BAUHAUS-UNIVERSITÄT WEIMAR
FAKULTÄT MEDIEN
WEB TECHNOLOGY AND INFORMATION SYSTEMS

Bachelorarbeit

Analyse von Verfahren zur Effizienzsteigerung von multidimensionaler Skalierung

Anita Schilling

Februar 2007

Betreuer:
Prof. Dr. Benno Stein
Dipl.-Inf. Martin Potthast

Erklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig angefertigt, anderweitig nicht für Prüfungszwecke vorgelegt, alle zitierten Quellen und benutzten Hilfsmittel angegeben sowie wörtliche und sinngemäße Zitate gekennzeichnet habe.

Weimar, den 08.02.2007 Anita Schilling

Kurzreferat

In dieser Arbeit werden Verfahren der multidimensionalen Skalierung analysiert. Das Ziel der multidimensionalen Skalierung ist die Einbettung hochdimensionaler Daten in eine niedrige Dimension um sie visualisieren zu können. Die dazu betrachteten Verfahren basieren auf kräftegerichteter Positionierung durch Distanzunterschiede. Der bekannteste Vertreter dieser Verfahren ist das Spring-Modell. Ein neues Verfahren dieser Art, das Fuzzy-Fingerprinting für eine approximative Nächste-Nachbar-Suche in Linearzeit einsetzt, wird vorgestellt und analysiert. Außerdem werden die Verfahren zur multidimensionalen Skalierung in eine Taxonomie eingeordnet. Eine Evaluation der betrachteten Verfahren auf Dokumentenkollektionen mit bis zu 100 000 Dokumenten und mehreren tausend Dimensionen wird durchgeführt. Es zeigt sich, dass das neue Verfahren eine Verbesserung darstellt und als das effizienteste Verfahren zur multidimensionalen Skalierung im Hinblick auf die Laufzeit und die Qualität der generierten Punktkonfiguration, im Vergleich zu den betrachteten Verfahren, bezeichnet werden kann.

Inhalt

1 Einleitung

In vielen Bereichen der Wissenschaft und des täglichen Lebens ist eine Visualisierung hochdimensionaler Daten, welche keine unmittelbare Entsprechung im zwei- oder dreidimensionalen Raum haben, wünschenswert. Methoden, die die dafür erforderliche Dimensionsreduktion der Daten erzielen, werden unter dem Begriff *multidimensionale Skalierung* zusammengefasst. Die Repräsentation von multivariaten Daten in einem niedrigdimensionalen Raum eröffnet die Möglichkeit, durch gewöhnliche 2D-Plots oder 3D-Visualisierungen Einblick in die Daten und die Beziehungen der Datenobjekte untereinander zu erhalten. Damit können Muster ersichtlich werden, die in einer Tabellenrepräsentation derselben Daten nur äußerst schwer zu erkennen wären. Im textbasierten Information-Retrieval stehen meist große Dokumentenkollektionen zur Verfügung, wobei die Ähnlichkeiten der Dokumente untereinander aus einer Stichwortliste der Dokumententhemen kaum offensichtlich werden. Dieselbe Dokumentenkollektionen als 2D-Plot visualisiert, zeigt schnell auf, welche Dokumente sich besonders ähnlich sind und daher als dichte Punktwolken dargestellt werden. Ein Bibliothekskatalog mit dieser Fähigkeit könnte dem Benutzer beispielsweise einen direkten und intuitiven Zugang zum Datenbestand verschaffen.

In dieser Arbeit wird ein Ansatz der multidimensionalen Skalierung näher untersucht, dem eine kräftegerichtete Positionierung auf Basis von Distanzunterschieden zwischen den Datenobjekten zugrunde liegt. Problematisch an dieser Art der multidimensionalen Skalierung ist die hohe Komplexität und die daraus in der Praxis resultierenden unvertretbar langen Rechenzeiten des Basisverfahrens, welches jedoch sehr gute Ergebnisse liefert. Bestehende Verfahren auf dieser Grundlage werden deshalb analysiert. Des Weiteren wird eine Weiterentwicklung dieses Verfahrens unter Verwendung des sogenannten Similarity-

Hashing vorgestellt. Außerdem wird eine Taxonomie der Verfahren zur multidimensionalen Skalierung aufgestellt. Ziel dieser Arbeit ist es, die ausgewählten Verfahren hinsichtlich ihrer Laufzeiten und generierten Punktkonfigurationen zu evaluieren, um das effizienteste Verfahren dieser Art der multidimensionalen Skalierung zu identifizieren. Das effizienteste Verfahren sollte im Vergleich mit anderen Verfahren, eine kürzere Laufzeit unter Beibehaltung der Qualität der berechneten Punktkonfiguration aufweisen.

Die Arbeit ist in vier Teile gegliedert. Zunächst wird die multidimensionale Skalierung mit ihren Eigenschaften erläutert. In Kapitel 3 werden die ausgewählten Verfahren beschrieben und analysiert. Außerdem wird das weiterentwickelte Verfahren intensiv betrachtet. Anschließend werden im 4. Kapitel die Testbedingungen der Evaluation der Verfahren spezifiziert, die Ergebnisse vorgestellt und bewertet. Am Ende folgt die Zusammenfassung der Arbeit und ein Ausblick auf mögliche alternative Ansätze, die aus den Ergebnissen resultieren. Auf der beiliegenden CD befindet sich der Quellcode der in Java implementierten Verfahren und die Ergebnisse der damit durchgeführten Evaluation sowie eine elektronische Version dieser Bachelorarbeit im PDF-Format.

2 Multidimensionale Skalierung

In diesem Kapitel werden die theoretischen Grundlagen und Voraussetzungen der multidimensionalen Skalierung erläutert. Des Weiteren wird ein Maß zur Bewertung der Ergebnisse dieses Ansatzes eingeführt.

2.1 Grundlagen

Multidimensionale Skalierung definieren Borg und Groenen (1997) als eine Methode, die Distanzen δ zwischen Objektpaaren (a,b) als Distanzen Δ zwischen Punkten (p_a, p_b) in einem niedrigdimensionalen Raum zu repräsentieren. Das heißt, die Formel 2.1

$$\delta(a,b) - \Delta(p_a, p_b) \to 0 \tag{2.1}$$

soll für jedes Objektpaar (a,b) aus der Datenmenge D gelten. Der Zielraum ist üblicherweise der zwei- oder dreidimensionale euklidische Raum, so dass die berechnete Punktkonfiguration als 2D-Plot oder 3D-Visualisierung dargestellt werden kann. Diese Methode kann für alle Arten von Daten genutzt werden, solange ein Ähnlichkeits- bzw. Distanzmaß zwischen den Objekten a und b der Datenmenge D definierbar ist.

In der weiteren Arbeit wird als Zielraum immer der zweidimensionale euklidische Raum angenommen. Darüber hinaus wird mit dem Objekt x ein Datenvektor $\mathbf{x}$ assoziiert und ein Punkt p_x, der den Datenvektor im niedrigdimensionalen Raum repräsentiert. Hierbei ist ein Objekt d immer gleichzusetzen mit einem Dokument d, dessen Wortvektor $\mathbf{d}$ bekannt ist und dem ein Punkt p_d zugeordnet werden kann. Außerdem muss der Fuzzy-Fingerprint von d berechenbar sein. Die Anzahl aller Dokumente d der Dokumentenkollektion D ist N.

2.2 Vorverarbeitung der Dokumentenkollektionen

Im Fall des textbasierten Information-Retrievals sind die Datenmengen Dokumentenkollektionen. Mit dem Begriff *Dokument* ist dabei ein in der realen Welt vorkommendes Textdokument gemeint, wie beispielsweise ein Buch, ein Zeitungsartikel oder eine Web-Seite (Stein (2005), Stein und Potthast (2006)). Um die Ähnlichkeit zwischen Dokumenten objektiv bestimmen zu können, gibt es verschiedene Ansätze, die es allerdings erfordern, die realen Dokumente d als Objekte in einem Dokumentenmodell darzustellen. Ein häufig benutztes Dokumentenmodell ist das sogenannte Vektorraummodell, bei dem ein Dokument d durch einen Wortvektor $\mathbf{d}$ repräsentiert wird. Die Dimension des Wortvektors $\mathbf{d}$ entspricht dabei der Menge der Wörter bzw. Indexterme ω, die in der Dokumentenkollektion D auftreten. Für jeden Indexterm ω hat der Wortvektor $\mathbf{d}$ von $d \in D$ somit ein Termgewicht. Oft wird als Termgewicht die Termfrequenz, das heißt die Häufigkeit des Indexterms ω in dem Dokument d verwendet (Ferber (2003)). Kommt beispielsweise ein bestimmtes Wort ω_i in der Dokumentenkollektion D vor, welches aber nicht im Dokument d der Kollektion auftritt, so hat der Wortvektor $\mathbf{d}$ von d für das Element mit dem Index i eine Termfrequenz von 0. Bei dieser Art der Dokumentenrepräsentation werden keine Informationen über die Beziehungen der Indexterme untereinander gespeichert. Alle Indexterme werden als unabhängig voneinander betrachtet.

Um den Wortvektor $\mathbf{d}$ mit den Termfrequenzen für jedes Dokument d der Dokumentenkollektion D zu erstellen, müssen alle Dokumente einzeln hinsichtlich des Auftretens der

Indexterme analysiert werden. Bei der Ermittlung der Termfrequenzen werden sogenannte Stoppworte ignoriert. Stoppworte sind sprachspezifische Terme, die häufig und gleichverteilt auftreten und für den Inhalt des Dokuments keine Bedeutung haben. In der deutschen Sprache sind dies zum Beispiel *"und"*, *"ist"* sowie *"in"*. Außerdem wird jeder Term auf seine Grund- bzw. Stammform reduziert, um sie zu Indextermen zu generalisieren. Die Grundformreduktion führt beispielsweise Verben auf ihre Infinitivform und Substantive auf den Nominativ Singular zurück (Ferber (2003)). Durch die Reduktion treten die Indexterme häufiger auf, da verschiedene Beugungsformen eines Wortes auf denselben Indexterm verallgemeinert werden. Die Stoppwortentfernung verbessert die Kompression der Wortvektoren um bis zu 40% laut Stein (2006). Dies ist besonders im Hinblick auf die Anwendung von multidimensionaler Skalierung von Bedeutung, da die Dimensionen der Wortvektoren schon bei der Generalisierung der Indexterme sehr hoch sind und dies einen direkten Einfluss auf die Berechnungszeit hat.

Weiterhin existieren verschiedene andere Termgewichtungsstrategien wie beispielsweise die *tf-idf*-Termgewichtung (Stein (2006)). Dabei werden die Termfrequenzen in Bezug zu ihrem Auftreten in den Dokumenten der gesamten Kollektion gesetzt. Dies relativiert jedoch die Unterschiede der Dokumente in einer Kollektion sehr stark. Auf Grund dessen wurde die Evaluation auf den Wortvektoren $\mathbf{d}$ der Dokumente durchgeführt.

2.3 Metriken im Raum

Ähnlichkeitsmaß des Vektorraums

Den Vektorraum der Wortvektoren kann als ein Koordinatensystem mit k Achsen repräsentiert werden. k steht für die Dimension der Wortvektoren $\mathbf{d}$ der Dokumente $d \in D$. Jede Achse stellt einen Indexterm dar, für den jeder Wortvektor eine Termfrequenz hat. Abbildung 2.1 veranschaulicht dies anhand eines Beispiels.

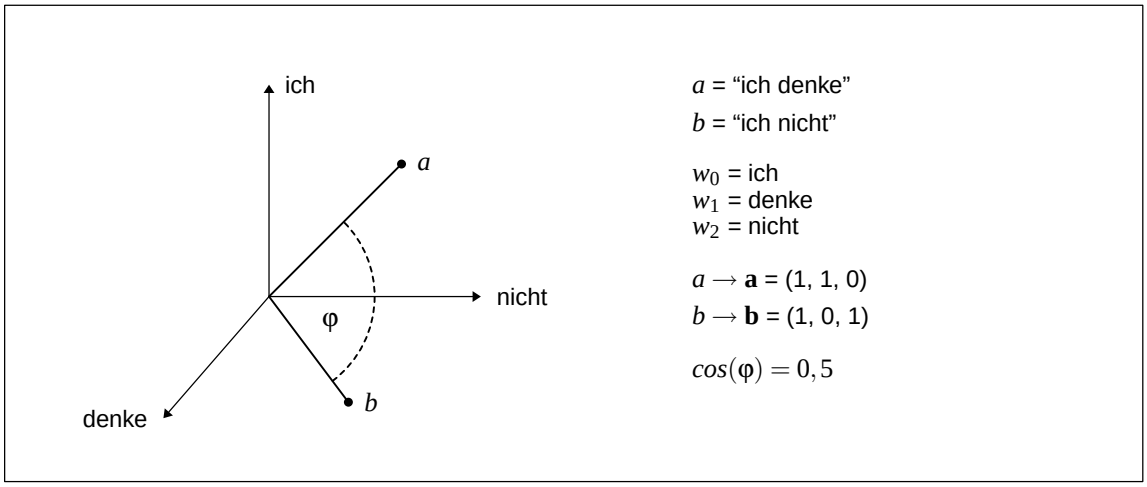


Abb. 2.1: Winkel φ als Ähnlichkeitsmaß zwischen den Wortvektoren $(\mathbf{a},\mathbf{b})$ der Dokumente *a* und *b* im Vektorraum

Zur Berechnung der Ähnlichkeit zweier Dokumente kann das Kosinus-Ähnlichkeitsmaß verwendet werden. Dabei wird die Ähnlichkeit zwischen den Dokumenten $(a,b) \in D$ als Kosinus des Winkels φ ihrer Wortvektoren $(\mathbf{a},\mathbf{b})$ aufgefasst. Der Winkel φ liegt immer im Intervall $[0^\circ\ ,90^\circ\ ]$, da die Termfrequenzen nur positive Werte annehmen können. Der Kosinus des Winkels φ skaliert diesen Wert somit auf das Intervall $[0,1]$ und ergibt sich aus dem Skalarprodukt der Wortvektoren $(\mathbf{a},\mathbf{b})$ wie folgt.

$$cos(\varphi) = \frac{\mathbf{a}^T\mathbf{b}}{\|\mathbf{a}\| \cdot \|\mathbf{b}\|} \tag{2.2}$$

Sind die Wortvektoren $\mathbf{a}'$, $\mathbf{b}'$ bereits normalisiert, berechnet sich das Skalarprodukt zwischen $\mathbf{a}'$ und $\mathbf{b}'$ durch

$$cos(\varphi) = \mathbf{a'}^T \cdot \mathbf{b}' = \sum_{i=1}^{k} \mathbf{a}'_\mathbf{i} \cdot \mathbf{b}'_\mathbf{i} \tag{2.3}$$

Ist der resultierende Wert 1, bedeutet dies, dass sich die Dokumente hinsichtlich des Kosinus-Ähnlichkeitsmaßes gleichen. 0 hingegen bezeichnet die größtmögliche Unähnlichkeit zwischen *a* und *b*.

Die Ähnlichkeit kann mit Formel 2.4 invertiert werden. Dann bedeutet $\delta(\mathbf{a},\mathbf{b}) = 1$ die maximale Distanz zwischen (a,b) und analog dazu $\delta(\mathbf{a},\mathbf{b}) = 0$, dass die Dokumente *a* und *b* sehr ähnlich sind.

$$\delta(\mathbf{a},\mathbf{b}) = 1 - cos(\varphi) \tag{2.4}$$

Distanzberechnung im Zielraum

Für die Berechnung der Distanzen zwischen zwei Punkten p_a und p_b im niedrigdimensionalen Raum wird normalerweise die L_2-Norm in Formel 2.5 verwendet (Mathar (1997)), wobei *n* die Dimension im Zielraum darstellt.

$$\Delta(p_a, p_b) = \sqrt{\sum_{i=1}^{n}(p_{ai} - p_{bi})^2} \tag{2.5}$$

Theoretisch können aber auch andere Normen verwendet werden. Jedoch geht möglicherweise der intuitive Zugang zur Punktdarstellung damit verloren.

Die Distanz δ zwischen den Wortvektoren **a** und **b** kann direkt auf die Distanz Δ zwischen den Punkten p_a und p_b im niedrigdimensionalen Raum übertragen werden (Formel 2.6).

$$\Delta(p_a, p_b) = \delta(\mathbf{a}, \mathbf{b}) \tag{2.6}$$

2.4 Bewertungsmaß der Punktkonfiguration

Eine perfekte Punktkonfiguration würde die Formel 2.7 für jedes Paar von Objekten (a, b) der Datenmenge D und ihrer korrespondierenden Punktpaare (p_a, p_b) erfüllen.

$$\delta(\mathbf{a}, \mathbf{b}) - \Delta(p_a, p_b) = 0 \tag{2.7}$$

Dieser Fall tritt im Allgemeinen nicht ein. Wie bereits erwähnt, sollte ein gutes Verfahren zur multidimensionalen Skalierung die Distanzen δ durch die Distanzen Δ möglichst gut approximieren, so dass

$$\delta(\mathbf{a}, \mathbf{b}) - \Delta(p_a, p_b) \to 0 \tag{2.8}$$

für alle Objektpaare $(a, b) \in D$ gilt.

Der Fehler, den die Punktkonfiguration nach der Berechnung noch enthält, wird als *Stress* bezeichnet. Allgemein wird unter *Stress* der *Stress-1* von Kruskal verstanden (Borg und Groenen (1997)). Der *Stress-1*-Wert wird berechnet mit

$$\sigma = \sqrt{\frac{\sum_{a,b \in D} (\delta(\mathbf{a},\mathbf{b}) - \Delta(p_a, p_b))^2}{\sum_{a,b \in D} \Delta(p_a, p_b)^2}} \tag{2.9}$$

Die Stress-Rechnung selbst hat eine Komplexität von $O(N^2)$. Dadurch wird sie besonders für große Datenmengen sehr zeitintensiv.

