

# **Document Clustering**

Inside Google: Algorithmen für Suchmaschinen

Vortrag 15. Januar 2007

von

Annette Hamm

Betreuer: Dejavuth Suwimonteerabuth

# Inhaltsverzeichnis

## 1. Einführung

## 2. Hierarchical Agglomerative Clustering (HAC)

- 2.1. HAC – ein Clusteringverfahren
- 2.2. Distanz

## 3. Web Document Clustering

- 3.1. Group-Average-Clustering
- 3.2. Word-Intersection-Clustering
- 3.3. Phrase-Intersection-Clustering
  - 3.3.1. Phrase-Intersection-Clustering mit GQF
  - 3.3.2. Phrase-Intersection-Clustering mit Suffixbaum

## 4. Vergleich

- 4.1. Versuchsbedingungen
- 4.2. Qualität der Clusterings
- 4.3. Zeiten der verschiedenen Algorithmen
- 4.4. Ergebnis des Vergleichs

## 5. Zusammenfassung und Fazit

Quellenangabe

## 1. Einführung [1], [2], [8]

Es existieren im Internet mehr als 300 Millionen Seiten und täglich kommen 1 Million neue Seiten dazu. Suchmaschinen speichern sich ein Kontingent an Seiten in einer Datenbank ab, wodurch die Zeit des Suchens wesentlich verkürzt wird. Erfolgt eine Suchanfrage, so werden diese ausgewählten Seiten auf Wortübereinstimmungen durchsucht. Die Anzahl dieser Seiten ist dabei sehr groß. Es kommt zu einem Ranking der Seiten und damit einer Auflistung, welche meist unstrukturiert ist. Oft wäre es jedoch möglich, die Ergebnisse in Themen aufzuteilen und sie in einer hierarchischen Anordnung auszugeben. Solche thematischen Einteilungen, auch als Cluster bezeichnet, sind Kategorien, Klassen oder Gruppen mit jeweils ähnlichem Inhalt. Daten eines Clusters sollten dabei möglichst ähnlich sein und Daten unterschiedlicher Cluster möglichst unähnlich.

Um Cluster zu erstellen, werden verschiedene Clusteringalgorithmen verwendet. Wichtige Anforderungen an solche Algorithmen sind

- Schnelligkeit, da der User nicht lange auf seine Informationen warten möchte,
- Erweiterbarkeit, damit neue Schnipsel aus dem Internet sofort weiterverarbeitet werden können,
- Unterscheidungsmöglichkeit, um wichtige von unwichtigen Informationen zu trennen, und
- Schnipselfähigkeit, da aus Zeitmangel nicht das ganze Dokument heruntergeladen werden kann und der Algorithmus somit anhand von Schnipseln Cluster mit hoher Qualität erzeugen muss.

Für eine Suchanfrage ergibt sich dadurch, in möglichst kurzer Zeit, eine leicht zu durchsuchende und mit hauptsächlich relevanten Informationen angereicherte Ausgabe.

Im folgenden werden einige Möglichkeiten und Voraussetzungen solcher Clusteringalgorithmen vorgestellt und verglichen.

## 2. Hierarchical Agglomerative Clustering (HAC)

### 2.1. HAC – ein Clusteringverfahren [4], [10]

Clusteringverfahren kann man grundsätzlich in partitionierende und hierarchische Verfahren unterscheiden.

Bei partitionierenden Verfahren benötigt man die gewünschte Anzahl  $k$  der Cluster und eine Distanzfunktion. Aus allen Dokumenten werden  $k$  Clusterrepräsentanten ausgewählt. Die Cluster werden solange modifiziert, bis das lokale Optimum der Partitions-güte erreicht ist. Da in jedem Schritt nur lokal optimiert wird, ist das Ergebnis sehr stark von der zu Beginn gewählten Zerlegung abhängig. Die Anzahl  $k$  der gewünschten Cluster hat einen großen Einfluss auf die Qualität des Clusteringergebnisses.

Bei hierarchischen Verfahren, nach der bottom-up oder top-down-Methode, benutzt man ebenfalls eine Distanzfunktion. Es wird eine Hierarchie bei der üblicheren bottom-up-Methode dadurch bestimmt, dass jeweils die zwei ähnlichsten Cluster gemischt werden. Dies wird solange wiederholt, bis alle Dokumente in einem Clustering zusammengefasst sind.

Im folgenden werden Hierarchische Agglomerative Clusteringverfahren, auch HAC genannt, betrachtet. Hierfür wird als erstes ein Binärbaum angelegt, welcher aus einzel-

nen Clustern besteht. Am Anfang wird jedes Dokument in ein Cluster abgelegt. In jedem folgenden Schritt werden zwei dieser Cluster, welche sich möglichst ähnlich sein sollten, entfernt, gemischt und als ein neues Cluster hinzugefügt. Dies wird solange fortgesetzt, bis alle Cluster, die verschmelzen könnten, zu unähnlich sind.

Bei der Anwendung des HAC stellt die Anzahl der Dokumente ein Problem dar, da diese nicht statisch ist, sondern sich erst nach und nach ergibt. Man müsste am besten jeweils alle Dokumente speichern und periodisch clustern.

