet sich insbesondere zur Berechnung der Wahrscheinlichkeitsverteilung nachfolgender Zustände des Prozesses. Zur Vereinfachung der Notation bezeichnen wir mit $p_i^{(t)} = P(X_t = i)$ die Wahrscheinlichkeit, dass sich das System zum Zeitpunkt t im Zustand i befindet. Des Weiteren sei $p^{(t)} = \left(p_i^{(t)}\right)$ der Vektor bei einer gegebenen Wahrscheinlichkeitsverteilung der Zustände zum Zeitpunkt t. Die Wahrscheinlichkeit, dass sich das System zum Zeitpunkt $t + 1$ im Zustand j befindet, ergibt sich wie folgt:

$$p_j^{(t+1)} = \sum\nolimits_i m_{ij} p_i^{(t)} \tag{2.16}$$

oder in Matrixschreibweise:

$$p^{(t+1)} = \mathrm{M}^T p^{(t)} \tag{2.17}$$

Eine alternative Darstellung einer *Markov-Kette* ist die eines gewichteten Digraphen $D = (V, A, w)$. Diese Repräsentation ist insbesondere für das Nachvollziehen der später eingeführten Eigenschaften von *Markov-Ketten* nützlich. Die Transformation von einer Darstellung in die Andere ist eindeutig und basiert auf folgenden Beziehungen: Die Knotenmenge des Graphen ist durch die Zustandsmenge der *Markov-Kette* gegeben und das Gewicht $w(i, j)$ einer gerichteten Kante (i, j) ist durch den entsprechenden Eintrag der Übergangsmatrix $\mathrm{M} = (m_{ij})$ gegeben, d.h. $w(i, j) = m_{ij}$. Eine Sequenz besuchter Zustände wird durch einen gerichteten Pfad im Graphen repräsentiert. Mit diesen Grundlagen und Definitionen können wir nun das *Random Surfer Modell* als *Markov-Kette* interpretieren: Der Zustandsraum S besteht aus den Knoten $V = \{v_1, ...v_n\}$ des Hyperlink-Graphen, und die Sequenz von Zufallsvariablen korrespondiert mit dem Pfad des *Random Surfers* im Graphen, wobei die Übergangsmatrix der *Markov-Kette* durch die Matrix $P = (p_{ij})$ aus (2.13) gegeben ist. Falls wir allerdings die Übergangsmatrix P des Hyperlink-Graphen strikt als *Markov-Kette* übernehmen, so ist diese nicht adäquat definiert. Für Knoten ohne ausgehenden Hyperlinks, auch als *Dangling Pages* bezeichnet, ergeben die Einträge der mit diesen korrespondierenden Zeilen der Matrix in ihrer Summe 0, wohingegen die Übergangsmatrix einer *Markov-Kette* stochastisch sein muss. Dieses Problem können wir jedoch einfach dadurch beheben, indem wir für solche Zeilen eine a priori Wahrscheinlichkeit von $e_n = (1/n, ..., 1/n)$ annehmen. Interpretieren wir die *Markov-Kette* als Graphen, so ergibt sich durch diese Modifikation, dass ein Knoten ohne ausgehenden Links mit allen anderen Knoten im Graphen durch künstlich hinzugefügte Hyperlinks verbunden wird, wobei die Wahrscheinlichkeiten uniform auf diese Transitionskanten verteilt werden. Wir notieren diese neue Matrix als $P' = (p'_{ij})$:

$$p'_{ij} = \begin{cases} a_{ij}/d_{out}(i) & \text{falls } d_{out}(i) > 0, \\ 1/n & \text{andernfalls.} \end{cases} \tag{2.18}$$

Sei nun eine Wahrscheinlichkeitsverteilung $r^{(t)} = \left(r_i^{(t)}\right)$ für einen Websurfer gegeben sich zum Zeitpunkt t im Knoten i zu befinden, dann ist die Wahrscheinlichkeit, sich zum Zeitpunkt $t + 1$ im Knoten j zu befinden, folgende:

$$r_j^{(t+1)} = \sum\nolimits_i p'_{ij} r_i^{(t)} \tag{2.19}$$

Dies können wir wieder äquivalent in Matrixnotation notieren:

$$r^{(t+1)} = P'^T r^{(t)} \tag{2.20}$$

Von besonderem Interesse ist nun eine Wahrscheinlichkeitsverteilung r, die sich nach einem Übergang nicht mehr ändert und stabil bleibt. Dieser Vektor reflektiert rein intuitiv die relative Häufigkeit für den Besuch einer Seite und entspricht somit dem gesuchten *PageRank*.

Definition 2.4 (Stationäre Verteilung): Eine stationäre Verteilung einer *Markov-Kette* M ist eine Wahrscheinlichkeitsverteilung π, so dass gilt:

$$\pi = \mathrm{M}^T \pi \tag{2.21}$$

Ein erster Ansatz zur Bestimmung einer stationären Verteilung π und somit dem *PageRank* besteht in der iterativen Anwendung von Gleichung (2.20), bis die Sequenz $r^{(t)},\ t = 0, 1, 2, ...$ gegen den Fixpunkt[1] π konvergiert. Dieses iterative Verfahren wird als Potenziteration bezeichnet. Das Verfahren hat jedoch ein Problem: Es muss nicht konvergieren und falls dennoch Konvergenz vorliegt, so ist dies kein Garant dafür den gesuchten *PageRank* Vektor tatsächlich gefunden zu haben. Um dieses Problem zu beheben, müssen wir zunächst einmal dessen Ursache feststellen. Wir müssen also untersuchen unter welchen Bedingungen wir eine stationäre Verteilung π erhalten und wann die Potenziteration gegen diese Lösung konvergiert. Eine erste Ursache für das Problem liegt an Zustandsmengen der *Markov-Kette*, in die der *Random Surfer* stecken bleiben kann. Betrachten wir dazu zwei Webseiten, die jeweils einen Hyperlink auf die andere Webseite und ansonsten zu keiner weiteren Seite enthalten. Die beiden Knoten bilden eine sogenannte Senke aus der kein Pfad hinausführt. Nehmen wir weiterhin an, dass eine Webseite existiert, die einen Link zu einer dieser Seite enthält, dann bleibt der *Random Surfer* beim Betreten der Senke in dieser stecken. Dies hat zur Folge, dass von dieser Senke nur Autoritätsgewicht empfangen und nicht weiterverteilt wird, so dass die ganzen Gewichte in der Senke konzentriert werden und alle anderen Seiten, die keine Senken sind, einen *PageRank* von 0 zugewiesen bekommen. Abgesehen davon, ist die Frage, welche Senke mit dem gesamten Gewicht endet, nicht eindeutig zu beantworten, da dies vom Startvektor abhängig ist – eine weitere unerwünschte Eigenschaft. Die Gründe für dieses Problem liegen zum einen in der *Reduzibilität* der Matrix.

Definition 2.5 (Reduzible Matrix): Eine $n \times n$ Matrix A wird reduzibel genannt genau dann, wenn eine Permutationsmatrix P existiert, so dass $P^T AP$ eine obere Dreiecksmatrix mit quadratischen Diagonalblöcken ist. A heißt irreduzibel, falls A nicht reduzibel ist.

[1] Als Fixpunkt wird in der Mathematik ein Punkt bezeichnet, der die Gleichung $f(x) = x$ erfüllt

Folgendes Lemma liefert eine einfache Charakterisierung der *Irreduziblität* einer Matrix über deren Repräsentation als Graph [MU05].

Lemma 2.6 *Eine endliche Markov-Kette ist irreduzibel genau dann, wenn der assoziierte Graph stark zusammenhängend ist.*

Unter der Bedingung, dass die Matrix *irreduzibel* ist, garantiert das *Perron-Frobenius* Theorem (siehe Anhang A.1), dass die *Markovkette* eine eindeutig stationäre Verteilung besitzt, die vom *prinzipalen*[2] Eigenvektor $\lambda_1 = 1$ angenommen wird. Letzteres ist eine unmittelbare Konsequenz der Stochastizität (die Zeilen summieren sich zu 1) der Übergangsmatrix. Es stellt sich die Frage, ob die Potenziteration gegen diese Lösung konvergiert oder nicht. Im Allgemeinen ist dies nicht gegeben, denn für die Konvergenz der Potenziteration ist mit der *Aperiodiziät* der Matrix noch eine zweite Bedingung erforderlich. Die *Aperiodizität* bezieht sich nicht auf die Existenz einer Lösung – diese garantiert die *Irreduzibilität* – sondern auf die Konvergenz der eindeutigen Lösung aus Gleichung (2.20) durch die Potenziteration, die in deren Abwesenheit nicht garantiert ist. Um dies zu illustrieren, haben wir weiter unten ein Gegenbeispiel angegeben. Zunächst einmal definieren wir, was unter der *Aperiodizität* einer Matrix zu verstehen ist.

Definition 2.7 (Aperiodizität): Die Periode eines Zustands ist das größte d, so dass gilt:

$$\forall k, n \in \mathbb{N}.\ P(X_{k+n} = x \,|X_k = x) > 0 \Rightarrow d \text{ teilt } n \tag{2.22}$$

Falls $d = 1$, dann heißt der Zustand aperiodisch. Eine *Markov-Kette* heißt aperiodisch, falls jeder Zustand aperiodisch ist.

Auch für die *Aperiodizität* erhalten wir wieder eine anschauliche Interpretation, wenn wir den mit der *Markov-Kette* assoziierten Graphen heranziehen. Der Graph und somit die *Markov-Kette* ist genau dann *aperiodisch*, wenn für zwei beliebigen Knoten $u, v \in V$ Pfade beliebiger Länge existieren, wobei als Ausnahme von dieser Bedingung eine endliche Menge von Pfadlängen nicht vorzukommen brauchen. Um zu verstehen warum diese Bedingung – neben der *Irreduzibilität* – notwendig ist, betrachten wir als Gegenbeispiel einen stark zusammenhängenden Graphen, der aus zwei Knoten u und v besteht und die *Aperiodizität* Bedingung nicht erfüllt (alle Pfade von u nach v haben ungerade Pfadlängen). Führen wir für dieses Beispiel die Iterationen der Gleichung (2.20) mit dem Startvektor $r^{(0)} = (q, 1-q)$ aus, so pendelt der Vektor r in den einzelnen Iterationen abwechselnd zwischen $r^{(0)}$ und $r^{(1)} = (1-q, q)$. Trotz der *Irreduzibilität* der *Markov-Kette* stellt sich also dennoch keine Konvergenz ein, da zwischen den beiden Knoten ein Zyklus entsteht, bei der das Gewicht zwischen den Knoten oszilliert und folglich divergiert.

[2] Der prinzipale Eigenvektor entspricht dem Eigenvektor mit dem betragsmäßig größten Eigenwert

Das folgende Theorem aus der *Markov*-Theorie stellt den Zusammenhang zwischen den beiden angeführten Bedingungen, *Irreduzibilität* und *Aperiodizität*, und der stationären Wahrscheinlichkeitsverteilung π her (Beweis siehe [MU05]):

Satz 2.8 *(Konvergenz): Falls die Markov-Kette* M *irreduzibel und aperiodisch ist, dann konvergiert diese zu einer eindeutig stationären Wahrscheinlichkeitsverteilung* π*, d.h.* $\pi = \mathrm{M}^{\mathrm{T}}\pi$*.*

Um diese Probleme zu beheben, haben Brin und Page ihr Grundkonzept vom *PageRank* einer Revision unterzogen. Nach wie vor von der Hyperlink Struktur des Webs ausgehend, haben sie die ursprüngliche Übergangsmatrix des Hyperlink-Graphens in eine *irreduzible*, *aperiodische Markovkette* modifiziert. Wie wir aus der vorausgegangen Theorie wissen garantiert die *Irreduzibilität* die Existenz einer eindeutigen stationären Verteilung π, die den *PageRank* Vektor charakterisiert. Die Potenzmethode angewendet auf dieser stochastischen Matrix konvergiert dann immer gegen π und zwar unabhängig vom verwendeten Startvektor. Es folgt nun die Beschreibung, wie die Transformation der Hyperlink Struktur in eine *irreduzible* und *aperiodische Markovkette* zu bewerkstelligen ist. Wie zuvor beschrieben produziert die Modifikation der ursprünglichen Matrix P nach P' eine stochastische Matrix. Diese Modifikation allein ist jedoch nicht ausreichend, um die *Irreduzibilität* zu garantieren, da die Hyperlink Struktur inhärent *reduzibel* ist. Es sind also weitere Modifikationen notwendig, um die *Irreduzibilität* zu garantieren. Zu diesem Zweck können wir uns desselben Tricks bedienen, wie wir es zuvor bei der Behandlung der *Dangling Pages* gezeigt haben. Dazu werden für alle Einträge der Übergangsmatrix Mindestwahrscheinlichkeiten eingeführt, die dem Hinzufügen von künstlichen Links entsprechen, wodurch jeder Zustand von jedem anderen Zustand aus erreichbar wird, d.h. es liegt ein vollständiger Graph vor. Die Übergangsmatrix wird modifiziert zu:

$$P'' = dP' + (1-d)E, \quad E = v(1, ..., 1), \quad 0 < d < 1 \tag{2.23}$$

Durch die Transformation wird die Stochastizität der Matrix beibehalten, da die Wahrscheinlichkeitsverteilung lediglich in eine andere transformiert wird. Es sei darauf hingewiesen, dass durch diese Modifikation auch automatisch die *Aperiodizität* der *Markov-Kette* sichergestellt wird.

Das *Random Surfer Modell* kann nun mit dieser neuen Übergangsmatrix wie folgt interpretiert werden: Der *Random Surfer* befindet sich zu jedem Zeitpunkt t in einem Knoten im Hyperlink Graph. Für den Zeitpunkt $t+1$ wählt er entweder mit Wahrscheinlichkeit d zufällig einen ausgehenden Link des Knotens aus, oder aber, er teleportiert sich mit einer Wahrscheinlichkeit $(1-d)$ zu einem beliebigen Knoten im Graphen. Aus diesem Grund wird der Vektor v auch als *Teleportationsvektor* (engl. *teleportation vector*) bezeichnet, für den üblicherweise ein uniformer Vektor verwendet wird, bei der die

Wahrscheinlichkeit für das Ziel der Teleportation über alle Knoten gleich verteilt ist. Mit dieser neuen Definition der Markov-Kette ändert sich die Gleichung (2.20) zu:

$$\begin{aligned} r^{(t+1)} &= (P'')^T r^{(t)} \\ &= (dP' + (1-d)E) r^{(t)} \\ &= dP' r^{(t)} + (1-d)v(1,...,1) r^{(t)} \\ &= dP' r^{(t)} + (1-d)v \end{aligned} \tag{2.24}$$

wobei sich letztere Umformung zur Vereinfachung der Gleichung auf die Tatsache begründet, dass es sich beim PageRank um einen Wahrscheinlichkeitsvektor handelt. In Algorithmus 2.1 findet sich eine Zusammenfassung des PageRank Algorithmus, entnommen aus [Ber05].

Algorithmus 2.1 PageRank

Input: Übergangsmatrix P, Teleportationsvektor v und Dämpfungsfaktor d
Output: PageRank r

1: Initialisiere $r^{(0)} \leftarrow v$, $t \leftarrow 0$
2: **repeat**
3: $\quad r^{(t+1)} \leftarrow dP^T r^{(t)}$
4: $\quad \gamma \leftarrow \left\| r^{(t)} \right\|_1 - \left\| r^{(t+1)} \right\|_1$
5: $\quad r^{(t+1)} \leftarrow r^{(t+1)} + \gamma v$
6: $\quad \delta \leftarrow \left\| r^{(t+1)} - r^{(t)} \right\|_1$
7: $\quad t \leftarrow t + 1$
8: **until** $\delta < \varepsilon$
9: **return** $r^{(t)}$

HITS

Unabhängig von Brin und Page (1998) hat Kleinberg im selben Jahr eine andere Definition für die Relevanz von Webseiten vorgeschlagen. Kleinberg argumentiert, dass autoritative Seiten nicht notwendigerweise auf andere autoritativen Seiten verweisen müssen. Stattdessen gibt es neben autoritativen Seiten, spezielle Knoten – Hubs genannt – die eine Sammlung von Links zu mehreren relevanten, autoritativen Seiten enthalten. In HITS verfügt also jede Seite über zwei Identitäten: Hubs und Autoritäten. Die Hub Identität enthält die Information, wie gut eine Seite als Verweis auf nützliche Ressourcen agiert, wohingegen die Autorität Identität die Qualität der Seite als Ressource selbst beschreibt. Zwischen diesen beiden Identitäten existiert eine wechselseitige Beziehung der Verstärkung: Ein guter Hub ist eine Seite, die auf guten Autoritäten zeigt, während eine gute Autorität eine Seite ist, die von guten Hubs verwiesen wird. Um die Qualität einer Seite, wie gut diese als Hub bzw. Autorität agiert, zu quantifizieren, sind mit jeder Seite

ein Hub- und ein Autoritätsgewicht assoziiert. Um diese Idee algorithmisch umzusetzen hat Kleinberg dazu ein zweistufiges Propagierungsschema von Autoritätsgewichten vorgeschlagen, bei der die Beurteilung autoritärer Seiten durch Hubs verliehen wird, anstatt direkt zwischen Autoritäten, wie dies beim *PageRank* der Fall ist. Um die wechselseitige Beziehung zwischen diesen beiden Identitäten zu beschreiben, hat Kleinberg das Hubgewicht eines Knotens als Summe der Autoritätsgewichte aller Knoten definiert, die von diesem Hub verwiesen wird. Umgekehrt ist das Autoritätsgewicht eines Knotens als Summe der Hubgewichte aller Knoten definiert, die diese Autorität verweisen. Um dies mathematisch zu formulieren, führen wir zwei Werte für eine Seite i ein: ein Autoritätsgewicht x_i und ein Hubgewicht y_i. Unter der Annahme, dass zwei Startwerte $x_i^{(0)}$ und $y_i^{(0)}$ gegeben sind, werden diese Werte iterativ berechnet:

$$x_i^{(t+1)} = \sum\nolimits_{j \to i} y_j^{(t)} \qquad \text{und} \qquad y_i^{(t+1)} = \sum\nolimits_{i \to j} x_j^{(t)} \tag{2.25}$$

wobei $x_i^{(t)}$ und $y_i^{(t)}$ die Autoritäts- und Hubwerte des Knotens i zum Zeitpunkt t sind. Diese beiden Gleichungen können wieder in Matrixform unter Verwendung der Adjazenzmatrix notiert werden.

$$x^{(t+1)} = A^T y^{(t)} \qquad \text{und} \qquad y^{(t+1)} = A x^{(t)} \tag{2.26}$$

Beachte, dass wir diese beiden Gleichungen durch Substitution vereinfachen können:

$$x^{(t+1)} = A^T A x^{(t)} \tag{2.27}$$

$$y^{(t+1)} = A A^T y^{(t)} \tag{2.28}$$

Diese beiden Gleichungssysteme können wieder durch die iterative Potenzmethode gelöst werden – wie wir sie bereits zur Berechnung des PageRanks verwendet haben – um den dominanten Eigenvektor der Matrizen $A^T A$ und AA^T zu berechnen. Da die Matrix $A^T A$ die Autoritätswerte bestimmt, bezeichnen wir diese als Autoritätsmatrix und analog dazu bezeichnen wir AA^T als Hubmatrix. Die Berechnung des Autoritätsvektors x und des Hubvektors y kann also als Bestimmung der rechtsseitigen, dominanten Eigenvektoren von $A^T A$ respektive AA^T interpretiert werden.

Abschließend wollen wir kurz die Konvergenz und die Eindeutigkeit der Potenziteration untersuchen, wie wir dies beim PageRank Algorithmus betrachtet haben. Die Matrizen $A^T A$ und AA^T sind symmetrisch, positiv semidefinit und nichtnegativ, woraus wir wiederum folgern können, dass die Eigenwerte $\{\lambda_1, ..., \lambda_m\}$ ebenso nichtnegativ und reell sind. Wir wollen zunächst die Annahme treffen, dass die Eigenwerte in folgende Ordnung $\lambda_1 > \lambda_2 \geqslant \lambda_3 \geqslant ... \geqslant \lambda_k \geqslant 0$ gebracht werden können. In anderen Worten existiert ein eindeutiger dominanter Eigenwert im Spektrum, womit die im Zusammenhang mit dem

Algorithmus 2.2 HITS

Input: Adjazenzmatrix A
Output: Autoritätsvektor x und Hubvektor y

1: Initialisiere $x^{(0)} \leftarrow (1, 1, ..., 1), y^{(0)} \leftarrow (1, 1, ..., 1)$, $t \leftarrow 0$
2: **repeat**
3: $\quad x^{(t+1)} \leftarrow A^T A x^{(t)}$
4: $\quad x^{(t+1)} \leftarrow x^{(t+1)} / \left\| x^{(t+1)} \right\|_1$
5: $\quad y^{(t+1)} \leftarrow A A^T y^{(t)}$
6: $\quad y^{(t+1)} \leftarrow y^{(t+1)} / \left\| y^{(t+1)} \right\|_1$
7: $\quad \delta \leftarrow \left\| x^{(t+1)} - x^{(t)} \right\|_1$
8: $\quad t \leftarrow t + 1$
9: **until** $\delta < \varepsilon$
10: **return** $x^{(t)}, y^{(t)}$

PageRank Verfahren aufgetretenen Konvergenzprobleme vermieden werden. Kleinberg hat gezeigt, dass unter obiger Annahme $\lambda_1 > \lambda_2$ die Potenzmethode gegen den dominanten Eigenvektor konvergiert. Die Konvergenz des HITS Verfahren unterliegt jedoch der Einschränkung, dass die initialen Autoritäts- und Hubsvektoren nicht orthogonal zu den dominanten Eigenvektoren der Matrizen $A^T A$ und AA^T sein dürfen. Da die Eigenvektoren dieser beiden Matrizen – wie bereits oben festgestellt – nichtnegative Werte haben, genügt es, die Autoritäts- und Hubsvektoren mit positiven Werten zu initialisieren. Dabei ist die Konvergenz des HITS Algorithmus und die relative Reihenfolge der Knoten im Ranking unabhängig von der verwendeten Norm, die im Algorithmus zur Normalisierung der Werte verwendet wird. Allerdings besteht ein Problem mit der Eindeutigkeit im Grenzwert des Autoritäts- bzw. Hubvektor, falls die Bedingung $\lambda_1 > \lambda_2$ nicht gilt und somit die algebraische Vielfachheit des dominanten Eigenwerts größer als 1 ist. Der mit diesem Eigenwert assoziierte Eigenraum ist mehrdimensional und die Potenzmethode konvergiert gegen einen nicht notwendigerweise eindeutigen Punkt in diesem Raum.

2.4. Netzwerk Clustering: Identifizierung von Community Struktur

Eines der Hauptinteressen der *sozialen Netzwerkanalyse* besteht, neben der Zentralität, in der Identifizierung kohäsiver (lat.: cohaerere = zusammenhängend) Untergruppen von Akteuren in einem Netzwerk. Dieser Aspekt hat in den letzten Jahren unter dem Schlagwort Community Struktur verstärkt Interesse auf sich gezogen. Bevor wir jedoch auf diesen neu geprägten Begriff eingehen, stellen wir einige Konzepte aus der klassischen Netzwerkanalyse vor. Diese sollen vorrangig der Motivation dienen, denn im Gegensatz zu den umfangreichen Modellen der Zentralität haben die Autoren in [WF94] diesen Aspekt nur theoretisch analysiert und sind auf mögliche Methoden nicht weiter eingegangen. Wir beginnen zunächst mit dem theoretischen Hintergrund zur Untersuchung kohäsiver Untergruppen und folgern daraus einige Definitionen für diese. Anschließend

werden wir mit der Modularität eine Metrik zur Evaluierung einer gefundenen Community Struktur einführen, bevor wir im Anschluss daran einige Methoden zur Identifizierung von Community Struktur beschreiben, die in der jüngsten Vergangenheit vorgeschlagen wurden.

2.4.1. Hintergrund

In der *sozialen Netzwerkanalyse* wird der Begriff der Untergruppe durch die allgemeine Eigenschaft der Kohäsion zwischen den Mitgliedern einer Untergruppe formalisiert. Die Kohäsion drückt dabei einen inneren Zusammenhalt aus und bezieht sich in unserem Kontext auf den Zusammenhalt der Mitglieder einer Untergruppe, die durch die relative Dichte, die relative Häufigkeit oder die Zusammenhangsstärke der Verbindungen innerhalb dieser Untermenge beschrieben werden kann. Alternativ dazu kann die Kohäsion einer Untergruppe auch durch den Grad der Konnektivität quantifiziert werden, d.h. der minimalen Anzahl der Knoten, die entfernt werden müssen, damit die Akteurmenge nicht mehr zusammenhängend ist. Der Begriff der kohäsiven Untergruppe ist eng mit einem anderen begrifflichen Konzept der SNA verwandt, der sozialen Gruppe. Obwohl die soziale Gruppe ein in der Soziologie weit verbreitetes Konzept darstellt, existieren über dessen Bedeutung unterschiedliche Auffassungen. Ebenso existiert in der Literatur auch für die kohäsive Untergruppe keine einheitliche Definition, da zum einen – ähnlich wie bei dem Begriff der sozialen Gruppe – die Ansichten darüber auseinander gehen und zum anderen, viele verschiedene spezifische Netzwerkeigenschaften mit der Kohäsion von Untergruppen in Zusammenhang gebracht werden können.

2.4.2. Definitionen von Communities

Obwohl in der Literatur auf verschiedene Art und Weise versucht wurde die Idee von kohäsiven Untergruppen, respektiver Communities, begrifflich zu fassen, ist in der Frage nach einer einheitlichen Definition kein Konsens erzielt worden. Wir wollen hier kurz auf einige Definitionen aus [WF94] eingehen, die für die spätere Betrachtung der Methoden zur Identifizierung von Communities relevant sind. Diese können nach [DDDA05] in zwei Kategorien eingeteilt werden: Selbst-Beschreibende und Vergleichende Definitionen.

Selbst-Beschreibende Definitionen

Die grundlegende Definition einer Community geht auf die Definition der Clique in einem Graphen zurück. Eine Clique ist definiert als Untergraph eines Graphen, der aus mehr als zwei Knoten besteht und in dem alle Knoten miteinander verbunden sind. Die Cliquendefinition stellt angesichts ihrer präzisen mathematischen Formulierung einen intuitiven Ausgangspunkt dar, um die formalen Eigenschaften einer Community zu spezifizieren. Dies ist jedoch eine sehr restriktive Definition, die von größeren Gruppen in realen, zumeist spärlichen Netzwerken selten erfüllt wird, da das Fehlen einer einzelnen Verbindung zwischen den Mitgliedern einer Community genügt, um das Kriterium zu

verletzen. Angesichts dieser Problematik wurden weitere Definitionen entwickelt, die versuchen, die obigen Einschränkungen zu beheben, indem die Cliquen-Eigenschaft relaxiert wird. Diese Erweiterungen basieren auf dem Prinzip, dass die geodesische Distanz zwischen den Mitgliedern einer Community möglichst klein sein sollte. Zur Erinnerung, die geodesische Distanz zwischen zwei Akteuren beschreibt die Länge des kürzesten Weges zwischen diesen. Da die geodesische Distanz in einer Clique eins ist, liegt es nahe, diese Bedingung zu relaxieren, indem wir für die längste geodesische Distanz höhere Werte akzeptieren. Durch diese Maßnahme gelangen wir direkt zur Definition von *n-Cliquen*, einem Untergraphen bestehend aus mehr als zwei Knoten, in der die längste geodesische Distanz zwischen zwei beliebigen Akteuren der Gruppe nicht größer als n ist. Die derzeitige Definition von *n-Cliquen* zeigt jedoch noch hinsichtlich der Charakterisierung von Communities einige Schwächen auf. Dazu zählt, dass nach dieser Definition die geodesischen Pfade nicht nur innerhalb der Gruppe verlaufen müssen, sondern auch Knoten außerhalb der *n-Cliquen* enthalten können. Dies kann zu dem Problem führen, dass die *n-Clique* nicht mehr zusammenhängend sein muss, was unserer intuitiven Vorstellung einer Community widerspricht. Um diese Problematik zu umgehen, wurden mit *n-Clans* und *n-Clubs* zwei weitere Variationen der Clique eingeführt, auf die wir jedoch nicht weiter eingehen wollen.

Vergleichende Definitionen

Die oben beschriebenen Ansätze zur Charakterisierung von Communities basieren ausschließlich auf den Eigenschaften der Verbindungen innerhalb der Gruppe und sind somit aus sich selbst heraus definiert. Eine etwas andere Betrachtungsweise geht auf eine Charakterisierung von Communities nach Seidmann [Sei83] zurück: *„... [communities] in social networks have usually been seen informally as sets of individuals more closely to each other than to outsiders“*. Zur Charakterisierung von Communities ist es demnach nicht ausreichend, sich allein auf die Verbindungen innerhalb einer Community zu konzentrieren, sondern entscheidend ist der Vergleich dieser mit den ausgehenden Verbindungen der Community, d.h. den Verbindungen, die Mitglieder mit Nicht-Mitgliedern verbinden. Dabei sollte eine Community, verglichen mit ihrer Umgebung, relativ kohäsiv sein, in dem Sinne, dass die Verbindungen innerhalb einer Community dicht und die ausgehenden Verbindungen zum Rest des Netzwerks spärlich sind.

In der Literatur finden sich viele derartige Definitionen von Communities, von denen wir zwei aus [RCC+04] näher betrachten wollen, und zwar diese, die versuchen obige, intuitive Beschreibung einer Community mathematisch formal zu erfassen. Diese beiden Definitionen vergleichen die Verbindungen zwischen den Mitgliedern einer Community mit den Verbindungen zwischen Mitgliedern und Nicht-Mitgliedern. Wir gehen dabei von der Annahme aus, dass das Netzwerk in Form eines ungerichteten gewichteten Graphen vorliegt (im Falle eines ungewichteten Graphen weisen wir jeder Kante ein Gewicht von 1 zu; dadurch können wir gewichtete und ungewichtete Graphen uniform behandeln). Dazu betrachten wir für eine Knotenmenge $U \subset V$ den von dieser induzierten Untergraphen $H = (U, E(U))$, wobei wir mit $E(U)$ die Menge der Kanten bezeichnen, für die beide

Endknoten in U sind. Des Weiteren bezeichnen wir $w(V', V'') = \sum_{i \in V',\, j \in V''} w_{ij}$ die Summe der Kantengewichte mit einem Endknoten in V' und einem Endknoten in V''.

Definition 2.9 (Starke Community): Ein Untergraph $H = (U, E(U))$ ist eine Community im starken Sinne, falls:

$$w(\{v\}, U)) > w(\{v\}, V - U)) \quad \forall v \in U \tag{2.29}$$

In einer starken Community ist die Summe der Kantengewichte des Knotens einer Community zu den übrigen Knoten der Community größer als die Summe der Kantengewichte des Knotens zu dem Rest des Graphen. Da die Bedingung von jedem Knoten erfüllt werden muss, haben wir wieder eine sehr restriktive Definition einer Community vorliegen. Um die Bedingungen weiter zu relaxieren, haben Raddichi et al. folgende Vereinfachung der Definition vorgeschlagen: Anstatt beim Vergleichen der internen und externen Verbindungen jeden Knoten einer Community einzeln zu betrachten, sehen sie die Community als Einheit an.

Definition 2.10 (Schwache Community): Ein Untergraph $H = (U, E(U))$ ist eine Community im schwachen Sinne, falls:

$$w(U, U)) > w(U, V - U)) \tag{2.30}$$

Eine Community im schwachen Sinne ist also als Untergraph definiert, bei dem die Gesamtsumme der Kantengewichte innerhalb der Community größer ist als die Gesamtsumme der Kantengewichte zum Rest des Graphen. Diese Definition spiegelt unsere intuitive Vorstellung einer Community Struktur am besten wieder und soll ab sofort das beschreiben, was wir unter einer Community verstehen wollen.

In gewisser Hinsicht führt der Vergleich der internen Struktur einer Community zur externen Struktur zu einer Metrik, mit der die Qualität einer bestimmten Partition evaluiert werden kann. Eine solche Metrik wollen wir im nächsten Abschnitt beschreiben.

2.4.3. Metrik zur Evaluierung von Community Struktur

Nachdem wir nun eine Definition von Communities eingeführt haben, wollen wir uns mit einer in den letzten Jahren aufgekommenen Frage beschäftigen: Wie lässt sich feststellen, wie gut – im Sinne der obigen Definition – eine gegebene Partitionierung eines Netzwerkes in Communities ist? Um diese Frage zu beantworten, haben Newman und Girvan in [NG03] ein a posteriori Maß zur Evaluierung der Qualität einer Community

Struktur vorgeschlagen, die sie die Modularität nennen. Das Maß basiert auf der intuitiven Idee, dass eine zufällig gewählte Anordnung der Kanten in einem Netzwerk keine Community Struktur aufweist. Falls, in einer vorliegenden Community Struktur, die Kantenanzahl zwischen den Communities nicht signifikant von der Kantenanzahl einer zufälligen Anordnung abweicht, widerspricht dies unserer intuitiven Vorstellung einer natürlichen Aufteilung des Netzwerkes. Auf der anderen Seite, falls die Kantenanzahl zwischen den Communities signifikant kleiner ist, als dass, was wir bei einer zufälligen Anordnung der Kanten erwarten würden – oder äquivalent dazu, falls die Anzahl innerhalb der Communities signifikant größer ist – dann stellt dies einen Indikator für Community Struktur dar. Um dies zu präzisieren, betrachten wir mit $P_k = \{V_1, ..., V_k\}$ eine Partitionierung der Knotenmenge V des Netzwerkes in k disjunkte, nicht-leere Teilmengen bzw. Communities. Die Modularitätsfunktion Q einer Partition P_k ist dann wie folgt definiert:

$$Q(P_k) = \sum_{c=1}^{k} \left(\frac{w(V_c, V_c)}{w(V, V)} - \left(\frac{w(V_c, V)}{w(V, V)} \right)^2 \right) \tag{2.31}$$

Demnach misst $w(V_c, V_c)$ die Summe der Kantengewichte innerhalb einer Community. Die Quantität $w(V_c, V)$ misst die Summe der Gewichte aller Kanten, die mit einem Knoten (oder beiden) der Community V_c inzidieren und $w(V, V)$ misst die Gesamtsumme aller Kantengewichte im Graphen. Betrachten wir der Einfachheit halber einen ungerichteten Graphen mit binären Gewichten, dann misst die Quantität $\frac{w(V_c,V_c)}{w(V,V)}$ die empirische Wahrscheinlichkeit $p_{c,c}$, dass beide Endknoten einer zufällig gewählten Kante aus G in der Community c liegen. In ähnlicher Weise ist $\frac{w(V_c,V)}{w(V,V)}$ die empirische Wahrscheinlichkeit p_c, dass mindestens ein Ende einer zufällig gewählten Kante in der Community c liegt. Daraus können wir nun die Wahrscheinlichkeit für eine Kante berechnen in eine Community c zu fallen, unter der Annahme, dass die Kanten – unabhängig von der Community Struktur – zufällig angeordnet werden. Dies ist einfach die Wahrscheinlichkeit, dass beide Knoten einer Kante in dieselbe Community fallen, d.h. p_c^2.

Die Modularitätsfunktion Q ist also ein Maß für die Abweichung zwischen der beobachteten Wahrscheinlichkeiten $p_{c,c}$ der Kanten, die innerhalb der Communities liegen, und der erwarteten Wahrscheinlichkeit p_c^2 derselben Quantität, bei der die Kanten jedoch zufällig angeordnet werden, unabhängig von der darunter liegenden Community Struktur. Die Modularität kann sowohl positive als auch negative Werte annehmen, wobei positive Werte auf die Präsenz von Community Struktur hinweisen. Falls zwischen den Communities einer Aufteilung keine Kanten existieren, dann nimmt Q den maximalen Wert bei 1 an. Im anderen Extrem, falls die Aufteilung der Intra-Community Kanten nicht viel mehr als von einer zufällige Anordnung abweicht, dann ist $Q = 0$. In empirischen Untersuchungen haben Newman und Girvan beobachtet, dass Netzwerke mit signifikanter Community Struktur Q-Werte im Bereich zwischen 0.3 und 0.7 annehmen. Abbildung 2.1 zeigt ein Beispiel eines Netzwerkes, dessen Struktur drei Communities suggeriert, mit der dazugehörigen Formel zur Bestimmung der Modularität.