Gilt Formel 2.7 für die Punktkonfiguration, so ist der resultierende *Stress-1*-Wert $\sigma = 0$. Ein *gutes* Verfahren sollte für jede Datenmenge D eine Punktkonfiguration generieren, die einen möglichst niedrigen Stresswert erzielt. Allerdings ist ein niedriger Stresswert nicht unbedingt eine Garantie für ein aussagekräftiges und verständliches Layout der Objekte im niedrigdimensionalen Raum. Chalmers u. a. (2003) bemerken, dass eine Punktkonfiguration mit einem etwas höherem Stresswert, subjektiv betrachtet, durchaus ein *besseres* Layout haben kann, als eine andere Punktkonfiguration derselben Daten mit niedrigerem Stress.

3 Verfahren

In diesem Kapitel werden die ausgewählten Verfahren zur multidimensionalen Skalierung erklärt und analysiert. Außerdem wird eine Weiterentwicklung eines Verfahrens vorgestellt.

3.1 Taxonomie der Verfahren

Die Anwendungsmöglichkeiten der multidimensionalen Skalierung sind vielfältig. Ebenso zahlreich sind die Ansätze der Berechnung. Ganz allgemein kann die multidimensionale Skalierung in metrische und nicht-metrische Verfahren eingeteilt werden. Diese Einteilung ergibt sich aus der Ähnlichkeit der Objekte, die bei metrischen Verfahren direkt zwischen allen Objektpaaren berechenbar sein muss. Bei nicht-metrischen bzw. ordinalen Verfahren ist dagegen nur die relative Beziehung der Ähnlichkeiten untereinander gegeben. Borg und Groenen (1997) sowie Cox und Cox (1994) erläutern diesen Ansatz ausführlich.

Die metrischen Verfahren können weiter unterschieden werden. Ein häufig verwendeter und effizienter statistischer Ansatz ist die Eigenvektoranalyse einer $N \times N$-Matrix wie sie auch beim *Latent Semantic Indexing*-Modell angewendet wird (Baeza-Yates und Ribeiro-Neto (1999)). Dabei wird die Punktkonfiguration auf der Basis von Linearkombinationen der Dimensionen erzeugt. Der Nachteil dieses Ansatzes ist ein hoher Rechenaufwand, der die Komplexität von $O(N^3)$ widerspiegelt (Chalmers u. a. (2003)). Außerdem muss die ge-

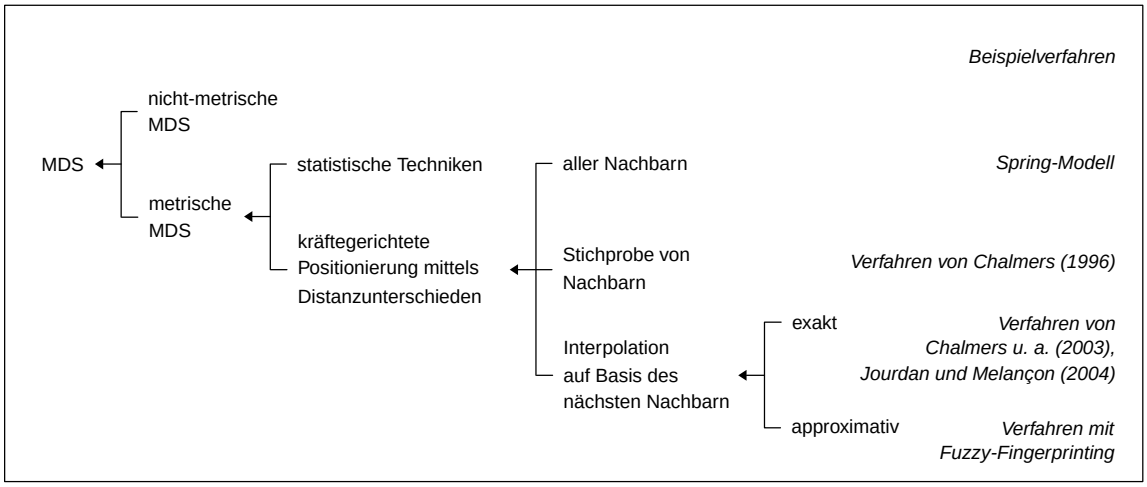


Abb. 3.1: Taxonomie der Verfahren zur multidimensionalen Skalierung

samte Rechnung wiederholt werden, wenn die Datenmenge erweitert wird. Bei Verfahren, die auf kräftegerichteter Positionierung mittels Distanzunterschieden basieren, entsteht die Punktkonfiguration durch Iteration. Theoretisch kann die Punktkonfiguration nach jeder Iteration als Plot ausgegeben werden. Daher ist es möglich, eine Berechnung vor dem Ende abzubrechen, wenn ersichtlich wird, dass die Punktkonfiguration unzureichend ist. Das ist ein wesentlicher Vorteil gegenüber statistischen Techniken. Abbildung 3.1 fasst die Taxonomie der Verfahren zusammen.

Abgesehen von Verfahren iterativer Natur sind auch deterministische Verfahren wie bei Don und Hanusse (2006) möglich. Allerdings sind die Ergebnisse mit den hier betrachteten Verfahren nicht direkt vergleichbar, deshalb wurde der deterministische Ansatz nicht weiter verfolgt.

Für die betrachteten Verfahren werden algorithmische Zusammenfassungen angegeben und ihre Komplexität im O-Kalkül abgeschätzt.

3.2 Kräftegerichtete Positionierung basierend auf Distanzunterschieden

3.2.1 Das Spring-Modell

Das einfachste Verfahren der metrischen multidimensionalen Skalierung ist das sogenannte Spring-Modell. Dabei wird die Distanz $\delta(\mathbf{a},\mathbf{b})$ zwischen einem Objektpaar (a,b) als Länge einer Feder im Ruhezustand aufgefasst. Die Distanz $\Delta(p_a,p_b)$ zwischen den Punkten p_a und p_b, die die Objekte a und b repräsentieren, stellt die Dehnung bzw. Stauchung der Feder dar. Bevor die Berechnung begonnen wird, werden die Positionen aller Punkte mit zufälligen Werten initialisiert. Deshalb sind die gedachten Federn zwischen den Punkten zufällig gedehnt oder gestaucht.

Die Feder der Länge $\Delta(p_a,p_b)$ versucht nun wieder ihren Ruhezustand der Länge $\delta(\mathbf{a},\mathbf{b})$ einzunehmen. Dabei wirken Kräfte auf die Federenden, die nach dem Hookeschen Gesetz (Formel 3.1) für eine Feder für alle Objektpaare (a,b) der Datenmenge D berechnet werden können.

$$F_s = k_s \cdot (\delta(\mathbf{a},\mathbf{b}) - \Delta(p_a,p_b)) \tag{3.1}$$

Die Federkonstante k_s hat Einfluss darauf, wie viel Kraft auf die Federenden wirken soll, bzw. wie schnell die Feder sich wieder ausdehnt oder zusammenzieht. Damit das System nicht unendlich schwingt, wird ein Dämpfungsfaktor (Formel 3.2) berechnet, der durch die Dämpfungskonstante k_d angepasst werden kann.

$$F_d = (\vec{v_b} - \vec{v_a}) \cdot \frac{p_b - p_a}{|p_b - p_a|} \cdot k_d \tag{3.2}$$

Für jeden Punkt muss somit eine Position, ein Kräfte- und ein Geschwindigkeitsvektor gespeichert werden. Bei der Initialisierung des Verfahrens sind alle Elemente des Kräfte- und Geschwindigkeitsvektors eines Punktes mit 0 belegt. Die anziehenden und abstoßenden Kräfte, die auf die Federenden wirken, ergeben sich für jedes Objektpaar (a,b) nach Matyka (2004) aus

$$\vec{F} = \frac{p_b - p_a}{|p_b - p_a|} \cdot (F_s + F_d) \tag{3.3}$$

Der Kräftevektor $\vec{F}$ wird als anziehende Kraft auf den Kräftevektor $\vec{F_a}$ des Objekts a addiert und als abstoßende Kraft vom Kräftevektor $\vec{F_b}$ des Objekts b subtrahiert. Nachdem die Kräfte für alle Objekte berechnet wurden, müssen diese integriert werden. Chalmers (1996) bemerkt, dass die Euler-Integration in diesem Fall ausreicht. Dafür wird zuerst die Geschwindigkeit mit der Kraft, die auf den Punkt wirkt, aktualisiert.

$$\vec{v_a} = (\vec{v_a} + \vec{F_a} \cdot \Delta t) \cdot c \tag{3.4}$$

c ist dabei ein Faktor, der zur Beeinflussung der Berechnung genutzt wird und Δt der Zeitschritt der Integrationsrechnung. Anschließend wird die Position nach Formel 3.5 neu berechnet. Abbildung 3.2 illustriert das Spring-Modell anhand einer vereinfachten Punktkonfiguration.

$$p_a = p_a + \vec{v_a} \cdot \Delta t \tag{3.5}$$

Durch die iterative Natur des Verfahrens wird die Punktkonfiguration kontinuierlich verändert, verbessert und die hochdimensionale Ähnlichkeit zwischen den Objektpaaren approximiert. Dieses Verfahren wird auch in der Computergrafik als sogenanntes Partikelsystem

oft zur Simulation angwandt. Die physikalischen Grundlagen werden bei Eberly (2004) ausführlich dargestellt.

Eingabe : Menge D von Datenobjekten ($i, j \in D$) der Größe $|D| := N$
Distanzmaß $\delta(\mathbf{i}, \mathbf{j})$ für alle Objektpaare $(i, j) \in D^2$
Ausgabe : $\forall i \in D \; \exists$ Punktkoordinaten p_i

```
1 begin
2     repeat
3         forall i ∈ D do
4             forall j < i do
5                 berechne Kräfte zwischen i und j (nach Formel 3.3);
6         forall i ∈ D do
7             aktualisiere Koordinaten von p_i (nach Formel 3.5);
8     until Abbruchkriterium erreicht, z.B. N Iterationen ;
9 end
```

Algorithmus 1 : Spring-Modell

Algorithmus 1 erfasst das Verfahren noch einmal als Pseudocode. Dabei zeigt sich die Komplexität einer Iteration von $O(N^2)$, die sich aus der Berechnung der Kräfte zwischen einem Objekt und allen anderen Objekten der Datenmenge ergibt. Die Anzahl der Iterationen, die benötigt werden um eine Punktkonfiguration zu generieren, wird durchschnittlich mit N abgeschätzt. Daraus resultiert eine allgemeine Komplexität des Spring-Modells von $O(N^3)$. Obwohl es die besten Punktkonfigurationen liefern würde, ist dieses Verfahren auf Grund der hohen Komplexität für Kollektionen mit mehr als 1 000 Objekten nicht anwendbar.

3.2.2 Das Verfahren von Chalmers (1996)

Um die Komplexität des Spring-Modells zu reduzieren, schlägt Chalmers (1996) vor die Kräfteberechnungen auf eine konstant niedrige Anzahl zu beschränken. Das Verfahren sieht vor, für jedes Objekt i eine Menge V_i der Größe $|V_i| := V_{max}$ und eine Menge S_i der Größe $|S_i| := S_{max}$ anzulegen. Die Menge V_i wird über alle Iterationen mitgeführt, während die Menge S_i jedes Mal neu erstellt wird. Die Menge V_i ist nach der Distanz $\delta(\mathbf{i}, \mathbf{v})$ für alle $v \in V_i$ aufsteigend sortiert. Die maximale Distanz δ zwischen dem Objekt i und den

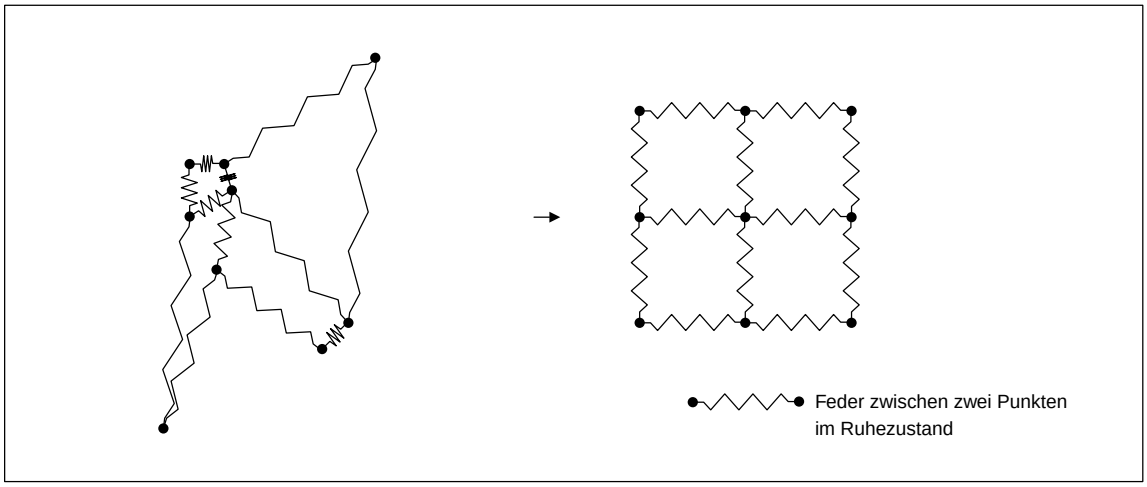


Abb. 3.2: Links: zufällige Punktkonfiguration mit gedehnten bzw. gestauchten Federn zwischen den Punkten; Rechts: die Punktkonfigurationen, wenn sich alle Federn zwischen den Punkten im Ruhezustand befinden und damit $\Delta = \delta$ gilt

Objekten der Menge V_i wird als m_i für jede Menge V_i zwischengespeichert und wenn sich die Menge ändert entsprechend aktualisiert. Für ein Objekt i werden Kandidatenobjekte j aus der Datenmenge D zufällig ausgewählt. Alle Kandidatenobjekte j werden geprüft, ob sie als nächster Nachbar zu i in Frage kommen. Das heißt, wenn die Distanz $\delta(\mathbf{i},\mathbf{j}) < m_i$ ist, wird j der Menge V_i hinzugefügt. Da V_i eine feste Größe hat, ersetzt möglicherweise j ein Objekt in der Menge V_i in diesem Schritt. Wenn j kein nächster Nachbar von i ist und daher nicht in V_i aufgenommen werden kann, wird es zur Menge S_i hinzugefügt. Ist die Menge S_i mit S_{max} Objekten vollständig gefüllt, werden die Kräfte zwischen i und den Objekten der Mengen S_i und V_i berechnet. Anschließend werden die aktualisierten Kräfte für alle Objekte i integriert. Algorithmus 2 fasst diesen Ablauf zusammen.

Durch die Beschränkung der Mengen auf V_{max} und S_{max} Objekte, müssen für alle Objekte i nur $(V_{max} + S_{max})$ Kräfteberechnungen durchgeführt werden. Damit reduziert sich die Komplexität einer Iteration auf $O(N)$. Mit N Iterationen, die für eine Punktkonfiguration benötigt werden, hat das Verfahren eine allgemeine Komplexität von $O(N^2)$. Trotz dieser Verbesserung bemerken Morrison und Chalmers (2003), dass dieses Verfahren für Datenmengen mit mehr als ein paar tausend Objekte nicht durchführbar ist.

Eingabe : Menge D von Datenobjekten ($i, j \in D$) der Größe $|D| := N$
Distanzmaß $\delta(\mathbf{i}, \mathbf{j})$ für alle Objektpaare $(i, j) \in D^2$
Ausgabe : $\forall i \in D \; \exists$ Punktkoordinaten p_i

```
1  begin
2      Menge S_i der Größe |S_i| := S_max ∀i ∈ D;
3      Menge V_i der Größe |V_i| := V_max ∀i ∈ D;
4      m_i := max_{v∈V_i} δ(i,v) ∀i ∈ D;
5      repeat
6          forall i ∈ D do
7              S_i := ∅;
8              while |S_i| < S_max do
9                  wähle ein Element j aus D mit j ≠ i;
10                 if δ(i,j) < m_i then
11                     V_i := (j, δ(i,j)) ∪ V_i;
12                 else
13                     S_i := j ∪ S_i;
14             forall j ∈ S_i ∩ V_i do
15                 berechne Kräfte zwischen i und j (nach Formel 3.3);
16         forall i ∈ D do
17             aktualisiere Koordinaten von p_i (nach Formel 3.5);
18     until Abbruchkriterium erreicht, z.B. N Iterationen ;
19 end
```

Algorithmus 2 : Verfahren von Chalmers (1996)

3.3 Verfahren mit Interpolation auf Basis des nächsten Nachbarn

3.3.1 Aufbau der hybriden Verfahren

Die Spring-basierten Verfahren beruhen darauf, dass die Punktkonfiguration im Verlauf der Iterationen optimiert wird. Wie viele Iterationen dafür tatsächlich notwendig sind, ist unbestimmt. Ein Verfahren mit quadratischer Komplexität ist allerdings für große Datenmengen nicht anwendbar. Aus diesem Grund schlagen Chalmers u. a. (2003) einen hybriden Ansatz vor, der das Spring-basierte Verfahren von Chalmers (1996) für einen kleinen Teil der Datenmenge verwendet und für die restlichen Objekte der Datenmenge auf dieser Basis Punktkoordinaten interpoliert.

