Clustering von Dokumenten (k-means, HCL)

Jonas Wolz

Universität Ulm

Zusammenfassung

Ein Überblick über das Clustering von Dokumenten. Außerdem werden zwei dafür verwendete Algorithmen vorgestellt (k-means und Hierarchisches Clustering (HCL)).

31. Mai 2005

Inhaltsverzeichnis

1 Einleitung

1.1 Ziele

In den bisherigen Vorträgen wurden u.a. Methoden zur Dokumentenklassifikation vorgestellt, die bei vorgegebenen Kategorien mit entsprechend eingeordneten Beispieldokumenten neue Dokumente (mehr oder weniger) richtig in die bestehenden Kategorien einordnen können. Dabei mussten aber immer zunächst die Beispieldokumente von einem Menschen kategorisiert werden, was bei größeren Mengen von Beispielen meist eine recht langwierige und damit aufwendige Aufgabe darstellt.

Beim Information Retrieval war diese Kategorisierung nicht notwendig, allerdings musste der Benutzer auch hier einen ungefähren Überblick über die Dokumente haben, um sinnvolle bzw. erfolgreiche Anfragen an das System stellen zu können.

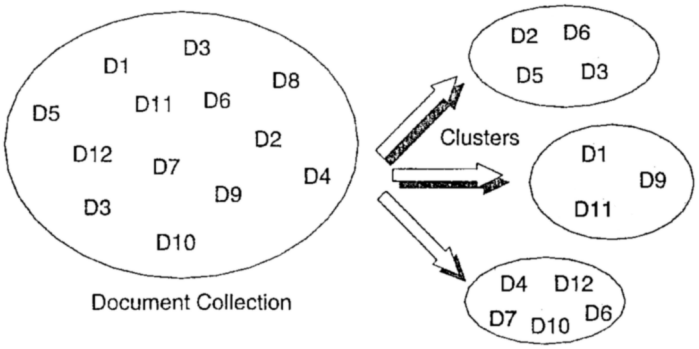

Abbildung 1. Schema Dokumentenclustering

Das Ziel des Dokumentenclustering ist es, die Einteilung einer Dokumentensammlung in (möglichst „sinnvolle“) Kategorien zu automatisieren, d.h. man übergibt eine ungeordnete Sammlung von Dokumenten und lässt diese durch den Computer in kleinere Gruppen/Cluster aufteilen (siehe auch Abbildung 1). Die Cluster sollten dabei natürlich jeweils zueinander „ähnliche“ Dokumente enthalten. Dies bedeutet insbesondere, dass jedes Dokument jedem beliebigen anderen Dokument im selben Cluster ähnlicher sein sollte als solchen in irgendeinem anderen.
Außerdem sollte, um ein Clusteringverfahrens auch praktisch anwenden zu können, der Rechenaufwand bzw. die rechnerische Komplexität klein genug für einen sinnvollen Einsatz sein.

Im Idealfall würde beim Clustering ein ähnliches Ergebnis entstehen wie bei einer Einteilung durch einen Menschen. Das ist in der Praxis jedoch meist leider nicht der Fall, wie auch im nächsten Abschnitt ausgeführt.

1.2 Probleme

Das Hauptproblem beim Dokumentenclustering ist, dass je nach Anwendung die Kriterien/Ziele der Kategorisierung unterschiedlich sind. Diese Ziele der Einteilung sind einem Menschen, der Dokumente kategorisiert i.d.R. vorher bekannt, dem Computer jedoch nicht.
Da der Computer zum Clustering Ähnlichkeitsmaße verwendet (und den Text nicht wirklich „versteht“), ist es oft sogar so, dass die „Kriterien“, nach denen sich die Dokumente in den einzelnen Clustern unterscheiden, erst *nach* dem Clustering erkennbar werden. Deshalb kann es auch passieren, dass eine maschinelle Einteilung als nicht sinnvoll verworfen werden muss.
Auch wenn die sich ergebende Einteilung als sinnvoll betrachtet werden kann, wird sie sich in den allermeisten Fällen von einer durch einen Menschen vorgenommenen Einteilung deutlich unterscheiden (da der Computer eben ein grundlegend anderes Verständnis von Ähnlichkeit hat als ein Mensch).
Als Extremfall wäre z.B. eine Sammlung von Artikeln in verschiedenen Sprachen denkbar: Ein Mensch würde die Artikel i.d.R. in Gruppen zusammenfassen, die ein ähnliches Thema behandeln (vorausgesetzt er versteht alle vorkommenden Sprachen). Ein Computer hingegen würde Artikel in verschiedenen Sprachen so gut wie nie in dieselbe Kategorie einordnen, da sie sich in den verwendeten Worten naturgemäß sehr stark unterscheiden (zumindest bei Verwendung der üblichen Ähnlichkeitsmaße).