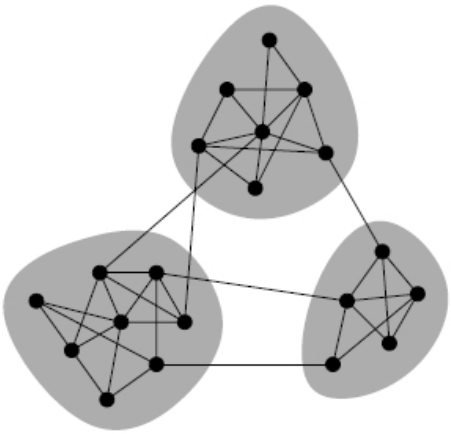


Abbildung 2.1.: Ein kleines Netzwerk mit offensichtlicher Community Struktur. In diesem Beispiel zerfällt das Netzwerk in drei Communities, hervorgehoben durch die Schattierungen, innerhalb derer die Verbindungen dicht sind, aber mit einer spärlichen Anzahl von Verbindungen zwischen den Communities. Die Modularität nimmt für die vorliegende Aufteilung mit 0.416 ihren optimalen Wert an. Die Modularität ergibt sich wie folgt: $Q(P_3) = \left(\frac{13}{37} - \left(\frac{17}{37}\right)^2\right) + \left(\frac{7}{37} - \left(\frac{10}{37}\right)^2\right) + \left(\frac{12}{37} - \left(\frac{15}{37}\right)^2\right) = 0.416$

Aus algorithmischer Sicht stellt sich natürlich gleich die Frage nach der Komplexität des Problems für ein festgelegtes $k > 1$ eine optimale Partitionierung des Graphen zu finden, bei der die Modularität maximiert wird. Auf Grund der kombinatorischen Natur dieses Optimierungsproblems scheint es in die Klasse der $\mathcal{NP}$-schweren[3] Probleme fallen, obwohl ein formaler Beweis ein noch offenes Problem ist. Um zu garantieren, eine Lösung für dieses diskrete kombinatorische Optimierungsprobleme zu finden, muss der gesamte Lösungsraum abgesucht werden. Die Anzahl der Möglichkeiten n Knoten in k nicht-leere Communities zu partitionieren ist durch die sogenannte Stirling Nummer $S_{n,k}$ gegeben [New04c]:

$$S_{n,k} = \begin{cases} 1 & \text{falls } k = 1 \text{ oder } k = n \\ S_{n-1,k-1} + kS_{n-1,k} & \text{andernfalls} \end{cases} \tag{2.32}$$

Wie die Rekurrenzrelation zeigt, wächst die Anzahl der möglichen Partitionen exponentiell mit der Anzahl der Knoten.

2.4.4. Netzwerk Clustering Methoden

Netzwerk Clustering bezeichnet das Problem eine – im Sinne der obigen Definition – gute Aufteilung des Netzwerks in Communities zu finden, in der jeder Akteur genau einer

[3] Ein Problem heißt $\mathcal{NP}$-schwer, wenn sich jedes andere Problem, das in $\mathcal{NP}$ liegt, in deterministisch polynomieller Zeit auf dieses reduzieren lässt.

Community zugeordnet ist. Netzwerk Clustering ist dabei nicht mit Daten Clustering zu verwechseln. Obwohl es sich bei beiden um Unsupervised Lernverfahren handelt, die eine Partitionierung einer Objektmenge vornehmen, unterscheiden sie sich in einem wesentlichen Punkt: Daten Clustering versucht die Objekte hinsichtlich ihrer Ähnlichkeit in den Attributwerten zu gruppieren, sodass die Intracluster-Ähnlichkeit maximiert und die Intercluster-Ähnlichkeit minimiert wird; Netzwerk Clustering zielt jedoch darauf ab, die Objekte hinsichtlich der Kohäsion im Netzwerk zu gruppieren, um die Intracluster-Kohäsion zu maximieren und die Intercluster-Kohäsion zu minimieren. Netzwerk Clustering bezieht sich demnach mit der Kohäsion auf eine topologische Eigenschaft des Netzwerkes, während das Daten Clustering die zu gruppierenden Objekte unabhängig von einer Netzwerkstruktur betrachtet. Hier sei noch anzumerken, dass beide Probleme auf Grund der ähnlichen Formulierung ihrer Zielfunktion in einem engen Zusammenhang stehen und Algorithmen für das eine Problem zur Lösung des anderen Problems herangezogen werden können. Diesen interessanten Aspekt wollen wir jedoch hier nicht weiter vertiefen. Die Untersuchung von Community Struktur in Netzwerken geht im Wesentlichen auf die Partitionierung von Graphen in der Informatik und dem Hierarchischen Clustering in der Soziologie zurück. Vor diesem historischen Hintergrund heraus sind zwei unterschiedliche Verfahren entstanden, die letztlich mit der Identifizierung von Community Struktur in einem Netzwerk, das gleiche Ziel verfolgen.

Hierarchisches Clustering

Die traditionelle Technik zur Extraktion von Community Struktur aus einem Netzwerk ist die Cluster Analyse, auch unter dem Namen Hierarchisches Clustering geläufig, die der Sozialen Netzwerkanalyse entspringt. Dieser Technik gehören eine Menge von Methoden an, die unter Verwendung verschiedener Metriken der Ähnlichkeit oder vielmehr der Konnektionsstärke zwischen den Knoten bzw. Akteuren, darauf abzielen, eine natürliche Aufteilung des Netzwerkes in kohäsive Gruppen zu identifizieren. In diesen Methoden werden den Knotenpaaren bzw. den Kanten zwischen den Knoten Ähnlichkeitswerte zugewiesen. Wenn nicht bereits ein eingebautes Ähnlichkeitsmaß im Netzwerk vorliegt, so kann nachträglich eine Metrik, auf die wir in Kürze eingehen werden, zur Bestimmung der Konnektionsstärke definiert werden. Im Normalfall wird jedem der $n(n-1)/2$ möglichen Knotenpaare in einem Netzwerk mit n Knoten eine solche Konnektionsstärke zugewiesen und nicht nur den Paaren, die durch eine Kante verbunden sind. Dies ist jedoch nicht dringend erforderlich, da für die verbleibenden, nicht adjazenten Knotenpaare eine Konnektionsstärke von 0 angenommen werden kann. Hierarchische Algorithmen können weiter in agglomerative und divisive klassifiziert werden, je nachdem ob das Hinzufügen oder Entfernen von Kanten zum oder vom Netzwerk betrachtet wird. In einer agglomerativen Methode wird mit n Knoten und der leeren Kantenmenge begonnen und sukzessive die Kanten in absteigender Reihenfolge der Ähnlichkeit zwischen den Knotenpaaren hinzugefügt. Hierbei sei erwähnt, dass die auf diese Weise hinzugefügten Kanten in keiner direkten Beziehung zu den Kanten des ursprünglichen Netzwerkes stehen, sondern letztere nur zur Bestimmung der Konnektionsstärke verwendet werden. Divisive Methoden arbeiten dem entgegengesetzt, indem ausgehend von einem vollständigen Graphen, die

Kanten in aufsteigender Reihenfolge entfernt werden.

Es bleibt nun zu klären, auf welche Weise wir die während des Prozedurverlaufs entstehenden Komponenten als Communities deklarieren. Indem wir – im Falle der agglomerativen Methode – dem Netzwerk mehr und mehr Kanten hinzufügen, vereinigen sich autarke Komponenten zu immer neuen Zusammenhangskomponenten. Eine auf der Hand liegende Methode besteht darin, die entstandenen Zusammenhangskomponenten als Communities zu deklarieren. Die auf diese Weise erzeugten Communities haben über den gesamten Prozedurverlauf hinweg folgende invariante Eigenschaft: Falls der Ähnlichkeitswert der zuletzt eingefügten Kante x beträgt, dann gehört jedes Knotenpaar mit einem höheren Ähnlichkeitswert als x zwangsläufig derselben Community an. Dies ist jedoch lediglich eine hinreichende, aber nicht notwendige Bedingung für die Mitgliedschaft zweier Knoten in derselben Community. In der Literatur wird diese Methode zur Identifizierung der Communities als *single-link* Methode bezeichnet. Hierarchische Clustering Algorithmen mit *single-link* produzieren eine Menge verschachtelter Komponenten der Knoten, die in Form eines Dendrogrammes visualisiert werden können. Ein Dendrogramm, wie aus Abbildung 2.2 ersichtlich, ist eine Baumstruktur, in der die Knoten den im Verlauf des Algorithmus zu Komponenten gruppierten Knoten entsprechen. Dabei sind die Blätter des Baumes die individuellen Knoten, die weiter oben in der Hierarchie, in den inneren Knoten, zu Komponenten vereinigt werden, bis hin zur Wurzel, die alle Knoten in sich vereinigt. Horizontale Schnitte im Dendrogramm reflektieren die unterschiedlichen Communities der Knotenmenge, wobei sich umso weniger Communities ergeben, je höher der Schnitt in Richtung Wurzel angesetzt wird. Eine andere Methode zur Extraktion von Community Struktur hierarchischer Verfahren ist die *complete-link* Methode, die in gewisser Weise das gegensätzliche Extrem zur *single-link* Methode beschreibt. In dieser Methode werden, wie zuvor, die Kanten sukzessive dem Netzwerk hinzugefügt, wobei diesmal die Communities als maximale Cliquen definiert sind, im Gegensatz zu zusammenhängenden Komponenten der single-link Methode. Wie in Abschnitt 2.4.2 erläutert, ist dies eine sehr restriktive Definition und die auf diese Weise erzeugten Communities haben über den gesamten Prozedurverlauf hinweg folgende invariante Eigenschaft: Falls der Ähnlichkeitswert der zuletzt eingefügten Kante x beträgt, dann hat eine auf diese Weise entstandene Community die Eigenschaft, dass zwei beliebige Mitglieder der Community einen höheren Ähnlichkeitswert als x haben. Demnach ist diese Bedingung, im Unterschied zur single-link Methode, eine notwendige aber nicht hinreichende Bedingung für die Mitgliedschaft zweier Knoten in derselben Community.

Einen wesentlichen Aspekt im Zusammenhang mit der Identifizierung von Communities haben wir jedoch bisher außer Acht gelassen und zwar fehlt eine Beschreibung wie wir aus den hierarchischen Verfahren die Communities extrahieren wollen. Da wir zum einen im Vorhinein keine Informationen über die tatsächliche Community Struktur vorliegen haben – wir sind ja genau an dieser interessiert – und zum anderen auch nicht wissen aus wie vielen Communities sich das Netzwerk zusammensetzt, müssen wir demnach angeben, wie wir aus den oben beschriebenen Verfahren die Community Struktur extrahieren wollen. Wie wir ja bereits wissen produzieren die hierarchischen Verfahren

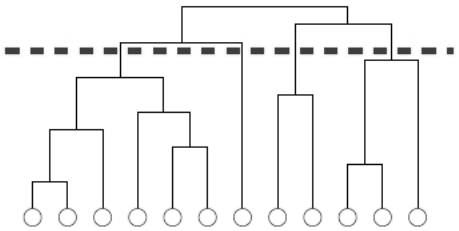


Abbildung 2.2.: Ein hierarchischer Baum oder Dendrogramm illustriert die Form von Ausgabe eines hierarchischen Clustering Verfahrens. Die Blätter im unteren Teil der Abbildung repräsentieren die individuellen Knoten des Netzwerks. Nach oben in der Hierarchie werden die Knoten zu immer größeren Communities vereinigt, bis hin zur Wurzel, in der alle Knoten zusammen zu einer einzigen Community vereint sind.

mit *single-link* mehrere mögliche Aufteilungen des Netzwerkes, die durch horizontale Schnitte im Dendrogramm reflektiert werden. Ein erster Ansatzpunkt besteht also darin die mit einem Schnitt im Dendrogramm induzierte Partition als Community Struktur herzunehmen. Diesbezüglich stellt sich die Frage, wo wir den Schnitt im Dendrogramm ansetzen müssen, um eine möglichst sensible Aufteilung des Netzwerks in Communities zu erhalten. Eine Antwort darauf liefert uns die in Abschnitt (2.4.3) eingeführte Modularitätsfunktion Q (2.31) zur Evaluierung von Community Struktur. Ausgehend von dieser Überlegung berechnen wir die Modularität für alle Aufteilungen des Netzwerks, die sich durch horizontale Schnitte des Dendrogrammes ergeben. Da größere Werte der Modularitätsfunktion auf die Präsenz von Community Struktur hinweisen, erhalten wir mit der Partition, deren Schnitt mit dem größten Q-Wert korrespondiert, die gewünschte Aufteilung des Netzwerkes in Communities. Wenn wir davon ausgehen, dass eine Konnektionsstärke bereits vorliegt, haben wir insgesamt betrachtet mit diesen Verfahren eine effiziente Methode zur Bestimmung von Communities vorliegen. Das hierarchische Verfahren besitzt eine Komplexität von $O(n^2 \log n)$, die aus der Sortierung der n^2 Knotenpaaren resultiert.

Hierarchische Clustering Verfahren werden im Wesentlichen durch die Wahl der Metrik für die Konnektionsstärke beeinflusst, da dadurch die Reihenfolge des Hinzufügens (Entfernens) der Kanten bei agglomerativen (divisiven) Verfahren eindeutig festgelegt wird. Es ist in der Literatur über soziale Netzwerke eine Vielfalt solcher Metriken vorzufinden, die jedoch im Kontext der so genannten Block Modelle auftreten, bei denen es sich um die Aufteilung des Netzwerkes in Communities oder Blöcken nach anderen Kriterien als der Kohäsion handelt. In der sozialen Netzwerkanalyse steht vielmehr die „Position“ von Akteuren im Mittelpunkt der Analyse. Die Position bezieht sich dabei auf eine Menge von Individuen, die durch Beziehungsstrukturen auf ähnliche Weise im Netzwerk eingebettet sind, wie beispielsweise die Ähnlichkeit ihrer Interaktionen oder sozialer Aktivitäten. Da die Position von Akteuren auf der Ähnlichkeit der Beziehungen zu anderen Akteuren beruht, anstatt ihrer Nähe oder Erreichbarkeit, unterscheidet sich

dieses Konzept von dem der kohäsiven Untergruppen. Insbesondere müssen Akteure, die dieselbe Position einnehmen, nicht notwendigerweise im direkten oder indirekten Kontakt stehen. In diesem Kontext beziehen sich die Kriterien der Knotenähnlichkeiten mehr auf die strukturelle Äquivalenz. Zwei Knoten werden strukturell äquivalent genannt, falls sie identische Verbindungen zu und von allen Akteuren im Netzwerk haben. Da exakte strukturelle Äquivalenz nur selten in Realwelt Netzwerken auftritt, wurden andere Maße definiert, die als Basis für die oben beschriebenen hierarchischen Algorithmen verwendet werden können, wenn auch in einem anderen Kontext als dem der Identifizierung von Communities. In Hinblick auf unsere Arbeit werden wir im nächsten Kapitel, bei der Beschreibung des Konsolidierungsalgorithmus, auf Maße eingehen, die versuchen die Konnektionsstärke zwischen Knotenpaaren zu quantifizieren.

Spektrales Clustering

Die Partitionierung von Graphen ist ein in der Informatik weit verbreitetes Problem. Eine typische Probleminstanz ist die Aufteilung der Knoten in Gruppen annähernd gleicher Größe, so dass die Anzahl der Kanten, die zwischen den Gruppen verlaufen, minimiert wird. Dieses Problem tritt beispielsweise bei der optimalen Zuteilung von Prozessen nach Prozessoren in parallelen Architekturen auf, mit dem Ziel, die Kommunikation zwischen den Prozessoren zu minimieren. Die meisten Verfahren zur Partitionierung von Graphen basieren dabei auf dem Prinzip *Teile und Herrsche* (engl. *divide and conquer*): Anfangs wird der Graph in zwei Partitionen aufgeteilt und dann das Verfahren rekursiv auf diese beiden angewendet, bis die gewünschte Anzahl an Gruppen erreicht ist. Da wir zur Modellierung sozialer Netzwerke Graphen verwenden, liegt es nahe, die Partitionierung von Graphen für das Clustering von Netzwerken einzusetzen. Da sich jedoch die Anforderungen an die Partitionen, wie sie von Anwendungen in der Informatik gestellt werden, sowohl in der Zielfunktion als auch in den Nebenbedingungen, von denen des Netzwerk Clusterings unterscheiden, sind diese neu zu formulieren. So streben beispielsweise die meisten Verfahren zur Partitionierung von Graphen möglichst balancierte Partitionen an, d.h. Knotenmengen gleicher Kardinalität, was aus der Perspektive des Netzwerk Clusterings – wo Communities unterschiedlich groß sein können – nur wenig Sinn macht. In diesem Abschnitt wollen wir mit *spektrales* Clustering ein Verfahren zur Partitionierung von Graphen einführen, das nicht auf dem Graphen direkt, sondern mit der Laplace Matrix auf einer Matrixrepräsentation des Graphen operiert. Mit *spektrale* Clustering Verfahren werden im Allgemeinen Methoden bezeichnet, die anhand des Spektrums – die Menge der Eigenwerte – der Matrix und den dazugehörigen Eigenvektoren eine Partitionierung des Graphen vornehmen. Wir wollen zunächst ungerichtete Graphen ohne Kantengewichte betrachten und anschließend zeigen, wie sich das Verfahren auf gewichtete Graphen erweitern lässt.

Die *Laplace Matrix* L eines ungerichteten Graphen $G = (V, E)$ ist eine $n \times n$ symmetrische Matrix mit Elementen:

$$L_{ij} = \begin{cases} -1 & \text{falls } i \neq j \text{ und } (i,j) \in E \\ 0 & \text{falls } i \neq j \text{ und } (i,j) \notin E \\ \deg(v_i) & \text{falls } i = j \end{cases} \tag{2.33}$$

Alternativ kann die *Laplace Matrix* als $L = D - A$ notiert werden, wobei A die *Adjazenzmatrix* des Graphen ist und D die Diagonalmatrix mit dem Grad der Knoten auf der Diagonalen ist. Die Laplace Matrix eines Netzwerks enthält, ähnlich wie die Adjazenzmatrix, alle wesentlichen Informationen über die Topologie des Netzwerks, besitzt jedoch darüber hinaus einige besondere algebraische Eigenschaften, die wir hier nachfolgend kurz aufführen wollen. Für Details und Beweis siehe [Mer94]. Die Laplace Matrix ist eine symmetrische[4], positive semi-definite[5] Matrix mit reellen, nicht-negativen Eigenwerten, deren korrespondierende Eigenvektoren eine reelle orthonormierte Basis[6] bilden. Nach Konstruktion der Laplace Matrix summieren sich alle Spalten und Zeilen zu null auf und folglich ist der Vektor $\mathbf{1} = (1, 1, ..., 1)$ ein Eigenvektor zum Eigenwert 0. Ein bedeutendes Theorem, das auf Fiedler zurück geht, besagt, dass die Anzahl der Eigenvektoren zum Eigenwert 0 mit der Anzahl der Zusammenhangskomponenten des Graphen korrespondiert. Dies ist leicht wie folgt einzusehen: Falls der Graph zusammenhängend ist, dann gibt es mit dem Vektor $\mathbf{1}$ nur einen Eigenvektor zum Eigenwert 0. Andernfalls -falls der Graph nicht zusammenhängend ist- besitzt die Laplace Matrix Blockdiagonalform, bei der jeder Diagonalblock mit einer Zusammenhangskomponente G_k, $(k = 1, ..., g)$, korrespondiert und demzufolge auch einen Eigenvektor $v^{(k)}$ zum Eigenwert 0 besitzt, mit Elementen $v_i^{(k)} = 1$ falls $i \in G_k$, andernfalls $v_i^{(k)} = 0$.

Um nun eine Brücke zu *spektrales* Clustering zu schlagen, wollen wir zeigen, wie der zweitkleinste Eigenvektor zur Partitionierung eines zusammenhängenden Graphen verwendet werden kann. Wie bereits oben erwähnt, ist bei einem zusammenhängenden Graph – also ein Graph mit nur einer Komponente – der kleinste Eigenwert λ_1 gleich 0 und der zweitkleinste Eigenwert λ_2 größer als 0. Der mit letzterem assoziierte Eigenvektor u_2 wird Fiedler Vektor genannt und enthält wesentliche Informationen über die Konnektivität des Graphen: Die Elemente des Vektors u_2 ordnen den korrespondierenden Knoten des Graphen Gewichte zu, wobei die Differenz in den Gewichten Aufschluss über die Distanzen zwischen den Knoten im Graphen gibt, d.h. je geringer die Knotengewichte des Fiedler Vektors auseinander liegen, desto näher liegen die korrespondierenden Knoten im Graph zusammen [Fie75]. Sortieren wir also die Elemente des Fiedler Vektors, so induziert diese eine Ordnung, bei der die korrespondierenden Knoten benachbarter Elemente des Fiedler Vektors auch topologisch gesehen nah zueinander liegen. Der Fiedler Vektor wird wegen dieser Eigenschaft auch als algebraische Konnektivität des Graphen bezeichnet und liefert zusätzlich ein Maß für die Qualität der Aufteilung, bei der kleinere Eigenwerte für eine bessere Aufteilung stehen. Dies legt die Vermutung nahe, dass

[4] Eine Matrix A heißt symmetrisch, falls $A = A^T$ gilt.

[5] Eine symmetische Matrix heißt positiv semidefinit, falls alle Eigenwerte größer oder gleich Null sind.

[6] Eine Basis eines Vektorraumes V heißt orthonormal, falls sich die Basis aus einer Menge von paarweisen orthogonalen Elementen zusammensetzt, die zudem normiert sind.

uns der Fiedler Eigenvektor eine Partitionierung mit geringem Schnitt liefert, wenn wir die Knoten bezüglich ihrer Vorzeichen der jeweiligen Komponente zuordnen. Diese Beschreibung ist der Kern der *spektralen* Bisektion [PSL90] bei der, ausgehend vom Fiedler Vektor, die Knoten von G in zwei Knotenmengen V^+ und V^- partitioniert werden.

Algorithmus 2.3 Spektrale Bisektion

Input: $G = (V, E)$
Output: Partitionen V^+ und V^-

```
1: u_2 ← Fiedler Eigenvektor des Graphen G
2: for all v ∈ V do
3:     if (u_2)_v < 0 then
4:         V^- ← V^- ∪ {v}
5:     else
6:         V^+ ← V^+ ∪ {v}
7:     end if
8: end for
```

Eine theoretische Rechtfertigung für die Spektrale Bisektion liefert der folgende Satz [Fie75]:

Satz 2.11 *(Satz von Fiedler): Sei $G = (V, E)$ ein zusammenhängender Graph und u der korrespondierende Fiedler Vektor des Graphen. Für eine gegebene reele Zahl $r \geqslant 0$ ist der induzierte Untergraph der Knotenmenge $V_1 = \{v_i \in V \,|u_i \geqslant -r\}$ zusammenhängend. Ähnlich gilt dies für jede reele Zahl $r \leqslant 0$. In diesem Fall ist der induzierte Untergraph der Knotenmenge $V_2 = \{v_i \in V \,|u_i \leqslant -r\}$ zusammenhängend.*

Dieser Satz impliziert, dass der Fiedler Vektor die Knotenmenge in einer sinnvollen Weise aufteilt, da wir mit einer Partitionierung gemäß Satz 2.11 mindestens eine zusammenhängende Partition erhalten. Nichtsdestotrotz ist nicht einsichtig, weshalb der Algorithmus 2.3 in der Praxis so gut funktioniert. Neben der graph-theoretischen Rechtfertigung aus Satz 2.11, erhalten wir eine einsichtigere Erklärung für die Wirksamkeit des Algorithmus, wenn das Bisektion Problem als diskretes Optimierungsproblem aufgefasst wird. Dazu betrachten wir das allgemeine Bisektion Problem: Gesucht ist die Partitionierung eines Graphen in zwei gleich große Knotenmengen V_1 und V_2, so dass der Schnitt $w(V_1, V_2)$ zwischen beiden Partitionen minimal ist. Wir wollen dieses Problem nun als diskretes Optimierungsproblem formulieren. Dazu führen wir für jeden Knoten eine Variable x_i ein, welche die diskreten Werte $+1$ oder -1 annehmen kann, je nachdem, welcher Partition der Knoten zugeordnet ist, d.h.:

$$x_i = \begin{cases} 1 & \text{falls } v_i \in V_1 \\ -1 & \text{falls } v_i \in V_2 \end{cases} \tag{2.34}$$

Das Gewicht des Kantenschnitts $w(V_1, V_2)$ eines Partitionsvektors $x \in \{+1, -1\}^n$ kann dann wie folgt bestimmt werden:

$$f(x) = w\left(V_1, V_2\right) = \frac{1}{4} \sum_{(i,j)\in E} (x_i - x_j)^2 \tag{2.35}$$

Die Gleichung ist korrekt, denn falls v_i und v_j in derselben Partition liegen, dann gilt $x_i - x_j = 0$. Umgekehrt, falls v_i und v_j in unterschiedlichen Partitionen liegen, dann gilt auf Grund der unterschiedlichen Vorzeichen $(x_i - x_j)^2 = 4$. Gleichung 2.35 können wir umschreiben zu:

$$\begin{aligned} f(x) &= \frac{1}{4} \sum_{(i,j)\in E} (x_i - x_j)^2 \\ &= \frac{1}{4} \left(\sum_{(i,j)\in E} (x_i^2 + x_j^2) - \sum_{(i,j)\in E} 2x_i x_j \right) \\ &= \frac{1}{4} \left(x^T D x - x^T A x \right) \\ &= \frac{1}{4} x^T L x \end{aligned} \tag{2.36}$$

Wir können nun das Bisketion Problem als diskretes Optimierungsproblem formulieren [Pot97]:

$$\begin{aligned} \text{Minimiere } f(x) &= \frac{1}{4} x^T L x \\ \text{s.t.} \quad x &\in \{+1, -1\}^n \\ x^T \mathbf{1} &= 0 \end{aligned} \tag{2.37}$$

Die Ganzzahligkeitsbedingung macht das Problem $\mathcal{NP}$-schwer, weshalb wir nicht erwarten können, eine optimale Lösung in polynomialer Zeit zu finden. Eine übliche Vorgehensweise bei Betrachtung solcher $\mathcal{NP}$-schwerer-Probleme besteht darin, diese zu vereinfachen, indem komplizierte Nebenbedingungen weggelassen oder abgeschwächt werden. Im vorliegenden Fall relaxieren wir die Ganzzahligkeitsbedingung und erlauben, dass die Lösungsmenge auch kontinuierliche Werte annehmen darf. Da die zulässige Lösungsmenge des diskreten Problems als Untermenge der reellen Zahlen in der Lösungsmenge des kontinuierlichen Problems enthalten ist, erhalten wir somit eine untere Schranke. Es bleibt zu klären, ob für das kontinuierliche Problem effiziente Lösungsverfahren existieren, und falls ja, stellt sich die Frage, wie wir aus der kontinuierlichen Lösung eine diskrete Lösung erhalten, um eine möglichst gute Approximation für das diskrete Problem zu erzielen. Die Optimierung des kontinuierlichen Problems ist

einfach und die Lösung kann nach einigen algebraischen Umformulierungen und unter Ausnutzung der Laplace Eigenschaften wie folgt angegeben werden [Pot97]:

$$\begin{aligned}\min_{x\in\{+1,-1\}^n, x^T1=0} f(x) &= \min_{x\in\{+1,-1\}^n, x^T1=0} \frac{1}{4}x^TLx \\ &\geqslant \min_{x^Tx=n, x^T1=0} \frac{1}{4}x^TLx \\ &= \frac{1}{4}u_2^TLu_2 \\ &= \frac{n}{4}\lambda_2\end{aligned} \tag{2.38}$$

wobei λ_2 der kleinste positive Eigenwert der Laplace Matrix L ist und u_2 der korrespondierende Eigenvektor (der Fiedler Eigenvektor). Das Größergleich-Zeichen signalisiert den Übergang des diskreten Problems (links des Zeichens) zum kontinuierlichen Problem (rechts des Zeichens), da wir mit dem Optimum des kontinuierlichen Problems eine untere Schranke erhalten. Das Minimum des kontinuierlichen Problems wird also vom Fiedler Vektor angenommen, der effizient durch die Lanczos Methode [GL89] bestimmt werden kann. Die Laufzeit der Lanczos Methode zur Berechnung des zweiten Eigenvektors kann durch den Term $m/(\lambda_3 - \lambda_2)$ approximiert werden, wobei m die Anzahl der Kanten im Graphen ist. Demnach hängt die Konvergenz auch davon ab, wie gut der Graph in Communities separiert werden kann, da, wie eingangs erwähnt, ein kleiner zweiter Eigenwert mit einer besseren Aufteilung korrespondiert.

Wie wir ja bereits aus der vorausgegangen Diskussion wissen, erhalten wir mit einer kontinuierlichen Lösung nur eine untere Schranke für unser eigentliches diskretes Problem. Nachdem wir nun eine kontinuierliche Lösung vorliegen haben, müssen wir uns ein Verfahren überlegen, um aus dieser eine möglichste gute Approximation für eine diskrete Partitionierung zu erhalten. Um eine diskrete Lösung aus der kontinuierlichen zu approximieren, können wir die Elemente des Fiedler Eigenvektors nach einem Rundungsschema in diskrete Werte bringen, wodurch wir eine zulässige Partitionierung erhalten. Wir betrachten dazu zwei mögliche Methoden. Die erste Methode setzt dabei den Schwerpunkt auf eine balancierte Partitionierung, indem die $n/2$ größten Elemente einer Partition zugeordnet werden und der Rest der anderen Partition. Bei der zweiten Methode erfolgt eine Zuordnung der Knoten anhand des Vorzeichens, d.h. $x_i = sign((u_2)_i)$. Insgesamt haben wir damit eine Heuristik vorliegen, die uns in einer akzeptablen Laufzeit eine zulässige Lösung liefert, dies jedoch ohne Garantie für die Güte der gefundenen Lösung.

Die *spektrale Bisektion* kann auf Graphen $G = (V, E)$ mit einer auf den Kanten definierten Gewichtsfunktion $w : E \to \mathbb{R}_0^+$ verallgemeinert werden. Zu diesem Zweck führen wir die gewichtete Laplace Matrix ein:

$$L_{ij} = \begin{cases} -w(i,j) & \text{falls } i \neq j \text{ und } (i,j) \in E \\ 0 & \text{falls } i \neq j \text{ und } (i,j) \notin E \\ \sum_{j=1}^{n} w(i,j) & \text{falls } i = j \end{cases} \tag{2.39}$$

Mit dieser Definition gelten die oben aufgeführten Herleitungen auf derselben Weise wie für die ungewichtete Laplace Matrix.

Wir haben nun mit der *spektralen Bisektion* ein Verfahren kennengelernt mit dem wir für einen Graphen bzw. einem Netzwerk eine gute Aufteilung in zwei Partitionen finden. Viele Netzerke setzen sich indes mehr als aus zwei Communities zusammen, so dass wir dieses *spektrale* Verfahren dahingehend erweitern wollen, um eine Aufteilung des Netzwerks in einer größeren Anzahl an Partitionen zu ermöglichen. Das Standardverfahren für dieses Problem ist es, diese bisektive Methode rekursiv auf das Netzwerk anzuwenden: Zunächst wenden wir die *spektrale Bisektion* an, um das Netzwerk in zwei Partitionen aufzuteilen und fahren dann mit den entstehenden Partitionen in analoger Weise fort, bis wir die gewünschte Anzahl an Partitionen erhalten.

3. Der Konsolidierungsalgorithmus

3.1. Einführung

Eine wichtige Aufgabe bei der Integration von Daten aus mehreren Datenquellen – wie dies beispielsweise im Data Warehousing stattfindet – ist die Entdeckung und Eliminierung von Duplikaten oder im Allgemeinen die Auflösung von Referenzen, die auf dieselbe Realwelt-Entität verweisen. In Datenquellen verschiedener Herkunft kann dieselbe Realwelt-Entität auf unterschiedliche Art und Weise repräsentiert werden. Die Variationen in der Repräsentation entstehen aus vielerlei Gründen: orthographische Fehler, Verwenden von Abkürzungen, unterschiedliche Namenskonventionen, über die Zeit variierende Namen und die Präsenz mehrerer Werte für Attribute. Um nun die Daten aus mehreren Quellen in eine strukturierte Repräsentation zusammenzuführen, muss erkannt werden, ob und welche Referenzen dieselbe Realwelt-Entität beschreiben. Dieses Problem wird als Objektkonsolidierung bezeichnet.

Objektkonsolidierung in digitalen Bibliotheken

Während es sich bei der Objektkonsolidierung um ein domänenübergreifendes Problem handelt, wollen wir dieses am konkreten Beispiel der digitalen Bibliothek *CiteSeer*[1][LGB99] illustrieren, in der das Problem mit Duplikaten inhärent auftritt. Bei dem Webportal *CiteSeer* handelt es sich um eine frei zugängliche Suchmaschine und Zitationsdatenbank für wissenschaftliche Informationen im Internet, die dem Auffinden elektronischer Versionen von Publikationen dient. Hinter diesem Portal verbirgt sich ein automatisiertes System, das zur Konstruktion seiner Zitationsdatenbank nach wissenschaftlichen Publikationen im Web sucht, um Informationen über die darin enthaltenen Autoren und Zitationen – Verweise von einer Publikation auf eine andere – zu extrahieren. Da die Daten aus diversen, voneinander unabhängigen Informationsquellen extrahiert werden, treten in diesem Zusammenhang eine Reihe von Ambiguitäten in Erscheinung, die einer Duplikatsbehandlung bedürfen. Ein erstes Problem betrifft die Publikationsdokumente selbst, von denen viele Exemplare im Web existieren können (z.B. kann eine von mehreren Autoren verfasste Publikation auf der jeweilig eigenen Publikationsseite der Autoren veröffentlicht sein). Während identische Dokumente einfach auszumachen sind (*CiteSeer* verwendet dazu eine Hashfunktion), führen geringfügige Änderungen in den Dokumenten ohne weitere Erkennungsmechanismen zu Duplikaten in der digitalen Bibliothek. Um dieses Problem zu beheben, bedient sich *CiteSeer* einer Technik des *Maschinellen Lernens*, bei der zwischen allen Dokumenten die in diesen vor-

[1] CiteSeer Scientific Literature Digital Library (dt.: Digitale Bibliothek wissenschaftlicher Literatur)

kommenden Sätze verglichen und Paare mit einer hohen Übereinstimmung als Duplikate identifiziert werden. Der Erkennungsmechanismus ist natürlich wesentlich komplexer als wir eben dargestellt haben. Ein anderes Problem in *CiteSeer*, das einer Duplikatsbehandlung bedarf, hängt mit der automatisierten Indizierung von Zitationen zusammen. Auf Grund des nicht einheitlichen Formats, wie Zitationen innerhalb von Publikationen aufgeführt sind, ist die Auflösung von Zitationen zu ihren korrespondierenden Publikationen eine Notwendigkeit, da erst durch diese Bereinigung die Voraussetzung für eine akkurate Zitationsanalyse geschaffen wird. Betrachten wir hierzu ein Beispiel aus [SB02], bei der die beiden angeführten Zitationen ein und derselben Publikation dem Mechanismus in *CiteSeer* zur Erkennung von Duplikaten entgangen sind:

- R. Agrawal, R. Srikant. Fast algorithms for mining association rules in large databases.In VLDB-94, 1994.
- Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules In Proc. Of the 20th Int'l Conference on Very Large Databases, Santiago, Chile, September 1994.

Dieses Beispiel unterstreicht die Nichttrivialität der Duplikatserkennung. Nur in den wenigsten Fällen lassen sich die Publikationen allein anhand der Titelinformation auflösen, so dass zusätzliche Informationen miteinbezogen werden müssen. Vorausgesetzt, dass die Unterfelder der Zitationen korrekt separiert wurden – das per se ein nicht einfaches Problem ist – kann das Wissen über den extrahierten Feldern, wie beispielsweise die Autorennamen, das Erscheinungsjahr der Veröffentlichung oder auch die tagende Konferenz, einen nützlichen Kontext darstellen, der zur Disambiguierung von Publikationen von Nutzen sein kann. Dieses Konsolidierungsproblem – zu identifizieren welche Zitationen mit derselben Publikation korrespondieren – wurde in der Vergangenheit intensiv untersucht, so dass in unserer Arbeit ein anderes Problem Gegenstand der Betrachtung sein soll: Die Konsolidierung mehrdeutiger Autorennamen in Publikationen zur Konstruktion eines bereinigten Koautorennetzwerks.

