Bauhaus-Universität Weimar
Fakultät Medien

Evaluierung von Algorithmen der MajorClust-Familie

Bachelorarbeit

Denis Kreis

Betreuer:	Tim Gollub
1. Gutachter:	Prof. Dr. Benno Maria Stein
Abgabedatum:	3. April 2009

Erklärung

Hiermit versichere ich, daß ich die vorliegende Bachelorarbeit selbständig und nur unter Verwendung der angegebenen Quellen angefertigt habe. Die Arbeit wurde in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Weimar, den 3. April 2009 Denis Kreis

Inhaltsverzeichnis

Vorwort

Auf dem Weg in die Wissensgesellschaft werden immer mehr Menschen von riesigen Mengen an Information überrollt. Exponentielles Wachstum der Informationsmenge bringt aber auch die Problematik der Zugänglichkeit des für eine einzelne Person in einer einzelnen Situation relevanten Wissens mit sich. Allein die Verfügbarkeit von relevanten Informationen reicht nicht aus, um die Informiertheit des Nachfragenden zu garantieren.

Um die Informationsflut zu überwinden werden immer raffiniertere Komplexitätsreduktionsmechanismen gebraucht, die dem Bedarf des Informationssuchenden entsprechend selektiertes Wissen liefern können. Einer dieser Mechanismen ist Dokumenten-Clustering.

Das Thema der vorliegenden Arbeit ist die Evaluierung der Clusteranalyse-Algorithmen, die im Dokumenten-Clustering ihre Verwendung finden. Dabei wird die Evaluierung auf die Algorithmen der MajorClust-Familie eingeschränkt.

1. Einleitung

Dokumenten-Clustering ist ein auf der Clusteranalyse basierendes Verfahren zum themenbezogenen Sortieren von Dokumenten. Dabei erfolgt die Gruppierung von semantisch ähnlichen Dokumenten mit Hilfe von Clustering-Algorithmen.

Je nach der Funktionsweise sind unterschiedliche Clustering-Algorithmen verschieden geeignet für das Dokumenten-Clustering. Die Auswahl eines geeigneten Verfahren beeinflußt stark die Qualität des Ergebnisses.

MajorClust ist ein von Benno M. Stein und Oliver Niggemann entwickelter Clustering-Algorithmus, der sich in Bereich des Dokumenten-Clusterings bewährt hat. In dieser Arbeit wird MajorClust und seine später entwickelten Modifikationen auf die Eignung für das Dokumenten-Clustering getestet und anhand von unterschiedlichen Qualitätsmaßen miteinander verglichen.

Die Arbeit ist folgendermaßen aufgebaut:

In Kapitel 2 werden Grundprinzipien und Arten der Clusteranalyse, die die Grundlage für das Dokumenten-Clustering bildet, vorgestellt.

Kapitel 3 beschreibt den allgemeinen Prozess des Dokumenten-Clusterings. Ferner werden verschiedene Anwendungsszenarien illustriert, die aus einer Integration des Dokumenten-Clusterings in andere Prozesse, wie z.B. Information Retrieval oder Data Mining, hervorgehen. Im Abschnitt 3.3 werden verschiedene Stufen der Datenaufbereitung und das weitverbreitete BagOfWords-Dokumentenmodell beschrieben.

In Kapitel 4 wird auf unterschiedliche Arten von Clustering-Algorithmen eingegangen und MajorClust und seine Nachfolger beschrieben.

Schließlich werden verschiedene Qualitätsmaße in Kapitel 5 beschrieben, Ergebnisse der in dieser Arbeit durchgeführten Experimente vorgestellt und diskutiert.

2. Clusteranalyse

Unter Clusteranalyse versteht man multivariate, strukturentdeckende Analyseverfahren, die in einer Menge von Objekten Objekte, die ähnliche Eigenschaften aufweisen, zu Gruppen(Clustern) zusammenfassen. Die Anzahl der Cluster kann dabei im Voraus festgelegt oder anhand von bestimmten Qualitätsgüten vom Clustering-Algorithmus selbst determiniert werden.

2.1. Zielsetzung und Arten der Clusteranalyse

Als Beispiel sei eine Obstschale gegeben, in der Äpfel, Birnen und Orangen enthalten sind. Das Ziel ist die Obstsorten so mit Labels zu versehen, dass jedes Label eine bestimmte Obstsorte markiert. Dabei kommt es nicht drauf an, wie die Gruppen markiert werden, da es kein Wissen über die Klassen der Objekte vorausgesetzt wird. Wird solches Wissen vorausgesetzt, handelt es sich um ein *Klassifikationsproblem*.

Das Ziel der Clusteranalyse ist das Aufspüren und Gruppieren von ähnlichen Objekten in einer gegebenen Objektmenge. Dabei wird zwischen

- objektorientierter und
- variablenorientierter Datenanalyse

unterschieden. Werden Objekte anhand von ihren Eigenschaften gruppiert, spricht man von objektorientierter Datenanalyse. Werden Eigenschaften von Objekten beispielsweise aufgrund ihrer Korrelation zu Gruppen zusammengefasst, spricht man von variablenorientierter Datenanalyse [Bac94]. Im Folgenden wird in dieser Arbeit mit Clusteranalyse objektorientierte Clusteranalyse gemeint.

Ferner unterscheidet man zwischen

- harten und
- weichen Clusteranalysemethoden.

Harte Methoden ordnen jedes Objekt deterministisch einem Cluster zu. D.h., das Objekt wird genau einem Cluster zugewiesen. Die Frage, ob ein Objekt zu einem bestimmten Cluster gehört oder nicht, kann dabei mit einer Wahrscheinlichkeit von 1 bzw. 0 beantwortet werden. Weiche Methoden ordnen dagegen jedes Objekt jedem Cluster mit einer gewissen Wahrscheinlichkeit zu. Entsprechend wird die Frage nach der Zugehörigkeit eines Objekts zu einem bestimmten Cluster mit einer Wahrscheinlichkeit zwischen 0 und 1 beantwortet. Daher sind weiche Clusteranalysemethoden nicht deterministisch. Eine besondere Stellung zwischen harten und weichen Clusteranalysemethoden nehmen *überlappende Clusteranalysemethoden* ein [Bac94]. Diese können ein Objekt deterministisch mehreren Clustern zuweisen.

2.2. Grundprinzipien der Clusteranalyse

Homogenität und Heterogenität sind zwei grundlegende Prinzipien, die bei der Clusteranalyse verfolgt werden. Unter Homogenität wird dabei die Ähnlichkeit zwischen Objekten innerhalb eines Clusters verstanden. Das bedeutet, dass die Objekte, die zu einem bestimmten Cluster gehören, untereinander ähnlich sein müssen. Das Heterogenitätsprinzip besagt, dass Objekte aus verschiedenen Clustern unterschiedlich sein sollen.

Beide Werte sollen maximiert werden.

2.3. Proximitätsmaß

Um ähnliche Objekte zu finden wird ein quantitatives Maß gebraucht, das die gegebenen Objekte in ein Verhältnis zueinander stellt. Dafür sollen zuerst die Eigenschaften der Objekte, die für deren Vergleich miteinander relevant sind, abstrahiert werden. Diese Eigenschaften werden dann in einer $n \times m$-Matrix, Rohdatenmatrix genannt, zusammengefasst (s. Tabelle 2.1). Die Zeilendimension n der Matrix entspricht der Anzahl der Objekte in der Objektmenge, die Spaltendimension m der Anzahl der relevanten Eigenschaften.

	Eigenschaft 1	Eigenschaft 2	...	Eigenschaft m
Objekt 1	...	...	...	...
Objekt 2	...	...	...	...
⋮	⋮	⋮	⋱	⋮
Objekt n	...	...	...	...

Tabelle 2.1.: Rohdatenmatrix

Die Rohdatematrix wird auch als Vektorrepräsentation der Objektmenge bezeichnet und

kann als allgemeingültige Ausgangsstruktur für die eigentliche Clusteranalyse betrachtet werden.

Im nächsten Schritt werden aus den in der Rohdatenmatrix enthaltenen Daten in Abhängigkeit von dem im Folgenden zu verwendenden Fusionierungsalgorithmus, d.h. dem Algorithmus, der die Clusteranalyse vornimmt, die Proximitätsmaße paarweise zwischen den gegebenen Objekten oder zwischen den gegebenen und virtuellen Objekten, die beispielsweise Schwerpunkte oder Mittelwerte von Clustern repräsentieren, berechnet. Für die gewonnenen Werte soll eine geeignete Datenstruktur gewählt werden, die dann als Input für den Fusionierungsalgorithmus dient. Solche Datenstrukturen werden im Unterabschnitt 4.1.2 näher beschrieben. Als Beispiel ist hier eine Matrix gegeben, die Proximitätswerte zwischen n Objekten enthält (s. Tabelle 2.2).

	Objekt 1	Objekt 2	...	Objekt n
Objekt 1	...	...	...	...
Objekt 2	...	...	...	...
⋮	⋮	⋮	⋱	⋮
Objekt n	...	...	...	...

Tabelle 2.2.: Ähnlichkeits- bzw. Unähnlichkeitsmatrix

2.3.1. Ähnlichkeitsmaße

Sei eine Proximitätsfunktion ρ gegeben, die zwei gegebene Objekte bzw. Zeilen aus einer Rohdatenmatrix auf eine reelle Zahl abbildet.

$$\rho(x, x') = a,\ a \in \mathbb{R}$$

Proximitätsmaße, die eine Ähnlichkeit zwischen zwei Objekten repräsentieren, d.h. je ähnlicher die Objekte sind, desto größer ist der Wert, werden Ähnlichkeitsmaße genannt. Für sie gelten folgende Axiome [Ste77]:

1. $\rho(x, x') = \rho(x', x)$
2. $\rho(x, x') \leq \rho(x, x)$

2.3.2. Distanzmaße

Außerdem verwendet man so genannten Unähnlichkeits- bzw. Distanzmaße. Diese zeigen, inwiefern sich zwei Objekte voneinander unterscheiden. Dabei wird deren Wert kleiner,

wenn Objekte ähnlicher sind, und größer, wenn Objekte verschieden sind. Für solche Proximitätsmaße gelten folgende Axiome:

1. $\rho(x, x') = \rho(x', x)$
2. $\rho(x, x') \geq \rho(x, x)$

2.4. Formalisierung

Sei X eine Objektmenge, Y eine Menge mit Labels, die Cluster repräsentieren. Ferner sei eine Proximitätsfunktion

$$\rho : X \times X \to \mathbb{R}$$

bzw.

$$\rho(x, x') = a,\ a \in \mathbb{R}$$

mit $x, x' \in X$ gegeben. Es wird gefordert, dass die Menge X in Untermengen $X_i \subseteq X$ mit $i = 1...k$, die als Cluster bezeichnet werden, geteilt wird. Dabei wird jedem $x \in X$ ein Label $y \in Y$ zugeordnet.

Zu Beachten ist, dass im Falle der harten Clusteranalyse

$$X_i \cap X_j = \emptyset$$

$\forall i, j$ mit $i \neq j$, $1 \leq i, j \leq k$ sein soll.

Ein Clustering-Algorithmus ist dann eine Funktion α:

$$\alpha : X \to Y,$$

die jedem Objekt $x \in X$ ein Label $y \in Y$ zuordnet.

3. Dokumenten-Clustering

Dokumenten-Clustering (auch als Text-Clustering bekannt) ist ein auf der Clusteranalyse basierendes Verfahren zur Gruppenbildung von Dokumenten, das beispielsweise in Information Retrieval und Data-Mining seine Anwendung findet.

3.1. Aufgabe des Dokumenten-Clustering

Die Aufgabe des Dokumenten-Clustering besteht darin, in einer gegebenen Dokumentenmenge, die als Dokumentenkollektion bezeichnet wird, Gruppen von semantisch ähnlichen Dokumenten zu entdecken. Es ist naheliegend, dass Dokumente, die ähnliche Vokabulare besitzen, auch semantisch ähnlich sind, also ähnliche Themen behandeln.

Außerdem ist es vorstellbar, Dokumente beispielsweise nach dem Schreibstil zu untersuchen, um diese dann nach den Erzählungsarten, Autoren oder Genres zu clustern.

Im nächsten Abschnitt werden die häufigsten Anwendungsszenarios erläutert.

3.2. Dokumenten-Clustering-Anwendungsszenarios

3.2.1. Data-Mining-Szenario

Eine der denkbaren Einsatzmöglichkeiten für Dokumenten-Clustering wird hier im Data-Mining-Szenario vorgestellt.

Unter Data-Mining versteht man das Entdecken von systematischen Gesetzmäßigkeiten und Zusammenhängen in großen Datenmengen unter dem Einsatz von unterschiedlichen Klassifikationsmethoden. Im Falle des Dokumenten-Clustering wäre die Datenmenge eine oder mehrere Dokumenten- bzw. Textkollektionen, und Gesetzmäßigkeiten, die darin zu entdecken sind, wären beispielsweise semantische Zugehörigkeiten von verschiedenen Dokumenten aus diesen Kollektionen zu thematischen Klassen.

Der schematische Ablauf dieses Szenarios wird in Abbildung 3.1 vorgestellt. Die Doku-

mentenkollektion ist eine unstrukturierte Ansammlung von Dokumenten. Das Dokumenten-Clustering-Modul beinhaltet einen Abstraktionsmechanismus, der Dokumente in ein geeignetes Repräsentationsmodell überführt, und einen Clustering-Algorithmus, der die Einteilung der heterogenen Dokumentenmenge in n kleinere homogene Gruppen(Clustern) vornimmt.

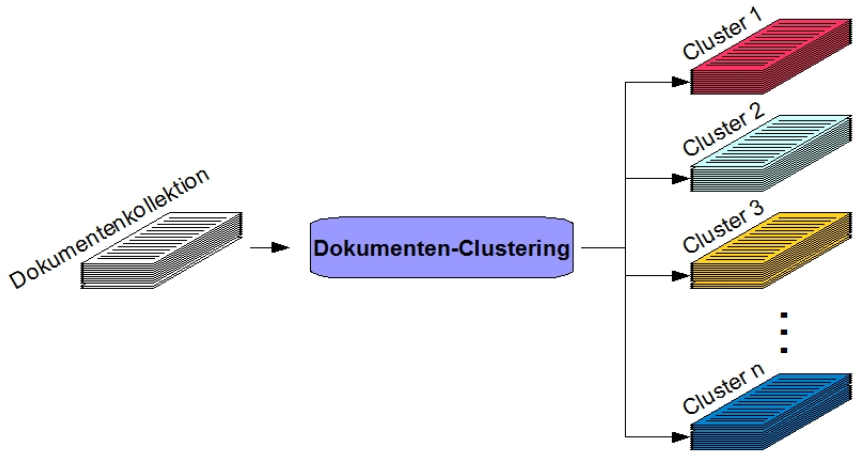

Abbildung 3.1.: Data-Mining-Szenario

3.2.2. Information-Retrieval-Szenario

Information Retrieval(IR) beschäftigt sich mit dem Suchen und Finden von Informationen in großen Datenbeständen [Sto06].

Es wird davon ausgegangen, dass ein Subjekt einen handlungsrelevanten Informationsbedarf hat, der als eine vage Anfrage formuliert an das System gerichtet wird. Das System hat dabei Zugriffsmöglichkeiten auf eine oder mehrere Datenquellen, die zur Informationsgewinnung benutzt werden können. Die vom Subjekt gestellte Anfrage soll vom Sys-

tem so interpretiert werden, dass der Informationsbedarf möglichst unverfälscht wieder abstrahiert werden kann. Dabei wird die Anfrage und die in den Datenbeständen enthaltenen Entitäten in ein einheitliches Modell übertragen (siehe dazu den Abschnitt 3.3). Das System führt den Vergleich der Anfrage mit jeder Entität durch und berechnet die Relevanz der Entität zur Anfrage. Als Antwort bekommt das Subjekt beispielsweise eine Auflistung aller relevanten Dokumente. Üblicherweise werden die Entitäten nach der Relevanz in absteigender Reihenfolge sortiert, und solch eine Auflistung wird als *Relevance Ranking* bezeichnet.