Zur Verwendung von HAC benötigt man Haltekriterien, die Distanz zwischen zwei Dokumenten und die Distanz zwischen zwei Clustern. Haltekriterien sollten zu einem ausgeglichenen Verhältnis zwischen der Anzahl der Cluster, deren Größe und ihrem Zusammenhalt führen. Sie werden je nach Algorithmus festgelegt, meist in Form von Kriterien wie beispielsweise der Ähnlichkeit der beinhalteten Dokumente.

Die Distanz zwischen zwei Dokumenten kann man mit Hilfe mathematischer Grundlagen errechnen. Werden die Dokumente  $a, b$  als „Punkte“ angesehen, so kann die

Formel  $dist(a, b) = \sqrt{\sum_{i=1}^n (a_i - b_i)^2}$  benutzt werden.

Eine weitere Möglichkeit ist, dass die Dokumente als gewichtete Vektoren gesehen werden, wobei jedes Wort, das im Dokument enthalten ist, als Attribut des Vektors dient, wodurch ein mehrdimensionaler Raum entsteht.

In der Mathematik wird der Winkel zwischen zwei Vektoren mit der Formel

$\cos \alpha = \frac{\vec{a} \circ \vec{b}}{|\vec{a}| \cdot |\vec{b}|}$  berechnet. Da der Winkel um so größer wird, je „ähnlicher“ sich die beiden Vektoren sind, wird die Formel durch Abzug von 1 erweitert. Somit wird die Distanz immer kleiner, je ähnlicher sich die Dokumente sind  $dist(a, b) = 1 - \frac{\vec{a} \circ \vec{b}}{|\vec{a}| \cdot |\vec{b}|}$

## 2.2. Distanz [5]

Um nun die Distanz zwischen zwei Clustern zu berechnen, muss darauf geachtet werden, dass das Ergebnis das resultierende Cluster ausreichend gut charakterisiert. Möglichkeiten der Berechnung sind Single-Link, Complete-Link oder Average-Link.

Seien  $x, y$  zwei Dokumente und  $X, Y$  die jeweiligen Cluster dieser Dokumente, dann lässt sich die Distanz zwischen den Clustern  $X$  und  $Y$  mit Hilfe der Distanz zwischen den einzelnen Dokumenten errechnen.

Single-Link  $dist_{sl}(X, Y) = \min_{x \in X, y \in Y} dist(x, y)$

Complete-Link  $dist_{cl}(X, Y) = \max_{x \in X, y \in Y} dist(x, y)$

Average-Link  $dist_{al}(X, Y) = \frac{1}{|X| \cdot |Y|} \cdot \sum_{x \in X, y \in Y} dist(x, y)$

### 3. Web Document Clustering

Im Folgenden werden vier verschiedene Clusteringalgorithmen vorgestellt. Diese sind ein kosinusbasierter Group-Average-Clusteringalgorithmus, ein Word-Intersection-Clustering-Algorithmus und zwei verschiedene Phrase-Intersection-Clustering-Algorithmen.

#### 3.1. Group-Average-Clustering [6], [7]

Der kosinusbasierte Group-Average-Clusteringalgorithmus ist ein klassischer HAC Algorithmus. Als erstes werden alle Dokumente als einzelne Cluster verstanden und als „aktiv“ markiert. Die aktiven Cluster werden nach und nach zusammengefügt, bis nur noch ein einziges Cluster als aktiv markiert übrig ist. Die Auswahl, welche der vorhandenen aktiven Cluster im nächsten Schritt vereint werden, erfolgt aufgrund der Kosinus-Group-Average-Methode.

Jedes Dokument wird als Vektor dargestellt. Die Distanz zweier Vektoren kann mit der Formel  $dist(\vec{x}, \vec{y}) = 1 - \frac{\vec{x} \circ \vec{y}}{|\vec{x}| \cdot |\vec{y}|}$  errechnet werden. Anhand dieser Distanz wird nun mit der Average-Link-Methode, siehe Kapitel 2.2., die Distanz zwischen zwei Clustern bestimmt. Das neu gebildete Cluster wird als aktiv markiert, die beiden dafür verwendeten Cluster als „inaktiv“.

Das Ergebnis dieses Algorithmus ist ein einziges Clustering, aufgebaut als Binärbaum.

#### 3.2. Word-Intersection-Clustering [1]

Word-Intersection-Clustering, kurz Word-IC, ist ein HAC-Algorithmus, bei dem sich das Cluster aus allen Wörtern, die im Dokument vorkommen, zusammensetzt. Das Zusammenfügen der Cluster erfolgt mit Hilfe einer Qualitätsfunktion, der Global Quality Function (GQF). Die GQF dient zum Ermitteln der Qualität des Clusters. In jedem Schritt des Word-Intersection-Clusterings werden die zwei Cluster zusammengefasst, bei denen sich die größte Zunahme der GQF nach dem Zusammenfügen einstellt. Abbruchbedingung ist, wenn ein Zusammenfügen zweier Cluster die GQF nicht mehr erhöhen würde. Alle gemeinsamen Wörter dieses Clusterings stellen das *centroid* dar.