Koautorennetzwerke

Bei einem Koautorennetzwerk handelt es sich um ein *soziales Netzwerk*, mit dem sich in gewisser Weise die Kollaboration zwischen den Autoren dokumentieren lässt. Dabei werden die Autoren in einem solchen Netzwerk als Knoten repräsentiert, zwischen denen jeweils eine Kante existiert, wenn sie gemeinsam eine oder mehrere Publikationen veröffentlicht haben. Wir sprechen dann davon, dass die Autoren zusammen eine Koautorenschaft eingehen. Die Struktur eines Koautorennetzwerks kann viele interessante Eigenschaften akademischer Communities offen legen und darüber hinaus zur Bestimmung des Status (bzw. des Prestiges) individueller Forscher verwendet werden. Obwohl eine gewisse Ähnlichkeit zu den in der Literatur intensiver untersuchten Zitationsnetzwerken besteht, impliziert die Koautorenschaft eine sozial stärkere Bindung als dies von Zitationen ausgeht. Der Grund hierfür ist, dass Zitationen in Publikationen erscheinen

können, ohne dass dazu notwendigerweise die Autoren in Kontakt zueinander stehen müssen. Die Koautorenschaft impliziert also eine kollegiale, soziale Beziehung, weshalb sich diese Netzwerke eher in die *soziale Netzwerkanalyse* einordnen lassen, als dies bei Zitationsnetzwerke der Fall ist. Für eine Übersicht über die Eigenschaften derartiger Koautorennetzwerke sei der interessierte Leser auf die Arbeit [New04a] von M. E. J. Newman verwiesen. Im Nachfolgenden wollen wir diese Netzwerke im Kontext der Objektkonsolidierung untersuchen.

Überblick über unseren Ansatz

Wie bereits in der Einleitung erwähnt, wollen wir uns in dieser Arbeit mit dem Problem der Konsolidierung mehrdeutiger Autorennamen in Publikationen befassen. Ziel soll sein, aus einer Menge von Publikationen ein bereinigtes Koautorennetzwerk zu konstruieren. Die Bereinigung bezieht sich dabei auf die Auflösung der Autorenreferenzen zu ihren korrespondierenden Entitäten, wodurch die Voraussetzung für eine zuverlässige Analyse in dem betreffenden Netzwerk geschaffen wird. Die Mehrheit der traditionellen Techniken der Datenbereinigung verwendet zur Auflösung der Referenzen eine Ähnlichkeitsfunktion, die anhand der Attributinformationen der Referenzen entscheidet, ob zwei Referenzen dieselbe Entität repräsentieren oder nicht. Die Attributwerte einer Referenz werden üblicherweise der Beschreibung der Entitäten – wie sie in der Datenmenge zu beobachten sind – entnommen, wobei auch als Alternative dazu, die Attributinformation aus dem spezifischen Kontext der Referenzen hervorgehen kann. Basierend auf diesen Informationen werden dann die zueinander ähnlichen Referenzen in Clusters zusammengefasst, so dass jeder Cluster mit einer einzelnen Realwelt-Entität korrespondiert und unterschiedliche Clusters mit unterschiedlichen Entitäten korrespondieren. Abgesehen von den Attributinformationen kann auch ein zusätzlicher Kontext von Interesse sein, bei der die Objekte durch Relationen in Beziehung zueinander stehen. Bei unserem behandelten Konsolidierungsproblem haben wir beispielsweise mit der Kollaboration zwischen Autor und Koautor einen solchen zusätzlichen Kontext in Form der Koautorenschaft vorliegen, den wir zur Konsolidierung – bzw. je nach Sichtweise auch Disambiguierung – der Autorenreferenzen verwenden wollen. Gegenstand dieser Arbeit soll sein, zu untersuchen, auf welche Weise das in den Relationen enthaltene Wissen ausgenutzt werden kann, um die Qualität der Objektkonsolidierung signifikant zu verbessern.

Betrachten wir zur Motivation den Fall, dass wir daran interessiert sind festzustellen, ob zwei Publikationen gemeinsame Autoren haben. Eine erste Idee wäre, die Autorennamen bezüglich ihrer Textähnlichkeit zu vergleichen, um eine Übereinstimmung festzustellen. Dies stellt jedoch ein Problem dar, denn die Darstellung von Autorennamen ein und desselben Autors kann in unterschiedlichen Publikationen signifikant variieren. Die häufigsten Unterschiede finden sich dabei in der großen Anzahl an Möglichkeiten vor, wie der Vorname oder mehrfache Vornamen eines Autors gebildet werden kann. Für einen Autor namens „Jeffrey David Ullman“ sind beispielsweise „J. D. Ullman“, „Jeff Ullman“, „Jeffrey D. Ullman“ usw. mögliche Namensbildungen, wie sie in Publikationen

auftreten können. Für selten vorkommende Namen ist das Problem durch spezielle Algorithmen für Namenstransformationen zu bewerkstelligen, aber für populäre Namen, wie beispielsweise dem englisch-amerikanischen Namen „J. Smith“, erscheint das Problem schwieriger: Auch wenn wir dazu in der Lage wären die Gleichheit der Namen festzustellen, so sind wir – aufgrund der Häufigkeit des Vorkommens dieses Namens – dennoch mit dem Dilemma konfrontiert, zu entscheiden, ob zwei Autorenreferenzen mit Namen „J. Smith“ denselben Autor repräsentieren. Nehmen wir nun aber an, dass die Koautoren von „J. Smith“ in beiden Publikationen dieselben sind, so können wir zu einer größeren Konfidenz daraus schließen, dass beide Referenzen tatsächlich denselben Autor repräsentieren. Dieses Beispiel wirft jedoch hinsichtlich der Auflösung von Referenzen zwei Fragestellungen auf: Wir haben zur Bestimmung der Referenzen die Koautoren verglichen, sind dabei jedoch stillschweigend von der Annahme ausgegangen, dass die Referenzen der Koautoren eindeutig sind. Was ist aber, wenn die Koautorenreferenzen selbst wiederum mehrdeutig sind und zuvor erst einmal aufgelöst werden müssen? Und gegeben des Falles, dass die Referenzen zwar direkt keine gemeinsamen Koautoren haben, jedoch die Koautoren der Koautoren in Beziehung stehen. Wie können wir diese indirekte Information bei der Auflösung der Referenzen berücksichtigen?

In dieser Arbeit wollen wir versuchen, Antworten auf diese Fragestellungen zu geben, wobei wir diese Fragen im Kontext der Konsolidierung von Autorenreferenzen nachgehen wollen. Zu diesem Zweck wollen wir auf einen bereits existierenden Konsolidierungsalgorithmus zurückgreifen, der 2005 von Chen et al. in einer von ihnen veröffentlichten Arbeit [CKM05] vorgestellt wurde. Allerdings ist deren Konsolidierungsalgorithmus vielmehr als ein Framework zu verstehen, welches lediglich ein Gerüst zur Verfügung stellt, in der die einzelnen Phasen noch durch spezifische Algorithmen zu implementieren sind. Der Konsolidierungsalgorithmus geht dabei von folgender Hypothese aus, die in [CKM05] als *Kontext-Attraktions-Prinzip* (engl.: *Context Attraction Principle*) bezeichnet wird:

- Wenn zwei Referenzen dieselbe Entität repräsentieren, ist es wahrscheinlich, dass diese beiden Referenzen über weiteren expliziten sowie impliziten Beziehungen miteinander verbunden sind.
- Wenn zwei Referenzen unterschiedliche Entitäten repräsentieren, dann ist davon auszugehen, dass diese nur schwach über andere Beziehungen verbunden sind, im Vergleich zu den Referenzen, die dieselbe Entität repräsentieren.

In dem von ihnen vorgestellten Verfahren wird die zugrunde liegende Datenmenge als attributierter relationaler *Referenzgraph* modelliert, mit den aufzulösenden Referenzen als Knoten und den eventuell vorhandenen Beziehung zwischen den Referenzen als Kanten. Im Falle eines Autorennetzwerkes ergibt sich dieser *Referenzgraph* aus den Autorenreferenzen und der Koautorenschaft, wobei erstere den Knoten und letztere den Kanten in diesem Graphen entspricht. Ziel ist dann, unter Ausnutzung der Attributinformationen und den Beziehungen zwischen den Referenzen, einen bereinigten *Entitätengraph* zu rekonstruieren, bei dem alle Referenzen, die sich auf dieselbe Entität beziehen, zu einem Knoten konsolidiert sind. Im Gegensatz zu deren Arbeit wollen wir den Prozess

zur Auflösung der Referenzen in einem Graphen als *graphbasierte Objektkonsolidierung* bezeichnen, anstatt nur als Objektkonsolidierung. Der Algorithmus läuft in drei Phasen ab: Zu Beginn verwendet das Framework in einer ersten Phase die Attributinformationen, um die Referenzen zu identifizieren, die auf Grund ihrer Ähnlichkeit potentiell dieselbe Entität repräsentieren. Anschließend werden in einer zweiten Phase die einander ähnlichen Referenzen paarweise analysiert, um die Konnektionsstärke zwischen diesen Referenzen im Graphen zu quantifizieren. Das Framework verwendet dann in einer abschließenden dritten Phase ein Verfahren zur Partitionierung von Graphen, um die Referenzen basierend auf der gemessenen Konnektionsstärke zu konsolidieren.

Zur Abgrenzung von dieser existierenden Arbeit besteht die Hauptmotivation unserer Arbeit darin, das Framework in dem übergeordneten Kontext der *sozialen Netzwerkanalyse* zu untersuchen. Dabei sind wir insbesondere an der Frage interessiert sind, wie oder ob überhaupt sich die verschiedenen Phasen des Frameworks mit Methoden aus der *sozialen Netzwerkanalyse* bestimmen lassen. Wir wollen dazu versuchen, ein entsprechendes Modell in der Theorie der *sozialen Netzwerkanalyse* ausfindig zu machen, in die sich die jeweilige Phase einordnen lässt. Anschließend wollen wir ausgehend von dieser Erkenntnis untersuchen, wie sich die Methoden der SNA zur Lösung der jeweiligen Phase anpassen lassen. Da die erste Phase als klassisches Clustering-Problem formuliert werden kann und daher dem Bereich des Maschinellen Lernens zuzuordnen ist, konzentrieren sich unsere Untersuchungen auf die zweite und dritte Phase des Konsolidierungsalgorithmus. Wie wir bald sehen werden, nehmen insbesondere die beiden Aspekte, Bestimmung der Akteurszentralität und die Identifizierung von Community Struktur aus den Abschnitten 2.3 respektiver 2.4, eine tragende Rolle beim Entwurf unserer Verfahren ein. Aufgrund der umfassenden Theorie, die sich hinter der *sozialen Netzwerkanalyse* verbirgt, hoffen wir, dass diese Verfahren in Bezug auf die Qualität der Konsolidierung besser performen als die in [CKM05] vorgeschlagenen Modelle. Ebenso wie bei anderen algorithmischen Betrachtungen auch, spielt die Frage nach der Effizienz der eingesetzten Methoden eine wesentliche Rolle, so dass wir diese jeweils in einem separaten Unterabschnitt untersuchen wollen.

Der Hauptteil dieser Arbeit ist wie folgt gegliedert: In Abschnitt 3.3.3 wird die Funktionsweise des Konsolidierungsalgorithmus aus [CKM05] beschrieben und die Terminologie für die weiteren Teile eingeführt. In den darauf folgenden beiden Abschnitten wollen wir die Phase 2 und Phase 3 des Frameworks aus der Sicht der *sozialen Netzwerkanalyse* formulieren und jeweils eine konkrete Methode aus dieser zur Bestimmung der jeweiligen Phase vorstellen. Wir wollen im Vergleich zu unseren Vorschlägen auch auf die in [CKM05] vorgeschlagenen Modelle eingehen. Das Kapitel schließt in Abschnitt 3.6 mit der Einführung einer Metrik ab, die wir im Experimentierteil unsere Arbeit dazu verwenden wollen, die in dem Konsolidierungsframework eingesetzten Verfahren zu evaluieren.

3.2. Problemstellung Objektkonsolidierung

Zur Formulierung des Problems der Objektkonsolidierung betrachten wir eine Datenmenge D, die aus einer Menge von Entitäten $E = \{e_1, ..., e_{|E|}\}$ besteht. Die Entitäten sind in der Datenbank jedoch nicht direkt zugänglich, sondern lediglich durch ihre korrespondieren Referenzen $X = \{x_1, x_2, ..., x_{|X|}\}$ beobachtbar, die aus einer Menge von Attributen bestehen und die unterliegenden Entitäten repräsentieren. Ziel der Objektkonsolidierung ist vorherzusagen, welche Referenzen mit derselben Entität korrespondieren, unter der Annahme, dass eine dem Konsolidierungsalgorithmus unbekannte Resolutionsfunktion $r : X \rightarrow E$ vorliegt, die jede Referenz auf ihre korrespondierende Entität abbildet. Die Resolutionsfunktion ist also eine surjektive[2] Funktion mit der Referenzmenge X als Definitionsbereich und der Entitätenmenge E als Wertebereich. Da wir diese Abbildung im Vorhinein nicht kennen, kommt aus der Sicht des Maschinellen Lernens nur ein Unsupervised Lernverfahren in Betracht. Dabei erscheint es naturgemäß die Objektkonsolidierung als Clustering Problem zu formulieren, da beide Probleme, speziell die Objektkonsolidierung, mit der Disambiguierung von Objekten im Zusammenhang stehen. Das Ziel besteht dann darin, alle Referenzen, die von der Resolutionsfunktion auf dieselbe Entität abgebildet werden, in denselben Cluster zu partitionieren. Dazu ist eine Partitionierung der Referenzen X in eine Clustermenge $\mathcal{C}$ von $|E|$ disjunkten, nicht-leeren Cluster $\mathcal{C} = \{C_1, C_2, ..., C_{|E|}\}$ zu finden, so dass einerseits alle Referenzen innerhalb eines Clusters und andererseits keine zwei Referenzen aus zwei verschiedenen Cluster, dieselbe Entität repräsentieren, d.h. die Clustermenge muss folgende zwei Bedingungen erfüllen:

$$\begin{aligned} &1. \quad \forall i \in \{1, ..., |E|\}. \quad x, y \in C_i \Rightarrow r(x) = r(y) \\ &2. \quad \forall i, j \in \{1, ..., |E|\}. \quad i \neq j \wedge (x \in C_i, y \in C_j) \Rightarrow r(x) \neq r(y) \end{aligned} \tag{3.1}$$

Beide Bedingungen müssen gleichzeitig erfüllt sein, da andernfalls für jede einzelne Bedingung eine triviale Clusterlösung angegeben werden kann. Um dies einzusehen können wir für den Fall, in der nur die erste Bedingung gefordert ist, die triviale Lösung konstruieren bei der jede Referenz einem unterschiedlichen Cluster zugewiesen wird. Im anderen Fall – nur die zweite Bedingung ist gefordert – besteht die triviale Clusterlösung darin alle Referenzen einem einzelnen Cluster zuzuweisen.

Hinsichtlich der Objektkonsolidierung sind mehrere Szenarien denkbar, je nachdem wie viel Wissen über die Entitäten bzw. der unbekannten Resolutionsfunktion vorliegt. In einem Szenario kann beispielsweise die Kardinalität der Entitätenmenge bekannt sein und als nützliche Information im Konsolidierungsalgorithmus berücksichtigt werden, da dadurch die Anzahl der Cluster im Vorhinein festgelegt werden kann. In einem anderen Szenario können teilweise Referenzen schon aufgelöst sein und zur weiteren Disambiguierung verwendet werden. In unserer Arbeit betrachten wir jedoch das allgemeine Problem

[2] Bei einer surjektiven Funktion wird jedes Element des Wertebereiches mindestens einmal als Funktionswert angenommen

Abbildung 3.1.: Diese Abbildung zeigt die Kontraktion der Knoten 3 und 4 und dem daraus resultierenden Graphen.

der Objektkonsolidierung, in der weder Informationen über die Anzahl der Entitäten noch Informationen über die Resolutionsfunktion zur Verfügung stehen.

3.3. Der Konsolidierungsalgorithmus als Framework

In diesem Abschnitt wollen wir den domänenunabhängigen Ansatz von Chen et al. zur Konsolidierung von Objekten am Beispiel von Koautorennetzwerken aufzeigen. In ihrem Ansatz wird die zugrunde liegende Datenmenge als attributierter relationaler Graph (ARG) modelliert – einer relationalen Struktur, bei der die Knoten mit den zu konsolidierenden Referenzen und die Kanten mit den beobachteten Relationen zwischen den Objekten korrespondieren. Das Ziel besteht darin, diesen *Referenzgraphen* zu bereinigen, um den eindeutigen minimalen Entitätengraphen zu identifizieren, bei dem alle Referenzen, die dieselbe Entität repräsentieren, zu einem Knoten konsolidiert sind. Nach einer kurzen Einführung in die verwendete Notation wollen wir zunächst in Erweiterung an der Problemdefinition *Objektkonsolidierung* aus Kapitel 3.2 die *graphbasierte Objektkonsolidierung* vorstellen. Im Anschluss daran folgt die Beschreibung des generischen Frameworks zur Konsolidierung von Objekten, wobei die einzelnen Schritte dieses Frameworks im Detail dargestellt werden.

3.3.1. Ergänzende Notation

An dieser Stelle folgen in Ergänzung an Kapitel 2.2 noch einige graphentheoretische Definitionen, die wir im Zusammenhang mit der *Graphbasierten Objektkonsolidierung* verwenden wollen. Zunächst definieren wir, was unter der Kontraktion einer Knotenmenge W zu verstehen ist. Für einen Graphen $G = (V, E)$ und eine Knotenmenge $W \subseteq V$ bezeichnen wir mit $G(W)$ den Graphen, der durch Kontraktion der Knotenmenge W entsteht. Das heißt, die Knotenmenge von $G(W)$ besteht aus den Knoten $V - W$ und einem neuen Knoten w, der an die Stelle der Knotenmenge W tritt. Die Kantenmenge von $G(W)$ enthält alle Kanten aus $E(V - W)$ und alle Kanten $\delta(W) = \{(u, v) \in E \,|u \in W, v \in V - W\}$, wobei der Endknoten aus W durch w ersetzt wird (siehe dazu Abbildung 3.1).

Da der Konsolidierungsalgorithmus auf attributierte relationale Graphen (ARG)

operiert, wollen wir der Vollständigkeit halber eine kurze Definition zu dieser relationalen Struktur angeben: Ein ungerichteter ARG besteht aus dem 6-Tupel $G = (V, E, A_V, A_E, \alpha_V, \alpha_E)$, wobei die Komponenten folgendes bezeichnen: V und E stehen für die Knotenmenge bzw. Kantenmenge. A_V (A_E) bezeichnet eine Menge von Knotenattributen (Kantenattributen), während α_V (α_E) eine Funktion ist, die mit jedem Knoten (Kante) das korrespondierende Attribut assoziiert. Im Folgenden wollen wir versuchen, diesen Formalismus weitestgehend zu vermeiden und uns lediglich darauf beschränken, dass die Kanten des ARG mit Attributen versehen sind, um zwischen den unterschiedlichen Relationen der Referenzen unterscheiden zu können.

3.3.2. Graphbasierte Objektkonsolidierung

In Ergänzung an die in Kapitel 3.2 eingeführten Problemdefinition der *Objektkonsolidierung* wollen wir mit der *graphbasierten Objektkonsolidierung* eine Erweiterung zu dieser angeben. Zu diesem Zweck wird die zugrunde liegende Datenmenge als attributierter relationaler Graph betrachtet, bei der die Knotenmenge mit den aufzulösenden Referenzen und die Kantenmenge mit den beobachteten Relationen zwischen den Referenzen korrespondiert. Sei also $X = \{x_1, x_2, ..., x_{|X|}\}$ die Referenzmenge und $\mathcal{R} = \{R_1, R_2, ..., R_m\}$ eine Menge von zweistelligen Relationen $R_i \subseteq X \times X$, die auf den Referenzen definiert sind. Der *Referenzgraph* ist ein ungerichteter ARG $G = (V, E, A_V, A_E, \alpha_V, \alpha_E)$ der wie folgt konstruiert wird: Für jede Referenz $x \in X$ führen wir einen Knoten ein, d.h. $V = X$, und für jedes Paar von Referenzen mit $(x_1, x_2) \in R_i$ fügen wir in dem ARG eine Kante (x_1, x_2) hinzu und versehen die Kante aus der i-ten Relation mit dem Attribut a_i, d.h. $E = R_1 \cup R_2 \cup ... \cup R_m$ und $A_E = \{a_1, ..., a_m\}$. Das Ziel der *graphbasierten Objektkonsolidierung* besteht nun darin, den eindeutigen minimalen *Entitätengraphen* zu identifizieren, bei dem alle zu einer Entität korrespondierenden Referenzen in einem Knoten konsolidiert sind. Die Transformation vom *Referenzgraphen* zum *Entitätengraphen* kann mit der oben eingeführten Graphenoperation der Kontraktion einer Knotenmenge beschrieben werden. In einfachen Worten ausgedrückt heißt das, dass wir eine Menge zusammengehörender Referenzen – Referenzen einer Entität – unter Berücksichtigung der zu diesen inzidenten Kanten zu einem neuen Knoten „verschmelzen“ , wobei erstere dann mit diesem neuen Knoten inzidieren. Wenn wir alle mit einer Entität korrespondierenden Referenzen kontrahieren, dann erhalten wir den gewünschten *Entitätengraphen*. Um dies zu konkretisieren sei also $E = \{e_1, ..., e_{|E|}\}$ die Entitätenmenge und $\mathcal{C} = \{C_1, C_2, ..., C_{|E|}\}$ die korrekte Clusterlösung mit den darin enthaltenen Referenzen. Der *Entitätengraph* geht aus dem *Referenzgraphen* durch sukzessive Anwendung der Kontraktionsoperation $G(C_i)$ auf den Clustern C_i hervor.

Beispiel 3.1 Betrachten wir hierzu das in der Abbildung 3.2 dargestellte Koautorennetzwerk, das wir in unserer Arbeit als fortlaufendes Beispiel verwenden wollen, um die auszuführenden Schritte des Algorithmus zu illustrieren. Die Abbildung 3.2(a) zeigt den fiktiven Referenzgraphen eines unbereinigten Koautorennetzwerks. Die zugrunde liegende Datenmenge besteht in diesem Fall aus fünf Publikationen mit insgesamt dreizehn

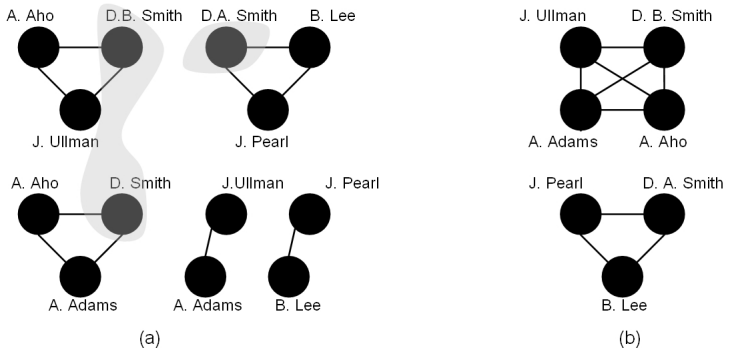


Abbildung 3.2.: (a) der *Referenzgraph* und (b) der *Entitätengraph* für das Konsolidierungsproblem in einem Koautorennetzwerk (weitere Erklärungen siehe Text)

Autorenreferenzen, die über die beobachtete Beziehung der Koautorenschaft miteinander verbunden sind. Wir wollen der Einfachheit halber annehmen, dass sich die nicht schattierten Referenzen anhand ihrer Namen korrekt identifizieren lassen und somit einfach zu konsolidieren sind. Ausgehend von dieser Annahme können wir also die Referenzen mit demselben Autorennamen in einem Cluster zusammenfassen und die entstehenden Clusters zu jeweils einem neuen Knoten kontrahieren. Dagegen ist die Konsolidierung der schattierten Autorenreferenzen nicht auf den ersten Blick ersichtlich, da wir nicht in der Lage sind diese allein anhand der Autorennamen zu disambiguieren. Wie aus der Schattierung ersichtlich ist, korrespondieren in diesem Beispiel die beiden Autorennamen „D. B. Smith“ und „D. Smith“ mit demselben Autor. Abbildung 3.2(b) zeigt den resultierenden Entitätengraphen mit den bereits aufgelösten Referenzen. So wurden beispielsweise die schattierten Knoten „D. B. Smith“ und „D. Smith“ zu einem Knoten kontrahiert.

3.3.3. Beschreibung des Konsolidierungsframeworks

Nachdem wir nun alle Konzepte und Notationen eingeführt haben, wollen wir das in [CKM05] entwickelte Konsolidierungsframework vorstellen, das zur Konsolidierung von Objekten sowohl die Attributinformationen der Referenzen als auch die in der Datenmenge beobachteten Objektbeziehungen verwendet. Das Framework erwartet dazu als Eingabe einen attributierten relationalen *Referenzgraphen*, der auf dieselbe Weise wie bei der *graphbasierten Objektkonsolidierung* konstruiert wird. Dazu wollen wir wieder davon ausgehen, dass wir eine Referenzmenge $X = \{x_1, x_2, ..., x_{|X|}\}$ vorliegen haben, auf denen eine Menge $\mathcal{R} = \{R_1, R_2, ..., R_m\}$ von zweistelligen Relationen $R_i \subseteq X \times X$, definiert sind. Für die Eingabe des Frameworks ist jedoch noch eine wichtige Erweiterung des *Referenzgraphens* erforderlich, die wir bereits bei der Kurzbeschreibung des Frameworks in der Einführung angedeutet haben. Wir haben die erste Phase damit beschrieben, dass wir zu Beginn die Attributinformationen verwenden wollen, um die Referenzen zu identifizieren, die auf Grund ihrer Ähnlichkeit zueinander potentiell die-

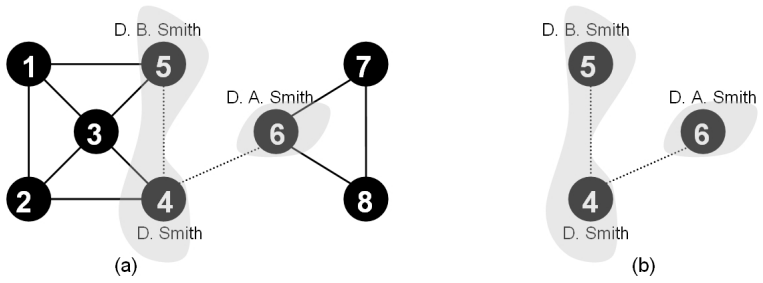


Abbildung 3.3.: (a) *Erweiterter Referenzgraph* und (b) *virtueller Konsolidierungsteilgraph*(weitere Erklärungen siehe Text)

selbe Entität repräsentieren. Wir wollen dies konkretisieren und die dahinterstehende Intention erklären. Zur Bestimmung der Attributähnlichkeit wollen wir annehmen, dass wir über eine Ähnlichkeitsfunktion $sim_{attr}(\cdot, \cdot)$ verfügen, die uns für zwei Referenzen einen Ähnlichkeitswert im Intervall $[0, 1]$ liefert, wobei ein höherer Wert auf eine größere Ähnlichkeit hinweisen soll. Diese Ähnlichkeitsinformation der Referenzen wird dann zur Erweiterung des *Referenzgraphens* verwendet, indem für jedes Referenzpaar mit einem Ähnlichkeitswert über einem vorgegebenen Schwellenwert τ, eine Ähnlichkeitskante in dem *Referenzgraphen* hinzugefügt wird. Der Schwellenwert steht dabei zur Parametrisierung offen und erfordert unter Umständen das Wissen über die Domäne. Wir bezeichnen diesen Graphen als *erweiterten Referenzgraphen* (siehe dazu Abbildung 3.3(a)).

Dies ist die zentrale Idee, die dem Konsolidierungsalgorithmus zugrunde liegt, denn durch diese Maßnahme werden die Referenzen über Relationen miteinander in Beziehung gesetzt, wodurch sich Aussagen über die Konnektionsstärke der Referenzen gewinnen lassen, die zur weiteren Disambiguierung der Referenzen verwendet werden können. Die Bestimmung der Konnektionsstärke erfolgt in der zweiten Phase des Konsolidierungframeworks, in der die existierenden Verbindungen zwischen den ähnlichen Referenzen im *erweiterten Referenzgraphen* analysiert werden, um die Konnektionsstärke zwischen diesen zu messen. In der Abschlussphase werden dann die Referenzen in Clusters partitioniert, so dass bezüglich der in der Vorphase errechneten Konnektionsstärke die Intracluster-Kohäsion maximiert und die Intercluster-Kohäsion minimiert wird. Dabei erfolgt die Partitionierung nicht auf den gesamten *erweiterten Referenzgraphen*, sondern wir können uns auf die Zusammenhangskomponenten beschränken, die von den Ähnlichkeitskanten induziert werden (siehe dazu Abbildung 3.3(b)). Betrachten wir dies etwas genauer. Dazu wollen wir uns eine Clustermenge C_{x_i} definieren, das die Menge aller Referenzen enthält, die dieselbe Entität wie x_i repräsentieren und demnach von der Resolutionsfunktion auf die selbe Entität abgebildet wird: $C_{x_i} = \{x_j \in X \,|r(x_i) = r(x_j)\}$. Da wir die Resolutionsfunktion nicht kennen, ist auch diese Clustermenge eine Unbekannte und deren Bestimmung das Ziel des Konsolidierungsalgorithmus. Zusätzlich definieren wir uns eine Konsolidierungsmenge S_{x_i}, die aus allen Referenzen besteht, die ausschließlich über Ähnlichkeitskanten von x_i aus zu erreichen sind: $S_{x_i} = \{x_j \in X \,|\exists\, [x_i, x_j] - \text{Pfad, der nur aus Ähnlichkeitskanten besteht}\}$. Den von ei-

Referenzen	C	S
$x =$ „D. Smith“	$C_x = \{x, x_B\}$	$S_x = \{x, x_A, x_B\}$
$x_A =$ „D. A. Smith“	$C_{x_A} = \{x_A\}$	$S_{x_A} = \{x, x_A, x_B\}$
$x_B =$ „D. B. Smith“	$C_{x_B} = \{x, x_B\}$	$S_{x_B} = \{x, x_A, x_B\}$

Tabelle 3.1.: Clustermenge C und Konsolidierungsmenge S

ner Konsolidierungsmenge induzierten Teilgraphen wollen wir als *virtuellen*[3] *Konsolidierungsteilgraphen* bezeichnen und treffen dabei die folgende wichtige Annahme: Falls eine Referenz einer bestimmten Entität in einem *virtuellen Konsolidierungsuntergraphen* enthalten ist, dann sind alle Referenzen derselben Entität ebenfalls in diesem Untergraphen enthalten, d.h. es gilt $C_{x_i} \subseteq S_{x_i}$. Diese Annahme stellt eine wesentliche Vereinfachung für den Entwurf des Algorithmus dar, da wir die Konsolidierung von Referenzen einer Entität auf einen einzelnen *virtuellen Konsolidierungsteilgraphen* beschränken können. Andererseits stellt diese Annahme auch eine Einschränkung dar, denn es kann der pathologische Fall auftreten, dass zwei Referenzen derselben Entität aufgrund ihrer Unähnlichkeit unterschiedlichen *Konsolidierungsteilgraphen* zugewiesen werden, obwohl die beiden Referenzen eine hohe Konnektionsstärke aufweisen. Der Konsolidierungsalgorithmus ist jedoch nicht in der Lage, derartige Situationen aufzulösen, um diese mit derselben Entität korrespondierenden Referenzen in denselben Cluster zu konsolidieren.

Die Abbildung 3.3 zeigt in (a) den *erweiterten Referenzgraphen* mit der durch die Koautorenschaft (durchgezogenen Linien) verbundenen Referenzen und in (b) den durch die Ähnlichkeitskanten (gebrochenen Linien) induzierten *virtuellen Konsolidierungsteilgraphen*. Die Tabelle 3.1 illustriert die verwendeten Bezeichnungen anhand unseres Beispiels aus Abbildung 3.3. Während wir uns bei der Partitionierung der Referenzen auf die einzelnen *Konsolidierungsteilgraphen* beschränken können, müssen wir hingegen zur Bestimmung der Konnektionsstärke zwischen ähnlichen Referenzen den gesamten *erweiterten Referenzgraphen* mit einbeziehen.

Das Konsolidierungsframework (Chen, Kalashnikov, Mehrotra)

1. **Konstruiere den *erweiterten Referenzgraphen*.** Der erste Schritt besteht darin für die betreffende Datenmenge den *erweiterten Referenzgraphen* zu konstruieren. Zur Konstruktion des Graphens setzen wir dabei eine attributbasierte Ähnlichkeitsfunktion $sim_{attr}(\cdot, \cdot)$ und einen vorgegeben Schwellenwert τ voraus. Identifiziere alle *virtuellen Konsolidierungsteilgraphen*.

2. **Wähle einen *virtuellen Konsolidierungsteilgraphen* und bestimme paarweise die Konnektionsstärke.** Wähle aus dem *erweiterten Referenzgra-*

[3] Mit dem Begriff virtuell wollen wir zum Ausdruck bringen, dass der Konsolidierungsteilgraph nicht dem tatsächlichen Entitätencluster entspricht, sondern es einer weiteren Disambiguierung bedarf.

phen den nächsten zu partitionierenden *Konsolidierungsteilgraphen* aus. Anschließend wird in diesem *virtuellen Konsolidierungsteilgraphen* paarweise die Konnektionsstärke für alle Referenzen u, v berechnet, die durch eine Ähnlichkeitskante miteinander verbunden sind.

3. **Partitionierung des *virtuellen Konsolidierungsteilgraphens***. Verwende einen Algorithmus zur Partitionierung von Graphen, um die Referenzen in den *virtuellen Konsolidierungsteilgraphen* basierend auf der gemessenen Konnektionsstärke in Clusters zu partitionieren. Nachdem der *Konsolidierungsteilgraph* partitioniert wurde, aktualisiere den *erweiterten Referenzgraphen* entsprechend der gefundenen Clusterlösung $\mathcal{C} = \{C_1, C_2, ..., C_m\}$. Führe dazu für alle $C_i \in \mathcal{C}$ die Kontraktion $G(C_i)$ durch. Falls noch weitere *virtuelle Konsolidierungsteilgraphen* vorhanden sind gehe nach Schritt 2, ansonsten Stop.

3.4. Phase 2: Bestimmung der Konnektionsstärke

In diesem Abschnitt stellen wir zwei Modelle vor, mit denen wir paarweise die Konnektionsstärke der Referenzen in den *virtuellen Konsolidierungsteilgraphen* bestimmen wollen. Mit der Konnektionsstärke beziehen wir uns dabei auf den Grad der Beziehungen, die zwei Referenzen eingehen und die Aufschluss darüber geben soll, welche Referenzen mit derselben Entität korrespondieren. Zur Erinnerung: Die *virtuellen Konsolidierungsteilgraphen* werden in der ersten Phase des Frameworks identifiziert, wobei diese aus den durch die Ähnlichkeitskanten induzierten Zusammenhangskomponenten des *erweiterten Referenzgraphens* hervorgehen. Diesbezüglich haben wir folgende wichtige Annahme getroffen: Wenn die Referenz einer Entität in einem *virtuellen Konsolidierungsteilgraphen* enthalten ist, dann sind alle Referenzen derselben Entität ebenfalls in diesem Teilgraphen enthalten. Diese Annahme stellt eine wesentliche Vereinfachung für den Entwurf der Modelle dar, da wir die Konsolidierung der Referenzen einer Entität auf einen *virtuellen Konsolidierungsteilgraphen* beschränken können und zur Bestimmung der Konnektionsstärke lediglich die zueinander ähnlichen Referenzen (Referenzen die mit einer Ähnlichkeitskante inzidieren) zu berücksichtigen haben. Betrachten wir dazu unser Beispiel aus Abbildung 3.3. Im Prinzip sind wir nur daran interessiert, die Ähnlichkeitskanten des *virtuellen Konsolidierungteilgraphens* zu bewerten, d.h. die Knotenpaare (v_4, v_5) und (v_4, v_6). Um für diese Kanten jedoch die Konnektionsstärke zu messen, müssen wir den gesamten *erweiterten Referenzgraphen* mit einbeziehen, da sich erst bei Betrachtung von diesem eine Aussage über die Konnektionsstärke anhand der Beziehungen gewinnen lässt. Die paarweise Analyse der zueinander ähnlichen Referenzen kann im Grunde als Versuch verstanden werden, das *Kontext-Attraktions-Prinzip* in ein quantifizierbares Modell zu gießen. Gehen wir von dieser Hypothese aus, dann sollte ein Modell zur Berechnung der Konnektionsstärke dazu in der Lage sein, den Referenzen, die dieselbe Entität repräsentieren, eine höhere Konnektionsstärke zuzuweisen als Referenzen unterschiedlicher Entitäten. Zur Konsolidierung der Referenzen werden dann in einer abschließenden dritten Phase die *virtuellen Konsolidierungsteilgraphen* basierend

auf der Konnektionsstärke partitioniert, so dass idealerweise diejenigen Referenzen, die mit der derselben Entität korrespondieren, in denselben Cluster fallen.