Als Beispiel für ein Information-Retrieval-System kann eine Websuchmaschine dienen. Sie bekommt eine vom Nutzer formulierte Anfrage in Form von Suchbegriffen und liefert dem Nutzer eine nach Relevanz sortierte Liste von Webdokumenten, die die Suchbegriffe enthalten.

Pre-Retrieval Clustering

In der Praxis werden von IR-Systemen Entitäten aus verschiedenen Themenbereichen als relevant zu einer bestimmten Anfrage eingestuft. Die Tatsache, dass die Dokumente aus verschiedenen Themenbereichen stammen, macht es offensichtlich, dass diese Entitäten verschiedene Informationsbedürfnisse befriedigen. Um den Bedürfnissen des Informationssuchenden näher zu kommen, wurde es vorgeschlagen, Dokumenten-Clustering in den Information-Retrieval-Prozeß zu integrieren.

Die vom C.J. van Rijsbergen vorgeschlagene *Cluster-Hypothese* [Rij79] besagt, dass eng zusammenhängende Dokumente dazu tendieren, zu derselben Anfrage relevant zu sein. Anders formuliert neigen thematisch ähnliche Dokumente dazu, denselben Informationsbedarf zu decken.

Somit kann die Suche auf Clustering der Dokumentenmenge und eine darauffolgende Auswahl eines Clusters, der den Informationsbedarf am besten deckt, begrenzt werden. Der Inhalt dieses Clusters wird dem Nutzer als Ergebismenge geliefert.

Zusätzlich kann die Ergebnismenge nach Relevanz sortiert oder mit einer Stichwortsuche verfeinert werden.

Dieses Szenario ist in der Abbildung 3.2 dargestellt. Das Dokumenten-Clustering-Modul beinhaltet einen Abstraktionsmechanismus zur Dokumentenrepräsentation und einen Clustering-Algorithmus. Jeder der resultierenden Cluster kann beispielsweise durch ein virtuelles Dokument, das den Mittelwert aller im Cluster enthaltenen Dokumente darstellt, repräsentiert werden. Die so gewonnenen Dokumente werden für die Relevanzbestimmung benutzt, um einen relevanten Cluster zu finden.

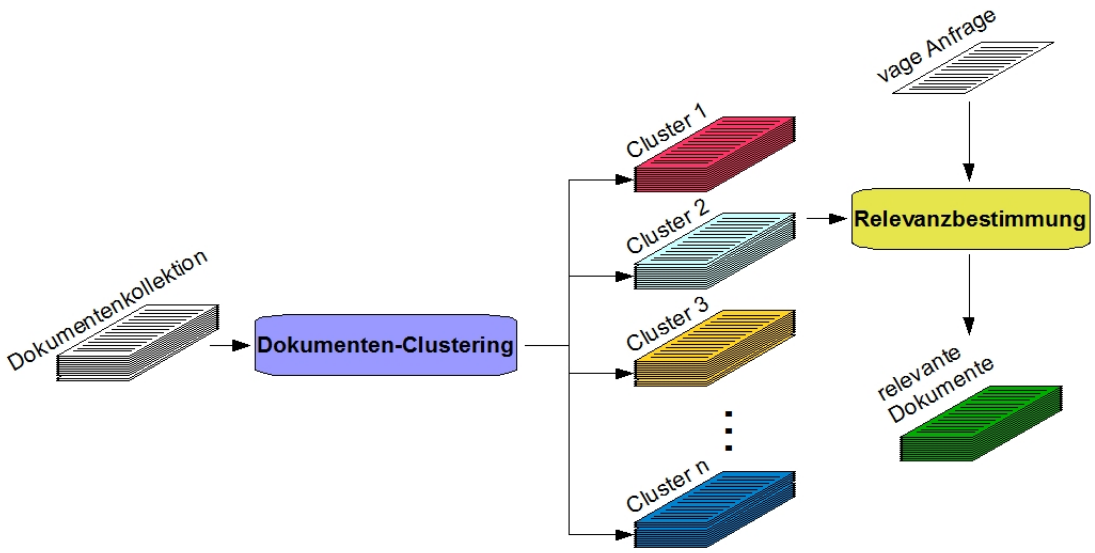

Abbildung 3.2.: Pre-Retrieval Clustering

Post-Retrieval Clustering

Die a priori Clusteranalyse der ganzen Dokumentenkollektion ist sehr aufwendig. Darüber hinaus kann sie nur serverseitig stattfinden. Das führt zu einer schlechten Performance. Außerdem haben Marti A. Hearst und Jan O. Pedersen gezeigt, dass die Annahme, dass zwei Dokumente, die für eine Anfrage relevant oder irrelevant sind, auch für eine andere Anfrage das gleiche Ergebnis liefern, falsch ist [HP96]. Darüber hinaus haben sie vorgeschlagen, Dokumenten-Clustering nach der Ähnlichkeitssuche in den Retrievalprozeß zu integrieren.

Dieses Vorgehen ist in der Abbildung 3.3 dargestellt. Im ersten Schritt wird auf die Dokumentenkollektion konventionelle Änlichkeitssuche angewendet. Im zweiten Schritt wird die Menge der relevanten Dokumente in kleineren Gruppen geteilt. Der Nutzer kann dann eine oder mehrere zu seinem Informationsbedürfnis passende Untermengen auswählen. Somit wird die Ergebnismenge verkleinert und verfeinert.

Da Clusteranalyse von großen Datenmengen ein rechenaufwendiger Prozeß ist, gewinnt

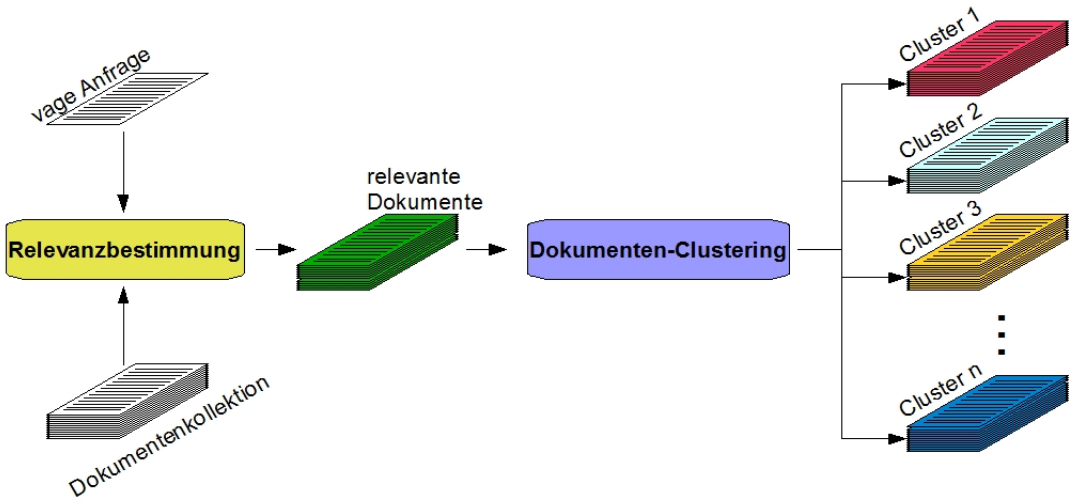

Abbildung 3.3.: Post-Retrieval Clustering

Post-Retrieval-Clustering gegenüber dem Pre-Retrieval-Clustering dadurch an Performance, dass die Menge der zu clusternden Dokumente nur die als relevant eingestuften Entitäten enthält und damit nur ein Bruchteil der ganzen Dokumentenkollektion ist.

Außerdem kann der Clustering-Prozeß auf die Client-Seite verlegt werden.

3.2.3. Scatter/Gather-Szenario

In den Fällen, in denen der Informationsbedarf des Informationssuchenden nicht scharf genug formuliert werden kann, oder der Informationssuchende sich einen Überblick über den Inhalt einer Dokumentenkollektion verschaffen möchte, bietet sich eine in [CKPT92] vorgeschlagene Scatter/Gather-Methode.

Das zugrunde liegende Verfahren für die Scatter/Gather-Idee ist Dokumenten-Clustering. Die Dokumentenkollektion wird dabei in eine bestimmte Anzahl von Gruppen mit ähnlichen Dokumenten geclustert. Ein Hilfsmechanismus extrahiert dann aus jedem Cluster

ein Schlüsselwort, das den Inhalt des Clusters repräsentiert. Das Schlüsselwort soll dabei semantisch möglichst genau die im Cluster enthaltenen Dokumente beschreiben. Im nächsten Schritt wählt der Nutzer einen oder mehreren Cluster aus, die seinem Informationsbedarf am nächsten liegen. Die gewählten Cluster werden zu einer neuen Dokumentenmenge vereinigt, die wiederum geclustert wird. Der gesamte Vorgang wird iterativ wiederholt, und nach jeder Iteration wird die Dokumentenmenge kleiner und nähert sich dem Informationsbedarf des Nutzers an.

3.3. Repräsentation von Dokumenten

Sei eine Dokumentenmenge D gegeben. Um semantisch ähnliche Dokumente $d \in D$ gruppieren zu können, d.h. eine Clusteranalyse auf der gegebenen Menge D durchführen zu können, wird ein quantitatives Proximitätsmaß zum Vergleich von Dokumentenpaaren benötigt. Damit eine automatisierte Berechnung des Proximitätsmaßes ermöglicht wird, müssen die in D enthaltenen Dokumente in ein Modell, das die Semantik der Dokumente repräsentieren kann, überführt werden.

Die für Dokumenten-Clustering geeigneten Modelle lassen sich aus dem Information-Retrieval-Bereich übernehmen. In der Abbildung 3.4 wird eine Taxonomie von Retrieval-Modellen dargestellt.

In diesen Modellen soll dabei die Anfrage durch ein zweites Dokument ersetzt werden (s. Abbildung 3.5).

Somit lässt sich die Definition für das Dokumenten-Clustering-Modell folgendermaßen formulieren (vgl. [Stea]):

Sei D eine Menge von Dokumenten. Ein Dokumenten-Clustering-Modell für D ist ein Tupel $(\boldsymbol{D}, \boldsymbol{D}, \rho_R)$, dessen Elemente wie folgt formuliert sind:

1. *$\boldsymbol{D}$ ist eine Menge der Repräsentationen der Dokumente $d \in D$. In $\boldsymbol{d} \in \boldsymbol{D}$ können Layout, logische und semantische Informationen codiert sein. Die Menge $\boldsymbol{D}$ wird wie folgt definiert: $\alpha : D \rightarrow \boldsymbol{D}$*
2. *R ist ein Retrieval-Modell, das ein Prinzip, ein Paradigma oder eine linguistische Theorie formalisiert.*

Auf der Grundlage von R ist die Proximitätsfunktion $\rho_R(\mathbf{d}, \mathbf{d'})$ definiert, die Ähnlichkeit bzw. Unähnlichkeit zwischen zwei Dokumentrepräsentationen $\mathbf{d}, \mathbf{d'} \in \mathbf{D}$ quantifiziert:

$$\rho_R : \mathbf{D} \times \mathbf{D} \rightarrow \mathbb{R}$$

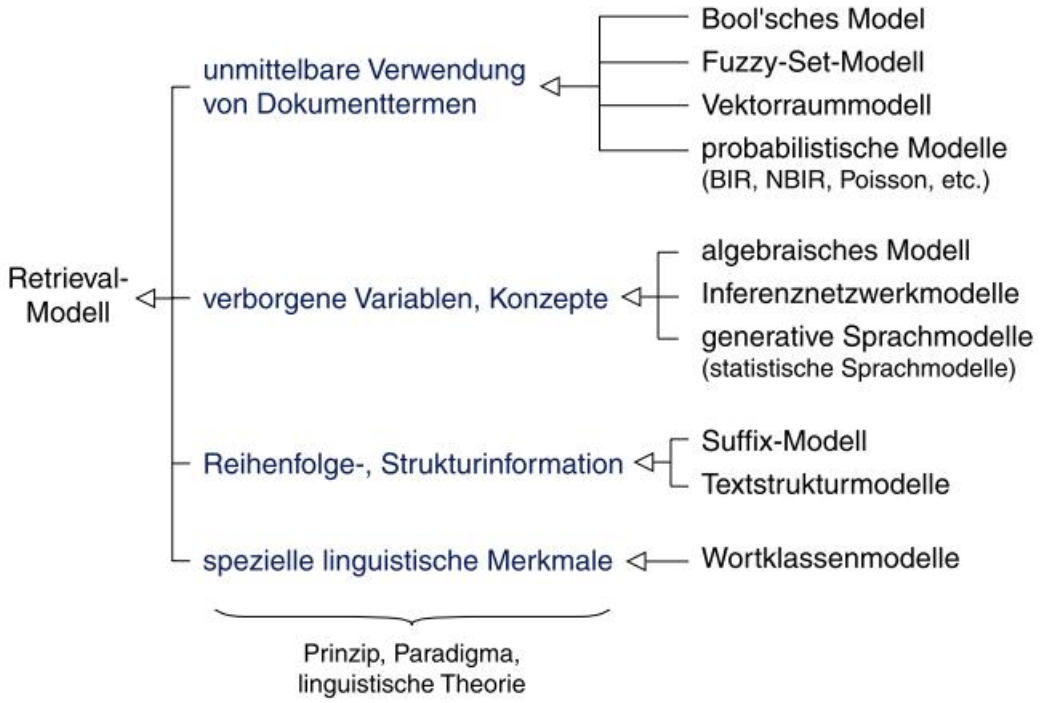

Abbildung 3.4.: Taxonomie von Retrieval-Modellen (Quelle: [Stea])

Die Funktion α ist vom zu verwendenden Dokumentenmodell abhängig. Sie konvertiert ein Dokument in eine Dokumentrepräsentation. Dabei werden zwei Phasen durchlaufen. In der ersten Phase wird eine Vorverarbeitung durchgeführt. Das Dokument wird auf eine vereinfachte Form reduziert, die nur für den Dokumentenvergleich relevante Informationen enthält. Diese Phase wird *Indexierung* genannt. Sie wird näher im nächsten Abschnitt beschrieben. In der zweiten Phase werden Dokumente in ein einheitliches Bezugssystem gestellt. In den Unterabschnitten 3.3.2 und 3.3.3 werden Vektorraum- und Suffix-Tree-Modelle näher beschrieben.

3.3.1. Indexierung

Als Indexierung wird ein mehrstufiger Prozess bezeichnet, der eine Reduktion der Dokumente vornimmt. Im Laufe dieses Prozesses werden aus einem Dokument, das in Form einer Zeichenkette vorliegt, für das zu verwendete Dokumentenmodell relevante Informationen extrahiert. Die nicht relevanten Informationen werden nicht berücksichtigt. In vielen Fällen wird dieser Prozeß auf die Auswahl repräsentativer Terme, die als Indexterme oder Deskriptoren bezeichnet werden, begrenzt. Daher wird das Ergebnis der Indexierung als Index bezeichnet.

