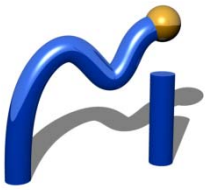
Andreas Henrich
Daniel Blank (Hrsg.)

Themen des Information Retrieval

Suchmaschinen und Web-Suche

Beiträge des Seminars

im Sommersemester 2012



Lehrstuhl für Medieninformatik

Otto-Friedrich-Universität Bamberg

Inhaltsverzeichnis

Designing Search User Interfaces

Evaluierung im Information Retrieval

Software-Bibliotheken für das Information Retrieval

Aspekte des Enterprise Search

The Design of Search User Interfaces

Eduard Anton

Otto-Friedrich-Universität-Bamberg, Lehrstuhl für Medieninformatik
eduard.anton@stud.uni-bamberg.de

Zusammenfassung. Eine Suchmaschine muss eine enorme Anzahl von Use Cases, über eine enorme Ansammlung von Informationen, benutzt von einem weiten Feld von Menschen unterstützen und muss dabei ein hohes Maß an Gebrauchstauglichkeit unterstützen. Die Gebrauchstauglichkeit und das Interface Design gehen dabei Hand in Hand. Die Wissenschaft, die sich damit befasst, nennt man Mensch-Computer-Interaktion. Ziel dieser Arbeit ist es, die Schwierigkeiten von Suchmaschinennutzern mit dem Interface Design von Suchmaschinen aufzuzeigen und die von der Mensch-Computer-Interaktion entwickelten Richtlinien für ein erfolgreiches Interface Design vorzustellen, um diese Schwierigkeiten zu lösen und an Beispielen in der Praxis zu zeigen. Ausgangspunkt dieser Arbeit stellt das erste Kapitel vom Buch *Search User Interfaces* von Marti A. Hearst [1] dar.

Schlagworte: Mensch-Computer-Interaktion, Gebrauchstauglichkeit, Design Richtlinien

1 Einleitung

Berühmte Persönlichkeiten, Preisvergleiche oder Informationen für ein bestimmtes Hausarbeitsthema - in der heutigen Gesellschaft ist die Informationsbeschaffung ein wichtiges Thema. Wo man früher in Bibliotheken, Büchereien und Zeitungsarchiven suchte, recherchiert man heute mit Suchmaschinen im Internet. Millionen von Menschen nutzen dabei täglich unterschiedliche Suchmaschinen. Um die Informationen zugänglich und nutzbar zu machen, braucht man effektive und effiziente Suchmaschinen. Dabei spielt das Design von Suchinterfaces eine wichtige Rolle, es „verwandelt unstrukturierte Rohdaten in brauchbare Informationen und beschreibt die Form der dargestellten Informationen. Es hat Einfluss darauf, wie die Nutzer weitere Informationen verarbeiten sowie rezipieren und trägt dazu bei, ob das Produkt wieder verwendet wird. Bei einem positiven Erlebnis kommen die Nutzer gerne wieder“ [2, S. 228].

Die Nutzung eines Suchinterfaces bringt aber auch Probleme mit sich. Vor allem unerfahrene User, ältere Menschen und Kinder haben Schwierigkeiten bei ihrer Informationssuche im Internet. Diese Schwierigkeiten führen darauf zurück, dass Menschen bei ihrer Informationssuche, bevor die Verfügbarkeit von Computern und

Internet so weit verbreitet war wie sie heute ist, sich andere Menschen zu Nutze machten. So fragte man Bibliothekare, wo eine bestimmte Information zu finden sei. Diese verstanden, was gesucht wird, obwohl bei der Frage vielleicht nicht die richtige Syntax oder richtigen Begriffe genannt wurden. Jetzt müssen Leute mit Suchmaschinen interagieren, obwohl sie sich mit moderner Informationsbeschaffung nur wenig oder gar nicht auskennen(vgl. [3, S. 1-2]). Um diese Interaktion für alle zu ermöglichen, ist es wichtig, das Interface möglichst zu simplifizieren. Dies wird im nächsten Kapitel näher erläutert.

2 Das Interface einfach halten

Nahezu bei jeder Suchmaschine tippt man ein Stichwort in ein Sucheingabefeld ein und die Ergebnisse werden in einer vertikalen Liste ausgegeben. Dabei hat sich auch in den letzten Jahren nicht viel verändert. Vergleicht man eine Informationssuche, mit dem gleichen Suchbegriff von 1997,2007 und heute, stellt man fest, dass keine größeren, gravierenden Veränderungen im Interface bestehen und es relativ einfach gehalten wird (siehe Abbildung 1).

Abbildung 1. Ergebnislisten (links von 1997, rechts von 2007) einer Googlesuche mit dem Suchbegriff „darter habitat“ [1]

Warum verändert sich das Standardinterface kaum? Gründe hierfür liegen beim Zweck der Suche. Bei einer Suche möchte der Nutzer seinem Informationsbedürfnis nachgehen. Dabei wirkt ein überladenes und aufdringliches Interface störend. Außerdem lenkt ein solches Interface beim Lesen ab, was ohnehin eine mental sehr

intensive Tätigkeit ist, die kaum multitaskfähig ist, und deshalb die volle Aufmerksamkeit des Nutzers erfordert. Außerdem soll die Simplizität des Interfaces auch für die bereits angesprochenen Menschen, die Problemen bei der Bedienung mit Suchmaschinen haben, verständlich sein.

Trotz des ohnehin einfach gehaltenen Interfaces haben Studien gezeigt, dass eine weitere Vereinfachung viele gemachte Fehler reduzieren würde. Fehler sind beispielweise die falsch angewandte Syntax in der Suchzeile, da Suchzeile und Adresszeile miteinander verwechselt werden, die Missinterpretation der Boolean Operatoren und die falsche Verwendung der Begriffreihenfolge in der Abfrage. In einem Versuch, die Websuche für ältere Leute zu vereinfachen, wurde ein Suchinterface gestaltet, das viele gebräuchliche Fehler reduzieren soll. Dieses Interface hat eine größere Suchleiste für die Abfragen und mehr Platz zwischen den Buchstaben, um es einfacher zu machen, Abfragen zu bearbeiten. Außerdem wurde ein Button hinzugefügt, der die Suchleiste leert, um neue Abfragen schneller starten zu können. Andere Vereinfachungen an der Ergebnisliste wurden zusätzlich vorgenommen, um die Resultate besser zu verstehen. Viele Nutzer dieses Versuchs verstanden die Funktionalität und schätzten die Klarheit (vgl. [3, S. 62-63]).

Man sieht also, dass eine weitere Vereinfachung durchaus möglich und sinnvoll ist, was auch das Scheitern mancher Versuche, komplexere Interfaces einzuführen, erklärt.

3 Suchinterfaces vor und nach dem Web

Modernes Information Retrieval, wie wir es heute haben, hat „vom Memex über den Sputnikschock zum Weinberg-Report" [8, S. 38] einen langen Weg hinter sich. Bevor das Internet ein weltweites Phänomen wurde, mussten die Menschen zum Lexikon greifen oder in anderen Büchern nachschlagen, um an ihre gewünschten Informationen zu gelangen. Die computergestützte Informationsbeschaffung war üblicherweise ein Privileg von ausgebildeten Nutzern wie Rechtsanwaltfachangestellten, Journalisten und Bibliothekare. Sie suchten in bibliografischen Datensätzen, Rechtsfällen oder Nachrichtensammlungen. Normalerweise wurde offline nach dem Ort der Quelle gesucht und anschließend wurde eine Papierkopie gemacht. Heute findet man sämtliche Texte direkt im Internet und der Text ist somit sofort verfügbar und lesbar. Ein weiterer Unterschied besteht darin, dass alte Systeme, bevor Grafikdisplays üblich waren, auf Kommandozeilen-Interfaces basiert haben. Das hatte zufolge, dass komplexe Kombinationen von Operatoren gemerkt werden mussten und ein Verständnis für Boolean Operatoren mit sich gebracht werden musste. Das Verständnis für Boolean Operatoren und Kommandosprache war nur einer kleinen Anzahl von Leuten bekannt und so gab es nur wenige, die Zugang zu den Inhalten hatten. Ein letzter Unterschied besteht darin, dass damals Suchmaschinen entweder nach Zeit, Anzahl der Abfragen oder Anzahl der Ergebnisse, Geld gekostet haben. Heute sind Suchmaschinen durch den großen Wettbewerb meistens kostenlos.

Diese Gegensätze helfen dabei, die Unterschiede der Suchinterfaces vor und nach dem Web zu verstehen.

4 User Interfaces in der Mensch-Computer Interaktion

Die Mensch-Computer Interaktion (Human-Computer Interaction) ist die Wissenschaft, die sich mit dem Interface Design und der Kommunikation zwischen Mensch und Computern auseinandersetzt. Das UCD (User Centered Design) ist eine Entwicklung der Mensch-Computer Interaktion. Es ist ein „ Vorgehensmodell zur benutzerzentrierten Gestaltung interaktiver Systeme“ [2, S. 226-227]. Das bedeutet, bei der Entwicklung einer Suchmaschine wird der zukünftige Nutzer mit seinen Aufgaben, Zielen und Eigenschaften in den Mittelpunkt des Entwicklungsprozesses gestellt. Dies wird durch ein iteratives Vorgehen über mehrere Phasen erreicht. Dadurch erhofft man sich die Entwicklung einer Suchmaschine, die über eine hohe Gebrauchstauglichkeit (Usability) verfügt. Die Gebrauchstauglichkeit ist eine qualitative Eigenschaft eines User Interface und mit dem Konzept der Benutzerfreundlichkeit vergleichbar. Die Gebrauchstauglichkeit besitzt bestimmte Attribute, die lauten:

- Erlernbarkeit: wie schnell und einfach ein Nutzer in einem neuen System produktiv arbeiten kann und wie leicht es ist sich die Funktionsweise des Systems zu merken
- Nutzungseffizienz: die Anzahl der Aufgaben pro Zeiteinheit, die der Nutzer mit dem System ausführen kann
- Zuverlässigkeit: Wie oft kommen beim Benutzen des Systems Fehler vor und wie lange dauert es sie zu beheben
- Zufriedenheit: Wie zufrieden ist man mit dem System

Man kann diese Eigenschaften, als Anforderungen (Requirements) der Suchmaschine betrachten. Um diese Eigenschaften gut umzusetzen und ein erfolgreiches User Interface Design zu entwickeln, gibt es in der Mensch-Computer Interaktion mehrere Richtlinien, an die man sich halten sollte:

- Bieten von informativen Feedback
- Userkontrolle und automatisierte Aktionen
- Kurzzeitgedächtnis weniger belasten
- Shortcuts anbieten
- Reduzieren von Fehlern
- Streben nach Konsistenz
- Das Rückgängig machen von Aktionen ermöglichen
- Design

In den Folgenden Kapiteln werden einzelne Richtlinien aufgegriffen und gezeigt, wie sie in der Praxis angewandt werden.

5 Bieten von informativen Feedback

Der Begriff Feedback fasst die Rückmeldung der Suchmaschine an den Nutzer zusammen. Gerade im Information Retrieval ist Feedback ein wichtiger Faktor, um die Qualität der Suchmaschine zu erhöhen. Feedback über die Formulierung der Abfragen und beim Anzeigen der Ergebnisse unterstützen den Nutzer bei der Websuche. Eine Art des Feedbacks ist das sofortige Anzeigen von Ergebnissen, noch während der Eingabe. So sieht der Suchende, ob er auf der richtigen Spur ist und unterstützt ihn mit Vorschlägen von verwandten Wörtern, die er möglicherweise für eine Neuformulierung der Abfrage benutzen kann. Vorschläge von verwandten Wörtern, die auch eine Rechtschreibverbesserung beinhaltet stellt eine weitere Methode des Feedbacks dar. Neuere Suchmaschinenversionen generieren die Begriffsvorschläge dynamisch beim Tippen. Viele Suchmaschinen machen den Fehler, dass sie dem Nutzer mit Rechtschreibverbesserungen und anderen Hilfsfunktionen überladen, noch bevor er ein Ergebnis angesehen hat. Abbildung 2 zeigt ein effektives und dynamisches Feedback. Noch während der Eingabe werden Ergebnisse zu einem Onlineshop angezeigt, und gleichzeitig Vorschläge für die Abfrage generiert.

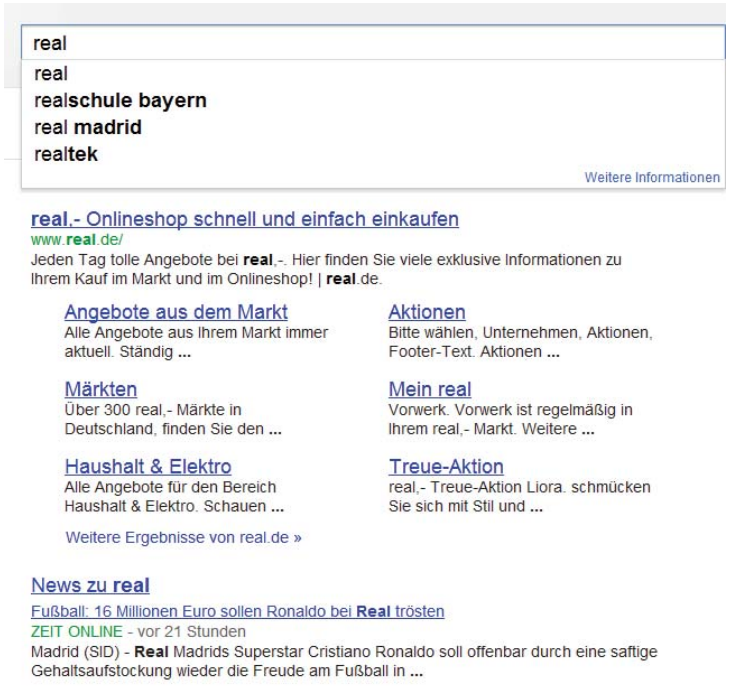


Abbildung. 2. Effektives und dynamisches Feedback einer Googlesuche[1]

Suchergebnisse werden in den meisten Suchmaschinen in einer vertikalen Liste ausgeben. Jedes Ergebnis besitzt Angaben über den Titel, die URL und eine kleine textuelle Zusammenfassung. Zusammen bilden diese Informationen das sogenannte Document Surrogate. Eine wichtige Form des Feedbacks ist, dass die Begriffe der Abfrage in dem Document Surrogates erscheinen. Eine Hervorhebung der Begriffe, z.B. fett markiert, unterstützt das ganze zusätzlich. Ältere Versionen von Suchmaschinen nahmen bei den textuellen Zusammenfassung des Document Surrogates die ersten paar Zeilen des Dokuments. Studien haben aber gezeigt, dass es informativer ist, wenn die Abfragebegriffe im Kontext, wie sie im Dokument auftauchen angezeigt werden (vgl. Abbildung 3).

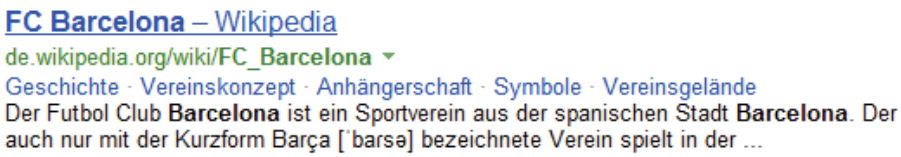


Abbildung 3. Document Surrogate der Bing-Suchmaschine[2] mit Titel, URL, Shortcuts und textueller Zusammenfassung (Suchbegriff: „FC Barcelona“)

Gewöhnlich muss ein Nutzer die textuellen Zusammenfassungen einzeln und in einer gewissen Weise vorgesehenen Reihenfolge durchlesen oder überfliegen, um Aussagen über die Relevanz der Document Surrogates für eine Informationssuche machen zu können. Eine schnelle Evaluation ist dadurch nur bedingt möglich. Die HotMap Search-Engine, geht deshalb einen anderen Weg, indem es visuelle Darstellungen der Suchergebnisse und weitere Extras für die interaktive Betrachtung der Suchergebnisse bereitstellt. Gewöhnliche Suchmaschinen sind sehr effektiv beim Finden von sehr spezieller Information. Jedoch ist es für Nutzer üblich, dass sie kein klares Verständnis dafür haben, wonach sie suchen. In diesen Fällen hilft das interaktive visuelle Interface, wie das von HotMap(vgl. [4, S. 1-2, S. 8-9]). Listenbasierte Ergebnisse geben normalerweise die Reihenfolge der Ergebnisse vor und der Nutzer handelt sie für gewöhnlich von oben nach unten ab. Die Reihenfolge wird nach Relevanz sortiert. In älteren Suchmaschinenversionen wurde die Relevanz mit Hilfe von Indikatoren, an der Seite von den Document Surrogates dargestellt. Meistens geschah das in Form von Sternen. Es hat sich aber gezeigt, dass diese Methode überholt ist und nur sehr sparsam mit solchen Indikatoren umgegangen werden sollte. Die Relevanz rein durch die Platzierung möglichst weit oben auf der Ergebnisliste, ist schon sehr aussagekräftig, denn die Relevanz ist ein Qualitätskriterium, das die Nutzer sofort erkennen. Die Nutzer schauen immer zuerst in den oberen Bereich und folgen immer einem gleichen Schema. Es wird immer vom ersten Ergebnis ausgegangen, dann wird der Fokus nach rechts und listenförmig nach

[1] www.google.de

unten versetzt. Das ganze bildet ein Dreieck, das auch „Golden Triangle“(Abbildung 4) genannt wird (vgl. [2, S. 224-225]). Solche Analysen wurden mithilfe von Eyetracking gemacht, was eine Form des impliziten Feedbacks darstellt. Implizites Feedback ist die Rückmeldung des Benutzers an Systeme, mit deren Hilfe Suchmaschinenentwickler ihr Interface Design ausrichten können und eine qualitativ gute Suchmaschine entwickeln können.

Die Anordnung der Suchergebnisse nach Relevanz, der sogenannten Rangordnung der Websuche, wird im folgenden Kapitel wieder aufgegriffen.

Abbildung. 4. „Golden Triangle“ [2, S. 224]

6 Userkontrolle und automatisierte Aktionen

Eine wichtige Richtlinie ist es, ein gesundes Gleichgewicht zwischen der Userkontrolle, also dem was der User steuert und automatisierten Aktionen, die von der Suchmaschine übernommen werden zu finden. Die bereits angesprochene Rangordnung dient der Anordnung nach der Relevanz von Dokumenten einer Websuche. Doch wie genau der Mechanismus funktioniert, dass eine bestimmte Anordnung stattfindet ist den meisten Menschen nicht bekannt. Es ist einer dieser automatisierten Mechanismen, die im Hintergrund ablaufen. Der Algorithmus der dabei verwendet wird, ist meist sehr komplex und kompliziert und bei vielen Providern auch ein streng bewachtes Geheimnis. Selbst bei fehlerhaften und lückenhaften Eingaben, kommen wie von Geisterhand, brauchbare Dokumente in den Ergebnislisten vor, die der Intention des Suchenden entsprechen. Das was an Transparenz diesem Mechanismus fehlt, muss durch gute und relevante Ergebnisse der Suchmaschine ausgeglichen werden.