Der erste Algorithmus den wir vorstellen wollen, ist das von Chen et al. verwendete Modell zur Bestimmung der Konnektionsstärke, die wir in unserer Arbeit auch als relative Zentralität oder Wichtigkeit bezeichnen wollen. Das Modell basiert auf einem graphentheoretischen Ansatz in der die kürzesten Wege verschiedener Länge zwischen zwei Knotenpaaren betrachtet werden. Dieses Verfahren ist im weitesten Sinne mit der *Closeness* Zentralität aus Abschnitt 2.3.1 verwandt. Mit dem zweiten Algorithmus, den wir in unserer Arbeit vorschlagen, verfolgen wir einen konzeptuell anderen Ansatz, bei der wir den Graphen – ähnlich wie im PageRank Algorithmus 2.3.3 – als stochastischen Prozess auffassen und als *Markovkette* modellieren. In Hinblick auf die erfolgreiche Anwendung des *PageRank* Algorithmus auf den Hyperlink-Graphen streben wir analog dazu an, einen modifizierten *PageRank* Algorithmus auf den *erweiterten Referenzgraphen* anzuwenden, um ein relatives Ranking zu produzieren. Wenn wir von der Modifikation des *PageRanks* sprechen, beziehen wir uns dabei weniger auf die Modifikation des Algorithmus selbst, sondern vielmehr auf die Interpretation des Modells an sich. Diesbezüglich besteht die Modifikation zum einem in einer personalisierten Variante des *PageRank* Modells und zum anderen soll in unserem Modell, im Gegensatz zum ursprünglichen *PageRank*, die Gewichtungen der Kanten berücksichtigt werden. Dazu wollen wir analog zum *PageRank* die *Markov*-Theorie zur theoretischen Rechtfertigung heranziehen, um zu zeigen, dass das Verfahren auch in unserem modifizierten Modell sinnvolle Resultate liefert. Bevor wir aber auf die eigentlichen Modelle zu sprechen kommen, wollen wir im nächsten Abschnitt zunächst einmal versuchen den Begriff der Konnektionsstärke im Kontext der *sozialen Netzwerkanalyse* zu charakterisieren.

3.4.1. Konnektionsstärke als relative Zentralität

Da wir in unserer Arbeit darauf fokussiert sind, wie sich Methoden der *sozialen Netzwerkanalyse* auf die einzelnen Phasen des Konsolidierungsalgorithmus anwenden lassen, wollen wir zunächst den Begriff der Konnektionsstärke im Kontext der *Netzwerkanalyse* neu definieren. Einen ersten Anhaltspunkt liefern uns dazu die in Abschnitt 2.3 eingeführten Zentralitätsindizes, die die Prominenz oder auch die Wichtigkeit eines Akteurs in einem Netzwerk charakterisieren. Bei diesen Indizes liegt der Fokus darin, in einer Menge von Akteuren den zentralen – oder äquivalent dazu den prominenten – Akteur zu bestimmen und in der Erstellung eines Rankings, in der jeder Akteur relativ zu allen anderen bezüglich seiner Prominenz geordnet wird. Mit der Konnektionsstärke beziehen wir uns auf ein verwandtes, aber anderes Problem und zwar die Bestimmung der relativen Wichtigkeit der Akteure in einem Netzwerk bezüglich eines ausgezeichneten Akteurs. Dieses Problem zur Bestimmung der „relativen Zentralität“ wird von White et al. in [WS03] untersucht. Ein erster, nahe liegender Ansatz zur Lösung dieses modifizierten Problems stellt die Verwendung einer „globalen“ Methode dar, in der zunächst – ähnlich wie im HITS-Algorithmus 2.3.3 – ein dem Hauptakteur umgebender fokussierter Untergraph konstruiert wird. Anschließend wird dann mit der „globalen“

Methode ein Ranking für alle Akteure in diesem fokussierten Untergraphen produziert. In diesem Ansatz besteht jedoch das Problem, dass dem Hauptakteur keine bevorzugte Behandlung im Ranking beigemessen wird. In Wirklichkeit werden nämlich die Akteure in einem dem Hauptakteur umgebenden lokalen Untergraphen geranked, anstatt diese relativ zum Hauptakteur selbst zu ranken. In unserem Problem sollte der Hauptakteur vielmehr eine gesonderte Stellung einnehmen, die in irgendeiner Form in das Ranking mit einfließt. Wir sind also an folgender konditionellen Problemstellung interessiert: Unter der Annahme, dass ein individueller Hauptakteur a priori als wichtig angenommen wird, produziere ein Ranking der restlichen Knoten nach der relativen Wichtigkeit bezüglich dieses ausgezeichneten Akteurs. Das Problem der Bestimmung der relativen Wichtigkeit lässt sich dann wie folgt formulieren:

Definition 3.2 (Problem der relativen Wichtigkeit): Gegeben sei ein Graph $G = (V, E)$, eine Teilmenge von Knoten $U \subseteq V$ und ein ausgezeichneter Hauptknoten $r \in V$. Bezeichne $I(u\,|r)$ die relative Wichtigkeit des Knotens u bezüglich des Hauptknotens r. Das Problem der relativen Wichtigkeit besteht darin die Werte $I(u\,|r)$ für alle $u \in U$ zu bestimmen und anhand dieser Werte ein Ranking der Knoten bezüglich r zu produzieren.

Im Weiteren wollen wir diese Problemformulierung der relativen Wichtigkeit mit der Bestimmung der Konnektionsstärke gleichsetzen und beim Aufstellen der Modelle der obigen Notation folgen. Halten wir kurz inne und überlegen, welche Anforderungen an ein solches Modell zu stellen sind, so dass das *Kontext-Attraktions-Prinzip* möglichst gut durch dieses abgebildet wird. Das *Kontext-Attraktions-Prinzip* besagt, dass wenn zwei Referenzen dieselbe Entität repräsentieren, es dann wahrscheinlich ist, dass diese beiden Referenzen über weiteren expliziten sowie impliziten Beziehungen miteinander verbunden sind. Zur Bestimmung der Konnektionsstärke spielen demnach die direkten und indirekten Wege im *erweiterten Referenzgraphen* zwischen den Referenzen eine entscheidende Rolle, da genau diese den Grad der Beziehungen reflektieren. Ein Modell sollte also folgende zwei Faktoren bei der Bestimmung der Konnektionsstärke berücksichtigen: Zum einen sollten Referenzen, die über viele redundante Wege zu erreichen sind, eine hohe Konnektionsstärke zugewiesen bekommen. Zum anderen sollte in diesem Maß auch die Weglänge an sich berücksichtigt werden, so dass längere Wege zwischen den Referenzen mit einem Strafparameter versehen werden. Eine weitere wünschenswerte Eigenschaft eines Modells ist die Symmetrie der relativen Wichtigkeit $I(u\,|v)$, d.h. es sollte $I(u\,|v) = I(v\,|u)$ gelten, da dies andernfalls unserer intuitiven Vorstellung widersprechen würde. Einen in diesem Zusammenhang wichtigen Punkt haben wir jedoch bisher außen vor gelassen: Wie sollen die unterschiedlichen Kantenattribute bei der Bestimmung der Konnektionsstärke gewichtet werden? Auf diesen Aspekt wollen wir jedoch erst bei Betrachtung der Modelle näher eingehen.

3.4.2. Chen's Vorschlag: Alle L-kürzeste Wege

Chen et al. verwenden zur Bestimmung der Konnektionsstärke in ihrer Arbeit einen graphentheoretischen Ansatz, in der die kürzesten Wege zwischen den Knoten eine zentrale

Rolle einnehmen. Wie wir bereits angemerkt haben, existieren im Allgemeinen mehrere mögliche Verbindungen zwischen zwei Referenzen, von denen viele – insbesondere die langen Wege – von marginaler Bedeutung sind. Um nun die wichtigsten Verbindungen zu erfassen und dennoch effizient zu bleiben, werden zur Bestimmung der Konnektionsstärke lediglich die Menge aller *L-kürzesten Wege* $\mathcal{P}_L(u, v)$ zwischen zwei Knoten (Referenzen) u und v im Graphen herangezogen, wobei ein Weg als *L-kurz* gilt, falls dessen Wegdistanz L nicht übersteigt. Die Prozedur zur Berechnung der Konnektionsstärke $I(u\,|v\,)$ setzt sich dabei aus zwei logischen Phasen zusammen. In der ersten Phase werden die kürzesten Wege zwischen einem Knoten u und dem Hauptknoten v identifiziert. In einer anschließenden zweiten Phase wird die totale Konnektionsstärke in den Verbindungen berechnet, die sich aus der Summe der in der ersten Phase gefundenen Wege zusammensetzt.

$$I(u\,|v\,) = \sum_{p \in P_L(v,u)} c(p) \tag{3.2}$$

Berechnung der Konnektionsstärke von Wegen

Wir müssen noch spezifizieren, wie die Konnektionsstärke eines individuellen Weges p aus $\mathcal{P}_L(u, v)$ zu berechnen ist. Dazu wollen wir zunächst einmal annehmen, dass sich ein Pfad nur aus den Relationskanten ohne Ähnlichkeitskanten zusammensetzt. Jede Kante ist mit einem Attribut versehen, welches Aufschluss darüber gibt, um welcher Relation es sich bei der Kante handelt. In unserem Beispiel zur Konsolidierung von Autorenreferenzen haben wir beispielsweise mit der Beziehung der Koautorenschaft nur eine Relation vorliegen. In vielen Datenmengen stehen jedoch die Referenzen über mehrere Relationen in Beziehung, sodass wir von diesem allgemeinen Fall ausgehen wollen. Wir können nun den Kanten entsprechend ihrer Relationszugehörigkeit ein Gewicht w_{a_i} (eine reele Zahl im Intervall $[0, 1]$) zuordnen. Wir werden uns am Ende zu diesem Abschnitt noch kurz mit der Frage befassen, wie wir diese Gewichte bestimmen können. Die Konnektionsstärke eines Pfades p ergibt sich dann aus dem Produkt der mit den Kantenattributen assoziierten Gewichten. Als Beispiel wollen wir annehmen, dass ein Pfad p aus n_1 Kanten des Attributs a_1, n_2 Kanten des Attribut a_2 usw. besteht, dann berechnet sich $c(p)$ auf folgende Weise:

$$c(p) = w_{a_1}^{n_1} \cdot w_{a_2}^{n_2} \cdot \ldots \cdot w_{a_m}^{n_m} \tag{3.3}$$

Es sei darauf hingewiesen, dass sich die zweite Anforderung an das Modell – längere Wege zu bestrafen – aus der Multiplikation der $[0, 1]$- Gewichte ergibt, sodass für längere Pfade eine kleinere Konnektionsstärke zu erwarten ist.

Betrachten wir nun den zweiten Fall, dass sich ein Pfad sowohl aus Relationskanten als auch Ähnlichkeitskanten zusammensetzt. Zur Erinnerung: Die Ähnlichkeitskante

Algorithmus 3.1 Alle L-Kürzeste Wege

Input: ARG G, Startknoten $s \in V$, Zielknoten $t \in V$, Länge L
Output: Menge aller L-Kürzesten Wege $\mathcal{P}_L(s,t)$

```
1: Initialisiere P_L ← ∅
2: Push(S, t)
3: while S ≠ ∅ do
4:     p ← Pop(S)
5:     if letzterKnoten(p) = t then
6:         P ← P ∪ p
7:     else if Länge (p) < L then
8:         Expandiere-Weg(p, S)
9:     end if
10:    return P
11: end while
12:
13:
14: Expandiere-Weg(p, S)
15: v ← letzterKnoten(p)
16: for all Knoten u ∈ V mit (v, u) ∈ E do
17:    if u ∉ p then
18:        Push(S, p → u)
19:    end if
20: end for
```

zwischen Knoten zweier Referenzen x_i und x_j beschreibt die Tatsache, dass die beiden Referenzen möglicherweise dieselbe Entität repräsentieren. Mit den Ähnlichkeitskanten ist also ein gewisses Maß an Unsicherheit verbunden, da wir nicht mit absoluter Bestimmtheit sagen können, ob es sich tatsächlich, um die Referenzen ein und derselben Entität handelt. Dies wirft das Problem auf, dass Pfade, die Ähnlichkeitskanten enthalten, in Realität überhaupt nicht existieren müssen, mit der Konsequenz, dass die Konnektionsstärke zwischen zwei Referenzen verfälscht wird. Andererseits stellen insbesondere die Ähnlichkeitskanten wichtige Verbindungen zu den Referenzen her, so dass wir auf diese nicht verzichten wollen. Um einen guten Mittelweg zu finden, besteht eine Möglichkeit darin, für alle Ähnlichkeitskanten ein kleines Basisgewicht w_ε zu assoziieren. Das Totalgewicht v_i einer Ähnlichkeitskante ergibt sich dann aus dem Produkt des Basisgewichts und dem von der Ähnlichkeitsfunktion $sim_{attr}(\cdot,\cdot)$ berechneten Ähnlichkeitswert: $v_i = w_\varepsilon \cdot sim_{attr}(x_i, x_j)$. Falls also ein Pfad p aus n_i Kanten des Attributs a_i und zusätzlich k Ähnlichkeitskanten mit dem Totalgewicht $v_1, ..., v_k$ enthält, dann berechnet sich $c(p)$ auf folgende Weise:

$$c(p) = w_{a_1}^{n_1} \cdot w_{a_2}^{n_2} \cdot ... \cdot w_{a_m}^{n_m} \cdot v_1 \cdot v_2 \cdot ... \cdot v_k \tag{3.4}$$

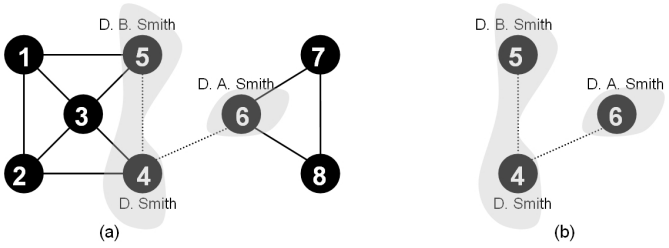


Abbildung 3.4.: (a) *erweiterter Referenzgraph* und (b) *virtueller Konsolidierungsteilgraph*

	Werte
$I(v_1 \mid v_4)$	4.707
$I(v_2 \mid v_4)$	3.403
$I(v_3 \mid v_4)$	3.339
$I(v_4 \mid v_4)$	1.0
$I(v_5 \mid v_4)$	*5.065*
$I(v_6 \mid v_4)$	*0.1*
$I(v_7 \mid v_4)$	0.171
$I(v_8 \mid v_4)$	0.171

Tabelle 3.2.: Chen's Modell der *L-Kürzesten Wege* zur Bestimmung der Konnektionsstärke mit $L = 5$

Beispiel 3.3 Betrachten wir hierzu das Beispiel aus Abbildung 3.4. Wir wollen dazu der Einfachheit halber annehmen, dass die Kantengewichte von einem Domänenanalysten vorgegeben werden, wobei wir mit der Koautorenschaftsbeziehung ein Gewicht $w_{a_1} = 0.9$ und mit den Ähnlichkeitskanten ein Gewicht $v_i = 0.1$ assoziieren. Da wir an der Konnektionsstärke der Kante (v_4, v_5) und (v_4, v_6) interessiert sind, wollen wir an diesen exemplarisch die Konnektionsstärke $I(v_5 \mid v_4)$ und $I(v_6 \mid v_4)$ mit dem oben eingeführten Modell der *L-kürzesten Wege* bestimmen, wobei wir den Parameter $L = 5$ wählen. Identifizieren wir also zunächst die *L-kürzesten Wege* von $\mathcal{P}_5(v_4, v_5)$ und $\mathcal{P}_5(v_4, v_6)$.

$$P_5(v_4, v_5) = \{(v_4, v_5), (v_4, v_3, v_5), (v_4, v_3, v_1, v_5), (v_4, v_2, v_3, v_5), (v_4, v_2, v_1, v_5), (v_4, v_2, v_3, v_1, v_5), (v_4, v_3, v_2, v_1, v_5), (v_4, v_2, v_1, v_3, v_5)\}$$

$$P_5(v_4, v_6) = \{(v_4, v_6)\}$$

Die Konnektionsstärke für diese beiden Knotenpaare berechnet sich dann mit Gleichung 3.2 wie folgt:

$$\begin{aligned} I(v_5 \mid v_4) &= \sum_{p \in P_5(v_4, v_5)} c(p) \\ &= 0.1 + 0.9^2 + 3 \cdot 0.9^3 + 3 \cdot 0.9^4 \\ &= 5.065 \end{aligned}$$

$$\begin{aligned} I(v_6 \mid v_4) &= \sum_{p \in P_5(v_4, v_6)} c(p) \\ &= 0.1 \end{aligned}$$

Laufzeitbetrachtung

3.4.3. Unser Vorschlag: Personalisierter PageRank

In Abschnitt 2.3.3 haben wir mit dem *PageRank* Algorithmus ein Verfahren kennen gelernt, das unter Ausnutzung der Hyperlink-Struktur des Webs die Webseiten hinsichtlich ihrer „globalen“ Wichtigkeit beurteilt. Zur Anfragezeit werden diese Wichtigkeitswerte dann in Verbindung mit den relevanten Seiten – die als Antwort auf eine Suchanfrage gefundenen Dokumente – verwendet, um die Suchergebnisse hinsichtlich ihrer globalen Wichtigkeit bzw. ihrer Autorität zu ranken. Der *PageRank* Algorithmus unterliegt dabei der Annahme, dass die als global wichtig erachteten Seiten, auch von einem individuellen Webbenutzer als wichtig erachtet werden. Allerdings ist die Wichtigkeit einer Seite ein rein subjektiver Begriff, deren Bewertung unter anderem von den spezifischen Interessen und Kenntnissen eines Webbenutzers abhängen kann. Im Grunde handelt es sich bei dem auf diese Weise produzierten Ranking um den Versuch die Subjektivität, die mit dem Begriff der Wichtigkeit einhergeht, über die Autorität einer Seite zu charakterisieren. In diesem Ranking Verfahren wird also die objektive Wichtigkeit der Webseiten widergespiegelt im Sinne, dass zur Bestimmung des *PageRank* Vektors die subjektiven Interessen des Nutzers keinen Einfluss auf die Bewertung einer Seite hinsichtlich ihrer Wichtigkeit nehmen. Zudem hat eine gestellte Suchanfrage keinen Einfluss auf die Berechnung der Relevanzwerte. In Wirklichkeit besteht jedoch aus der Sicht eines Benutzers ein Interesse an einer Personalisierung des Webs, in der die vom Benutzer als wichtig erachteten Kategorien einen Vorzug im Ranking genießen. So könnte man beispielsweise die persönlichen Bookmarks als Quellen für den personalisierten *PageRank* verwenden. Die Idee eines personalisierten *PageRanks* wurde erstmals in der Arbeit von [PBMW98] erwähnt, ohne diesen jedoch näher zu spezifzieren. Haveliwala [Hav02] und Jeh und Widom [JW02] haben diese Idee in ihren Arbeiten aufgegriffen und ein personalisierte Variante des *PageRanks* entwickelt.

Im nächsten Abschnitt wollen wir zunächst einmal das Modell von Haveliwala zur Bestimmung eines themenspezifischen *PageRanks* vorstellen. Ausgehend von diesem Modell wollen wir aufzeigen, wie der *PageRank* Algorithmus zur Bestimmung der relativen Wichtigkeit in unserem Framework verwendet werden kann. Dabei müssen wir beachten, dass wir im Gegensatz zum Hyperlink-Graphen als gerichteten, ungewichteten Graphen, mit dem *erweiterten Referenzgrapen* einen ungerichteten Graphen mit einer auf den Kanten definierten Gewichtsfunktion vorliegen haben. Diesbezüglich müssen wir uns überlegen, wie der PageRank Algorithmus auf diese neue Situation angepasst werden kann. Dazu wollen wir die beiden folgenden ausstehenden Fragestellungen untersuchen:

- Arbeitet der Algorithmus zur Bestimmung des PageRanks auch in einem ungerichteten Graphen korrekt?
- Wie kann der PageRank Algorithmus auf gewichtete Graphen generalisiert werden?

Themenspezifischer PageRank

Im ursprünglichen *PageRank* Modell wird unabhängig von einer spezifischen Suchanfrage einmalig ein einzelner *PageRank* Vektor offline berechnet und dieser dazu verwendet, die als Antwort auf eine Suchanfrage gefundenen Dokumente hinsichtlich ihrer Autorität zu ranken. Diesbezüglich erweist sich das Verfahren von Page und Brin im Vergleich zum Verfahren von Kleinberg, dem *HITS*-Algorithmus, als effizienter, da im Letzteren die Berechnung der Autoritätswerte online stattfindet. Nichtsdestotrotz hat das Verfahren von Kleinberg den entscheidenden Vorteil, dass die Autoritätsbewertung in einem auf die Suchanfrage fokussierten Untergraphen stattfindet, wodurch die subjektive Wichtigkeit in Bezug auf eine Suchanfrage akkurater reflektiert wird als dies im *PageRank* Verfahren der Fall ist. Um auch für das *PageRank* Verfahren ein themenspezifisches Ranking zu erzielen, das mehr auf die individuellen Belange der Benutzer ausgerichtet ist, hat Haveliwala eine personalisierte Variante des *PageRank* Modells vorgeschlagen. In diesem personalisierten Modell werden eine Menge von *PageRank* Vektoren mit einem Bias für diverse Themengebiete (wie z.B. Sport oder Wirtschaft) berechnet, so dass mit jeder Seite eine Anzahl an unterschiedlichen Autoritätswerten – für jedes Themengebiet ein Wert – assoziiert wird. Für dieses „Biasing“ bedient sich Haveliwala einer kleineren Auswahl an repräsentativen Themengebieten aus dem *Open Directory Project*[4], wobei für jedes dieser Themengebiete ein themenspezifischer *PageRank* generiert wird. Anstatt nun einen einzelnen globalen Vektor für das Ranking heranzuziehen, werden Linearkombinationen dieser themen-sensitiven Vektoren genommen, die bezüglich der Ähnlichkeiten der Suchanfrage zu den einzelnen Kategorien gewichtet werden. Dabei ergibt sich der Ähnlichkeitswert aus einer Abschätzung der Klassenwahrscheinlichkeiten der Suchanfrage für jedes der repräsentativen Themengebiete, wofür ein Textklassifzier, wie beispielsweise der *Naive Bayes Klassifizierer* [Cha02], eingesetzt werden kann. Durch die Einbeziehung einer Menge von PageRank Vektoren verschiedener Kategorien wird gewährleistet, dass bei der Bestimmung der Wichtigkeitswerte die in Zusammenhang mit einer Suchanfrage stehenden Themen einen Vorzug im Ranking genießen. Auf die gleiche Weise wie beim ursprünglichen PageRank, findet die Berechnung der *PageRank* Vektoren offline statt, so dass die Antwortzeit einer Suchanfrage im Vergleich zum normalen *PageRank* nur minimal beeinträchtigt wird. Mit dem *PageRank* werden somit die Vorteile der beiden Rankingverfahren, die Einbeziehung der Suchanfrage im *HITS*-Algorithmus und die kurze Antwortzeit bei einer Suchanfrage im *PageRank* Algorithmus, auf eine einfache und effiziente Weise kombiniert.

Um zu zeigen, wie ein solcher Bias für spezifische Kategorien in die Berechnung des PageRank Vektors mit einfließt, wollen wir kurz die wesentlichen Erkenntnisse aus Abschnitt 2.3.3 in Bezug auf den PageRank wiederholen. Zum Korrektheitsbeweis des PageRank Algorithmus 2.1 haben wir den Hyperlink-Graphen als *Markovkette* aufgefasst, wodurch einige wichtige Resultate der Markov-Theorie für den Korrektheitsbeweis des PageRank Algorithmus zum Tragen kamen. Um die Existenz des PageRank Vektors und

[4] Das Open Directory Project (ODP), auch bekannt als DMoz (für „Directory at Mozilla“), ist mittlerweile ein von AOL zur Verfügung gestelltes umfangreiches Webverzeichnis, das von einer Gemeinschaft freiwilliger Editoren gepflegt wird. http://www.dmoz.org/.

die Konvergenz der Potenzmethode zu garantieren, haben Page und Brin den Hyperlink-Graphen einiger technischer Modifikationen unterzogen, indem sie die Übergangsmatrix dieses Graphens in eine *irreduzible*, *aperiodische Markovkette* modifiziert haben. Der PageRank Vektor lässt sich dann als Lösung der folgenden iterativen Gleichung angeben:

$$r^{(t+1)} = dP'r^{(t)} + (1-d)v \tag{3.5}$$

wobei in der Regel für den Vektor v eine uniforme Wahrscheinlichkeitsverteilung $(1/n)_{n\times 1}$ angenommen haben. Dieser Vektor wird unter anderem als Teleportationsvektor bezeichnet, da dessen Elemente die Wahrscheinlichkeiten der Seiten angeben mit der diese als Ziel der Teleportation ausgewählt werden. Bei einer uniformen Wahrscheinlichkeitsverteilung ist das Ziel über alle Knoten gleich verteilt. Abgesehen von der technischen Funktionalität, die der Teleportationsvektor v innehat – die *Irreduzibilität* der *Markovkette* zu erzwingen – hat sich dieser Vektor auch als wichtiger Parameter zur Personalisierung des *PageRank* Vektors herausgestellt. Um einen personalisierten bzw. themenspezifischen *PageRank* zu berechnen hat Haveliwala die Idee aus [PBMW98] aufgegriffen, bei der anstelle eines uniformen Teleportationsvektors v eine nicht-uniforme Wahrscheinlichkeitsverteilung des Vektors angenommen wird, wodurch die Berechnung des *PageRank* Vektors hinsichtlich spezifischer Themengebiete beeinflusst wird. Mit diesem „biased“ *PageRank* gelangen wir zu einer modifizierten Interpretation des *Random Surfer Modells*: Anstatt sich mit einer uniformen Wahrscheinlichkeitsverteilung zu einem beliebigen anderen Knoten bzw. Kategorie im Graphen zu teleportieren, folgt der *Random Surfer* der nicht-uniformen Wahrscheinlichkeitsverteilung in v, in der die für den *Random Surfer* interessanten Kategorien eine höhere Wahrscheinlichkeit zugesprochen bekommen. Durch dieses Biasing bekommen also zum einen die mit den Kategorien korrespondierenden Knoten direkt einen höheren *PageRank* zugewiesen und zum anderen profitieren ebenfalls die in der Nachbarschaft dieser Kategorien liegenden Knoten davon. Um ein personalisiertes *PageRank* zu produzieren, muss also nur der Vektor v entsprechend hinsichtlich der Belange eines individuellen Benutzers modifiziert werden.

Um dies zu konkretisieren, betrachten wir hierzu die von Haveliwala vorgeschlagene Personalisierung etwas näher. Dazu wollen wir zunächst davon ausgehen, dass wir c_j Kategorien verwenden wollen, der jeweils eine Menge von T_j Seiten angehören, d.h. Seiten, die vom Inhalt her zur Kategorie c_j passen. Für jedes dieser Kategorien wird nun ein themenspezifischer *PageRank* Vektor berechnet, bei dem anstatt des uniformen Teleportationvektors folgender nicht-uniformer Personalisierungsvektor $v_{(j)}$ definiert wird:

$$v_{(j)i} = \begin{cases} 1/\left|T_j\right| & \text{falls Knoten } i \text{ der Kategorie } T_j \text{ angehört,} \\ 0 & \text{andernfalls.} \end{cases} \tag{3.6}$$

Mit den auf diese Weise definierten Personalisierungsvektoren sehen wir uns jedoch wieder dem Problem gegenüber, dass die *Irreduzibilität* nicht gewährleistet ist. Um die *Irreduzibilität* der *Markovkette* in allen Fällen zu wahren, müssen wir den Teleportationsvektor auf positive Werte beschränken, d.h. $v > 0$. Zu diesem Zweck können für die 0-Einträge des Vektors kleine Mindestwahrscheinlichkeiten angenommen werden.

Unser Modell zur Bestimmung der Konnektionsstärke

Bei der Betrachtung des *PageRank* Modells sind wir stets von einem ungewichteten Graphen ausgegangen. Dies hat sich auf der Tatsache begründet, dass die mit den Kanten korrespondierenden Hyperlinks – als Verweise zu anderen Dokumenten – selbst keine Gewichtung besitzen. Im Gegensatz zum Hyperlink-Graphen, haben wir im Kontext unserer Arbeit mit dem *erweiterten Referenzgraphen* einen ungerichteten, gewichteten Graphen vorliegen. Zur Erinnerung: Bei dem *erweiterten Referenzgraphen* handelt es sich um einen ungerichteten Graphen, dessen Kanten mit Attributen versehen sind, die zum einen aus den vorliegenden Relationen und zum anderen aus der Attributähnlichkeit zwischen den Referenzen hervorgehen. Wie bereits erläutert, wäre es sinnvoll mit den Attributen unterschiedliche Gewichte zu assoziieren. Diesbezüglich müssen wir uns überlegen, ob diese Gegebenheiten der Markov-Theorie einen Abbruch tun. Die Tatsache, dass es sich um einen ungerichteten Graph handelt, stellt für den *PageRank* Algorithmus kein Problem dar, da jede ungerichtete Kante als zwei gerichtete, symmetrische (bezüglich der Kantengewichtung) Kanten dargestellt werden kann. Vielmehr werden sich die ungerichteten Kanten in der weiteren Betrachtung unseres Modells als nützliche Eigenschaft erweisen. Wir wollen nun untersuchen, wie wir die Gewichtungen der Kanten in das *PageRank* Modell inkorporieren können. Zur intuitiven Rechtfertigung des *PageRank* Modells haben wir in dem *Random Surfer Modell* eine adäquate Interpretation gefunden, in welchem wir den *PageRank* als Wahrscheinlichkeit aufgefasst haben, dass der *Random Surfer* sich nach unendlich vielen Schritten auf eine bestimmte Seite befindet. Ausgehend von dieser Interpretation können wir den stochastischen Prozess nun dahingehend generalisieren, dass wir die Gewichtung der Kanten als Wahrscheinlichkeiten interpretieren, mit die der *Random Surfer* einen Nachfolgerknoten auswählt. Dies wiederum steht in einem direkten Zusammenhang mit der Konnektionsstärke: Je höher die Wahrscheinlichkeit für den Besuch von einem Knoten zu einem anderen Knotens ist, desto stärker müssen die beiden Knoten im Graphen miteinander verbunden sein.

Nachdem wir nun den Zusammenhang zur Konnektionsstärke hergestellt haben, fehlt noch die Darstellung, auf welcher Weise wir die Gewichtung der Kanten in das *PageRank* Modell inkorporieren wollen. Zur Unterscheidung der beiden Modelle wollen wir das modifizierte Modell als *Weighted PageRank* bezeichnen. Sei dazu $W = (w_{ij})$ die gewichtete Adjazenzmatrix eines Graphens $G = (V, E, w)$. Die Übergangsmatrix $P = (p_{ij})$ der induzierten Markovkette definieren wir wie folgt[5]:

[5] Der *virtuelle Referenzgraph* muss jedoch nicht zusammenhängend sein, sondern kann aus mehreren Komponenten bestehen, wobei sich in diesem Fall die Definition der Übergangsmatrix auf die jeweilige Zusammenhangskomponente des Graphens beziehen soll.

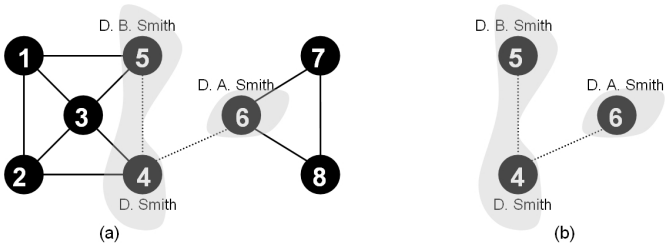


Abbildung 3.5.: (a) *erweiterter Referenzgraph* und (b) *virtueller Konsolidierungsteilgraph*

	Werte
$I(v_1 \mid v_4)$	0.145
$I(v_2 \mid v_4)$	0.190
$I(v_3 \mid v_4)$	0.235
$I(v_4 \mid v_4)$	0.259
$I(v_5 \mid v_4)$	0.102
$I(v_6 \mid v_4)$	0.027
$I(v_7 \mid v_4)$	0.019
$I(v_8 \mid v_4)$	0.019

Tabelle 3.3.: Unser Modell des *personalisierten gewichteten PageRanks* zur Bestimmung der Konnektionsstärke mit Dämpfungsfaktor $d = 0.85$

$$p_{ij} = \begin{cases} w_{ij} / \sum_j w_{ij} & \text{falls } \sum_j w_{ij} > 0, \\ 0 & \text{andernfalls.} \end{cases} \tag{3.7}$$

Die obige Definition können wir noch dahingehend vereinfachen, dass der zweite Fall (eine Zeile mit nur 0-Einträgen) in einem ungerichteten Graphen nicht eintreten kann. Zur Berechnung der Konnektionsstärke $I(u \mid v)$ bezüglich eines Hauptknotens v definieren wir einen Teleportationsvektor v mit $v_i = 1$ falls der i-te Eintrag mit dem Knoten v korrespondiert und für alle anderen Indizes setzen wir $v_i = 0$. Der Dämpfungsvektor d lässt nach wie vor eine Möglichkeit zur Parametrisierung offen und gibt in unserem Modell die Wahrscheinlichkeit an, mit die sich der *Random Surfer* zurück zum Hauptknoten teleportiert und von dort aus seinen Pfad wieder aufnimmt. Dieser stochastische Prozess definiert also eine unendliche Menge von Pfaden variabler Länge, die unter einer gegebenen Wahrscheinlichkeit (dem Dämpfungsfaktor) immer wieder von dem Hauptknoten aus starten. Der PageRank Algorithmus liefert mit diesen Eingaben eine stationäre Wahrscheinlichkeitsverteilung π der Markovkette, die den Zeitanteil für den Aufenthalt des *Random Surfers* in einem bestimmten Knoten angibt:

$$r^{(t+1)} = dPr^{(t)} + (1 - d)v \tag{3.8}$$

Der resultierende *PageRank* Vektor bzw. die stationäre Verteilung π verwenden wir als unsere Definition der relativen Wichtigkeit, d.h. $I(u \mid v) = \pi_u$. Es bleibt zu untersuchen, ob dieses modifizierte Modell unseren eingangs gestellten Anforderungen zur Bestimmung der Konnektionsstärke standhält. Diese werden offensichtlich erfüllt: Zum einen werden den Knoten, die durch viele Verbindungen vom Hauptknoten erreichbar sind, ein höherer Wert zugewiesen. Zum anderen werden die längeren Wege durch den

Dämpfungsvektor mit einem Strafparameter versehen. Im Allgemeinen können wir mit dieser Definition nicht davon ausgehen, dass die *Irreduzibilität* und *Aperiodizität* der *Markovkette* gewährleistet ist. Beim *PageRank* haben wir diese durch die Bedingung erzwungen, dass wir für den Teleportationsvektor positive Werte gefordert haben, d.h. $v > 0$. Die erste Möglichkeit würde also darin bestehen, für alle Elemente des Vektors kleine Mindestwahrscheinlichkeit einzuführen, wodurch jedoch unsere Vorstellung der Konnektionsstärke nicht mehr akkurat reflektiert werden würde. Falls wir uns aber auf ungerichtete Graphen beschränken – wie wir sie in unserem Framework verwenden – so können wir folgendes Lemma ausnutzen:

Lemma 3.4 *Sei $G = (V, E)$ ein ungerichteter, zusammenhängender Graph, der nicht bipartit[6] ist, dann ist die durch diesen Graphen gegebene Markovkette M irreduzibel und aperiodisch.*

Beweis. Aufgrund von Lemma 2.6 und der Tatsache, dass ein ungerichteter, zusammenhängender Graph auch stark zusammenhängend ist, wissen wir, dass die durch den Graphen gegebene *Markovkette irreduzibel* ist. Außerdem gilt: M ist genau dann *aperiodisch*, wenn G nicht bipartit ist. Für einen Graphen aus genau einem Knoten ist dies trivial. Bei einem Graphen, mit mehr als einem Knoten ist dies ist wie folgt einzusehen: Ist der Graph bipartit, dann können wir immer nur in geraden Zeitpunkten zu einem Zustand i zurückkehren. Ist der Graph nicht bipartit, dann enthält er einen Kreis ungerader Länge. Somit können wir von jedem Zustand i auf einem Weg ungerader Länge zu i zurückkehren. Andererseits gibt es von jedem Zustand i aus einen Weg der Länge 2 zum Zustand i zurück (von i aus können wir eine Kante verlassen und über die gleiche Kante sofort zurückkehren). Damit ist der größte gemeinsame Teiler 1 und die *Markovkette M* nach Definition 2.7 *aperiodisch*. □

Bleibt natürlich noch die Frage zu klären, ob wir ohne weiteres davon ausgehen können, dass der *erweiterte Referenzgraph* nicht bipartit ist. Zunächst sei erwähnt, dass ein Graph genau dann nicht bipartit ist, wenn dieser einen Kreis ungerader Länge enthält. Da jede Koautorenschaft einer Publikation eine Clique aus Autorenreferenzen bildet, können wir im Allgemeinen von der Nicht-Bipartität des *erweiterten Referenzgraphens* ausgehen, da wir bereits bei Publikationen mit mehr als zwei Autoren einen ungeraden Kreis in diesem Graphen erhalten.