Zum Ermitteln der GQF benötigt man den Zusammenhang, cohesion  $h(c)$ , und den score  $s(c)$  eines einzelnen Clusters. Die cohesion  $h(c)$  stellt die Anzahl der Wörter, die alle Dokumente des Clusters gemeinsam haben, dar. Der score  $s(c)$  berechnet sich mit

Hilfe von  $h(c)$  folgendermaßen  $s(c) = |c| \cdot \frac{1 - e^{-\beta h(c)}}{1 + e^{-\beta h(c)}}$ . Im ersten Schritt, wenn jedes

Cluster aus einem einzigen Dokument besteht, wird der score  $s(c)$  als 0 definiert.

Der Faktor  $\beta$  stoppt den Anstieg der „sigmoid“-Funktion. Sein Wert kann als Stoppunkt angesehen werden, ab dem der score nicht mehr von der cohesion, sondern nur noch von der Größe des Clusters beeinflusst wird.

Um die Global Quality Function zu bestimmen, benötigt man eine Funktion  $f(C)$ , eine Funktion  $g(|C|)$  und die Summe der scores aller Cluster. Die Funktion  $f(C)$  steht dabei für den Anteil der geclusterten Dokumente. Mit abnehmender Anzahl der

Cluster steigt die Qualität, deshalb wird die von der Anzahl der Cluster abhängige Funktion  $g(|C|)$  im Nenner der Formel eingefügt.

$$GQF(C) = \frac{f(C)}{g(|C|)} \cdot \sum_{c \in C} s(c)$$

Somit lässt sich nun die Global Quality Function eines Clusters bestimmen und damit die Cluster, die im folgenden Schritt zusammengefasst werden.

Das folgende Bild stellt ein Dendrogramm dar. Hierbei werden die verschiedenen Ebenen der Clusterverschmelzung sichtbar. Dies ist ein wesentliches Merkmal hierarchischer Clusteringverfahren. Im Beispiel wurde anhand der Distanzfunktion Single-Link geclustert.

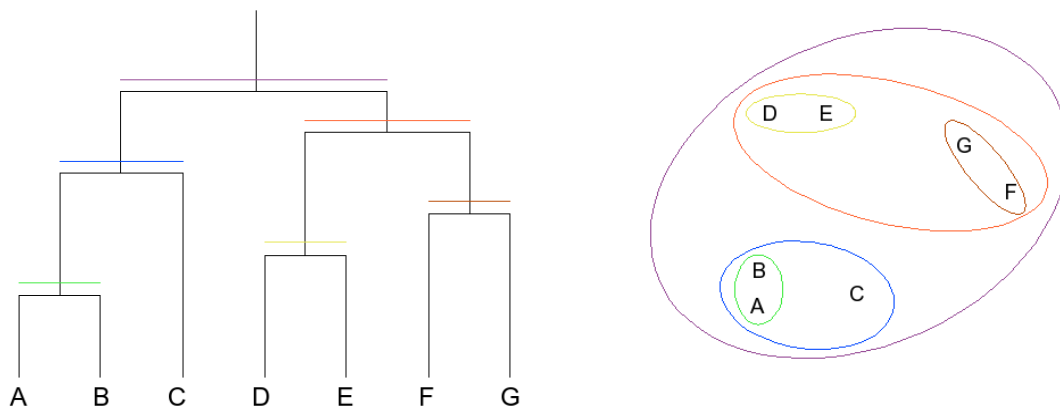


Bild 1 Beispiel eines Dendrogramms mit skizzierten Clustering für Word-Intersection-Clustering

### 3.3. Phrase-Intersection-Clustering [1], [2]

Bei Phrase-Intersection-Clustering werden, zusätzlich zu den Wörtern, wie bei Word-Intersection-Clustering, auch Phrasen, also Satzsequenzen und ganze Sätze, berücksichtigt. Diese in allen Dokumenten eines Clusters gemeinsamen Sequenzen stehen als Indikator der cohesion des jeweiligen Clusters.

Im Folgenden werden zwei verschiedene Vorgehensweisen erklärt. Als erstes Phrase-Intersection-Clustering mittels, der im vorherigen Kapitel erklärten, Global Quality Function (GQF), danach Phrase-Intersection-Clustering anhand von Suffixbäumen.

#### 3.3.1. Phrase-Intersection-Clustering mit GQF

Phrase-Intersection-Clustering mittels Global Quality Function ist ein HAC Algorithmus. Zur Ausführung wird, ebenso wie bei Word-Intersection-Clustering, die cohesion  $h(c)$ , der score  $s(c)$  und die GQF benötigt. Die cohesion  $h(c)$  steht hierbei jedoch nicht für die Anzahl der Wörter, sondern für die Länge der längsten gemeinsamen Phrase aller Dokumente des jeweiligen Clusters. Der score  $s(c)$  errechnet sich wie im vorherigen Kapitel dargestellt aus der cohesion.

Aufgrund der GQF, ebenfalls im vorherigen Kapitel näher erläutert, wird, in jedem Schritt erneut, entschieden, welche Cluster zusammengefügt werden.

Es besteht die Option, als Abbruchkriterium eine gewünschte Anzahl an Clustern anzugeben. Sobald diese Anzahl erreicht ist oder ein Zusammenfügen der verschiedenen Cluster die GQF nicht mehr erhöht, wird die Iteration abgebrochen.