Der erste Schritt der hybriden Verfahren ist die Wahl einer Teilmenge S der Größe $\sqrt{N}$ aus der Datenmenge D. Für diese Teilmenge S wird mit dem Verfahren von Chalmers (1996) eine Punktkonfiguration berechnet. Wenn auf diese Teilmenge $\sqrt{N}$ Iterationen angewendet werden, hat diese erste Phase eine Komplexität von $O(\sqrt{N}\sqrt{N}) = O(N)$. Die übrigen $(N - \sqrt{N})$ Objekte der Menge $D \backslash S$ werden anschließend mit einer leicht adaptierten Version der Interpolationsstrategie von Brodbeck und Girardin (1998) in die Punktkonfiguration eingefügt. Die Interpolation, die von Chalmers u. a. (2003) für alle Objekte i der Menge $D \backslash S$ verwendet wird, geht in folgenden Schritten vor:

1. Suche den nächsten Nachbar n bzw. das ähnlichste Objekt n zu i in der Teilmenge S.
2. $\delta(\mathbf{i}, \mathbf{n})$ wird als Radius des Kreises um den Punkt p_n aufgefasst. Für die Winkel $\alpha = 0^\circ$, 90° , 180° , 270° werden die Positionen c_α auf diesem Kreis berechnet.
3. Für alle Positionen c_α wird die Summe 3.6 über alle Objekte s der Teilmenge S berechnet.

$$q_\alpha = \sum_{s \in S} |\delta(\mathbf{i}, \mathbf{s}) - \Delta(c_\alpha, p_s)| \tag{3.6}$$

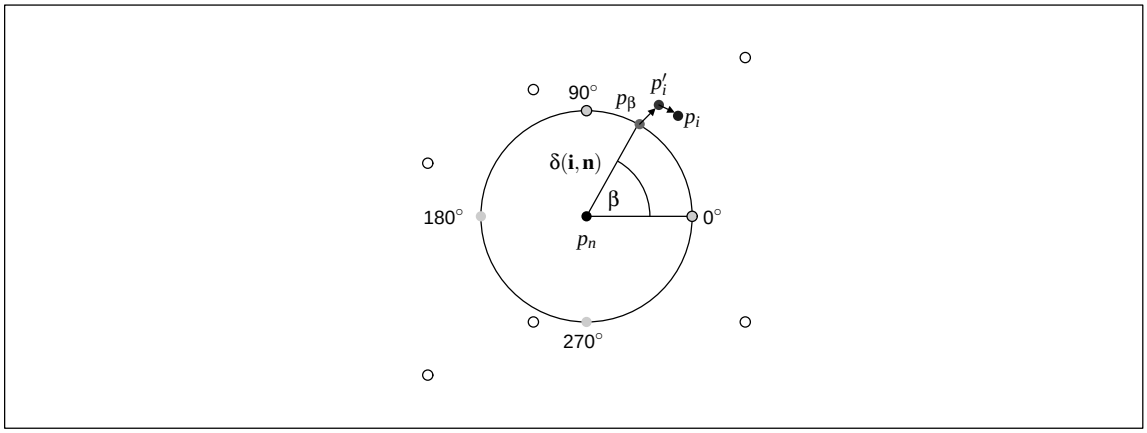


Abb. 3.3: Interpolation des Objekts i auf einem Kreis um seinen nächsten Nachbarn p_n mit dem Radius $\delta(\mathbf{i},\mathbf{n})$

Anhand der niedrigsten Werte von q_α werden die Winkel für Positionen auf dem Kreisumfang bestimmt, die die Grenzen des Kreisquadranten darstellen, der für die weitere Interpolation am geeignetsten ist.

4. In dem gewählten Kreisquadranten wird eine binäre Suche nach dem Winkel β ausgeführt. Der Punkt p_β des Kreises mit dem Winkel β soll die Summe 3.7 über alle Objekte s der Teilmenge S minimieren.

$$z = \sum_{s \in S} (\delta(\mathbf{i},\mathbf{s}) - \Delta(p_\beta, p_s)) \tag{3.7}$$

5. Der Punkt p_β auf dem Kreis mit dem Winkel β wird berechnet und dem Punkt p_i zugewiesen.
6. Für eine konstante Anzahl an Iterationen wird der akkumulierte Kräftevektor zwischen dem Objekt i und einer zufälligen Stichprobe der Teilmenge S der Größe $N^{1/4}$ berechnet und auf die Position des Punktes p_i addiert.

Abbildung 3.3 verdeutlicht die angewendete Interpolationsstrategie, die in Funktion 4 zusammengefasst ist.

Bei Chalmers u. a. (2003), Morrison u. a. (2002b) und Morrison u. a. (2002a) wird nicht deutlich, mit welchen Objekten die Summen 3.6 und 3.7 berechnet werden bzw. wie groß die verwendete Teilmenge in diesem Fall ist. Chalmers u. a. (2003) geben an, dass alle Parameter von Brodbeck und Girardin (1998) übernommen worden sind. Allerdings machen Brodbeck und Girardin (1998) keine Angaben darüber, wie die Parameter gewählt wurden. Die ursprüngliche Interpolation von Brodbeck und Girardin (1998) berechnete nur für eine festgelegte Anzahl n_f die Positionen auf dem Kreis. Für diese n_f Positionen wurden die Summen der Differenzen wie in Formel 3.6 berechnet. Dem Punkt p_i wurde die Position mit dem kleinsten Summenergebnis zugewiesen. Chalmers u. a. (2003) führt aber nach Feststellung des geeigneten Kreisquadranten eine binäre Suche aus. Dabei wird bei der Berechnung der Summe der Differenzen (3.7) von der Teilmenge gesprochen. Da vorher keine Einschränkung gemacht wurde, ist die Teilmenge S der Größe $\sqrt{N}$ als Basis für die Summenrechnung anzunehmen.

Für den Interpolationsteil - die Punkte 2 bis 6 - ergibt sich damit auf Grund der Verwendung der gesamten Teilmenge S eine allgemeine Komplexität von $O(N\sqrt{N})$. Chalmers u. a. (2003) geben diesen Interpolationsteil mit $O(N)$ an. Dies ist nicht nachvollziehbar, da die Einzelschritte der Interpolation direkt von der Größe der Teilmenge S oder einer Stichprobe aus S abhängen. Schon bei Punkt 6 werden mehrere Iterationen mit einer Stichprobe der Größe $N^{1/4}$ berechnet. Die Größe dieser Stichprobe ist folglich direkt abhängig von der Größe der Datenmenge D und sollte deshalb nicht vernachlässigt werden. Damit hat schon Punkt 6 eine allgemeine Komplexität von $O(N \cdot N^{1/4})$. Da in Schritt 3 und 4 Summen über die gesamte Teilmenge S gebildet werden, haben diese Teile eine allgemeine Komplexität von $O(N\sqrt{N})$. Morrison und Chalmers (2004) verwenden für die Summenberechnung ebenfalls eine Stichprobe der Größe $N^{1/4}$. So lässt sich die Komplexität der Interpolation allerdings nur auf $O(N \cdot N^{1/4})$ reduzieren. Die Komplexität von $O(N\sqrt{N})$ bzw. $O(N \cdot N^{1/4})$ erklärt auch die Beobachtung von Morrison und Chalmers (2003), dass der Interpolationsteil trotz der in Chalmers u. a. (2003) angenommenen Komplexität $O(N)$ die zeitintensivere Phase ist. Dabei betrachten Chalmers u. a. (2003) die Nächste-Nachbar-Suche als aufwendigsten Schritt des Verfahrens. Morrison und Chalmers (2003) gehen davon aus, dass dieses Verhalten bei ausreichend großen Datenmengen umschlagen wird. Mit den korrigierten Komplexitäten ist dies nicht zu erwarten. Wenn jedoch die Größe der Teilmenge konstant auf einen Wert unabhängig von N und wesentlich kleiner festgelegt wird, reduziert sich die Komplexität der Interpolation auf $O(N)$.

Im Folgenden wird die Interpolation bis auf die zufällige Stichprobe der Größe $N^{1/4}$ auf Basis der gesamten Teilmenge S ausgeführt. Da bei den folgenden Verfahren die Betonung auf den unterschiedlichen Strategien für Punkt 1 der Interpolation liegt, wurde die ganze Teilmenge S gewählt, um eine möglichst gute Interpolation zu gewährleisten. Das bedeutet, es wird für die Interpolation eine Komplexität von $O(N\sqrt{N})$ angenommen.

```
    Eingabe  : Punkt p
               Winkel α
               Radius r
    Ausgabe  : Position p_α auf dem Kreis mit dem Radius r um p
1   begin
2   |   p_αx := p_x + cos(α) · r;
3   |   p_αy := p_y + sin(α) · r;
4   |   return p_α
5   end
```

Funktion 3 : circlePoint(p,α,r)

Nach der Interpolation aller Objekte, wird für eine konstante Anzahl an Iterationen das Verfahren von Chalmers (1996) auf die gesamte Datenmenge D angewendet, um die Punktkonfiguration zu verfeinern. Dieser letzte Schritt hat eine Komplexität von $O(N)$ auf Grund der konstanten Iterationszahl.

Die Verfahren von Chalmers u. a. (2003), Jourdan und Melançon (2004) sowie die Verbesserung durch eine Methode des Similarity-Hashings unterscheiden sich jeweils nur in der Strategie für die Nächste-Nachbar-Suche und werden im Folgenden näher erläutert.

3.3.2 Das Verfahren von Chalmers u. a. (2003)

Das in Chalmers u. a. (2003) ursprünglich als *hybride multidimensionale Skalierung* bezeichnete Verfahren verwendet eine lineare Suche, um den nächsten Nachbarn für das Objekt i in der Teilmenge S zu finden. Algorithmus 5 veranschaulicht dies. Die Suche hat für jedes Objekt i eine Komplexität von $O(\sqrt{N})$. Damit hat die Nächste-Nachbar-Suche dieselbe Komplexität wie der bereits erläuterte Interpolationsteil mit $O(N\sqrt{N})$. Folglich ist die Gesamtkomplexität des Verfahrens $O(N\sqrt{N})$.

Eingabe : Objekt i, das interpoliert werden soll
der nächste Nachbar n von i
Distanzmaß $\delta(\mathbf{i},\mathbf{j})\ \forall i,j \in D$
Distanzmaß $\Delta(p_i,p_j)\ \forall i,j \in D$
Ausgabe : $\forall i \in D\ \exists$ Punktkoordinaten p_i
Variablen : Teilmenge S der Größe $|S| := \sqrt{N}$

1 **begin**
2 $r := \delta(\mathbf{i},\mathbf{n})$;
3 **for** $\alpha := 0^\circ, 90^\circ, 180^\circ, 270^\circ$ **do**
4 $q_\alpha := \sum_{s\in S} \delta(\mathbf{s},\mathbf{i}) - \Delta(p_s, \texttt{circlePoint}(n,\alpha,r))$;
5 *bestimme die obere und untere Grenze B_o bzw. B_u des Kreisquadranten anhand der niedrigsten Werte von q_α;*
6 $\beta := null$;
7 **while** $B_u < B_o$ **do**
8 $mid := \lceil \frac{B_u+B_o}{2} \rceil$;
9 **if** $mid = B_u \vee mid = B_o$ **then**
10 $o := \sum_{s\in S} |\delta(\mathbf{s},\mathbf{i}) - \Delta(p_s, \texttt{circlePoint}(n,B_o,r))|$;
11 $u := \sum_{s\in S} |\delta(\mathbf{s},\mathbf{i}) - \Delta(p_s, \texttt{circlePoint}(n,B_u,r))|$;
12 **if** $u < o$ **then**
13 $\beta := B_u$;
14 **else**
15 $\beta := B_o$;
16 **else**
17 $o := \sum_{s\in S} |\delta(\mathbf{s},\mathbf{i}) - \Delta(p_s, \texttt{circlePoint}(n,mid+1,r))|$;
18 $m := \sum_{s\in S} |\delta(\mathbf{s},\mathbf{i}) - \Delta(p_s, \texttt{circlePoint}(n,mid,r))|$;
19 $u := \sum_{s\in S} |\delta(\mathbf{s},\mathbf{i}) - \Delta(p_s, \texttt{circlePoint}(n,mid-1,r))|$;
20 **if** $m > o$ **then**
21 $B_u := mid + 1$;
22 $\beta := B_u$;
23 **else if** $m > u$ **then**
24 $B_o := mid - 1$;
25 $\beta := B_o$;
26 **else**
27 $\beta := mid$;
28 $p_i := \texttt{circlePoint}(n,\beta,r)$;
29 *wähle eine Stichprobe R der Größe $|R| := N^{1/4}$ aus S;*
30 **for** *konstante Anzahl an Iterationen* **do**
31 $\vec{f} := null$;
32 **forall** $r \in R$ **do**
33 $\vec{f} := \vec{f} + k_s \cdot (\delta(\mathbf{i},\mathbf{r}) - \Delta(p_i,p_r)) \cdot \frac{p_r - p_i}{|p_r - p_i|}$;
34 $p_i := p_i + \vec{f}$;
35 **end**

Funktion 4 : interpolate(i,n,$\delta(\mathbf{i},\mathbf{n})$)

Eingabe : Menge D von Datenobjekten ($i, j \in D$) der Größe $|D| := N$
Distanzmaß $\delta(\mathbf{i},\mathbf{j})$ für alle Objektpaare $(i, j) \in D^2$
Ausgabe : $\forall i \in D\ \exists$ Punktkoordinaten p_i

```
1  begin
2      generiere die Punktkonfiguration der Teilmenge S mit dem Verfahren von Chalmers (1996);
3      forall i ∈ D\S do
4          n := ∅;
5          forall s ∈ S do
6              if δ(i,s) < δ(i,n) then
7                  n := s;
8          interpoliere Koordinaten für p_i mit Funktion 4: interpolate(i,n,δ(i,n));
9      verfeinere die Punktkonfiguration mit dem Verfahren von Chalmers (1996) über alle
       Objekte i ∈ D;
10 end
```

Algorithmus 5 : Verfahren von Chalmers u. a. (2003)

Morrison und Chalmers (2003) bzw. Morrison und Chalmers (2004) stellen eine Verbesserung des Verfahrens vor, bei dem die Nächste-Nachbar-Suche auf Basis von Pivotobjekten und stufenweiser Sortierung der Teilmenge S in Intervalle funktioniert. Diese Version hat eine Komplexität von $O(N \cdot N^{1/4})$, wenn von der Interpolation ebenfalls mit $O(N \cdot N^{1/4})$ ausgegangen wird. Dieses Verfahren wurde allerdings zugunsten des Verfahrens von Jourdan und Melançon (2004) nicht weiter betrachtet.

3.3.3 Das Verfahren von Jourdan und Melançon (2004)

Bei Jourdan und Melançon (2004) wird ein Verfahren vorgestellt, das die Nächste-Nachbar-Suche durch sortierte Listen beschleunigt. Nach der Generierung der Punktkonfiguration für die Teilmenge S werden zunächst in einem Vorverarbeitungsschritt $N^{1/4}$ Pivotobjekte zufällig aus S gewählt. Für jedes dieser Pivotobjekte o wird eine Liste L_o angelegt, die alle Objekte s der Teilmenge S unter der Bedingung $o \neq s$ enthält. Diese Liste L_o ist aufsteigend nach der Distanz $\delta(\mathbf{i},\mathbf{s})$ für alle $s \in S$ geordnet. Dieser Schritt hat nach Jourdan und Melançon (2004) im Durchschnitt eine Komplexität von $O(\sqrt{N} \cdot log(N))$ und im schlechtesten Fall von $O(N)$.

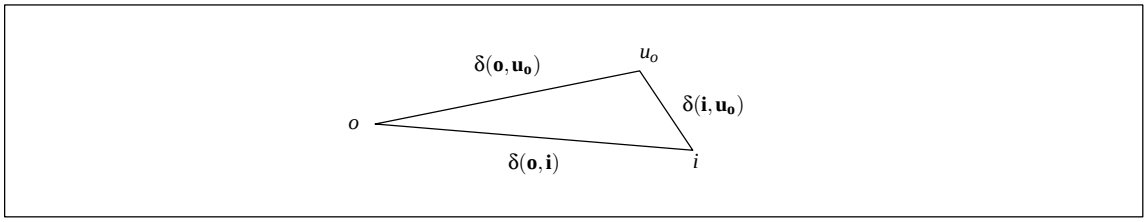


Abb. 3.4: Wenn $\delta(\mathbf{o},\mathbf{i}) \approx \delta(\mathbf{o},\mathbf{u_o})$, impliziert dies die Annahme, dass $\delta(\mathbf{i},\mathbf{u_o})$ klein ist

Um nun für ein zu interpolierendes Objekt i den nächsten Nachbarn n zu finden, werden alle Listen L_o binär durchsucht, um die Formel 3.8 mit einem Objekt u_o zu minimieren.

$$|\delta(\mathbf{o},\mathbf{i}) - \delta(\mathbf{o},\mathbf{u_o})| \tag{3.8}$$

Bei dieser Strategie muss die Dreiecksungleichung für die Unähnlichkeiten δ gelten. Das heißt, wenn Formel 3.8 gültig ist, impliziert dies die Annahme, dass $\delta(\mathbf{i},\mathbf{u_o})$ klein ist, wie Jourdan und Melançon (2004) erläutern. Abbildung 3.4 veranschaulicht diese Theorie.

Das Verfahren ist in Algorithmus 6 zusammengefasst. Das Durchsuchen einer Liste L_o hat eine Komplexität von $O(log(N))$. Die Anzahl der Pivotobjekte wird, wie Morrison und Chalmers (2003) vorschlagen, bei Jourdan und Melançon (2004) mit $N^{1/4}$ beibehalten. Damit kommt die Nächste-Nachbar-Suche auf eine Komplexität von $O(N \cdot N^{1/4} \cdot log(N))$. Wenn die Anzahl der Pivotobjekte konstant gewählt wird, würde das Verfahren auf eine Komplexität von $O(N \cdot log(N))$ kommen. Die Komplexität der Nächsten-Nachbar-Suche liegt allerdings in jedem Fall unter der des Interpolationsteils. Damit fällt das Verfahren insgesamt auf $O(N\sqrt{N})$ zurück.