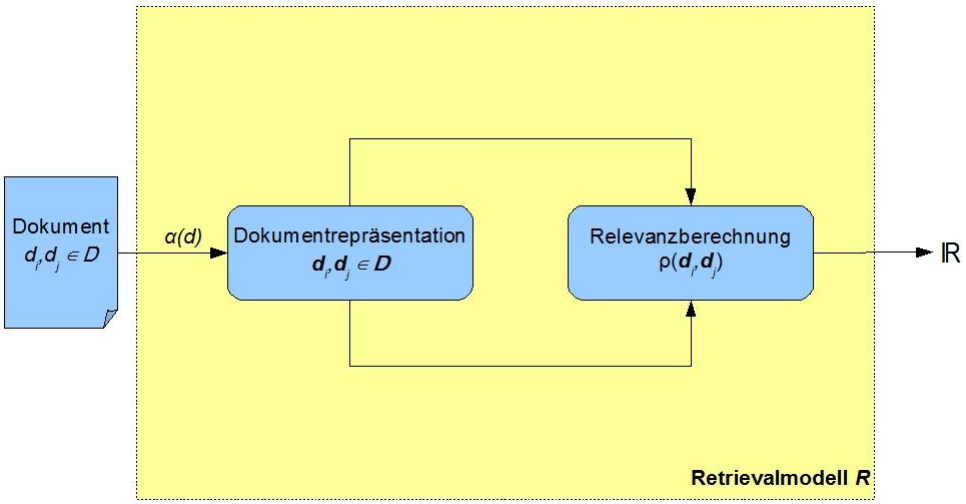

Abbildung 3.5.: Konzeptueller Aufbau eines Dokumenten-Clustering-Modells (vgl. [Gol08])

Die einzelnen Verarbeitungsstufen der Indexierung werden in der Abbildung 3.6 schematisch dargestellt. Es ist aber zu berücksichtigen, dass üblicherweise nicht alle der in der Abbildung vorgestellten Techniken in einzelnen Anwendungen ihre Verwendung finden. Außerdem ist der Gebrauch von einzelnen Techniken von dem Dokumentenmodell abhängig.

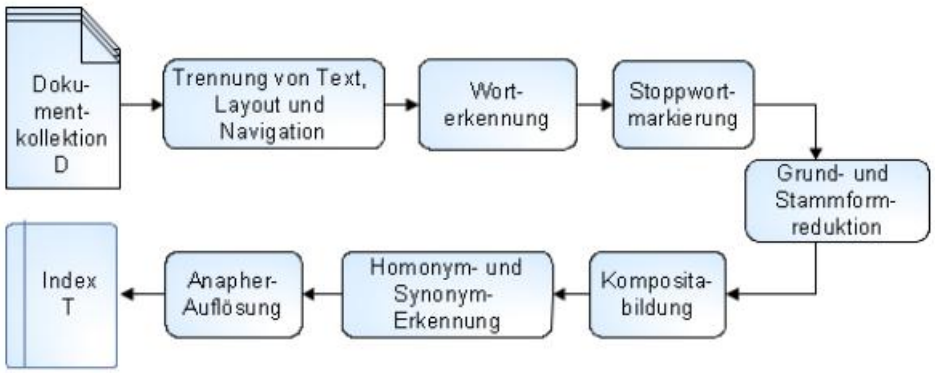

Abbildung 3.6.: Mehrstufiger Indexierungsprozeß (Quelle: [Gol08])

Im Folgenden wird auf einzelne Techniken der Indexierung näher eingegangen.

Trennung von Text und Layout

In diesem Schritt werden Strukturinformationen wie Layout und Formatierung entfernt. Solche Informationen können beispielsweise in Form von Markup Language Tags (HTML, XML) vorliegen. Obwohl Strukturinformationen bei den gängigsten Retrieval-Modellen nicht berücksichtigt werden, gibt es einige Modelle, die diese Informationen gebrauchen. In solchen Modellen wird angenommen, dass solche Dokumentbereiche wie Titel oder Abstract Terme enthalten, die genaueren Aussagen über den Inhalt des Dokumentes beinhalten.

Worterkennung

Die Zeichenketten werden in einzelne Tokens zerlegt. Als Tokenseparatoren dienen dabei Leerräume und Interpunktionszeichen.

Stoppwortmarkierung

Als Stoppworte werden die Worte bezeichnet, die rein grammatische Bedeutung tragen und keinen Einfluss auf den Inhalt des Dokumentes haben. Solche Worte kommen mit gleicher Wahrscheinlichkeit in allen Dokumente vor. Sie haben damit keine Repräsentanz und folglich keinen Unterscheidungspotential. Da Stoppworte sprachspezifisch sind, wird für jede vom System unterstützte Sprache eine Stoppwortliste erstellt. Obwohl Stoppworte nicht diskriminierend sind, werden sie oft nicht vollständig eliminiert, sondern nur markiert. So können sie beispielsweise für Phrasensuche benutzt werden, während sie in anderen Suchvorgängen nicht berücksichtigt werden [Sto06].

Grund- und Stammformreduktion

Grundformreduktion ist eine linguistische Technik. Dabei werden einzelne Terme auf ihre Grundform, das Lexem, zurückgeführt. Die Grundform für Substantive wäre beispielsweise Nominativ Singular, für Verben Infinitiv.

Stammformreduktionen ist eine nicht-linguistische Technik. Einzelne Terme werden dabei auf ihre Stammform zurückgeführt. Stammformreduktion beschränkt sich beispielsweise für das Deutsche und Englische auf das Entfernen von Suffixen. Präfixe werden aber nicht entfernt, da sie Wortbedeutungen völlig ändern können.

Komplexität und Genauigkeit der Grund- bzw. Stammformreduktion sind sprachabhängig. Bei flektierenden Sprachen steigt die Komplexität und sinkt die Genauigkeit.

Kompositabildung

Einige Terme können ihre Bedeutung verlieren, wenn sie unabhängig von anderen Termen betrachtet werden. Die Bedeutung wird dabei erst aus der Komposition von zwei oder mehreren Worten ersichtlich. Dieses Phänomen wird als Komposita bezeichnet. Als Beispiel seien hier solche englische Begriffe wie ice cream oder hot dog genannt. Komposita in Form von mehreren getrennten Termen kommen im Englischen viel häufiger als im Deutschen vor. Typischer für das Deutsche sind Komposita in Form von zusammengesetzten Worten.

Aufgabe der Kompositabildung ist das Entdecken von Komposita in Form von mehreren getrennten Termen. Ein zu Kompositabildung entgegengesetzter Prozeß ist Kompositazerlegung. Der Prozeß ist charakteristischer für Komposita, die in Form von zusammengesetzten Worten vorliegen. Die Aufgabe dieses Prozesses ist das Zerlegen von Komposita in einzelne Terme.

Kompositabildung und -Zerlegung sind lexikalische Verfahren, die wörterbuchbasiert realisiert werden.

Homonym- und Synonym-Erkennung

Homonym- und Synonym-Erkennung werden basierend auf einem Thesaurus realisiert.

Anapher-Auflösung

In den Fällen, in denen ein Ausdruck(Referenzausdruck) auf einen Anderen(Referent) verweist bzw. stellvertretend für einen anderen Ausdruck steht, liegt eine Anapher vor. Um z.B. das Auftrittshäufigkeit von einzelnen Begriffen richtig zählen zu können ist es notwendig, die Anapher aufzulösen. D.h. es soll zu einem gegebenen Referenzausdruck der Referent determiniert werden. Die Anaphern werden durch syntaktische Analysen oder durch den semantischen Abstand zwischen dem Referenten und dem referenzierenden Ausdruck aufgelöst ([Sto06]).

3.3.2. Vektorraummodell (BagOfWords-Modell)

Das Vektorraummodell ist eins der am häufigsten im Information Retrieval verwendeten Dokumentenmodellen. Es wurde in den 1960er und 1970er Jahren von Gerard Salton in seiner Arbeit am SMART-System entwickelt.

Sei D eine Dokumentkollektion und $index_D$ die Menge aller indexierten Dokumenten aus D. $n := |D| = |index_D|$ ist die Anzahl der Dokumente in der Dokumentenkollektion. Sei m die Gesamtzahl der nach der Indexierung in $index_D$ enthaltenen Termen:

$$m = \left| T = \left\{ t : t \in \bigcup_{i=1}^{n} index_{d_i}, index_{d_i} \in index_D \right\} \right|$$

Die Menge T ist die Menge aller in $index_D$ enthaltenen Terme und wird als *Bag Of Words* bezeichnet.

Alle indexierten Dokumente werden durch Vektoren in einem m-dimensionalen Vektorraum repräsentiert. Dabei entspricht jede Dimension des Vektorraums einem Term aus T (Abb. 3.7).

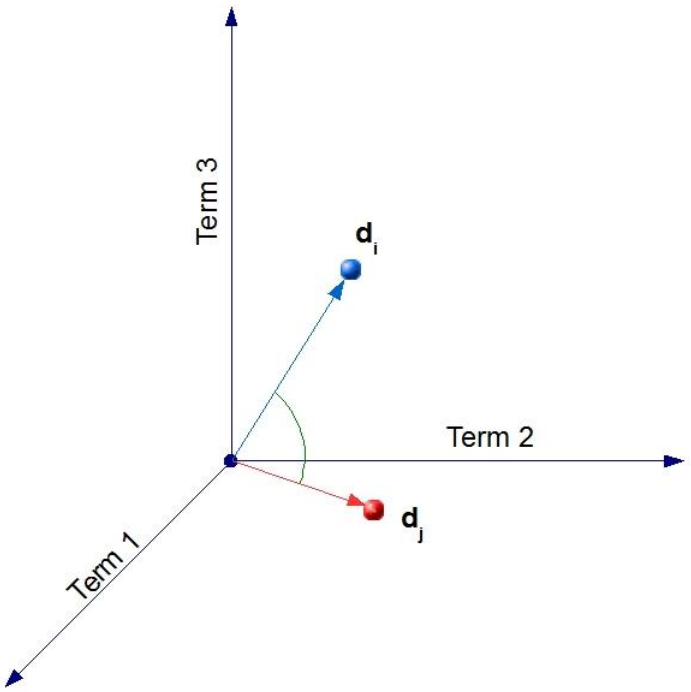

Abbildung 3.7.: Vektorraum

3.3.3. Suffix Tree Dokumentenmodell

Das Vektorraumdokumentenmodell enthält keine Informationen über die Reihenfolge der Terme in Dokumenten. Als Beispiel für Dokumentenmodelle, die solche Informationen berücksichtigen, wird hier Suffix-Tree-Dokumentenmodell vorgestellt.

Als i-ter Suffix eines Dokumentes $d = \omega_1 \dots \omega_n$ wird ein Teilstring der Länge $n-i+1$ bezeichnet, der mit dem Wort ω_i beginnt [MSP05]. Ein Suffix Tree ist ein gewurzelter Baum, der alle Suffixe aller in der Kollektion enthaltenen Dokumente enthält. Der Baum wird so konstruiert, dass beipspielsweise für das längste Suffix aus einem Dokument $d_1 \in D$ einen Ast ausgehend vom Wurzelknoten gezogen (s. Abbildung 3.8) wird. Die dabei entstandene Kante wird mit dem das Suffix bildenden String markiert. Anschließend werden entsprechend Äste für alle kleineren Suffixe aus d gezogen, die als Anfangswort ein Wort, das sich von allen Anfangsworten der schon im Baum enthaltenen Suffixrepräsentationen unterscheidet, haben. Wenn ein Suffix dagegen mit einem solchen Wort ω_i anfängt, wird seine Repräsentation in den bestehenden Ast integriert. Die Integration erfolgt derart, dass die Kante an der Stelle, ab der die Suffixe sich unterscheiden, geteilt wird. Dabei wird ein neuer Knoten hinzugefügt, und an diesem Knoten erfolgt eine Verzweigung. Da die Verzweigung nur an der Stelle erfolgt, an der sich die Suffixe unterscheiden, wird es gewährleistet, dass jeder innere Knoten mindestens zwei Nachfolgeknoten hat, wobei die Nachfolger nie mit dem gleichen Wort anfangen.

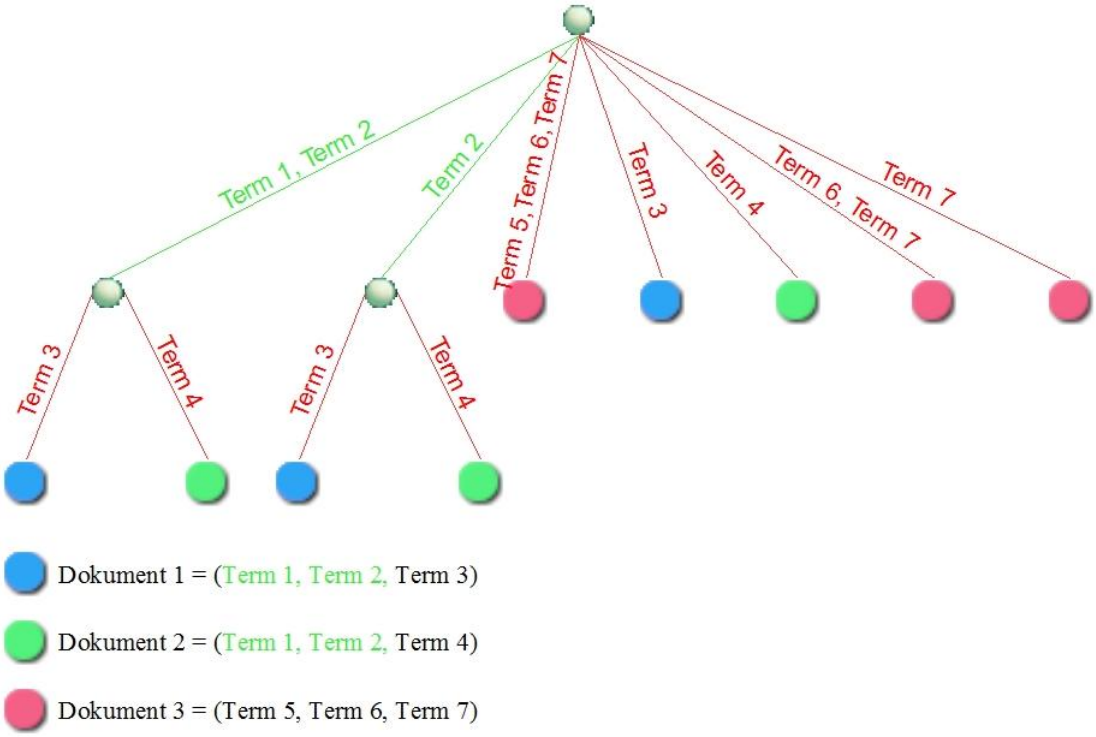

Abbildung 3.8.: Suffix Tree Modell

Jeder innere Knoten im Suffix Tree repräsentiert eine Menge von Dokumenten und einen Teilstring bzw. eine Phrase, die für alle Dokumente in dieser Menge gemeinsam ist [ZE98]. Die Proximitätsfunktion im Suffix-Tree-Modell kann beispielsweise aus dem Überlap-

pungsgrad von Teilbäumen abgeleitet werden.

3.3.4. Gewichtung der Dokumentterme

Im Vektorraummodell werden einzelne Dokumente auf Vektoren eines endlichen Vektorraums abgebildet. Dabei soll jede Komponente des Vektors Repräsentanz eines bestimmten Terms für den Inhalt des gesamten Dokumentes vertreten. Solche quantitavive Größe $\omega(t, d)$ wird als Gewicht des Terms t im Dokument d bezeichnet.

Absolute Termhäufigkeit

Die von Hans Peter Luhn stammende sog. "These von Luhn" besagt, dass sowohl die Terme, die sehr große, als auch die, die sehr kleine Auftrittshäufigkeit in einem Dokument aufweisen, kaum Diskriminanz besitzen [Sto06]. Daraus folgt, dass nur Terme aus dem mittleren Häufigkeitsbereich für die inhaltliche Beschreibung des Dokuments eine signifikante Rolle spielen. Deshalb werden die häufigsten Terme nicht in die Gewichtung einbezogen (s. Unterabschnitt 3.3.1).

Daraus resultiert, dass die *absolute Häufigkeit* eines Terms im Dokument die Bedeutung dieses Terms für den Inhalt des Dokuments angibt:

$$\omega_{tf}(t, d) = tf(t, d)$$

Relative Termhäufigkeit

Die Benutzung von absoluter Termhäufigkeit für die Gewichtung der Terme führt allerdings dazu, dass zwei Dokumente, die zwar sehr ähnlich zueinander sind, aber unterschiedlichen Längen bei gleicher Verteilung eines Terms haben, ferner voneinander im Vektorraum platziert werden als ähnliche Dokumente mit der gleichen Länge. D.h., es wird auch die Länge der Dokumente in die Beurteilung der Ähnlichkeit bzw. Unähnlichkeit dieser einbezogen.