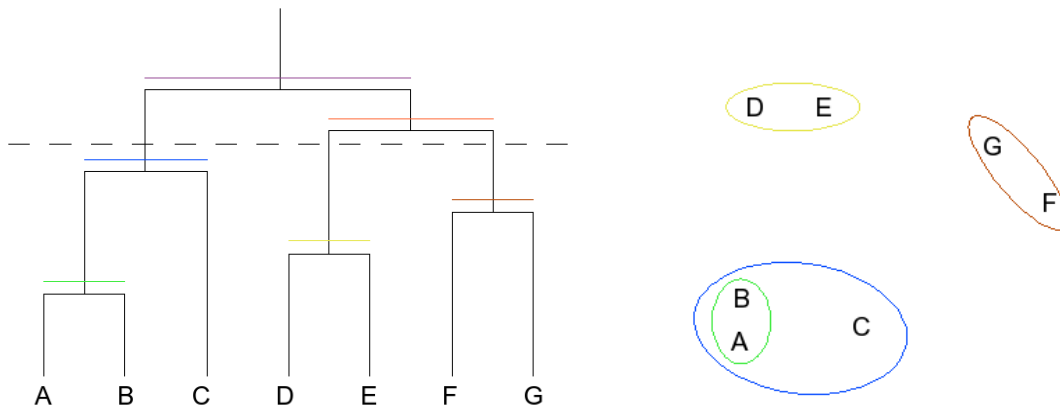


Bild 2 Beispiel eines Dendogramms mit skizziertem Clustering für Phrase-Intersection-Clustering

Bild 2 zeigt links das Dendogramm eines Clusterings, bei dem als Abbruchbedingung die Anzahl der Cluster auf drei gesetzt wurde. Die rechte Seite des Bild 2 stellt das skizzierte Clustering des Dendogramms dar. Geclustert wurde in diesem Beispiel anhand der Distanzfunktion Single-Link.

### 3.3.2. Phrase-Intersection-Clustering mit Suffixbaum [3]

Phrase-Intersection-Clustering mittels Suffixbäumen besteht aus drei Schritten. Im ersten Schritt wird das Originaldokument „gesäubert“, d.h. der String, bestehend aus Wörtern, wird umgeformt (z.B. Plural wird zu Singular, Satzenden werden markiert und Zeichen gekürzt). Das Originaldokument bleibt dabei vollständig erhalten. Von jedem Wort des transformierten Strings werden Verweise zur jeweiligen Position im Original hergestellt. Somit besteht die Möglichkeit, wenn auf Suchanfragen eine passende Phrase gefunden wird, dem User das jeweilige Originaldokument anzuzeigen.

Der zweite Schritt besteht darin, Basiscluster zu erkennen. Hierzu wird ein Suffixbaum erstellt, was nach Ukkonen (1995) in linearer Zeit möglich ist. Dieser Suffixbaum besteht aus Wörtern, nicht wie bisher aus Buchstaben. Da der Suffixbaum aus allen Dokumenten erzeugt wird, sind die Suffixe aller Dokumente bzw. Strings enthalten.

Ein Suffixbaum ist ein von der Wurzel aus gerichteter Baum, welcher keine leeren Kanten enthält. Jeder innere Knoten besitzt mindestens 2 Söhne. Ausgehende Kanten des selben Knotens beginnen stets mit unterschiedlichen Wörtern. Jeder Suffix bezeichnet exakt einen Weg. Das Label eines Knotens ist der Weg von der Wurzel bis zum jeweiligen Knoten. An jedem inneren Knoten trennen sich mindestens zwei verschiedene Suffixe. Gleiche Suffixe verschiedener Dokumente kommen im gleichen Blatt zusammen, wohingegen gleiche Satzsequenzen im gleichen Knoten zusammentreffen.

Ein Beispiel eines solchen Suffixbaumes wäre für die Strings „cat ate cheese“, „mouse ate cheese too“ und „cat ate mouse too“:

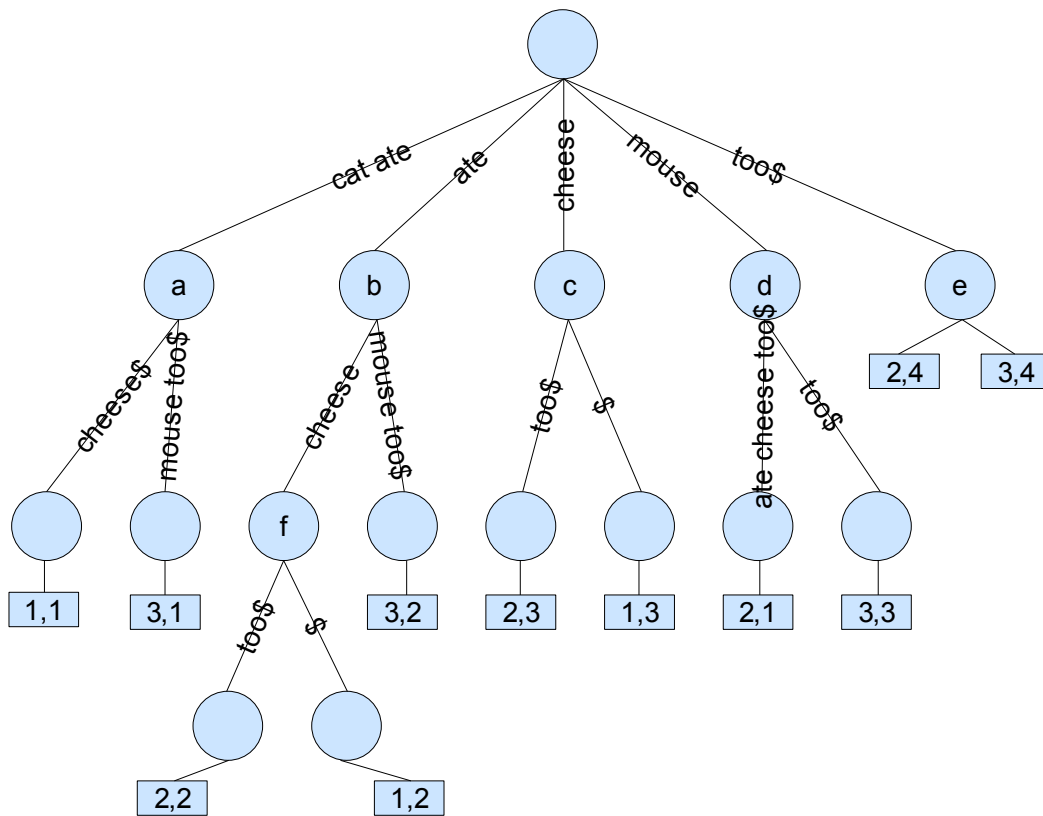


Bild 3 Suffixbaum der Beispielstrings

Die Kästchen an den verschiedenen Blättern stehen für die Verweise in das jeweilige Originaldokument. Die erste Zahl verweist auf das Dokument, die zweite zeigt das Startwort der Phrase im Original an.

Das kennzeichnende Merkmal eines Basisclusters ist, dass die Phrase, die dieser Knoten darstellt, Bestandteil verschiedener Dokumente ist. Im Beispiel Bild 3 sind die Knoten a bis f Basiscluster.

Knoten	Phrase	Dokument
a	cat ate	1,3
b	ate	1,2,3
c	cheese	1,2
d	mouse	2,3
e	too	2,3
f	ate cheese	1,2

Tabelle 4 Basiscluster des Beispiels

Tabelle 4 stellt die sechs Basiscluster des Suffixbaumes aus Bild 3 dar. Die Phrasen sind dabei die vom jeweiligen Knoten dargestellten Phrasen. Dokument zeigt die verschiedenen Dokumente, in denen diese Phrase vorkommt.

Anhand dieser Tabelle lässt sich der score eines Basisclusters errechnen. Der score  $s(B)$  setzt sich dabei aus der Anzahl der Dokumente, die das Basiscluster beinhaltet, und der dargestellten Phrase zusammen. Aus Basiscluster B und Phrase P wird der score  $s(B)$  durch  $s(B) = |B| \cdot f(|P|)$  definiert. Die Funktion  $f$  benachteiligt Phrasen die nur aus einem Wort bestehen, ist linear für Phrasen der Länge zwei bis sechs Wörter und wird konstant für Phrasen länger als sechs Wörter.

Der dritte und letzte Schritt besteht darin, die verschiedenen Basiscluster zu kombinieren. Dabei ist die Ähnlichkeit der zu kombinierenden Basiscluster ausschlaggebend. Seien  $B_m$  und  $B_n$  zwei Cluster, deren jeweilige Größe mit  $|B_n|$  und  $|B_m|$  bezeichnet wird. Die Anzahl der Dokumente, die in beiden Clustern gleich sind, ist  $|B_m \cap B_n|$ . Um ein Maß für die Ähnlichkeit der beiden Cluster zu bekommen, definiert man die Ähnlichkeit als 1, wenn  $|B_m \cap B_n|/|B_m| > 0,5$  und  $|B_m \cap B_n|/|B_n| > 0,5$  erfüllt ist. Ansonsten definiert man die Ähnlichkeit als 0.

Als Nächstes muss die paarweise Ähnlichkeit der Basiscluster errechnet werden, damit ein Basisclustergraph erzeugt werden kann. Für unser Beispiel aus Bild 3 ergeben sich folgende Ähnlichkeiten.

	a	b	c	d	e	f
a	-	1	0	0	0	0
b	1	-	1	1	1	1
c	0	1	-	0	0	1
d	0	1	0	-	1	0
e	0	1	0	1	-	0
f	0	1	1	0	0	-

Tabelle 5 Ähnlichkeit der Basiscluster

Anhand dieser Tabelle ergibt sich folgender Basisclustergraph. Die Knoten stehen dabei für die ermittelten Basiscluster. Eine Kante zwischen zwei Basisclustern bedeutet, dass die Ähnlichkeit, anhand Tabelle 5, als 1 ermittelt wurde.