Ein weiterer automatisierter Mechanismus ist die Abfrage-Transformation. Kleinere Änderungen, wie die von der Microsoft Websuche, die das eingegebene „vs.“, automatisch in „versus“ transformiert sind nicht sehr gravierend, wenn man den Aspekt der Kontrolle über das System betrachtet. Die Suchmaschine verändert nicht den Suchenden seine Intention und solange diese gewahrt bleibt, sind solche Transformationen sehr hilfreich. Ein anderer Aspekt ist das automatische Entfernen von Stopwords in der Abfrage. Als Stopwords bezeichnet man sehr häufig vorkommende Wörter, die als unwichtig gesehen werden. Solche Wörter sind meistens Artikel oder Präpositionen, können sich aber je nach Datenbank auf der man sucht unterscheiden. Dieser Automatismus birgt aber Gefahren, denn eine Suchabfrage nach „Alexander der Große“ hat eine andere Bedeutung als „Alexander“ und „Groß“. Deshalb muss die Suchmaschine fähig sein, bestimmte Phrasen als Ganzes zu erkennen (vgl. [5, S. 3]). In vielen Fällen schreibt ein Nutzer einer Suchmaschine eine natürliche Frage in ein Suchfeld. Typische Suchmaschinen interpretieren die Frage, als eine Liste von Begriffen und erstellen auf dieser Basis ihre Ergebnisliste. Die Frage „What is a hard disc?“, würde bei den meisten Suchmaschinen Ergebnisse mit Herstellern von Festplatten ausgeben, weil die Suchmaschine die Abfrage nicht als Frage wahrnimmt. Gesucht wäre bei dieser Frage vielmehr eine Definition von Festplatten und wozu sie gebraucht werden, auf Englisch „used to“. So ist eine Suchmaschine, die diese Frage automatisch in die Abfrage {„hard disc“ NEAR „used to“} verwandelt, eine die dem Nutzer sehr hilfreich wäre. Ein solches System ist „Tritus“, das natürlich gesprochene Fragen versucht direkt zu beantworten (vgl. [6, S. 1]). Eine weitere Abfrage Transformation ist die automatische Rechtschreibkorrektur, die dem Nutzer nicht aufgezwängt werden sollte, sondern er die freie Wahl hat, ob er diese nutzen möchte. Jedoch ist hier wieder ein Gleichgewicht zu finden zwischen zu viel und zu wenig Spielraum, denn vertippt sich ein Nutzer gravierend, ist es oft ärgerlich keine brauchbaren Informationen als Ergebnis zu bekommen.

7 Das Kurzzeitgedächtnis weniger belasten

Diese Interface Richtlinie zielt darauf ab, wichtige Informationen dem Nutzer direkt anzuzeigen. Die Informationen sollen direkt in den Fokus fallen und der Suchende soll sie sich nicht merken müssen, oder im Auge behalten müssen.

Abbildung. 5. Dogpile Web Search[2]

Abbildung 5 zeigt die Dogpile Suchmaschine. Im Suchfeld ist in ausgegrauter Schrift, das Wort „Search“ zu erkennen. Es zeigt an, dass dies das Suchfeld ist, das verwendet werden muss, um eine Suchabfrage zu tätigen. Der Text ist in ausgegrauter Schrift, um anzudeuten, dass er durch den Text des Nutzers ausgetauscht wird. Für erfahrene Nutzer erscheint es banal und logisch, dass in diesem Feld die Begriffe eingegeben werden sollen, allerdings gibt es viele, vor allem alte Menschen mit wenig Erfahrung, denen eine solche Andeutung weiter hilft. Ein weiterer Aspekt ist das Anzeigen des Suchverlaufs, also eine Dokumentation der gemachten Suchabfragen und angesehenen Dokumente.

[2] www.dogpile.com

Fig. 6. PubMed[3] Suchverlauf

Gefundene Informationen wiederfinden ist oftmals ein Problem. Wie wurden die Webseiten gefunden? Eine Frage, die sich viele Menschen häufig stellen. Manche schicken sich die URL per Email, andere speichern sich die gewünschte Information in einem Dokument auf der Festplatte ab(vgl. [7, S. 1]). Ein anderer Weg bereits gefundene Informationen wiederzufinden, der die Suche produktiver macht, ist eine Darstellung eines Suchverlaufs, wie in Abbildung 6. Man kann neben kürzlich gestellten Suchabfragen, auch die angesehenen Seiten verfolgen und so jederzeit auf die Informationen wieder zurückgreifen. Das alles wird mit einem simplen Verlaufsfenster neben den Suchergebnissen angezeigt und steigert die Qualität der Suche. Webbrowser besitzen schon lange die Funktion eines Verlaufs. Bei Suchmaschinen ist diese Funktion kaum zu finden, das liegt auch daran, dass nicht alle Menschen eine solche Anzeige gut finden würden, da auch Suchabfragen und Dokumente angezeigt werden, die man nicht wieder zur Anzeige bringen möchte. Ein weiterer Punkt ist die Kombination aus Navigation und Suche. Je mehr Informationen eine Webseite besitzt, desto schwieriger ist es die passenden Informationen zu finden. Orientierung und das gezielte Auffinden und das Vergleichen von Inhalten werden wesentlich schwieriger. Dies hat vor allem Auswirkungen auf die Gebrauchstauglichkeit des Suchinterfaces. Suchergebnisse und Produktkataloge werden in wenig strukturierten Listendarstellungen präsentiert. Für solche Fälle eignet

[3] http://www.ncbi.nlm.nih.gov/pubmed

sich das Einbinden einer Navigationsstruktur sehr gut. Navigation erleichtert dem Nutzer gewünschte Informationen zu finden(vgl. [7. S. 1]). Es sollten nach einer Stichwortsuche, Resultate in einer Navigationsstruktur organisiert sein und nach dem Navigationsschritt, sollte eine erneute Stichwortsuche über das Subset möglich sein. Navigation wird in der Websuche durch Kategoriesysteme umgesetzt. Man unterscheidet in flache, hierarchische und facettierte Kategoriesysteme. Ein flaches Kategoriessystem ist eine Auswahlliste, durch die Inhalte eingegrenzt werden können. Hierarchische Kategoriesysteme sind Systeme, die als Baumstruktur umgesetzt werden.

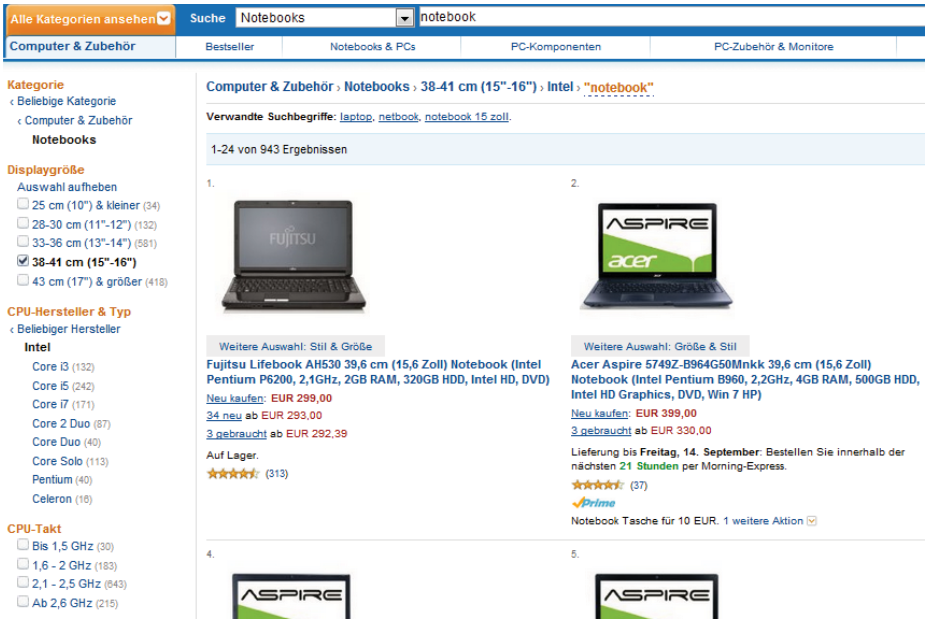


Abbildung. 7. Navigation und Suche der Amazon[4] Webseite

Bei der facettierten Navigation werden Suchergebnisse, „durch die Vorgabe der Ausprägungen verschiedener bedeutsamer Merkmalsdimensionen („Facetten“) quasi vorwegnehmend untergliedert. Dabei werden die bei der Auswahl der einzelnen Facettenausprägungen zu erwartenden Häufigkeiten angezeigt, was die weitere Einschränkung der ursprünglichen Ergebnismenge erleichtern soll" [9, S. 60]. Vielen Nutzern ist die facettierte Navigation vor allem von Amazon (siehe Abbildung 7) her bekannt. Nach Eingabe eines Stichworts ins Sucheingabefeld bzw. nach Auswahl einer Produktkategorie aus dem Submenü, erhält der Nutzer eine Seite, auf der sich im linken Seitenbereich eine Anzahl von Facetten befindet. Mit Hilfe eines Filter-

[4] www.amazon.de

Mechanismus kann er die angezeigte Produktauswahl auf seine Bedürfnisse hin, schnell und einfach selektieren. Vorteile sind die beliebige Reihenfolge der Auswahl und vor allem keine Null-Ergebnisse.

8 Shortcuts anbieten

Abbildung. 8. „Deep links“ bei einer Goolgesuche nach „Apple“

Shortcuts auf der Tastatur um Zeit und Mausklicks zu sparen, kennt wahrscheinlich jeder. Bei dieser Richtlinie ist damit etwas anderes gemeint. Es ist ein schnellerer Weg direkt auf eine andere Navigationsstufe innerhalb einer Webseite zu kommen. Dies wird mit Hilfe der sogenannten „Deep Links“ erreicht. Sie werden meistens in dem Document Surrogate des höchstgelegenen Suchergebnisses dargestellt. Die „Deep Links“ verweisen meistens direkt auf bestimmte Produkte, die häufig gesucht und geklickt werden. In Abbildung 8, sieht man eine Googlesuche mit dem Suchbegriff „Apple“. Im Document Surrogate sind sechs „Deep Links“ enthalten. Alle verweisen direkt auf eine Navigationsstufe zum direkten Kauf eines Produkts, wie das „iPhone“ oder „iPad“ oder auf den „Apple Store“ und den „Apple-Support“. Außerdem enthalten die „Deep Links“ jeweils eine kleine Beschreibung, wohin der Weg führt bzw. welches Produkt angesteuert wird.
Durch das Einbauen solcher Interface-Gadgets, wird dem Nutzer Zeit gespart, sich durch Webseiten zu navigieren und erhöht die Qualität der Suche.

9 Design

Das Design selbst ist die letzte und eine wichtige Richtlinie, um ein erfolgreiches Interface Design zu kreieren. Die Wichtigkeit von kleinen Details und die Ästhetik im Design spielen dabei eine große Rolle. Designer von Suchinterfaces müssen überlegen, wie sie die vielen, komplexen Informationen auf einer Seite unterbekommen. Dabei können kleine Details ausschlaggebend sein, wie die Nutzer ein solches Interface annehmen. Ein Beispiel dafür ist die breite der Suchleiste. Umso breiter diese ist, desto mehr ermutigt sie den Nutzer längere Abfragen zu tätigen. Die erfolgreichste Suchmaschine: „Google“, hat auch eine der breitesten Suchfelder (vgl. Abbildung 9).

Abbildung. 9. Vergleich der Suchfeldbreite zwischen www.google.de und www.dogpile.com

Ein anderes Beispiel lässt sich ebenfalls bei Google finden. Die Vorschläge der Rechtschreibkorrektur wurden in alten Versionen im oberen Bereich der Seite angezeigt. Doch zu fixiert auf die Suchergebnisse, registrierten viele Nutzer die Vorschläge nicht. So sahen sie sich unbrauchbare Ergebnisse an, da sie Fehler in der Eingabe machten. Dieses Problem wurde durch zwei kleine Details behoben. Zum einen wurde die getätigte Abfrage am unteren Ende der Seite noch einmal angezeigt, zum anderen wurde vor die Korrekturvorschläge der Zusatz „DID YOU MEAN“

davorgesetzt. Man sieht also, dass kleine Details, große Wirkung auf Nutzer haben können, was sich auf die Nutzerbindung an eine Suchmaschine positiv auswirkt.

Spricht man von der Ästhetik, meint man eine Wahrnehmung oder ein Empfinden von etwas Schönem. Der Mensch hat angeborene Wahrnehmungsprinzipien, aus denen Gestaltgesetzte abgeleitet werden können. Gestaltprinzipien sind das Gesetz der Nähe, der Ähnlichkeit, der Geschlossenheit, der guten Fortsetzung, des gemeinsamen Bereichs und des Zusammenhangs(vgl. [2, S. 229]). Bei den Interfaces der Suchmaschinen sorgen diese Gesetze „für eine visuelle Struktur, für Gruppierung und Abgrenzung“[2, S.229]. Besonderes Augenmerk bei der Gestaltung sollte man auf die Suchergebnisse legen, denn diese sind das „Endprodukt“, also die Informationen, nach denen die Nutzer streben. Wichtige Merkmale sind Intensität, Farbe, Position, Ausnahme, Dissonanz, Eye Catcher und Gewöhnung(vgl. [2, S.229-230]).

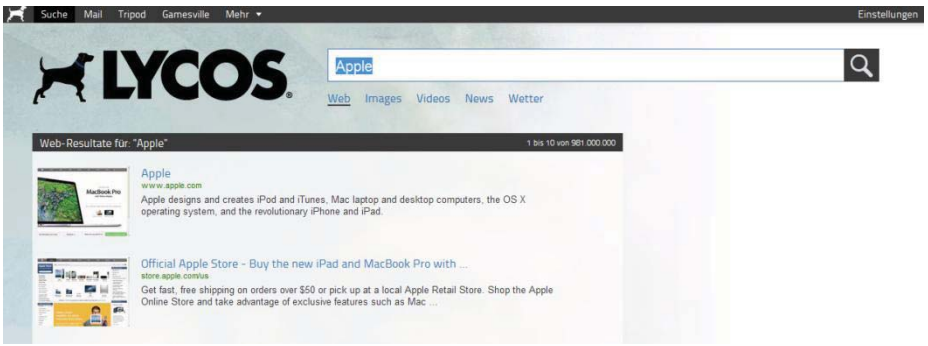


Abbildung. 10. Lycos[5] Suchinterface

Viele angesprochene Prinzipien und Merkmale fallen bei der „Lycos-Suchmaschine“ auf (vgl. Abbildung 10). Die Suchergebnisse enthalten neben dem Document Surrogate, auch einen Screenshot der Seite. Diese Screenshots geben dem Nutzer zum einen, einen kleinen Eindruck wohin die Seite führt oder hilft ihm dabei vergessen geratene Seiten, von denen man nur noch Farbe oder Details weiß, schneller wiederzufinden. Diese kleinen Bilder sind die bereits erwähnten Eye Catcher dieser Suchergebnisse, die kombiniert mit vielen Gestaltprinzipien, wie das Gesetz der Nähe (Screenshot und Document Surrogate nah beieinander) ein sehr ansprechendes Interface ergeben. Eine solche ästhetische visuelle Gestaltung zeugt von Kompetenz des Providers und bindet den Nutzer emotional an eine Suchmaschine. Das hat auch eine Studie gezeigt, bei der die Teilnehmer länger auf einer Seite geblieben sind, wenn sie das Design ansprechend empfanden. Außerdem wurde den Teilnehmern eine Aufgabe gestellt, die wesentlich schneller durchgeführt wurde, als bei einem Interface, das die Teilnehmer nicht so schön fanden.

[5] www.lycos.de

10 Schlusswort

Im Internet findet man heute unzählige Anbieter von Suchmaschinen. Alle versuchen ein ansprechendes Interface Design für die Nutzer anzubieten, um möglichst viele an ihre Suchmaschine zu binden. Doch nicht allen gelingt es, ein erfolgreiches Interface auf den Markt zu bringen und sich von der Konkurrenz abzuheben. Das Interface Design und seine Auswirkungen auf die Suche werden oft unterschätzt. Es hat großen Einfluss darauf, wie man seine Suche gestaltet, wie man Information aufnimmt und bewertet. Designer von Suchinterfaces sollten sich also sehr genau an die in dieser Arbeit genannten Richtlinien halten, wenn sie ein erfolgreiches Interface entwickeln möchten und Nutzer langfristig an ihre Suchmaschine binden möchten. Diese Arbeit hat versucht einige Einblicke in die Gestaltung des Suchinterface Designs zu geben. Letztendlich ist jedes Suchmaschinendesign nur so gut, wie sie jeder Nutzer selbst subjektiv empfindet.

Literatur

1 Hearst, Marti: Search User Interfaces. Cambridge University Press, 2009
2 Lewandowski, D.: Handbuch Internet-Suchmaschinen. AKA Verlag Heidelberg, 2009
3 Aula, Anne (2005): Studying User Strategies and Characteristics for Developing Web Search Interfaces. URL: http://acta.uta.fi/pdf/951-44-6488-5.pdf [Stand: 11.09.2012]
4 Hoeber, Orland; Yang Dong, Xue: A Comparative User Study of Web Search Interfaces: HotMap, Concept Highlighter, and Google. URL: http://orland.hoeber.net/download/2006_wi_comparison.pdf [Stand: 7.09.2012]
5 Kible, Ralf (2004): Information Retrieval, Information Extraction. URL: http://www.iis.uni-stuttgart.de/lehre/ws04-05/SemanticWeb/thema_1.pdf [Stand: 8.09.2012]
6 Agichtein, Eugene; Lawrence, Steve; Gravano, Luis: Learning Search Engine Specific Query Transformation for Question Answering. URL: http://web3.cs.columbia.edu/~gravano/Papers/2001/www10.pdf [Stand: 11.09.2012]
7 Jones, William; Dumais, Susan; Bruce, Harry: Once Found, What Then?: A Study of "Keeping" Behaviors in the Personal Use of Web Information. URL: http://research.microsoft.com/en-us/um/people/sdumais/asist-final.pdf [Stand: 12.09.2012]
8 Stock, G., Wolfgang: Information Retrieval. Informationen suchen und finden. München: Oldenbourg Verlag, 2007
9 Oberhauser, Otto (2008): Sachliche Erschließung im österreichischen Verbundkatalog: Status und Perspektiven. URL: http://eprints.rclis.org/bitstream/10760/12231/1/vm_61_3.pdf [Stand 14.11.2012]

Models of the Information Seeking Process

Andreas Hünlein

Otto-Friedrich Universität Bamberg, Lehrstuhl für Medieninformatik
Feldkirchenstraße 21, 96052 Bamberg

andreas-gunther.huenlein@stud.uni-bamberg.de

Abstrakt. Um gute Interfaces für Suchmaschinen zu entwickeln ist es notwendig sich mit dem Informationsbeschaffungsprozess zu beschäftigen. Diese Arbeit behandelt die theoretischen Ansätze, die hinter dem Standardmodell, dem kognitivem Modell, dem dynamischen Modell, der Suche in Phasen, der Suche als strategischer Prozess sowie der sozialen Suche liegen.