Um den durch verschiedenen Dokumentlängen verursachten Störfaktor auszuschließen, sollte die Gewichtung über die Dokumentlänge normiert werden. Daraus ergibt sich die relative Häufigkeit eines Term t im Dokument d:

$$\omega_{tf_l}(t, d) = \frac{tf(t, d)}{l(d)}$$

Inverse Dokumenthäufigkeit

Terme innerhalb von Dokumentenkollektionen sind unterschiedlich zu Diskrimination der Dokumente geeignet. Wenn ein Term in allen Dokumenten vorhanden ist, besitzt er keine Diskriminanzkraft. Ferner gilt, je mehr Dokumente einen Term enthalten, desto kleiner ist sein Unterscheidungspotetial und umgekehrt. Aus diesen Vorüberlegungen folgt, dass die Terme, die in vielen Dokumenten vorkommen, weniger zur semantischen Trennung der Dokumente beitragen können, und aus diesem Grund kleinere Gewichte bekommen müssen.

$$idf(t) = ld\left(\frac{N}{n}\right)$$

Um die resultierenden Werte auf positive Zahlen zu beschränken, wird die Formel folgendermaßen erweitert:

$$idf(t) = ld\left(\frac{N}{n} + 1\right)$$

Tf-Idf

Da die inverse Dokumenthäufigkeit nur das Unterscheidungspotential von einzelnen Termen widerspiegelt und nicht dokumentspezifisch ist, ist sie für eine direkte Anwendung bei der Gewichtung der Dokumentterme nicht geeignet. Jedoch ist sie gut zur Anpassung von schon nach einem dokumentenspezifischen Verfahren berechneten Gewicht geeignet, um die Trennfähigkeit der Terme hervorzuheben.

Um die Verteilung der Terme sowohl in einem Dokument als auch in der ganzen Kollektion zu berücksichtigen, wird relative Häufigkeit mit der inversen Dokumenthäufigkeit kombiniert:

$$\omega_{tfIdf}(t,d) = \omega_{tf_l}(t,d) \cdot idf(t)$$

3.4. Maß zum Vergleich von Dokumenten

Um Ähnlichkeit bzw. Unähnlichkeit zwischen zwei Dokumentvektoren zu bestimmen, wird ein quantitatives Maß gebraucht. Die Maße, die im Dokumenten-Clustering benutzt werden, sind dem Bereich der Clusteranalyse entnommen (s. Abschnitt 2.3). Im Falle vom Dokumenten-Clustering mit Vektorraummodell sind Komponenten jedes Vektors metrischskalierte Variablen. Somit sind die Vektoren selbst metrischskaliert und die allgemeine Proximitätsfunktion

$$\rho : X \times X \to \mathbb{R}$$

wird auf metrische Parameter begrenzt:

$$\rho : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$$

Im Folgenden wird ein kleiner Auswahl von metrischen Proximitätsmaßen vorgestellt. Nach wie vor wird dabei zwischen Ähnlichkeits- und Distanzmaßen unterschieden.

Euklidische Distanz Ein Unähnlichkeitsmaß, das seine Verwendung im Vektorraum-Dokumenten-Clustering findet, ist die Euklidische Distanz. Dabei werden für jedes Dokumentenpaar die Differenzwerte der Gewichte jedes Terms quadriert und anschließend summiert und radiziert.

$$D(\mathbf{d}_i, \mathbf{d}_j) = \sqrt{\sum_{k=1}^{n} (\omega(t_k, d_i) - \omega(t_k, d_j))^2}$$

Die euklidische Distanz entspricht für zwei und drei Dimensionen dem geometrischen Abstand.

Skalarprodukt Als geometrisches Ähnlichkeitsmaß zweier Dokumenten bietet sich das Skalarprodukt der entsprechenden Vektorrepräsentationen an.

$$\rho(\mathbf{d}_i, \mathbf{d}_j) = \sum_{i=1}^{n} (\omega(t_k, d_i) \cdot \omega(t_k, d_j))$$

Das Skalarprodukt ist von den Längen der beiden Vektoren abhängig.

Kosinusähnlichkeit Ein Ähnlichkeitsmaß, das unabhängig von den Längen der Vektoren ist, ist die Kosinusänhlichkeit. Die Kosinusänhlichkeit ergibt sich aus dem Skalarprodukt durch das Normieren der Vektoren auf die Einheitslänge.

$$\rho(\mathbf{d}_i, \mathbf{d}_j) = \frac{\sum_{i=1}^{n} (\omega(t_k, d_i) \cdot \omega(t_k, d_j))}{\sqrt{\sum_{i=1}^{n} \omega(t_k, d_i)^2} \cdot \sqrt{\sum_{i=1}^{n} \omega(t_k, d_i)^2}}$$

4. MajorClust-Familie

In diesem Kapitel wird ein Versuch unternommen, Clustering-Algorithmen zu systematisieren. Anschließend wird der MajorClust-Algorithmus näher beschrieben und auf seine verschiedenen Modifikationen eingegangen.

4.1. Clustering-Algorithmen

Der Prozess der Clusteranalyse bzw. des Dokumenten-Clusterings durchläuft im Wesentlichen drei Phasen:

1. Aufbereitung von Daten bzw. Dokumentenmodellbildung
2. Bestimmung von Ähnlichkeiten zwischen Datenobjekten bzw. Dokumentenrepräsentationen
3. Gruppieren von Datenobjekten bzw. Dokumentenrepräsentationen

Im Falle des Dokumenten-Clusterings besteht die erste Phase aus dem mehrstufigen Indexierungsprozess (s. Unterabschnitt 3.3.1), der Auswahl eines Dokumentenmodells (3.3.2) und ggf. der Gewichtung der Dokumentterme (3.3.4)

In der zweiten Phase wird ein Proximitätsmaß ausgewählt und anschließend werden Ähnlichkeits- bzw. Unähnlichkeitswerte für jedes Dokumentenpaar berechnet. Eine kleine Auswahl der beim Dokumenten-Clustering zu verwendenden Proximitätsmaßen wurden im Abschnitt 3.4 vorgestellt.

Während die ersten zwei Phasen lediglich als Vorbereitungsprozesse betrachtet werden können, übernimmt die dritte Phase die eigentliche Clusteranalyse. Dabei ist die Auswahl eines geeigneten Mechanismus entscheidend für die Qualität des Ergebnisses.

4.1.1. Einteilung der Clusteralgorithmen

Die Einteilung der Clusteralgorithmen ist im Allgemeinen nicht einheitlich und lässt sich nach verschiedenen Kriterien vornehmen.

Werden Clusteralgorithmen nach der Form des Resultats systematisiert, spricht man von der schon in Kapitel 2 erwähnten Einteilung in *harte* und *weiche Clusteralgorithmen* in Abhängigkeit davon, ob die resultierenden Cluster disjunkt oder nicht-disjunkt sind.

Darüber hinaus werden Algorithmen als *exhaustiv* bezeichnet, wenn alle Objekte in Clustern enthalten sind. Werden dagegen einige Objekte keinem Cluster zugeordnet, wird der Algorithmus als *nicht-exhaustiv* bezeichnet.

Die in [Stec] vorgeschlagene Einteilung systematisiert Clusteralgorithmen nach der Funktionsweise des Gruppierungsprozesses (s.Abbildung 4.1). Danach werden Algorithmen in fünf Gruppen geteilt, die im Folgenden näher beschrieben werden.

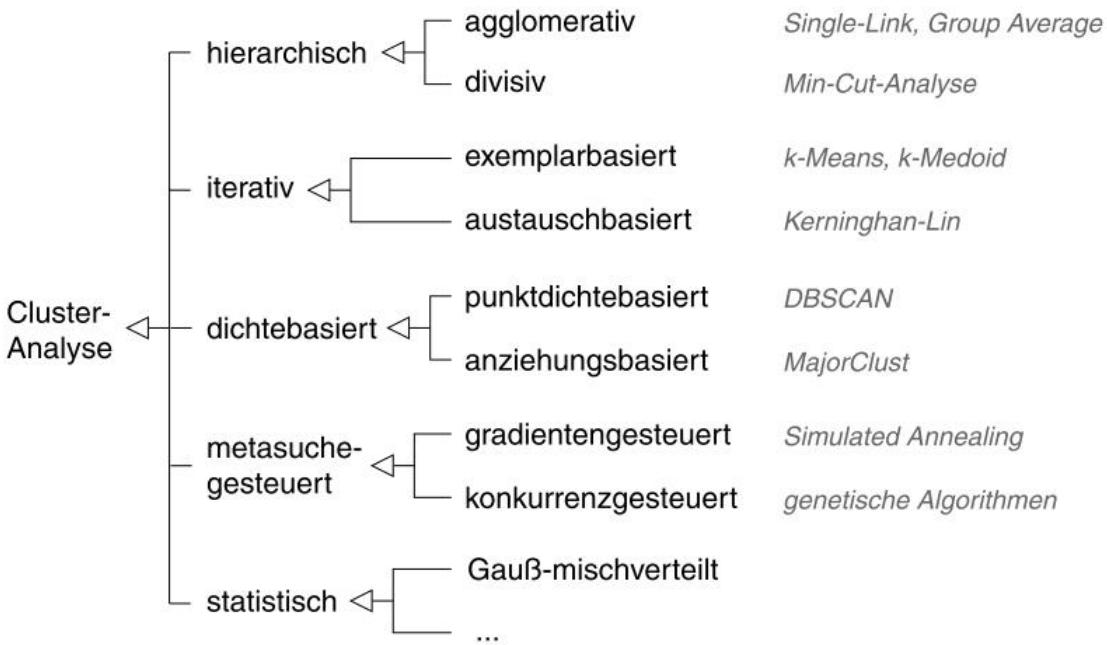

Abbildung 4.1.: Systematisierung der Clustering-Algorithmen (Quelle: [Stec])

Hierarchische Clustering-Algorithmen

Hierarchische Clustering-Algorithmen lassen sich grundsätzlich in agglomerative und divisive einteilen. *Agglomerative* Algorithmen gehen von der feinsten Partition, in der jeder Knoten in einem eigenen Cluster liegt, aus und vergröbern diese Partition in jedem Schritt, in dem kleinere Cluster zu größeren fusionieren. Im Allgemeinen endet der Fusionierungsprozess, wenn die ganze Partition nur aus einem Cluster besteht. *Divisive* Algorithmen beginnen dagegen mit der gröbsten Partition und verfeinern diese schrittweise bis die feinste Partition entsteht.

Bei hierarchischen Algorithmen entsteht in jedem Schritt ein neues Clustering. Die Hierarchie von diesen Clusterings lässt sich mit Hilfe eines Dendrogramms visualisieren.

Eins der Hauptmerkmale der hierarchischen Clustering-Algorithmen besteht darin, dass Knoten ihre Clusterzugehörigkeit im Laufe der Fusionierung nur dann wechseln, wenn zwei Cluster zu einem Neuen verschmelzen, wie es im Falle von agglomerativen Algorithmen geschieht, bzw. ein Cluster in zwei Neuen geteilt wird (im Falle von divisiven Algorithmen).

Iterative Clustering-Algorithmen

Iterative Clustering-Algorithmen beginnen mit einer Startpartition, die die gewünschte Anzahl an Clustern enthält, und der Algorithmus versucht iterativ diese Partition durch Verlagerung von Objekten zwischen den Clustern so zu verändern, dass eine gewünschte Eigenschaft optimiert wird. Lässt sich diese Eigenschaft nicht mehr verbessern, terminiert der Algorithmus.

Iterative Clustering-Algorithmen werden ferner in zwei Untergruppen geteilt. Bei *exemplarbasierte* Algorithmen wird für jedes Cluster ein repräsentatives Objekt bestimmt, zu dem die Objekte aufgrund einer Proximitätsfunktion zugeordnet werden. Dabei kann das repräsentative Objekt ein Objekt aus der Kollektionsmenge sein, oder aber ein virtuelles Objekt kann beispielsweise mittels Berechnung des Mittelwertes aller im Cluster vorhandenen Objekte erschaffen werden. Bei *austauschbasierten* Algorithmen werden einzelne Objekte aus verschiedenen Clustern vertauscht, um die Zieleigenschaft zu verbessern.

Als Beispiel für iterative exemplarbasierte Clustering-Algorithmen wird im folgenden k-Means-Algorithmus beschrieben.

k-Means-Algorithmus ist ein weit verbreiteter und gut erforschter Clustering-Algorithmus. Der Algorithmus besteht aus folgenden vier Schritten:

1. Bestimme die Anfangspartition
2. Berechne das Zentrum für jedes Cluster
3. Ordne jedes Objekt dem am nächsten liegenden Zentrum zu
4. Wenn bei der Verteilung im 3. Schritt mindestens ein Objekt einem neuen Cluster zugeordnet wurde, gehe zu Schritt 2

Dabei kann die Anfangspartition durch zufälliges Auswählen von k Zentren und darauf folgendem Zuordnen jedes Knotens zum am nächsten liegenden Zentrum gebildet

werden. Alternativ kann aber auch ein anderer Algorithmus zum Erstellen der Startpartition benutzt werden. Beispielsweise kann ein hierarchischer Clustering-Algorithmus zum Determinieren der optimalen Clusteranzahl k und dem Erstellen der Startpartition verwendet werden.

Dichtebasierte Clustering-Algorithmen

Dichtebasierte Algorithmen basieren auf der Vorstellung, dass Objekte in einem mehrdimensionalen Vektorraum nicht gleichmäßig verteilt sind, sondern Regionen mit höherer und geringerer Dichte bilden. Die Regionen mit der höheren Dichte stellen eine Anhäufung von zueinander ähnlichen Objekten dar und werden als Cluster erkannt.

Der in dieser Arbeit zu evaluierende Algorithmus MajorClust und seine Variationen werden den dichtebasierten Clustering-Algorithmen zugeordnet. Ihre Funktionsweise wird in späteren Abschnitten beschrieben.

4.1.2. Strukturen zur Repräsentation von Eingangsdaten

Die allen Clustering-Algorithmen gemeinsame Ausgangsdatenstruktur ist die als Ergebnis der Indexierung und Termgewichtung hervorgehende schon im Abschnitt 2.3 beschriebene Rohdatenmatrix (s. Tabelle 2.1). Die meisten Clustering-Algorithmen benötigen für ihre Artbeit aber keine in der Rohdatenmatrix enthaltenen Termgewichte, sondern nur die daraus resultierenden Proximitätsmaße. Eine Datenstruktur, die solche Werte für alle Knotenpaare einer Objektmenge enthält, wird in Abhängigkeit von der verwendeten Proximitätsfunktion Ähnlichkeits- bzw. Unähnlichkeits-(Distanz-)Matrix genannt (s. Tabelle 2.2). Eine Ausnahme stellen iterative Clustering-Algorithmen, wie z.B. k-Means, dar, die aufgrund ihrer Funktionsweise keine Proximitätswerte von realen Objektpaaren verwenden können. Solche Algorithmen benötigen nur Proximitätswerte zwischen realen und repräsentativen (virtuellen) Objekten. Da repräsentative Objekte im Voraus nicht bekannt sind, bleibt einem nicht viel anderes übrig als direkt mit Rohdatenmatrizen zu arbeiten.