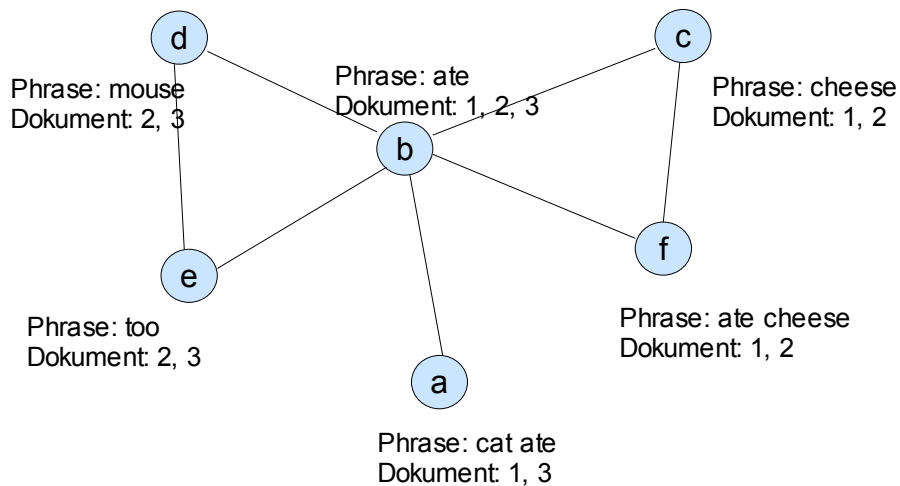


Bild 6 Basisclustergraph

Anhand des dargestellten Basisclustergraphs lässt sich erkennen, dass in diesem Beispiel alle Dokumente in einem einzigen Clustering zusammengefasst werden.

Wenn nun „ate“ auf einer Stopliste, einer Liste der nicht zu beachtenden Wörter wäre, dann würde das Basiscluster b wegfallen und damit auch jede Verbindung zu b. Somit sähe der Basisclustergraph folgendermaßen aus.

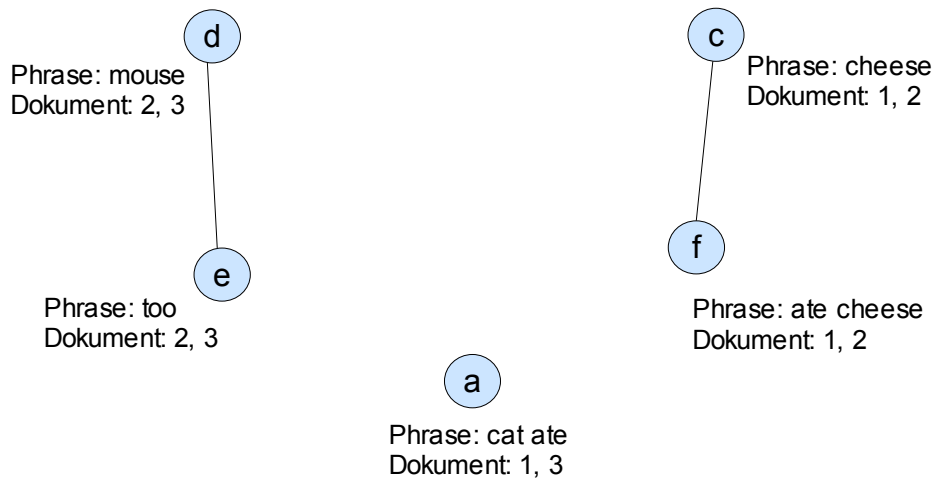


Bild 7 Basisclustergraph, wenn „ate“ auf der Stopliste wäre

Der dargestellte Basisclustergraph besteht nun nicht mehr nur aus einem einzigen Clustering, sondern aus drei voneinander unabhängigen.

Diese fertigen Cluster werden anhand der errechneten scores der jeweiligen Basiscluster und ihrer Überlappung sortiert und aufgelistet. Da die Anzahl der Cluster variiert, werden nur die höchsten aufgelistet. Normalerweise sind nur die ersten 10 Cluster von Interesse.

Wenn ein neues Dokument zu einem Cluster hinzugefügt wird, könnte sich damit das jeweilige Cluster verändern und somit auch die Basiscluster. Um einen korrekten Basisclustergraph zu erhalten, müsste eigentlich ein erneuter Vergleich mit allen anderen Basisclustern erfolgen. Doch darauf wird verzichtet. Stattdessen findet nur ein Vergleich mit den k ranghöchsten Clustern statt, da dies die relevanten und damit wichtigen Cluster sind.

## 4. Vergleich

In diesem Kapitel werden die Clusteringverfahren Group-Average-Clustering, Word-Intersection-Clustering mittels GQF, Phrase-Intersection-Clustering mittels GQF und Phrase-Intersection-Clustering mittels Suffixbaum verglichen. Vergleichsmerkmale werden die Qualität der erzeugten Cluster und die benötigte Zeit sein.

### 4.1. Versuchsbedingungen [1]

Für den Vergleich wurden 88 Anfragen bei MetaCrawler, mit jeweils 1 bis 4 Schlüsselwörtern, gestellt. Aus den Resultaten wurden 88 Basis-Erfassungen, im folgenden mit B bezeichnet, aufgestellt. Jede dieser Erfassungen beinhaltet circa 100 Schnipsel. Jeder Schnipsel besteht aus circa 40 Wörtern. Für den Test wurden jeweils 1 bis 8 Basis-Erfassungen gemischt, was jeweils 100 bis 800 Schnipsel ergab. Dieses Clustering wird im folgenden als C bezeichnet. Von jeder dieser Größen wurden 20 Clusterings erstellt.