1 Einleitung

1.1 Motivation

Jeder Mensch wird mehrmals täglich mit der Aufgabe konfrontiert Informationen zu suchen und auszuwerten. Diese Aufgabe kann eine sehr einfache Faktensuche sein, z.B. „Was ist der höchste Berg Europas?“, „Wie heißt die Hauptstadt von Uruguay?“. Es kann sich aber auch um sehr viel komplexere Fragestellungen handeln, wie z.B. die Informationsbeschaffung für eine wissenschaftliche Arbeit. Seit der flächendeckenden Verbreitung des Internets, entscheiden sich die meisten Menschen für eine Onlinesuche um dieses Problem zu lösen.

Aus diesem Grund findet sich im World Wide Web eine Vielzahl von Suchmaschinen. Alle haben mehr oder weniger dasselbe Ziel, sie wollen dem Nutzer zu einer Suchanfrage, die beste Antwort bzw. Informationen liefern. Um eine erfolgreiche, effektive und effiziente Suche und die entsprechenden Ergebnisse anzubieten, mussten alle Suchmaschinenanbieter dasselbe tun: Verstehen wie sich ein Mensch bei der Informationsbeschaffung verhält, welchen Prozess er dabei durchläuft und welche Strategien er dabei anwendet [1].

Mehrere Autoren haben sich bislang mit diesem komplexen Thema beschäftigt und haben diverse Modellen vorgestellt. In dieser Arbeit sollen die wichtigsten theoretischen Modelle vorgestellt werden: **Das Standardmodell**, **das kognitive Modell**, **das dynamische Modell**, **die Informationssuche in Phasen**, **die**

Informationssuche als strategischer Prozess und **die soziale Suche**. Die Auswahl soll einen (kleinen) Umriss über Modelle geben die aktuell in der Wissenschaft anerkannt werden.

Das Standardmodell und **das kognitive Modell**, welches eine Erweiterung des Standartmodells ist, wurden dabei gewählt, da es das einfachste, wie der Name schon sagt, den Standard darstellt und im täglichen Leben mehrmals angewendet wird. **Das dynamische Modell** gehört ebenfalls zu den Modellen, die im täglichen Leben immer wieder Anwendung finden. Und soll daher auch näher betrachtet werden. **Die Informationssuche in Phasen** und **die Informationssuche als strategischer Prozess** sind besonders interessant, da sie einen Leitfaden für die Informationsbeschaffung bei komplexen Fragestellungen bietet, wie beispielsweise zur Erstellung einer wissenschaftlichen Arbeit. Zu letzte wird noch die Idee einer **sozialen Suche** betrachtet. Diese stellt eine etwas andere Methode dar, kann aber schneller zu einem passenden Ergebnis führen.

2 Die Modelle zur Informationsbeschaffungsprozesse

2.1 Das Standardmodell

Viele verschiedene Autoren, die sich mit dem Thema der Suchprozesse beschäftigt haben, sind dabei zu der Erkenntnis gekommen, dass es sich beim Suchen um einen Kreislauf von Aktionen und Reaktionen handelt[1]. So hat unter anderem Salton die folgenden Aktionen erkannt

1. Identifying an information need (Informationsbedürfnis erkennen)
2. Query specification (Suchanfrage erstellen)
3. Examination of retrieval results (Ergebnisse bewerten)
4. Reformulation of the query (Suchanfrage neuformulieren)

Die Schritte 2 – 4 werden so lange wiederholt bis das Informationsbedürfnis gestillt ist [2].

Viele weitere Autoren haben sich mit diesem Modell auseinander gesetzt und haben dabei ähnliche Schritte bei der Informationssuche definiert. Im ersten Schritt muss der Mensch zunächst seine **Aufgabe erkennen**, das heißt ein Ziel definieren, welches erreicht werden soll. Im nächsten Schritt muss erfasst werden, welche Informationen benötigt werden, um das Ziel zu erreichen, das **Informationsbedürfnis**. Danach muss aus dem Informationsbedürfnis eine **Suchanfrage geschaffen** werden. Die Schwierigkeit ist hierbei, dass eine natürlich sprachige Frage von einer Maschine nur bedingt erkannt werden kann. Die Suchmaschine hat nun die Aufgabe aus einer riesigen Sammlung an gesammelten Daten die passenden **Ergebnisse zu finden**. In den meisten Fällen werden die Ergebnisse in einer Rangfolge angezeigt. Dies soll dem Nutzer den nächsten Schritt erleichtern: **Die Bewertung der Ergebnisse**, auf der Grundlage, des

Informationsbedürfnisses. Wenn der Nutzer mit den Ergebnissen zufrieden ist, dann wird die **Suche beendet**. Andernfalls muss die **Suchanfrage überarbeitet** bzw. **neu formuliert** werden und gegebenenfalls mit einer neuen Suchanfrage der Prozess wiederholt werden.

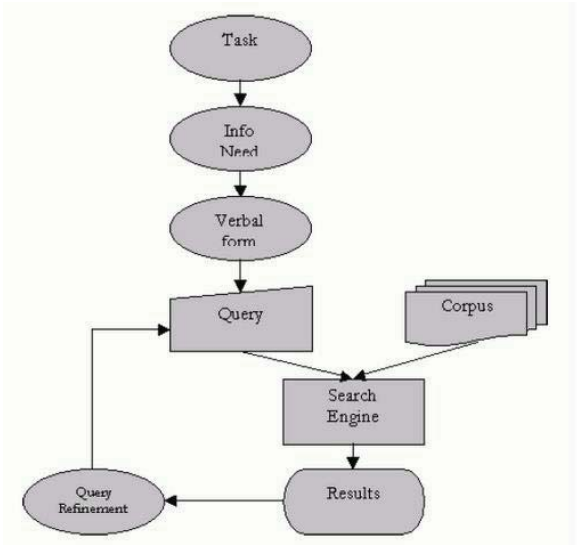


Fig. 1. Das Standardmodell des Suchprozesses , adaptiert von Broder [3]

Bei einfachen Fragstellung (s. Einleitung) ist der gesamte Suchprozess sehr einfach und schnell durchführbar. Die meisten Menschen sind sich nicht einmal bewusst, dass sie das Modell durchlaufen. Es bedarf keiner bzw. nur einer sehr geringen technischen Unterstützung. In den meisten modernen Web Suchmaschinen wird das **Erstellen Suchanfragen**, **das Bewerten der Ergebnisse** und, bis zu einem gewissen Grad, auch **die Neuformulierung der Suchanfrage** unterstützt [1].

1. Erstellen der Suchanfage

Fig. 2. Screenshot der Suggestionsfunktion bei google.de

Bei den meisten Suchmaschinendiensten im Internet ist es üblich, dass, während der Eingabe einer Anfrage, Vorschläge zur Vervollständigung der Anfrage angezeigt

werden. Dies soll einerseits den Eingabevorgang beschleunigen aber auch bei der richtigen Formulierung geholfen werden.

2. Das Bewerten der Ergebnisse

Die Ergebnisse der Suchmaschinen sind meist in absteigender Reihenfolge angeordnet, d.h. das erste Ergebnis wird das Passendste vorgeschlagen. Diese Funktion soll dabei helfen möglichst schnell die richtige Antwort zu finden.

3. Das Neuformulieren der Suchanfrage

Obwohl die Entwickler von Suchmaschinen immer weiter daran ihre Funktionen und Unterstützung zuarbeiten verbessern, werden Nutzer bei der Neuformulierung nur sehr gering unterstützt. Neben der Suggestionsfunktion, die auch hier hilft, wird beispielsweise eine Rechtschreibkorrektur angeboten, damit falsche Ergebnisse durch fehlerhafte Eingaben direkt ausgeschlossen werden.

Fig. 3. Rechtschreibkorrektur von google.de

2.2 Das kognitive Modell

Donald A. Norman, Professor für Informatik und Kognitionswissenschaften, hat in 1988 in seinem Buch „The psychology of everyday things“ [4] ein Modell entwickelt, das beschreibt, wie sich Menschen generell in ihrer Umwelt verhalten. Aus diesem Modell lässt sich ein kognitiver Ansatz für das Standardmodell ableiten [1]. Aus dem Modell von Norman lässt sich eine Ergänzung zum Standardmodell ableiten, das kognitive Modell des Suchprozesses.

Das Modell von Norman läuft in 3 Schritten ab. Zu aller erst muss sich die Person über das **Ziel** klar werden. Er braucht eine allgemeine Idee über das, was erreicht werden soll. Darauf folgen zwei Aktionen. Norman nennt diese „**Execution**“ (Ausführung) und „**Evaluation**“ (Überprüfung). Nachdem eine Aktion ausgeführt wurde, muss die Person beurteilen welche Art der Veränderung sich ereignet hat und, wenn es eine Veränderung gab, ob die das gesteckte Ziel erfüllt. Norman bezeichnet die Lücke zwischen dem was beabsichtigt war und dem was erreicht wurde als „**gulf of execution“.** Die Bestimmung ob die Ziele einer Person erfüllt wurden oder nicht, ist der „**gulf of evaluation**“. Je kleiner diese „gulfs“ sind desto verwendbarer ist die

Benutzeroberfläache einer Suchmaschine. Außerdem legt das nahe, je weniger Wissen ein Nutzer über seine Aufgabe hat, umso weniger sind sie in der Lage ein Ziel zu formulieren und Resultate zu erhalten [1].

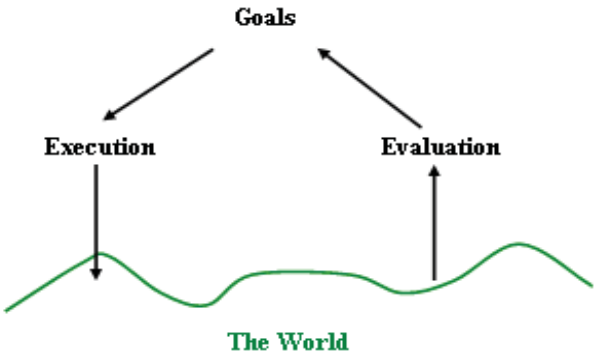


Fig. 4. Eine Zeichung des von Normans Modell, adaptiert von Norman [4].

Wie oben beschrieben kann das kognitive Modell, als Ergänzung zum Standardmodell gesehen werden. Die einzelnen Schritte in Normans Modell können mit den Schritten des Standardmodells assoziiert werden. Das Erkennen der Aufgabe und des Informationsbedürfnisses ist als Zielsetzung interpretierbar. Die Execution umfasst das verbale Formulieren der Frage und das anschließende Erstellen, der Suchanfrage. Das was Norman als (Um-)Welt bezeichnet, ist im Fall der Websuche, die Suchmaschine und die Datenbank die dahinter steht. Die Evaluation ist dann schließlich die Bewertung der Resultate und gegebenenfalls die anschließende Neuformulierung der Suchanfrage [1].

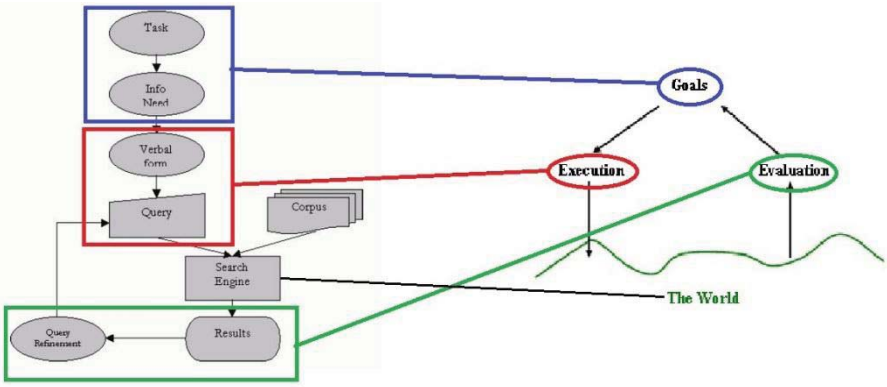


Fig. 5. Zusammenhang zwischen Standardmodell und kognitivem Modell

2.3 Das dynamische Modell

Dem **Standardmodell** des Prozesses der Informationsbeschaffung liegt die Annahme zu Grunde, dass das Informationsbedürfnis des Nutzers **statisch** ist. Der Informationsbeschaffungsprozess ist dann erfolgreich, wenn eine Suchanfrage soweit verfeinert und verbessert wurde bis alle und nur die Dokumente, die für das **ursprüngliche Informationsbedürfnis** relevant sind, gefunden wurden [1].
Studien, die das Suchverhalten beobachten, zeigen, dass sich das Informationsbedürfnis eines Suchenden **verändert** während sie mit dem Suchsystem interagieren. Der Suchende lernt mehr über sein Thema beim Lesen der Ergebnisse und durch Begriffsvorschläge der Suchmaschine. Das führt dazu, dass sich neue **Unterfragen** auftun, die beantwortet werden sollen [1].

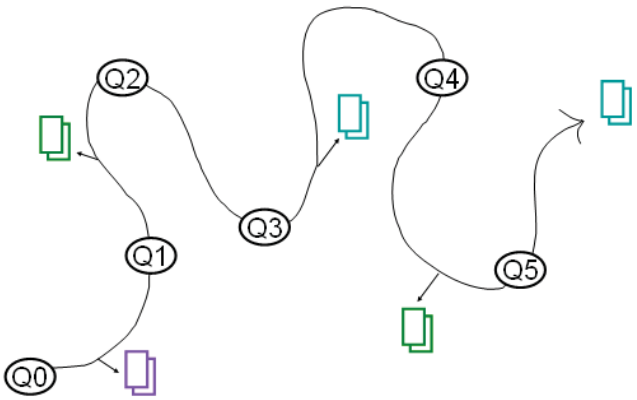


Fig. 6. Eine Zeichung des dynamischen „berry-picking“ models von Bates

Das „berry-picking“ Modell besteht aus zwei Hauptpunkten. Der erste ist, das Lesen und Lernen von den Informationen, die während des Suchprozesses gefunden werden, führen dazu, dass sich das Informationsbedürfnis und daraus folgend auch die Suchanfragen fortlaufend ändern. Informationen, die an einem bestimmten Punkt in der Suche erhalten wurde, können in eine neue, nicht vorhergesehene Richtung führen. Das ursprüngliche Ziel wird so zum Teil erfüllt, was die Priorität eines Ziels zugunsten eines anderen verringert. Der zweite Punkt ist, dass sich das Informationsbedürfnis des Suchenden nicht durch eine einzelne, endgültig gefundene Sammlung von Dokumenten erfüllen. Es handelt sich eher um eine Auswahl von Dokumenten und Informationen, die entlang des Weges gefunden werden. Das steht im Gegensatz zu der Annahme, dass das Ziel des Suchprozesses eine Sammlung von Dokumenten so anzupassen, dass sie zum ursprünglichen Suchziel passt [5].

Zahlreiche Studien unterstützen das „beery-picking“ Modell, wie die O’Day und Jeffries aus dem Jahr 1993. In dieser wurden 15 Geschäftsanalysten über ihre typischen Suchaufgaben befragt. Sie haben herausgefunden, dass der Informationssuchprozess aus einer Serie von verbundenen aber unterschiedlichen Suchen. Sie haben auch herausgefunden, dass die Suchergebnisse nach einem Ziel eine Suche nach neuen Zielen auslösen und führen in eine neue Richtung. Dennoch wird der Kontext eines Problems und die vorherigen Suchen von einer Stufe in die nächste mitgenommen [1].

2.4 Die Informationssuche in Phasen

Einige Forscher haben sich damit auseinander gesetzt, wie sich der Suchprozess über eine längere Zeitperiode entwickelt. Kuhltau leitete 1991 eine Studie, die zeigte, dass, bei komplexen Fragestellungen, der Suchende durch verschiedene Phasen geht, sowohl in dem Wissen über ein Thema, als auch in der Einstellung zu dem Thema.
Um ihr Modell zu entwickeln hat die mehrere Studien durchgeführt. In ihrer abschließenden Studie, hat sie 385 Teilnehmer an öffentlichen, akademischen und Schulbibliotheken an 21 verschiedenen Orten befragt. Die Teilnehmer waren hauptsächlich Schüler oder Studenten, die gerade beim Schreiben einer wissenschaftlichen Arbeit waren. Das Schreiben der Arbeit dauerte meist mehrere Monate und in den meisten Fällen wurde den Studenten das Thema zugewiesen. Kuhltaus Methode war auch deshalb ungewöhnlich, da sie neben den Fragen über den Suchprozess auch Fragen zum emotionalen Status stellte [1].

Kuhltaus Ergebnisse gaben sowohl Aufschluss über den Informationstand der Teilnehmer, als auch über den emotionalen Zustand. Sie unterteilte den Prozess der Informationsbeschaffung in sechs Phasen:

- **Initiation:** Die Aufgabe besteht darin sich über sein Bedürfnis nach Information klar zu werden. Der Suchende bezieht sich auf sein allgemeines Hintergrundwissen. Wenn sich die Person über das fehlende Verständnis bewusst wird, treten Gefühle der Unsicherheit und der Ängstlichkeit auf. Die Gedanken konzentrieren sich darauf das Problem zu verstehen und mit früheren Problemen in Beziehung zu setzten.

- **Selection:** Die Aufgabe ist nun das allgemeine Thema oder den Ansatz, der verfolgt werden soll, festzulegen. Die Gedanken sind sehr generell und undifferenziert, und drehen sich um Anforderungen, die Zeitvorgaben und darum welches Thema oder welcher Ansatz das beste Ergebnis verspricht. Gefühle der Unsicherheit werden durch einen gewissen Optimismus ersetzt nachdem die Auswahl gemacht wurde.