Beispiel 3.5 Betrachten wir hierzu wieder das Beispiel aus Abbildung 3.5. Da wir an der Konnektionsstärke der Kante (4,5) und (4,6) interessiert, wollen wir an diesen exemplarisch die Konnektionsstärke von $I(v_5 \,|v_4)$ und $I(v_6 \,|v_4)$ mit dem personalisierten *PageRank* bestimmen. Für die Ähnlichkeitskanten verwenden wir ein Gewicht von $0,1$ und für die Kanten der Koautorenschaftsbeziehung ein Gewicht von 0,9. Unter Benutzung dieser Gewichte ergibt sich folgende Übergangsmatrix $P = (p_{ij})$ der induzierten Markovkette:

[6] Ein Graph heißt bipartit, falls dessen Knotenmenge V in zwei disjunkte nicht-leeren Teilmengen V_1, V_2 mit $V_1 \cup V_2 = V$ eingeteilt werden kann, sodass keine zwei Knoten in V_1 und keine zwei Knoten in V_2 benachbart sind.

$$P = \begin{pmatrix} 0 & 0.33 & 0.33 & 0 & 0.33 & 0 & 0 & 0 \\ 0.33 & 0 & 0.33 & 0.33 & 0 & 0 & 0 & 0 \\ 0.25 & 0.25 & 0 & 0.25 & 0.25 & 0 & 0 & 0 \\ 0 & 0.45 & 0.45 & 0 & 0.05 & 0.05 & 0 & 0 \\ 0.47 & 0 & 0.47 & 0.05 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.05 & 0 & 0 & 0.47 & 0.47 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0 \end{pmatrix} \quad v = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Aus diesen beiden Eingaben ergibt sich mit dem *PageRank*-Algorithmus die stationäre Verteilung π, wobei für die Konnektionsstärke $I(u\,|v) = \pi_u$ gilt.

$$\pi = \begin{pmatrix} 0.145 \\ 0.190 \\ 0.235 \\ 0.259 \\ 0.102 \\ 0.027 \\ 0.019 \\ 0.019 \end{pmatrix}$$

Komplexitätsanalyse

Wie wir bereits in Kapitel 2.3.3 erklärt haben, besteht die Lösung des *PageRank* Vektors darin, den stationären Punkt des Eigensystems aus (2.14) zu bestimmen. Zu diesem Zweck verwenden wir die Potenziteration, in welcher die Gleichung (2.20) ausgehend von einem initialen Startvektor solange iteriert wird, bis die Sequenz $r^{(t)},\ t = 0, 1, 2, ...$ gegen den Fixpunkt π konvergiert. Die Potenziteration besitzt demnach eine Laufzeit von $O(M(n)nI)$, wobei $M(n)$ die Kosten einer einzelnen Matrix-Vektor Berechnung, n die Größe der Matrix – bzw. in unserem Fall die Anzahl der Knoten – und I die Anzahl der Iterationen bezeichnet. Unter Ausnutzung einiger graphspezifischen Eigenschaften, wie beispielsweise, dass die gewichtete Adjazenzmatrix dünn besetzt[7] ist, kann die Matrix-Vektor Berechnung auf $O(n)$ reduziert werden, womit sich dann für den *PageRank* Algorithmus eine Gesamtlaufzeit von $O(nI)$ ergibt. Es verbleibt noch die Frage nach der Konvergenzrate der Potenzmethode, mit der die maximale Anzahl an Iterationen festgelegt ist. In [LM03] wird die Anzahl der benötigten Iterationen I für eine Konvergenzgenauigkeit von ε mit dem Term $I \approx \log \varepsilon / \log(1 - d)$ abgeschätzt. Für eine Konvergenzgenauigkeit von 10^{-6} und einem Dämpfungsfaktor von $0,85$ werden also schätzungsweise 85 Iterationen bis zum Eintritt der Konvergenz benötigt. In der Literatur finden sich neben einigen Verbesserungen der Potenzmethode noch weitere Ansätze für eine effiziente Berechnung des *PageRanks*, auf die wir jedoch in unserer Arbeit nicht weiter eingehen wollen. Wir verweisen hierzu auf die Arbeiten von Berkhin [Ber05] und und Langville et al. [LM03].

[7] Eine dünn besetzte Matrix (engl.: sparse matrix) ist eine Matrix, welche nur sehr wenige von Null verschiedenen Einträge besitzt.

3.5. Phase 3: Partitionierung der virtuellen Konsolidierungsteilgraphen

Zur Konsolidierung der Referenzen in den *virtuellen Konsolidierungsteilgraphen* müssen diese in ihre korrespondierenden Entitätenclusters partitioniert werden. Zu Beginn des Kapitels haben wir diesbezüglich bereits darauf hingewiesen, dass wir folgende, in diesem Zusammenhang wichtige Annahme treffen: Wenn eine Referenz einer bestimmten Entität in einem *virtuellen Konsolidierungsteilgraphen* enthalten ist, dann sind nach Konstruktion dieser, alle weiteren Referenzen derselben Entität ebenfalls in diesem enthalten. Diese Annahme stellt eine wesentliche Vereinfachung für die Implementierung des Konsolidierungsframeworks dar, da wir uns zur Konsolidierung der Referenzen einer Entität auf einen einzelnen *Konsolidierungsteilgraphen* beschränken können. Auf der anderen Seite ist mit dieser Annahme auch das Manko verbunden, dass zwei Referenzen derselben Entität aufgrund ihrer Unähnlichkeit zueinander unterschiedlichen Konsolidierungsteilgraphen zugewiesen werden, obwohl beide über die Beziehungen des Netzwerks stark miteinander verbunden sein können. Das Konsolidierungsframework ist jedoch nicht in der Lage solche Situationen zu erkennen bzw. aufzulösen und somit besteht nach der Identifizierung der *Konsolidierungsteilgraphen* in der ersten Phase keine Möglichkeit mehr, dass derartige Referenzen noch in denselben Entiätencluster konsolidiert werden.

3.5.1. Konsolidierung als Identifizierung von Communities

Im Gegensatz zu der Arbeit von Chen et al. wollen wir die Partitionierung der *virtuellen Konsolidierungsteilgraphen* in dem übergeordneten Kontext der *sozialen Netzwerkanalyse* neu aufrollen. Unserer Meinung nach entspricht das uns vorliegende Partitionierungsproblem dem in Kapitel 2.4 vorgestellten Netzwerk Clustering zur Identifizierung von Community Struktur. Communities in den *Konsolidierungsteilgraphen* entsprechen dabei den Referenzmengen die dieselbe Entität repräsentieren. Zur Partitionierung der Referenzen verwendet Chen das von Shi und Malik in [SM00] vorgeschlagene *spektrale Clustering* Verfahren des *normalisierten Schnitts*. Wir wollen im nächsten Abschnitt mit einer Beschreibung des Verfahrens beginnen und dieses hinsichtlich seiner Fähigkeit zur Konsolidierung der Referenzen untersuchen. Zu diesem Zweck wollen wir einen direkten Bezug zu den in Kapitel 2.4.4 eingeführten Netzwerk Clustering Methoden herstellen. Im Anschluss daran werden wir die mit diesem Verfahren einhergehenden Nachteile und Schwächen aufzeigen und begründen, weshalb dieser Algorithmus unserer Meinung nach für den Einsatz in Phase 3 des Konsolidierungsalgorithmus nicht sonderlich geeignet ist. Im darauf folgenden Abschnitt stellen wir mit dem von Newman und Girvan vorgeschlagenen Algorithmus eine Alternative vor, die in empirischen Untersuchungen sehr gute Resultate hinsichtlich der Identifizierung von Communities geliefert hat.

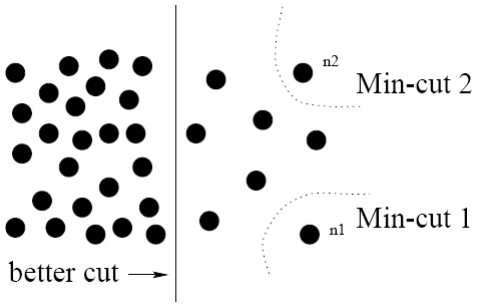


Abbildung 3.6.: Die Abbildung zeigt einen Beispielgraphen, in welcher der minimale Schnitt eine schlechte Partitionierung liefert.

3.5.2. Chen's Vorschlag: Spektrales Clustering von Shi u. Malik

In Kapitel 2.4.4 haben wir mit der *spektralen Bisektion* ein Verfahren zur Partitionierung von Graphen kennen gelernt, welches unter anderem zur Identifizierung von Community Struktur in einem Netzwerk verwendet werden kann. Der Algorithmus strebt dabei unter Berücksichtigung der Balance zweier Partitionen im Graphen einen minimalen Schnitt zwischen diesen Beiden an. Zum intuitiven Verständnis der *spektralen Bisektion* haben wir diese als ein diskretes Minimierungsproblem formuliert, dessen relaxierte Variante sich auf effiziente Weise durch die Lösung eines Eigenwertproblems bestimmen lässt. Die Aufteilung des Netzwerks in zwei Partitionen erfolgt dann über den zum zweitkleinsten Eigenwert korrespondierenden Eigenvektor (dem Fiedler Eigenvektor), bei der die Zuordnung der Knoten in die jeweilige Partition anhand des Vorzeichens der mit den Knoten korrespondierenden Fiedlerkomponenten getroffen wird. Newman hat diesbezüglich die Beobachtung gemacht, dass dieses *spektrale* Verfahren unter der Voraussetzung gut arbeitet, dass sich das Netzwerk gut in zwei Communities separieren lässt und weniger gut, wenn dem nicht so ist [NG03]. Diese Einschränkung macht die *spektrale Bisektion* für ein allgemeines Verfahren zur Identifizierung von Community Struktur nur bedingt geeignet. Die Schwäche des Verfahrens ist zum Teil auf die Zielfunktion des dazugehörigen Optimierungsproblems zurückzuführen: Da der Schnitt zwischen den beiden Partitionen minimiert werden soll, favorisiert die Zielfunktion Partitionen, die sich aus einer im Verhältnis zur Gesamtknotenmenge geringen Anzahl von überwiegend isolierten Knoten zusammensetzt (siehe dazu Abbildung 3.6). Das ist aber keine Überraschung, denn relaxieren wir zusätzlich die Balancebedingung, dann erhalten wir aus der Formulierung des diskreten Optimierungsproblems (2.37) das wohlbekannte Problem der Bestimmung eines minimalen Schnitts in einem Graphen, für welches effiziente Lösungsverfahren existieren. Um diesen unnatürlichen Bias hinsichtlich der Partitionierung isolierter Knoten bzw. kleiner Untergraphen zu vermeiden, haben Shi und Malik in ihrer Arbeit [SM00] eine Normalisierung der Zielfunktion vorgeschlagen. Anstatt das Gewicht des Kantenschnitts zwischen zwei Partitionen zu betrachten, strebt deren normalisierte Zielfunktion eine Minimierung vom Verhältnis des Kantenschnitts zur Gesamtsumme der gesamten von einer Partition ausgehenden Kantengewichte an (also auch die innerhalb der Parti-

tion liegenden Kanten):

$$Ncut(V_1, V_2) = \frac{w(V_1, V_2)}{w(V_1, V)} + \frac{w(V_1, V_2)}{w(V_2, V)} \tag{3.9}$$

Mit dieser Definition des normalisierten Schnitts wird die Trennung von isolierten Knoten dahingehend bestraft, dass das Verhältnis zwischen dem Schnitt und den gesamten von dieser Partition ausgehenden Verbindungen zu einem größeren Wert führt. So ist beispielsweise das Verhältnis des „Min-cut 1“ Schnitts aus Abbildung 3.6 zu den gesamten ausgehenden Verbindungen des Knotens n_1 maximal. Dieser Schnitt führt also zu keinem guten $Ncut(\cdot, \cdot)$ Wert.

Wir wollen uns im Folgenden überlegen, ob die auf diese Weise definierte Zielfunktion eine adäquate Lösungsmethode zur Identifizierung von *Communities* darstellt. Zu diesem Zweck wollen wir zunächst die Gleichung (3.9) ein wenig umschreiben:

$$\begin{aligned} Ncut(V_1, V_2) &= \frac{w(V_1, V_2)}{w(V_1, V)} + \frac{w(V_1, V_2)}{w(V_2, V)} \\ &= \frac{w(V_1, V) - w(V_1, V_1)}{w(V_1, V)} + \frac{w(V_2, V) - w(V_2, V_2)}{w(V_2, V)} \\ &= \left(\frac{w(V_1, V)}{w(V_1, V)} - \frac{w(V_1, V_1)}{w(V_1, V)}\right) + \left(\frac{w(V_2, V)}{w(V_2, V)} - \frac{w(V_2, V_2)}{w(V_2, V)}\right) \\ &= 2 - \left(\frac{w(V_1, V_1)}{w(V_1, V)} + \frac{w(V_2, V_2)}{w(V_2, V)}\right) \end{aligned} \tag{3.10}$$

Bei Betrachtung der letzten Umformung erhalten wir im Prinzip die Charakterisierung von Communities der Definition 2.10 aus Kapitel 2.4, bei der wir eine Maximierung der Intracluster-Kohäsion bei gleichzeitiger Minimierung der Intercluster-Kohäsion anstreben. Die Zielfunktion des normalisierten Schnitts stellt somit eine mit unserer Definition konformen Netzwerk Clustering Methode dar, mit der wir uns eine gute Aufteilung in Communities erhoffen.

Unglücklicherweise gehört die Minimierung des *normalisierten Schnitts* der Klasse der $\mathcal{NP}$-schweren Probleme an. Der Beweis hierzu kann in [SM00] nachgelesen werden. Wir können aber wieder mit einer Relaxierung des Problems der gleichen Strategie nachgehen, wie wir dies bereits bei der Handhabung des $\mathcal{NP}$-schweren Problems der *spektralen Bisektion* gesehen haben. Shi und Malik haben diesbezüglich gezeigt, dass durch die Relaxierung der Ganzzahligkeitsbedingung eine Lösung für das diskrete Problem des normalisierten Schnitts auf effiziente Weise approximiert werden kann. In ihrer Arbeit haben Sie nachgewiesen, dass die Optimierung der Zielfunktion des kontinuierlichen Problems auf die Berechnung eines generalisierten Eigensystems zurückzuführen ist, d.h. einem Eigensystem der Form:

$$(D - W)x = \lambda Dx \tag{3.11}$$

mit der Diagonalmatrix D und der gewichteten Adjazenzmatrix W. Wie schon bei der *spektralen Bisektion*, nimmt auch hier der Eigenvektor zum zweitkleinsten Eigenwert dieses generalisierten Eigensystems eine besondere Stellung ein, da wir mit genau diesem Vektor die reellwertige Optimallösung für das kontinuierliche Problem des *normalisierten Schnitts* erhalten. Im Allgemeinen kann man jedoch nicht davon ausgehen, dass die Komponenten des Vektors diskrete Werte annehmen, so dass Shi und Malik überlegt haben, wie sie aus der kontinuierlichen Optimallösung eine Lösung für das diskrete Problem approximieren können. Neben den im Zusammenhang mit der *spektralen Bisektion* genannten Möglichkeiten, den Median oder die Vorzeichen als Grundlage für die Partitionierung zu verwenden, haben Shi und Malik einen weitere Methode vorgeschlagen, die in direkter Weise auf die Minimierung der Zielfunktion abzielt. Zu diesem Zweck verwenden Sie l in gleichen Abständen verteilte Splitpunkte und berechnen für jeden dieser Splitpunkte den jeweiligen $Ncut(\cdot,\cdot)$ -Wert. Der Splitpunkt mit dem besten $Ncut(\cdot,\cdot)$ -Wert wird dann zur Partitionierung des Graphens ausgewählt. Eine Zusammenfassung des Algorithmus findet sich in Algorithmus 3.2 wieder.

Algorithmus 3.2 Shi und Malik's normalisierter Schnitt

Input: $G = (V, E, w)$
Output: Partitionen V_1 und V_2

1: $u_2 \leftarrow$ Berechne zweitkleinster Eigenvektor von $(D - W)x = \lambda Dx$
2: Bringe die Vektorelemente $(u_2)_j$ in aufsteigender Reihenfolge
3: **for** $i = 1$ to l **do**
4: $\quad j \leftarrow j + \lfloor |V| / l \rfloor$
5: $\quad V_1' \leftarrow \emptyset, V_2' \leftarrow \emptyset$
6: $\quad$ **for all** $v \in V$ **do**
7: $\quad\quad$ **if** $(u_2)_v < (u_2)_j$ **then**
8: $\quad\quad\quad V_1' \leftarrow V_1' \cup v$
9: $\quad\quad$ **else**
10: $\quad\quad\quad V_2' \leftarrow V_2' \cup v$
11: $\quad\quad$ **end if**
12: $\quad$ **end for**
13: $\quad$ **if** $Ncut(V_1', V_2') < Ncut(V_1, V_2)$ **then**
14: $\quad\quad V_1 \leftarrow V_1'$
15: $\quad\quad V_2 \leftarrow V_2'$
16: $\quad$ **end if**
17: **end for**

Laufzeitbetrachtung des spektralen Verfahrens von Shi u. Malik

Zur Berechnung des zweiten Eigenvektors kann wieder die Lanczos Methode verwendet werden, die unter Ausnutzung einiger graphspezifischen Eigenschaften, wie beispielsweise, dass die gewichtete Adjazenzmatrix dünn besetzt[8] ist, eine Laufzeit von $O(kn) + O(kM(n))$ besitzt [GL89]. Dabei bezeichnet n die Knotenanzahl, k die maximal notwendige Anzahl an Matrix-Vektor Berechnungen und $M(n)$ die Kosten einer einzelnen Matrix-Vektor Berechnung von Ax, wobei $A = D^{-1/2}(D - W)D^{-1/2}$ ist[9]. Unter Ausnutzung der Dünnbesetztheit der Matrix kann die Matrix-Vektor Berechnung auf $O(n)$ reduziert werden. Die Anzahl k der notwendigen Iterationen hängt von mehreren Faktoren ab und ist in der Regel kleiner als $O(n^{0,5})$. Setzen wir alles zusammen, ergibt sich für die Lanczos Methode eine Gesamtlaufzeit von $O(n^{1,5})$.

Analyse des Verfahrens von Shi u. Malik

Obwohl das *spektrale* Verfahren von Shi und Malik mit unserer Definition von Communities konform läuft – die Verbindungen zwischen den Clusters zu minimieren und die Verbindungen innerhalb der Clusters zu maximieren – sind mit dieser Methode hinsichtlich der Identifizierung von Community Struktur einige Probleme und Nachteile verbunden:

- Das Verfahren teilt – ebenso wie die *spektrale Bisektion* – ein Netzwerk in zwei Partitionen auf. Aufteilungen in eine größere Anzahl von Communities erfolgt durch eine rekursive Bisektion des Netzwerks. In empirischen Untersuchungen hat sich jedoch gezeigt, dass die dadurch entstehende Partitionierung nicht immer der tatsächlich existierenden Community Struktur entspricht.

- Auch wenn das Verfahren dazu in der Lage ist eine gute Partitionierung zu finden, wissen wir jedoch nicht im Vorhinein, in wie vielen Communities das Netzwerk zu partitionieren ist. Im Konsolidierungsframework von Chen wird diesem Problem dadurch entgegnet, indem das Verfahren einen vordefinierten Schwellenwert τ als Parameter übergeben bekommt. Anhand dieses Parameters trifft der Algorithmus dann die Entscheidung, ob ein *Konsolidierungsteilgraph* weiter aufzuteilen ist oder nicht. Unserer Meinung nach stellt dieses Vorgehen einen unbefriedigenden Lösungsversuch dar, da dieser Parameter zum einen abhängig von einer spezifischen Domäne sein kann und zum anderen auch vom jeweils vorliegenden *Konsolidierungsteilgraphen* selbst abhängen kann.

- Darüber hinaus besitzt dieses *spektrale* Verfahren keine eingebaute Metrik mit der die Qualität einer gefundenen Aufteilung des Netzwerks evaluiert werden kann. Ein solches Maß wäre aber dazu hilfreich die im Verlaufe des Partitionierungsprozesses

[8] Eine dünn besetzte Matrix (engl.: sparse matrix) ist eine Matrix, welche nur sehr wenige von Null verschiedenen Einträge besitzt.

[9] Das generalisierte Eigensystem $(D-W)x = \lambda Dx$ kann in das Standard Eigenwert Problem $D^{-1/2}(D-W)D^{-1/2}x = \lambda x$ transformiert werden

entstehenden Partitionen des *Konsolidierungsteilgraphens* gegeneinander abwägen zu können und die nach diesem Maß beste gefundene Partition auszuwählen.

Aus den oben genannten Gründen erachten wir das von Chen in Phase 3 eingesetzte *spektrale Verfahren* zur Konsolidierung der Referenzen als nicht geeignet. Kehren wir also zu Kapitel 2.4 zurück, in welchem wir mit den hierarchischen Clustering Verfahren noch eine weitere Methode zur Identifizierung von Community Struktur vorgestellt haben. Ein solches hierarchische Verfahren wollen wir im nächsten Abschnitt vorstellen und argumentieren, weshalb wir der Auffassung sind, dass sich dieses in Bezug auf die Konsolidierung der Referenzen eher eignet, als das eben vorgestellte *spektrale* Verfahren.

3.5.3. Unser Vorschlag: Algorithmus von Newman u. Girvan

In Kapitel 2.4.4 haben wir mit dem hierarchischen Clustering ein der *sozialen Netzwerkanalyse* entspringendes Verfahren zur Identifizierung von Community Struktur eingeführt. Wie wir in dem Kapitel beschrieben haben, lassen sich die hierarchischen Verfahren in zwei Klassen einteilen: agglomerative und divisive Verfahren, je nachdem ob das Hinzufügen oder das Entfernen der Kanten zum oder aus dem Netzwerk betrachtet wird. Die Reihenfolge des Hinzufügens (Entfernens) der Kanten bei agglomerativen (divisiven) Verfahren basiert dabei auf verschiedenen Metriken der Ähnlichkeit oder der Konnektionsstärke zwischen den Knoten, wobei auch für einige Netzwerke bereits ein natürliches Ähnlichkeitsmaß eingebaut sein kann. Im Falle der Partitionierung der *virtuellen Konsolidierungsteilgraphen* haben wir mit der Konnektionsstärke solch ein natürliches Ähnlichkeitsmaß vorliegen, auf der die Aufteilung des Netzwerks in Communities beruhen soll. Da wir im Prinzip mit der Konnektionsstärke als Metrik die einzige Voraussetzung für ein agglomeratives Verfahren erfüllen, könnten wir dieses zur Partitionierung der *Konsolidierungsteilgraphen* einsetzen, indem wir ausgehend von der leeren Kantemenge, sukzessive die Kanten in absteigender Reihenfolge dem Netzwerk hinzufügen. Wie wir die Communities letzten Endes erhalten, haben wir bereits in Kapitel 2.4.4 im Zusammenhang mit der Beschreibung der *single-link* Methode erläutert, bei der wir die entstehenden Zusammenhangskomponenten als Communities deklariert haben.

Auf den ersten Blick scheint dieses auf der Hand liegende Vorgehen plausibel zu sein, jedoch hat Newman in empirischen Untersuchungen festgestellt, dass die agglomerativen Verfahren in Netzwerken in denen die korrekte Aufteilung in Communities bekannt ist, häufig im Finden dieser fehlschlagen. Ein weiteres Problem mit den agglomerativen Verfahren besteht darin, dass diese die Tendenz haben nur die Kernkomponenten der Communities zu finden, die Peripherie jedoch vernachlässigen. Dies liegt darin begründet, dass die im Kern einer Community befindlichen Knoten eine hohe Konnektionsstärke aufweisen, während die peripheren Knoten im Vergleich zu Ersteren nur schwach mit den Communities verbunden sind und während des Prozedurverlaufs vernachlässigt werden. Abbildung 3.7 illustriert das in Verbindung mit der agglomerativen Methode existente Problem, dass einige periphere Randknoten, deren Zugehörigkeit zu den Communities

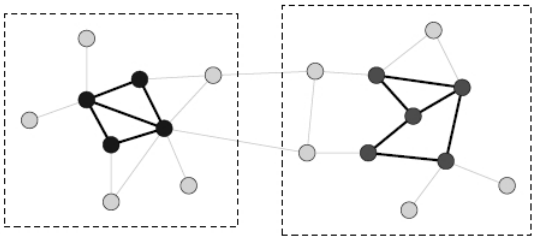


Abbildung 3.7.: Agglomerative Clustering Verfahren eignen sich in der Regel gut bei der Identifizieung der stark verbundenen Kernkomponenten, tendieren jedoch dazu periphere Knoten auszulassen. Diese Schwäche findet sich auch in Netzwerken wieder, bei der die Zuordnung der Knoten klar hervorgeht, wie dies beispielsweise in der Abbildung der Fall ist.

klar ersichtlich ist, am Ende keiner oder einer falschen Community zugewiesen werden. Hinsichtlich der Konsolidierung von Autorenreferenzen kann sich dies negativ auswirken, da wir von der Annahme ausgehen können, dass Autoren ohne Koautoren – d.h. also Publikationen mit nur einem Autor – nur schwach mit den korrespondierenden Entitätenclusters verbunden sind.

In diesem Abschnitt wollen wir mit dem von Newman und Girvan vorgeschlagenen Community Algorithmus ein divisives Verfahren vorstellen, bei der im Gegensatz zu den hierarchischen Standardverfahren ein anderer philosophischer Standpunkt vertreten wird. Anstatt die Kante mit der geringsten Konnektionsstärke als nächste zu entfernende Kante auszuwählen, wird die Kante mit der höchsten *Betweenness* – einer Generalisierung der *Betweenness* Zentralität aus Kapitel 2.3.1 – als nächste aus dem Netzwerk entfernt. Bei der *Betweenness* handelt es sich um ein Maß, welches die zwischen den Communities liegenden Kanten bevorzugt und die Kanten benachteiligt, die innerhalb der Communities liegen. Zur Bestimmung der *Betweenness* einer Kante werden dazu für alle möglichen Knotenpaare im Netzwerk die Häufigkeit gezählt, wie oft die betreffende Kante auf den *geodesischen* Pfaden zwischen diesen Knotenpaaren vorkommt. Die Kanten mit einem hohen *Betweenness*-Wert sind also in gewisser Weise für den Zusammenhalt vieler Knotenpaare verantwortlich und nehmen gleichsam eine Art Brücken-Funktion ein.

Der Algorithmus

Die Idee hinter dieser Methode ist folgende: In einem Netzwerk mit schwach zusammenhängenden Communities entwickeln sich bei Betrachtung der *geodesischen* Pfade die wenigen Kanten, die zwischen den Communities liegen, zu Flaschenhälsen. Demnach führen die kürzesten Wege von einer Community in eine andere genau über einer oder mehrere dieser Flaschenhälse, sodass diesen bei der Betrachtung aller Knotenpaare zwangsläufig eine größere *Betweenness* beigemessen wird als den anderen Kanten im Netzwerk. Eine solche Inter-Community Verbindung haben wir im Netzwerk aus Ab-

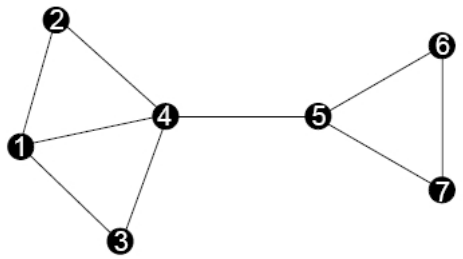


Abbildung 3.8.: Die Kanten *Betweenness* ergibt sich aus der Häufigkeit, wie oft die betreffende Kante auf den *geodesischen* Pfaden zwischen allen Knotenpaaren vorkommt. In der obigen Abbildung ist die mit den Knoten v_4 und Knoten v_5 inzidente Kante diejenige mit dem größten *Betweenness* Wert. Durch das Entfernen dieser Kante zerfällt das Netzwerk in die beiden offensichtlichen Communities.

bildung 3.8 mit der Kante (v_4, v_5) vorliegen. Durch das Entfernen der Kanten mit dem höchsten *Betweenness* Wert sollten wir demnach eine natürliche Aufteilung in Communities erzielen.

Sortieren wir die Kanten in absteigender Reihenfolge ihrer *Betweenness* so kann diese Methode in ein hierarchisches Clustering Verfahren eingebettet werden, wie wir dies in Kapitel 2.4.4 beschrieben haben, wobei wir in diesem Fall ein divisives Standardverfahren vorliegen haben. Dem Standardverfahren zufolge werden also die Kanten in der oben beschriebenen Reihenfolge solange sukzessive entfernt, bis keine Kanten mehr im Netzwerk vorhanden sind. Der gesamte Prozess vom Beginn bis zum Ende lässt sich wieder in Form eines Dendrogramms darstellen, wobei, wie gehabt, die Communities durch horizontale Schnitte aus dieser Baumstruktur hervorgehen. Newman und Girvan haben festgestellt, dass diese Vorgehensweise zu einem unerwünschten Verhalten führen kann: Durch das Entfernen einer Kante kann es vorkommen, dass das Netzwerk nich mehr akkurat durch die *Betweenness*-Werte der übrigen Kanten reflektiert wird. Dies soll am nachfolgenden Beispiel verdeutlicht werden: Angenommen den Fall, dass zwei Communities durch zwei Kanten verbunden sind und aus irgendeinem Grund der überwiegende Teil der geodesischen Pfade zwischen den beiden Communities über nur einer dieser beiden Kanten verläuft, so wird – obwohl beide Kanten strukturell äquivalent sind – der stärker frequentierten Kante eine höhere *Betweenness* beigemessen als der weniger stark frequentierten Kante. In einem Verfahren bei der die *Betweenness* nur einmalig berechnet und anschließend die Kanten in der Reihenfolge dieser entfernt werden, würde also die erste Kante frühzeitig aus dem Netzwerk entfernt werden, wohingegen die zweite, weniger frequentierte Kante möglicherweise erst zu einem sehr viel späteren Zeitpunkt. Dies hat zur Folge, dass die tatsächliche Community Struktur vom Algorithmus unentdeckt bleibt. Im schlimmsten Szenario könnten sogar die beiden Communities selbst individuell aufgeteilt werden, noch bevor die Partitionierung der Beiden erfolgt ist. Aus diesem Grund wird im Algorithmus von Newman nach jedem Entfernen einer Kante das *Betweenness* Maß für alle Kanten neu berechnet. In empirischen Untersuchungen konnte

Newman nachweisen, dass die Aktualisierung der *Betweenness* Werte einen entscheidenden Faktor für den Algorithmus darstellt, was die Qualität der Ergebnisse anbetrifft. Dieser Schritt führt zwar zu einer Erhöhung der Komplexität, die jedoch angesichts der signifikanten Verbesserung der Resultate in Kauf genommen wird. Der Newman und Girvan Algorithmus ist in 3.3 dargestellt.

Algorithmus 3.3 Newman&Girvan's Kanten Betweenness

Input: $G = (V, E)$
Output: Partitionierung $\mathcal{P} = \{V_1, ..., V_k\}$

```
1: Initialisere P ← {V}
2: repeat
3:     for all e ∈ E do
4:         Berechne Betweenness C_B(e)
5:     end for
6:     e' ← arg max {C_B(e) | e ∈ E}
7:     E ← E − {e'}
8:     P' ← Extrahiere Zusammenhangskomponenten
9:     if Q(P') ⩾ Q(P) then
10:        P ← P'
11:    end if
12: until E = ∅
13: return P
```

Um nun eine gute Aufteilung des Netzwerks in Communities zu erhalten, verwenden wir die von Newman und Girvan vorgeschlagene Modularitätsfunktion Q (2.31) mit der die mit einem Schnitt des Dendrogramms induzierten Aufteilungen evaluiert werden können. Die Modularitätsfunktion liefert uns somit ein Kriterium in wie vielen Communities das Netzwerk aufgeteilt werden soll. Hinsichtlich der Objektkonsolidierung ist dies ein entscheidender Faktor, da wir die Größe eines Clusters – die zusammengefassten Referenzen einer Entität – im Vorhinein nicht kennen und mit der Modularitätsfunktion über ein adäquates Mittel verfügen, um zu entscheiden in wie vielen Clusters ein *virtueller Konsolidierungsteilgraph* partitioniert werden sollte. Dies stellt auch einen entscheidenden Vorteil gegenüber den von Chen et al. verwendeten *spektralen Clustering* Verfahrens dar, bei der die Entscheidung zur weiteren Partitionierung durch einen Parameter τ getroffen wird, der jedoch zum einen abhängig von einer spezifischen Domäne sein kann und zum anderen auch vom jeweils vorliegenden *Konsolidierungsteilgraphen* selbst abhängen kann.

Komplexitätsanalyse

Auf den ersten Blick erscheint die Berechnung der Kanten *Betweenness* in einem Graphen mit n Knoten und m Kanten basierend auf den geodesischen Pfaden $O(n^2m)$ Graphenoperationen zu benötigen: Die Berechnung der kürzesten Wege zwischen einem bestimmten Knotenpaar kann in einer Laufzeit von $O(m)$ implementiert werden, womit

sich dann mit $O(n^2)$ Knotenpaaren die obige Laufzeit ergibt. Durch den Einsatz effizienterer Methoden, die Newman [NG03] und Brandes [Bra01] unabhängig voneinander entwickelt haben, kann die Laufzeit für nicht gewichtete Netzwerke in einer Laufzeit von $O(nm)$ durchgeführt werden. Da aber diese Berechnung für jede zu entfernende Kante wiederholt werden muss – von denen es m Kanten gibt – resultiert der vollständige Algorithmus zur Identifizierung von Communities in einer Worst-Case Laufzeit von $O(m^2n)$.

Generalisierung auf gewichtete Graphen

Nachdem wir nun mit dem Newman und Girvan Algorithmus ein Verfahren zur Identifizierung von Communities beschrieben haben, müssen wir uns noch überlegen, wie wir diesen auf gewichtete Graphen – wie wir sie mit den *virtuellen Konsolidierungsteilgraphen* vorliegen haben – erweitern können. Zur Erinnerung: In Phase 2 des Konsolidierungsframeworks wird für jeden *Konsolidierunsteilgraphen* paarweise die Konnektionsstärke zwischen ähnlichen Referenzen bestimmt. In der Phase 3 soll dann basierend auf dieser Konnektionsstärke die Identifizierung von Communities erfolgen, um eine weitere Disambiguierung der *Konsolidierungsteilgraphen* zu bewirken. Da das *Kontext-Attraktions-Prinzip*, auf dem das Konsolidierungsframework beruht, besagt, dass wenn zwei Referenzen dieselbe Entität repräsentieren, diese dann über weiteren expliziten sowie impliziten Beziehungen miteinander verbunden sind. Demzufolge sollten diese Referenzen eine stärkere Konnektionsstärke zugewiesen bekommen als Referenzen unterschiedlicher Entitäten und folglich in dieselbe Community partitioniert werden. Der bisher beschriebene Algorithmus 3.3 ignoriert jedoch diese wichtige Information der Konnektionsstärke, so dass wir der Frage nachgehen müssen, wie wir diese in den Algorithmus mit einfließen lassen können. In [New04b] geht Newman dieser Frage nach und gibt eine simple Erweiterung des Algorithmus für gewichtete Graphen an. Ein zunächst naheliegender Ansatz besteht darin, die Definition der Kanten *Betweenness* auf gewichtete Graphen zu erweitern. Zu diesem Zweck können beispielsweise die Pfade in einem gewichteten Netzwerk über die Kantengewichte definiert werden, indem wir für die Länge einer Kante den Kehrwert ihres Gewichts annehmen. Man beachte hierbei, dass dies mit unserer Interpretation der Konnektionsstärke konform läuft, da dadurch miteinander stärker verbundene Knoten näher zusammen liegen. Um die *geodesischen* Pfade in einem derartigen Netzwerk zu bestimmen kann ein Kürzester-Wege-Algorithmus, wie beispielsweise der *Dijsktra* Algorithmus, verwendet werden. Die *Betweenness* einer Kante ergibt sich dann wieder aus der Häufigkeit, wie oft die betreffende Kante auf den geodesischen Pfaden zwischen allen Knotenpaaren vorkommt. Der Algorithmus zur Identifizierung der Community Struktur funktioniert dann also analog zum Algorithmus für ungewichtete Graphen. Trotz dieser naheliegenden Erweiterung zur ursprünglichen Methode liefert der Algorithmus schlechte Resultate. Der Grund hierfür liegt an der Definition der Pfadlänge als Kehrwert der Konnektionsstärke, wodurch zwei stark miteinander verbundene Knoten entlang ihrer inzidenten Kante über einer kurzen Distanz verbunden sind. Die *geodesichen* Pfade verlaufen dann vorzugsweise über diese Kanten, anstelle der Kanten, die weniger stark verbunden sind, wodurch Erstere – Kanten mit

einer hohen Konnektionsstärke – eine größere *Betweenness* akquirien. Das ist aber genau das Gegenteil von dem, was wir eigentlich mit dem Algorithmus bezwecken wollten – die Inter-Communities Kanten durch eine hohe *Betweenness* zu identifizieren und frühzeitig aus dem Netzwerk zu entfernen. Dies hat zur Folge, dass durch die Erweiterung des Algorithmus die stark zusammenhängenden Knotenpaaren eher separiert, als dass diese in dieselbe Community platziert werden.