### 4.2. Qualität der Clusterings

Um ein Maß für die Qualität zu erhalten, werden nun die vom Algorithmus erzeugten Cluster mit den Basisclustern der Versuchsreihe verglichen. Zuerst werden alle Paare eines Dokuments betrachtet und auf true-positive Paare, Dokumente die aus dem sel-

ben Basiscluster stammen, bzw. false-positive Paare, Dokumente die nicht aus dem selben Basiscluster stammen, untersucht. Die Werte werden in  $t(c)$  und  $f(c)$  gespeichert. Weiterhin wird die Anzahl der nicht geclusterten Dokumente  $u(C)$  des Clusterings  $C$  benötigt.

Die Qualität eines Clusterings lässt sich nun wie folgt berechnen

$$Q(C) = \frac{\sum_{c \in C} (\sqrt{t(c)} - \sqrt{f(c)}) - u(C)}{\sum_{c \in B} \sqrt{\binom{|c|}{2}}}$$

Die Qualität  $Q(C)$  nimmt einen Wert zwischen -1 und 1 an, wobei ein Wert zwischen -1 und 0 bedeutet, dass mehr Dokumente im Cluster falsch als richtig sind. Demzufolge ist ein Ergebnis zwischen 0 und 1 wünschenswert, je größer, um so besser.

Die Ergebnisse für Group-Average Clustering, Word-Intersection-Clustering, Phrase-Intersection-Clustering mit GQF und Phrase-Intersection-Clustering mit Suffixbäumen ergeben das folgende Schaubild.

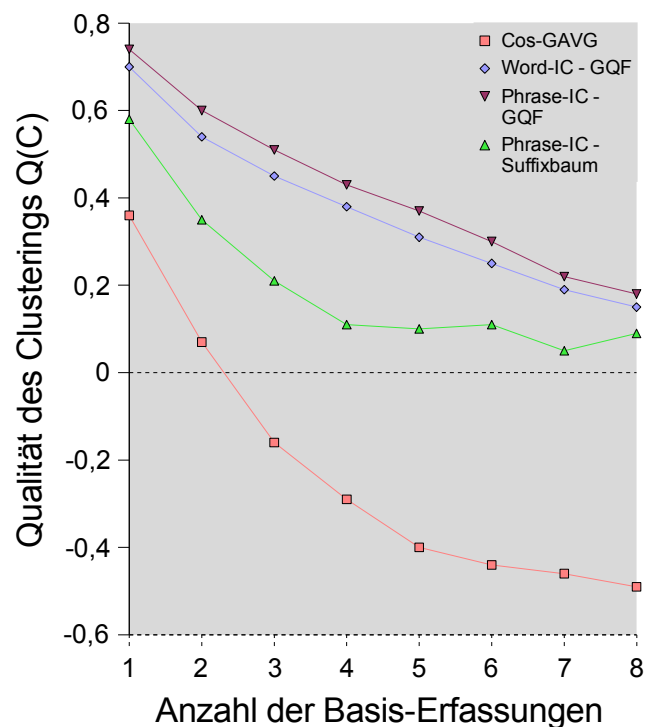


Schaubild 8 Vergleich der Qualitäten eines erzeugten Clusterings

#### 4.3. Zeiten der verschiedenen Algorithmen

Um ein Vergleichsmaß für die benötigte Zeit zu erhalten, wurden die vier Algorithmen in C++ implementiert. Es wurden Suchanfragen gestellt und die benötigte Zeit gemessen, wobei die Wartezeit des Systems, auf Daten aus dem Netz, nicht berücksichtigt wurde. Die dabei jeweils gemessene Zeit wurde in folgendem Schaubild dargestellt.

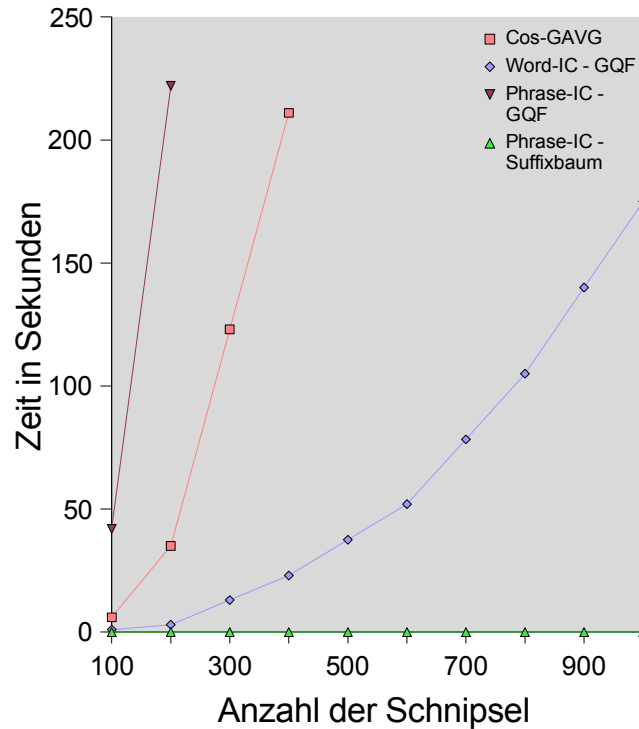


Schaubild 9 Gemessene Zeit der verschiedenen Algorithmen

Die Zeit für Phrase-Intersection-Clustering mittels Suffixbaum ist nicht darstellbar, da eine Suche in 1000 Schnipseln in weniger als 0,25 Sekunden erfolgt.