- **Exploration:** Die Aufgabe ist es Informationen über das Thema zu recherchieren um das Verständnis zu erweitern. In dieser Phase ist der Suchende nicht in der Lage auszudrücken welche Informationen er braucht und ist deshalb nicht fähig Suchanfragen zu formulieren und die Relevanz der Ergebnisse zu beurteilen. Informationen, die in dieser Phase gefunden wurden, stehen oft im Konflikt zu vorherigem Wissen und Informationen aus anderen Quellen scheinen widersprüchlich und inkompatibel. Diese Phase ist geprägt durch Verwirrung, Unsicherheit und Zweifel, und die Teilnehmer fühlen sich entmutigt und unpassend, oder sind mit dem Suchsystem an sich unzufrieden.

- **Formulation:** Diese Phase markiert den Wendepunkt im gesamten Informationsbeschaffungsprozess, an dem eine konzertiertere auf das Thema entsteht, in dem einige der vorher entstandenen Konflikte gelöst werden. Suchen werden so geführt, dass die Hypothesen überprüft werden. Eine Veränderung der Gefühle tritt auf, Unsicherheit wird reduziert und Das Selbstvertrauen gestärkt. Leider gab es bei der Hälfte der Teilnehmer an der Studie keine Anzeichen dafür, dass während des gesamten Prozesses eine konzentrierte Perspektive auf ihr Thema erlangten.

- **Collection:** In dieser Phase ist das Suchsystem am nützlichsten für den Suchenden, da die Aufgabe darin besteht Informationen über das engere Thema zu sammeln. Die Suche wird dazu benutzt um Informationen zu finden, die den Fokus definieren, ausweiten oder unterstützen sollen. Relevanz Beurteilungen werden genauer und die Gefühle der Sicherheit werden stärker.

- **Presentation:** In dieser Phase ist die eigentliche Suche fertig; Suchen sollten Informationen zurück geben, die sich entweder mit dem decken, was schon gefunden wurde, oder weniger relevant sind für das Thema. Die Teilnehmer haben entdecken für gewöhnlich Gefühle Erleichterung und Zufriedenheit, wenn die Suche erfolgreich war, oder Enttäuschung wenn nicht.

[6]

Ähnliche Ergebnisse wurden auch in einer anderen Studie von Vakkari, aus dem Jahr 2000 gefunden, in der 11 Studenten beobachtet wurden, die ihre Masterarbeit geschrieben haben, über einen Zeitraum von 4 Monaten [1].

“In general, all the participants proceeded in their task according to Kuhlthau, 1991's model at varying paces. In the first search session, the students were moving from topic selection to exploration of the topic. In the middle of their task they were typically exploring the topic and trying to formulate a research problem. By the end of the project most of the students had been able to construct a focus and they were at the collection or presentation stage.” [7]

Diese Phasen beschreiben eine Veränderung beim Suchen über eine gewisse Zeit bei tiefgründigen und komplexen Informationsbedarf, sind aber nicht unbedingt repräsentativ bei oberflächlichen Themen. Außerdem spiegeln diese Studien die Erfahrungen von Studenten die sich mit herausfordernden Aufgaben befassten. Es ist wahrscheinlich, dass sich die Gefühle der Ängstlichkeit bei anderen Aufgaben und Umwelten nicht beobachten lassen. Zusätzlich waren die Werkzeuge, die in der Studien von Kuhltau verwendet wurden, unter Umständen weniger vertraut und brauchbar, als Werkzeuge die es heutzutage gibt [1].

2.5 Die Informationssuche als strategischer Prozess

Einige der Informationssuchmodelle verstehen den Prozess als Strategie und wie Entscheidungen über den Ablauf der Suche getroffen werden. Bei Marchionini et al. wird es so ausgedrückt:
„search is an interplay of analytical and interactive problem solving strategies." [8]
In manchen Fällen versteht man unter einem strategischen Modell das bewusste planerische Verhalten von professionellen Suchern. In anderen ist damit weniger das Planen als das mehr interaktive Verhalten eines typischen Informationssuchers gemeint [1].

2.5.1 Strategie als eine Abfolge von Taktiken.

Das Verhalten eines Suchenden kann durch Suchstrategien beschrieben werden, diese bestehen wiederum aus einer Abfolge von Taktiken. Eine Taktik ist eine unmittelbare Entscheidung oder Aktion, die im Anbetracht des aktuellen Stands der Suche, gemacht werden. Strategien beziehen sich dabei auf eine Kombination von Taktiken, welche genutzt werden um Informationszugangsaufgaben zu bewältigen. Diese Strategien sind eine Folge von Taktiken, die zusammen betrachtet, die dabei helfen einige Aspekte oder Unterziele des Hauptzieles des Nutzers zu erreichen. Bates nennt mehrere Taktiken, die sie in vier Kategorien zusammenfasst [9].

- **Begriffs Taktiken:** Bezieht sich darauf die aktuelle Suchanfrage mit Begriffen und Sätzen anzupassen. Das beinhaltet den gebraucht von Begriffen die vom Suchsystem vorgeschlagen werden und die Auswahl aus einem Online Thesaurus.

- **Informationsstruktur Taktiken:** Techniken um sich durch die Informations- oder Linkstrukturen zu bewegen, um Quellen oder Informationen in den Quellen zu finden. Ein Beispiel einer Informationsstruktur Taktik für eine akademisches Arbeit ist es die wissenschaftlichen Arbeiten zu betrachten, die eine bereitgestellte Arbeit zitieren, und der Zitationskette zu folgen. Ein anderes Beispiel ist es, wenn man in einer Onlinesammlung oder Webseite sucht, und dabei den vielversprechendsten Hyperlinks folgt.

- **Anfrage Neuformulierungs Taktiken:** Beispielsweise das Annähern einer gegebenen Suchanfrage indem man genauere Begriffe benutzt oder indem man eine Suchanfrage besser strukturiert über Boolean-Operatoren.

- **Überwachungs Taktiken:** Überwachen bezieht sich darauf den Überblick zu behalten während sich eine Situation entwickelt. Darunter fällt, nach Bates, z.B. eine Kosten-Nutzen-Analyse, das ständige vergleichen des aktuellen Standes zum ursprünglichen Ziel oder das Aufzeichnen von unbeendeten Suchpfäden um später dort weiter zu suchen.

[9]

Dabei stellt sich die Frage was einen suchenden dazu bewegt eine Strategie, die er verfolgt zu beenden und eine andere zu verfolgen. O'Day und Jeffries definieren eine Menge von Auslösern, die einen Sucher dazu motivieren seine Strategie zu ändern [1].

- Das Fertigstellen eines Schrittes und Anfangen des nächsten logischen Schrittes in einem Plan.
- Etwas interessantes wird aufgedeckt, dass eine neue Art des Denkens bietet oder einen neuen Blickwinkel eröffnet.
- Aufdecken einer Änderung oder eines Bruchs mit dem bisher gefundenem, was weitere Nachforschungen nötig macht

[10]

Einige Suchaufgaben sind einfach genug, dass per se keine Strategie von Nöten ist. Einfache Faktensuchen im Web sind ein Beispiel dafür. Der Suchende öffnet den Web-Browser, navigiert zum Eingabeformular der Suchmaschine, gibt die Informationen ein die er braucht und durchsucht die erhaltenen Ergebnisse um die Antwort zu finden oder einen Link auf eine Web-Seite, die die Antwort enthält [1].

2.5.2 Kosten-Nutzen-Analyse

Wie zuvor erwähnt, gehört die Kosten-Nutzen-Analyse zu den Überwachungstaktiken. Die Kosten-Nutzen-Analyse geht davon aus, dass zu jedem Zeitpunkt im Suchprozess, die Strategie verfolgt wird, die den größten Nutzen hat. Wenn eine Strategie, als Konsequenz aus taktischen Entscheidungen eine höheren Nutzen aufweist, als die, die gerade verfolgt wird, dann wird die alte Strategie verlassen und die Neue verfolgt.

2.6 Die soziale Suche

Das letzte Prozessmodell, das vorgestellt werden soll unterscheidet sich grundsätzlich von allen bisher gezeigten. Üblicherweise war das Vorbild für die Informationsbeschaffung eine Bibliothek. In der Tat hat die Informationsbeschaffung seine Wurzeln in den Bibliotheken Wissenschaften, Google selbst entstammt dem „Stanford Digital Library“ Projekt. Obwohl dieses Vorbild in zahlreichen Zusammenhängen offenkundig sehr gut funktioniert hat, ignoriert es ein anderes Jahrzehnte altes Modell der Wissensbeschaffung. In einem Dorf wird die Wissensverbreitung sozial erreicht. Informationen werden von Person zu Person weitergegeben. Die Aufgabe der Informationsbeschaffung ist es die richtige Person zu finden, und nicht mehr das richtige Dokument um die Frage zu beantworten.
Die Unterschiede zwischen der Art und Weise wie Leute Informationen in einer Bibliothek finden und wie sie diese in einem Dorf finden, legen einige nützliche Regeln nahe, um eine soziale Suchmaschine zu entwickeln. In einer Bibliothek verwenden die Menschen Schlagworte zum Suchen, die Wissensgrundlage wird von wenigen Herausgebern erschaffen bevor die Fragen gestellt werden, und die Grundlage für Vertrauen ist die Autorität. Im Gegensatz dazu, wird in einem Dorf natürliche Sprache verwendet um Fragen zu stellen, Antworten werden in Echtzeit von irgendjemandem aus der Gemeinschaft erstellt, und Vertrauen basiert auf Vertrautheit. Diese Eigenschaften haben verschachtelte Effekte – zum Beispiel, haben Echtzeit Antworten von echten Personen funktionieren sehr gut für Fragen die einen hohen Kontext benötigen und für sehr subjektive Anfragen. Beispielsweise wird die Frage „Habt ihr irgendwelche Empfehlungen für einen guten Babysitter in Palo Alto für meine 6-jährigen Zwillinge? Ich suche nach jemandem der sie kein Fernsehen schauen lässt.“ wird von einem Freund besser beantwortet als in einer Bibliothek. Diese Unterschiede im Vorbild der Informationsbeschaffung fordern, dass soziale Suchmaschinen ganz andere Architekturen, Algorithmen und Nutzeroberflächen benötigen, als eine Suchmaschine, die die Bibliothek als Vorbild nutzen. [11]

Abschließendes Fazit

In der Arbeit wurden mehrere Prozessmodelle betrachtet. Die Auswahl soll zum einen ein grundlegendes Verständnis der Informationsbeschaffung liefern. Andererseits sollen die Weiterführenden Modelle ein tieferes Verständnis schaffen, und verschiedene Seiten der Informationsbeschaffung beleuchten. Auch wenn der Sucher, die einzelnen Schritte der Modelle, z.B. die des Standardmodells, oft instinktiv und von sich selbst unbemerkt durchläuft, vor allem bei einfachen, unkomplizierten Fragestellungen, ist es dennoch für die Entwickler von Suchmaschinen von größter Bedeutung diese Phasen zu kennen, und zu verstehen.
Am Beispiel des Standardmodells wurde verdeutlicht, dass es zum Teil schon Werkzeuge und Funktionen, der Suchmaschinen gibt, die den Nutzer in den einzelnen Phasen unterstützten sollen. Es fehlt aber an anderen Stellen noch sehr an den richtigen, hilfreichen und verwendbaren Funktionen. So tritt bei vielen Nutzern immer wieder das Problem der Bewertung, der gefundenen Ergebnisse auf. Suchmaschinen versuchen zwar, durch immer bessere Algorithmen, dem Nutzer eine Hilfestellung zu geben und die Ergebnisse beispielsweise nach absteigender Relevanz zu ordnen, oder durch Anzeigen des „Page Rank“ eine gewisse Relevanz zu verdeutlichen. Diese Funktionen helfen aber auch nur bedingt und können, nicht als allgemein gültige und beste Lösung gelten, da verschiedenen Nutzer, mit unterschiedlichem Vorwissen, bei der gleichen oder ähnlichen Suchanfragen, den Ergebnissen eine andere Relevanz zuordnet. Während bei einfachen Faktenfragen wohl jeder Nutzer die gleiche Seite, zumindest i, gleichen Sprachraum, angeboten werden kann. Dazu kommen noch ggf. kommerzielle Interessen der Entwickler, die von einer Suchmaschine umgesetzt werden müssen. Auch schon bestehende Tools, die man als sehr ausgereift betrachten kann, müssen sich immer wieder neuen Umweltbedingungen stellen. Man betrachte einmal den Fall um Bettina Wulff, die Google verklagt, da ihr Name mit Hilfe dieser Suggestionfunktion direkt mit unangebrachten Wörtern in Verbindung gebracht wird. Dies kann eine neue Rechtslage für Google schaffen, die vorübergehend das Abschalten dieser Funktion zur Folge hat.
Dies alles sind Probleme mit denen sich die Suchmaschinen Entwickler auseinander setzten müssen um den Informationsbeschaffungsprozess von Nutzern möglichst optimal zu unterstützten und somit den Nutzer an sich zu binden.

Literaturverzeichnis

1. Hearst, M.A.: Search User Interfaces. CH. 3: Models of the information seeking process. Cambridge University Press (2009)

2. Salton, G.: Automatic text processing: the transformation, analysis, and retrieval of information by computer. Addison-Wesley, Reading, MA (1989)

3. Broder, A.:. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10 (2002)

4. Norman, D.A.: The psychology of everyday things. Basic Books (1988)

5. Bates, M.J.: The design of browsing and berrypicking techniques for the on-line search interface. *Online Review*, 13(5):407–431. (1989)

6. Kuhlthau, C.C.: Inside the search process: Information seeking from the user's perspective. *Journal of the American Society for Information Science*, 42(5):361–371. (1991)

7. Vakkari, P.: Relevance And Contributing Information Types Of Searched Documents In Task Performance. Proceedings of the 23th Annual International ACM SIGIR Conference on Research and development in information retrieval (SIGIR'00), pp. 2–9 (2000)

8. Marchionini, G., Geisler, G. , Brunk, B.: Agileviews: A Human-Centered Framework For Interfaces To Information Spaces. *Proceedings of the American Society for Information Science Annual Meeting*, 37:271–80 (2000)

9. Bates, MJ.: Information search tactics. *Journal of the American Society for Information Science*, 30(4): pp. 205–214. (1979)

10. O'Day, V.L. and Jeffries, R.: Orienteering in an information landscape: how information seekers get from here to there. In *Proceedings of the INTERCHI Conference on Human Factors in Computing Systems (CHI'93)*, IOS Press. (1993)

11. Horowitz, D., Kamvar, S.D.: Searching the Village: Models and Methods for Social Search. Communications of the ACM, 55(4), pp. 111-118. (2012)

Search User Interfaces: Presentation of Search Results

Alexander Schreiner,

Am Wiesengrund 4, 91207 Lauf an der Pegnitz, Germany
alexander-maximilian.schreiner@stud.uni-bamberg.de

Abstract. This Paper is about how search engine results can be presented and visualized. It describes what different features the search result page can have and what different ways there are to display search results. It explains what summaries are and what they should look like. It explains highlighting features on the result page and different studies that have been done to find out what the best and most popular way is to present different parts of the result page as well as what the results to these studies were. It discusses the importance of search result ordering and shows examples of different approaches to search result page styles.

Keywords: Search Results, Search Hits, Presentation, Visualization, Snippets, Abstracts, Summaries, Document Surrogates, Search Engine Result Page

1 Document Surrogates

When a user enters a query into the search mask of any web search engine he will get to a page where several results, which are also called hits, are shown. The page that contains the results is called the “search engine results page”, or abbreviated SERP. The most common way to display the results of a query is a vertical list. The items in the list are called “document surrogates”. Document surrogates typically consist of the document’s title and some important metadata. This metadata contains a short summary of the important parts of the document, as well as the URL, but may also contain the author, the date and the length of the article or other information (see Figure 1).

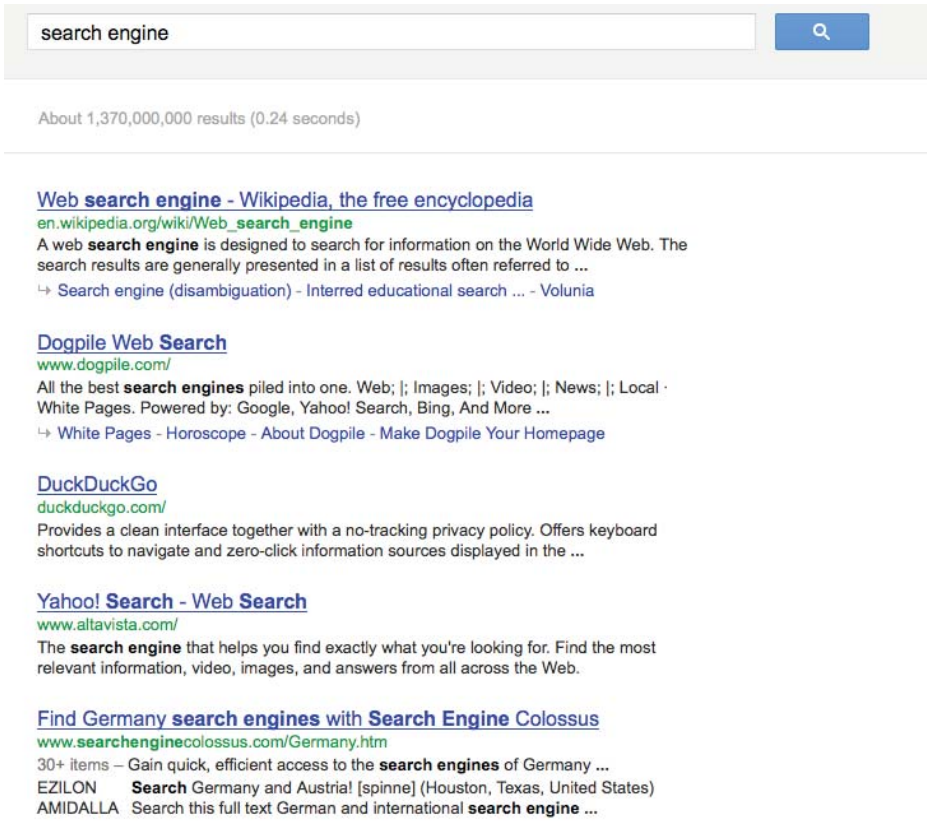


Fig. 1. The results of the query *search engine*, each has a title, URL and summary, some provide further metadata

The document surrogates are very important to help the user understand what information the documents are intended to give. Therefore they show which document suits the user's information needs best. Also the user can take information from the shown metadata. However, the metadata is mainly meant for computer programs or web search engines.
The document surrogates try to show the relevance of the document. A good result document should have a surrogate that shows that the information searched for matches the information given in the document. The most important parts of the surrogate are the title, the summary and the URL. Clarke et al. (2007) [1] tested which aspects are important for a good document surrogate. A perfect surrogate should meet the following requirements:

- It should show a title, summary and URL
- The summary should not be too short
- The summary should contain as many of the query terms as possible
- The summary should start with a phrase from the query

- Title, summary and URL together should contain the query as a phrase match
- The summary should contain exactly one match for every query term
- The URL should be of the form www.query.com
- The URL should be short in terms of slashes
- The URL should be short in terms of characters
- The summary should pass a simple readability test

2 KWIC, or query-oriented Summaries

As mentioned above, one of the most important parts of the document surrogate is the summary, which is also called snippet or abstract. A good summary has the query in it. This is called a keyword-in-context (KWIC) extraction. KWIC snippets are also referred to as query-biased, query-dependent, query-oriented, or user directed snippets. In KWIC summaries sentences or sentence fragments that contain the searched keywords are shown, together with other information that might be relevant to the user.
A regular abstract is different from a KWIC abstract. Regular snippets summarize the content of a document, whereas KWIC snippets show the way the query terms are used in the document and the context in which they relate in the document.
There are several theories on what a summary should contain. According to often-used heuristics a good brief summary should simply show the first few sentences of a document. A good query-biased summary should show sentences that contain the query terms. Researches show that query-biased summaries are more helpful to the user, because the chances of finding the best documents are higher and being led to irrelevant documents can be avoided better. As a result query-biased summaries have become the recent standard for search engines. Brief summaries, that show only the first few sentences have been completely replaced.
In many cases the answer can be found by looking at the result page alone, if query-oriented summaries are used. For example the result page for the query *how do I take a screenshot In Mac OS X* already offers some suggestions on how to solve the problem (see Figure 2).