Trotzdem kann Dokumentenclustering in der Praxis sehr nützlich sein, da eine Dokumentensammlung dadurch zusätzlich an Struktur gewinnt und die gefundenen Kategorien trotz dessen, dass ein Mensch andere gewählt hätte, sinnvoll sein können. Im Allgemeinen ist diese Strukturierung insbesondere dann praktisch, wenn man bei einer Dokumentensammlung noch keinen genaueren Überblick über die Inhalte der einzelnen Dokumente besitzt und damit auch keine möglicherweise sinnvollen Kategorien kennt.
Das kann auch dabei helfen, „richtige“ Anfragen an ein Information Retrieval System über die bis dahin „unbekannte“ Dokumentsammlung zu formulieren.

1.3 Anwendungsbeispiele

Eine vorstellbare Anwendung für Dokumentenclustering wäre ein Call Center, an das die Kunden als Freitext Problembeschreibungen senden können. Der Betreiber möchte nun wissen, welche Gruppen von Problemen am häufigsten auftreten. Dabei kann man davon ausgehen, dass wahrscheinlich z.B. Druckerprobleme in einem Cluster und Netzwerkprobleme in einem anderen landen. Nach dem Clustering mit einer festen Anzahl von Clustern (z.B. 100) kann man dann die einzelnen Cluster untersuchen, um einen Überblick zu bekom-

men (z.B. anhand von darin häufig vorkommenden Schlüsselwörtern).

Eine andere Anwendung ist, die Ergebnisse von Suchmaschinen zu kategorisieren, um dem Benutzer thematisch sortierte Suchergebnisse (ähnlich dem Inhaltsverzeichnis eines Buchs) anbieten zu können. Je nach Anwendung lässt sich allerdings darüber streiten, ob diese Darstellung nun praktischer/„besser“ oder weniger praktisch als die typischerweise nach Relevanz sortierte Ergebnisliste einer „normalen“ Suchmaschine ist.
Eine Suchmaschine mit dieser Arbeitsweise kann man z.B. unter `http://www.vivisimo.com/` finden, so dass sich hierzu auch jeder selbst ein Bild machen kann. Diese Suchmaschine verwendet nach [5], eine (leider nicht näher beschriebene) Form des hierarchischen Clustering, um die Suchergebnisse in Kategorien einzuteilen.

2 Clustering und Ähnlichkeitsmaße

2.1 Unterschiede zum Information Retrieval

Weil beim Dokumentenclustering die Ähnlichkeit der Dokumente das zentrale Kriterium für die Gruppierung darstellt, spielen hier die Ähnlichkeitsmaße aus dem Information Retrieval eine wichtige Rolle.