Newman hat also diesen Ansatz verworfen und ist zu der Erkenntnis gelangt, dass die einfachste Erweiterung auf gewichtete Netzwerke darin besteht, die Gewichte zunächst einmal zu ignorieren und die *Betweenness* wie gewohnt zu berechnen. Um die Konnektionsstärke wieder ins Spiel zu bringen, werden dann die *Betweenness* Werte durch das Gewicht ihrer korrespondierenden Kante dividiert. Durch diese Division erreichen wir, dass die Betweenness Werte von Kanten mit einer hohen Konnektionsstärke relativiert werden. Um also einen Algorithmus für gewichtete Graphen zu erhalten müssen wir lediglich die Zeile 4 im Algorithmus 3.3 zu $C_B(e)/w_e$ abändern. Newman hat gezeigt, dass diese einfache Erweiterung des ursprünglichen Netzwerks in der Praxis gute Resultate liefert. Ein weiterer Vorteil, der daraus resultiert ist, dass die Laufzeit im Vergleich zum ursprünglichen Algorithmus kaum beeinträchtigt wird (zusätzlich eine Extradivision).

3.6. Metriken zur Evaluation

Um unsere in dem Konsolidierungsalgorithmus eingesetzten Verfahren miteinander vergleichen zu können, sind Metriken erforderlich, mit der wir eine gefundene Partitionierung der (Autoren)Referenzen bewerten können. Eine Evaluation kann natürlich nur dann erfolgen, wenn wir die, dem Konsolidierungsalgorithmus unbekannte Resolutionsfunktion $r : X \rightarrow E$, die jede Referenz auf ihre korrespondierende Entität abbildet, kennen. Zu Beginn des Kapitels haben wir in diesem Zusammenhang erläutert, dass das Ziel der Objektkonsolidierung in der Identifizierung mehrfacher Repräsentationen einer Realwelt-Entität besteht. In der Literatur existieren für dieses Problem der Datenbereinigung zwei verschiedene Ansätze: Der eine Ansatz – den wir in unserer Arbeit verfolgen – formuliert die Objektkonsolidierung als Clustering Problem, bei der die Aufgabe darin besteht, alle mit einer Entität korrespondierenden Referenzen demselben Cluster zuzuweisen. Neben diesem Ansatz existiert mit der Objektkonsolidierung als paarweises Klassifizierungsproblem noch eine weitere Methode, bei der die Aufgabe eines Klassifizierers darin besteht, Paare von Referenzen in Duplikate und Nicht-Duplikate einzuteilen. Aus diesen beiden Sichtweisen heraus resultieren zwei unterschiedliche Metriken zur Evaluation der Objektkonsolidierung. Mit der Objektkonsolidierung als Klassifizierungsproblem sind die aus dem Information Retrieval bekannten Metriken *Precision* und *Recall* verbunden, die zur Bewertung von Suchergebnissen herangezogen werden. Im Falle der Objektkonsolidierung als Clusteraufgabe sind Metriken erforderlich, die sich mehr auf den Vergleich der erzeugten Clusters zu den tatsächlichen Entitätenclusters beziehen. Zu diesem Zweck wird zum einen die Diversität eines Clusters betrachtet, d.h. die Anzahl an unterschiedlichen Entitäten in einem Cluster, und zum anderen die Dispersion einer Entität mit der wir eine Aussage über die Streuung der Referenzen einer Entität

erhalten.

Im nächsten Abschnitt wollen wir zunächst einmal die *Cluster Diversität* und *Entitäten Dispersion* näher untersuchen, deren Zusammenhänge darstellen und zeigen, wie sich aus diesen beiden Eigenschaften zwei einfache Metriken gewinnen lassen, die uns Aufschluss über die Güte einer gefundenen Partitionierung geben können. Anschließend werden wir anhand eines Beispiels die Schwächen dieser eingeführten Metriken aufzeigen und mit der *Entropie*-basierten *Cluster Diversität* und *Entitäten Dispersion* zwei weitere Metriken vorstellen, die sich hinsichtlich der Qualitätsbewertung einer Clusterlösung als geeigneter herausstellen werden. Diese Metriken wurden in [BG06] vorgeschlagen. Abschließend wollen wir noch betrachten, wie die beiden Metriken, *Precision* und *Recall*, auf die Evaluation einer Clusterlösung übertragen werden können.

3.6.1. Cluster Diversität und Entitäten Dispersion

Idealerweise sollte jeder Cluster auf sich selbst bezogen homogen sein, d.h. alle darin enthaltenen Referenzen sollten mit derselben Entität korrespondieren. Diese Eigenschaft eines Clusters bezeichnen wir als die *Cluster Diversität*. Je mehr Diversität in Bezug auf die auftretenden Entitäten in einem Cluster vorhanden ist, desto inhomogener ist der Cluster, was wiederum eine Qualitätsminderung des Clusters mit sich zieht. Um diese Eigenschaft zu quantifizieren, könnte eine einfache Metrik darin bestehen, die Anzahl der in einem Cluster vorkommenden Entitäten festzustellen, wobei eine Minimierung dieser Quantität anzustreben ist. Im Idealfall sollte der Cluster mit genau einer Entität korrespondieren und folglich den optimalen Wert von 1 annehmen. Allerdings ist die Minimierung der *Cluster Diversität* ein unzureichendes Kriterium, denn wir können jederzeit diesen optimalen Wert erzielen, indem wir jeder Referenz ihren eigenen Cluster zuweisen. Dadurch wird jedoch die zweite Bedingung der in Kapitel 3.2 geforderten Konsolidierungskriterien (3.1) verletzt, da die Referenzen einer Entität nunmehr auf viele unterschiedlichen Clusters verteilt sind. Daher führen wir mit der *Entitäten Dispersion* eine zweite Eigenschaft zur qualitativen Bewertung einer Clusterlösung ein, die sich anstelle der Clusters auf die Entitäten selbst bezieht. Um diese Eigenschaft zu quantifizieren, betrachten wir für jede Entität die Verteilung ihrer Referenzen auf die verschiedenen Clusters. In gleicher Weise wie bei der *Cluster Diversität* streben wir eine niedrige *Entitäten Dispersion* an, so dass im Idealfall die Referenzen genau einem Cluster zugeordnet sind und das Maß den optimalen Wert von 1 annimmt. Auch hier können wir jederzeit einen optimalen Wert erzielen, indem wir, unter Verletzung der ersten Bedingung aus (3.1), eine triviale Clusterlösung angeben, bei der alle Referenzen einem einzelnen Cluster zugewiesen sind. Demnach bilden die beiden Metriken quasi einen *Trade-off* und müssen zur qualitativen Bewertung einer Clusterlösung gleichermaßen herangezogen werden.

Obwohl die auf diese Weise definierten Metriken der *Cluster Diversität* und *Entitäten Dispersion* einfach und intuitiv sind, so reflektieren diese nicht immer die Qualität einer vom Konsolidierungsalgorithmus gefundenen Clusterlösung wieder. Wir wollen dies an einem Beispiel illustrieren. Dazu nehmen wir an, dass die zu partitionierende Datenmen-

	Div(C_1)	Div(C_2)	Dis(E_A)	Dis(E_B)
Perfektes Clustering (3.12)	1	1	1	1
Szenario 1 Clustering (3.13)	2	2	2	2
Szenario 2 Clustering (3.14)	2	2	2	2

Tabelle 3.4.: Bei der perfekten Clusterlösung nehmen die naiv definierten Metriken der *Cluster Diversität* und *Entitäten Dispersion* jeweils das Optimum von 1 an. In den beiden Szenarien sind die Referenzen beider Entitäten über zwei Clusters verstreut und folglich beträgt die *Entitäten Dispersion* für E_A und E_B jeweils 2. Die *Cluster Diversität* beider Clusters beträgt ebenfalls 2, da jeder Cluster Referenzen beider Entitäten enthält. Die *Cluster Diversität* berücksichtigt dabei jedoch nicht die Tatsache, dass im zweiten Szenario jeweils nur eine falsche Referenz je Cluster enthalten ist und somit diese eine bessere Clusterlösung darstellt als die des ersten Szenarios.

ge aus $2n$ Referenzen $a_1, a_2..., a_{2n}$ einer Entität E_A und aus $2n$ Referenzen $b_1, b_2..., b_{2n}$ einer Entität E_B zusammengesetzt ist. Die perfekte Clusterlösung besteht dann also aus den beiden Clustern:

$$\begin{aligned} C_1 &= \{a_1, a_2, ..., a_{2n}\} \\ C_2 &= \{b_1, b_2, ..., b_{2n}\} \end{aligned} \tag{3.12}$$

Wir betrachten nun zwei mögliche Szenarien einer fehlerhaften Objektkonsolidierung und vergleichen diese mit den beiden oben eingeführten Metriken. In einem ersten Szenario haben wir folgende Clusterlösung vorliegen:

$$\begin{aligned} C_1 &= \{a_1, a_2, ...a_n, b_1, b_2, ..., b_n\} \\ C_2 &= \{a_{n+1}, a_{n+2}, ...a_{2n}, b_{n+1}, b_{n+2}, ..., b_{2n}\} \end{aligned} \tag{3.13}$$

Das heißt, die Hälfte der Referenzen wird dem falschen Cluster zugewiesen. In einem zweiten Szenario haben wir nun eine Clusterlösung vorliegen, in der jeder Cluster nur eine falsch zugewiesene Entität enthält:

$$\begin{aligned} C_1 &= \{a_1, a_2, ..., a_{2n-1}, b_{2n}\} \\ C_2 &= \{b_1, b_2, ..., b_{2n-1}, a_{2n}\} \end{aligned} \tag{3.14}$$

Offensichtlich ist letztere Lösung verglichen mit Ersterer wesentlich besser. Nichtsdestotrotz, falls wir zur Bewertung der beiden Clusterlösungen die oben eingeführten

Metriken heranziehen, sind aus dieser Sicht, beide Clusterlösungen gleich gut (siehe dazu Tabelle 3.4). Demnach sind diese naiv definierten Metriken der *Entitäten Dispersion* und *Cluster Diversität* nicht in der Lage, die qualitativen Unterschiede der beiden Clusterlösungen akkurat wiederzuspiegeln. Um nun die *Dispersion* und *Diversität* hinsichtlich der Qualitätsbewertung einer Clusterlösung akkurater zu quantifizieren, wollen wir zu diesem Zweck mit der *Entropie* ein Maß aus der Informationstheorie heranziehen. In erster Linie wird dieses Maß dazu verwendet, die Unbestimmtheit von Zufallsexperimenten zu messen. Auf den ersten Blick scheint diese Vorgehensweise weit hergeholt zu sein; wie wir aber gleich sehen werden gibt es neben dieser Interpretation noch andere mögliche Interpretationen der *Entropie*, von der sich eine im Kontext der Evaluation von Clusterlösungen als wichtig herausstellen wird.

3.6.2. Entropie-basierte Cluster Diversität und Entitäten Dispersion

Im Allgemeinen ist der Ausgang eines Zufallsexperiments im Vorhinein nicht vorhersagbar, da die Verwendung von Wahrscheinlichkeiten ein gewisses Maß an Unsicherheit über den zu erwartenden Ausgang eines Zufallsexperimentes impliziert. Um nun die Verlässlichkeit der Vorhersagen besser einschätzen zu können, wäre es nützlich, ein Maß zu haben, das als Parameter die Wahrscheinlichkeitsverteilung entgegennimmt und als Ergebnis die Unbestimmtheit der mit dieser assoziierten Verteilung liefert. Beschränken wir uns im Weiteren auf Experimente mit nur endlich vielen Ereignissen $\{x_1, x_2, ..., x_n\}$, in der jedes Ereignis x_i eine nach einer Wahrscheinlichkeitsverteilung bestimmte Auftrittswahrscheinlichkeit $p(x_i)$ besitzt. Überlegen wir uns nun, welche Anforderungen an ein solches Maß zu stellen sind. Um dies zu konkretisieren, nennen wir das Maß $H(X)$, das die Wahrscheinlichkeitsverteilung $X = \{p(x_1), p(x_2), ..., p(x_n)\}$ als Argument nimmt und als Funktionswert eine reelle Zahl liefert. Als erstes sollte die Funktion H die Eigenschaft besitzen, dass diese bei einer uniformen Wahrscheinlichkeitsverteilung den maximalen Wert annimmt, da mit dieser die größte Unbestimmtheit in der Vorhersage des Ausgangs assoziiert ist. Zweitens sollte die Funktion ihren minimalen Wert annehmen, wenn wir den Ausgang eines Experiments mit absoluter Gewissheit vorhersagen können. Dies trifft auf den Fall zu, wenn für ein Ereignis i die sichere Wahrscheinlichkeit $p(x_i) = 1$ gilt, so dass keine Unbestimmtheit mit der Verteilung X assoziiert ist. Diese Eigenschaften führen uns zu der fundamentalen Definition der *Entropie*, die erstmals von Shannon in seiner grundlegenden Arbeit [WS49] von 1948 über die Informationstheorie eingeführt wurde.

Definition 3.6 (Entropie): Sei X eine Zufallsvariable über einer endlichen Menge von Ereignissen $\{x_1, x_2, ..., x_n\}$. Mit den Ereignissen sind Wahrscheinlichkeitsfunktionen $\{p(x_1), p(x_2), ..., p(x_n)\}$ assoziiert, so dass $p(X) = \sum_{i=1}^{n} x_i = 1$ gilt. Die *Entropie* von X, $H(X)$ genannt, ist dann definiert als:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2(p(x_i)) \tag{3.15}$$

Die Notation $H(X)$ kann etwas irreführend sein. Wir beziehen uns dabei nicht auf eine Funktion der Zufallsvariable X, sondern auf eine Funktion der mit dieser Zufallsvariable assoziierten Wahrscheinlichkeitsverteilung. Die *Entropie* $H(X)$ nimmt den minimalen Wert bei $H(X) = 0$ an, wenn für ein Ereignis x_i die sichere Wahrscheinlichkeit $p(x_i) = 1$ gilt. In diesem Fall ist mit der Variablen X keine Unbestimmtheit assoziiert, da immer das sichere Ereignis x_i eintritt. Im Gegensatz dazu nimmt die *Entropie* $H(X)$ ihren maximalen Wert $H(X) = \log_2 n$ im ungewissesten Szenario an, wenn die Eintrittswahrscheinlichkeiten der Ereignisse uniform verteilt sind. Die *Entropie* einer diskreten Zufallsvariable X reflektiert also den Grad der Ungewissheit der mit dieser assoziierten Wahrscheinlichkeitsverteilung im Intervall $[0, \log_2 n]$. Eine Interpretation von *Entropie* aus der Informationstheorie besteht darin, dass diese die minimale Anzahl an Bits spezifiziert, die notwendig sind, um eine Nachricht mit einer bestimmten Auftrittswahrscheinlichkeit zu kodieren. Die Idee dabei ist, für Nachrichten mit hohen Auftrittswahrscheinlichkeiten kürzere Codes zu verwenden, so dass diese effizienter übertragen werden können.

Wir wollen nun zu unserem ursprünglichen Ziel zurückkehren, das darin bestand, ein geeignetes Maß zur Evaluation von Clusterlösungen zu finden. Zu diesem Zweck wollen wir die *Entropie* heranziehen. Anstatt, dass wir jedoch die *Entropie* als Maß für die Unbestimmtheit in der Vorhersage einer Zufallsvariable nehmen, interpretieren wir die *Entropie* als Reinheitsmaß, mit der die Homogenität eines Clusters gemessen werden kann. Stellt sich die Frage, ob dies eine adäquate Interpretation hinsichtlich der Evaluation von Clusterlösungen ist? Dies ist zu bejahen, denn wenn ein Cluster aus Referenzen einer einzelnen Entität besteht, so sollte das Maß den Wert 0 liefern. Für den Fall, dass ein Cluster unrein ist – der Cluster besteht aus vielen Referenzen unterschiedlicher Entitäten – sollte das Maß den maximalen Wert liefern. Diese Eigenschaften werden von der Entropiefunktion erfüllt. Wir zeigen nun, wie die beiden oben eingeführten Eigenschaften, *Cluster Diversität* und *Entitäten Dispersion*, durch die *Entropie* quantifiziert werden können.

Cluster Entropie

Betrachten wir zunächst die *Cluster Diversität*. Dazu nehmen wir an, dass der Konsolidierungsalgorithmus einem Cluster $c \in C$ insgesamt m Referenzen zuordnet, die zusammen n Entitäten beschreiben, mit $m \geqslant n$. Des Weiteren nehmen wir an, dass m_i Referenzen mit der Entität e_i korrespondieren und $\sum_{i=1}^{n} m_i = m$ gilt. Die Wahrscheinlichkeitsverteilung der mit den Referenzen korrespondierenden Entitäten in einem Cluster ist dann $\left\{\frac{m_1}{m}, \frac{m_2}{m}, ..., \frac{m_n}{m}\right\}$. Wir verwenden jetzt die Entropie unter dieser Wahrscheinlichkeitsverteilung der Entitäten, um die Diversität bzw. Homogenität des Clusters zu quantifizieren:

$$H(C) = -\sum_{i=1}^{n} \frac{m_i}{m} \log_2 \frac{m_i}{m} \tag{3.16}$$

	H(C_1)	H(C_2)	H(E_A)	H(E_B)
Ideales Clustering (3.12)	0	0	0	0
Szenario 1 Clustering (3.13)	1	1	1	1
Szenario 2 Clustering (3.14)	0.65	0.65	0.65	0.65

Tabelle 3.5.: Die ideale Clusterlösung nimmt die optimalen Entropiewerte für die Cluster und Entitäten Entropie im Minimum 0 an. Für die *Entropie*-basierten Metriken nehmen wir für die Anzahl der Referenzen der Entitäten E_A und E_B jeweils den Wert $m = 6$ an. Für das erste Szenario ergibt sich dann für die *Cluster Entropie* und *Entitäten Entropie* der maximale (schlechtester) Wert von 1.

Als Beispiel nehmen wir an, dass ein erzeugter Cluster zehn Referenzen enthält, von denen fünf Referenzen mit einer Entität korrespondieren, drei mit einer zweiten Entität und die restlichen zwei mit einer dritten Entität. Die Wahrscheinlichkeitsverteilung der mit den Referenzen korrespondierenden Entitäten in einem Cluster ist dann $\{0.5, 0.3, 0.2\}$. Die Entropie für diesen Cluster berechnet sich dann wie folgt: $H(C) = -\left(\frac{5}{10}\log_2\frac{5}{10} + \frac{3}{10}\log_2\frac{3}{10} + \frac{2}{10}\log_2\frac{2}{10}\right) = 1.485$

Entitäten Entropie

Die *Entitäten Entropie* definieren wir analog zu der *Cluster Entropie*. Anstatt der Diversität der Cluster betrachten wir bei der *Entitäten Entropie* die Dispersion der Referenzen einer Entität. Dazu nehmen wir an, dass in der Datenbank m Referenzen einer Entität $e \in E$ enthalten sind, die vom Konsolidierungsalgorithmus in n Clusters $c_1, c_2, ..., c_n$ partitioniert werden. Des Weiteren nehmen wir an, dass jeweils m_i Referenzen dem Cluster C_i zugewiesen werden, so dass $\sum_{i=1}^{n} m_i = m$ gilt. Um die Streuung der Referenzen von e über die Clusters zu quantifizieren, betrachten wir – analog zu der *Cluster Entropie* – die Wahrscheinlichkeitsverteilung $\left\{\frac{m_1}{m}, \frac{m_2}{m}, ..., \frac{m_n}{m}\right\}$. Verwenden wir dann wieder das Entropiemaß, so können wir die Streuung der Referenzen folgendermaßen quantifizieren:

$$H(e) = -\sum_{i=1}^{n} \frac{m_i}{m} \log_2 \frac{m_i}{m} \tag{3.17}$$

Als Beispiel für die *Entitäten Entropie* nehmen wir an, dass eine Entität durch zehn Referenzen in der Datenmenge repräsentiert wird und diese auf drei verschiedene Clusters verteilt sind. Der erste Cluster enthält fünf Referenzen der Entität, der zweite Cluster drei und der dritte Cluster enthält die restlichen zwei Referenzen. Die Wahrscheinlichkeitsverteilung der auf die drei Cluster verteilten Referenzen ist dann $\{0.5, 0.3, 0.2\}$. Die *Entropie* für diese Entität berechnet sich dann wie folgt: $H(e) = -\left(\frac{5}{10}\log_2\frac{5}{10} + \frac{3}{10}\log_2\frac{3}{10} + \frac{2}{10}\log_2\frac{2}{10}\right) = 1.485$

Wir wollen nun für die beiden obigen Szenarien die *Cluster* und *Entitäten Entropie* angeben. Eine Zusammenfassung der berechneten Werte für die beiden Szenarien findet sich in Tabelle 3.5 wieder. Im Gegensatz zur *Cluster Diversität* und *Entitäten Dispersion*, sind die *Entropie* basierten Metriken in der Lage die qualitativen Unterschiede der beiden Partitionen festzustellen, da $0.65 < 1.0$ ist.

Abschließend wollen wir noch zeigen, wie wir – ausgehend von den beiden eingeführten *Entropie*-Metriken – eine Gesamtbewertung der Clusters und Entitäten für eine vorliegende Clusterlösung erhalten. Im Falle der *Cluster Diversität* betrachten wir dazu den gewichteten Durchschnitt der Entropiewerte der gefundenen Clusters, wobei die Entropie eines Clusters mit der Anzahl der darin enthaltenen Referenzen gewichtet wird. Durch diese Gewichtung sollen größere Clusters das Maß dominieren. Die kombinierte *Cluster Diversität* einer Clusterlösung ergibt sich demnach wie folgt:

$$Div(\mathcal{C}) = \sum_{C \in \mathcal{C}} \frac{|C|}{|X|} H(C) \tag{3.18}$$

wobei X die Referenzmenge bezeichnet. Auf ähnlicher Weise erhalten wir für die kombinierte *Entitäten Dispersion* folgendes Maß:

$$Dis(E) = \sum_{e \in E} \frac{|r^{-1}(e)|}{|X|} H(e) \tag{3.19}$$

wobei $r^{-1}(e) = \{x \in X \,|r(x) = e\}$ die Umkehrrelation[10] der Resolutionsfunktion bezeichnet.

Da wir sowohl die *Cluster Diversität* als auch die *Entitäten Dispersion* über die Entropiefunktion definiert haben, streben wir für die Clusterlösung eine Minimierung dieser beiden Metriken an. Eine perfekte Clusterlösung hat somit eine *Cluster Diversität* und *Entitäten Dispersion* von 0. In der Praxis werden diese Werte in den seltensten Fällen erreicht und eine Erhöhung des einen Wertes hat die Senkung des anderen Wertes zur Folge, d.h. die beiden Metriken bilden wieder einen *Trade-off*. Durch die Bildung des arithmetischen Mittels der beiden Maße erhalten wir eine kombinierte Metrik:

$$Entropie_{Div/Dis} = \frac{Div(C) + Dis(E)}{2} \tag{3.20}$$

3.6.3. Precision und Recall

In analoger Weise wie mit der Klassifizierung in relevanten und irrelevanten Dokumenten im Information Retrieval, lässt sich die Erkennung von Duplikaten als binäres Entscheidungsproblem formulieren, in der ein Klassifizierer die Kandidatenpaare in positive

[10] Im Allgemeinen erhalten wir durch die Invertierung einer Funktion eine Relation, die Umkehrrelation.

	Klass. als Duplikat	Klass. als Nicht-Duplikat
Ist Duplikat	richtig positiv (RP)	falsch positiv (FP)
Ist Nicht-Duplikat	falsch negativ (FN)	richtig negativ (RN)

Tabelle 3.6.: Konfusionsmatrix für das Duplikatsproblem

und negative Beispiele einteilt, in Abhängigkeit davon, ob diese vom Klassifizierer als Duplikate respektive Nicht-Duplikate identifiziert wurden. Die vom Klassifizierer getroffene Entscheidung kann in einer sogenannten Konfusionsmatrix dargestellt werden, die vier Kategorien enthält: Richtig positiv (RP) sind die positiven Kandidatenpaare, die korrekt als positiv erkannt wurden. Falsch positiv (FP) bezieht sich auf die negativen Kandidatenpaare, die inkorrekt als positiv klassifiziert wurden. Richtig negativ (RN) korrespondiert mit den negativen Kandidaten, die korrekt als negativ erkannt wurden und zu guter Letzt beziehen sich die falsch Negativen (FN) auf die positiven Kandidatenpaare, die inkorrekt als negativ klassifiziert wurden. Eine Konfusionsmatrix für das Duplikatsproblem ist in Tabelle 3.6 dargestellt.

Bei der Wahl einer adäquaten Metrik für das paarweise Klassifizierungsproblem der Objektkonsolidierung sind wir vor dem Problem gestellt, dass im Allgemeinen die zu deduplizierenden Datenmengen in ihrer Klassenverteilung stark verzerrt sind – der Anteil an Duplikatspaaren beträgt in der Regel weniger als 1% von der Menge der Nicht-Duplikate. Die *Accuracy*, mit der der Prozentsatz aller korrekt klassifizierten Paare, Duplikate und Nicht-Duplikate (*richtig Positive* und *richtig Negative*), gemessen wird, hat sich diesbezüglich als keine allzu aussagekräftige Metrik erwiesen. In einem Szenario mit nur 1% an Duplikaten erhalten wir mit einem trivialen Klassifzierer, der alle Paare als Nicht-Duplikate klassfiziert, eine *Accuracy* von 99%. Aus diesem Grund wollen wir zur Evaluation der Objektkonsolidierung die beiden aus dem Information Retrieval bekannten Metriken, *Precision* und *Recall*, heranziehen und auf unser Konsolidierungsproblem übertragen. Im Information Retrieval dienen diese beiden Maße zur Beschreibung der Güte eines Suchergebnisses. Der *Recall* gibt dabei die Vollständigkeit eines Suchergebnisses an, d.h. der Anteil, der gefundenen relevanten Dokumente zu der Gesamtanzahl der relevanten Dokumente. Bei der *Precision* wird der Anteil der gefundenen relevanten Dokumente zu der Gesamtanzahl der gefundenen Dokumente betrachtet, wodurch der Genauigkeit eines Suchergebnisses Ausdruck verliehen wird. Übertragen wir diese beiden Metriken auf das Konsolidierungsproblem, dann misst der *Recall* den Anteil aller als korrekt klassifzierten Duplikate zu der Gesamtmenge der tatsächlichen Duplikate. Die *Precision* hingegen bezieht sich auf den Anteil der korrekt klassifzierten Duplikate zu allen als Duplikate klassifizierten Paaren.

Wir wollen nun diese beiden Metriken bezüglich der Objektkonsolidierung in einem etwas formaleren Kontext betrachten. Sei dazu $f : X \times X \rightarrow \{Positiv, Negativ\}$ die Funktion des Klassifizierers bezüglich derer die Kandidatenpaare in eine der beiden Kategorien als positiv (Duplikat) oder negativ (Nicht-Duplikat) klassifiziert werden. Des Weiteren bezeichnen wir mit M^{pos} die Menge aller tatsächlichen Duplikatspaare in der

Datenmenge und mit M_f^{pos} die von dem Klassifizierer f als Duplikate klassifizierten Paare. Dann definieren sich *Recall* und *Precision* wie folgt:

$$Recall = \frac{\left|M^{pos} \cap M_f^{pos}\right|}{|M|} \tag{3.21}$$

$$Precision = \frac{\left|M^{pos} \cap M_f^{pos}\right|}{\left|M_f^{pos}\right|} \tag{3.22}$$

Idealerweise sollte sich eine Qualitätsbeurteilung des Klassfizierers auf beide Maße separat stützen, da keine für sich alleinstehende Metrik existiert, die beide in einem adäquaten Maß kombiniert. Vielmehr stehen beide Maße in negativer Korrelation zueinander, so dass sich die Erhöhung des einen Wertes auf die Senkung des anderen Wertes auswirkt. Um dennoch beide Maße gemeinsam betrachten zu können, verwenden wir mit dem *F-Measure* ein weiteres Maß aus dem Information Retrieval, die den *Trade-Off* zwischen den beiden Maßen in Zusammenhang bringt. Bei dem *F-Measure* handelt es sich um das harmonische Mittel von *Precision* und *Recall*:

$$F\text{-}Measure = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.23}$$

Um diese beiden Metriken auf die Evaluation einer gefundenden Clusterlösung anwenden zu können, müssen wir aus dieser zunächst einmal die entsprechende Menge der Duplikatspaare konstruieren. Zu diesem Zweck wollen wir die von dem Konsolidierungsframework gefundenen Clusters als Menge von Paaren auffassen, d.h. ein Cluster der sich beispielsweise aus den Referenzen $C = \{a, b, c\}$ zusammensetzt besteht aus den drei Duplikatspaaren $\{(a, b), (a, c), (b, c)\}$.

Beispiel 3.7 Wir wollen die Berechnung von *Precision* und *Recall* anhand eines Beispiels demonstrieren. Hierzu betrachten wir wieder die korrekte und eine vom Konsolidierungsframework gefundene Clusterlösung. Sei also folgende die korrekte Clusterlösung:

$$\begin{aligned} C_1 &= \{a_1, a_2, a_3\} \\ C_2 &= \{b_1, b_2, b_3\} \end{aligned} \tag{3.24}$$

und eine vom Konsolidierungsframework gefundene Clusterlösung:

$$\begin{aligned} C_1 &= \{a_1, a_2, a_3, b_1\} \\ C_2 &= \{b_2, b_3\} \end{aligned} \tag{3.25}$$

	Klass. als Duplikat	Klass. als Nicht-Duplikat
Ist Duplikat	4	2
Ist Nicht-Duplikat	3	9

Tabelle 3.7.: Die Konfusionsmatrix für das betrachtete Beispiel

Daraus erhalten wir dann die Menge der tatsächlich vorhandenen Duplikate $M^{pos} = \{(a_1, a_2), (a_2, a_3), (a_1, a_3), (b_1, b_2), (b_2, b_3), (b_1, b_3)\}$ und die vom Klassifizierer als Duplikate erkannte Menge $M_f^{pos} = \{(a_1, a_2), (a_2, a_3), (a_1, a_3), (a_1, b_1), (a_2, b_1), (a_3, b_1), (b_2, b_3)\}$. Diese resultieren in eine *Precision* von 4/7 und einem *Recall* von 4/6. Die korrespondierende Konfusionsmatrix für dieses Beispiel ist in Tabelle 3.7 angegeben.

4. Experimentelle Evaluation

In diesem Kapitel wollen wir in verschiedenen Experimenten untersuchen, ob die graphbasierte Objektkonsolidierung einen Vorteil gegenüber einem rein attributbasierten Konsolidierungsverfahren aufzeigt. Zu diesem Zweck wollen wir die unterschiedlichen Implementierungen des Frameworks auf den DBLP-Daten evaluieren. An dieser Stelle sei nochmals betont, dass Chen's Framework auf einem attributbasierten Verfahren aufsetzt und in diesem Sinne keine Alternative zu letzterem darstellt, sondern vielmehr als eine generische Ergänzung zu einem attributbasierten Verfahren zu verstehen ist. Da die graphbasierte Objektkonsolidierung insbesondere für Domänen konzipiert ist, in denen nur sehr wenige Informationen bzw. Attribute über die Objekte zur Verfügung stehen, soll das in unseren Versuchsreihen eingesetzte attributbasierte Verfahren ausschließlich den Autorennamen als Attribut verwenden.

In Domänen mit wenigen Attributinformationen über die Objekte gestaltet sich das Konsolidierungsproblem als besonders schwierig. Dies begründet sich auf der Tatsache, dass in Domänen, in denen reichhaltige Attributinformationen über die Objekte zur Verfügung stehen, diese auch effektiv für die Deduplikation eingesetzt werden können, so dass eine zusätzliche Performanzsteigerung durch die Analyse der Beziehungen wenn überhaupt eher gering ausfallen sollte. Von diesem Standpunkt aus betrachtet stellt sich die Frage inwiefern die Fähigkeit des Frameworks zur Konsolidierung von dem Grad der Beziehungen abhängt. Das heißt, wir wollen in einem unserer Experimente der Frage nachgehen, ob der Grad der Koautorenschaftsbeziehung einen Einfluß auf die Performanz hat. Für die Evaluation bedienen wir uns dabei einer in der Datenbereinigung praktizierten Standardtechnik, bei der durch die Einführung von Fehlern in den Daten ein bestimmter Grad an Ungewissheit herbeigeführt wird. Diesbezüglich interessieren wir uns in einer der Versuchsreihen, wie sich eine Erhöhung des Fehlerprozentsatzes in den Daten auf die Performanz des Frameworks auswirkt. Unsere Experimente sind dabei in erster Linie darauf ausgerichtet unsere Verfahren mit denen von Chen et al. vorgeschlagenen Verfahren hinsichtlich der Güte einer gefundenen Clusterlösung zu vergleichen.

Zur Evaluation des Frameworks wollen wir die Performanz der Implementierungen mit den in Kapitel 3.6 eingeführten Metriken messen, wobei wir sowohl die *Entropie*-basierten Metriken zur direkten Evaluation einer gefundenen Clusterlösung, als auch die traditionellen Information Retrieval Metriken, *Precision* und *Recall*, verwenden wollen. Bei letzterem bezieht sich die Evaluation auf ein paarweises Klassifizierungsproblem bei der die als Duplikate klassifzierten Kandidatenpaare aus der gefundenen Clusterlösung hervorgehen (vgl. hierzu Kapitel 3.6.3). Eine qualitative Bewertung der Verfahren sollte dabei nicht nur auf der Güte des Ergebnisses basieren, sondern für eine Gesamtbeurtei-

lung muss auch die Effizienz – sprich die Laufzeit – der Verfahren als ein wesentlicher Aspekt mit in die Beurteilung einfließen.

Im Idealfall wäre es wünschenswert, die Evaluation unseres Frameworks auf die gesamten DBLP-Daten anzuwenden, um alle die darin enthaltenen, mehrdeutigen Autorenreferenzen aufzulösen. Diesbzüglich sehen wir uns jedoch zwei Problemen gegenüber: Zum einen benötigen wir zur Evaluation das Wissen über die unbekannte Resolutionsfunktion, d.h. der Abbildung der Autorenreferenzen auf ihre korrespondierenden Realwelt-Entitäten. Zum anderen ist in Anbetracht des Umfangs an Publikationen und der damit einhergehenden Komplexität des Problems eine Evaluation nicht ohne weiteres zu bewerkstelligen. In den folgenden beiden Abschnitten nehmen wir uns dieser Probleme an, indem wir eine geeignete Testumgebung für unsere Versuchsreihen definieren. Das Kapitel schließt dann mit der Präsentation und der Interpretation der erzielten Resultate ab.

4.1. Konstruktion der Experimentdaten

4.1.1. DBLP-Daten

Die digitale Bibiliothek „Digital Bibliography & Library Project“ (DBLP) der Universität Trier besteht aus einer Sammlung von bibliographischen Informationen zu den wichtigsten Informatikjournale und Tagungsbänden aus vielen Teilgebieten der Informatik. Die DBLP-Datensammlung hat sich dabei als ein wichtiger Informationsdienst etablieren können. Zum Zeitpunkt der Experimente indizierte der DBLP-Server mehr als 765.000 Publikationen und enthielt darüber hinaus mehrere Tausend Hyperlinks zu den persönlichen Homepages von Wissenschaftlern. Intern werden die einzelnen Metadaten auf Basis einer *eXtensibe Markup Language* (XML) definierten Sprache verwaltet, die mittlerweile zu einer 300 MByte großen XML-Datei aufgebaut wurde, von der eine Untermenge Gegenstand unserer Experimentierumgebung sein soll.