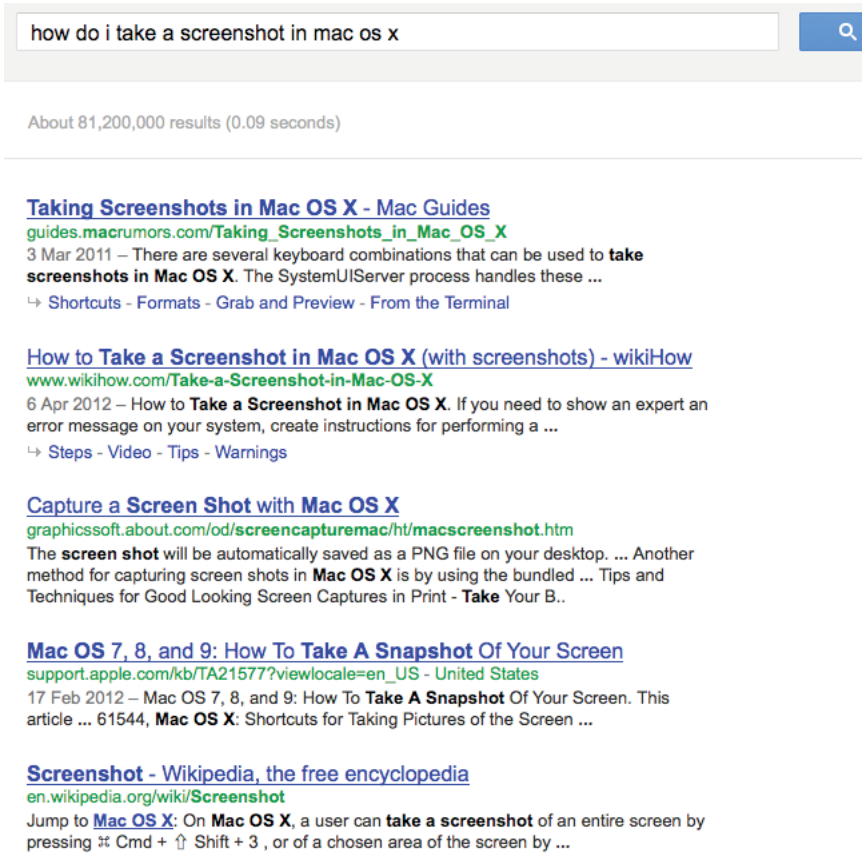


Fig. 2. The search engine result page for the query *how do I take a screenshot In Mac OS X* already shows the solution to the problem

The following three subchapters describe how a good query-oriented snippet should be structured and what is important to search engines when deciding which sentences or sentence fragments should be displayed in the surrogate.

2.1 Sentence Selection for query-oriented Summaries

When setting up a query-oriented Summary the first question is what sentences to use. There are several attributes to sentences that could be important to find out which sentences are most relevant. Some of these are position in the document, words contained and number of query terms contained.
Many researchers have experimented with these and other aspects of sentences that could give a good ranking on which sentences are most important. Number of query

terms written in certain text formatting is one aspect that researchers experimented with. J. Goldstein [2] found out that indefinite articles (e.g. a) are a sign of relevance rather than definite articles (e.g. the). Also proper nouns and other named entities are better than common nouns.
In 2006 a group of scientists developed a method that took relations between sentences into account by combining the most relevant fragments of the document. This method turned out to be more effective than the commercial methods.

2.2 Summary length for query-oriented Summaries

When displaying the search engine result page the search engine has to decide how long the summary should be. The advantage of longer summaries is that they usually show more relevant information, whereas short summaries result in shorter document surrogates, which means there is more space for other search hits.
Since it is good to show as many of the query terms in the snippet as possible, there might be a problem when they are further apart, because we don't know which parts of the text in between should be shown. One option to solve these problems would be to show short summaries that expand when they are clicked.
Studies showed that longer snippets significantly improve performance for information tasks, but when someone is simply searching for a specific URL bigger amounts of text are only distracting.
All in all it can be said that there is no general answer to this problem because different types of queries need different kinds of information, and also different amounts of information. When someone is searching for a certain person, place, product, etc. and enters a query like *Who was the 20th president of the United States?*, he is looking for a short and simple answer. However, someone who enters a query like *How do I make pancakes?* is looking for a description and wants to get a longer and more detailed answer.

2.3 Sentence Fragments vs. Full Sentences for query-biased Summaries

The next important question is whether it is more productive to use full sentences or just sentence fragments in the snippets. Several researchers have done many experiments to find an answer. They found out different things. The use of bulleted lists as a summary can answer user questions more efficiently but standard snippets containing of sentence fragments are more useful to show if a document is relevant or not. Very choppy sentences have negative effects. High-ranking sentences alone can be better than snippets. Many capital letters, stopwords, or long words, as well as a large percentage of punctuation have negative influence on the readability of the summaries.
As a result to these researches it can be said that sentence fragments should be used when necessary, but high-ranking sentences should be displayed as complete sentences.

3 Highlighting Query Terms

Highlighting of query terms in the search hits is a useful feature to improve efficiency. There are different ways to highlight a word in the document surrogate. The most common way is boldface. Other options would be colored backgrounds or reverse video. Highlighting helps the user find relevant documents faster, because these visual effects draw his attention to the important parts of the text.
Highlighting is not only used in search results, but also in full documents to find the important passage of the text. This is very useful for long texts. For example the search tool in web browsers highlights searched words on your current web page, by giving them colored backgrounds. A different color is used for every query term. Showing the positions of highlighted words in a text can also be done in other ways. One possibility is to show marks on the scrollbar. This can easily lead users to the right location in the text (see Figure 3).

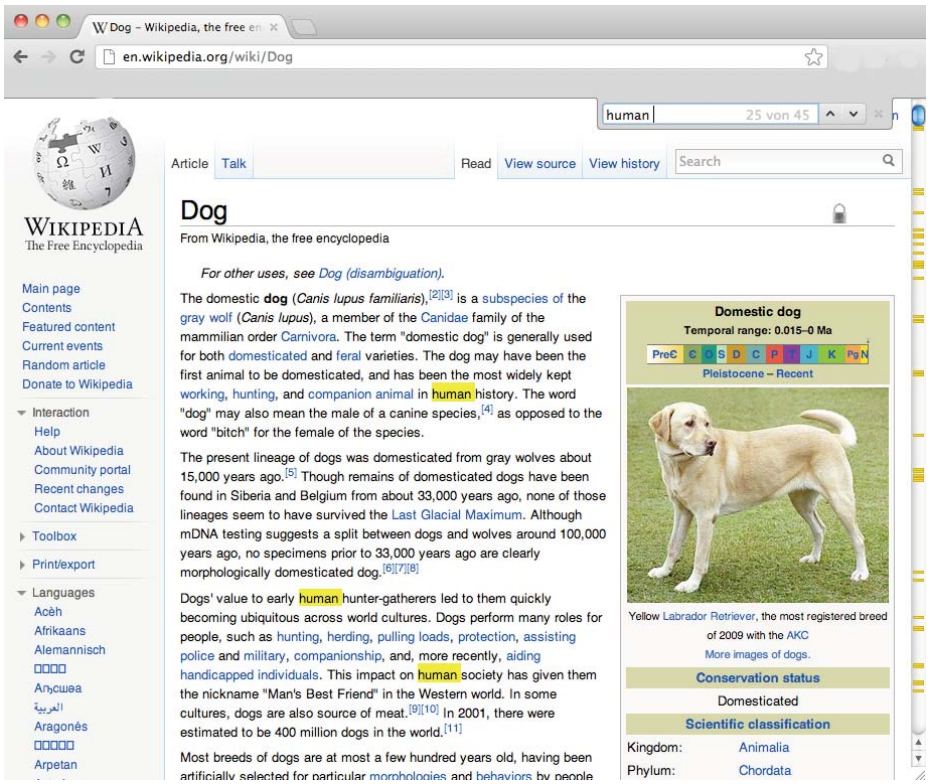


Fig. 3. Browser search result for *human* in Google Chrome, which uses highlighting in the text and in the scrollbar

Another way is to show an overview of the text using a miniature version of it with highlighted query terms. The Browser *fishnet* uses this, as well as providing highlighting on the main page (see Figure 4).

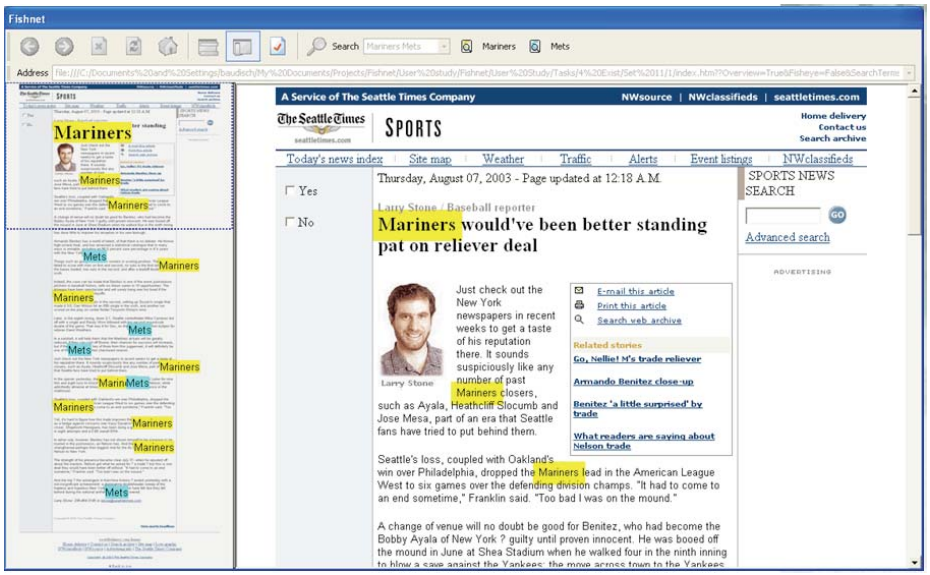


Fig. 4. The result of the browser search for *Mariners Mets* in the Fishnet-browser

Several tests have been done to show which highlighting method is the most efficient and most popular. The result was that most participants preferred the highlighted overview using different colors (as seen in Figure 3).

4 Additional Features of results listings

Next to the usual information that search results contain, search engines sometimes show further data. In this chapter several examples of additional features that the search engine result page can have are explained.
The *Number of search hits per page* can be varied, but most search engines show 10 hits on every page. Experiments showed that an increase to 30 hits per page reduced traffic by 20% because the result page took 0.5 seconds longer to be generated.
Graphical Displays of Relevance Score have been used for many years. Next to every search result title used to be a score or an icon that tried to indicate the relevance of the corresponding document surrogate. But this has been found useless and unpopular for several reasons. First, knowledge was needed to interpret the scores, which most users did not have. Second, often the top scores were very close to each other so these numbers or symbols did not give much further information. Also, studies found out that users did not want these scores, nor did they improve efficiency.

Previews of the Document Content is shown in many search interfaces. By clicking the title or a representing symbol the user can preview the content of the search hit without leaving the result page (see Figure 5).

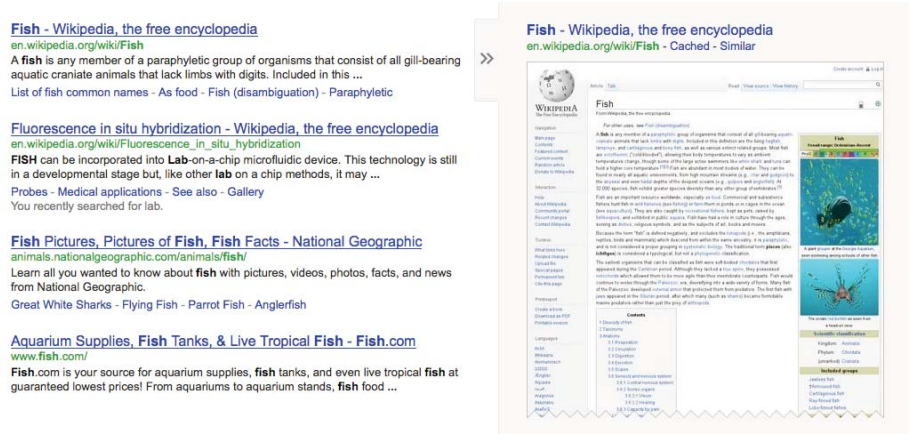


Fig. 5. On the right hand side Google shows a preview for the selected search hit, an article about fish on Wikipedia

Indicators of Search Result Diversity is especially important for ambiguous search terms. Search Engines try to show a diversity of search hits within the first few results (see Figure 6).
Some Search engines show *Indicators of Additional/Related Hits*. They group related hits from one website and show them together (also see Figure 6, blue arrow).

Fig. 6. Search Result Diversity: different types of hits for the different meaning of the ambiguous search term *labs* are shown
Additional/Related Links (blue arrow): related hits from labs.adobe.com are grouped

Sitelinks are also called *deep links* and are sometimes shown right beneath the top scoring website. They contain links to more pages from the same website which are buried one ore more levels within the site. These links are chosen based on popularity and title. They often spare the user the need to load the homepage and navigate through the page (see Figure 7).

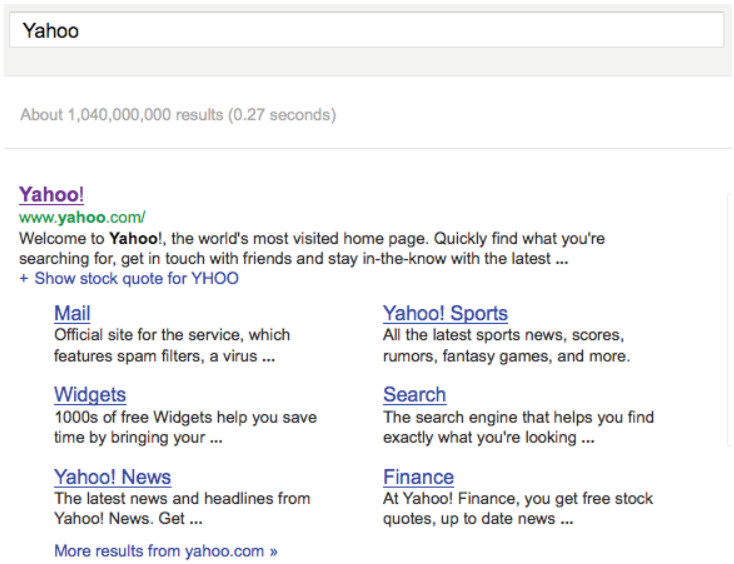


Fig. 7. The query *Yahoo* provides sitelinks that lead directly to several sub sites of yahoo.com

Shortcuts answer directed or focused information needs directly on the result page (see Figure 8).

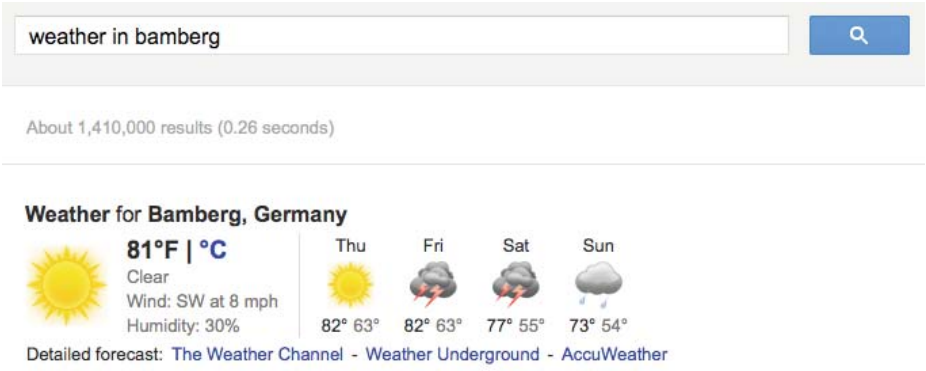


Fig. 8. The query *weather in bamberg* offers an answer to the problem directly on top of the search engine result page

Some queries result in *Blended Results and Media Types*. Different types of results shown directly on the result page may contain sports scores, news, weather, web search results or others (see Figure 9).

Fig. 9. The query *uefa.com euro 2012* shows the scores of the latest matches on the search engine result page

One last feature that is often shown on the result page is *Advertisement*. Many search engines show different kinds of advertisement on the right side or at the top of the search engine result page.