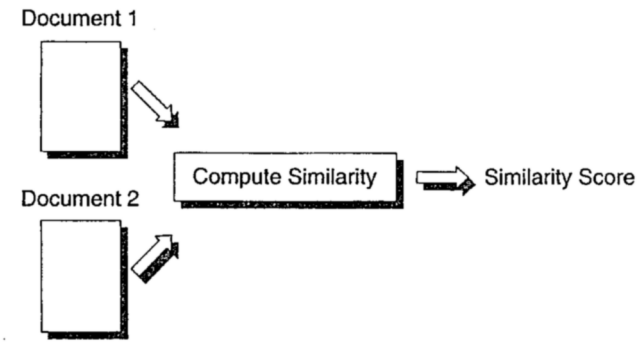

Abbildung 2. Ähnlichkeitsvergleiche beim Information Retrieval

Der Hauptunterschied zum Information Retrieval besteht darin, dass nicht wie beim Information Retrieval zwei einzelne Dokumente (Abbildung 2) sondern meist zwei „Kompositdokumente“ (also Zusammenfassungen mehrerer [1] Einzeldokumente) verglichen werden (Abbildung 3).
Wie die Kompositdokumente genau repräsentiert werden, hängt stark vom verwendeten Algorithmus ab. So verwendet k-means z.B. den „Durchschnitt“

[1] im Sinne von „≥ 1 Dokument“

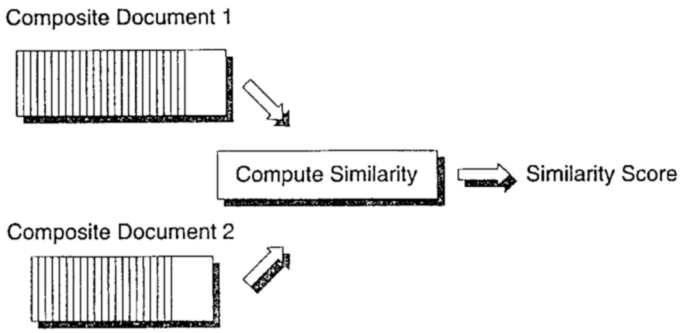

Abbildung 3. Ähnlichkeitsvergleiche beim Dokumentenclustering

der Dokumente in Vektordarstellung, während beim Hierarchischen Clustering eine Baumstruktur mit Verweisen auf die Dokumente verwendet wird.

2.2 Ähnlichkeit von Kompositdokumenten

Die beim Clustering verwendeten Ähnlichkeitsmaße sind im Großen und Ganzen die selben wie beim Information Retrieval.
Am häufigsten wird dabei die Kosinusähnlichkeit (vgl. [3]) verwendet, insbesondere da sich diese relativ schnell berechnen lässt.

Um Kompositdokumente/Cluster mit Hilfe von Ähnlichkeitsmaßen vergleichen zu können, müssen sie so dargestellt werden, dass das Ähnlichkeitsmaß etwas mit ihnen „anfangen“ kann, was meist bedeutet, dass ein Kompositdokument irgendwie als ein Einzeldokument dargestellt werden muss. Dafür gibt es mehrere Möglichkeiten:

Der einfachste und naheliegendste Weg ist, einfach den Durchschnitt der Vektoren[2] der Dokumente in jedem Cluster (manchmal auch als „Schwerpunkt“ (engl. centroid) bezeichnet) zu berechnen. Man erhält dadurch für jedes Cluster einen einzigen Durchschnittsvektor, der den Cluster repräsentiert, und den man (ggf. normalisiert) bei Vergleichen wie einen Vektor für einzelnes Dokument behandeln kann.

Eine Alternative dazu ist es, *ein einzelnes* Dokument aus dem Cluster als Repräsentanten auszuwählen. Die Ähnlichkeit zweier Cluster wird dann über die Ähnlichkeit beider Repräsentanten gemessen. Um den Repräsentanten zu finden, vergleicht man ein einzelnes Dokument außerhalb des Clusters mit allen Dokumenten im Cluster bzw. beim Vergleich zweier Cluster alle möglichen Paare von Dokumenten der beiden Cluster[3].

[2] z.B. tf-idf-Vektoren; vgl. [2]
[3] Also alle Elemente aus dem „kartesischen Produkt“ der beiden Cluster

Um das „beste“ Paar und damit das ähnlichste Cluster zu finden, verwendet man meist eine der folgenden Möglichkeiten (vgl. auch Abbildung 4):

single link: Man verwendet die beiden ähnlichsten Dokumente. Dazu untersucht man einmal alle möglichen Paare und wählt dann die beiden Cluster, zu denen das Paar mit der größten Ähnlichkeit gehört.

complete link: Man verwendet die beiden *un*ähnlichsten Dokumente. Dazu geht man in zwei Schritten vor:

(1) Man untersucht paarweise alle Cluster und wählt die beiden unähnlichsten Dokumente aus.