Eingabe : Menge D von Datenobjekten ($i, j \in D$) der Größe $|D| := N$
Distanzmaß $\delta(\mathbf{i}, \mathbf{j})$ für alle Objektpaare $(i, j) \in D^2$
Ausgabe : $\forall i \in D \; \exists$ Punktkoordinaten p_i

```
1  begin
2      generiere die Punktkonfiguration der Teilmenge S mit dem Verfahren von Chalmers (1996);
3      Menge O der Pivotobjekte aus S der Größe |O| := N^{1/4};
4      forall o ∈ O do
5          sortiere alle Objekte s ∈ S in die Liste L_o aufsteigend nach δ(o, s) ;
6      forall i ∈ D\S do
7          n := ∅;
8          forall o ∈ O do
9              finde mit binärer Suche in L_o ein Objekt u_o, dass |δ(o, i) − δ(o, u_o)| minimiert;
10             if δ(i, u_o) < δ(i, n) then
11                 n := u_o;
12         interpoliere Koordinaten für p_i mit Funktion (4): interpolate(i,n,δ(i, n));
13     verfeinere die Punktkonfiguration mit dem Verfahren von Chalmers (1996) über alle
       Objekte i ∈ D;
14 end
```

Algorithmus 6 : Verfahren von Jourdan und Melançon (2004)

3.3.4 Der Multiscale-Ansatz von Jourdan und Melançon (2004)

Jourdan und Melançon (2004) stellen neben der Verbesserung der Nächsten-Nachbar-Suche noch ein weiteres Verfahren vor, das in ihren Experimenten sowohl schnellere Laufzeiten vorweisen konnte als auch bessere Stresswerte lieferte.

Chalmers u. a. (2003) schlagen vor die Teilmengenkonfiguration mit dem Verfahren von Chalmers (1996) zu generieren. Dies basiert auf der Vermutung einen repräsentativen Teil der Datenmenge möglichst optimal und mit vertretbarem Zeitaufwand zu berechnen. Jourdan und Melançon (2004) sind der Auffassung, diese Annahme sei unbegründet. Stattdessen wird vorgeschlagen, zunächst nur wenige Objekte und Kräfte der Punktkonfiguration zu betrachten und die Anzahl der Objekte sukzessiv zu erweitern. Die Strategie, die das Verfahren von Jourdan und Melançon (2004) verwendet und bereits erläutert wurde, soll dafür rekursiv zur Generierung der Teilmengenkonfiguration verwendet werden. Eine detailliertere Beschreibung dieses Ansatzes gibt es nicht. In Funktion 7 wird dieser Ansatz dargestellt. Die Komplexität lässt sich dadurch abschätzen, dass die Interpolation, die

ebenfalls auf die $N^{1/2} - N^{1/4}$ Objekte der Teilmenge angewendet wird auf $N^{1/4}$ Objekten der Teilmenge S basiert. Die Komplexität der weiteren rekursiven Aufrufe ist kleiner als die Komplexität des ersten Aufrufs. Die rekursive Generierung der Punktkonfiguration der Teilmenge S hat somit eine Komplexität von $O(N^{3/4})$. Auch bei diesem Ansatz überwiegt weiterhin die Komplexität der Interpolation. Das Verfahren hat eine Gesamtkomplexität von $O(N\sqrt{N})$.

```
Eingabe  : Teilmenge S
           Distanzmaß δ(i, j) für alle Objektpaare (i, j) ∈ D²
Ausgabe  : ∀s ∈ S ∃ Punktkoordinaten p_s
1  begin
2      if |S| = 1 then
3          p_s := zufällige Punktkoordinaten;
4          return S
5      else
6          wähle eine Teilmenge S_a der Größe |S_a| := √|S| aus S;
7          S_a :=multiscale(S_a);
8          Menge O der Pivotobjekte aus S_a der Größe |O| := |S_a|^(1/4);
9          forall o ∈ O do
10             sortiere alle Objekte s ∈ S_a in die Liste L_o aufsteigend nach δ(o, s)
11         forall i ∈ S\S_a do
12             n := ∅;
13             for o ∈ O do
14                 finde mit binärer Suche in L_o ein Objekt u_o, dass |δ(o, i) − δ(o, u_o)|
                   minimiert;
15                 if δ(i, u_o) < δ(i, n) then
16                     n := u_o;
17             interpoliere Koordinaten für p_i mit Funktion (4): interpolate(i,n,δ(i, n));
18         return S
19 end
```

Funktion 7 : multiscale(S)

3.3.5 Das Verfahren mit Fuzzy-Fingerprinting

Die Ähnlichkeitsberechnung bei hochdimensionalen Datenmengen trägt signifikant zur Gesamtlaufzeit bei. Eine weitere Verbesserung des hybriden Verfahrens wäre folglich eine Minimierung der Anzahl dieser Berechnungen bei der Nächsten-Nachbar-Suche. In Chal-

mers u. a. (2003) wird bereits die Möglichkeit diskutiert, dass die Bestimmung des exakten nächsten Nachbarn unnötig ist und eine gute Approximierung meist ausreichen würde. Deshalb wird vorgeschlagen, die Nächste-Nachbar-Suche durch Hashing zu ersetzen. Das sogenannte Similarity-Hashing bezeichnet nach Potthast (2006) die Techniken, die speziell konstruierte Hashfunktionen auf Dokumente anwenden. Die Hashfunktionen H werden so konstruiert, dass die normalerweise unerwünschten Hashkollisionen in diesem Fall mit großer Wahrscheinlichkeit als Zeichen für die Ähnlichkeit zwischen Dokumenten, bei denen sie auftreten, aufgefasst werden können. Eine Hashfunktion $h \in H$ muss dafür die in Formel 3.9 angegebene Eigenschaft besitzen (Stein (2005)).

$$h(a) = h(b) \Rightarrow cos(\varphi) \geq 1 - \varepsilon \text{ mit } a, b \in D,\ a \neq b,\ 0 < \varepsilon \ll 1 \tag{3.9}$$

Die Hashkollisionen reduzieren die Ähnlichkeit zwischen den Dokumenten auf die Unterscheidung zwischen *ähnlich* und *nicht ähnlich*. Nach Einordnung der Dokumente in eine Hashtabelle können keine Rückschlüsse mehr darüber gezogen werden, wie ähnlich die Dokumente in einem Bucket der Hashtabelle sind, ohne die Ähnlichkeiten tatsächlich zu berechnen. Die Voraussetzung, dass die Ähnlichkeiten der Dokumente eines Buckets in dem Intervall $[1-\varepsilon, 1]$ liegen, muss durch die Hashfunktion erfüllt werden. Das bedeutet, bei einer Suche nach dem nächsten Nachbarn zum Anfragedokument q muss der Hashwert $h(q)$ berechnet werden. Das Ergebnis der Suche ist die Menge der Dokumente, die für eine Hashfunktion h denselben Hashwert haben. Alle Dokumente der Ergebnismenge sind folglich approximative nächste Nachbarn von q.

Das Fuzzy-Fingerprinting von Stein (2005) ist eine Variante des Similarity-Hashings und wird im Folgenden genauer erläutert.

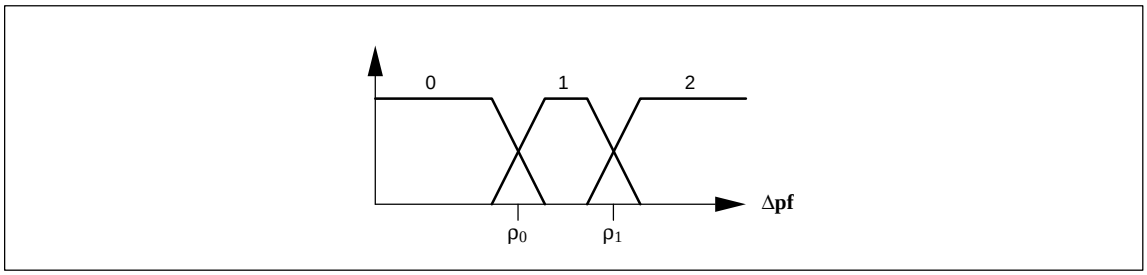


Abb. 3.5: Fuzzifizierungsschema ρ mit $r = 3$ Intervallen: 0 - keine Abweichung, 1 - kleine Abweichung, 2 - große Abweichung von der erwarteten Frequenzverteilung

Fuzzy-Fingerprinting

Fuzzy-Fingerprinting als eine Methode des Similarity-Hashings wurde speziell für die Anwendung im textbasierten Information-Retrieval konzipiert. Der Fuzzy-Fingerprint eines Dokuments $d \in D$ ergibt sich wie folgt (Stein (2005)):

1. Die Indexterme müssen für jedes Dokument $d \in D$ extrahiert werden. Dazu kann ein Vektorraummodell aufgestellt werden, wie schon in Abschnitt 2.2 angesprochen wurde.
2. Mit einer festgelegten Anzahl k von Präfixäquivalenzklassen wird aus den Indexstermen von d der Vektor **pf** der relativen Frequenzen erstellt. Eine Präfixäquivalenzklasse umfasst alle Wörter bzw. Indexterme, die mit demselben Präfix beginnen. Der Präfix, nach dem die Indexterme klassifiziert werden, kann beispielsweise der Anfangsbuchstabe sein. Auch eine Folge von Buchstaben ist als Präfix möglich. Bei Stein (2005) liegt k im Normalfall zwischen 10 und 30.
3. Der Vektor **pf** wird anhand eines Referenzkorpus normalisiert. Der Vektor Δ**pf** wird berechnet und enthält die Abweichungen der Frequenzen in **pf** von der erwarteten Frequenzverteilung des Referenzkorpus.
4. Δ**pf** wird anschließend basierend auf einem Fuzzifizierungsschema ρ abstrahiert. Ein Fuzzifizierungsschema besteht aus zwei oder drei Intervallen. Jedes Element des Vektors Δ**pf** fällt in eins der Intervalle ρ_i des Schemas ρ.

Für ein gegebenes Fuzzifizierungsschema ρ mit r Intervallen, wie in Abbildung 3.5 veranschaulicht wird, lässt sich der Hashwert $h_\varphi^{(\rho)}$ von $\Delta\mathbf{pf}$ mit Formel 3.10 berechnen.

$$h_\varphi^{(\rho)}(d) = \sum_{i=0}^{k-1} \mu_i^{(\rho)} \cdot r^i \text{ mit } \mu_i^{(\rho)} \in \{0,..,r-1\} \tag{3.10}$$

Dabei gibt $\mu_i^{(\rho)}$ das Intervall an, in welches das Element $\Delta\mathbf{pf}_i$ unter Anwendung des Fuzzifizierungsschemas ρ fällt.

Abbildung 3.6 fasst die Berechnung des Fuzzy-Fingerprint schematisch zusammen.

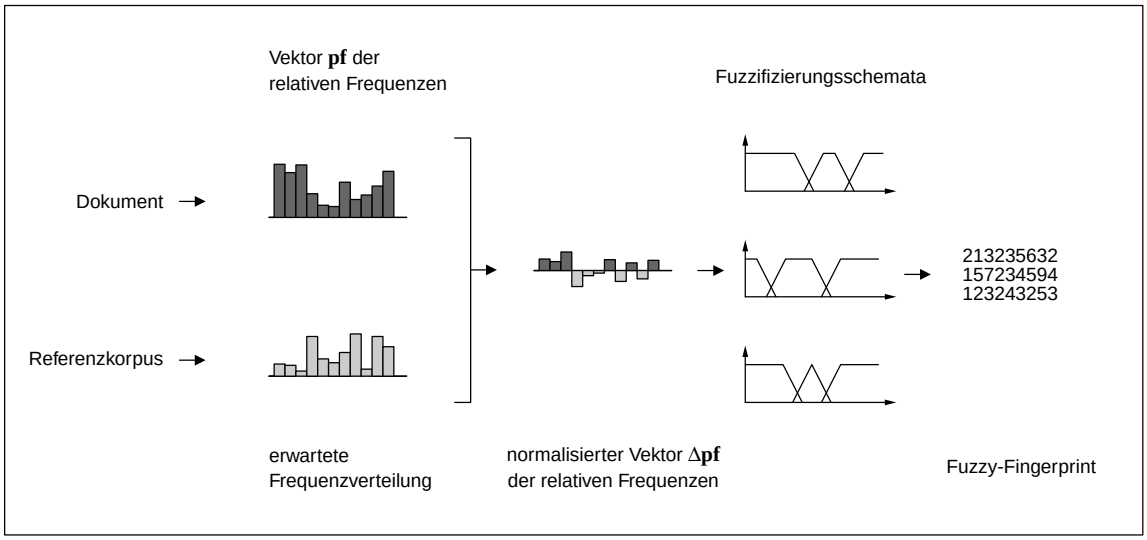


Abb. 3.6: Überblick über die Generierung des Fuzzy-Fingerprints für ein Dokument nach Stein (2005)

Die berechneten Hashwerte $h_\varphi^{(\rho)}(d)$ für alle $d \in D$ werden mit einer Standardhashfunktion auf die Speicherstellen einer Hashtabelle T_ρ abgebildet, die die Referenzen auf die entsprechenden Dokumente d enthält (Stein und Potthast (2006)).

Eine Hashfunktion $h_\varphi^{(\rho)}$ partitioniert den Raum, in dem sich die Dokumente befinden, in diskrete Regionen, wie in Abbildung 3.7 ersichtlich ist. Offenbar kann es vorkommen, dass sich zwei Dokumente a und b in einer kleinen ε-Umgebung ähnlich sind und dennoch zwei

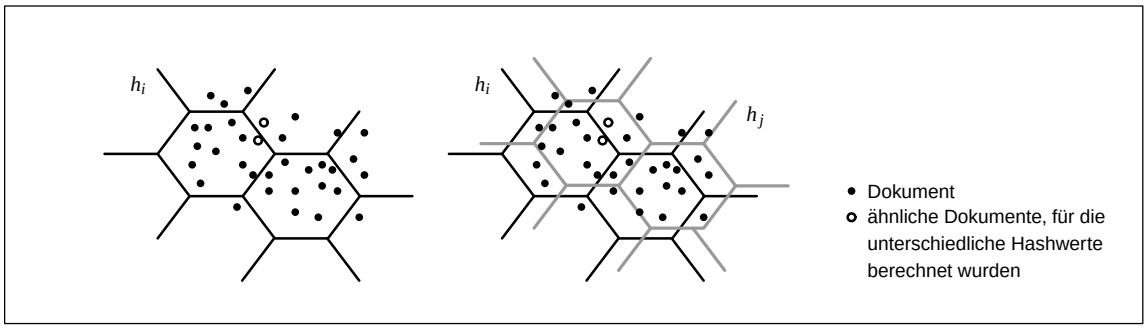


Abb. 3.7: Partitionierung des Dokumentenraums durch eine Hashfunktion in *ähnlich* und *nicht ähnlich* (in Anlehnung an Stein und Potthast (2006)). Die Verwendung mehrerer Hashfunktionen kompensiert den Effekt, dass auch für ähnliche Dokumente eventuell unterschiedliche Hashwerte berechnet werden.

unterschiedliche Hashwerte $h_\varphi(a)$ und $h_\varphi(b)$ berechnet werden. Um diesen Effekt zu reduzieren, werden üblicherweise zwei oder drei Hashfunktionen mit unterschiedlichen Fuzzifizierungsschemata ρ angewendet (Stein und Potthast (2006)). Bei einer Suche nach dem nächsten Nachbarn zum Anfragedokument q wird für jede Hashfunktion $h_\varphi^{(\rho)}$ der Hashwert berechnet und das entsprechende Bucket $T_\rho[h_\varphi^{(\rho)}(q)]$ einer Hashtabelle T_ρ zurückgegeben. Im Normalfall gibt jede Hashfunktion ein Bucket zurück. Alle gefundenen Buckets bilden zusammen die Ergebnismenge. Der Fuzzy-Fingerprint eines Dokumentes d besteht somit aus einer Menge von mehreren Hashwerten.

Die Verwendung mehrerer Fuzzifizierungsschemata ρ liefert für eine Suche nach den nächsten Nachbarn für ein Anfragedokument q möglicherweise mehrere relevante Dokumente als Ergebnis. Gleichzeitig können sich aber auch irrelevante Dokumente in der Ergebnismenge befinden. Die Parameter, mit denen das Fuzzifizierungsschema ρ festgelegt wird, haben großen Einfluss auf die Granularität der Partitionierung des Dokumentenraums durch die Hashwerte. Werden die Intervalle des Fuzzifizierungsschemas zu klein gewählt, wird ε in der Formel 3.9 ebenfalls sehr klein. Daraus resultiert, das für jedes Dokument ein individueller Hashwert berechnet wird und jedes Bucket der Hashtabelle folglich nur ein Dokument enthält. Sind die Intervalle allerdings zu groß, stimmen die Hashwerte von zu vielen Dokumenten überein, so dass zu viele unähnliche Dokumente gefunden werden. Die Parameter des Fuzzifizierungsschemas müssen mit großer Sorgfalt und angepasst an die Dokumente gewählt werden, damit diese sich möglichst gleichmäßig über die Buckets

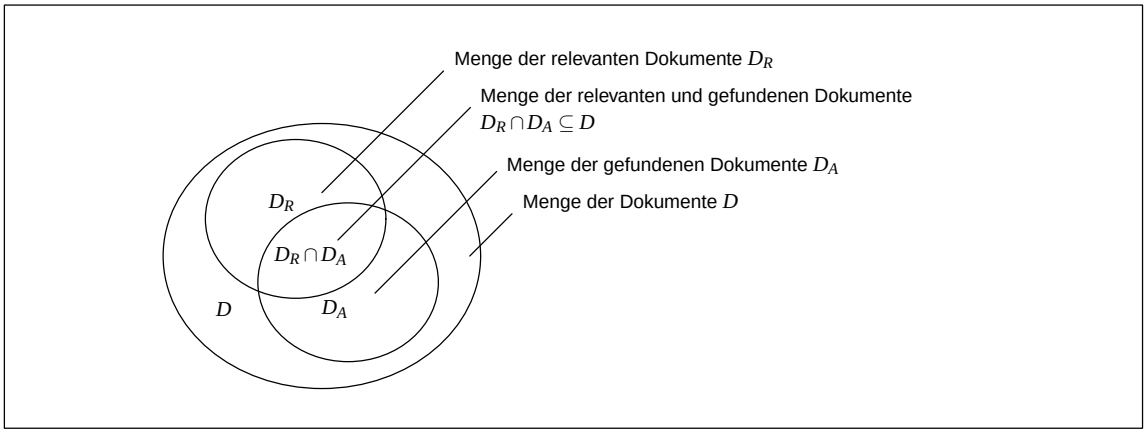


Abb. 3.8: Mengendiagramm der relevanten und gefundenen Dokumente der Kollektion bei der Suche mit einem Anfragedokument (nach Potthast (2006))

der Hashtabelle verteilen. Ein Maß für die Qualität des Similarity-Hashings bieten dabei die statistischen Größen *Precision* und *Recall*.

$$Precision = \frac{|D_R \cap D_A|}{|D_A|} \tag{3.11}$$

$$Recall = \frac{|D_R \cap D_A|}{|D_R|} \tag{3.12}$$

D_R bezeichnet dabei für ein Anfragedokument q die Menge der relevanten Dokumente in D. D_A ist die Menge aller gefundenen Dokumente für die Anfrage q. Abbildung 3.8 stellt dies als Mengendiagramm dar. Der *Recall* ist folglich der Teil der relevanten Dokumente, die gefunden wurden und die *Precision* der Teil der gefunden Dokumente, die relevant sind (Baeza-Yates und Ribeiro-Neto (1999)). Im Idealfall sollten beide Größen einen Wert um 1 besitzen. In der Praxis weisen sie allerdings ein umgekehrt proportionales Verhältnis auf (Ferber (2003)). Das heißt, bei einem hohen *Recall* ist die *Precision* eher niedrig. Entsprechend ist der *Recall* bei hoher *Precision* gering.