Eine Alternative für Ähnlichkeitsmatrix ist die Darstellung der Objektenproximitäten in Form eines ungerichteten gewichteten Graphen $G = (V, E, \omega)$. Die Knotenmenge V entspricht dabei der Objektmenge, die Kantenmenge E den Beziehungen zwischen Objekten. Die Funktion ω determiniert den Grad der Nachbarschaftsbeziehungen, der den Proximitätswerten entspricht:

$$\omega : E \rightarrow \mathbb{R}$$

bzw.

$$\omega(e_{i,j}) = \omega(v_i, v_j) = \omega(\mathbf{d}_i, \mathbf{d}_j) = a$$

wobei

$$e_{ij} \in E,\ v_i, v_j \in V,\ a \in \mathbb{R}$$

4.2. MajorClust

Bei dem in [SN99] erstmals publizierten MajorClust-Algorithmus werden Kantengewichte als Kräfte interpretiert. Dabei bündeln die Kanten, die einen Knoten mit einem Cluster verbinden, ihre Kräfte. Der Algorithmus geht von der feinsten Partition aus, in der jeder einzelne Knoten einem separaten Cluster zugeordnet wird. In jeder Iteration des Algorithmus wird für jeden Knoten ein Cluster ermittelt, der auf diesen Knoten die größte Anziehungskraft ausübt. Wenn der ermittelte Cluster nicht mit der aktuellen Zuordnung des Knotens übereinstimmt, wird der Knoten dem ermittelten Cluster zugeordnet. Findet im Laufe einer Iteration keine Neuzuordnung statt, terminiert der Algorithmus.

Algorithmus 1 beschreibt den formalisierten implementierungsunabhängigen Ablauf des MajorClust-Algorithmus.

MajorClust
Input : A graph $G = \langle V; E \rangle$.
Output : A function $c : V \mapsto \mathbb{N}$ which assigns a cluster number to each node.

$n = 0, t = false$
forall $v \in V$ **do** $n = n + 1, c(v) = n$ **end**
while $t = false$ **do**
  $t = true$
  forall $v \in V$ **do**
    $c^* = \arg\max_{i:i\in\{1,\ldots,|V|\}} \sum_{\{u,v\}:\{u,v\}\in E \wedge c(u)=i} \omega(u, v)$
    if $c(v) \neq c^*$ **then** $c(v) = c^*, t = false$
  end
end

Algorithmus 1 : MajorClust (Quelle: [SN99][Steb])

Zu den Vorteilen des MajorClust-Algorithmus zählt seine Fähigkeit, selbstständig die Anzahl der Cluster zu determinieren. Zu den Nachteilen gehört seine Ergebnisqualität beeinflussende Determiniertheit, die durch Randomisierung der Ablaufreihenfolge der einzelnen Knoten entsteht. Diese Randomisierung wird ihrerseits durch Ungewißheit der optimalen Reihenfolge hervorgerufen.

Ein weiterer Nachteil des MajorClust-Algorithmus ist, dass er bei Eingabe von beinahe homogenen Graphen zu feine Partitionen liefert. Mit im Folgenden beschriebenen Modifikationen des Algorithmus wird versucht, diesen Nachteil zu eliminieren bzw. zu entschärfen.

4.2.1. ExtendedMajorClust

ExtendedMajorClust ist eine von Tim Gollub entwickelte Erweiterung des MajorClust-Algorithmus. Im Gegensatz zum MajorClust iteriert der Algorithmus in jedem Durchgang einmal durch alle Knoten und einmal durch alle Cluster, d.h. es wird in jedem Durchgang zusätzlich für jedes Cluster geprüft, ob die Summe der Anziehungskräfte innerhalb dieses Clusters größer ist als die Summe der Kräfte, die dieses Cluster und jeweil ein Anderer gegenseitig aufeinander ausüben. Ist das nicht der Fall, werden diese Cluster vereinigt.

ExtendedMajorClust
Input : A graph $G = \langle V; E \rangle$.
Output : A function $c : V \mapsto \mathbb{N}$ which assigns a cluster number to each node.

$n = 0, t = false$
forall $v \in V$ **do** $n = n + 1, c(v) = n$ **end**
while $t = false$ **do**
  $t = true$
  forall $v \in V$ **do**
    $c^* = \arg\max_{i:i\in\{1,\ldots,|V|\}} \sum_{\{u,v\}:\{u,v\}\in E \wedge c(u)=i} \omega(u,v)$
    if $c(v) \neq c^*$ **then** $c(v) = c^*, t = false$
  end
  forall $l \in c(V)$ **do**
    forall $m \in c(V) \setminus l$ **do**
      if $\sum_{\{u,v\}:\{u,v\}\in E \wedge c(u)=c(v)=l} \omega(u,v) < \sum_{\{u,v\}:\{u,v\}\in E \wedge c(u)=l \wedge c(v)=m} \omega(u,v)$
      then
        join the clusters l and m
        $t = false$
      end
    end
  end
end

Algorithmus 2 : ExtendedMajorClust

4.2.2. StrongMajorClust

Die von Mikhail Alexandrov [Ale] entwickelte Erweiterung des MajorClust-Algorithmus setzt voraus, dass die Summe der Anziehungskräfte, die einen Knoten mit seinem eigenen Cluster verbinden, K-mal größer sein soll als die Summe der Kräfte, die diesen Knoten mit einem anderen Cluster verbinden. Ist das nicht der Fall wird die Clusterzugehörigkeit des Knotens geändert. Der Parameter K wird als eine Eingangsgröße an den Algorithmus gegeben und erlaubt die Steuerung des Grades der Objektgruppierung.

StrongMajorClust
Input : A graph $G = \langle V; E \rangle$, a parameter $K > 1$ ($K \in \mathbb{N}$).
Output : A function $c : V \mapsto \mathbb{N}$ which assigns a cluster number to each node.

$n = 0$
forall $v \in V$ **do** $n = n + 1, c(v) = n$ **end**
while *clustering not stable* **do**
  forall $v \in V$ **do**
    $c^* = \arg\max_{i:i\in\{1,\ldots,|V|\}} \begin{cases} \sum_{\{u,v\}:\{u,v\}\in E \wedge c(u)=i} \omega(u,v) & i \neq c(v) \\ \frac{1}{K} \cdot \sum_{\{u,v\}:\{u,v\}\in E \wedge c(u)=i} \omega(u,v) & i = c(v) \end{cases}$
    if $c(v) \neq c^*$ **then** $c(v) = c^*$
  end
end

Algorithmus 3 : StrongMajorClust

4.3. MCProb

MCProb ist eine in [NS09] vorgestellte probabilistische Variante des MajorClust-Algorithmus, die hauptsächlich dafür gedacht ist, den Nachteil des MajorClust beim Clustering von beinahe homogenen Graphen zu überwinden.

In allen Algorithmen der MajorClust-Familie wird der Cluster, der die größte Anziehungskraft auf einen Knoten ausübt, als Mehrheitsentscheidung für die Neuzuordnung dieses Knotens interpretiert. Die Zuordnung zu anderen Cluster, die kleinere Kräfte auf den Knoten ausüben, wird als Minderheitsentscheidung gesehen. Die nicht-probabilistische Varianten des MajorClust-Algorithmus treffen immer Mehrheitsentscheidungen, wogegen beim MCProb jeder Cluster, der eine Anziehungskraft auf den Knoten ausübt, eine Möglichkeit für eine Neuzuordnug bekommt. Dabei bekommt jeder solche Cluster einen Wahrscheinlichkeitswert, mit dem der Cluster als Zuordnungskandidat betrachtet wird. Der Wahrscheinlichkeitswert steht dabei in der Abhängigkeit von der ausgeübten Anziehungskraft, d.h. je größer die Anziehungskraft ist, desto höher ist die Wahrscheinlichkeit, die dem Cluster zugewiesen wird.

Die Auswahl des Clusters für die Neuzuordnung erfolgt zufällig entsprechend den jedem Cluster zugewiesenen Wahrscheinlichkeitswerten. Somit wird es ermöglicht, dass die Zuordnung des Knotens entsprechend einer Minderheitsentscheidung vollzogen werden kann, was bei beinahe homogenen Graphen zur Vergröberung der Ergebnispartition führen soll.

In [NS09] wurde für die Berechnung von Wahrscheinlichkeitswerten eine Normalverteilung vorgeschlagen, wobei ausdrücklich darauf hingewiesen wurde, dass auch alternative Wahrscheinlichkeitsverteilungen benutzt werden können.

MCProb
Input : A graph $G = \langle V; E \rangle$.
Output : A function $c : V \mapsto \mathbb{N}$ which assigns a cluster number to each node.

$n = 0$
forall $v \in V$ **do** $n = n + 1, c(v) = n$ **end**
while *clustering not stable* **do**
  $c(v) = i$ with probability $Z_v \cdot Norm(m_v, \sigma_v, q_v(i))$
  where $Norm$ is the gaussian distribution[1]
  at place q_v with mean m_v and standard deviation σ_v
  with $m_v = max\left\{q_v(j) | j \in c^{-1}(V)\right\}, \sigma_v = 1$
  and $q_v(x) = \sum_{u:\{u,v\}\in E} \omega(u, v), c(u) = x$
  (Z_v is a normalization factor).
end

Algorithmus 4 : MCProb (vgl. [NS09])

4.3.1. Abbruchkriterien

Der MajorClust-Algorithmus terminiert, wenn keine Neuzuordnung innerhalb einer Iteration erfolgt. Das gleiche Abbruchkriterium ist auch für ExtendedMajorClust geeignet. Für StrongMajorClust und MCProb hat er sich aber als ungeeignet erwiesen. Der Grund dafür kann mit einem Beispiel demonstriert werden.

Der Leser stelle sich eine Situation vor, wo auf einen Knoten 91 Clustern ihre Anziehungskraft ausüben. Dabei ist die größte Anziehungskraft so, dass ihr zugewiesener Wahrscheinlichkeitswert 10% beträgt. Die anderen Anziehungskräfte sind gleich groß, und die entsprechenden Wahrscheinlichkeitswerte betragen jeweils 1%. Daraus ergibt sich, dass die Wahrscheinlichkeit, die Mehrheitsentscheidung zu treffen, bei 10% liegt, dagegen aber die Wahrscheinlichkeit, dass eine Minderheitsentscheidung getroffen wird, 90% ausmacht. Folglich wird für jeden Knoten in überwiegeder Anzahl von Fällen eine Minderheitsentscheidung getroffen. Als Ergebnis davon entsteht eine Situation, wo die Zuordnungen von einem oder mehreren Knoten sich in jeder Iteration ändern, mit anderen Worten springen ein oder mehrere Knoten von einem Cluster zum Anderen, und der Algorithmus terminiert nicht.

Aus dem oben genannten Grund wurden in dieser Arbeit neue Abbruchkriterien entwickelt, die im Folgenden kurz beschrieben werden.

[1] $Norm(m, \sigma, x) = \frac{1}{2\pi\sigma} e^{-\left(\frac{x-m}{\sigma}\right)^2}$

ClusterCounter

Bei diesem Abbruchkriterium wird nach jeder Iteration geprüft, ob die Clusteranzahl sich im Vergleich zur letzten Iteration geändert hat. Ist das nicht der Fall, terminiert der Algorithmus

Dynamic

Das dynamische Abbruchkriterium basiert auf der Vorstellung, dass bei der Annäherung des Verhaltens des Algorithmus an MajorClust, d.h. wenn der Algorithmus gezwungen wird, mit der Zeit immer mehr Mehrheitsentscheidungen zu treffen, das Abbruchkriterium des MajorClust-Algorithmus erfüllt wird.

Dieses Abbruchkriterium wurde für Normalverteilung so realisiert, dass die Standardabweichung σ in Abhängigkeit von der Anzahl in einer Iteration getroffenen Minderheitsentscheidungen angepasst wird:

$$\sigma = \frac{Anzahl\ der\ Minderheitsentscheidungen}{Anzahl\ der\ Knoten}$$

Somit wird sichergestellt, dass der Wahrheitswert für die Mehrheitsentscheidung um so größer und die für die Minderheitsendscheidung um so kleiner werden, um so kleiner σ wird (s. Abbildung 4.2).

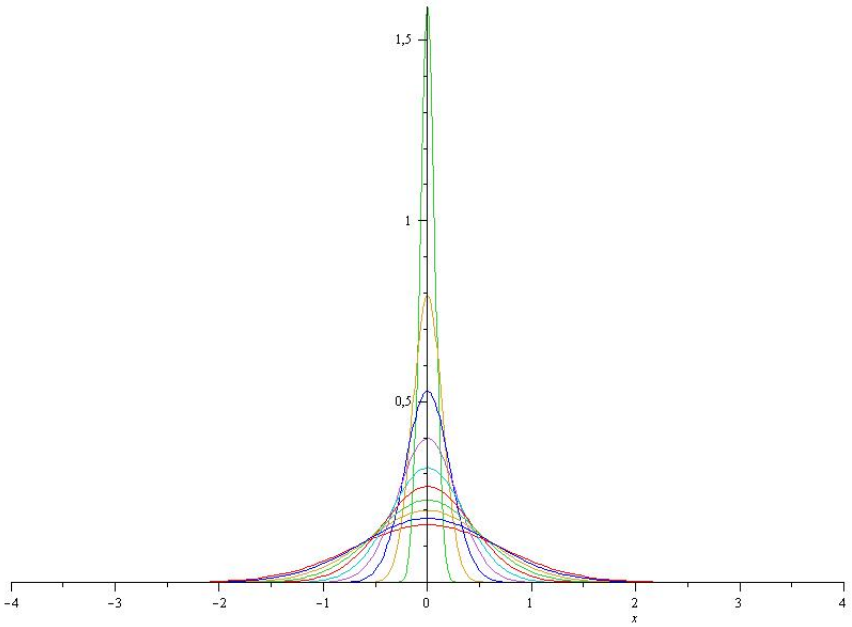

Abbildung 4.2.: Normalverteilung

Das dynamische Abbruchkriterium garantiert eine Terminierung des Algorithmus, da mit der Verkleinerung der Standardabweichung mehr Mehrheitsentscheidungen getroffen werden und σ als Folge wieder kleiner wird.

4.3.2. Einsatz von alternativen Wahrscheinlichkeitsverteilungen

Als eine Alternative zur Normalverteilung wurde in dieser Arbeit auch die Exponentialverteilung eingesetzt.

Die Dichtefunktion für Exponentialverteilung ist folgendermaßen definiert:

$$f(x) = \begin{cases} 0, & x < 0 \\ \lambda \cdot e^{-\lambda \cdot x}, & x \geq 0 \end{cases}$$

Beim dynamischen Abbruchkriterium wird der Parameter λ vergrößert (s. Abbildung 4.3), was den gleichen Effekt erzielt wie im Falle von Normalverteilung.

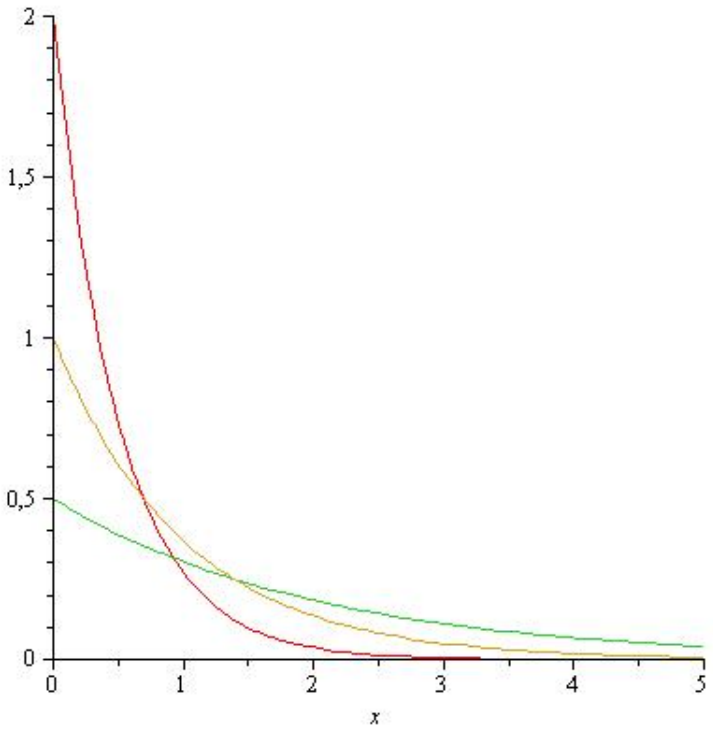

Abbildung 4.3.: Exponentialverteilung

4.3.3. BalancedMCProb

Eine alternative in dieser Arbeit entwickelte Variante des MCProb-Algorithmus basiert auf der Idee, Wahrscheinlichkeiten für Mehr- und Minderheitsentscheidungen gleich zu setzten, d.h. die Wahrscheinlichkeitwerte werden so angepasst, dass sowohl die Wahrscheinlichkeit für die Mehrheitsetscheidung als auch die Summe von Wahrscheinlichkeiten für alle Minderheitsentscheidungen bei 50% liegen.