(2) Man wählt das Paar von Clustern aus, bei denen sich die beiden unähnlichsten Dokumente am ähnlichsten sind.

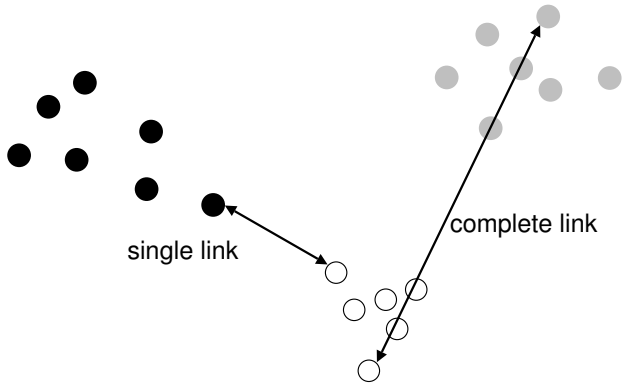

Abbildung 4. single link und complete link

3 *k*-means

3.1 Überblick

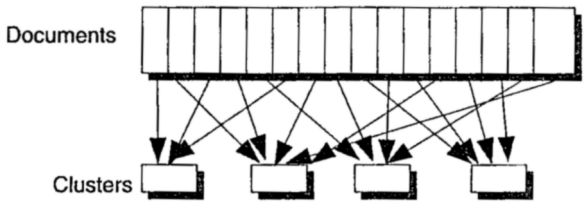

Abbildung 5. Grundidee bei *k*-means

k-means ist eine klassische und weit verbreitete Methode zum Clustering. Die Grundidee (vgl. Abbildung 5) ist einfach die Menge von Dokumenten auf k Cluster von ähnlichen Dokumenten zu verteilen. Cluster werden dabei, wie der Name *k*-„means“ [4] schon sagt, anhand von Durchschnittsvektoren gebildet.

[4] engl. mean: Durchschnitt

(1) Verteile alle Dokumente auf k Cluster

(2) Berechne den Durchschnittsvektor für jeden Cluster

(3) Vergleiche alle Dokumente mit den Durchschnittsvektoren aller Cluster und „notiere“ den jeweils ähnlichsten für jedes Dokument

(4) Verschiebe alle Dokumente in den jeweils ähnlichsten Cluster

(5) Wenn keine Dokumente in einen anderen Cluster verschoben wurden, halte; sonst gehe zu (2).

Abbildung 6. k-means Algorithmus

Abbildung 6 beschreibt den Algorithmus in Kurzform. Der Wert von k muss beim Start des Algorithmus gegeben sein. Die geschickte Wahl von k stellt eines der größten Probleme beim Einsatz des Algorithmus dar (siehe auch nächster Abschnitt).

k-means hält bei einer Art von lokalem Minimum des Durchschnittsvektors, weshalb das Endergebnis (je nach Daten mehr oder weniger stark) mit von der Verteilung der Dokumente in Schritt (1) abhängt.
Klassischerweise verteilt man die Dokumente einfach zufällig auf die Cluster. Hierbei kann es dabei allerdings für gute Ergebnisse nötig sein, den Algorithmus mehrmals laufen zu lassen und sich dann eine (für den jeweiligen Zweck) brauchbare Aufteilung herauszusuchen.
Ein anderer Ansatz ist es, zuerst den Durchschnittsvektor über alle Dokumente auszurechnen, die Dokumente nach Ähnlichkeit zu diesem zu sortieren und dann eine jeweils gleiche Anzahl von Dokumenten in Sortierreihenfolge auf die Cluster zu verteilen. Bei der danach folgenden Anwendung des k-means Algorithmus werden dann nur noch Verschiebungen zwischen „benachbarten“ Clustern zugelassen.