4.1.2. Konstruktion der Daten

Die DBLP-Datensammlung, die in Form einer XML-Datei vorliegt, enthält verschiedene Arten von Publikation, wie z.B. Artikel, Bücher, Inproceedings, wobei wir uns beim Parsen auf die Extraktion von Artikeln beschränkt haben. Für diese Einschränkung hat es keinen besonderen Grund gegeben, sondern diese hat sich vielmehr aus den technischen Gegebenheiten heraus ergeben, weil damit eine einfachere Realisierung in Bezug auf das Parsen[1] der Datei verbunden war.

Die Einschränkung auf eine Publikationsart stellt keinen Abbruch an unseren Untersuchungen dar, da wir – wie eingangs erwähnt – zur Evaluation des Frameworks

[1] Das Parsen der XML-Datei haben wir mit SAX realisiert, einer ereignisgesteuerten API, die bei speziellen Parsing Ereignissen Meldungen an einen Handler sendet.

lediglich eine Untermenge aus der Sammlung betrachten werden. Aus jeder Publikation der DLBP-Datensammlung extrahieren wir die Koautorenliste und konstruieren aus dieser für jeden Autorennamen einen eigenen Datensatz, der sich aus den folgenden Attributen zusammensetzt: Autorenname und Koautorenliste. Die Titelinformation kann dabei optional als zusätzliches Attribut in den Datensatz aufgenommen werden. Die Attributinformationen werden für das rein attributbasierte Verfahren in der Phase 1 des Konsolidierungsframeworks verwendet. Die Ähnlichkeitsfunktion anhand dessen eine Ähnlichkeit zwischen den Referenzen festgestellt wird basiert dabei auf der Textähnlichkeit, auf die wir in dem Abschnitt der Implementierung näher eingehen werden. Im Allgemeinen können wir davon ausgehen, dass je mehr Attribute einer Referenz vorliegen, desto leichter lassen sich Duplikate als solche identifizieren. Nachfolgend ein Fragment der XML-Datei.

```
<article mdate="2003-12-02" key="journals/apin/BridgesHWKS99">
<author>Susan Bridges</author>
<author>Julia E. Hodges</author>
<author>Bruce Wooley</author>
<author>Donald Karpovich</author>
<author>George Brannon Smith</author>
<title>Knowledge Discovery in an Oceanographic Database.</title>
<pages>135-148</pages>
<year>1999</year>
<volume>11</volume>
<journal>Appl.Intell.</journal>
<number>2</number>
<url>db/journals/apin/apin11.html#BridgesHWKS99</url>
</article>
```

Listing 4.1: XML-Fragment aus der DBLP-Datensammlung

4.1.3. Synthetisieren der Daten

Im Unterschied zu *CiteSeer* tritt das Problem mit mehrdeutigen Autorennamen in den DBLP-Daten nicht dermaßend gravierend auf, so dass wir von größtenteils bereinigten Autorennamen ausgehen können. Zum Zwecke der Evaluation wollen wir aufgrund dieser Annahme den Autorennamen als eindeutigen Identifzierer betrachten und künstliche Fehler in die Autorennamen hinzufügen. In [OLKM05] werden dazu für Namen (z.B. „Ji-Woo K. Li“) mögliche Fehlertypen vorgeschlagen, die sich aus Abkürzungen („J. K. Li“), Vertauschungen („Li, Ji-Woo K.“), Rechtschreibfehler („Ji-Woo K. Lee“), der Konkatenation von Vor- und Nachnamen („Jiwoo K. Li“) oder sich aus einer Kombination von diesen ergeben. Für unsere Experimentierumgebung haben wir uns dazu entschieden, die Autorennamen in den Publikationen auf die Initialen des Vornamens bzw. mittlerer Namen und dem Nachnamen zu vereinfachen. Die Wahl für diese Vereinfachung liegt in der bibliographischen Domäne begründet, da dieses Namensformat häufig in Publikationen vorzufinden ist. Da im Allgemeinen reichhaltige Namensinformationen zur Disambiguierung der Autoren nützlich sein können, stellen unzureichende Namensinformationen einen guten Ausgangspunkt für die Evaluation des Frameworks dar. Durch diese Vereinfachung erhalten wir also viele mehrdeutige Autorennamen in

den Daten. Es bleibt abzuwarten, wie sich der Grad an Ungewissheit in den Daten auf die Effektivität des Frameworks niederschlägt und ob durch die zusätzliche Ausbeutung der Beziehungen der Zuwachs an Fehlern kompensiert werden kann.

4.2. Experimentelle Testumgebung

Während wir im vorherigen Abschnitt beschrieben haben, wie die Experimentdaten für die Evaluation des Frameworks aussehen sollen, folgt in diesem Abschnitt eine Beschreibung der Konfiguration des Frameworks. Wir beginnen diese mit einer Beschreibung des attributbasierten Verfahrens, welches wir in der ersten Phase des Frameworks verwenden und auf das die verschiedenen Implementierungen des Frameworks aufsetzen. Im darauffolgenden Abschnitt geben wir dann einige Implementierungdetails zu den beiden in Kapitel 3.4 und 3.5 vorgestellten Verfahren an, insbesondere was die Wahl der offenen Parameter anbetrifft. Die eigentliche Beschreibung der Experimente ist Inhalt des letzten Abschnitts.

4.2.1. Attributbasiertes Verfahren: ATTR

Einer der häufigsten Fehlerquellen für die Entstehung von Duplikaten sind typografische Variationen in textuellen Daten. Aus diesem Grund basieren viele der traditionellen Ansätze der Objektkonsolidierung auf Techniken zur Bestimmung der Ähnlichkeit in Textdaten, die aus den Beschreibungen der Realwelt-Entitäten hervorgehen. Um zu entscheiden, ob zwei Referenzen dieselbe Entität repräsentieren, ist also ein Ähnlichkeitsmaß erforderlich, mit dem die Ähnlichkeit von textuellen Attributwerten zwischen zwei Referenzen bestimmt werden kann. In der Literatur gibt es dazu umfangreiche Arbeiten, die sich mit genau diesem Problem befassen, Maße für approximative Textähnlichkeit zu definieren, die unter anderem zur Deduplikation von Objekten eingesetzt werden können. In unserem Falle stehen uns für die Konsolidierung der Autorenreferenzen zwei textuelle Attribute zur Verfügung: die Autorennamen selbst und die Koautorenliste der Autoren. Neben den zahlreichen Möglichkeiten Ähnlichkeitsmaße bzw. Entfernungsmaße zu definieren, verwenden wir mit dem *Jaro-Winkler* ein auf der Editier-Distanz basiertes Distanzmaß. Mit diesem Maß wollen wir die Distanz zwischen den Autorennamen zweier Referenzen bestimmen. Ein anderes denkbare attributbasierte Verfahren könnte die Koautorenliste als zusätzliches Attribut aufnehmen. In diesem Fall können wir mit der hybriden *SoftTF-IDF*[2] ein Token-basiertes Distanzmaß verwenden. Während wir beim *Jaro-Winkler* Maß die Distanz zwischen den Autorennamen zweier Referenzen berechnen, können wir das *SoftTF-IDF* Maß dazu verwenden die Distanz zwischen den Koautoren zweier Referenzen zu bestimmen, d.h. anstatt der Distanz *dist*(„Jeffrey David Ullman“, „J.D. Ullman“), berechnen wir die Distanz *dist*(Koautoren(„Jeffrey David

[2] Bei der *SoftTF-IDF* Metrik werden in ihrer Schreibweise ähnliche Tokens als identisch betrachtet. Dazu wird das *TF-IDF* Ähnlichkeitsmaß für Tokens unter Verwendung eines sekundären Textähnlichkeitsmaßes (z.B. *Jaro-Winkler*) um ein approximatives Token Matching erweitert.

Ullman“), Koautoren („J.D. Ullman“)). Cohen et al. haben in [CRF03] gezeigt, dass diese Maße in Bezug auf das Matching-Problem bei Namen eine gute Performanz aufweisen. Details zu diesen Metriken finden sich ebenfalls in [CRF03]. Für unsere Experimente soll das in Phase 1 eingesetzte Grundverfahren lediglich auf den Autorennamen basieren. In diesem Grundverfahren – nennen wir es ATTR – werden die bezüglich des Distanzmaßes als ähnlich identifizierten Autorenreferenzen in denselben *virtuellen Konsolidierungsteilgraphen* partitioniert und die resultierende Partitionierung als Clusterlösung übernommen. Das heißt die Konsolidierungsteilgraphen werden nicht weiter partitioniert, sondern direkt als die Entiätenclusters übernommen. Diese Clusterlösung dient uns später in den Versuchsreihen als Referenzbasis, mit der wir die Gesamtperfomanz des Konsolidierungsalgorithmus vergleichen wollen. Nach unseren bisherigen Annahmen aus Kapitel 3.3.3 sollte das Grundverfahren ATTR idealerweise eine optimale *Entitäten Dispersion* erzielen, wobei dies in den Versuchsreihen nicht zwingend erforderlich sein muss.

4.2.2. Framework Implementierungen: LKW-SM u. PPR-NG

Das primäre Ziel der Evaluation soll es sein die von Chen et al. vorgeschlagenen Verfahren mit den unseren hinsichtlich ihrer Performanz und Effizienz zu vergleichen. In den beiden folgenden zwei Unterabschnitten wollen wir einige Details zu den Implementierungen der beiden in Kapitel 3.4 und 3.5 beschriebenen Verfahren angeben und die für die Experimente erforderlichen Parameter festlegen. Zur Bestimmung der geeigneten Parameter für die Verfahren haben wir jeweils den besten Wert verwendet, den wir durch Justierung der Schwellenwerte in den Experimenten erhalten haben. Bei der Wahl der Parameter für die Verfahren von Chen et al. haben wir uns überwiegend an deren Arbeit orientiert. Zwar haben wir bei der Parameterwahl weder alle Kombinationen noch Möglichkeiten der Schwellenwerte ausprobiert, aber aufgrund der empirischen Untersuchungen in [CKM05] konnten wir diese weitesgehend eingrenzen. Aus diesem Grund wollen wir von der Annahme ausgehen, dass die auf diese Weise getroffene Wahl der Parameter zu guten Ergebnissen führen sollten.

Chen's Framework Implementierung LKW-SM

Chen's et al. Implementierung des Frameworks besteht aus dem in Phase 2 verwendeten Verfahren der *L-kürzesten Wege* (LKW) und dem in Phase 3 eingesetzten *spektralem* Clustering Verfahren von Shi und Malik (SM). Der Algorithmus zur Bestimmung der *L-kürzesten Wege* haben wir dabei aus einer von Kalashnikov et al. publizierten Arbeit [KMC05] entnommen, in welcher mit der Referenz Disambiguierung ein der Objektkonsolidierung verwandtes Probem behandelt wird. Für die Wahl des Parameters L ist wichtig zu beachten, dass das Verfahren der *L-kürzesten Wege* exponentiell in L ist und somit eine bereits geringe Erhöhung des Parameters zu einer drastischen Verschlechterung der Laufzeit führt. In den empirischen Untersuchungen aus [CKM05] hat sich die Festlegung auf den Parameterwert $L = 4$ als ein guter Kompromiss zwischen der Performanz und der Effizienz erwiesen.

Für das *spektrale* Clustering Verfahren haben wir die von Luigi Dragone[3] geschriebene Implementierung verwendet, die ursprünglich als *spektraler* Clustering Algorithmus für das WEKA[4] Framework konzipiert war. Die Implementierung basiert auf dem Java Package COLT[5], das eine Open Source Bibliothek für effiziente Datenstrukturen und Algorithmen aus der Linearen Algebra zur Verfügung stellt und ursprünglich am Forschungszentrum CERN in Genf entwickelt wurde. Bei der Analyse dieses Verfahrens in Kapitel 3.5.2 haben wir erläutert, dass der *spektrale* Algorithmus eine Partitionierung des Netzwerks durch eine fortlaufende Bisektion erzielt und aus diesem Grund die Angabe eines vordefinierten Schwellenwertes erfordert, anhand dessen die Entscheidung zur weiteren Partitionierung (Bisketion) getroffen wird. Dies haben wir als Nachteil des Verfahrens identifiziert, da wir im Vorhinein nicht wissen, aus wie vielen unterschiedlichen Entitäten sich ein Cluster zusammensetzt. In der Arbeit von Chen et al. wird davon ausgegangen, dass der Schwellenwert von einem Domänenanalysten vorgegeben wird, aber idealerweise aus den Daten gelernt werden sollte. Es wurde in deren Arbeit auch der Effekt des Schwellenwerts τ auf die gefundene Clusterlösung untersucht. In diesem Zusammenhang wurde festgestellt, dass der Parameter τ den natürlichen *Trade-off* zwischen *Cluster Diversität* und *Entitäten Dispersion* widerspiegelt, was wie folgt einzusehen ist: Falls für den Schwellenwert τ ein kleiner Wert angenommen wird, so wirkt sich dies positiv auf die *Entitäten Dispersion* aus, da die Partitionierung der *virtuellen Konsolidierungsteilgraphen* aufgrund des niedrigen Schwellenwertes in einer geringen Anzahl an Clusters resultiert. Die sich ergebende Clusterlösung ist also der Lösung des attributbasierten Verfahrens umso ähnlicher, je kleiner der Paramter τ gewählt wird. Auf der anderen Seite führt ein großer Wert für τ zu einer größeren Anzahl an Clusters, da aufgrund des hohen Schwellenwertes mehr Bisektionen in den *Konsolidierungsteilgraphen* durchgeführt werden. Letzteres führt zwar zu einer Verbesserung der *Cluster Diversität*, aber zu einer Verschlechterung der *Entitäten Dispersion*. Diese Erkenntnis hat uns zu den Überlegungen geführt, dass wir anstatt eines fest eingestellten Parameters das *spektrale* Verfahren für mehrere Splitpunkte wiederholen lassen wollen und die resultierenden Clusterlösungen gegeneinander abwägen. Zu diesem Zweck haben wir das Intervall $[0, 2]$ des *normalisierten Schnitts* in gleichmäßig verteilten Splitpunkten aufgeteilt. In den ersten Versuchsreihen hat sich dann gezeigt, dass die jeweils besten Splits im Intervall zwischen 0,9 und 1,1 liegen. Diese Beobachtung deckt sich auch mit dem von Chen et al. vermuteten Zusammenhang zwischen dem Schwellenwert τ und dem *Trade-off* von *Cluster Diversität* und *Entitäten Dispersion*. Aus diesem Grund wollen wir uns im weiteren Verlauf auf diese Splitpunkte beschränken. Da der *spektrale* Algorihmus sehr effizient arbeitet und wir uns darüber hinaus auf das Intervall beschränkt haben, wollen wir den sich ergebenden Effizienzverlust in Kauf nehmen.

[3] http://www.luigidragone.com/datamining/spectral-clustering.html

[4] http://www.cs.waikato.ac.nz/ml/weka/. WEKA steht für Waikato Environment for Knowledge Analysis und bezeichnet ein Open Source Java-Framework, das eine Sammlung von Algorithmen aus dem Bereich des Maschinellen Lernens für Data Mining Aufgaben zur Verfügung stellt.

[5] http://hoschek.home.cern.ch/hoschek/colt/.

Unsere Framework Implementierung PPR-NG

Bei unserer Implementierung des Frameworks haben wir für die Phase 2 den *personalisierten PageRank* Algorithmus (PPR) verwendet und in Phase 3 das von Newman und Girvan vorgeschlagene hierarchische Verfahren (NG) zur Identifizierung von Communities. Für die Implementierung des *personalisierten PageRanks* Algorithmus haben wir den Algorithmus aus 2.1 zugrunde gelegt, wobei wir noch für den Dämpfungsfaktor d einen Wert $0 < d < 1$ zu wählen haben. Wie wir bereits in Kapitel 3.4.3 ausführlich beschrieben haben, gibt der Dämpfungsfaktor in dem stochastischen Prozess des *personalisierten PageRanks* die Wahrscheinlichkeit an, mit die sich der *Random Surfer* zurück zum Hauptknoten teleportiert, um von dort aus seinen Pfad wieder aufzunehmen. Für diesen Dämpfungsfaktor finden sich in der Literatur die unterschiedlichsten Vorschläge, die jedoch überwiegend im Intervall zwischen 0,85 und 0,9 liegen. In unseren Experimenten haben wir für den Dämpfungsfaktor den Wert $d = 0,85$ angenommen, der ursprünglich von Page und Brin in [PBMW98] vorgeschlagen wurde. Es sei noch erwähnt, dass die Wahl des Dämpfungsfaktors sowohl einen Einfluss auf das Konvergenzverhalten als auch auf die Genauigkeit des *PageRank* Algorithmus hat. Je kleiner der Wert für den Dämpfungsfaktor gewählt wird, desto langsamer ist die Konvergenzrate, womit sich folglich auch die Anzahl der benötigten Iterationen erhöht, um die Abbruchbedingung des *PageRanks* Algorithmus zu erreichen [LM03].

Für das hierarchische Verfahren von Newman und Girvan zur Identifizierung der Community Struktur in den *virtuellen Konsolidierungsteilgraphen* sind keine weiteren Parameter erforderlich. Vielmehr sollten wir nach Newman und Girvan eine gute Partitionierung dadurch erhalten, indem wir die durch horizontale Schnitte des Dendrogramms induzierten Partitionen mittels der Modularitätsfunktion Q gegeneinander abwägen. Die Bestimmung der nach der Modularitätsfunktion besten Partition ist kein zusätzlicher Schritt, sondern integraler Bestandteil des Algorithmus.

4.2.3. Beschreibung der Experimente

In diesem Abschnitt wollen wir den Aufbau unserer Versuchsreihen beschreiben, in denen zum einen die Performanz des Frameworks evaluiert und zum anderen die Effizienz der Verfahren als zweiter wichtiger Aspekt untersucht werden soll. Diesbezüglich sehen wir uns jedoch einigen Problemen gegenüber. Um beispielsweise die Performanz der graphbasierten Objektkonsolidierung demonstrieren zu können, sind idealerweise mehrere Relationen zwischen den zu deduplizierenden Autorenreferenzen notwendig, von denen uns aber mit der Koautorenschaft nur eine Beziehung in den Daten zur Verfügung steht. Des Weiteren sehen wir uns dem Problem gegenüber, dass durch die Vielzahl an Autoren eine große Streuung der Referenzen in den DBLP-Daten zu erwarten ist, so dass die Menge an Beziehungen eventuell zu schwach ausfallen könnte und somit die Resultate zu falschen Schlussfolgerungen führen. Ein weiteres Problem ergibt sich aus dem Umfang der DBLP-Daten – zum heutigen Stand auf 300 MByte angewachsen – so dass wir nicht mehr in der Lage sind, die Evaluation mit den gesamten DBLP-Daten durchzuführen, sondern uns auf einen Teilausschnitt der Daten beschränken müssen. In

der ersten Versuchsreihe wollen wir das zuvor vermutete Problem der unter Umständen zu gering ausfallenden Beziehungen ignorieren und die graphbasierte Konsolidierung auf die gesamten Autorenreferenzen im betrachteten Teilausschnitt anwenden. Im darauffolgenden Abschnitt beschreiben wir dann einen zweiten Versuchsaufbau, bei der wir dem Problem der schwach ausfallenden Beziehungen entgegenwirken, indem wir die Konsolidierung auf einen einzelnen Autorennamen fokussieren und um diesen den Koautorengraphen konstruieren.

Versuchsreihe „Vollständige Konsolidierung“

Mit der Versuchsreihe „Vollständige Konsolidierung“ beziehen wir uns auf die Konsolidierung der gesamten Autorenreferenzen, die in dem betrachteten Teilausschnitt der DBLP-Daten vorkommen. Die Versuchsreihe wollen wir dabei für unterschiedliche große Probleminstanzen durchführen, wobei sich die jeweilige Problemgröße anhand der Anzahl der betrachteten Autorenreferenzen ergibt. Aus den Publikationen der DBLP-Daten werden dazu die Autorenreferenzen extrahiert und aus der Koautorenschaftsbeziehung und den Ähnlichkeitskanten der *Referenzgraph* konstruiert auf dem der Konsolidierungsalgorithmus angwendet wird. Nach unserer Annahme sind die DBLP-Daten im Gegensatz zu den Daten in *CiteSeer* weitesgehend bereinigt, so dass wir die Autorennamen als eindeutige Identifizierer betrachten wollen. Zum Zwecke der Evaluation werden die Autorennamen in den DBLP-Daten durch die in Abschnitt 4.1.3 vorgeschlagene Vereinfachung der Autorennamen künstlich modifziert. Dem Konsolidierungsframework sind die Autorenreferenzen also nur durch den vereinfachten Autorennamen als einziges Attribut zugänglich, was die Disambiguierung mehrdeutiger Referenzen erschwert. Dies begründet sich auf der Tatsache, dass durch die Vereinfachung der Autorennamen dem attributbasierten Verfahren nur sehr wenige und ungenügende Informationen über die beobachteten Referenzen zur Verfügung stehen. Die Versuchsreihe „Vollständige Konsolidierung“ stellt somit ein adäquates Szenario für die Evaluation des Konsolidierungsframeworks dar.

In dieser Versuchsreihe sind wir in erster Linie daran interessiert, die Performanz zwischen den von Chen et al. vorgeschlagenen Verfahren mit den unseren hinsichtlich der in Kapitel 3.6 eingeführten Metriken zu vergleichen. Als Referenzbasis dient jeweils die Clusterlösung des attributbasierten Verfahrens, so dass wir neben dem direkten Vergleich der Implementierungen auch den Vergleich zu einem rein attributbasierten Verfahren anstellen können. Auf die Untersuchung der Effizienz bzw. der Laufzeit gehen wir am Ende in einem separaten Abschnitt ein.

Versuchsreihe „Fokussierter Autorenname“

Die weiter oben dargestellte Versuchsreihe beschreibt das eigentliche Konsolidierungsproblem, wie es in der Praxis vorzufinden ist. Diesbezüglich sehen wir uns jedoch der Problematik gegenüber, dass durch die Betrachtung eines kleinen Teilausschnitts der Daten die Koautorenschaftsbeziehung sehr spärlich ausfallen und – zumindest was unsere Implementierung des Frameworks anbetrifft – zu einem schlechten Ergebnis führen

	Experiment Smith	Experiment Chen
Fokussierter Autorenname	Smith	Chen
Artikelanzahl	22584	22584
Fokussierte Autorenreferenzen	146	336
Phase 1	ATTR	ATTR
Implementierung LKW-SM	$L = 4; \tau = \{0,9; 1,0; 1,1\}$	$L = 4; \tau = \{0,9; 1,0; 1,1\}$
Implementierung PPR-NG	d=0,15; I=30	d=0,15; I=30

Tabelle 4.1.: Grundeinstellungen für die Versuchsreihe „Fokussierter Autorenname“

könnte. Diese Problematik wird zusätzlich dadurch verstärkt, dass es viele Autoren in den Daten gibt, die alleine publizieren und folglich keine Beziehungen zu den anderen Autoren eingehen, wodurch der Einfluss der Koautorenschaftsbeziehung verringert wird und dies zu einer Beeinträchtigung der Effektivität der Verfahren führen könnte. Es stellt sich die Frage, wie eine sinnvollere Evaluation auf den Daten durchgeführt werden kann, bei der die Bedeutung der Koautorenschaftsbeziehung mehr in den Vordergrund gestellt wird. Die Idee besteht darin die Konsolidierung bzw. Disambiguierung auf einen einzelnen Nachnamen, wie z.B. dem Namen „Smith“ zu fokussieren. Anstatt also alle Referenzen zu disambiguieren, fokussieren wir die Konsolidierung auf die Autorenreferenzen mit demselben Nachnamen und nehmen den Rest der Referenzen als bereits konsolidiert an. Offensichtlich scheint dieses Problem schwieriger als die „Vollständige Konsolidierung“ zu sein, da im Gegensatz zu letzterem die Fokussierung auf einen einzelnen Nachnamen zu verhältnismäßg vielen Duplikaten führt. Umgekehrt können bei Betrachtung der gesamten Referenzen diese aufgrund der Vielseitigkeit der Nachnamen einfacher durch das attributbsierte Verfahren disambiguiert werden. Ausgehend von dieser Überlegung erwarten wir eine etwas niedrigere Gesamtperformanz der Verfahren, die insbesondere auf die Verschlechterung des attributbasierten Verfahrens zurückzuführen ist. Dies stellt aber kein eigentliches Problem dar, da unser Blickpunkt vielmehr auf den direkten Vergleich der verschiedenen Implementierungen des Frameworks ausgerichtet ist.

Um nun die Beziehung zwischen den zu disambiguierenden Referenzen herzustellen, wird um die fokussierten Autorenreferenzen herum ein Koautorennetzwerk unterschiedlicher Tiefen (die Koautoren und die Koautoren dieser Koautoren usw.) konstruiert. Die Information, die dann von der Koautorenschaftsbeziehung ausgeht, sollte dann ausreichen, um die Effektivität des Frameworks zu demonstrieren. In dieser Versuchsreihe des „Fokussierten Autorennamens“ wollen wir dann die von Chen et al. vorgeschlagenen Verfahren mit den unseren hinsichtlich der Qualtität ihrer gefundenden Clusterlösung vergleichen. In einfachen Worten gesagt, soll der Algorithmus dann herausfinden, welche „Smithe“ davon zusammengehören oder nicht und diese Autorenreferenzen in ihre korrespondierenden Entitätenclusters partitionieren. Zusätzlich wollen wir untersuchen, welchen Einfluss der Grad der Beziehungen auf die Performanz der Verfahren hat, indem wir das Experiment für unterschiedlichen Koautorenschaftstiefen wiederholen wollen. Bei

einer weiteren Betrachtung soll der Einfluss an der Erhöhung des Fehlerprozentsatzes untersucht werden, wobei wir die in Abschnitt 4.1.3 eingeführte künstliche Modifikation der Daten verwenden wollen. Die Vereinfachung der Autorennamen werden wir jedoch nicht – wie beim Experiment „Vollständige Konsolidierung“ – auf die gesamten Autorennamen anwenden, sondern orientiert sich vielmehr an den typischen Fehlerprozentsatz, wie dieser in Realwelt-Datensätzen vorhanden ist. Ausgehend davon führen wir unsere Experimente mit einem variierenden Fehlerprozentsatz ρ von $1\%, 5\%, 10\%, 15\%, 20\%, 25\%, 30\%$ durch. Wir erwarten, dass sich eine Erhöhung des Prozentsatzes negativ auf die Performanz des in der ersten Phase eingesetzten attributbasierten Verfahrens auswirkt. Es bleibt abzuwarten, wie sich der Grad an Ungewissheit in den Daten auf die Effektivität des Frameworks niederschlägt und ob durch die zusätzliche Ausbeutung der Beziehungen der Zuwachs an Fehlern kompensiert werden kann.

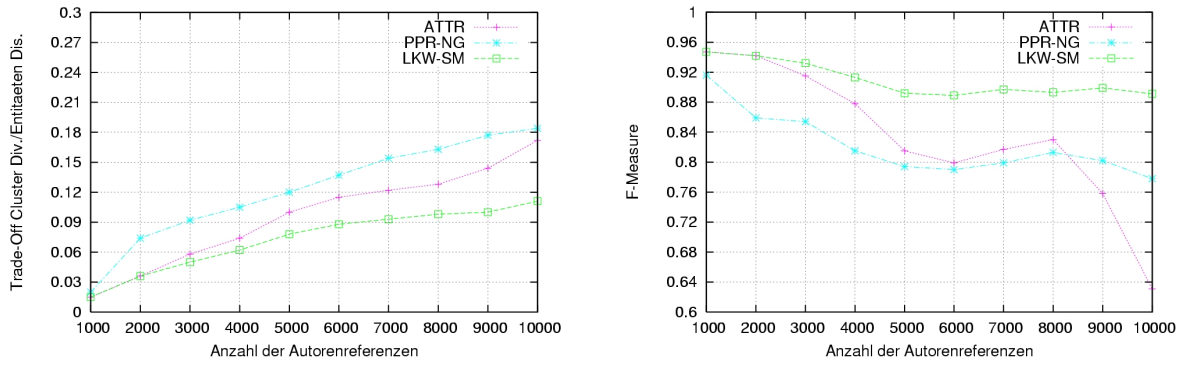
4.3. Ergebnisse

In diesem Abschnitt präsentieren wir die Ergebnisse aus den beiden zuvor beschriebenen Versuchsreihen „Vollständige Konsolidierung“ und „Fokussierter Autorenname“. Im Vordergrund steht dabei der direkte Performanzvergleich der verschiedenen Implementierungen des Frameworks, zu dessen Zweck die jeweils gefundene Clusterlösung mit den zuvor eingeführten Metriken evaluiert wird. Neben der direkten Gegenüberstellung der Implementierungen interessieren wir uns auch für den Vergleich mit dem attributbasierten Verfahren, das uns quasi als Referenzbasis dienen soll. Wir beginnen unsere Untersuchungen mit der Versuchsreihe „Vollständige Konsolidierung“ und präsentieren dann im nächsten Abschnitt die Ergebnisse aus der Versuchsreihe „Fokussierter Autorenname“. Abschließend wollen wir die Verfahren hinsichtlich ihrer Laufzeit untersuchen, wobei wir das Komplexitätsverhalten der Verfahren aus den einzelnen Phasen des Frameworks separat voneinander betrachten.

4.3.1. Direkter Vergleich „Vollständige Konsolidierung“

Zum direkten Vergleich der von Chen et al. vorgeschlagenen Verfahren mit den unseren Verfahren führen wir die Versuchsreihe in Abhängigkeit von der Anzahl der Autorenreferenzen durch, wobei wir diese auf 10.000 Referenzen beschränkt haben. Als Vergleichsbasis dient jeweils das attributbasierte Verfahren. Die Abbildung 4.1 zeigt die aus der Versuchsreihe gelieferten Ergebnisse unter Verwendung der (a) *Entropie*-basierten Metrik (3.20) und (b) der Information Retrieval Metrik (3.23).

Aus der Abbildung ist zunächst einmal ersichtlich, dass die Performanz des attributbasierten Verfahrens und somit die des Konsolidierungsframeworks mit Zunahme der Autorenreferenzen monoton abfällt. Diese Beobachtung lässt sich darauf zurückführen, dass je mehr Referenzen bei der Betrachtung berücksichtigt werden, umso größer der Anteil an Duplikaten in der Datenmenge wird. Damit geht einher, dass durch die Vereinfachung der Autorennamen (Initialen des Vornamens bzw. mittlerer Namen und vollständiger

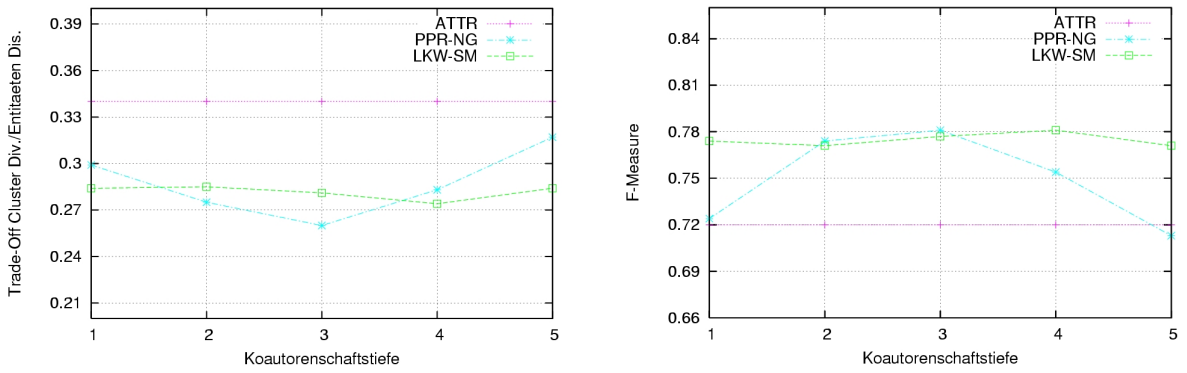


(a) (b)

Abbildung 4.1.: Die Kurve (a) zeigt den Verlauf der Cluster Div./Entitäten Dis., (b) zeigt den Verlauf des F-Measures.

Nachname) vermehrt Autorenreferenzen existieren, die sich alleine anhand des Autorennamens nicht eindeutig identifizieren lassen. Dieser Abfall in der Performanz zeigt sich insbesondere in der Abbildung 4.1 (b), in der bei einer Anzahl von mehr als 8.000 Referenzen das attributbasierte Verfahren einbricht. Bei der *Entropie*-basierten Metrik in Abbildung 4.1 (a) zeichnet sich die Verschlechterung der Performanz durch eine monoton anwachsende Kurve ab. Im direkten Performanzvergleich liegen die Werte unseres Verfahrens sogar noch unter denen des attributbasierten Verfahrens, während das von Chen et al. vorgeschlagene Verfahren in allen Messungen am besten abschneidet. Bei der *F-Measure* Metrik in Abbildung 4.1 (b) zeigt sich das Verfahren von Chen et al. bezüglich der Performanz als relativ konstant und zeigt in jedem Durchlauf eine Verbesserung gegenüber dem rein attributbasierten Verfahren ATTR. Im Gegensatz hierzu, zeigt sich unser Verfahren – wie schon bei der *Entropie*-basierten Metrik – bis zu einer Anzahl von 8.000 Referenzen als weniger performant als das attributbasierte Verfahren. Während jedoch die Performanz des attributbasierten Verfahrens danach stark abfällt, bleibt unser Verfahren relativ stabil. Anhand dieser Beobachtung können wir schlussfolgern, dass bei einer Erhöhung der Autorenanzahl unser Verfahren resistenter ist als das rein attributbasierte Verfahren.

Aufgrund dieser Resultate müssen wir unsere eingangs gestellten Erwartungen in Bezug auf die Performanz unseres Verfahrens zurückschrauben, da es sogar zum größten Teil zu einer Verschlechterung des attributbasierten Verfahrens geführt hat. Dies ist aber genau das Gegenteil von dem, was wir mit dem Algorithmus eigentlich bezwecken wollten. Es stellt sich nun die Frage, ob unser Algorithmus unter allen Bedingungen schlecht arbeitet, oder ob es in einer anderen Testumgbung bessere Resultate liefert. Eine Antwort darauf wollen wir im nächsten Abschnitt mit der Versuchsreihe „Fokussierter Autorenname“ liefern.



(a) (b)

Abbildung 4.2.: Einfluss der Koautorenschaftstiefe auf die Performanz. Die Kurve (a) zeigt den Verlauf der Cluster Div./Entitäten Dis., (b) zeigt den Verlauf des F-Measures.