Das dynamische Abbruchkriterium ist hier so realisiert, dass jedesmal, wenn nach einer Iteration die Anzahl der Cluster im Vergleich zur letzten Iteration unverändert bleibt, die Wahrscheinlichkeit für die Mehrheitsentscheidung um 10% erhöht und entsprechend für alle Minderheitentscheidungen um 10% erniedrigt wird.

5. Evaluierung

In diesem Kapitel wird ein kurzer Überblick über verschiedenen Qualitätsmaße gegeben. Danach werden die Ergebnisse der Evaluierung von verschiedenen Algorithmen aus der MajorClust-Familie vorgestellt und diskutiert.

5.1. Qualitätsmaße

Um eine Aussage treffen zu können, inwiefern das Ergebnis einer Clusteranalyse eine optimale Lösung approximiert, wird ein quantitatives Qualitätsmaß benötigt. Je nachdem welche Informationen zur Beurteilung der Qualität des Ergebnis-Clusterings herangezogen werden, werden grundsätzlich zwei Arten von Qualitätsmaßen unterschieden:

- Externe Qualitätsmaße
- Interne Qualitätsmaße

Externe Qualitätsmaße vergleichen Ergebnis-Clustering mit einem manuell erstellten Referenz-Clustering. Interne Qualitätsmaße benutzen dagegen ausschließlich nur die Informationen, die das Ergebnis-Clustering selbst enthält.

Externe Qualitätsmaßen werden ferner in *klassifikations-* und *ähnlichkeitsbasierte* (classification-oriented and similarity-oriented) Maßen eingeteilt [TSK06]. Die klassifikationsbasierten Qualitätsmaße stammen aus dem Klassifikation-Bereich und bestimmen den Grad, zu dem ein Cluster des Ergebnis-Clusterings Objekte einer Klasse des Referenz-Clusterings enthält. Die ähnlichkeitsbasierten Qualitätsmaße bestimmen den Übereinstimmungsgrad des Ergebnis-Clusterings mit dem Referenz-Clustering in Abhängigkeit davon, ob Objekte, die im Referenz-Clustering in der gleichen Klasse liegen, sich auch in dem gleichen Cluster des Ergebnis-Clusterings befinden.

5.1.1. F-Measure

Ein weitverbreitetes und für die in dieser Arbeit durchgeführten Experimente verwendetes externes klassifikationsbasiertes Qualitätsmaß ist das F-Measure (oder auch F-Maß

genannt). F-Measure ist ein kombiniertes Maß, das aus zwei anderen externen Qualitätsmaßen besteht.

Precision. Sei D eine Objektmenge, $C^{ref} = \left\{C_1^{ref}, C_2^{ref}, \ldots, C_m^{ref}\right\}$ ein Referenz-Clustering und $C = \{C_1, C_2, \ldots, C_n\}$ ein Ergebnis-Clustering. Wobei

$$\bigcup_{i=1}^{m} C_i^{ref} = \bigcup_{i=1}^{n} C_i = D$$

Precision ist ein quantitatives Maß, das die Reinheit eines Clusters $C_j \in C$ bezüglich eines anderen Clusters $C_i^{ref} \in C^{ref}$ angibt [Mey07].

$$precision : D_i \times D_j \rightarrow [0; 1]$$
$$D_i, D_j \subset D$$

Precision wird folgendermaßen berechnet:

$$precision(C_i^{ref}, C_j) = \frac{\left|C_i^{ref} \cap C_j\right|}{|C_j|}$$

Recall. Recall gibt den Anteil an Dokumenten aus dem Cluster C_j an, die auch im Cluster C_i enthalten sind.

$$recall(C_i^{ref}, C_j) = \frac{\left|C_i^{ref} \cap C_j\right|}{\left|C_i^{ref}\right|}$$

F-Measure. Precision und Recall einzeln genommen können jedoch nicht als zuverlässige Kenngrößen für die Qualität eines Clusterings angesehen werden. Der Grund dafür ist die leichte Konstruktion von trivialen Fällen, in denen diese Größen ihre maximale Werte einnehmen. Der triviale Fall für Precision entsteht, wenn im Ergebnis-Clustering jedes Dokument sich in einem eigenen Cluster befindet. Für Recall wäre so ein Fall ein Cluster mit allen Dokumenten.

Bei einem hochqualitativen Clustering sollten demnach beide Kenngrößen große Werte aufweisen. Das F-Measure ergibt sich aus dem harmonischen Mittel von Precision und Recall:

$$F(C_i^{ref}, C_j) = \frac{2 \cdot precision(C_i^{ref}, C_j) \cdot recall(C_i^{ref}, C_j)}{precision(C_i^{ref}, C_j) + recall(C_i^{ref}, C_j)}$$

Das gemittelte F-Measure lässt sich auf zweierlei Weise berechnen. Beim Micro-Averaging gehen einzelne Werte gewichtet bezüglich der Clustergröße in den Mittelwert ein:

$$F_{micro} = \sum_{i=1}^{m} \frac{\left|C_i^{ref}\right|}{|D|} \cdot \max_{j=1,\ldots,n} \left\{ F(C_i^{ref}, C_j) \right\}$$

Beim Macro-Averaging werden alle Werte gleich gewichtet:

$$F_{macro} = \frac{1}{m} \sum_{i=1}^{m} \max_{j=1,\ldots,n} \left\{ F(C_i^{ref}, C_j) \right\}$$

Beim Micro-Averaging werden *alle Dokumente*, beim Macro-Averaging dagegen *alle Cluster* als gleichberechtigt behandelt [Mey07].

In den Experimenten, die in dieser Arbeit durchgeführt werden, wird das Micro-Averaged-F-Measure verwendet.

5.1.2. RandIndex

RandIndex ist ein externes ähnlichkeitsbasiertes Qualitätsmaß, das folgendermaßen berechnet wird:

Es werden die folgenden vier Variablen definiert:

- f_{00} Anzahl der Objekte-Paare, bei denen beide Objekte sowohl im Ergebnis-Clustering als auch im Referenz-Clustering in einem Cluster liegen
- f_{01} Anzahl der Objekte-Paare, bei denen beide Objekte im Ergebnis-Clustering in verschiedenen, im Referenz-Clustering aber in dem gleichen Cluster liegen
- f_{10} Anzahl der Objekte-Paare, bei denen beide Objekte zwar im Ergebnis-Clustering im selben Cluster liegen, im Referenz-Clustering aber in verschiedenen
- f_{11} Anzahl der Objekte-Paare, bei denen beide Objekte sowohl im Ergebnis-, als auch im Referenz-Clustering in verschiedenen Clustern liegen

$$R = \frac{f_{00} + f_{11}}{f_{00} + f_{01} + f_{10} + f_{11}} = \frac{f_{00} + f_{11}}{\binom{n}{2}} = \frac{f_{00} + f_{11}}{\frac{n(n-1)}{2}}$$

Wobei n die Anzahl der Objekte ist.

Der Wertebereich des RandIndexes entspricht dem Intervall $[0, 1]$

5.2. Experimente

Um objektiv beurteilen zu können, wie gut die Algorithmen der MajorClust-Familie die Herausforderung des Dokumenten-Clusterings bewältigen können und um die Algorithmen miteinander vergleichen zu können, wurden in dieser Arbeit mehrere Experimente durchgeführt. Als Basis für diese Experimente diente der *Reuters Corpus Volume 1 (RCV1)*.

RCV1 ist eine Sammlung von über 800000 kurzen englischsprachigen Nachrichtenartikeln, die von der Nachrichtenagentur Reuters 2003 für wissenschaftliche Untersuchungen herausgegeben wurde [LYRL04]. Alle Artikeln der Kollektion sind manuell klassifiziert, was die leichte Erstellung von Referenz-Clusterings und somit den Einsatz von externen Qualitätsmaßen in den Experimenten erlaubt.

Auf der Basis des RCV1 wurden sieben Testkollektionen erstellt, die sich durch eine unterschiedliche Anzahl an Dokumentenklassen, eine unterschiedliche Anzahl von Dokumenten aus jeder Klasse und gleichmäßiger bzw. ungleichmäßiger Verteilung von Dokumenten über Klassen auszeichnen. Einen Überblick über die erstellten Testkollektionen bietet die Tabelle 5.1. Darin ist die Anzahl der Dokumente aus verschiedenen Themenbereichen angegeben, wobei keins der Themen sich mit den Anderen in der Themenhierarchie überschneidet.

Kollektion	C11	E131	GCAT	GSCI	GSPO	M143
RC2E200	100	-	-	-	-	100
RC2E2000	1000	-	-	-	-	1000
RC6E120	20	20	20	20	20	20
RC6E600	100	100	100	100	100	100
RC6U1550	500	400	300	200	100	50
RC6U1650	500	500	500	50	50	50
RC6E3000	500	500	500	500	500	500

Tabelle 5.1.: In den Experimenten verwendete Dokumentenkollektionen

C11 Strategy/Plans

E131 Consumer Prices

GCAT Government/Social

GSCI Science and Technology

GSPO Sports

M143 Energy Markets

Alle Experimente wurden mit insgesamt zehn Algorithmen durchgeführt (s. 5.2.1), wobei die Ergebnise vom k-Means-Algorithmus als anzustrebenede Vergleichswerte benutzt wurden. Hier ist zu beachten, dass k-Means einen großen Vorteil gegenüber den anderen Algorithmen erhält, in dem ihm die richtige Anzahl der Cluster im Voraus mitgeteilt wird.

5.2.1. In den Experimenten verwendete Algorithmen

1. *k-Means.* (s. 4.1.1)
2. *MajorClust.* (s. 4.2)
3. *ExtendedMajorClust.* (s. 4.2.1)
4. *StrongMajorClust_CC.* StrongMajorClust-Algorithmus (s. 4.2.2) implementiert mit dem ClusterCounter-Abbruchkriterium.
5. *MCProb_CC.* MCProb-Algorithmus (s. 4.3) implementiert mit dem ClusterCounter-Abbruchkriterium.
6. *ExponentialMCProb_CC.* MCProb-Algorithmus mit Exponentialverteilung implementiert mit dem ClusterCounter-Abbruchkriterium.
7. *DynamicMCProb.* MCProb-Algorithmus implementiert mit dem dynamischen Abbruchkriterium.
8. *DynamicExponentialMCProb.* MCProb-Algorithmus mit Exponentialverteilung implementiert mit dem dynamischen Abbruchkriterium.
9. *BalancedMCProb_CC.* BalancedMCProb (s. 4.3.3) implementiert mit dem ClusterCounter-Abbruchkriterium.
10. *BalancedMCProb_CCd.* BalancedMCProb implementiert mit dem dynamischen Abbruchkriterium.

5.2.2. Datenaufbereitung und Ablauf der Experimente

In den durchgeführten Experimenten wurden im Zuge des Indexierungsprozesses aus allen Dokumenten der Testkollektionen die Stoppwörter entfernt und die restlichen Terme mit Hilfe vom Porter-Stemmer auf ihre Stammforme zurückgeführt. Anschließend wurden die Dokumententerme mit Hilfe der Tf-(Plain-) und TfIdf-Indexer gewichtet. Da die Ergebnisse für TfIdf-Indexer sich als schlechter gegenüber den Ergebnissen des Tf-Indexers erwiesen haben, werden sie im weiteren nicht betrachtet, sind aber im Anhang A zu finden.

Als Proximitätsmaß wurde die Kosinusähnlichkeit gewählt.

Darüber hinaus wurde in den Experimenten *Expected Similarity* in zwei Variationen verwendet, um das Rauschen auszuschließen (Beschreibung in [Gol08]). Mit *Average Expected Similarity* wurde eine erhebliche Verschlechterung aller Ergebnisse gegenüber *Harmonic Expected Similarity* festgestellt (s. Anhang B), weswegen im Weiteren nur die unter Verwendung von Harmonic Expected Similarity gewonnenen Ergebnisse betrachtet werden.

5.2.3. Ergebnisse

Die Ergebnisse der Experimente mit Harmonic Expected Similarity und dem PlainIndexer sind in der Tabelle 5.2 und in der Abbildung 5.1 als Diagramm dargestellt. Dabei wurde jeder Algorithmus jeweils zehnmal pro Testkollektion ausgeführt, die Ergebnisse für jede Testkollektion gemittelt, und anschließend Mittelwerte über alle sieben Testkollektionen gebildet.

Für die Bestimmung der Clustering-Qualität wurden die externen Qualitätsmaße F-Measure und RandIndex verwendet.

Aus dem Diagramm in der Abbildung 5.1 lässt sich leicht ablesen, dass die beiden Maße, F-Measure und RandIndex, beinahe gleiche Aussagen über die Qualität des Clusterings liefern. Den beiden Maßen folgend schneiden alle nicht-probabilistischen Algorithmen (MajorClust, ExtendedMajorClust und StrongMajorClust) sehr gut ab, wobei, was die Clusteranzahl angeht, ExtendedMajorClust und StrongMajorClust dem MajorClust überlegen sind.

Von den Variationen des MCProb-Algorithmus zeigen nur die beiden balancierten Modifikationen recht gute Ergebnisse, die die Ergebnisse der nicht-probabilistischen Algorithmen aber nicht übertreffen.

Die Algorithmen MCProb_CC, ExponentialMCProb_CC, DynamicMCProb und Dy-

Algorithmus	F-Measure	RandIndex	Clusteranzahl
k-Means	0.844	0.909	0%
MajorClust	0.855	0.889	+66%
ExtendedMajorClust	0.868	0.900	+32%
StrongMajorClust_CC	0.857	0.890	+35%
MCProb_CC	0.552	0.565	+79%
ExponentialMCProb_CC	0.536	0.563	+88%
DynamicMCProb	0.649	0.630	+11%
DynamicExponentialMCProb	0.633	0.607	0%
BalancedMCProb_CC	0.787	0.852	+70%
BalancedMCProb_CCd	0.838	0.873	+13%

Tabelle 5.2.: Durchschnittswerte

namicExponentialMCProb haben sich als ungeeignet für die gestellte Aufgabe erwiesen. Der Grund dafür ist auch, dass diese Algorithmen eine hohe Anzahl an Ausreiser bei der Clusteranzahl aufweisen. Beispielsweise kann bei sechs Dokumentenklassen und 2000 Dokumenten in der Testkollektion der MCProb_CC-Algorithmus bei mehreren Ausführungen sowohl einen als auch 120 Cluster liefern.