Hybride multidimensionale Skalierung mit Fuzzy-Fingerprinting

Durch die Anwendung des Fuzzy-Fingerprintings zur Bestimmung des nächsten Nachbarn n reduziert sich die Komplexität dieses Schrittes. Für das zu interpolierende Objekt i muss der Fuzzy-Fingerprint des Dokuments i berechnet werden. Die Hashwerte $h_\varphi^{(\rho)}(i)$ werden mit den entsprechenden Hashtabellen T, welche die Teilmenge S enthalten, verglichen. Das heißt für jede Hashfunktion $h_\varphi^{(\rho)}$ wird das Bucket $T_\rho[h_\varphi^{(\rho)}(i)]$ aus der Hashtabelle T_ρ zurückgegeben. Alle so in den Hashtabellen gefundenen Buckets bilden zusammen die Ergebnismenge. Wurden keine Ergebnisse gefunden, weil kein Dokument der Hashtabellen denselben Hashwert hat, wird zufällig ein Bucket je Hashfunktion ausgewählt. Aus der Ergebnismenge werden anschließend t Dokumente ausgewählt. Von den t Dokumenten wird mit einer linearen Suche der exakte nächste Nachbar bestimmt. Algorithmus 8 verdeutlicht den Ablauf mit Hilfe von Pseudocode.

Das Verfahren mit Fuzzy-Fingerprinting wurde bei der Evaluation in zwei Varianten angewendet. Bei der ersten Variante wird der Fuzzy-Fingerprint des Objekts i nach der Interpolation zur Menge der Fuzzy-Fingerprintings der Teilmenge S hinzugefügt um das Ergebnis der Nächsten-Nachbar-Suche zu verbessern. So kann das bereits interpolierte Objekt i ebenfalls als nächster Nachbar für ein anderes Objekt fungieren. Jedoch wurde die Interpolation mit der ursprünglichen Teilmenge S ausgeführt um die Laufzeit dieser Phase nicht zu verlängern. Bei der zweiten Variante wird sowohl für die Nächste-Nachbar-Suche als auch für die Interpolation die ursprüngliche Teilmenge S angewendet.

Die lineare Suche in der Stichprobe ist nicht notwendig, wenn sowohl der *Recall* als auch die *Precision* des Fuzzy-Fingerprintings sehr hoch sind. Dann könnte zufällig ein Dokument aus der Ergebnismenge gewählt werden. In der Praxis ist die Wahrscheinlichkeit, dass ein Dokument der Ergebnismenge D_A, das nicht zum relevanten Teil D_R der Datenmenge D dieser Anfrage gehört, recht hoch. Deshalb soll die lineare Suche bei der begrenzten Anzahl t den besten Kandidaten finden. Um die Effizienz zu steigern, können die Fuzzy-Fingerprints, wie die Wortvektoren der gesamten Dokumentenkollektion vorberechnet werden. So hat die Nächste-Nachbar-Suche mit Fuzzy-Fingerprinting eine allgemeine Komplexität von $O(N)$, da nur ein Vergleich der gesuchten Hashwerte mit den

Hashtabellen notwendig ist. Die Suche über die Stichprobe t fällt als konstanter Anteil weg, da t unabhängig von N gewählt wird. Damit ist die Nächste-Nachbar-Suche in linearer Zeit möglich. Hier zeigt sich deutlich, dass der aufwendigste Schritt auch in diesem Verfahren die Interpolation mit $O(N\sqrt{N})$ ist und die allgemeine Komplexität bestimmt.

Eingabe : Menge D von Datenobjekten ($i, j \in D$) der Größe $|D| := N$
Distanzmaß $\delta(\mathbf{i},\mathbf{j})$ für alle Objektpaare $(i, j) \in D^2$
Menge H der Hashfunktionen des Fuzzy-Fingerprintings
Ausgabe : $\forall s \in S \; \exists$ Punktkoordinaten p_s

```
1  begin
2      generiere die Punktkonfiguration der Teilmenge S mit dem Verfahren von Chalmers (1996);
3      forall i ∈ D\S do
4          E := {s ∈ S | ∃ h_φ ∈ H : h_φ(i) = h_φ(s)};
5          if E = ∅ then
6              E := zufälliges Bucket ∀ h_φ ∈ H;
7          wähle eine kleine Stichprobe P aus E;
8          n := ∅;
9          forall p ∈ P do
10             if δ(i,p) < δ(i,n) then
11                 n := p;
12         interpoliere Koordinaten für p_i mit Funktion (4): interpolate(i,n,δ(i,n));
13     verfeinere die Punktkonfiguration mit dem Verfahren von Chalmers (1996) über alle
       Objekte i ∈ D;
14 end
```

Algorithmus 8 : Verfahren mit Fuzzy-Fingerprinting von Stein (2005)

Die Tabelle 3.1 gibt einen zusammenfassenden Überblick über die Komplexitäten der behandelten Verfahren. Morrison und Chalmers (2003) und Jourdan und Melançon (2004) sehen den aufwendigsten Schritt in der Nächsten-Nachbar-Suche. Offensichtlich stellt allerdings die Interpolation den zeitaufwendigsten Schritt des hybriden Ansatzes dar.

	Komplexitäten:	
Betrachtetes Verfahren:	Nächste-Nachbar-Suche	Gesamtkomplexität
Spring-Modell	-	$O(N^3)$
Verfahren von Chalmers (1996)	-	$O(N^2)$
Verfahren von Chalmers u. a. (2003)	$O(N\sqrt{N})$	$O(N\sqrt{N})$
Verfahren von Jourdan und Melançon (2004)	$O(N^{5/4}log(N))$	$O(N\sqrt{N})$
Multiscale-Verfahren von Jourdan und Melançon (2004)	$O(N^{5/4}log(N))$	$O(N\sqrt{N})$
Verfahren mit Fuzzy-Fingerprinting	$O(N)$	$O(N\sqrt{N})$

Tabelle 3.1: Komplexitäten der betrachteten Verfahren

4 Evaluation

In diesem Kapitel werden die Testkollektionen und Voraussetzungen der Evaluation dargestellt. Die Ergebnisse der Evaluation der betrachteten Verfahren werden vorgestellt und ausgewertet.

4.1 Testbedingungen

4.1.1 Wahl der Kollektionen

Die Verfahren werden mit Dokumentenkollektionen des Reuters-Korpus evaluiert. Dieser Korpus ist eine Sammlung von ca. 800 000 Dokumenten der Nachrichtenagentur Reuters. Die Nachrichtendokumente wurden hierarchisch in vier Hauptkategorien und verfeinernde Unterkategorien klassifiziert. Außerdem wurden sie um Metainformationen wie Region, Themengebiete und Industriesektor erweitert. Alle Dokumente des Reuters-Korpus Volume 1 sind in englischer Sprache verfasst und stammen aus dem Zeitraum von 1996 bis 1997. Die Dokumentgröße reicht von ein paar hundert bis einigen tausend Wörtern. Ein Dokument kann mehreren Kategorien aus verschiedenen Ebenen der Klassifizierungshierarchie des Korpus angehören. Für die Evaluation wurde festgelegt, dass zwei Dokumente zu einer Kategorie gehören, wenn sie sowohl in der Hauptkategorie als auch in der niedrigsten Ebene der Kategorisierung überein stimmen. Unterkategorien einer Kategorie, aus

der Dokumente für eine Kollektion ausgewählt worden, wurden nicht verwendet. Die Kollektionen für die Evaluation (Tabelle 4.1) wurden so gewählt, dass möglichst kein Dokument mehrfach in einer Kollektion auftritt. Lediglich in den 80 000 und 100 000 Objekte umfassenden Kollektionen treten maximal 0,15% der Dokumente mehr als einmal in der Kollektion auf.

Größe:	**Kategorien:**	**Dimension:**	**Größe der Teilmenge ($\sqrt{N}$):**
1000	3	6 992	32
10000	5	27 612	100
40000	8	52 420	200
60000	3	105 976	245
80000	12	94 798	283
100000	12	124 093	316

Tabelle 4.1: Daten der Evaluationskollektionen

Die Kategorien des Reuters-Korpus unterscheiden sich thematisch. Deshalb ist anzunehmen, dass sich dies in einer entsprechenden Clusterung der Punktkonfiguration zeigen wird. Allerdings können die Dokumente einer Kategorie trotz derselben Thematik sehr unähnlich sein. Um eine minimale Clusterung der Punktkonfiguration zu sichern, wurde eine Vorauswahl der Dokumente einer Kategorie getroffen. Das heißt, es wurden nur Dokumente aus einer Kategorie verwendet, deren Kosinus-Ähnlichkeit untereinander mindestens γ betrug. Je nach Größe der betrachteten Kategorie nahm γ einen Wert zwischen 0,2 und 0,5 an. Die zwei größten Kollektionen wurden aus mehreren kleineren Kategorien zusammengestellt, auf der Vermutung, das damit mehr kleinere Cluster in der Punktkonfiguration entstehen würden.

4.1.2 Bestimmung der Parameter

Kriterien zum Abbruch des Spring-Modells und des Verfahrens von Chalmers (1996) können zum Beispiel der durchschnittliche Betrag der Geschwindigkeitsvektoren des Systems zwischen den Iterationen oder der Stresswert sein, wie sie Chalmers u. a. (2003) verwenden. Allerdings könnten die Verfahren bei diesen Abbruchkriterien prinzipiell auch deutlich mehr als N Iterationen berechnen, da die durchschnittliche Geschwindigkeit des Systems nicht unter die festgelegte Schranke fallen muss. Um die Vergleichbarkeit des

Spring-Modells und des Verfahrens von Chalmers (1996) zu verbessern, wurde bei diesen Methoden die Anzahl der Iterationen mit N als Abbruchkriterium genutzt. Ebenfalls wurde die Punktkonfiguration der Teilmenge S der hybriden Verfahren mit $\sqrt{N}$ Iterationen generiert.

Die Parameter der Kräfteberechnung sind in der Tabelle 4.2 aufgeführt. Die Größen der Teilmengen S_i und V_i beim Verfahren von Chalmers (1996) wurden bei den hybriden Verfahren zur Generierung der Punktkonfiguration der Teilmenge S mit $S_{max} = 10$ und $V_{max} = 5$ beibehalten. Für den Durchlauf des Verfahrens von Chalmers (1996) auf der ersten Kollektion mit 1 000 Dokumenten wurden sie allerdings auf $S_{max} = 15$ und $V_{max} = 7$ erhöht. Mit den ursprünglichen Größen der Teilmengen hätte dieses Verfahren für 1 000 Dokumente mehr als N Iterationen für die Generierung einer vergleichbaren Punktkonfiguration benötigt.

	Verfahren:		
Parameter:	Spring-Modell	Chalmers (1996)	Standard
k_s	0,6	0,7	0,7
k_d	0,2	0,2	0,2
α	0,5	0,8	0,8
Δt	0,5	0,5	0,5
V_{max}	-	7	5
S_{max}	-	15	10
Anzahl Iterationen zur Positionierung (Interpolation)	-	-	20
Anzahl Stichproben des Fuzzy-Fingerprintings	-	-	4
Anzahl Iterationen für Verfeinerung	-	-	30

Tabelle 4.2: Parameter der Verfahren

Für die Verfeinerung der Punktkonfiguration wurden 30 Iterationen des Verfahrens von Chalmers (1996) auf der ganzen Datenmenge D berechnet.

Die Wahl der Parameter für das Fuzzy-Fingerprinting orientierte sich an den von Potthast (2006) verwendeten Werten. Insgesamt wurden drei Fuzzifizierungsschemata ρ mit jeweils drei Intervallen angewendet: (0,9;1,5), (1,2;1,7), (1,6;2,1). In Abbildung 4.1 sind die durchschnittlichen Werte von *Precision* und *Recall* in Abhängigkeit der Ähnlichkeit dargestellt. Die Werte wurden mit Testkollektionen des Reuters-Korpus berechnet.

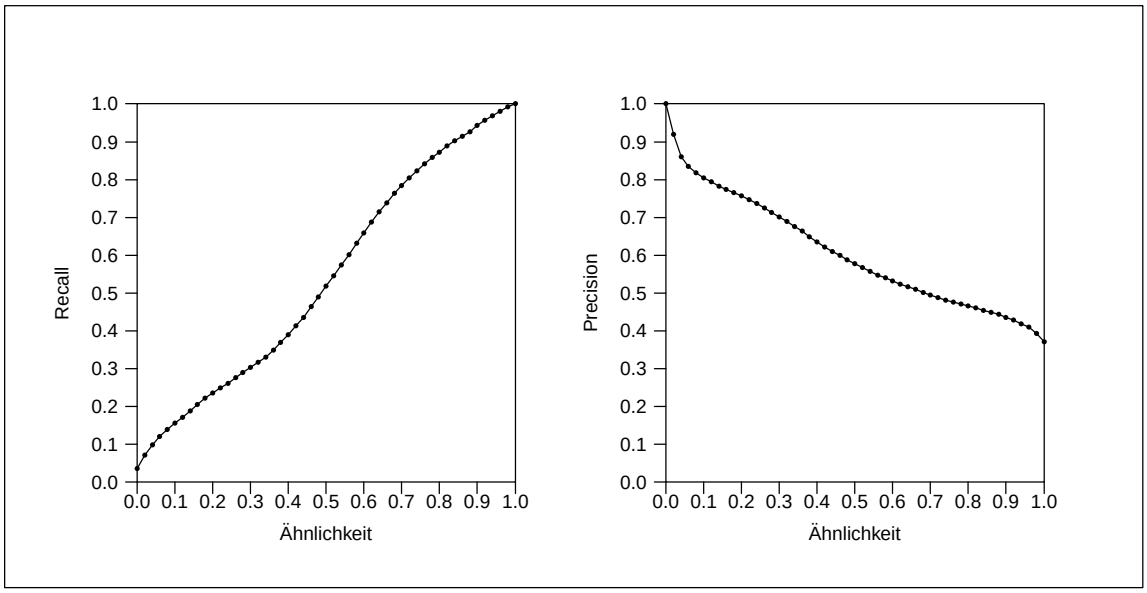


Abb. 4.1: Recall- und Precisionwerte in Abhängigkeit von festgelegten Ähnlichkeitsschwellwerten

4.1.3 Implementierung der Verfahren

Da das FSMvis[1]-Tool, welches Chalmers u. a. (2003) für ihre Experimente entwickelten und benutzen, auf Grund seiner Ableitungsstuktur der Klassen nur sehr schlecht erweiterbar ist, wurden alle erwähnten Verfahren mit Java 5.0 neu implementiert. Der Quellcode dieser Implementierung befindet sich auf der beiliegenden CD.

Die Evaluation wurde auf einem Rechner mit Intel Pentium 4 CPU 3 GHz und 2 GB RAM durchgeführt.

Für die Erstellung der Plots wurde Gnuplot 4.0[2] verwendet.

In einem Vorverarbeitungsschritt wurden die Wortvektoren der Dokumente aller Kollektionen berechnet, mit denen die Evaluation durchgeführt wurde. Die Fuzzy-Fingerprintings

[1] http://www.dcs.gla.ac.uk/~matthew/papers/FSMvis.tar.gz (letzter Zugriff: 30.01.2007)
[2] http://www.gnuplot.info (letzter Zugriff: 30.01.2007)

wurden ebenfalls vorher errechnet, da es nicht möglich ist alle Dokumente gleichzeitig im Hauptspeicher zu halten.

4.2 Analyse

Für jede Kollektion wurde in Anlehnung an die Experimente von Morrison und Chalmers (2003) 6 Durchläufe je Verfahren durchgeführt. Die Stresswerte wurden für alle so erhaltenen Plots berechnet. Anschließend wurden für die Evaluationsergebnisse je Kollektion und Verfahren die Mittelwerte berechnet, auf die im folgenden Bezug genommen wird. Das Spring-Modell und das Verfahren von Chalmers (1996) wurde auf Grund der langen Laufzeiten nur für die erste Kollektion mit 1 000 Dokumenten angewendet. Wie bereits in Abschnitt 3.3.1 angesprochen, wurde die Interpolation auf Basis der gesamten Teilmenge *S* ausgeführt.