Typischerweise konvergiert k-means nach einer relativ geringen Anzahl von Iterationen. Wenn k, also die Anzahl der Cluster, im Vergleich zur Anzahl der Dokumente klein ist, kann man k-means als effizient ansehen (annähernd $O(n)$). Bei großem k vermindert sich die Effizienz allerdings stark [5]; im Grenzfall, nämlich $k = n =$“Anzahl der Dokumente”, muss in jedem Schritt jedes Dokument mit allen anderen verglichen werden (die Komplexität ist also annähernd $O(n^2)$).

[5] Nach [4] beträgt die Komplexität $O(iknm)$. Dabei ist i die Anzahl der Iterationen, k die Anzahl der Cluster, n die Anzahl der Dokumente und m die Dimension der Dokumentvektoren. k und m sind konstant und i, dessen Wert u.a. von k, n und den konkreten Daten abhängt, wird meist nur relativ klein.
$\Rightarrow$ Man kann dies als annähernd $O(n)$ betrachten.

Abbildung 7 zeigt einen Beispieldurchlauf für den Algorithmus (wobei hier jedem Dokument nur eine Zahl und nicht ein ganzer Vektor zugeordnet ist).

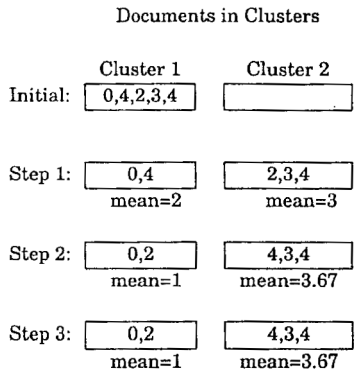

Abbildung 7. Beispiel für k-means ($k = 2$)

3.2 Bestimmung von k

Der k-means Algorithmus selbst beinhaltet keine Möglichkeit, k zu wählen. Daher bleibt es dem „Benutzer“ des Algorithmus überlassen, ein günstiges k zu wählen.

Oft ergibt sich dabei glücklicherweise aus der jeweiligen Anwendung ein möglicher Wert für k. Wenn man die zu erwartende Anzahl von Kategorien für die Dokumente einigermaßen einschätzen kann, kann man oft ein passendes k raten. Z.B. ist möglicherweise klar, dass mehr als 10 Cluster nicht sinnvoll sind bzw. dass (wie im Call-Center-Beispiel von oben) eine großes k (z.B. 100) nötig ist. Auf jeden Fall sollte k für brauchbare Ergebnisse immer wesentlich kleiner als die Anzahl der Dokumente gewählt werden.

Eine anderer Ansatz ist es, eine Funktion zu definieren, die die „Qualität“ des Ergebnisses misst und k dann so zu wählen, dass die Qualität möglichst gut wird. Beispielsweise lässt sich sich dafür die Varianz der Dokumentenvektoren in Bezug auf die Durchschnittsvektoren verwenden:
Sei n die Anzahl der Dokumente, k die Anzahl der Cluster, x^i der i-te Dokumentvektor, $c_i \in \{1, \cdots, k\}$ der Index des entsprechenden Clusters und m_{c_i} der entsprechende Durchschnittsvektor (bzw. „Schwerpunkt“):

$$E(k) = \sum_{i=1}^{n} \frac{(x^i - m_{c_i})^2}{n}$$

Zur Bestimmung von k beginnt man mit einem kleinen Wert für k (z.B. $k = 2$), wendet k-means an und berechnet die Varianz. Danach vergrößert (z.B. verdoppelt) man k, wendet wieder k-means an und berechnet wieder die Varianz.

Dies wiederholt man so lange, bis k zu groß wird. k wählt man dann so, dass die Vergrößerung des Werts keine signifikante Verkleinerung der Varianz mehr mit sich bringt.

4 Hierarchisches Clustering (HCL)

4.1 Überblick

(1) Beginne mit vielen Clustern, die jeweils genau ein Dokument enthalten.

(2) Finde das ähnlichste Paar B und C von Clustern, die keinen Elternknoten besitzen.