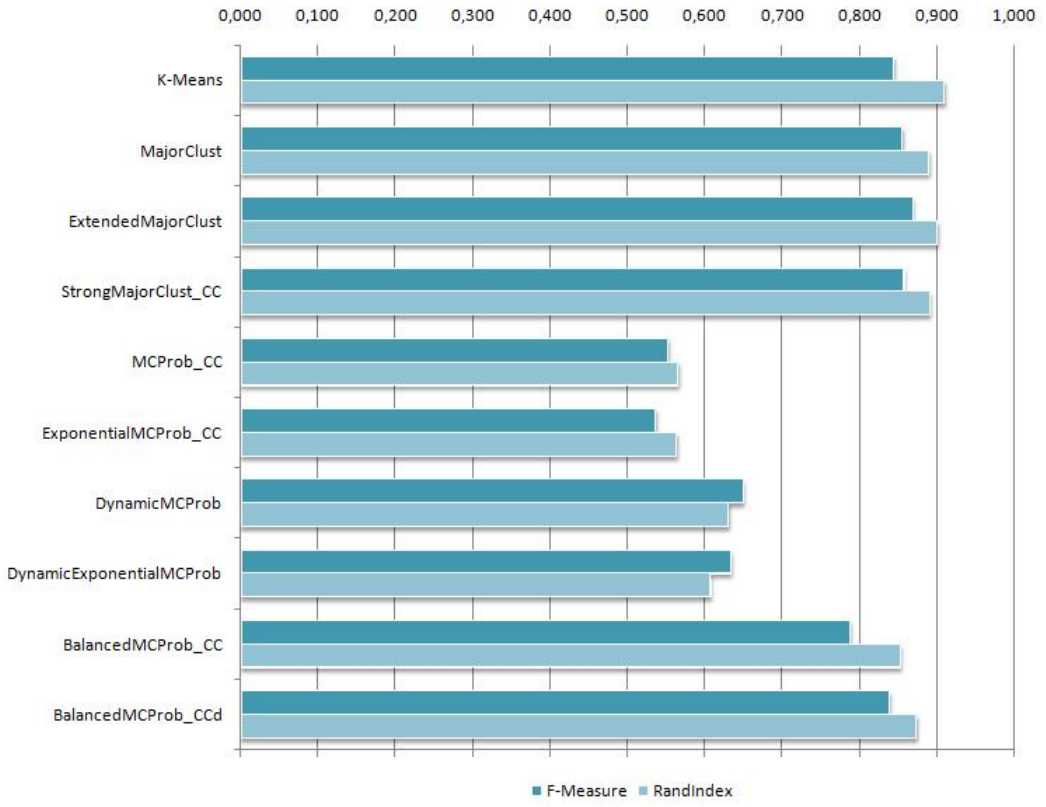

Abbildung 5.1.: F-Measure & RandIndex

6. Zusammenfassung und Ausblick

Die vorliegende Arbeit beschäftigt sich primär mit der Evaluierung der Clustering-Algorithmen aus der MajorClust-Familie beim Einsatz für das Dokumenten-Clustering.

Es wurde anhand einiger Beispiele gezeigt, wie Dokumenten-Clustering in verschiedene Prozesse integriert werden kann, um die Komplexität von Informationsquellen zu reduzieren. Darüber hinaus wurden verschiedene Modifikationen des MajorClust-Algorithmus vorgestellt und, Aufgrund der Abwesenheit einer klarer Definition eines für den MCProb-Algorithmus geeigneten Abbruchkriteriums, weitere Modifikationen des eben genannten entwickelt.

Die Experimente haben gezeigt, dass der von Tim Gollub entwickelte ExtendedMajorClust und der von Mikhail Alexandrov entwickelte StrongMajorClust den MajorClust-Algorithmus übertreffen. Die guten Ergebnisse für BalancedMCProb mit den beiden Abbruchkriterien und die schlechten Ergebnisse für alle anderen probabilistischen Modifikationen zeigen, dass das Verhalten der Algorithmen stark vom Abbruchkriterium abhängt. Aus diesem Grund wäre es gut möglich zukünftig neuere, raffiniertere Abbruchkriterien zu entwickeln, die die Funktionsweise des MCProb-Algorithmus verbessern würden.

Literaturverzeichnis

[AKG04] Andronov, Kopytov, and Gringlaz. *Теория вероятностей и математическая статистика.* Издательский дом "Питер", 2004.

[Ale] Mikhail Alexandrov. Managing degree of grouping by strong majorclust.

[Bac94] Johann Bacher. *Clusteranalyse. Anwendungsorientierte Einführung.* Oldenbourg R. Verlag GmbH, 1994.

[BEPW08] Klaus Backhaus, Bernd Erichson, Wulff Plinke, and Rolf Weiber. *Multivariate Analysemethoden: Eine anwendungsorientierte Einführung.* Springer, Berlin, 12., vollständig überarbeitete auflage. edition, 9 2008.

[Bus05] Michael Busch. *Analyse dichtebasierter Clusteralgorithmen am Beispiel von DBSCAN und MajorClust.* Studienarbeit, Universität Paderborn, Fakultät für Elektrotechnik, Informatik und Mathematik, March 2005.

[CKPT92] Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. Scatter/gather: a cluster-based approach to browsing large document collections. In *SIGIR '92: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 318–329, New York, NY, USA, 1992. ACM.

[Gol08] Tim Gollub. Verfahren zur Modellbildung für das Dokumenten-Clustering. Diplomarbeit, Bauhaus-Universität Weimar, Fakultät Medien, Mediensysteme, April 2008.

[HP96] Marti A. Hearst and Jan O. Pedersen. Reexamining the cluster hypothesis: scatter/gather on retrieval results. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 76–84, New York, NY, USA, 1996. ACM.

[LYRL04] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361–397, 2004.

[Mey07] Sven Meyer zu Eissen. *On Information Need and Categorizing Search.* Dissertation, University of Paderborn, Feb 2007.

[MSP05] Sven Meyer zu Eißen, Benno Stein, and Martin Potthast. The Suffix Tree Document Model Revisited. In Klaus Tochtermann and Hermann Maurer, editors, *Proceedings of the 5th International Conference on Knowledge Management (I-KNOW 05), Graz, Austria*, Journal of Universal Computer Science, pages 596–603. Know-Center, July 2005.

[NS09] Oliver Niggemann and Benno Stein. A probabilistic majorclust variant. 2 2009.

[Rij79] C. J. Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, 2nd edition, 3 1979.

[SB05] Benno Stein and Michael Busch. Density-based Cluster Algorithms in Low-dimensional and High-dimensional Applications. In Benno Stein and Sven Meyer zu Eißen, editors, *Second International Workshop on Text-Based Information Retrieval (TIR 05), Koblenz, Germany*, Fachberichte Informatik, pages 45–56. University of Koblenz-Landau, Germany, September 2005.

[SM02] Benno Stein and Sven Meyer zu Eißen. Document Categorization with MajorClust. In Amit Basu and Soumitra Dutta, editors, *Proceedings of the 12th Workshop on Information Technology and Systems (WITS 02), Barcelona, Spain*, pages 91–96. Technical University of Barcelona, December 2002.

[SM03] Benno Stein and Sven Meyer zu Eißen. Automatic Document Categorization: Interpreting the Perfomance of Clustering Algorithms. In Andreas Günter, Rudolf Kruse, and Bernd Neumann, editors, *KI 2003: Advances in Artificial Intelligence*, volume 2821 LNAI of *Lecture Notes in Artificial Intelligence*, pages 254–266, Berlin Heidelberg New York, September 2003. Springer.

[SM04] Benno Stein and Sven Meyer zu Eißen. Automatische Kategorisierung für Web-basierte Suche: Einführung, Techniken und Projekte. *KI – Künstliche Intelligenz: Special Issue on Adaptive Multimedia Retrieval*, 4:11–17, November 2004.

[SN99] Benno Stein and Oliver Niggemann. On the Nature of Structure and its Identification. In Peter Widmayer, Gabriele Neyer, and Stefan Eidenbenz, editors, *Graph-Theoretic Concepts in Computer Science*, volume 1665 LNCS of *Lecture Notes in Computer Science*, pages 122–134, Berlin Heidelberg New York, June 1999. Springer.

[Stea] Benno Stein. Information retrieval: Retrieval-modelle. http://www.uni-weimar.de/medien/webis/teaching/lecturenotes/information-retrieval/unit-de-retrieval-models.pdf. Letzter Zugriff: 27.12.2008.

[Steb] Benno Stein. Maschinelles lernen und data mining: Dichtebasierte verfahren. http://www.uni-weimar.de/medien/webis/teaching/lecturenotes/

machine-learning/unit-de-cluster-analysis-density.pdf. Letzter Zugriff: 21.03.2009.

[Stec] Benno Stein. Maschinelles lernen und data mining: Einführung in die cluster-analyse. http://www.uni-weimar.de/medien/webis/teaching/lecturenotes/machine-learning/unit-de-cluster-analysis-intro.pdf. Letzter Zugriff: 02.12.2008.

[Ste77] Detlef Steinhausen. *Clusteranalyse: Einf. in Methoden u. Verfahren d. automatischen Klassifikation : mit zahlr. Algorithmen, FORTRAN-Programmen, Anwendungsbeispielen u.e. ... (De Gruyter Lehrbuch) (German Edition)*. de Gruyter, 1. aufl edition, 1977.

[Sto06] Wolfgang G. Stock. *Information Retrieval: Informationen suchen und finden.* Oldenbourg, 10 2006.

[TSK06] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining.* Pearson, 2006.

[ZE98] Oren Zamir and Oren Etzioni. Web document clustering: a feasibility demonstration. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 46–54, New York, NY, USA, 1998. ACM.

A. Ergebnisse der Experimente mit Harmonic Expected Similarity

A.1. mit PlainIndexer

Algorithmus	F-Measure	RandIndex	Clusteranzahl
k-Means	0.84427	0.90912	0%
MajorClust	0.85513	0.88877	+66%
ExtendedMajorClust	0.86809	0.90049	+32&
StrongMajorClust_CC	0.85663	0.88964	+35%
MCProb_CC	0.55181	0.56533	+79%
ExponentialMCProb_CC	0.53647	0.56266	+88%
DynamicMCProb	0.64895	0.62998	+11%
DynamicExponentialMCProb	0.63347	0.60661	0%
BalancedMCProb_CC	0.78746	0.85182	+70%
BalancedMCProb_CCd	0.83751	0.87300	+13%

Tabelle A.1.: Durchschnittswerte

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.9855	0.97134	2	0%
MajorClust	0.88202	0.84777	6.4	+220%
ExtendedMajorClust	0.91854	0.89461	4.2	+110%
StrongMajorClust_CC	0.94622	0.90421	5.2	+160%
MCProb_CC	0.59383	0.66836	12.7	+535%
ExponentialMCProb_CC	0.61223	0.68774	11.4	+470%
DynamicMCProb	0.89123	0.84953	5.1	+155%
DynamicExponentialMCProb	0.8814	0.82823	4.8	+140%
BalancedMCProb_CC	0.78952	0.75683	8.4	+320%
BalancedMCProb_CCd	0.94415	0.91026	3.4	+70%

Tabelle A.2.: RC2E200

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.99285	0.9859	2	0%
MajorClust	0.95285	0.93541	4.4	+120%
ExtendedMajorClust	0.97962	0.96079	3.1	+55%
StrongMajorClust_CC	0.97428	0.95095	2.9	+45%
MCProb_CC	0.68218	0.54014	3.0	+50%
ExponentialMCProb_CC	0.6812	0.52456	2.1	+5%
DynamicMCProb	0.78149	0.67631	3.6	+80%
DynamicExponentialMCProb	0.66285	0.50028	2.6	+30%
BalancedMCProb_CC	0.94958	0.92968	3.9	+95%
BalancedMCProb_CCd	0.96702	0.94777	3.6	+80%

Tabelle A.3.: RC2E2000

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.57334	0.79854	6	0%
MajorClust	0.73647	0.88112	13.0	+117%
ExtendedMajorClust	0.77978	0.89737	10.1	+68%
StrongMajorClust_CC	0.75124	0.86531	9.5	+58%
MCProb_CC	0.58023	0.81136	10.1	+68%
ExponentialMCProb_CC	0.55243	0.82985	12.7	+112%
DynamicMCProb	0.75161	0.87108	7.8	+30%
DynamicExponentialMCProb	0.7438	0.86773	7.6	+27%
BalancedMCProb_CC	0.49208	0.79217	11.2	+87%
BalancedMCProb_CCd	0.68775	0.81157	6.0	0%

Tabelle A.4.: RC6E120

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.90832	0.94432	6	0%
MajorClust	0.83166	0.89374	6.2	+3%
ExtendedMajorClust	0.85069	0.90924	5.8	-3%
StrongMajorClust_CC	0.8055	0.87917	5.7	-5%
MCProb_CC	0.55721	0.66942	9.4	+57%
ExponentialMCProb_CC	0.52621	0.71896	15.2	+153%
DynamicMCProb	0.6751	0.75574	4.6	-23%
DynamicExponentialMCProb	0.66286	0.73104	4.2	-30%
BalancedMCProb_CC	0.7836	0.86659	5.7	-5%
BalancedMCProb_CCd	0.78501	0.85673	4.9	-18%

Tabelle A.5.: RC6E600

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.85849	0.92037	6.0	0%
MajorClust	0.8747	0.91998	6.0	0%
ExtendedMajorClust	0.85238	0.90433	5.4	-10%
StrongMajorClust_CC	0.85329	0.90839	5.9	-2%
MCProb_CC	0.48308	0.48784	2.5	-58%
ExponentialMCProb_CC	0.43707	0.41594	2.3	-62%
DynamicMCProb	0.43621	0.41372	2.1	-65%
DynamicExponentialMCProb	0.46949	0.46587	2.3	-62%
BalancedMCProb_CC	0.82239	0.89048	5.3	-12%
BalancedMCProb_CCd	0.80595	0.8705	5.1	-15%

Tabelle A.6.: RC6E3000

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.8127	0.89218	6	0%
MajorClust	0.85443	0.88152	6.1	+2%
ExtendedMajorClust	0.85686	0.88545	6.0	0%
StrongMajorClust_CC	0.8434	0.87727	6.0	0%
MCProb_CC	0.48772	0.42582	3.0	-50%
ExponentialMCProb_CC	0.48633	0.42946	3.4	-43%
DynamicMCProb	0.48892	0.42805	2.7	-55%
DynamicExponentialMCProb	0.49683	0.44291	2.7	-55%
BalancedMCProb_CC	0.8487	0.87924	5.8	-3%
BalancedMCProb_CCd	0.84077	0.86337	5.5	-8%

Tabelle A.7.: RC6U1550

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.77868	0.8513	6.0	0%
MajorClust	0.8538	0.86185	6.0	0%
ExtendedMajorClust	0.83877	0.85165	5.8	-3%
StrongMajorClust_CC	0.82248	0.84216	5.4	-10%
MCProb_CC	0.47844	0.3544	2.9	-52%
ExponentialMCProb_CC	0.4598	0.33209	4.9	-18%
DynamicMCProb	0.51808	0.41542	3.1	-48%
DynamicExponentialMCProb	0.51709	0.41023	3.0	-50%
BalancedMCProb_CC	0.82636	0.84772	6.4	+7%
BalancedMCProb_CCd	0.83195	0.85078	4.8	-20%

Tabelle A.8.: RC6U1650

A.2. mit TfIdfIndexer

Algorithmus	F-Measure	RandIndex	Abweichung
k-Means	0.74453	0.81884	0%
MajorClust	0.65898	0.69124	+148%
ExtendedMajorClust	0.64742	0.66051	+87%
StrongMajorClust_CC	0.66777	0.67374	+66%
MCProb_CC	0.50718	0.57364	+165%
ExponentialMCProb_CC	0.42367	0.59081	+754%
DynamicMCProb	0.60369	0.58636	+27%
DynamicExponentialMCProb	0.57522	0.54127	+1%
BalancedMCProb_CC	0.59277	0.63636	+92%
BalancedMCProb_CCd	0.61926	0.61440	+9%

Tabelle A.9.: Durchschnitt

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.72214	0.65526	2	0%
MajorClust	0.6587	0.69796	16	+700%
ExtendedMajorClust	0.69765	0.69962	12.5	+525%
StrongMajorClust_CC	0.79156	0.75704	9.6	+380%
MCProb_CC	0.50433	0.60255	16.3	+715%
ExponentialMCProb_CC	0.31016	0.52937	18.3	+815%
DynamicMCProb	0.78744	0.73301	6.2	+210%
DynamicExponentialMCProb	0.74024	0.64055	5	+150%
BalancedMCProb_CC	0.58066	0.62096	9.4	+370%
BalancedMCProb_CCd	0.68422	0.57062	4.1	+105%