5 The Effects of Search Results Orderings

Usually the results of a query are ordered according to a calculated relevance score. However, in some cases it is more useful to use an order specified by date of release for news search, number of appearances in citing papers for journal article search or other metadata such as author name for articles or sender name for emails.
There have been many different studies and researches to find out how important Search Results Ordering really is. These studies have found out several things.
One thing that was found out is that users expect the best answer to be among the first or second hit. The top result was clicked 85% of the time, the second one only 10% of the time. These numbers held even when the positions of the first and second result were swapped. Most of the time users only look at the first page. If they can't find what they are looking for on the first page they give up or reformulate their query.
An eye-tracking study found out that users mainly look at the top hits, only 5% of the time or less they looked at the results 7 to 10. This result did not change even when the search hits were shown in reverse order.
Another study found out that users rather view the snippets above the result they click on than below it. They look at the abstract below only 50% of the time.
Joachims et al. (2005) [3] did a research that completely reversed the order of the search hits. The average rank of a click went from 2.66 in the normal order to 4.03 in the reversed order. So people did notice that something was different and scanned the

lower results more carefully but they didn't go all the way down to find the most relevant hits. The average relevance was lower than in the normal condition.
Another test that was done used eye-tracking technology to find out what results users look at the most. When the order of the results was changed, so that the most relevant result was not at the top, it took users longer to make a decision but they were not more successful in finding the target hit. They still got manipulated by the order and selected the top hits most of the time. When users didn't find the wanted result within the top three hits, they either selected the first hit (about 50% of the time) or reformulated their query. Users always expected the best hits to be at the top.
Another eye-tracking study found out that experiences searchers use a much more efficient style of searching and find results much faster than inexperienced searchers.

6 Visualization of Search Results

The vertical list of search results as described above is the standard and most popular way to display the search hits. However there have also been other ideas on how the search results could be visualized.
One idea was to show search hits with thumbnail images instead of textual surrogates. But this idea never showed any advantage, so it is not used in anymore. Another idea that was more successful was a "cover flow", which was introduced by a search engine called SearchMe (see Figure 10).

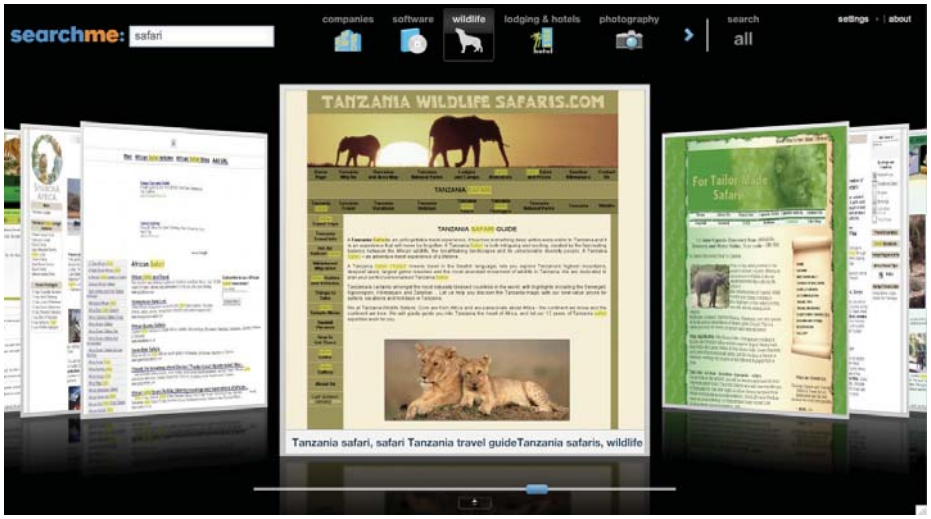


Fig. 10. The results of the query *safari* in the search engine SearchMe, presented as a "cover flow"

The user can immediately see the content of the result websites and can select one hit by clicking on the left or right side of the current search result, which will cause the

next page to slide in the middle. The downside is that this visualization takes longer to load and you can only view one result at once. That is why SearchMe had to go offline in 2009.
This seems to be influenced by another approach that was developed in 1996. A search engine called WebForager used “virtual books”, so called WebBooks, to present their search results. Users can flip through the pages of the books. Every page represents another web site. They can also switch to another book to see another category of search results (see Figure 11) [4].

Fig. 11. The result page of the search engine *WebForager,* which displays the results as virtual books (WebBooks)

7 Conclusions

A good presentation and visualization of the search results is a crucial part of the search cycle of every search engine.
The basic look of search result listings has not changed much within the last ten years. However, there have been many small or subtle changes and new features have been added. Examples would be showing query terms in the search result surrogates, optimizing the summaries, differentiating and adapting the presentation for ambiguous queries, improving ranking algorithms and many others like those mentioned in chapter 4.
Other improvements include faster results and paying attention to small details in the design to improve the layout, font, color and spacing of the results.

References

1. Clarke, C., Agichtein, E., Dumais, S.,White, R.: The influence of caption features on clickthrough patterns in web search. In Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'07). ACM Press, New York, NY, USA, pages 135-142 (2007)
2. Goldstein, J., Kantrovitz, M., Mittal, V., Carbonell, J.: Summarizing text documents: sentence selection and evaluation metrics. In Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99). ACM Press, New York, NY, USA, pages 121–128 (1999)
3. Joachims, T., Granka, L., Pan, B., Hembrooke, H., Gay, G.: Accurately Interpreting Clickthrough Data As Implicit Feedback. *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'05)*. ACM Press, New York, NY, USA, pages 154–161 (2005)
4. www.infovis.net
5. Hearst, M.: Search User Interfaces

Pictures from www.google.com, www.wikipedia.com and www.searchuserinterfaces.com

Query specification in information retrieval

Alexander Knogl

Otto-Friedrich-Universität Bamberg, Lehrstuhl für Medieninformatik
alexander.knogl@stud.uni-bamberg.de

Abstract. Die Formulierung einer Suchanfrage stellt eine der wichtigsten Phasen im Information Retrieval dar. Bei der Formulierung von Suchanfragen stellen sich mehrere komplexe Probleme. Der Nutzer kann die von ihm gesuchte Information nicht präzise in Suchwörter fassen und darüber hinaus nicht wissen, welche Suchwörter zum Ziel führen. Im Laufe der Zeit haben sich Tools und Algorithmen rundum Suchmaschinen soweit entwickelt, dass die zuvor genannten Probleme abgeschwächt werden konnten. Dazu tragen beispielsweise Thesauri, Vorschlagslisten bei der Eingabe von Suchwörtern, Stemming, quorum-level Ranking und weitere Mechanismen bei. In der Seminararbeit wird zudem auf das unterschiedliche Suchverhalten der Nutzer Bezug genommen, die entweder mit ganzen Sätzen oder Suchwörtern nach Informationen suchen. Eine Unterscheidung Boolescher Operatoren, Statistiken zur Verwendung von Operatoren und Vorteile von Stoppwörtern in Verbindung mit Suchmaschinen schließen dieses Thema ab.

Keywords: Anomalous State of Knowledge, stemming, Google Instant, query types, keyword queries, stop words, synonyms, Boolean queries

1 Anomalous State of Knowledge

Über zwei Milliarden Menschen, 30% der Weltbevölkerung, nutzen pro Monat im Durschnitt ca. 16 Stunden das Internet. Dabei fällt mit einem Anteil von 21% die Nutzung auf die Suche nach Informationen aus.[1] Alleine die Suchanfragen des Suchmaschinenanbieters Google Inc. sind von 2010 auf 2011 um über eine Milliarde auf 4,7 Milliarden Suchanfragen pro Tag gestiegen. Innerhalb einem Jahr verzeichnete die Suchmaschine über 1,7 Billionen Suchanfragen.[2] Weil immer mehr Informationen online verfügbar sind, um Nutzern einen schnelleren Zugriff auf diese Informationen zu ermöglichen, ist es für den Nutzer wichtig diese Informationen einfach und schnell zu finden. In Folge dessen muss sich der Nutzer bewusst werden, wie er eine Suchanfrage stellt, um an seine gewünschten Informationen zu kommen. Belkin, Oddy und Brooks stellen zu diesem Problem die „Anomalous State of Knowledge“ Hypothese auf.

[1] http://www.go-gulf.com/blog/online-time, aufgerufen am 24.08.2012
[2] http://www.statisticbrain.com/google-searches/, aufgerufen am 24.08.2012

"The ASK hypothesis is that an information need arises from a recognized anomaly n the user's state of knowledge concerning some topic or situation and that, in general, the user is unable to specify precisely what is needed to resolve that anomaly. Thus for the purpose of IR, it is more suitable to attempt to describe that ASK than to ask the user to specify her/his need as a request to the system." [1]

Mit dem Begriff "Anomalous State of Knowledge" (ASK) beschreibt Belkin et al. ein Informationsproblem auf Seiten des Nutzers, dass durch eine Anomalie des Wissenszustandes in Bezug auf ein Thema oder einer Situation hervorgerufen wird. Der Nutzer kann dieses Problem nicht genau spezifizieren, da ihm die exakten Kenntnisse zu diesem Informationsproblem fehlen. Des Weiteren kann sich der tatsächliche Informationsbedarf des Nutzers von der letztendlichen Nachfrage des Nutzers unterscheiden, weil dieser die Suchanfrage bzw. das Informationsproblem nicht entsprechend zu formulieren vermag. Ein Informationssystem sollte demnach dem Nutzer, der sein Informationsbedürfnis nicht umfassend erfassen und nur teilweise artikulieren kann, bei der Beseitigung der Anomalie unterstützen und nicht dazu zwingen sein vages Informationsbedürfnis zu spezifizieren.

Ein Informationsbedürfnis reflektiert den genauen Bedarf des Nutzers und ist subjektiv geprägt. Unter Informationsbedarf hingegen wird nach Picot et al. [2] die Art, Menge und Qualität von Informationen verstanden, die in einer bestimmten Zeit von einer Person zur Erfüllung ihrer Aufgaben benötigt werden. Im Gegensatz zu einem Informationsbedürfnis wird bei einem Informationsbedarf der objektive Sachverhalt analysiert.

Suchmaschinen sollten demnach einen Nutzer bei seinem Prozess der Informationssuche unterstützen, dies kann in Form von Vorschlagslisten, Transformationen von Suchwörtern, das Anzeigen ähnlicher Suchanfragen, oder vielen weiteren Tools geschehen.

2 Textual Query Specification

Im Folgenden werden verschiedene Arten textueller Suchanfragen betrachtet. Dabei wird auf die Suche mit Suchwörtern, s.g. keywords und mit ganzen natürlichen Sätzen eingegangen sowie auf verschiedene Transformationen, die mit den eingegebenen Suchwörtern einer Anfrage, durch eine Suchmaschine durchgeführt werden.

2.1 Subject indexing and older bibliographic search systems

Der historische Hintergrund für die Motivation einer Suche ausgehend von einem expliziten Informationswunsch ein bedeutsames Dokument zu finden stammt aus dem Bibliothekskontext. Ein Nutzer sucht in diesem Fall Bücher oder Artikel der Bibliothek, die ihm dabei helfen sollen, seine konkrete Aufgabenstellung zu lösen. [3, S. 15]

In früheren Bibliotheks-Suchsystemen konnten Nutzer nur nach Metadaten suchen die manuell hinterlegt wurden und meistens nur den Inhalt eines Buches kurz zusammenfassten, sowie den Autor und den Titel beschreiben. Die Suche in einem online

Bibliothekskatalog war dem Nutzer folglich nur nach dem Autor, Titel und weiteren Metadaten möglich, die der Bibliothekar eigenhändig hinterlegt hat. [4]

Ein Buch mit interessanten Thesen und wissenschaftlichen Ergebnissen die nur zweitrangig in einem Buch vorkommen, wurden gewöhnlich nicht mit dazugehörigen Metadaten versehen, da der Aufwand zu hoch gewesen wäre, eine manuelle Indexierung auch für untergeordnete Themenbereiche vorzunehmen. [5]

Die Güte, die die Indexierung aufweist hängt überwiegend von der Person ab, die die Indexierung vornimmt. Diese Person muss eine fachliche Kompetenz vorweisen, um eine zutreffende Indexierung vornehmen zu können und diese einheitlich über den gesamten Zeitraum hinweg gestalten, damit für Bücher mit gleichen Themen dieselben Deskriptoren verwendet werden. Dieses Verfahren, der manuellen Indexierung, hat den Nachteil, dass es sehr aufwändig und teuer ist. Dadurch, dass die Indexierung manuell vorgenommen wird, dauert dieses Verfahren zudem sehr lange und die Qualität des Index hängt von den Fähigkeiten des Bibliothekars ab, der die Deskriptoren in den Index mitaufnimmt.

2.2 Keyword queries

Die Suche mit Suchwörtern s.g. keywords ist zurzeit die gängigste Methoden, um Suchanfragen im World Wide Web zu formulieren. Suchwörter bestehen aus einem oder mehreren Wörtern mit der Intension Dokumente zu finden, die diese Suchwörter enthalten.

Aula et al. [6] zitiert einen Experten, der über die unnatürliche Strategie des Suchens mittels Suchwörter folgendes aussagt: „Ich wähle Suchwörter nicht auf der Basis der spezifischen Information, die ich haben will, sondern vielmehr wie ich mir vorstellen könnte das jemand eine Seite so formuliert, die diese Informationen beinhaltet.“ Der Nutzer muss sich laut dieser Aussage in die Lage des Autors versetzen, wie dieser seinen Artikel beschreiben würde. Nur mit den vom Autor verwendeten Wörtern kann eine Suche erfolgreich sein.

Im Gegensatz zur Suche mit ganzen Sätzen, ist das Ziel einer Suche mit Suchwörtern ein Dokument zu finden, dass jedes eingegebene Suchwort enthält und das Dokument beschreibt. Bei der Suche mit ganzen Sätzen wird meist versucht eine Antwort auf den eingegebenen Satz zu bekommen, das Ergebnis muss dabei nicht zwingend alle eingegebenen Wörter des Satzes beinhalten. Dies zeigt sich vor allem, wenn der eingegebene Satz als Frage formuliert ist.

In früheren Jahren des World Wide Webs benutzten Suchmaschinen textstatistische Verfahren, um Suchergebnisse zu bestimmen. Dabei wird über die relative Häufigkeit von Suchwörtern auf die inhaltliche Bedeutung von Textdokumenten geschlossen. Ein Suchwort erweist sich insbesondere dann als bedeutsam, wenn es innerhalb eines Textdokumentes häufig vorkommt und in allen anderen Dokumenten selten enthalten ist. Hier spricht man von einer inversen Dokumenthäufigkeit. Als grundlegende Entscheidung ist dabei zu beachten, ob jedes Suchwort in die Statistik mit einfließen soll, oder ob eine bestimmte Auswahl an eingegebenen Suchwörtern nicht berücksichtigt werden soll. Diese Wörter werden in s.g. Stoppwortlisten gespeichert, die bei der Indexierung durch Platzhalter ersetzt werden. [7]

Die Bedeutung von AND-Verknüpfungen ist deutlich leichter zu verstehen als das statistische Rankingverfahren [8]. Wenn es sich jedoch nur um einen kleinen Datenbestand von Dokumenten handelt, ist ein Ranking-Algorithmus, der automatisch alle eingegebene Suchwörter mit einer logischen AND-Funktion zusammenfügt nicht optimal, da die Wahrscheinlichkeit hoch ist, dass der Nutzer ein leeres Ergebnis zurückbekommt. In Anbetracht der rapide angestiegenen Anzahl an Internetseiten in den letzten Jahren, hat sich dieser Nachteil aufgelöst und die logische AND-Funktion wurde die Basis für die Suche im Word Wide Web bei den meisten Suchmaschinenanbietern.

Neben der Suche mit keywords gibt es eine Suche, bei der der Nutzer einen ganzen Satz in das Suchfeld eingibt und dazu Informationen zurückbekommen will. In den meisten Fällen handelt es sich hierbei um Fragen auf denen der Nutzer eine Antwort erhalten möchte. Diese Form der Suche, mit ganzen Sätzen, wird als Suche mit „natürlicher Sprache“ bezeichnet [9]. Da der Nutzer in seiner sozialen Umgebung auch durch Fragestellungen in ganzen Sätzen nach Informationen frägt, ist diese Methode der Suche am einfachsten und intuitivsten.

Studien von, Pollock und Hockley 1997 [10], Bilal 2000 [11], und Schacter et. al. 1998 [12] zufolge geben vor allem Kinder und Nutzer, die zum ersten Mal eine Suchmaschine nutzen einen ganzen Satz ein, um ein Ergebnis zu finden. Mit der Suchmaschine Ask, die 1996 gegründet wurde[3], sollten Nutzer in ihrer natürlichen Sprache Fragen stellen können, die die Suchmaschine beantworten konnte. Diese Suchmaschine unterstütze zu diesem Zeitpunkt aber noch nicht ausreichend die gewünschte Funktionalität, um die Fragen der Nutzer zufriedenstellend zu beantworten.

Fig. 1. Kein Ergebnis auf die Suchanfrage nach dem Geburtsdatum von Barack Obama (Quelle: ask.com)

Figure 1 zeigt, dass auch im Jahr 2012 Suchmaschinen wie Ask, die speziell für Anfragen in ganzen Sätzen entwickelt wurden, selbst bei einfachen Fragen, wie nach dem Geburtsdatum von Barack Obama, keine konkrete Antwort zurückliefert.

[3] http://www.ask.com/wiki/Ask.com, aufgerufen am 14.06.2012

Komplexere und kaskadierende Fragen sind zudem ein großes Problem, dass fast alle Suchmaschinen vor eine unüberwindbare Hürde stellt, sowie die Unterstützung verschiedener Sprachen.

Führt die Suchanfrage zu einem Erfolg wie in *Figure 2*, muss abgewogen werden, ob die exakte Antwort, die Antwort in einem Satz, Paragraphen oder als ganzes Dokument zurückgegeben werden soll. Diese Abwägung ist schwierig, da man nicht eruieren kann wie viel zusätzliches Wissen der Nutzer zu seiner Suchanfrage noch benötigt.

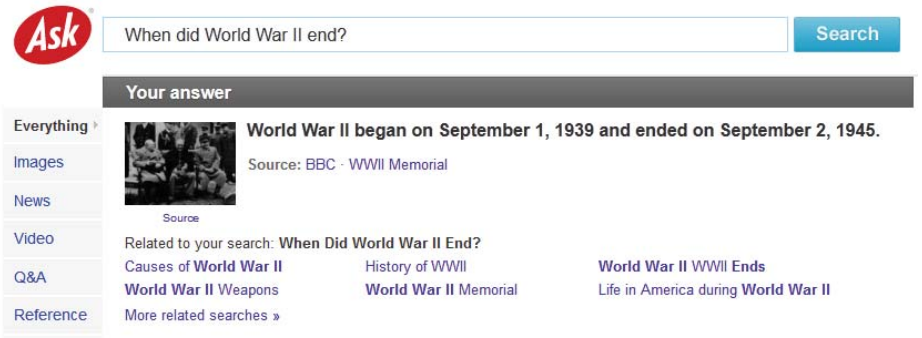


Fig. 2. Ergebnis der Suchanfrage über das zeitliche Ende des zweiten Weltkrieges (Quelle: ask.com)

Als weiterer Internetdienst kann Wolfram Alpha genannt werden. Die semantische Suchmaschine ist besonders im mathematisch naturwissenschaftlichen Kontext für gezielte Antworten auf Fragen ausgerichtet.