4.2.1 Laufzeiten der Verfahren

Die Laufzeiten der Verfahren sind in Tabelle 4.3 angegeben. In Abbildung 4.2 wird ersichtlich, dass erst ab 40 000 Objekten Unterschiede zwischen den Laufzeiten der verschiedenen Verfahren heraustreten. Die erste Variante des Verfahrens auf Basis des Fuzzy-Fingerprintings, bei dem die Fuzzy-Fingerprints der interpolierten Objekte bei der Nächsten-Nachbar-Suche berücksichtigt wurden, zeigt nur eine geringe Verbesserung des Laufzeitverhaltens. Die zweite Variante weist hingegen eine deutlich kürzere Laufzeit auf. Allerdings beträgt die maximale Zeitdifferenz der Verfahren nur ca. 3 Minuten zwischen dem Verfahren von Jourdan und Melançon (2004) und der zweiten Variante mit Fuzzy-Fingerprintings bei 100 000 Objekten. Ob dieser - gemessen an der Berechnungsdauer insgesamt - geringe Unterschied in den Laufzeiten eine wesentliche Verbesserung des Verfahrens darstellt, muss offen bleiben. Die Beobachtung von Jourdan und Melançon (2004),

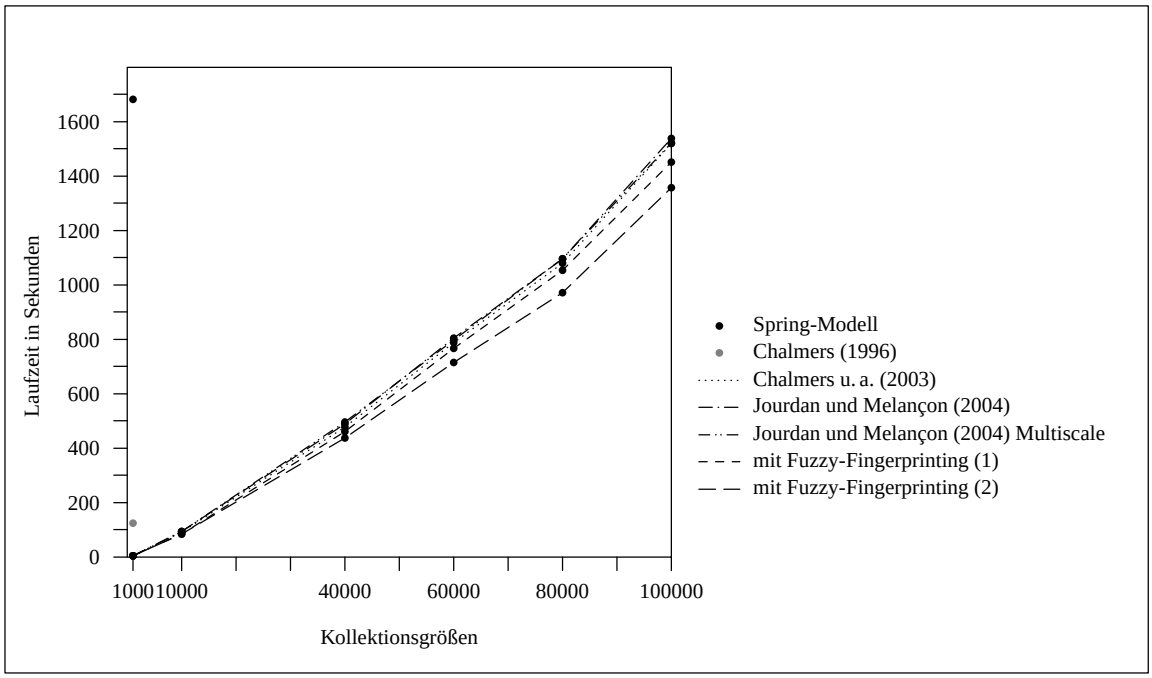


Abb. 4.2: Gesamtlaufzeiten der Verfahren

dass der Multiscale-Ansatz eine signifikante Laufzeitverbesserung darstellt, hat sich nicht bestätigt.

	Kollektionsgrößen:					
Verfahren:	1000	10000	40000	60000	80000	100000
Spring-Modell	1682,78					
Chalmers (1996)	125,74					
Chalmers u. a. (2003)	4,79	91,29	478,27	788,40	1080,34	1518,61
Jourdan und Melançon (2004)	4,86	93,66	488,42	804,18	1094,89	1538,75
Jourdan und Melançon (2004) Multiscale	5,05	94,75	495,39	797,71	1094,90	1552,34
mit Fuzzy-Fingerprinting (1)	4,66	85,49	461,68	766,01	1054,13	1451,28
mit Fuzzy-Fingerprinting (2)	4,62	84,71	437,09	714,43	970,49	1357,17

Tabelle 4.3: Mittelwerte der Gesamtlaufzeiten in Sekunden über 6 Durchläufe je Verfahren

Abbildung 4.3 macht deutlich, dass die Nächste-Nachbar-Suche auf Basis von Fuzzy-Fingerprinting sehr viel schnellere Laufzeiten aufweisen kann. Allerdings gilt dies nur bei der zweiten Variante dieses Verfahrens. Trotzdem liegt die Nächste-Nachbar-Suche der ersten Variante des Fuzzy-Fingerprintings insgesamt unter den Laufzeiten der Nächsten-Nachbar-Suche von Chalmers u. a. (2003) sowie Jourdan und Melançon (2004). Das die erste Variante nur wenig unter den Laufzeiten der anderen Verfahren liegt, ist vermutlich

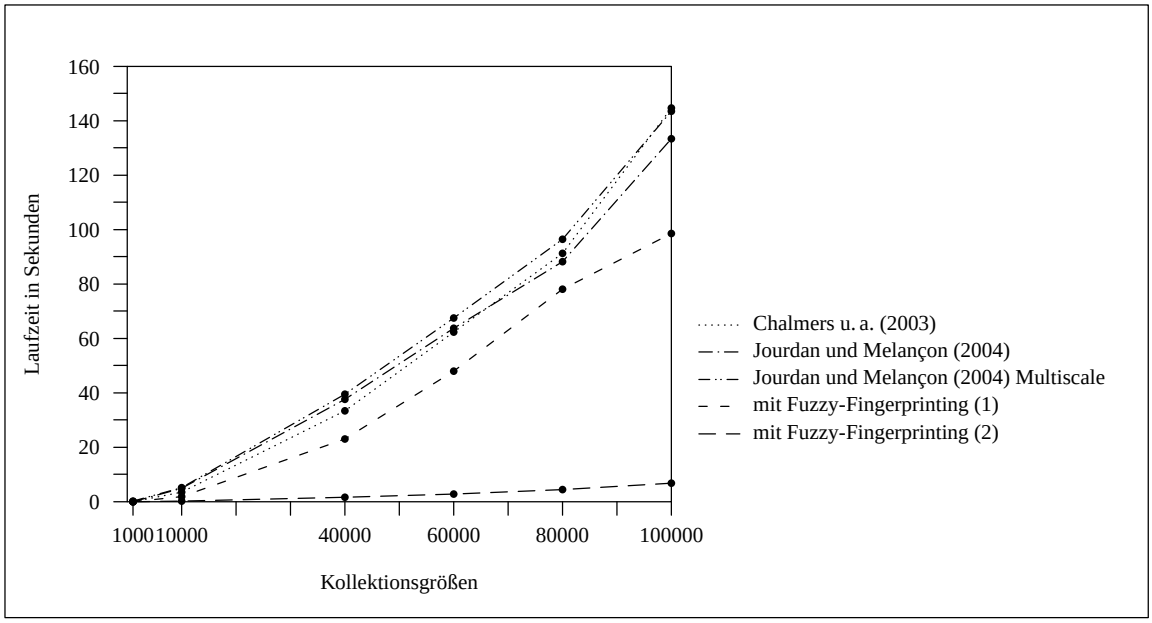


Abb. 4.3: Laufzeiten der Nächsten-Nachbar-Suche der Verfahren

ein implementationsabhängiger Effekt. Jedoch würde die Nächste-Nachbar-Suche der ersten Variante auch bei einer besseren Implementierung die Laufzeit der zweiten Variante nicht übertreffen. Die Laufzeiten der Nächsten-Nachbar-Suche sind in Tabelle 4.4 aufgeführt. Die Vorteile der Nächsten-Nachbar-Suche mittels Pivotobjekten gegenüber einer linearen Suche wie bei Chalmers u. a. (2003) treten erst bei der 80 000 Objekte umfassenden Kollektion auf. Ein ähnliches Verhalten beobachteten auch Jourdan und Melançon (2004) bei ihren Experimenten.

	Kollektionsgrößen:					
Verfahren:	1000	10000	40000	60000	80000	100000
Chalmers u. a. (2003)	0,07	3,56	33,32	62,24	91,22	144,72
Jourdan und Melançon (2004)	0,11	5,05	37,73	63,66	88,13	133,30
Jourdan und Melançon (2004) Multiscale	0,15	5,12	39,46	67,44	96,36	143,53
mit Fuzzy-Fingerprinting (1)	0,03	1,90	23,05	47,98	78,08	98,46
mit Fuzzy-Fingerprinting (2)	0,02	0,33	1,72	2,80	4,54	6,72

Tabelle 4.4: Mittelwerte der Laufzeiten der Nächsten-Nachbar-Suche in Sekunden über 6 Durchläufe je Verfahren

Dasselbe Verhalten, das in Abbildung 4.3 auftritt, fällt auch in Tabelle 4.5 auf. Die zweite Variante des Verfahrens mit Fuzzy-Fingerprinting ist offensichtlich bei der Interpolati-

	Kollektionsgrößen:					
Verfahren:	1000	10000	40000	60000	80000	100000
Chalmers u. a. (2003)	0,90	29,97	203,10	368,73	541,98	793,73
Jourdan und Melançon (2004)	0,94	32,22	213,16	383,76	557,48	812,96
Jourdan und Melançon (2004) Multiscale	1,10	35,18	227,77	391,54	576,28	822,43
mit Fuzzy-Fingerprinting (1)	0,78	24,47	191,15	352,84	525,51	739,02
mit Fuzzy-Fingerprinting (2)	0,77	23,23	167,70	303,85	445,21	644,64

Tabelle 4.5: Mittelwerte der Interpolationszeiten in Sekunden über 6 Durchläufe je Verfahren

on ebenfalls etwas schneller. Insgesamt unterscheiden sich die Verfahren bei der Berechnungsdauer des Interpolationsteils nur unwesentlich. Die maximale Laufzeitdifferenz des Interpolationsteils zwischen den verschiedenen Verfahren beträgt fast 3 Minuten, stellt allerdings im Hinblick auf die Gesamtlaufzeiten keinen signifikanten Unterschied dar.

In Tabelle 4.6 sind die Zeiten angegeben, die für die Iterationen zur Verfeinerung der gesamten Punktkonfiguration mit dem Verfahren von Chalmers (1996) benötigt wurden. Alle Verfahren benötigen logischerweise für diesen Schritt auf denselben Daten auch dieselbe Zeit. Insgesamt wird ungefähr die Hälfte der Gesamtlaufzeit für die Iterationen zur Verfeinerung der Punktkonfiguration gebraucht.

	Kollektionsgrößen:					
Verfahren:	1000	10000	40000	60000	80000	100000
Chalmers u. a. (2003)	3,80	60,09	269,79	411,55	527,95	710,72
Jourdan und Melançon (2004)	3,85	60,23	269,97	412,40	526,83	711,62
Jourdan und Melançon (2004) Multiscale	3,93	59,47	267,37	405,86	518,23	699,44
mit Fuzzy-Fingerprinting (1)	3,79	59,77	265,34	405,27	518,42	698,60
mit Fuzzy-Fingerprinting (2)	3,78	60,22	264,15	402,66	515,18	698,83

Tabelle 4.6: Mittelwerte der Laufzeiten der Verfeinerung in Sekunden über 6 Durchläufe je Verfahren

	Kollektionsgrößen:					
Verfahren:	1000	10000	40000	60000	80000	100000
Chalmers u. a. (2003)	0,09	1,21	5,33	8,04	10,33	14,04
Jourdan und Melançon (2004)	0,08	1,21	5,24	7,96	10,49	14,07
Jourdan und Melançon (2004) Multiscale	0,02	0,08	0,20	0,24	0,28	0,38
mit Fuzzy-Fingerprinting (1)	0,09	1,24	5,13	7,83	10,11	13,54
mit Fuzzy-Fingerprinting (2)	0,08	1,23	5,20	7,85	10,01	13,61

Tabelle 4.7: Mittelwerte der Laufzeiten der Generierung der Teilmengenkonfiguration in Sekunden über 6 Durchläufe je Verfahren

Interessant ist die Tatsache, dass der Multiscale-Ansatz von Jourdan und Melançon (2004) deutlich schneller bei der Generierung der Punktkonfiguration der Teilmenge *S* ist (Tabelle 4.7). Die übrigen Verfahren wenden für diesen Schritt das Verfahren von Chalmers (1996) an und weisen deshalb nahezu die gleichen Laufzeiten für diese Phase auf.

4.2.2 Ergebnisse der Stressberechnung

Abbildung 4.4 fasst die Ergebnisse der Stressberechnung zusammen. Auffällig ist das Verhalten, welches das Multiscale-Verfahren von Jourdan und Melançon (2004) aufweist. Jourdan und Melançon (2004) betonen, das der Multiscale-Ansatz bei ihren Experimenten sehr viel niedrigere Stresswerte ($\sigma < 0,2$) lieferte. In diesem Fall hat er jedoch die schlechtesten Ergebnisse gezeigt. Erwartungsgemäß liefert das Spring-Modell bei der Kollektion mit 1 000 Objekten die besten Ergebnisse der Stressberechnung mit durchschnittlich $\sigma = 0,10$. Auch das Verfahren von Chalmers (1996) liefert einen ähnlich guten Wert. Die Stresswerte der übrigen Verfahren sind bei der ersten Kollektion etwas gestreut, allerdings sind alle $\sigma < 0,2$. In der Tat sind die Ergebnisse der Stressberechnung für die Kollektionen sehr stabil, ganz im Gegensatz zu den Ergebnissen von Jourdan und Melançon (2004). In Tabelle 4.8 sind die exakten Ergebnisse der Stress-Berechnung aufgelistet.

Neben den Verfahren wurde noch ein Test gemacht, inwiefern die Nächste-Nachbar-Suche überhaupt einen Einfluss auf die Interpolation und somit den Stresswert hat. Bei diesem Test wurde, anstatt eine Nächste-Nachbar-Suche auszuführen, zufällig ein Objekt der Teilmenge *S* als nächster Nachbar ausgewählt. Der Stresswert zeigt, das die Nächste-Nachbar-Suche für die Interpolation durchaus einen positiven Effekt auf den Stress der Punktkonfiguration hat.

Die Abbildungen 4.5 und 4.6 zeigen die 2D-Plots der Punktkonfigurationen, die in einem Durchlauf je Verfahren entstanden sind. Die Punktarten zeigen die verschiedenen Kategorien an. In der Legende neben den 2D-Plots sind die Kategorieabkürzungen des Reuters-Korpus aufgeführt.

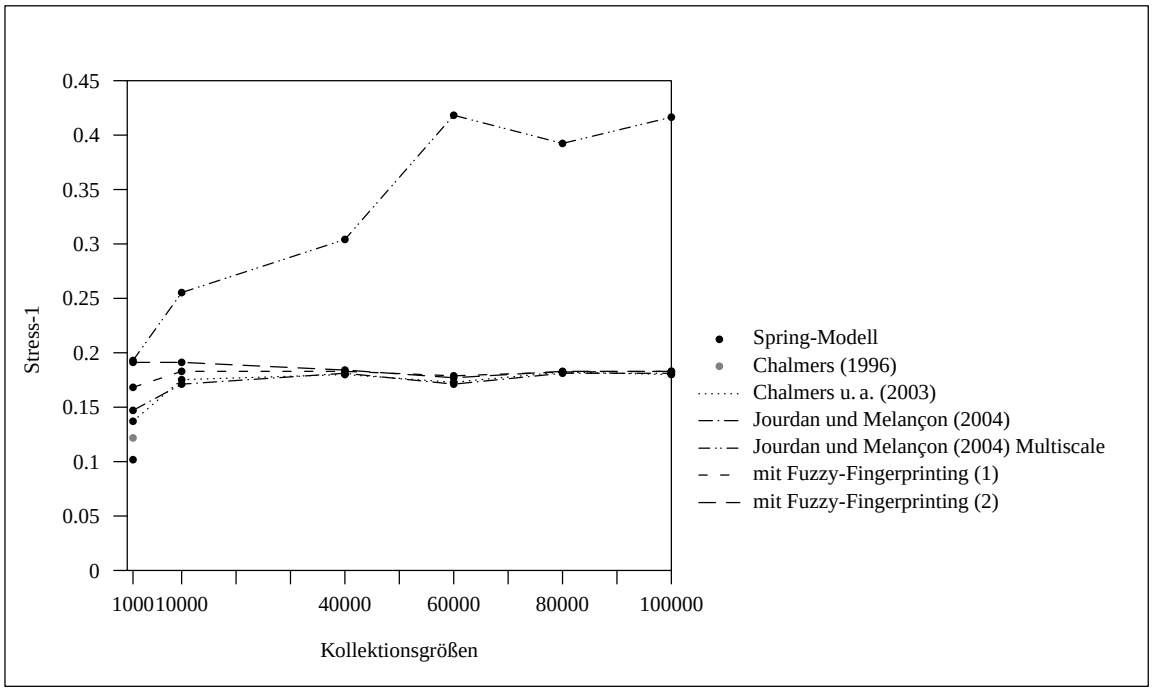
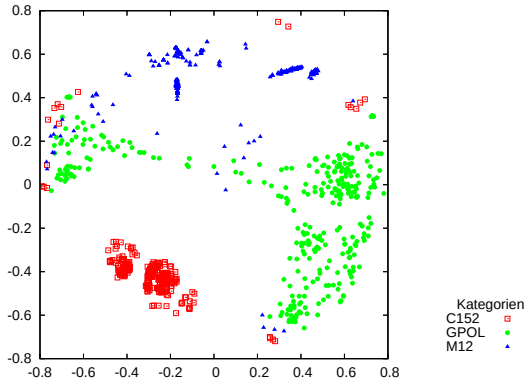


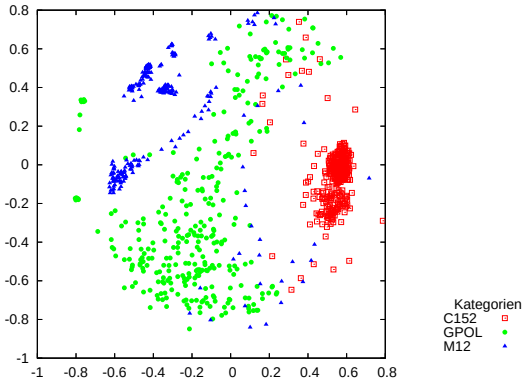
Abb. 4.4: Stresswerte der Verfahren

Deutlich zu sehen ist, dass sowohl das Spring-Modell als auch das Verfahren von Chalmers (1996) die Punkte im Plot sehr viel stärker clustern, wodurch viele kleine, deutlich abgegrenzte Gruppen entstehen. Das Verfahren von Chalmers u. a. (2003) trennt die unterschiedlichen Gruppen ebenfalls sehr deutlich. Allerdings treten die Ähnlichkeitsunterschiede innerhalb einer Gruppe, wie sie bei dem Spring-Modell-Plot offensichtlich werden, nicht so stark hervor.