Tabelle A.10.: RC2E200

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.88592	0.8353	2	0%
MajorClust	0.88355	0.8766	6.7	+235%
ExtendedMajorClust	0.95057	0.92897	4.8	+140%
StrongMajorClust_CC	0.93176	0.89551	4.9	+145%
MCProb_CC	0.65888	0.50768	4.1	+105%
ExponentialMCProb_CC	0.51288	0.66254	69.4	+3370%
DynamicMCProb	0.6755	0.53621	4.9	+145%
DynamicExponentialMCProb	0.65734	0.50616	3.5	+75%
BalancedMCProb_CC	0.8299	0.78184	6.1	+205%
BalancedMCProb_CCd	0.85327	0.82703	5.1	+155%

Tabelle A.11.: RC2E2000

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.51134	0.76601	6	0%
MajorClust	0.66512	0.87979	18.5	208%
ExtendedMajorClust	0.68822	0.88258	14.2	+137%
StrongMajorClust_CC	0.68403	0.85905	13.2	+120%
MCProb_CC	0.47151	0.8066	13.4	+123%
ExponentialMCProb_CC	0.45028	0.80557	12.8	+113%
DynamicMCProb	0.69053	0.84821	7.2	+20%
DynamicExponentialMCProb	0.65316	0.8121	6.8	+13%
BalancedMCProb_CC	0.61279	0.84688	10.4	+73%
BalancedMCProb_CCd	0.62648	0.80535	6.7	+12%

Tabelle A.12.: RC6E120

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.78729	0.89118	6	0%
MajorClust	0.7251	0.80926	6.1	+2%
ExtendedMajorClust	0.59156	0.64613	3.6	-40%
StrongMajorClust_CC	0.6686	0.74328	4.4	-27%
MCProb_CC	0.48879	0.8443	26.8	+347%
ExponentialMCProb_CC	0.39991	0.83977	33.3	+455%
DynamicMCProb	0.5912	0.65706	4.3	-28%
DynamicExponentialMCProb	0.56255	0.61764	3.9	-35%
BalancedMCProb_CC	0.56775	0.76143	12.9	+115%
BalancedMCProb_CCd	0.59621	0.6506	3.2	-47%

Tabelle A.13.: RC6E600

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.86147	0.92314	6	0%
MajorClust	0.72378	0.80607	5.7	-5%
ExtendedMajorClust	0.66567	0.73585	4.3	-28%
StrongMajorClust_CC	0.66241	0.73622	4.0	-33%
MCProb_CC	0.49032	0.50581	2.6	-57%
ExponentialMCProb_CC	0.40333	0.37954	4.2	-30%
DynamicMCProb	0.53564	0.57531	3.3	-45%
DynamicExponentialMCProb	0.49326	0.50742	2.4	-60%
BalancedMCProb_CC	0.61682	0.6821	4.3	-28%
BalancedMCProb_CCd	0.60803	0.66842	3.7	-38%

Tabelle A.14.: RC6E3000

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.73113	0.84472	6	0%
MajorClust	0.49893	0.44094	3.2	-47%
ExtendedMajorClust	0.48522	0.41845	2.4	-60%
StrongMajorClust_CC	0.4826	0.41319	2.5	-58%
MCProb_CC	0.48288	0.42646	3.7	-38%
ExponentialMCProb_CC	0.45277	0.56081	31.8	+430%
DynamicMCProb	0.49108	0.43272	2.5	-58%
DynamicExponentialMCProb	0.47626	0.40522	2.2	-63%
BalancedMCProb_CC	0.48392	0.42603	3.3	-45%
BalancedMCProb_CCd	0.48322	0.41377	2.2	-63%

Tabelle A.15.: RC6U1550

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.71242	0.81625	6.0	0%
MajorClust	0.45768	0.32805	2.5	-58%
ExtendedMajorClust	0.45307	0.31197	1.9	-68%
StrongMajorClust_CC	0.45346	0.31191	2.2	-63%
MCProb_CC	0.45354	0.32208	3.4	-43%
ExponentialMCProb_CC	0.43636	0.35805	13.9	+127%
DynamicMCProb	0.45441	0.32197	2.5	-58%
DynamicExponentialMCProb	0.44373	0.2998	1.7	-72%
BalancedMCProb_CC	0.45755	0.33529	3.4	-43%
BalancedMCProb_CCd	0.4834	0.36502	2.2	-63%

Tabelle A.16.: RC6U1650

B. Ergebnisse der Experimente mit Average Expected Similarity

B.1. mit PlainIndexer

Algorithmus	F-Measure	RandIndex	rel.Abweichung
k-Means	0.84427	0.90912	0%
MajorClust	0.68932	0.68865	-36%
ExtendedMajorClust	0.67840	0.67321	-39%
StrongMajorClust_CC	0.64548	0.64108	-37%
MCProb_CC	0.48498	0.41175	-42%
ExponentialMCProb_CC	0.45095	0.41540	+8%
DynamicMCProb	0.49340	0.39467	-63%
DynamicExponentialMCProb	0.48879	0.38906	-64%
BalancedMCProb_CC	0.64544	0.64777	-38%
BalancedMCProb_CCd	0.63770	0.61477	-45%

Tabelle B.1.: Durchschnitt

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.9855	0.97134	2.0	0%
MajorClust	0.97646	0.95407	2.2	+10%
ExtendedMajorClust	0.97698	0.95497	2.0	0%
StrongMajorClust_CC	0.96895	0.93982	2.2	+10%
MCProb_CC	0.65652	0.50921	2.8	+40%
ExponentialMCProb_CC	0.61478	0.57034	8.2	+310%
DynamicMCProb	0.69538	0.53911	1.2	-40%
DynamicExponentialMCProb	0.6985	0.54477	1.1	-45%
BalancedMCProb_CC	0.96866	0.94946	1.9	-5%
BalancedMCProb_CCd	0.97798	0.95694	2.0	0%

Tabelle B.2.: RC2E200

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.99285	0.9858	2.0	0%
MajorClust	0.98135	0.9634	2.0	0%
ExtendedMajorClust	0.98055	0.96186	2.0	0%
StrongMajorClust_CC	0.98065	0.96206	2.0	0%
MCProb_CC	0.66667	0.49975	1.0	-50%
ExponentialMCProb_CC	0.66667	0.49975	1.0	-50%
DynamicMCProb	0.6666	0.49975	1.1	-45%
DynamicExponentialMCProb	0.66667	0.49975	1.0	-50%
BalancedMCProb_CC	0.98045	0.96168	2.0	0%
BalancedMCProb_CCd	0.98135	0.96339	2.0	0%

Tabelle B.3.: RC2E2000

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.57334	0.79854	6.0	0%
MajorClust	0.60776	0.64391	3.9	-35%
ExtendedMajorClust	0.56031	0.58735	3.2	-47%
StrongMajorClust_CC	0.49437	0.51324	3.6	-40%
MCProb_CC	0.41093	0.55999	6.3	+5%
ExponentialMCProb_CC	0.36966	0.76859	12.4	+107%
DynamicMCProb	0.47071	0.45608	2.6	-57%
DynamicExponentialMCProb	0.44435	0.42483	2.6	-57%
BalancedMCProb_CC	0.42277	0.50634	4.5	-25%
BalancedMCProb_CCd	0.45101	0.43448	2.3	-62%

Tabelle B.4.: RC6E120

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.90832	0.94432	6.0	0%
MajorClust	0.55741	0.57789	3.0	-50%
ExtendedMajorClust	0.59999	0.63833	3.3	-45%
StrongMajorClust_CC	0.44471	0.44914	3.0	-50%
MCProb_CC	0.44183	0.41527	2.3	-62%
ExponentialMCProb_CC	0.37628	0.31534	1.8	-70%
DynamicMCProb	0.38758	0.33026	1.8	-70%
DynamicExponentialMCProb	0.41301	0.37158	2.0	-67%
BalancedMCProb_CC	0.56824	0.61693	3.2	-47%
BalancedMCProb_CCd	0.50066	0.49956	2.6	-57%

Tabelle B.5.: RC6E600

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.85849	0.92037	6.0	0%
MajorClust	0.57233	0.59736	3.1	-48%
ExtendedMajorClust	0.51577	0.52351	2.7	-55%
StrongMajorClust_CC	0.49816	0.51806	2.8	-53%
MCProb_CC	0.40186	0.35462	1.9	-68%
ExponentialMCProb_CC	0.32442	0.22907	1.3	-78%
DynamicMCProb	0.38744	0.3452	1.7	-72%
DynamicExponentialMCProb	0.34787	0.28763	1.4	-77%
BalancedMCProb_CC	0.51841	0.53234	2.8	-53%
BalancedMCProb_CCd	0.50151	0.50374	2.6	-57%

Tabelle B.6.: RC6E3000

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.8127	0.89218	6.0	0%
MajorClust	0.5701	0.56329	2.7	-55%
ExtendedMajorClust	0.58628	0.59314	2.8	-53%
StrongMajorClust_CC	0.57337	0.58516	2.8	-53%
MCProb_CC	0.38882	0.26549	1.3	-78%
ExponentialMCProb_CC	0.37654	0.24643	1.2	-80%
DynamicMCProb	0.42138	0.31448	1.7	-72%
DynamicExponentialMCProb	0.42286	0.31711	1.8	-70%
BalancedMCProb_CC	0.52196	0.48764	2.4	-60%
BalancedMCProb_CCd	0.51403	0.46493	2.2	-63%

Tabelle B.7.: RC6U1550

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.77868	0.8513	6.0	0%
MajorClust	0.55984	0.52061	1.6	-73%
ExtendedMajorClust	0.52891	0.45333	1.5	-75%
StrongMajorClust_CC	0.55812	0.5201	1.6	-73%
MCProb_CC	0.4282	0.27791	1.1	-82%
ExponentialMCProb_CC	0.4283	0.27826	1.2	-80%
DynamicMCProb	0.42818	0.2778	1.0	-83%
DynamicExponentialMCProb	0.42828	0.27803	1.1	-82%
BalancedMCProb_CC	0.53756	0.48001	1.6	-73%
BalancedMCProb_CCd	0.53739	0.48033	1.5	-75%

Tabelle B.8.: RC6U1650

B.2. mit TfIdfIndexer

Algorithmus	F-Measure	RandIndex	rel.Abweichung
k-Means	0.74453	0.81884	0%
MajorClust	0.66157	0.64255	-16%
ExtendedMajorClust	0.68707	0.67838	-33%
StrongMajorClust_CC	0.65625	0.64444	-34%
MCProb_CC	0.42231	0.44488	+96%
ExponentialMCProb_CC	0.30233	0.56691	+785%
DynamicMCProb	0.48860	0.39532	-55%
DynamicExponentialMCProb	0.46910	0.35771	-63%
BalancedMCProb_CC	0.55274	0.56705	+29%
BalancedMCProb_CCd	0.56712	0.51506	-50%

Tabelle B.9.: Durchschnitt

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.72214	0.65526	2.0	0%
MajorClust	0.93569	0.89232	3.4	+70%
ExtendedMajorClust	0.90883	0.85369	2.3	+15%
StrongMajorClust_CC	0.874	0.80816	2.2	+10%
MCProb_CC	0.46589	0.64243	17.0	+750%
ExponentialMCProb_CC	0.25377	0.53032	22.4	+1020%
DynamicMCProb	0.66519	0.4979	1.3	-35%
DynamicExponentialMCProb	0.66599	0.49759	1.2	-40%
BalancedMCProb_CC	0.61006	0.59849	9.1	+355%
BalancedMCProb_CCd	0.69605	0.54099	1.3	-35%

Tabelle B.10.: RC2E200

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.88592	0.8353	2.0	0%
MajorClust	0.82357	0.73403	2.8	+40%
ExtendedMajorClust	0.9264	0.88625	1.9	-5%
StrongMajorClust_CC	0.88309	0.817	1.9	-5%
MCProb_CC	0.6648	0.49997	1.8	-10%
ExponentialMCProb_CC	0.6659	0.49984	1.7	-15%
DynamicMCProb	0.66475	0.49998	2.0	0%
DynamicExponentialMCProb	0.66602	0.49983	1.3	-35%
BalancedMCProb_CC	0.75952	0.63909	2.2	+10%
BalancedMCProb_CCd	0.66483	0.49995	1.7	-15%

Tabelle B.11.: RC2E2000

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.51134	0.76601	6.0	0%
MajorClust	0.73677	0.83592	6.7	+12%
ExtendedMajorClust	0.68096	0.75359	4.8	-20%
StrongMajorClust_CC	0.67711	0.76734	5.4	-10%
MCProb_CC	0.29849	0.78926	14.5	+142%
ExponentialMCProb_CC	0.33322	0.79803	14.1	+135%
DynamicMCProb	0.44068	0.45881	2.5	-58%
DynamicExponentialMCProb	0.4416	0.43532	2.5	-58%
BalancedMCProb_CC	0.42607	0.75685	10.4	+73%
BalancedMCProb_CCd	0.53059	0.60661	3.0	-50%

Tabelle B.12.: RC6E120

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.78729	0.89118	6.0	0%
MajorClust	0.5597	0.60084	3.1	-48%
ExtendedMajorClust	0.59288	0.63294	3.2	-47%
StrongMajorClust_CC	0.59323	0.65688	3.4	-43%
MCProb_CC	0.3725	0.37528	7.6	+27%
ExponentialMCProb_CC	0.18413	0.8298	60.2	+903%
DynamicMCProb	0.45427	0.44286	2.3	-62%
DynamicExponentialMCProb	0.41924	0.37965	2.1	-65%
BalancedMCProb_CC	0.57143	0.6371	3.8	-37%
BalancedMCProb_CCd	0.54858	0.57872	2.9	-52%

Tabelle B.13.: RC6E600

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.86147	0.92314	6.0	0%
MajorClust	0.59668	0.63617	3.3	-45%
ExtendedMajorClust	0.65621	0.71653	3.6	-40%
StrongMajorClust_CC	0.53089	0.55944	2.8	-53%
MCProb_CC	0.34316	0.26365	1.4	-77%
ExponentialMCProb_CC	0.28571	0.16639	1.0	-83%
DynamicMCProb	0.33223	0.24814	1.3	-78%
DynamicExponentialMCProb	0.28571	0.16639	1.0	-83%
BalancedMCProb_CC	0.56937	0.6127	3.0	-50%
BalancedMCProb_CCd	0.55407	0.58892	2.9	-52%

Tabelle B.14.: RC6E3000

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.73113	0.84472	6.0	0%
MajorClust	0.50209	0.43892	2.3	-62%
ExtendedMajorClust	0.5427	0.50372	2.4	-60%
StrongMajorClust_CC	0.51687	0.46678	2.3	-62%
MCProb_CC	0.38898	0.26574	1.2	-80%
ExponentialMCProb_CC	0.22509	0.55624	116.7	1845%
DynamicMCProb	0.43478	0.34153	1.6	-73%
DynamicExponentialMCProb	0.37683	0.24719	1.3	-78%
BalancedMCProb_CC	0.48064	0.40666	2.1	-65%
BalancedMCProb_CCd	0.52295	0.47077	2.2	-63%

Tabelle B.15.: RC6U1550

Algorithmus	F-Measure	RandIndex	Cluster	Abweichung
k-Means	0.71242	0.81625	6.0	0%
MajorClust	0.47649	0.35962	1.3	-78%
ExtendedMajorClust	0.50154	0.40197	1.4	-77%
StrongMajorClust_CC	0.51859	0.43545	1.5	-75%
MCProb_CC	0.42818	0.2778	1.0	-83%
ExponentialMCProb_CC	0.1685	0.58774	107.2	1687%
DynamicMCProb	0.42828	0.27803	1.1	-82%
DynamicExponentialMCProb	0.42828	0.27803	1.1	-82%
BalancedMCProb_CC	0.45207	0.31844	1.1	-82%
BalancedMCProb_CCd	0.45274	0.31943	1.1	-82%

Tabelle B.16.: RC6U1650