Pollock und Hockley [10] fanden außerdem heraus, dass neue Nutzer mit dem iterativen Suchen nicht vertraut sind. Sie nahmen an, dass wenn kein Ergebnis zu ihren Suchwörtern erschien, sie nicht in der Lage waren die Suchmaschine richtig zu bedienen oder die Suchmaschine keinen Datenbestand zu den eingegebenen Suchwörtern hatte. Eine Suche neu zu beginnen oder das Abändern von Suchwörtern waren diesen Nutzern fremd, genau wie das durchscrollen der Ergebnislisten, das Navigieren zu Webseiten und das Durchlesen ganzer Webseiten zur Suche ihres gewünschten Ergebnisses.

Im Jahr 2007 äußerte sich ein Mitarbeiter von Google Inc., dass sich Suchanfragen von „Give me what I typed“ zu „Give me what I want“ gewandelt haben.[4] Diese Aussage soll verdeutlichen, dass Google zu dieser Zeit schon Algorithmen entwickelt hat, die versuchen den Sinn der Suchanfrage zu erfassen, was der Nutzer mit den eingegebenen Suchwörtern finden möchte.

[4] http://www.nytimes.com/2007/06/03/business/yourmoney/03google.html?pagewanted=all, aufgerufen am 18.06.2012

2.3 Different types of queries

Suchanfragen lassen sich nach Border [13] in drei verschiedene Anfragetypen unterscheiden. Als Basis für diese Unterscheidungen dienten eine Nutzerbefragung sowie die Auswertung eines query logs der Suchmaschine AltaVista. Als Ergebnis diese Auswertung wurden folgende drei Arten von Abfragetypen identifiziert: navigational (navigationsorientiert); informational (informationsorientiert) und transactional (transaktionsorientiert).

Bei navigationsorientierten Anfragen ist das Ziel des Nutzers eine bestimmte Webseite zu suchen, wobei der Nutzer diese bereits kennt oder die Existenz dieser Seite vermutet. Nach dem Auffinden dieser Seite ist das Informationsbedürfnis des Nutzers befriedigt. Im klassischen Information Retrieval ist diese Art der Anfrage mit den known item searches vergleichbar. Diese Art von Suchanfragen werden auf zehn Prozent geschätzt [14].

Der Nutzer sucht bei informationsorientieren Anfragen thematisch passende Dokumente. Diese Art der Anfrage entspricht am meisten dem im klassischen Information Retrieval gestellten Anfragen. Nachdem der Nutzer die Informationen erhalten hat, ist die Suche beendet und die einzige weitere Aktion besteht im Lesen und Analysieren der gefundenen Dokumente. Informationsorientierte Anfragen machen mit 80% einen Großteil der Anfragen aus [14].

Transaktionsorientierte Anfragen beziehen sich auf anschließende Transaktionen die der Nutzer nach dem Auffinden der entsprechenden Seite durchführt. Diese Transaktionen können beispielsweise der Einkauf in einem Internetshop, der Download einer Datei oder Chats sein. Diese Art der Anfragen kommt so nicht im klassisches Information Retrieval vor und beschränkt sich entweder auf extra Systeme oder rein auf web-spezifisches Information Retrieval. Zehn Prozent der gesamten Anfragen entsprechen den transaktionsorienteren Anfragen [14].

Eine weitere Studie stellt einen signifikanten Unterschied zwischen der Art der Anfrage und der Anzahl der benutzen Suchwörtern fest. Informationsorientierte Suchanfragen werden im Durchschnitt mit 3,85 Suchwörtern ausgeführt, transaktionsorientierte Anfragen mit 2,43 Suchwörtern und navigationsorientierte Anfragen mit 1,7 Suchwörtern.[5] In den letzten Jahren fand bei dieser Verteilung der Anfragearten eine bedeutende Entwicklung statt. In der ersten Suchmaschinen-Generation, die sich noch am klassischen Information Retrieval orientierte lag der komplette Schwerpunkt bei informationsorientierten Anfragen. Durch die Integration webspezifischer Daten wie z.B. Linkstrukturen oder Ankertexten konnten sowohl informationsorientierte als auch navigationsorientierte Anfragen abgedeckt werden. In der dritten und vorerst letzten Stufe war das Ziel auch transaktionsorientierte Anfragen besser beantworten zu können. Suchmaschinen versuchen dabei die Nutzeranfragen besser zuzuordnen oder auch dem Nutzer in einem iterativen Prozess selbst einordnen zulassen. [13]

[5] http://www.librarystudentjournal.org/index.php/lsj/article/view/228/305#note4, Bridwell T.: Hourly analysis of navigational, transactional, and informational user-intents in search engine queries. University of Wisconsin Milwaukee, Wisconsin; School of Information Studies; Library Student Journal, June 2011, aufgerufen am 26.08.2012

Ein weiterer Unterschied zwischen den Anfragearten kann durch das jeweilige Ergebnis festgestellt werden. Während informationsorientierte Anfragen durch eine gewisse Anzahl von mehreren Dokumenten erfüllt werden, soll bei den transaktionsorientierten und navigationsorientieren Anfragen nur ein Dokument gefunden werden. Anfragen in natürlicher Sprache ließ Border bei seiner Studie zur Einteilung der Suchanfragen unberücksichtigt.

2.4 Statistics on query length

Dem Online-Forschungsunternehmen Experian Hitwise zufolge hat sich die Anzahl der Suchwörter in Suchanfragen in den letzten Jahren deutlich erhöht. Eine Studie, die die Jahre 2007 bis 2009 betrachtet zeigt, dass Suchanfragen, die nur ein Wort beinhalten um bis zu 20% zurückgegangen sind, währenddessen Suchanfragen mit vier oder fünf Wörtern um über zehn bzw. 13% gestiegen sind.[6]

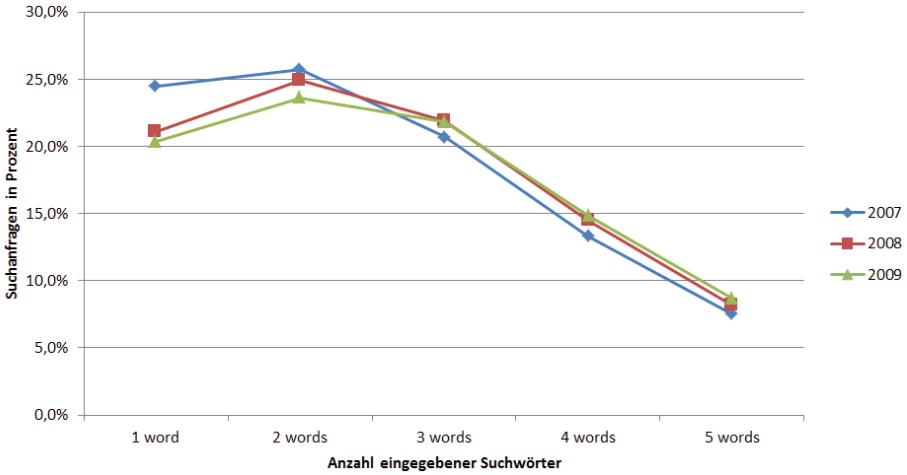


Fig. 3. Statistik über die Anzahl der eingegebenen Suchwörter für Suchanfragen[6]

Dieses veränderte Verhalten der Nutzer, immer mehr Suchwörter für Suchanfragen zu verwenden, resultiert aus einer Vielzahl von Eigenschaften. Zum einen gibt es immer mehr Internetseiten mit gleichen oder ähnlichen Inhalten im World Wide Web, was zu einer Vielzahl von Suchtreffern führt, die nur durch die Verwendung von zusätzlichen Suchwörtern zu besseren Ergebnissen führen. Zum anderen werden dem Nutzer von den Suchmaschinen immer mehr Vervollständigungen der Suchanfragen oder zusätzliche Suchvorschläge angeboten. Diese Suchvorschläge verfügen über gleichviel oder mehr Suchwörter als die eingegebene Suchanfrage des Nutzers. Ein weiterer Punkt ist die gestiegene Erfahrung der Nutzer die sich im Laufe der Zeit entwickelt hat und ein gewisser Anspruch an Suchergebnisse durch die Nutzer.

[6] http://www.blindfiveyearold.com/the-future-of-search-is-numbered, aufgerufen am 20.07.2012

2.5 Automated Transformations of Textual Queries

Suchmaschinen formen testbasierte Anfragen in bestimmten Fällen nach gewissen Regeln um. Dabei normalisieren einige Systeme automatisch morphologische Varianten von Wörtern. So wird beispielsweise aus dem Wort „dogs“ sowohl „dogs“ als auch „dog“, beide Wörter werden dementsprechend mit einer OR-Verknüpfung zusammengefasst. Die Zurückführung verschiedener morphologischer Varianten eines Wortes auf ihre Stammform wird Stemming genannt. Im informationswissenschaftlichen Kontext kann Stemming und Lemmatisierung als ein Wort zusammengefasst werden. [15, S. 106ff]

Affix removal, das Entfernen von Prä- und Suffixen, table lookup, ein Ansatz basierend auf Wörterbüchern und die N-Gram-Methode sind verschiedene Ansätze um Stemming durchzuführen. [15, S. 106ff] [16]

Affix removal ist die am weitesten verbreitete Art des Stemming, da die Anwendung einfach zu implementieren ist und wenig Aufwand benötigt wird. Besonders effektiv erweist sich das Entfernen von Affixen bei stark flektierenden Sprachen, wie z.B. der englischen Sprache. Durch das Entfernen von Affixen werden so regelbasiert Pluralformen von Wörtern auf den jeweiligen Wortstamm reduziert. Das Wort „cats“ wird in seiner Pluralform durch das Entfernen seines Affix auf das Wort „cat“ reduziert. Neben den Vorteilen der leichten Implementierung und schnellen Anwendung gibt es auch Nachteile, dass Wörter falsch gestemmt werden und somit ein Präzisionsverlust einhergeht. Die meisten dieser Stemmingverfahren sind iterativer Natur und entfernen längstmögliche Affixe wie z.B. Lovins-68 und Porter-80. Manche stemmen auch flach durch das Entfernen bestimmter Flexionen (Inflection Stemmer).[7]

Der Porter-Stemming-Algorithmus von Martin Porter 1980 ist einer der bedeutendsten in diesem Gebiet. Dieser wendet auf ein zu stemmendes Wort eine Menge von Verkürzungsregeln an, bis dieses Wort nur noch eine minimale Anzahl an Silben aufweist.[8] Table lookup wird vorwiegend in der deutschen Sprache eingesetzt, da das regelbasierte affix removal Verfahren wegen der komplexen Wortbildung im Deutschen und den häufigen Ausnahmen nicht eingesetzt werden kann. Der Nachteil des table lookup Verfahrens ist die manuelle Pflege des Wörterbuchs, das bei jedem neuen Begriff aktualisiert werden muss.

Term	Stem
Spiel spielerisch gespielt Spieler	spiel

Fig. 4. Table lookup am Beispiel des Wortes „spiel“

[7] http://www.cl.uni-heidelberg.de/~weissman/Ausarb.pdf, aufgerufen am 17.08.2012

[8] http://www.comp.lancs.ac.uk/computing/research/stemming/general/porter.htm, aufgerufen am 17.08.2012

Der N-Gram-Methode liegt ein automatisches Verfahren zu Grunde, das Wörter und Komposita in ihre Bestandteile zerlegt. Für die jeweilige Umwandlung werden verschiedenste morphologische Regeln angewandt. N-Gramme zerlegen einen Term in gleich große Substrings einer bestimmten Länge. Je nach Größe der Zeichenketten werden die n-Gramme Monogramme, Bigramme, Trigramme, Tetragramme und Pentagramme genannt. Der Wortanfang bzw. Wortende wird anschließend mit Leerzeichen aufgefüllt, um die gleiche Wahrscheinlichkeit zu gewährleisten, dass die Zeichen am Wortanfang und Wortende in einem n-Gram vorkommen. Bei einer Suchanfrage werden die eingegebenen Suchwörter in n-Gramme zerlegt und mit den n-Grammen der Datenquelle der Suchmaschine verglichen. Ein Dokument ist dabei besonders relevant, wenn es häufig in einem Dokument vorkommt. [17]

Dieser Ansatz ist ein sprachunabhängiges Verfahren und durch die automatische Zerlegung in Teile der Länge *n* kostengünstig. [3, S. 121]

Die Zurückführung eines Wortes auf seine Grundform ermöglicht es alle möglichen Formen eines Wortes zu finden. Dadurch werden potentiell mehr relevante Dokumente gefunden. Durch Stemming wird weiterhin die Indexdatei wesentlich verkleinert.

2.6 Stop words

Stoppwortlisten dienen dazu, häufig auftretende Wörter oder Wörtern denen keine Relevanz beigemessen wird, in einer Suchanfrage heraus zu filtern. In diesen Listen werden z. B. bestimmte und unbestimmte Artikel, Präpositionen und Konjunktionen gespeichert. Diese Wörter werden vorwiegend für einen grammatikalischen und syntaktischen Satzbau gebraucht und lassen wenig Rückschlüsse auf den eigentlichen Inhalt eines Dokumentes zu. Stoppwörter werden in einer Liste zusammengefasst, wobei jede Liste nur Stoppwörter einer Sprache beinhaltet. [9]

Sobald ein Information Retrieval System gestartet wird, werden diese Stoppwortlisten in dafür geeignete Datenstrukturen geladen, beispielsweise in einer Hashtabelle oder einem AVL-Baum. Dadurch wird es möglich zu identifizieren, ob einzelne Wörter ein Stoppwort repräsentieren und folglich aus der Repräsentation von Dokumenten und Anfragen zu entfernen sind. [3, S. 97]

Bei der Indexierung werden diese Wörter nicht berücksichtigt, sind jedoch für die Suche nach Phrasen wichtig. Diese Stoppwörter werden bei der Indexierung mit Platzhalter ersetzt [18, S. 48]. Würden Stoppwörter nicht gefiltert werden bei einer Suchanfrage, würde die Ergebnisliste nahezu jedes Dokument einschließen. Dies wäre für den Nutzer nicht zielführend, daher werden Stoppwörterlisten benutzt, um eine Steigerung der Effizienz von Suchmaschinen zu erreichen.

Ein Pionier in diesem Gebiet war Hans Peter Luhn, der die Annahme traf, dass die Häufigkeit, wie oft ein Wort in einem Dokument vorkommt, ein Indikator für dessen Signifikanz ist. Gleichwohl werden aber neben den häufig auftretenden Wörtern, Stoppwörtern, auch zu seltene Begriffe ausgeschlossen. Nach Luhn finden sich die

[9] http://de.wikipedia.org/wiki/Stoppwort, Version 15.12.2011 17:08 Uhr, aufgerufen am 13.06.2012

Textwörter mit guter Signifikanz für die Indexierung in der Mitte der Verteilung. Der Graph in *Figure 5* zeigt einen typischen Verlauf von Worthäufigkeiten innerhalb einer Dokumentenkollektion. Die wichtigsten Terme mit der größten Signifikanz befinden sich nach Luhn bei dem Punkt E. Ausgehend von diesem Punkt nimmt die Signifikanz der Wörter im Verlauf nach links und rechts immer weiter ab. [19]

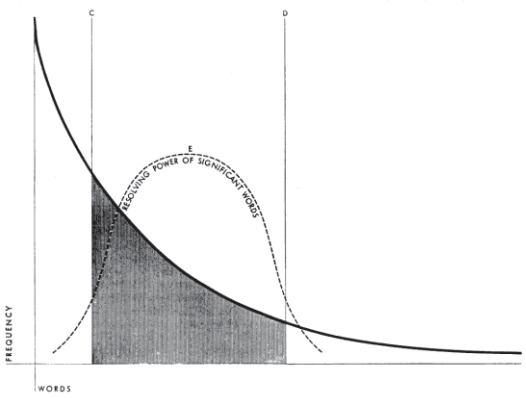


Fig. 5. Darstellung der signifikanten Textwörter in einem Dokument [19, S. 161]

Ein Vorteil durch die Nicht-Indexierung der Stoppwörter ist zum einen die Reduzierung des Speicherplatzbedarfes um bis zu 50% durch die Eliminierung häufig vorkommender Wörter. Aus der Reduktion des Speicherplatzbedarfes resultiert eine Performancesteigerung der Matchingalgorithmen durch die reduzierte Datenmenge. Zum anderen wird eine Verbesserung der Precision und des Recalls erreicht. [3, S. 96]

Ein typisches Beispiel für die Phrasensuche war in den früheren Jahren des Information Retrieval die Suchanfrage „to be or not to be“. Diese Suchanfrage hat neben den Stoppwörtern „to“ und „be“, noch Operatoren wie „or“ und „not“, was zu einem leeren Ergebnis geführt hat. Die Algorithmen haben sich jedoch so weit verbessert, dass solche Zitate trotz vorkommender Stoppwörter und Operatoren richtig interpretiert werden.

Der Ranking-Algorithmus von Suchmaschinen muss den idealen Weg finden, Transformationen an Wörter die versteckt vorgenommen wurden hilfreich zu gestalten und dem Nutzer nicht das Gefühl geben, dass sich das System unerklärlich verhält.

3 Query specification via entry form interfaces

Das Standardinterface für Suchanfragen ist heutzutage ein Eingabefeld mit einem Suchbutton. Suchformulare schließen oft eine Drop-Down Liste mit ein, die entweder die zuletzt gesuchten Anfragen anzeigt, die zu den Präfixen passen, die gerade eben eingegeben wurden oder schlagen eigens generierte Vorschläge auf Grund bestimmter Matching-Verfahren vor. Ein Beispiel ist in *Figure 6* zu sehen.

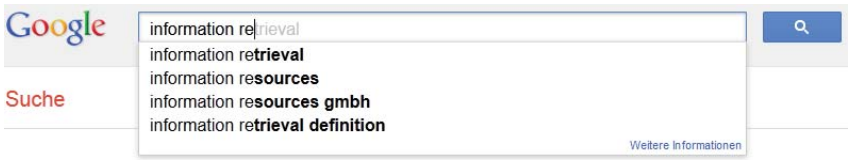


Fig. 6. Drop-Down Liste von Vorschlägen der Suchmaschine Google (Quelle: google.de)

Webseiten, deren Inhalt in verschieden Kategorien geteilt ist, benutzen oft eine Kombination aus einem Drop-Down Menü, um eine Kategorie auszuwählen in der gesucht werden soll, in Verbindung mit einem Eingabefeld für die Suchwörter. Als Beispiel kann Amazon oder Ebay genannt werden.