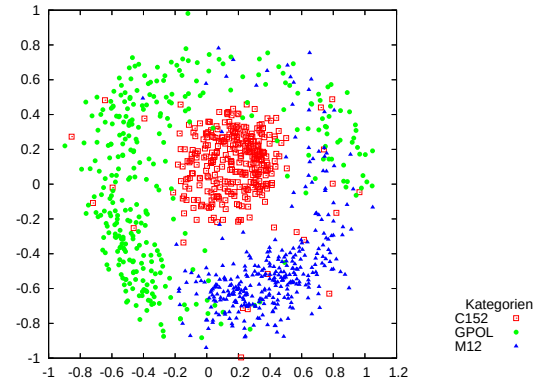
Die Verfahren von Jourdan und Melançon (2004) sowie die Verbesserung des Verfahrens mit Fuzzy-Fingerprinting erzeugen vergleichbar gute Clusterungen der Punkte. Die Gruppen sind weniger deutlich voneinander abgegrenzt, als bei dem Spring-Modell oder dem Verfahren von Chalmers (1996). Trotzdem lassen sich die wichtigsten Muster in der Datenmenge aus der Punktkonfiguration entnehmen. Bei den zwei Varianten des durch Fuzzy-Fingerprinting verbesserten Verfahrens ist ersichtlich, dass die Berücksichtigung der Fuzzy-Fingerprints der interpolierten Objekte bei der Nächsten-Nachbar-Suche direkt in einer exakteren Punktkonfiguration resultiert. Dies liegt darin begründet, dass die Wahrscheinlichkeit eine nicht leere Ergebnismenge zu erhalten mit der Anzahl der Fuzzy-



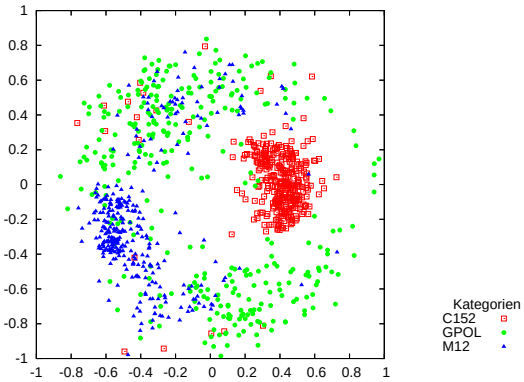
(a) Spring-Modell



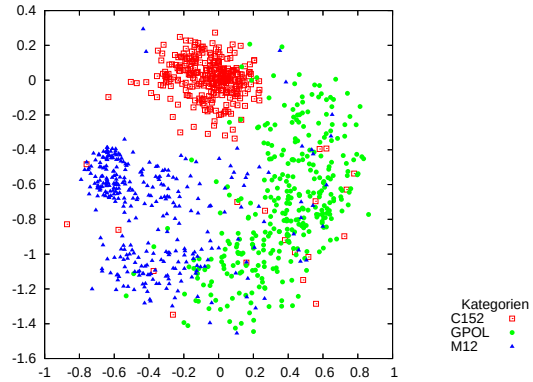
(b) Chalmers (1996)



(c) Chalmers u. a. (2003)

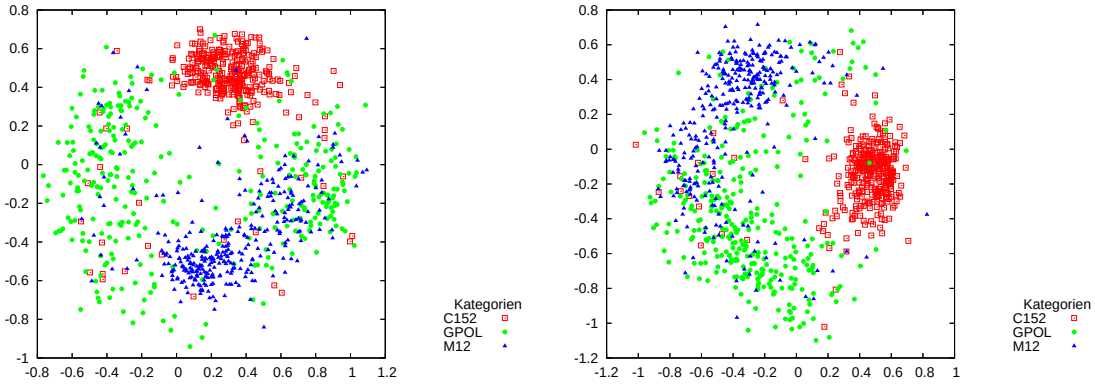


(d) Jourdan und Melançon (2004)



(e) Jourdan und Melançon (2004) Multiscale

Abb. 4.5: Punktkonfigurationen der Evaluationskollektion mit 1 000 Dokumenten



(a) mit Fuzzy-Fingerprinting (1) (b) mit Fuzzy-Fingerprinting (2)

Abb. 4.6: Punktkonfigurationen der Evaluationskollektion mit 1 000 Dokumenten

Fingerprints in der Hashtabelle steigt. Somit müssen bei der zweiten Variante sehr viel mehr leere Ergebnismengen durch eine zufällige Auswahl von Buckets aus den Hashtabellen kompensiert werden. Dadurch wird die Punktkonfiguration ungenauer im Vergleich zur ersten Variante.

Abbildung 4.7 gibt einen Überblick der Punktkonfigurationen mit 10 000 Objekten und 27 612 Dimensionen. Bei diesen Plots ist erkennbar, dass die wichtigsten Cluster entstanden sind. Allerdings können ab dieser Menge an Punkten kaum Aussagen über die Qualität der Punktkonfiguration getroffen werden, weshalb der Stresswert ein wichtiger Indikator dafür ist.

	Kollektionsgrößen:					
Verfahren:	1000	10000	40000	60000	80000	100000
Test mit zufällig gewähltem nächsten Nachbarn	0,222					
Spring-Modell	0,102					
Chalmers (1996)	0,122					
Chalmers u. a. (2003)	0,137	0,175	0,180	0,173	0,182	0,180
Jourdan und Melançon (2004)	0,147	0,171	0,181	0,171	0,181	0,181
Jourdan und Melançon (2004) Multiscale	0,193	0,255	0,304	0,418	0,392	0,416
mit Fuzzy-Fingerprinting (1)	0,168	0,183	0,183	0,179	0,182	0,183
mit Fuzzy-Fingerprinting (2)	0,189	0,191	0,184	0,177	0,183	0,183

Tabelle 4.8: Mittelwerte der Stresswerte über 6 Durchläufe je Verfahren

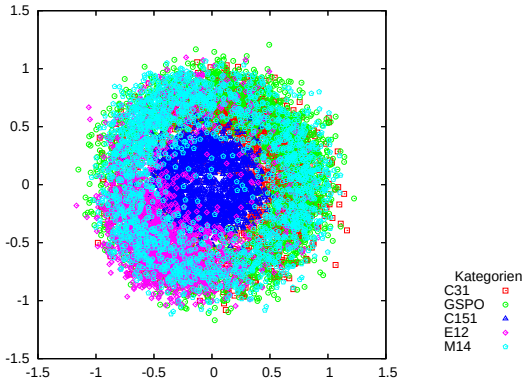
4.2.3 Bewertung der Ergebnisse

Die wichtigsten Kriterien zur Beurteilung der Verfahren sind die Ergebnisse der Stressberechnung sowie die Laufzeiten der Nächsten-Nachbar-Suche und des Verfahrens insgesamt. Die Laufzeiten der Interpolation und der Iteration zur Verfeinerung der Punktkonfiguration sind für alle Verfahren annähernd dieselben.

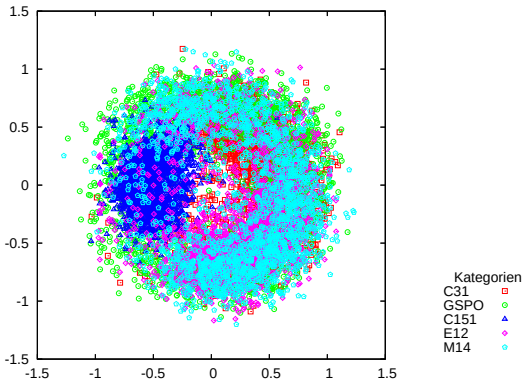
Tabelle 4.8 zeigt, dass der Multiscale-Ansatz von Jourdan und Melançon (2004) nur unzureichende Punktkonfigurationen generiert, obwohl in den dort beschriebenen Experimenten das Gegenteil beobachtet wurde. Möglicherweise liegt ein Fehler in der Beschreibung des Verfahrens vor, wie sie unter 3.3.4 gegeben wurde, da der in Jourdan und Melançon (2004) verwendete Ansatz dort nur oberflächlich dargestellt ist. Die Stresswerte der übrigen Verfahren liegen alle bei ca. 0,18. Chalmers u. a. (2003) erreichen mit ihrem hybriden Verfahren im Durchschnitt Stresswerte von 0,06. Dabei wurden jedoch Datenmengen mit maximal 13 Dimensionen verwendet, welches eine wesentlich niedrigere Dimensionalität darstellt als bei den in dieser Arbeit benutzten Dokumentenkollektionen.

Das die Stresswerte für die 10 000 und mehr Objekte umfassenden Kollektionen dicht um den Wert 0,18 liegen, spricht dafür, dass die Verfahren die gleiche Qualität bei der Generierung der Punktkonfiguration aufweisen. Das heißt, auch bei größeren Datenmengen können ähnlich Werte und folglich auch entsprechend *gute* Punktkonfigurationen erwartet werden. Das entscheidende Kriterium stellt somit die Laufzeit des Verfahrens dar. Schwerpunkt der Betrachtungen ist dabei die Nächste-Nachbar-Suche für die Interpolation. Die zweite Variante des Verfahrens mit Fuzzy-Fingerprintings ist, wie Abbildung 4.3 zeigt, wesentlich schneller. Die erste Variante läuft zwar ebenfalls schneller als die übrigen Verfahren, weist jedoch keinen signifikanten Unterschied zu ihnen auf. Die Verfahren von Jourdan und Melançon (2004) und Chalmers u. a. (2003) unterscheiden sich nur minimal in den Laufzeiten. Damit kann das Verfahren mit Fuzzy-Fingerprinting als Verbesserung des hybriden Verfahrens zur multidimensionalen Skalierung betrachtet werden.

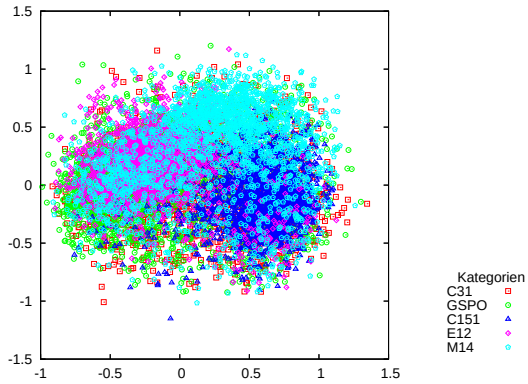
Eine weitere Frage ist, ob die Iterationen zur Verfeinerung der Punktkonfiguration, wie sie Chalmers u. a. (2003) vorschlagen, wirklich sinnvoll sind. Abbildungen 4.8 bis 4.12 zei-



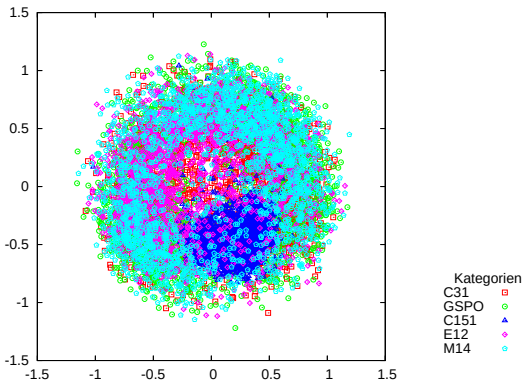
(a) Chalmers u. a. (2003)



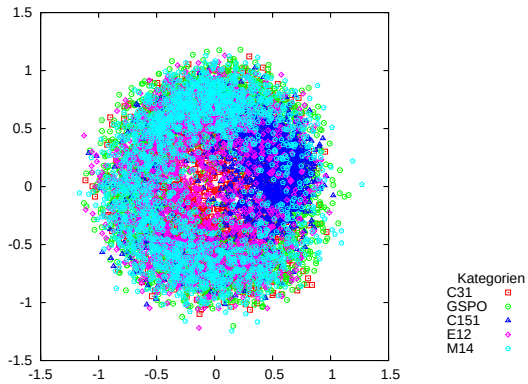
(b) Jourdan und Melançon (2004)



(c) Jourdan und Melançon (2004) Multiscale



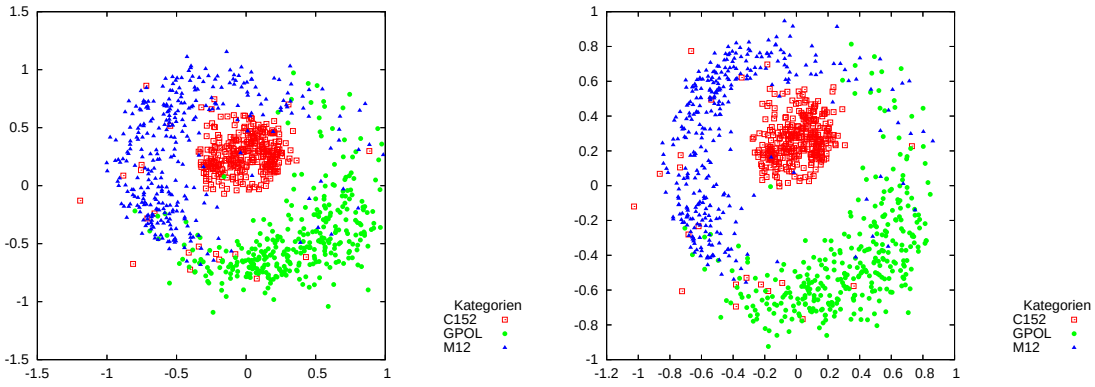
(d) mit Fuzzy-Fingerprinting (1)



(e) mit Fuzzy-Fingerprinting (2)

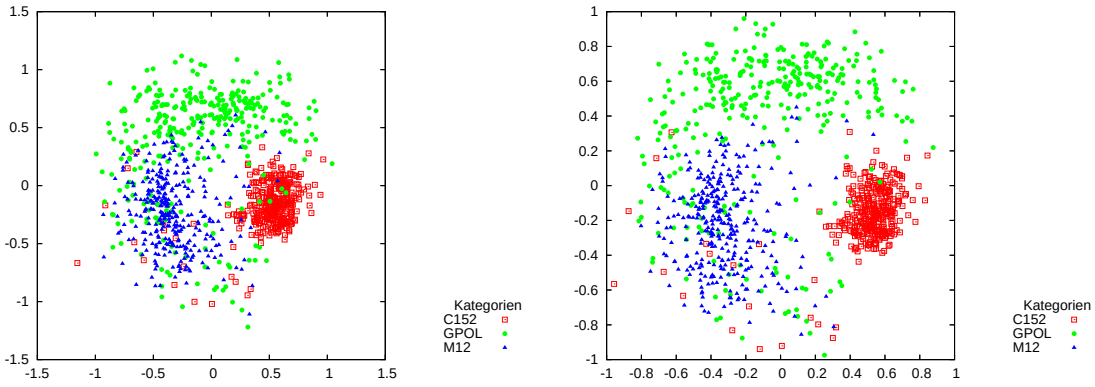
Abb. 4.7: Punktkonfigurationen der Evaluationskollektion mit 10 000 Dokumenten

gen die Punktkonfigurationen vor und nach den Iterationen zur Verfeinerung der Kollektion mit 1 000 Objekten. Offenbar wird die Punktkonfiguration dadurch nicht wesentlich verändert. In Anbetracht der Tatsache, dass dieser Verfeinerungsschritt ungefähr die Hälfte der Gesamtlaufzeit ausmacht, ohne eine signifikante Verbesserung der Punktkonfiguration zu erzielen, könnte auf ihn verzichtet werden. Allerdings basiert diese Annahme nur auf dem subjektiven Eindruck der während der Evaluation entstandenen Punktkonfigurationen und wurde nicht durch eigene Stressberechnungen für die Punktkonfigurationen vor dem Verfeinerungsschritt verifiziert.