(3) Vereinige B und C zu einem Eltern-Cluster A.

(4) Wenn mehr als ein Cluster ohne Eltern übrigbleibt, gehe zu (2).

Abbildung 8. Algorithmus zum Hierarchischen Clustering

Hierarchisches (agglomeratives[6]) Clustering ist eine populäre Alternative zu k-means. Auch hier werden (selbstverständlich) Cluster erzeugt, die allerdings in einer hierarchischen Baumstruktur angeordnet werden. Es können viele verschiedene Ähnlichkeitsmaße verwendet werden, u.a. der Durchschnitt, single/complete link, aber auch der Minimal- bzw. Maximalabstand der Dokumente innerhalb eines Clusters.

Die Hauptschwäche des hierarchischen Clustering ist die rechnerische Komplexität ($O(n^2)$ oder schlechter; vgl. auch nächsten Abschnitt) und damit die Geschwindigkeit. Hierarchisches Clustering ist daher hauptsächlich zum Clustering relativ kleiner Mengen von Dokumenten geeignet (wofür es eigentlich auch gedacht ist).

Abbildung 8 zeigt einen Algorithmus für hierarchisches agglomeratives Clustering. Das Endresultat ist ein Binärbaum, bei dem die Wurzel ein Cluster mit allen Dokumenten darstellt. Die Kinder stellen jeweils eine Aufteilung des Eltern-Cluster in zwei kleinere dar. Die Blätter schließlich enthalten die kleinsten Cluster, i.d.R mit jeweils nur einem Dokument. Abbildung 9 zeigt ein Beispiel für hierarchisches Clustering.

[6] „agglomerativ“ bzw. „vereinigend“ bedeutet, dass man mit kleinen Clustern beginnt und diese dann schrittweise zu größeren vereinigt; das Gegenteil davon heißt „divisiv“ bzw. „teilend“, dabei wird ein großes Cluster schrittweise in kleinere aufgespaltet

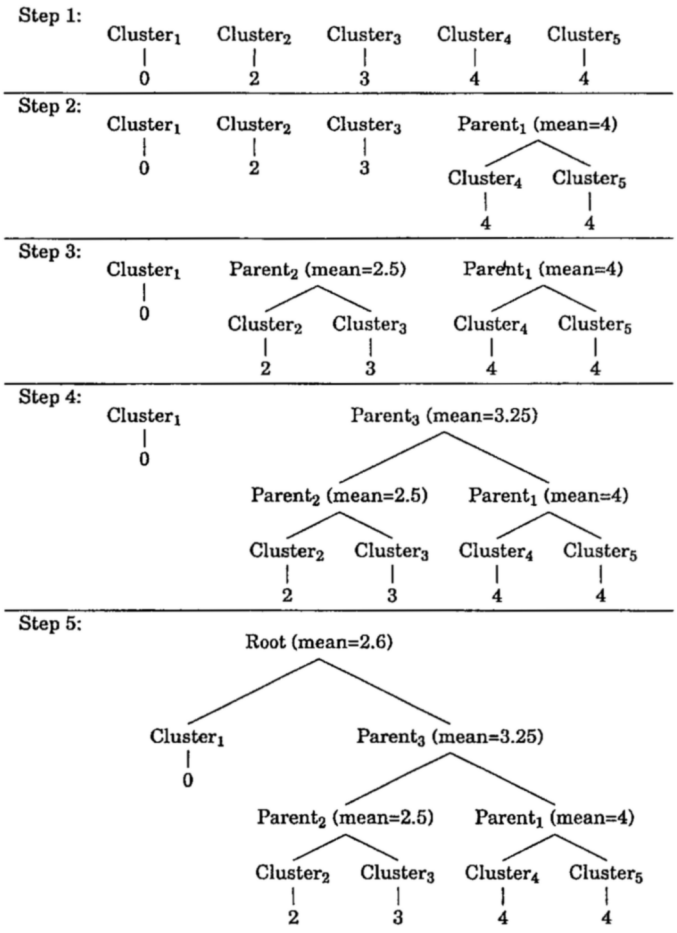

Abbildung 9. Beispiel für Hierarchisches Clustering

Die Struktur des entstehenden Baums hängt stark vom verwendeten Ähnlichkeitsmaß ab. So kann single link z.B. manchmal zu einer langen Kette von Clustern führen, die dem Rest vom Baum ziemlich unähnlich sind (der Baum degeneriert teilweise also zu einer linearen Liste). Complete Link verhält sich hier besser, braucht aber weitaus mehr Rechenzeit.