#### 4.4. Ergebnis des Vergleichs

Der durchgeführte Vergleich bezüglich Qualität und Zeit – dargestellt in den Schaubildern 8 und 9 – kann grundsätzlich in folgender Form zusammengefasst werden:

	Qualität	Zeit
Cos-GAVG	gering	hoch
Word-IC – GQF	sehr hoch	mittel
Phrase-IC – GQF	sehr hoch	sehr hoch
Phrase-IC – Suffixbaum	hoch	sehr gering

Tabelle 10 Vergleich der Clusteringalgorithmen

Dabei ergeben sich, unter der Anforderung hohe Qualität bei geringer Zeit zu erhalten, folgende Ergebnisse. Group-Average-Clustering – als klassischer HAC-Algorithmus – schneidet in beiden Kriterien schlecht ab. Die drei anderen Algorithmen haben Stärken und Schwächen.

Word-Intersection-Clustering und Phrase-Intersection-Clustering mittels GQF erzeugen im Vergleich die qualitätsmäßig besten Cluster. Hinsichtlich der Zeit sind beide Algorithmen jedoch unteres Mittelfeld. Phrase-Intersection-Clustering mittels Suffixbaum läuft sehr schnell, bietet jedoch nur eine durchschnittliche Qualität.

## 5. Zusammenfassung und Fazit <sup>[9]</sup>

Der Zielstellung eines möglichst schnellen und qualitativ guten Ergebnisses bei Suchanfragen in einer Menge von Dokumenten kann man mit Hilfe von Clustering recht gut entsprechen. Dabei bieten die betrachteten Verfahren zum Clustering auf Basis von Wörtern – Group-Average-Clustering und Word-Intersection-Clustering – und zum Clustering auf Basis von Satzsequenzen und Sätzen – Phrase-Intersection-Clustering mittels GQF oder Suffixbaum – verschiedene Ansätze und unterschiedliche Leistungsparameter.

Ein Vergleich der vier vorgestellten Clusteringverfahren zeigt, dass die Algorithmen, basierend auf Word- bzw. Phrase-Intersection-Clustering, ein qualitätsmäßig wesentlich besseres Ergebnis erzeugen.

Basierend auf der vorangegangenen Ausarbeitung ist Phrase-Intersection-Clustering mittels Suffixbaum der vorteilhafteste Algorithmus. Er arbeitet sehr schnell, da bereits mit der Abarbeitung des Algorithmus begonnen werden kann, während noch weitere Dokumente aus dem Internet geladen werden. Ein weiterer Vorteil ist, dass ein Dokument in mehreren Clustern vorkommen kann, und nicht auf ein Gebiet beschränkt ist. Somit erfüllt Phrase-Intersection-Clustering viele, der in der Einführung genannten, Anforderungen an einen guten Clusteringalgorithmus.

## Quellenangabe

- [1] Oren Zamir, Oren Etzioni, Omid Madani, Richard M. Karp. Fast and Intuitive Clustering of Web Documents. University of Washington, 1997
- [2] Oren Zamir, Oren Etzioni. Web Document Clustering: A Feasibility Demonstration. University of Washington
- [3] Esko Ukkonen. On-line construction of suffix-trees. *Algorithmica*, 14: 249-260, 1995
- [4] Internet Suchmaschinen. Universität Kassel Homepage. [http://www.kde.cs.uni-kassel.de/lehre/ws2005-06/IR/fohlen/4Folie\\_15\\_TextClustering.pdf](http://www.kde.cs.uni-kassel.de/lehre/ws2005-06/IR/fohlen/4Folie_15_TextClustering.pdf)
- [5] Dr. Matthias Schubert. Knowledge Discovery in Databases. Ruprecht-Karls-Universität Heidelberg Homepage. <http://www-dbs.informatik.uni-heidelberg.de/teaching/ss2005/kdd/skript/kdd-2-clustering2.pdf>
- [6] Mark Dunlop. Development and evaluation of clustering techniques for finding people. Centre for Human-Machine Interaction. [http://www.cis.strath.ac.uk/~mdd/research/publications/00dunlop\\_b.pdf](http://www.cis.strath.ac.uk/~mdd/research/publications/00dunlop_b.pdf)
- [7] Lev Reyzin. Online Clustering of Linguistic Data. Junior Independent Work. <http://www.cs.princeton.edu/cluster/pub/lreyzin.pdf>
- [8] Ludwig-Maximilians-Universität München. Web Document Clustering, Projektarbeit. [http://www.dbs.informatik.uni-muenchen.de/Forschung/KDD/Clustering/student\\_work/docs/WebClusteringDist.html](http://www.dbs.informatik.uni-muenchen.de/Forschung/KDD/Clustering/student_work/docs/WebClusteringDist.html)
- [9] Heiko Fröhlich. Suchmaschinen. Ausarbeitung zum Proseminar Text Mining, Universität Ulm, 2005. <http://www.informatik.uni-ulm.de/ni/Lehre/SS05/ProseminarTextMining/ausarbeitungen/Froehlich.pdf>
- [10] Elke Achtert. Inkrementelles hierarchisches Clustering. Diplomarbeit, Ludwig-Maximilians-Universität. <http://www.dbs.informatik.uni-muenchen.de/~achtert/papers/da.pdf>