(a) vorher (b) nachher

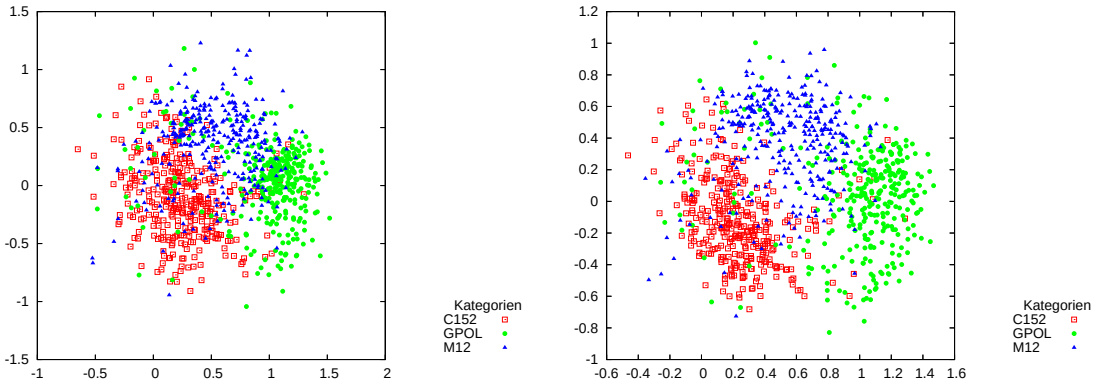
Abb. 4.8: Chalmers u. a. (2003) vor und nach dem Verfeinerungsschritt



(a) vorher (b) nachher

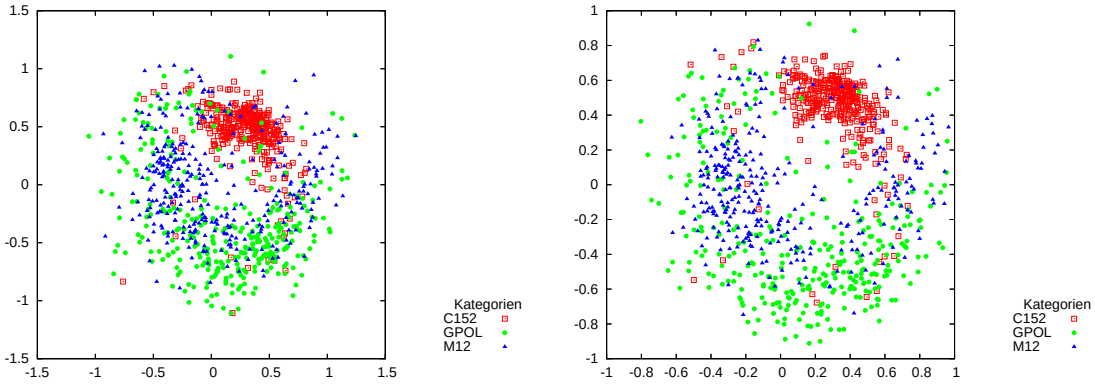
Abb. 4.9: Jourdan und Melançon (2004) vor und nach dem Verfeinerungsschritt

Des Weiteren hat sich durch die Evaluation klar gezeigt, dass die Interpolation zeitintensiver ist, als bisher angenommen wurde. Abbildung 4.13 veranschaulicht, wie viel von



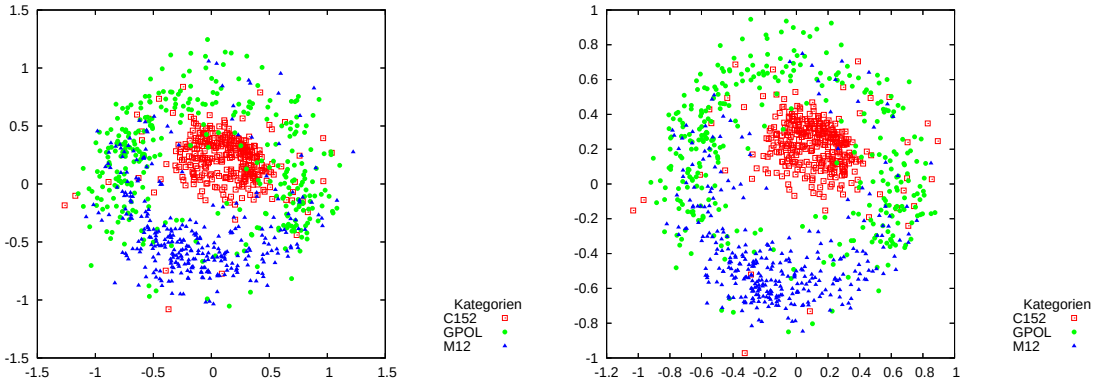
(a) vorher (b) nachher

Abb. 4.10: Jourdan und Melançon (2004) Multiscale vor und nach dem Verfeinerungsschritt



(a) vorher (b) nachher

Abb. 4.11: mit Fuzzy-Fingerprinting (1) vor und nach dem Verfeinerungsschritt



(a) vorher (b) nachher

Abb. 4.12: mit Fuzzy-Fingerprinting (2) vor und nach dem Verfeinerungsschritt

der Gesamtlaufzeit des Verfahrens von Chalmers u. a. (2003) und der Verbesserung mit Fuzzy-Fingerprinting auf die einzelnen Phasen fallen. Eine einfache Beschränkung der Objektmenge, auf Basis derer ein Objekt *i* interpoliert wird, resultiert aber möglicherweise in einer schlechteren Positionierung von *i* in der Punktkonfiguration und somit einem höheren Stresswert. Dies könnte sicherlich ein Ansatzpunkt für eine weitere Verbesserung des hybriden Verfahrens sein.

Außerdem sollte überprüft werden, ob speziell die Distanzen δ zwischen den Dokumenten auf ein anderes Distanzmaß Δ im niedrigdimensionalen Raum übertragbar sind. Die direkte Verwendung der Kosinus-Unähnlichkeit als Distanz δ hat den Nachteil, das alle Objekte auf eine kleine Region im Zielraum beschränkt werden, ungeachtet der Größe der Datenmenge. Folglich wird diese Region in einem Plot der Punktkonfiguration vollständig gefüllt, so dass keine Cluster mehr erkennbar sind. Dies ist eine Folge der Restriktion, dass die Distanz δ maximal den Wert 1 haben kann, auf Grund derer viele Punkte zwangsläufig übereinander liegen müssen. Williams und Munzner (2004) schlagen deshalb einen Ansatz vor, das auf dem Verfahren von Morrison und Chalmers (2003) basiert und die Möglichkeit gibt in der Punktkonfiguration zu navigieren, allerdings nicht alle Objekte gleichzeitig darstellen kann.

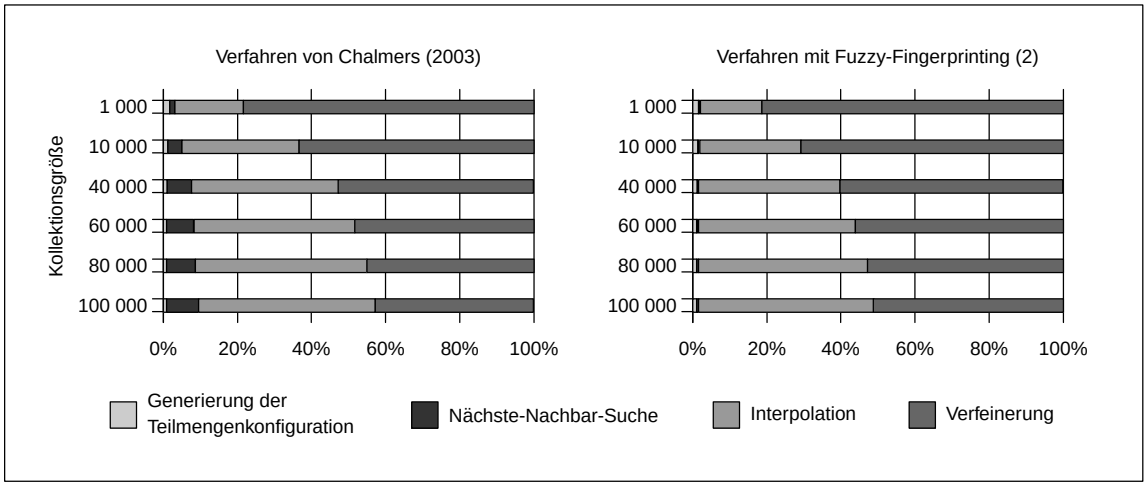


Abb. 4.13: Prozentuale Anteile der verschiedenen Phasen an der Gesamtlaufzeit des Verfahrens

5 Zusammenfassung

Verfahren der multidimensionalen Skalierung, die auf kräftegerichteter Positionierung mittels Distanzunterschieden basieren, wurden in der vorliegenden Arbeit im Hinblick auf ihre Laufzeiten und die Qualität der Punktkonfigurationen, die sie generieren, im Rahmen des textbasierten Information-Retrieval analysiert und evaluiert. Zunächst wurde die multidimensionale Skalierung allgemein erläutert und die Vorverarbeitung der Dokumente erklärt, welche die hochdimensionalen Eingabedaten der Verfahren darstellten. Die Verfahren wurden mit ihren Spezifikationen vorgestellt und ihre Laufzeit-Komplexität erfasst. Außerdem wurde ein neues Verfahren vorgestellt, welches die Nächste-Nachbar-Suche als Ausgangspunkt der Interpolation mittels Fuzzy-Fingerprinting beschleunigt. Eine Taxonomie der Verfahren zur multidimensionalen Skalierung wurde aufgestellt und die betrachteten Verfahren in diese eingegliedert. Die Analyse der Komplexitäten hat aufgezeigt, dass die Interpolationsphase der hybriden Verfahren den zeitaufwendigsten Schritt darstellt, das auch durch die Evaluation bestätigt wurde und aus diesem Grund weiterer Optimierung bedarf. Für die Evaluation wurden die betrachteten Verfahren implementiert. Anschließend wurden die Verfahren umfassend evaluiert, indem 6 Durchläufe der ausgewählten Verfahren für 6 unterschiedlich große Dokumentekollektionen mit bis zu 100 000 Dokumenten des Reuters-Korpus berechnet wurden. Der Stresswert als Bewertungsmaßstab für die so erhaltenen Punktkonfigurationen wurde ermittelt.

Durch die Evaluation konnte das effizienteste Verfahren hinsichtlich der kürzesten Laufzeit unter Beibehaltung der Qualität der generierten Punktkonfiguration im Vergleich mit ähnlichen Verfahren identifiziert werden. Die Evaluation hat gezeigt, dass die Weiterentwicklung des hybriden Verfahrens durch Fuzzy-Fingerprinting eine Verbesserung darstellt. Damit ist das Verfahren mit Fuzzy-Fingerprinting die effizienteste Methode innerhalb der

hier betrachteten Verfahren. Die in der Einleitung gestellte Aufgabe, die verschiedenen Ansätze bezüglich ihrer Laufzeiten sowie generierten Punktkonfigurationen zu untersuchen und zu bewerten, um das effizienteste Verfahren zu ermitteln, konnte somit erfüllt werden.

Eventuell sollte die Möglichkeit untersucht werden, die Exaktheit der Nächsten-Nachbar-Suche mit Fuzzy-Fingerprinting noch zu steigern. Ein Problem der Hashfunktionen des Fuzzy-Fingerprintings ist die ungleichmäßige Verteilung der Hashwerte und folglich der Dokumente in einer Hashtabelle. Das heißt, es gibt meist einige große Buckets und für viele Anfragen leere Ergebnismengen. Die Kompensation der leeren Ergebnismengen durch die zufällige Wahl von Buckets verschlechtert die Exaktheit der Nächsten-Nachbar-Suche maßgeblich. Eine hierarchische Verkettung der Hashfunktionen wäre möglicherweise vorteilhaft. Die Fuzzifizierungsschemata könnten den Dokumentenraum in verschieden große Regionen je Hashfunktion teilen. So könnte die erste Hashfunktion eine sehr feine Teilung generieren, was die *Precision* der Ergebnismenge bei einer Übereinstimmung einer Anfrage in der Hashtabelle erhöhen würde. Bei einer leeren Ergebnismenge würde die nächste Hashtabelle der Hashfunktion, die eine gröbere Partitionierung des Dokumentenraums generiert, geprüft werden. Somit würden bei einer leeren Ergebnismenge jeweils die nächste Hashfunktion in der Rangfolge hinsichtlich eines Ergebnisbuckets geprüft werden. Wenn die letzte Hashfunktion entsprechend gestaltet ist, dass sie eine sehr grobe Partitionierung des Dokumentenraums generiert, würde sich für nahezu alle Anfragen eine Ergebnismenge finden lassen, ohne zufällig Buckets auswählen zu müssen. Eine nähere Betrachtung dieser Möglichkeit wäre sicherlich aufschlussreich.

Literaturverzeichnis

[Baeza-Yates und Ribeiro-Neto 1999] BAEZA-YATES, R. ; RIBEIRO-NETO, B.: *Modern Information Retrieval*. Harlow, England : ACM Press/ Pearson Addison-Wesley, 1999

[Borg und Groenen 1997] BORG, I. ; GROENEN, P.: *Modern multidimensional scaling: theory and applications*. New York : Springer-Verlag, 1997

[Brodbeck und Girardin 1998] BRODBECK, D. ; GIRARDIN, L.: *Combining topological clustering and multidimensional scaling for visualizing large data sets*. 1998. – unpubliziert, akzeptiert für Proceedings of IEEE Information Visualization

[Chalmers 1996] CHALMERS, M.: A Linear Iteration Time Layout Algorithm for Visualising High-Dimensional Data. In: *Proceedings of 7th IEEE Visualization 96*. San Francisco, 1996, S. 127 – 132

[Chalmers u. a. 2003] CHALMERS, M. ; MORRISON, A. ; ROSS, G.: Fast Multidimensional Scaling through Sampling, Springs and Interpolation. In: *Proceedings of Information Visualization* Bd. 2, 2003, S. 68–77

[Cox und Cox 1994] COX, T.F. ; COX, M.A.A.: *Multidimensional scaling*. 1. London : Chapman and Hall, 1994

[Don und Hanusse 2006] DON, A. ; HANUSSE, N.: A Deterministic Multidimensional Scaling Algorithm for Data Visualisation. In: *Proceedings of Information Visualization*, 2006

[Eberly 2004] EBERLY, D.H.: *Game Physics*. San Francisco : Elsevier, Morgan Kaufmann Publishers, 2004

[Ferber 2003] FERBER, R.: *Information Retrieval - Suchmodelle und Data-Mining-Verfahren für Textsammlungen und Web*. 2003. – URL `http://information-retrieval.de/irb/irb.html`. – (letzter Zugriff: 04.02.2007)

[Jourdan und Melançon 2004] JOURDAN, F. ; MELANÇON, G.: Multiscale hybrid MDS. In: *Proceedings of 8th International Conference on Information Visualization (IV'04)*, 2004, S. 388–393

[Mathar 1997] MATHAR, R.: *Multidimensionale Skalierung: Mathematische Grundlagen und algorithmische Aspekte*. Stuttgart : B.G. Teubner, 1997

[Matyka 2004] MATYKA, M.: *How To Implement a Pressure Soft Body Model*. 2004. – URL `http://panoramix.ift.uni.wroc.pl/~maq/soft2d/howtosoftbody.pdf`. – (letzter Zugriff: 30.01.2007)

[Morrison und Chalmers 2003] MORRISON, A. ; CHALMERS, M.: Improving Hybrid MDS with Pivot-Based Searching. In: *2003 IEEE Symposium on Information Visualization*, 2003, S. 11

[Morrison und Chalmers 2004] MORRISON, A. ; CHALMERS, M.: *A Pivot-Based Routine for Improved Parent-Finding in Hybrid MDS*. 2004

[Morrison u. a. 2002a] MORRISON, A. ; ROSS, G. ; CHALMERS, M.: Combining and comparing clustering and layout algorithms. Department of Computing Science, University of Glasgow, 2002. – Forschungsbericht

[Morrison u. a. 2002b] MORRISON, A. ; ROSS, G. ; CHALMERS, M.: A Hybrid Layout Algorithm for Sub-Quadratic Multidimensional Scaling. In: *Proceedings of IEEE Information Visualisation 2002*, 2002, S. 152 – 160

[Potthast 2006] POTTHAST, M.: *Hashing-basierte Indizierungsverfahren im textbasierten Information-Retrieval*, Diplomarbeit, 2006

[Stein 2005] STEIN, B.: Fuzzy-Fingerprints for Text-Based Information Retrieval. In: *Tochtermann, Maurer (Eds.): Proceedings of the 5th International Conference on Knowledge Management (I-KNOW 05), Journal of Universal Computer Science*. Graz, 2005, S. 572–579

[Stein 2006] STEIN, B.: *Advanced Web Technology*. 2006. – URL `http://www.uni-weimar.de/cms/Lecture_Notes.550.0.html`. – (letzter Zugriff: 30.01.2007), Vorlesungsskript.

[Stein und Potthast 2006] STEIN, B. ; POTTHAST, M.: *Hashing-basierte Indizierung: Anwendungsszenarien, Theorie und Methoden*. 2006

[Williams und Munzner 2004] WILLIAMS, M. ; MUNZNER, T.: Steerable, Progressive Multidimensional Scaling. In: *IEEE Symposium on Information Visualization (INFOVIS'04)*. Austin, 2004, S. 57–64