Wenn man in den Blättern größere Cluster bekommen möchte, kann der Baum auf die gewünschte Größe „zugeschnitten“ werden. Wie man den Baum dazu am sinnvollsten abschneidet, lässt sich allerdings nicht eindeutig sagen. Der einfachste Weg besteht darin, ihn einfach ab einer gewissen Tiefe abzuschneiden. Dadurch erhält man i.d.R. auch einen halbwegs ausbalancierten Baum. Ein anderer Ansatz besteht darin, ihn so zuzuschneiden, dass die Varianz in den übrig bleibenden Clustern möglichst gering wird.

4.2 Vergleich zu k-means

Wie oben schon angedeutet ist ein Hauptnachteil des Hierarchischen Clustering die Rechenzeitkomplexität: Hierarchische Clusteringalgorithmen, die Paare von Dokumenten und keine Durchschnittsvektoren vergleichen (z.B. single link), besitzen laut [1] und [4] eine Komplexität von $O(n^2)$ (n ist die Anzahl der Dokumente). Algorithmen, die die unähnlichsten Dokumente zum Vergleich verwenden (z.B. complete link) sind noch ineffizienter. k-means hingegen kann für konstantes k und eine kleine Zahl von Iterationen nach [4] und [1] als annähernd $O(n)$ betrachtet werden.

Nach [1] erhält man bei einer vorgegebenen Zahl von Clustern i.d.R. mit k-means bessere Ergebnisse als mit Hierarchischem Clustering, d.h. die Varianz der Ergebnisse ist geringer. Die Abbildungen 7 und 9 zeigen beide Algorithmen auf die gleichen Daten angewandt.
Andererseits lässt sich bei Hierarchischem Clustering eine sinnvolle Anzahl von Clustern direkt aus dem Baum gewinnen, ohne den Algorithmus mehrmals ausführen zu müssen, während es bei k-means möglicherweise nötig ist, verschiedene Werte von k „auszuprobieren“ (wobei für jeden Wert von k der Algorithmus einmal vollständig ausgeführt werden muss).

Zusammenfassend lässt sich sagen, dass sich hierarchisches Clustering hauptsächlich dann lohnt, wenn man eine Hierarchie der Dokumente benötigt. Braucht man eine solche Hierarchie nicht, ist k-means in vielen Fällen besser geeignet.

Außerdem sei zum Abschluss noch angemerkt, dass sich beide Algorithmen nicht nur zum Clustering von Dokumenten, sondern von beliebigen als Vektor darstellbaren Daten eignen und auch dazu verwendet werden, wie z.B. eine Internetsuche nach „k-means“ oder „Hierarchical Clustering“ schnell zeigt.

Literatur

[1] Sholom Weiss, Nitin Indurkhya, Tong Zhang, Fred Damerau: *Text Mining*, Springer Verlag 2005.

[2] Julian Forster: *Die Transformation von Text in Vektoren*, URL: `http://www.informatik.uni-ulm.de/ni/Lehre/SS05/ProseminarTextMining/ausarbeitungen/Forster.pdf`

[3] Ferdinand Hofherr: *Information Retrieval*, URL: `http://www.informatik.uni-ulm.de/ni/Lehre/SS05/ProseminarTextMining/ausarbeitungen/Hofherr.pdf`

[4] Gilad Mishne, Maarten de Rijke: *Clustering* (Vorlesungsfolien), URL: `http://lit.science.uva.nl/Teaching/0405/II/page9.html`

[5] *Vivísimo // Frequently Asked Questions*, URL: `http://vivisimo.com/html/faq`