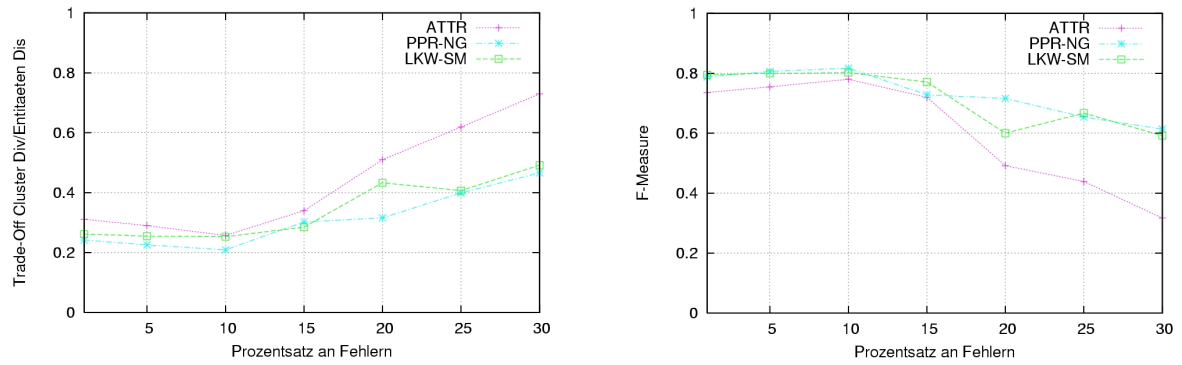
4.3.2. Direkter Vergleich „Fokussierter Autorenname“

In dem ersten Experiment hat sich gezeigt, dass unsere Implementierung des Frameworks zu einem schlechten Ergebnis führt. Da sich die von Chen et al. vorgeschlagenen Verfahren mit zunehmender Anzahl der Referenzen umso besser zeigen, haben sich unsere anfangs gestellten Bedenken bestätigt – dass durch die Betrachtung eines kleinen Teilausschnitts der Daten die Koautorenschaftsbeziehung sehr spärlich ausfällt und, zumindest was unsere Implementierung des Frameworks anbetrifft, zu einem schlechten Ergebnis führt. In diesem Abschnitt präsentieren wir die Ergebnisse aus der Versuchsreihe „Fokussierter Autorenname“. Die Einstellungen für diese Versuchsreihe sind in der Tabelle 4.1 festgehalten. Eine erste Zusammenfassung der Resultate für die Versuchsreihe „Fokussierter Autorenname“ findet sich in der Tabelle 4.2 wieder. Die erste Spalte gibt die jeweilige Tiefe der Koautorenschaft an, die wir für die Versuchsreihe auf Tiefe 5 beschränkt haben. In der zweiten Spalte sind für jede Tiefe die in Kapitel 3.6 eingeführten *Entropie*-basierten Metriken, *Cluster Diversität* und *Entitäten Dispersion*, angegeben, wobei wir für einen direkten Vergleich das arithmetische Mittel dieser beiden Werte herangezogen haben. Der Mittelwert soll den *Trade-Off* der beiden Maße widerspiegeln. Die prozentuale Abweichung nach der Spalte LKW-SM bzw. PPR-NG nimmt jeweils als Bezugspunkt die Werte in der Spalte ATTR. Der prozentuale Unterschied wurde wie folgt berechnet:

$$\begin{aligned} \Delta_1 &= (Metrik_{LKW-SM}/Metrik_{ATTR} - 1) \cdot 100 \\ \Delta_2 &= (Metrik_{PPR-NG}/Metrik_{ATTR} - 1) \cdot 100 \end{aligned} \tag{4.1}$$

Der direkte Vergleich der Implementierung LKW-SM und PPR-NG ergibt sich anhand der in der letzten Spalte angegebenen prozentualen Abweichung

$$\Delta_3 = (Metrik_{LKW-SM}/Metrik_{PPR-NG} - 1) \cdot 100 \tag{4.2}$$



(a) (b)

Abbildung 4.3.: Einfluss des Fehlerprozentsatzes auf die Performanz. Die Kurve (a) zeigt den Verlauf der Cluster Div./Entitäten Dis., (b) zeigt den Verlauf des F-Measures.

Zum Vergleich dazu haben wir im unteren Teil der Tabelle 4.2 nochmal für die Ergebnisse derselben Versuchsreihe die Information Retrieval Metriken, *Precision*, *Recall* und *F-Measure*, angegeben.

Auffällig ist zunächst einmal, dass beide Implementierungen des Frameworks eine Verbesserung (negative Abweichungen sind besser) gegenüber dem rein attributbasierten Verfahren aufzeigen. Die Implementierung LKW-SM von Chen et al. zeigt im Mittel bessere Werte als unsere Implementierung PPR-NG, mit Ausnahme der gemessenen Werte bei Tiefe 3, in der unser Verfahren am besten abschneidet. Des Weiteren fällt bei den gemessenen Werten auf, dass die Tiefe der Koautorenschaft nicht in jedem Fall eine Verbesserung mit sich bringt, sondern sich in Bezug auf die Werte vielmehr nicht konstant zeigt, so dass anhand dieser Daten keine direkten Rückschlüsse auf den Einfluß der Tiefe gefolgert werden können. Anhand der Koautorenschaftstiefe lässt sich aber erkennen, dass die Implementierung LKW-SM konstantere Werte aufweist, da der maximale Wert von -19,41 und der minimale Wert von -16,18 relativ nah beisammen liegen. Im Gegensatz dazu sind die Werte unserer Implementierung (PPR-NG) mit einem maximalen Wert von -23,82 und einem minimalen Wert von -6,76 stärkeren Schwankungen unterworfen. Hierbei können wir von einer signifikanten Abweichung sprechen. Unserer Meinung nach ist diese Beobachtung auf das in Phase 2 verwendete Verfahren zur Bestimmung der Konnektionsstärke zurückzuführen. Rein intuitiv sollte die Berechnung der *L-kürzesten Wege* weniger stark von der Koautorenschaftstiefe abhängen, da sich durch die Erweiterung des Graphens nicht zwangsläufig neue kürzeste Wege ergeben. Andererseits haben wir mit unserem eingesetzten Verfahren, dem *personalisierten PageRank*, einen stochastischen Prozess vorliegen, der viel stärker von Änderungen des Graphens unterworfen ist. In der Abbildung 4.2 ist der Zusammenhang von Performanz und Koautorenschaftstiefe nochmal graphisch dargestellt.

Im unteren Teil der Tabelle 4.2 werden zum Vergleich für das gleiche Experiment die aus dem Information Retrieval bekannten Metriken, *Precision* und *Recall*, verwendet.

		ATTR	LKW-SM	Δ_1 [%]	PPR-NG	Δ_2 [%]	Δ_3 [%]
Tiefe 1	Cluster Div.	0,648	0,349	-	0,377	-	8,02
	Entitäten Dis.	0,032	0,219	-	0,223	-	1,83
	Arithm. Mittel	0,340	0,284	-16,47	0,300	-11,76	5,63
Tiefe 2	Cluster Div.	0,648	0,356	-	0,346	-	-2,81
	Entitäten Dis.	0,032	0,213	-	0,204	-	-4,23
	Arithm. Mittel	0,340	0,285	-16,18	0,275	-19,12	-3,51
Tiefe 3	Cluster Div.	0,648	0,349	-	0,391	-	12,03
	Entitäten Dis.	0,032	0,213	-	0,127	-	-40,38
	Arithm. Mittel	0,340	0,281	-17,35	0,259	-23,82	-7,83
Tiefe 4	Cluster Div.	0,648	0,356	-	0,341	-	-4,21
	Entitäten Dis.	0,032	0,191	-	0,225	-	17,80
	Arithm. Mittel	0,340	0,274	-19,41	0,283	-16,76	3,28
Tiefe 5	Cluster Div.	0,648	0,356	-	0,383	-	7,58
	Entitäten Dis.	0,032	0,213	-	0,250	-	17,37
	Arithm. Mittel	0,340	0,285	-16,18	0,317	-6,76	11,23
Tiefe 1	Precision	0,578	0,797	-	0,689	-	-13,55
	Recall	0,955	0,753	-	0,763	-	1,33
	F-Measure	0,720	0,774	7,50	0,724	0,56	-6,46
Tiefe 2	Precision	0,578	0,785	-	0,715	-	-8,92
	Recall	0,955	0,758	-	0,773	-	1,98
	F-Measure	0,720	0,771	7,08	0,743	3,19	-3,63
Tiefe 3	Precision	0,578	0,798	-	0,713	-	-10,65
	Recall	0,955	0,758	-	0,864	-	13,98
	F-Measure	0,720	0,777	7,92	0,781	8,47	0,51
Tiefe 4	Precision	0,578	0,789	-	0,722	-	-8,49
	Recall	0,955	0,773	-	0,788	-	1,94
	F-Measure	0,720	0,781	8,47	0,754	4,72	-3,46
Tiefe 5	Precision	0,578	0,785	-	0,694	-	-11,59
	Recall	0,955	0,758	-	0,732	-	-3,43
	F-Measure	0,720	0,771	7,08	0,713	-0,97	-7,52

Tabelle 4.2.: Versuchsreihe „Fokussierter Autorenname“ für den Autorennamen „Smith“. Der obere Teil der Tabelle zeigt den Vergleich zwischen dem rein attributbasierten Verfahren (ATTR), der von Chen et al. vorgeschlagenen (LKW-SM) und unserer (PPR-NG) Implementierung des Konsolidierungsframeworks unter Verwendung der *Entropie*-basierten Metriken, wobei negative Werte auf eine Verbesserung hinweisen. Die prozentuale Verbesserung bzw. Verschlechterung in der Spalte Δ_1 sowie Δ_2 bezieht sich nur auf das arithmetische Mittel von *Cluster Diversität* und *Entitäten Dispersion*, die zusammen einen *Trade-Off* bilden. Der untere Teil der Tabelle zeigt den Vergleich unter Verwendung der Information Retrieval Metriken *Precision*, *Recall* und dem *F-Measure*, wobei diesmal positive Werte auf eine Verbesserung hinweisen. Die prozentuale Verbesserung bzw. Verschlechterung in der Spalte Δ_1 sowie Δ_2 bezieht sich nur auf das *F-Measure*, dem harmonischen Mittel zwischen der *Precision* und dem *Recall*.

	Verfahren	Komplexität
Phase 2	PPR	$O(nI)$
	LKW	$O(n^L)$
Phase 3	NG	$O(nm^2)$
	SM	$O(n^{1,5})$

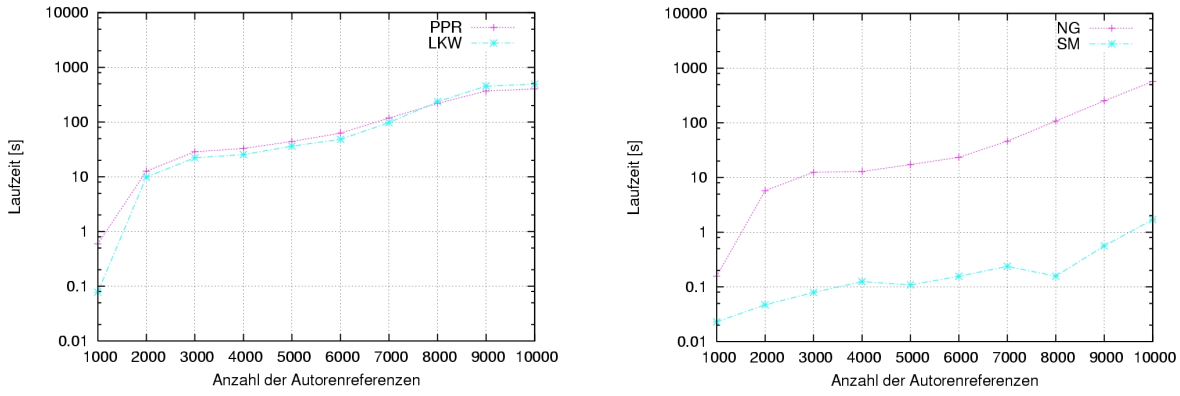
Tabelle 4.3.: Vergleich der Komplexitäten. Die angegeben Größen sind: n Anzahl der Knoten, m Anzahl der Kanten, L Weglänge und I Anzahl der Iterationen. Hierbei ist zu beachten, dass sich die Angabe der Komplexität für die Verfahren auf den betrachteten Teilgraphen beziehen und nicht auf die Gesamtanzahl der Referenzen, da der *erweiterte Referenzgraph* bzw. *virtueller Konsolidierungsteilgraph* nicht zwangsläufig zusammenhängend sein müssen.

Da größere Werte bei diesen Metriken mit besseren Ergebnissen korrespondieren, weisen positive Werte in der prozentualen Abweichung auf eine Verbesserung hin. Auch hier zeichnet sich wieder in Bezug auf die prozentuale Abweichung ein ähnliches Bild ab, wie wir dies zuvor bei den *Entropie*-basierten Metriken gesehen haben. Anhand der Werte in der Spalte LKW-SM lässt sich unsere zuvor gemachte Erkenntnis bestätigen, dass das Verfahren der *L-kürzesten Wege* bei Änderungen der Koautorenschaftstiefe keinen starken Schwankungen unterworfen ist. Auffallend an diesen Werten ist diesmal, dass unser Verfahren schlechter abschneidet als das von Chen et al. und sogar für die Koautorenschaftstiefe 5 noch hinter dem Wert des attributbasierten Verfahrens liegt. In der Abbildung 4.2 wird der Einfluss der Koautorenschaftstiefe illustriert.

Beide Verfahren zeigen also sowohl bei den *Entropie*-basierten Metriken als auch bei den aus den Information Retrieval stammenden Metriken eine Verbesserung gegenüber dem rein attributbasierten Verfahren. Auch zeigt sich, dass das von Chen et al. vorgeschlagene Verfahren weniger abhängig von der Koautorenschaftstiefe ist und relativ konstante Werte aufweist. Für dieselbe Versuchsreihe haben wir noch untersucht, wie sich eine Erhöhung des Fehlerprozentsatzes auf die Gesamtperformanz der Verfahren auswirkt. Aus der Abbildung 4.3 erkennen wir, dass bei einem Fehlerprozentsatz von bis zu 15% die Verfahren relativ stabil laufen, in dem Sinne, dass die Performanz relativ unberührt davon bleibt. Bei einem Fehlerprozentsatz von mehr als 15% fällt die Performanz des attribubasierten Verfahrens monoton ab, während die Gesamtperformanz nicht so stark abfällt. Es scheint also, dass sich sowohl unsere Implementierung des Frameworks als auch die von Chen et al. resistenter gegenüber Fehlern in den Attributen zeigt.

4.3.3. Laufzeit der Verfahren

Bei der Bewertung von Algorithmen ist nicht nur die Qualität des Ergebnisses ausschlaggebend, sondern auch dessen Effizienz, d.h. die zur Lösung benötigte Rechenzeit und der Ressourcenverbrauch. Aus diesem Grund wollen wir in diesem Abschnitt die Laufzeit der verschiedenen Implementierungen des Frameworks untersuchen, wobei wir



(a) (b)

Abbildung 4.4.: Laufzeit in Abhängigkeit der Anzahl von Autorenreferenzen. Die Grafik (a) zeigt die Laufzeit der Verfahren aus Phase 2, (b) zeigt die Laufzeit der Verfahren aus Phase 3.

die eingesetzten Verfahren der jeweilgen Phase separat voneinander betrachten wollen. Da die Effizienz von Algorithmen von der Größe der Probleminstanz – in unserem konkreten Fall der Anzahl der Autorenreferenzen – abhängt, benötigen wir dazu ausreichend große Datenmengen. Diese Anforderung verlangt eine andere Testumgebung als wir dies bei der Versuchsreihe „Fokussierter Autorenname“ verwendet haben. Indem wir die Konsolidierung auf einen einzelnen Namen fokussiert haben, konnten wir sicherstellen, dass ausreichend viele Koautorenschaftsbeziehungen in unseren Testdaten vorhanden sind. Zur Messung der Laufzeit ist diese Bedingung jedoch nicht zwangsläufig erforderlich, so dass wir in diesem Fall die Konsolidierung aller Autorennamen betrachten wollen. Ein weiteres Argument, welches gegen letztere Versuchsreihe zur Messung der Laufzeit spricht, ist, dass wir durch die Beschränkung auf einen einzelnen Namen nur wenige Referenzen für die Evaluation der in Phase 3 eingesetzten Verfahren erhalten. Im Unterschied zu Phase 2 findet nämlich die Berechnung der Verfahren aus Phase 3 auf den einzelnen *virtuellen Konsolidierungsteilgraphen* (alle in der ersten Phase gefundenen Smithe Clusters) statt. Aus diesem Grund wollen wir zur Messung der Effizienz die Laufzeiten aus dem Experiment „Vollständige Konsolidierung“ heranziehen.

Eine Übersicht für die Komplexitäten der einzelnen Verfahren ist der Tabelle 4.3 zu entnehmen. Dabei ist zu beachten, dass sich die Angabe der Komplexität der Verfahren aus Phase 2, LKW und PPR, auf den jeweils betrachteten *virtuellen Konsolidierungsteilgraphen* und nicht dem gesamten *Referenzgraphen* bezieht, wobei das Verfahren für jedes Paar von Referenzen, die über eine Ähnlichkeitskante verbunden sind, durchzuführen ist. Die Gesamtkomplexität dieser Verfahren muss also noch mit $O(m)$ multipliziert werden, wobei sich m auf die Anzahl der Ähnlichkeitskanten für den jeweils betrachteten *Konsolidierungsteilgraphen* bezieht. Die paarweise Analyse ist im Prinzip der „Flaschenhals“ des Gesamtverfahrens. In diesem Experiment wollen wir überprüfen, ob sich die erhaltenen Laufzeiten der Verfahren mit den in der Tabelle angegebenen Komplexitäten decken.

Die in der Abblidung 4.4 dargestellten Laufzeitwerte stammen aus der Versuchsreihe „Vollständige Konsolidierung“. Die Abbilung 4.4 (a) zeigt den Vergleich der Laufzeiten zwischen unserem Modell und dem Modell von Chen aus Phase 2. Analog dazu zeigt die Abbilung 4.4 (b) den Vergleich der Laufzeiten der Verfahren aus Phase 3. Wie aus der Grafik (a) hervorgeht, sind die beiden Modelle hinsichtlich ihrer Effizienz nahezu gleich, wobei wir für eine größere Anzahl an Autorenreferenzen erwarten, dass unser Modell (PPR) aus Phase 2 dem von Chen (LKW) überlegen ist. Dies gilt insbesondere dann, wenn für den Parameter L ein größerer Wert verwendet wird (exponentiell in L). Die Laufzeitwerte aus der Grafik (b) decken sich relativ gut mit den in der Tabelle 4.3 angegebenen Komplexitäten. Aufgrund der quadratischen Laufzeit bezüglich der Kantenanzahl m im Verfahren von Newman und Girvan schneidet dieser im Vergleich zum *spektralen* Clustering Verfahren von Shi und Malik deutlich schlechter ab.

5. Zusammenfassung und Ausblick

Der Einsatz von Data Mining Techniken zur Extraktion von nützlichen Informationen und Wissen aus großen Datenmengen setzt voraus, dass die Daten in einem bereinigten Zustand vorliegen, in der inkorrekte, unvollständige oder redundante Daten entdeckt und entsprechend behandelt wurden. Die Datenbereinigung stellt somit einen entscheidenden Faktor dar, um die Qualität der Daten und folglich die Ergebnisse des Minings zu verbessern. Diese Arbeit befasst sich mit der Objektkonsolidierung, die im engen Zusammenhang mit der Erkennung und Eliminierung von Duplikaten steht. Der Umgang mit Duplikaten ist ein grundlegendes Problem in der Datenbereinigung, das insbesondere in großen Realwelt-Datenbeständen vorzufinden ist, in der die Daten auf Grund der unterschiedlichen Repräsentation von Objekten und fehlender konsistenter Identifizierer eine stärkere Anfälligkeit für Dateninkonsistenzen aufweisen. Ziel der Objektkonsolidierung ist, alle Referenzen bzw. Duplikate einer Entität in einer Datenmenge korrekt zu gruppieren, d.h. es wird eine Partitionierung der Referenzen in Gruppen angestrebt, so dass einerseits alle Referenzen innerhalb einer Gruppe und andererseits keine zwei Referenzen aus zwei verschiedenen Gruppen dieselbe Entität repräsentieren.

In der vorliegenden Arbeit haben wir uns mit der Konsolidierung mehrdeutiger Autorennamen in digitalen Bibliotheken einem Problem angenommen, zu dem es in der Literatur bereits viele Lösungsansätze gegeben hat. Den meisten dieser Ansätze ist gemein, dass sie die Attributinformationen der Objekte verwenden, um diese anhand der Ähnlichkeit zu disambigieren. In Domänen, in denen nur sehr wenige Informationen über die zu deduplizierenden Objekte selbst zur Verfügung stehen, gestaltet sich jedoch dieses Problem als schwierig. Neuere Ansätze zur Konsolidierung verwenden aus diesem Grund neben den Attributinformationen der Objekte, einen weiteren semantischen Kontext, bei der die Objekte durch Relationen in Beziehung zueinander stehen. Zu diesen neuen Ansätzen zählt die in dieser Arbeit betrachtete graphbasierte Objektkonsolidierung, die neben der Link-basierten Klassifikation und der Link-basierten Clusteranalyse zu einem weiteren Anwendungsgebiet von Link Mining – ein erst kürzlich in Erscheinung getretenes Forschungsgebiet – gezählt werden kann. Mit Link Mining werden im Allgemeinen Lernverfahren aus dem Bereich des Data Minings bezeichnet, die versuchen, dass in den vernetzten Strukturen vohandene Wissen für eine Verbesserung dieser klassischen Verfahren zu verwenden. Der Fokus auf die relationalen Strukturen erfordert jedoch eine Reihe von Methoden und Konzepten, die sich von den traditionellen Methoden der Datenanalyse unterscheiden.

Das Ziel unserer Arbeit bestand darin, die Objektkonsolidierung im allgemeineren Kontext der *sozialen Netzwerkanalyse* zu untersuchen, in der die Analyse der Beziehungen zwischen Entitäten eine zentrale Rolle einnimmt. Zu diesem Zweck haben wir

auf einen bereits bestehenden Konsolidierungsalgorithmus zurückgegriffen, der 2005 von Chen et al. veröffentlicht wurde und als Framework konzipiert ist. Dieses Framework stellt also ein Gerüst zur Verfügung, in der die einzelnen Phasen durch spezifische Algorithmen zu implementieren sind. Während die in deren Arbeit vorgeschlagenen Modelle für die Implementation der einzelnen Phasen einzig allein auf das Kontext-Attraktions-Prinzip ausgerichtet waren, bestand die Hauptmotivation unserer Arbeit darin, das Framework in dem übergeordneten Kontext der *sozialen Netzwerkanalyse* zu untersuchen. Dabei haben wir gezeigt, wie sich die verschiedenen Phasen des Frameworks mit Methoden aus der *sozialen Netzwerkanalyse* beschreiben lassen und dazu jeweils alternative Modelle zu denen von Chen et al. vorgeschlagenen Modelle angegeben. Bei den beiden von uns vorgeschlagenen Modellen hat es sich zum einen um die Bestimmung der Akteurszentralität und zum anderen um die Identifzierung von Community Struktur gehandelt. Da diese beiden Aspekte einen wesentlichen Bestandteil unserer Arbeit ausmachen, haben wir diese relativ umfassend dargestellt und detailliert beschrieben. Bei der Einführung dieser der *sozialen Netzwerkanalyse* entspringenden Verfahren haben wir dabei jeweils einen Bezug zu praxisnahen Anwendungen genommen, in denen diese Verfahren eingesetzt werden.

Aufgrund der umfassenden Theorie, die sich hinter der *sozialen Netzwerkanalyse* verbirgt, haben wir gehofft, dass diese Verfahren in Bezug auf die Qualität der Konsolidierung besser performen als die von Chen et al. vorgeschlagenen Modelle. Allerdings konnten wir diese Vermutung in unseren Experimenten nicht bestätigen, da unsere Implementation des Frameworks zumindest in der ersten Versuchsreihe („Vollständige Konsolidierung“) schlechte Resultate hinsichtlich der gefundenen Clusterlösung geliefert hat. Durch eine gezielte Verstärkung der Beziehungen konnten wir jedoch mit der Versuchreihe („Fokussierter Autorenname“) ein positiveres Ergebnis für unser Verfahren abgewinnen. Diesbezüglich besteht für zukünftige Arbeiten die Klärung der Fragen, weshalb die Verfahren so unterschiedliche Resultate geliefert haben und auf welche Ursachen das schlechte Abschneiden unseres Verfahrens in der ersten Versuchsreihe zurückzuführen ist. Ein anderer interessanter Aspekt, der im Verlauf unserer Arbeit aufgekommen war, ist die Frage nach dem Zusammenhang zwischen den *Entropie*-basierten Metriken und den Information Retrieval Metriken, wobei insbesondere der Frage nachzugehen ist, welche der Metriken für die Qualitätsbewertung einer Clusterlösung geeigneter ist.

A. Anhang

A.1. Theoreme

Satz A.1 *(Perron-Frobenius Theorem) Sei $A = (a_{ij})$ eine reelle $n \times n$ Matrix mit positiven Einträge $a_{ij} > 0$, dann existiert ein reeller Eigenwert $\lambda_1 > 0$ mit korrespondierendem Eigenvektor $v_1 > 0$, so dass folgendes gilt:*

(i) $Av_1 = \lambda_1 v_1$,

(ii) falls $\lambda \neq \lambda_1$ ein beliebig anderer Eigenwert von A ist, dann gilt $|\lambda| < \lambda_1$,

(iii) λ_1 ist ein Eigenwert mit einer geometrischen und algebraischen Vielfachheit von 1.

(iv) Falls für eine reelle $n \times n$ Matrix $A \geqslant 0$ ein m existiert, so dass $A^m > 0$ gilt, dann treffen die Folgerungen $(i) - (iii)$ ebenso auf die Matrix A zu.

Satz A.2 *(Eigenschaften der Laplace Matrix)*

(i) L ist reel und symmetrisch [1]; die Eigenwerte $\lambda_1 \leqslant \lambda_2 \leqslant ... \leqslant \lambda_n$ der Matrix L sind reel und alle Eigenvektoren bilden eine reelle orthonormierte Basis.

(ii) Alle Eigenwerte λ_i sind nicht-negativ und L ist positiv semidefinit[2].

(iii) Der Vektor $\mathbf{1} = (1, 1, ..., 1)$ ist ein Eigenvektor mit Eigenwert 1, d.h. $L\mathbf{1} = 0\mathbf{1}$

(iv) Die Anzahl der Eigenvektoren mit Eigenwert 0 korrespondiert mit der Anzahl der Zusammenhangskomponenten von G. Insbesondere gilt:

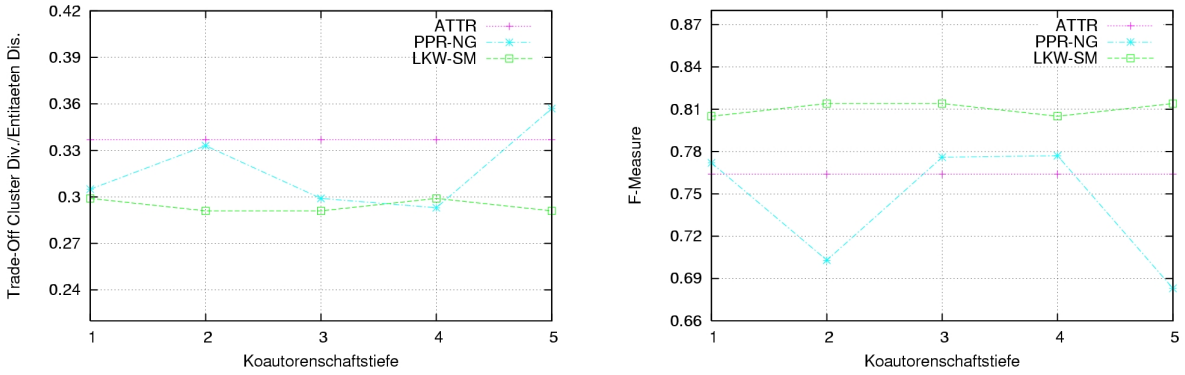
$$\lambda_2 \neq 0 \Leftrightarrow G \text{ ist zusammenhängend} \tag{A.1}$$

A.2. Experimente

[1] Eine Matrix A heißt symmetrisch, falls $A = A^T$ gilt.
[2] Eine symmetische Matrix heißt positiv semidefinit, falls alle Eigenwerte größer oder gleich Null sind.

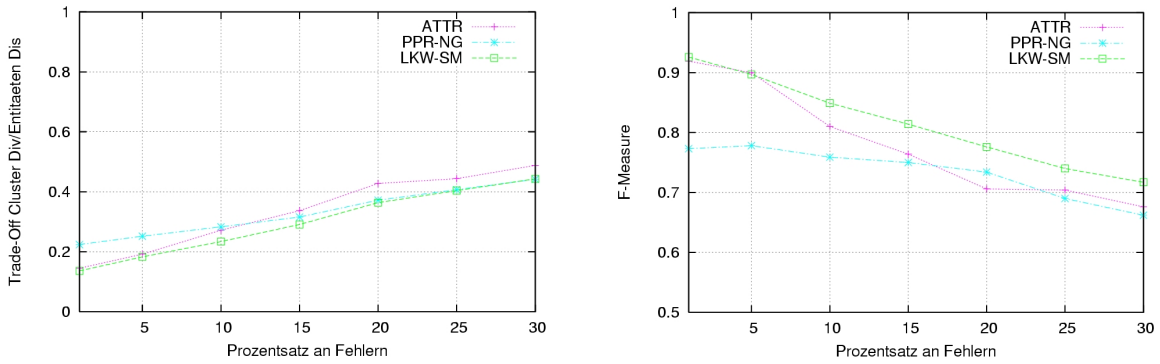
		ATTR	LKW-SM	Δ_1 [%]	PPR-NG	Δ_2 [%]	Δ_3 [%]
Tiefe 1	Cluster Div.	0,572	0,497	-	0,355	-	-28,57
	Entitäten Dis.	0,101	0,101	-	0,255	-	152,48
	Arithm. Mittel	0,337	0,299	-11,28	0,305	-9,50	2,01
Tiefe 2	Cluster Div.	0,648	0,473	-	0,331	-	-30,02
	Entitäten Dis.	0,032	0,109	-	0,336	-	208,26
	Arithm. Mittel	0,340	0,291	-14,41	0,333	-2,06	14,43
Tiefe 3	Cluster Div.	0,648	0,473	-	0,343	-	-27,48
	Entitäten Dis.	0,032	0,109	-	0,254	-	133,03
	Arithm. Mittel	0,340	0,291	-14,41	0,299	-12,06	2,75
Tiefe 4	Cluster Div.	0,648	0,497	-	0,355	-	-28,57
	Entitäten Dis.	0,032	0,101	-	0,232	-	129,70
	Arithm. Mittel	0,340	0,299	-12,06	0,293	-13,82	-2,01
Tiefe 5	Cluster Div.	0,648	0,473	-	0,354	-	-25,16
	Entitäten Dis.	0,032	0,109	-	0,360	-	230,28
	Arithm. Mittel	0,340	0,291	-14,41	0,357	5,00	22,68
Tiefe 1	Precision	0,658	0,722	-	0,775	-	7,34
	Recall	0,911	0,911	-	0,769	-	-15,59
	F-Measure	0,764	0,805	5,37	0,772	1,05	-4,10
Tiefe 2	Precision	0,658	0,739	-	0,759	-	2,71
	Recall	0,911	0,907	-	0,654	-	-27,89
	F-Measure	0,764	0,814	6,54	0,703	-7,98	-13,64
Tiefe 3	Precision	0,658	0,739	-	0,783	-	5,95
	Recall	0,911	0,907	-	0,769	-	-15,21
	F-Measure	0,764	0,814	6,54	0,776	1,57	-4,67
Tiefe 4	Precision	0,658	0,722	-	0,770	-	6,65
	Recall	0,911	0,911	-	0,784	-	-13,94
	F-Measure	0,764	0,805	5,37	0,777	1,70	-3,48
Tiefe 5	Precision	0,658	0,739	-	0,741	-	0,27
	Recall	0,955	0,907	-	0,633	-	-30,21
	F-Measure	0,720	0,814	13,06	0,683	-5,14	-16,09

Tabelle A.1.: Versuchsreihe „Fokussierter Autorenname“ für den Autorennamen „Chen“. Der obere Teil der Tabelle zeigt den Vergleich zwischen dem rein attributbasierten Verfahren (ATTR), der von Chen et al. vorgeschlagenen (LKW-SM) und unserer (PPR-NG) Implementierung des Konsolidierungsframeworks unter Verwendung der *Entropie*-basierten Metriken, wobei negative Werte auf eine Verbesserung hinweisen. Die prozentuale Verbesserung bzw. Verschlechterung in der Spalte Δ_1 sowie Δ_2 bezieht sich nur auf das arithmetische Mittel von *Cluster Diversität* und *Entitäten Dispersion*, die zusammen einen *Trade-Off* bilden. Der untere Teil der Tabelle zeigt den Vergleich unter Verwendung der Information Retrieval Metriken *Precision*, *Recall* und dem *F-Measure*, wobei diesmal positive Werte auf eine Verbesserung hinweisen. Die prozentuale Verbesserung bzw. Verschlechterung in der Spalte Δ_1 sowie Δ_2 bezieht sich nur auf das *F-Measure*, dem harmonischen Mittel zwischen der *Precision* und dem *Recall*.



(a) (b)

Abbildung A.1.: Einfluss der Koautorenschaftstiefe auf die Performanz. Die Kurve (a) zeigt den Verlauf der Cluster Div./Entitäten Dis., (b) zeigt den Verlauf des F-Measures.



(a) (b)

Abbildung A.2.: Einfluss des Fehlerprozentsatzes auf die Performanz. Die Kurve (a) zeigt den Verlauf der Cluster Div./Entitäten Dis., (b) zeigt den Verlauf des F-Measures.

Literaturverzeichnis

[Ber05] Pavel Berkhin. A survey on pagerank computing. *Internet Mathematics*, 2(2):73–120, 2005.

[BG06] Indrajit Bhattacharya and Lise Getoor. Entity resolution in graphs. In Lawrence B. Holder and Diane J. Cook, editors, *Mining Graph Data*. Wiley, 2006.

[Bra01] U. Brandes. A faster algorithm for betweenness centrality. In *Journal of Mathematical Sociology, 25(2):163–177, 2001.*, 2001.

[Cha02] Soumen Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan-Kauffman, 2002.

[CKM05] Zhaoqi Chen, Dmitri V. Kalashnikov, and Sharad Mehrotra. Exploiting relationships for object consolidation. In *Proc. of International ACM SIGMOD Workshop on Information Quality in Information Systems (ACM IQIS 2005)*, Baltimore, MD, USA, June 17 2005.

[CRF03] W. Cohen, P. Ravikumar, and S. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the IJCAI*, 2003.

[DDDA05] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas. Community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 9, May 2005.

[Fie75] M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czech. Math. J.*, 25:619–637, 1975.

[FPSSU96] Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining.* AAAI/MIT Press, 1996.

[Get03] Lise Getoor. Link mining: a new data mining challenge. *SIGKDD Explorations*, 5(1):84–89, 2003.

[GL89] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Baltimore, MD, USA, second edition, 1989.

[Hav02] Taher H. Haveliwala. Topic-sensitive pagerank. In *Proceedings of the Eleventh International World Wide Web Conference*, Honolulu, Hawaii, May 2002.

[JW02] G. Jeh and J. Widom. Scaling personalized web search. Technical report, Stanford University Technical Report, 2002., 2002.

[Kle99] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[KMC05] Dmitri V. Kalashnikov, Sharad Mehrotra, and Zhaoqi Chen. Exploiting relationships for domain-independent data cleaning. In *SIAM International Conference on Data Mining (SIAM SDM)*, Newport Beach, CA, USA, April 21–23 2005.

[LGB99] Steve Lawrence, C. Lee Giles, and Kurt Bollacker. Digital libraries and autonomous citation indexing. *IEEE Computer*, 32(6):67–71, 1999.

[LM03] A. Langville and C. Meyer. A survey of eigenvector methods of web information retrieval. *SIAM Review (Forthcoming)*, 2003.

[Mer94] R. Merris. Laplacians matrices of graphs: A survey. *Linear Algebra and its Apllications*, 197,198:143–176, 1994.

[MU05] Michael Mitzenmacher and Eli Upfal. *Probability and Computing : Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, January 2005.

[New04a] M. Newman. Coauthorship networks and patterns of scientific collaboration. In *Proceedings of the National Academy of Sciences, 101: 5200-5205.*, 2004.

[New04b] M. E. J. Newman. Analysis of weighted network. *Physical Review E*, 70(5):56131, 2004.

[New04c] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69:066133, 2004.

[NG03] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *cond-mat/0308217*, August 2003.

[OLKM05] Byung-Won On, Dongwon Lee, Jaewoo Kang, and Prasenjit Mitra. Comparative study of name disambiguation problem using a scalable blocking-based framework. In *JCDL '05: Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, pages 344–353, New York, NY, USA, 2005. ACM Press.

[PBMW98] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[Pot97] A. Pothen. Graph partitioning algorithms with applications to scientific computing. *Parallel Numerical Algorithms, D. E. Keyes, A. Sameh, and V. Venkatakrishnan, eds., Kluwer, 1997, pp. 323–368.*, 1997.

[PSL90] Alex Pothen, Horst D. Simon, and Kan-Pu Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Anal. Appl.*, 11(3):430–452, 1990.

[RCC+04] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. Defining and identifying communities in networks. *PROC.NATL.ACAD.SCI.USA*, 101:2658, 2004.

[SB02] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. *In The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002.*, 2002.

[Sei83] S.B. Seidman. Internal cohesion of ls sets in graphs. *Social Networks 5*, pages 97–107, 1983.

[SM00] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[WF94] Stanley Wasserman and Katherine Faust. *Social Network Analysis*. Cambridge University Press, Cambridge, 1994.

[WS49] W. Weaver and C.E. Shannon. *The Mathematical Theory of Communication*. University of Illinois Press, Urbana, Illinois, 1949. republished in paperback 1963.

[WS03] Scott White and Padhraic Smyth. Algorithms for estimating relative importance in networks. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 266–275, New York, NY, USA, 2003. ACM Press.