Fig. 7. Drop-Down Menü für verschiedene Kategorien von Amazon (Quelle: amazon.de)

Forscher haben vermutet, dass kurze Eingabefelder zu kurzen Suchanfragen führen. Diese Hypothese wurde von Franzen and Karlgen [20] mit zwei verschiedenen Eingabefeldern getestet. Das erste Eingabefeld bestand aus einer Zeile und konnte bis zu 18 Zeichen sichtbar darstellen, jedoch akzeptierte das Eingabefeld bis zu 200 Zeichen. Das zweite Eingabefeld zeigte sechs Zeilen zu je 80 Zeichen, in denen fast beliebig lange Suchanfragen eingegeben werden konnten. Nach der Untersuchung stellte sich ein signifikanter Unterschied dar, in das kürzere Eingabefeld wurden 2.81 Suchwörter eingegeben, in das längere Eingabefeld 3,43 Suchwörter.

Eine ähnliche Theorie bestätigte auch Belkin et al. [21] in der davon ausgegangen wird, dass der Text, der über Suchformulare steht die Suchanfrage des Nutzers beeinflusst. Als Eingabefeld für die Suchanfrage stand ein großes Textfeld über fünf Zeilen zur Verfügung. Über einem der Textfelder stand der Ausdruck „Query Terms“, über dem anderen „Information problem“ und zusätzlich „Please describe your information problem in detail: (The more you say, the better the results are likely to be.)“. Im Durschnitt wurden in das zuletzt genannte Textfeld 6,02 Wörter eingegeben, dem gegenüber standen 4,19 Wörter in dem Textfeld mit der Überschrift „Query Terms“.

Die Adresszeile eines Webbrowsers wird fälschlicherweise auch von unerfahrenen Nutzern verwendet, um nach Informationen zu suchen. In früheren Jahren verlief die Suche mit der Eingabe von Suchwörtern in die Adresszeile des jeweiligen Webbrowsers ergebnislos. Nach und nach unterstützten jedoch immer mehr Webbrowser eine Suche über die Adresszeile, folglich verschwand der Unterschied zwischen der Adresszeile des Browsers und des Suchformulars einer Suchmaschine.

Bevor ein Nutzer ein Suchformular zur Eingabe seiner Suchanfrage zur Verfügung hat sollte er nicht dazu gezwungen werden seine Suche vorher Einschränken zu müssen. Die Suche sollte anfangs immer alle Dokumente miteinschließen, mit der Option diese während oder nach der Suche zu verfeinern.

4 Synonyms

Als synonym werden zwei oder mehr Wörter bezeichnet, die in ihrer Bedeutung gleich oder ähnlich sind. Synonyme werden, genau wie Ober- und Unterbegriffe in s.g. Thesauri gespeichert, um den Nutzer bei der Suche zu unterstützen. Bei der Verwendung von Suchwörtern, zu denen Synonyme vorhanden sind, ist die Ergebnismenge relevanter Dokumente geringer. Werden bei der Suche schon mehrere Synonyme verwendet, erhöht sich die Chance, dass das relevante Dokument eher gefunden wird, jedoch wird ein Nutzer nicht alle bekannten Synonyme kennen, oder bei der Suche eingeben. Diese grundsätzliche Annahme kann deshalb nicht getroffen werden, des Weiteren gibt es dadurch Probleme bei der Wortzählung und das dafür ermittelte Ranking der Dokumente. Dokumente mit unterschiedlichen Synonymen für einen Begriff, lassen sich zwar dadurch bei einer Suchanfrage zu den entsprechenden Synonymen finden, werden aber beim Ranking benachteiligt. [15, S. 112]

Bevor die Suche ausgeführt und ein Ergebnis zurückgeliefert wird, findet eine Suche anhand eines Thesaurus nach Synonymen oder anderen verwandten Begriffen zu dem Thema der Anfrage statt. Bei einem Treffer werden diese Synonyme anschließend der ursprünglichen Anfrage im Hintergrund hinzugefügt und die modifizierte Anfrage ausgeführt. Durch die große Anzahl von Dokumenten im Internet ist es jedoch fraglich, ob ein Abgleich anhand eines Thesauri mit Synonymen überhaupt notwendig ist um die Treffermenge zu erhöhen, da meistens ohnehin genügend Dokumente als Ergebnis zurückgegeben werden.

Bei der Suche nach „District of Columbia Museums“ ist das erste Suchergebnis eine Seite, die den Begriff District of Columbia gar nicht beinhaltet. Google scheint hier eine Art Transformation zwischen Disctrict of Columbia und der Abkürzung D.C. vorzunehmen. Es ist ersichtlich, dass das Akronym D.C. fettgedruckt ist, wie die anderen Suchbegriffe. Google hebt üblicherweise alle Suchbegriffe fett hervor, wenn diese in den Suchergebnissen vorkommen, um den Nutzer zu zeigen, dass die Ergebnisliste auf die eingegebenen Suchbegriffe zutrifft.[10]

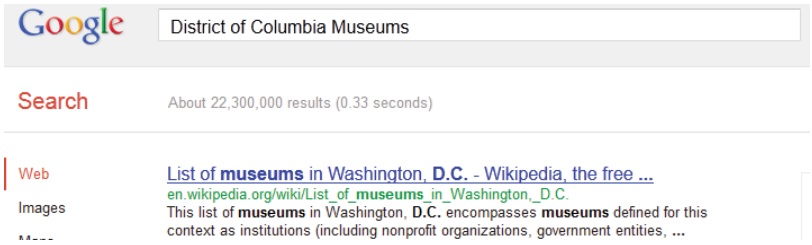


Fig. 8. District of Columbia wird in das Akronym D.C. transformiert (Quelle: google.de)

Ein weiteres Beispiel betrifft die beiden englischen Wörter „pictures“ und „photos“, die unter den meisten Umständen dasselbe bedeuten. Bei der Suche nach „pic-

[10] http://www.seobythesea.com/2009/12/how-google-may-expand-searches-using-synonyms-for-words-in-queries/, aufgerufen am 19.06.2012

tures developed with coffee“ muss eine Suchmaschine in der Lage sein zu erkennen, dass auch wenn eine Webseite „photos“ beinhaltet und nicht „pictures“ die Seite dennoch relevant für den Nutzer ist.[11] Das Wort „photos“ ist bei der Ergebnisliste der Suche fettgedruckt, obwohl der Suchbegriff „pictures“ verwendet wurde.

Für das System der Synonymerkennung in Verbindung mit dem Website-Ranking hat Google fünf Jahre Entwicklungszeit investiert. Ein System zur Synonymerkennung hat eine Verbesserung des Recalls, aber auch eine Reduzierung der Precision zur Folge.

5 Dynamic Term Suggestions During Query Specification

Google bietet seit 2008 „Google Suggest“ bzw. „Autocomplete“ auf ihrer Webseite an, wobei erste experimentelle Versuche bereits 2004 gestartet sind. Mit Google Instant hat im Jahr 2012 eine weiterentwickelte Funktion Einzug gehalten, die automatisch Suchresultate zu bereits eingegebenen Buchstaben oder Wörtern präsentiert und diese dynamisch mit jedem weiteren Buchstaben ändert und anpasst. Alle Suchvorschläge basieren dabei auf echte Suchanfragen die von Nutzern eingegeben wurden. Die Popularität, also die Häufigkeit der eingegebenen Suchwörter, bestimmt dabei das Ranking der vorgeschlagenen Wörter. Laut Google werden jedoch auch andere Verfahren benutzt, um die Vorschlagsliste zu generieren, trotzdem wird jeder Vorschlag einzig durch eine bereits durchgeführte Suche von einem Nutzer ausgewählt. [12]

Nicht jeder Nutzer sieht weltweit allerdings die gleichen Vorschläge zu denselben eingegebenen Suchwörtern. Diese Vorschläge werden von Google auf den jeweiligen Standort des Nutzers und dessen Sprache angepasst. Zum einen kann der Nutzer manuell seinen Standort in seinem Google-Konto hinterlegen, oder automatisch beispielweise mittels GPS bestimmen lassen. Zum anderen benutzt Google die Top-Level-Domain oder IP-Adresse des Nutzers, um seinen Standort ungefähr zu bestimmen. Die Standortsbestimmung geht dabei nicht nur auf das Land des Nutzers, sondern sogar auf den Staat bzw. Bundesland bis hin zur Stadt bei denen unterschiedliche Vorschläge bei der Suche generiert werden. Als Beispiel dient hierzu die Suche über Google.com nach „Coupons“ durch die Top-Level-Domain des amerikanisches Unternehmens VeriSign Inc. werden dem Nutzer Vorschläge wie „Coupons for walmart“ oder „Coupons for target“ präsentiert. Diese Vorschlagsliste wird in Deutschland über Google.de nicht präsentiert, da diese Händler in Deutschland nicht existieren.[13]

Sollte der Nutzer bei der Eingabe der Suchwörter einen Rechtschreibfehler machen oder getrennte Wörter zusammenschreiben, werden diese soweit wie möglich von Google Instant ignoriert und die richtigen Suchvorschläge unterbreitet, in denen die korrekte Schreibweise angezeigt wird. Gibt ein Nutzer beispielweise „Hägen Daszs“ ein wird ihm trotz Rechtschreibfehler eine Vorschlagsliste mit der richtigen Recht-

[11] http://googleblog.blogspot.de/2010/01/helping-computers-understand-language.html, aufgerufen am 19.06.2012

[12] http://searchengineland.com/how-google-instant-autocomplete-suggestions-work-62592, aufgerufen am 22.06.2012

[13] ebd.

schreibung „Hägen Dazs“ generiert. Obwohl Google Instant beim Ranking der vorgeschlagenen Wörter die Popularität der Wörter über einen längeren Zeitraum betrachtet, können Wörter die sich schnell großer Beliebtheit unter den Nutzern erfreuen, an erster Stelle vorgeschlagen werden. Google richtet so seine Vorschlagsliste nach aktuellen Ereignissen aus. Folglich wird im Jahr 2012 bei der Eingabe des Wortes „Fußball“ als erster Vorschlag „Fußball em 2012“ präsentiert, da zu dieser Zeit viele Leute Informationen über die aktuelle Europameisterschaft suchten. Wie schnell solche neuen Begriffe in die Vorschlagsliste aufgenommen werden verrät Google allerdings nicht. Beobachtungen zufolge soll sich ein neuer Trend bei den Suchwörtern jedoch schon innerhalb von wenigen Stunden auswirken.[14]

Google filtert automatisch generierte Suchvorschläge heraus die gegen die Richtlinien von Google verstoßen. Dazu gehören u.a. personenbezogene Daten, Hass- und Gewaltverbreitungen und jugendgefährdendes Material. Solche Vorschläge werden in einer Blacklist gesammelt und dem Nutzer bei der Eingabe von Suchwörtern nicht vorgeschlagen. Eine direkte Entfernung von Suchvorschlägen durch Privatpersonen oder Firmen ist auf der Seite von Google nicht durchführbar. Ein Produktmanager von Google, der für Google Instant zuständig ist, Jonathan Effrat, weist darauf hin, dass gewöhnlicher Weise keine Entfernung seitens Google vorgenommen wird. Mit der Begründung, dass viele Nutzer nach diesen Begriffen suchen und somit kein legitimer Grund besteht diese Begriffe zu entfernen.[15] Jedoch muss in diesem Bereich ein gewisser Balanceakt gehalten werden, wie beispielsweise die Suchanfragen nach „PIP Implantate“ oder „Foxconn“ zeigen, bei denen weitere Suchwörter wie „Schadensersatz oder Klage“ bzw. „Selbstmord“ vorgeschlagen werden.[16]

Durch die von Google eingeführte automatische Vorschlagsliste kann natürlich auch ein neues Geschäftsmodell entstehen, indem man von dem üblichen Ansatz weggeht und die Vorschläge nicht nach der Popularität gewichtet, sondern durch ein Bezahlmodell. Firmen bezahlen z.B. dafür, dass sie bei der automatischen Vorschlagsliste möglichst hoch gerankt werden. Als fiktives Beispiel kann der Gebrauchtwagenmarkt von Pkws dienen. Gibt der Nutzer „Gebrauchtwagen“ ein kann als Vorschlag an erster Stelle „Gebrauchtwagen BMW“ erscheinen, weil BMW für diesen Vorschlag eine gewisse Summe an Google gezahlt hat und somit vor den Konkurrenten wie Mercedes-Benz oder Audi gerankt wird. Google selbst hat zwar in diese Richtung noch keinerlei konkrete Andeutungen gemacht, jedoch würde sich ein solches Geschäftsmodell als Erweiterung für Google Adwords anbieten.

6 Query Specification using Boolean and other Operators

Viele kommerzielle Volltextsysteme und Bibliotheks-Systeme unterstützten vor dem Aufblühen des World Wide Webs nur boolesche und kommandobasierte Anfra-

[14] http://searchengineland.com/how-google-instant-autocomplete-suggestions-work-62592, aufgerufen am 22.06.2012

[15] ebd.

[16] Ergebnis der Suche nach „PIP Implantate“ sowie „Foxconn“. Durchgeführt mit der Suchmaschine Google am 26. August 2012

gen. Nutzer konnten damals in diesen Systemen meist nur Anfragen über eine Zusammenfassung von Dokumenten verfassen, weil nicht alle Dokumentenbestände in elektronischer Form vorlagen. [22]

Obwohl viele erfahrene Nutzer Systeme mit boolescher Kommandosprache nutzen und einige Zeit für die Formulierung ihrer Anfragen aufwendeten, waren sich viele der Nutzer nicht immer im Klaren wie das Resultat aussehen könnte. Ein Problem war dabei, dass die Ergebnisliste entweder zu groß war oder aus einer leeren Menge bestand, weil Nutzer zu viele Suchwörter mit AND- bzw. OR-Verknüpfungen verbunden haben. [3, S. 175]

Ein weiteres Problem ergibt sich aus der Semantik von booleschen Anfragen, die für viele Nutzer kontraintuitiv ist. Viele Nutzer interpretieren die Bedeutung von AND und OR falsch. AND wird von den Nutzern fälschlicherweise oft als Erweiterung des Suchfeldes verstanden, um mehr Informationen über ein bestimmtes Thema zu bekommen. Bei Anfrage mit den Suchwörtern „Haus AND Wohnung“ wird folglich fälschlicherweise erwartet, dass Dokumente über das Thema Haus und das Thema Wohnung erscheinen und nicht Dokumente in denen Haus und Wohnungen gleichzeitig vorkommen.

In der täglichen Umgangssprache kann eine Anfrage mit den Suchwörtern „Kaffee OR Tee“ so interpretiert werden, dass entweder nach Kaffee oder Tee gesucht wird. Demzufolge genau das Gegenteil gemeint wird. Systeme die ausschließlich auf Boolesche Abfragen spezialisiert sind vergleichen die eingegebenen Suchwörter nicht auf Ähnlichkeiten mit den zurückgegebenen Dokumenten, entweder kommen exakt die eingegebenen Suchwörter vor oder es wird kein Ergebnis zurückgegeben.[17]

6.1 Post-Coordinate and Faceted Boolean Queries

Eine Technik um die Reihenfolge von Ergebnissen einer Booleschen Suchanfrage darzustellen ist das post-coordinate oder quorum-level Ranking. Bei diesem Verfahren werden Dokumente nach der Anzahl der Teilmengen, die eine Suchanfrage beinhaltet gerankt. Bei einer Suchanfrage, die die Wörter Karotte, Tomate, Erdbeere und Gurke enthalten, werden Ergebnisse zurückgegeben die mindestens eines dieser vier Wörter enthalten. Ist in einem Dokument mehr als ein Begriff enthalten wird dieses höher gerankt, als ein Suchergebnis bei dem der Begriff nur einmal, dafür aber sehr oft vorkommt. In Folge dessen wird das Dokument, welches Karotte, Gurke und Tomate beinhaltet höher gerankt als ein Dokument, dass 20-mal das Wort Karotte aufweist. Ein gutes Beispiel für die Visualisierung eines post-coordinate oder quorum-level Ranking ist die Auktionsplattform ebay. Bei einer Suchanfrage mit den Suchwörtern „Salamon XA Pro 4D Ultra GX“ liefert die Auktionsplattform keine Übereinstimmung zurück. Der Nutzer wird jedoch darauf aufmerksam gemacht, wie viele Suchresultate es geben würde, wenn die Suchanfrage nur aus *k* von *n* Suchwörtern bestehen würde. Wie in *Figure 10* zu sehen ist, gibt es für die Suchanfrage „Salamon

[17] http://dspace.library.cornell.edu/bitstream/1813/6351/1/82-511.pdf, aufgerufen am 23.08.2012

XA Pro Ultra GX“, also ohne dem Suchwort „4D“ 1323 Artikel die mit der Anfrage übereinstimmen.

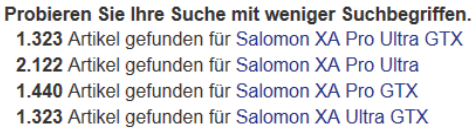


Fig. 9. Quorum Ranking Vorschläge am Beispiel von ebay (Quelle: ebay.de)

Eine andere Vorgehensweise, um die Ergebnisliste von Booleschen Suchanfragen zu verbessern ist, dem Nutzer die Suchanfrage in unterschiedliche Themenbereiche (Facts) aufteilen zu lassen. Jeder Themenbereich soll durch den Nutzer mit einer Anzahl von Begriffen genauer spezifiziert werden. Die Begriffe der einzelnen Themenbereiche werden mit einer OR-Verknüpfung verbunden. Anschließend wird die gesamte Suchanfrage, folglich die einzelnen Themenbereiche, mit einer AND-Verknüpfung verbunden, damit mindestens ein Begriff jedes Themenbereichs in der Ergebnisliste vorhanden ist. [26]

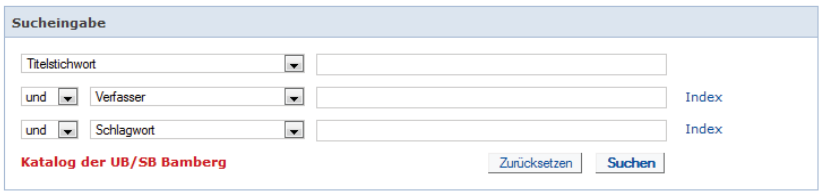


Fig. 10. Suchanfrage mit verschiedenen Facets am Beispiel der Bibliothekssuche der Otto-Friedrich-Universität Bamberg (Quelle: https://katalog.ub.uni-bamberg.de/InfoGuideClient.ubgsis/search.do?methodToCall=switchSearchPage&SearchType=2)

Das Beispiel der Bibliothekssuche der Otto-Friedrich-Universität Bamberg zeigt ein typisches Suchlayout für facettierte Boolesche Suchanfragen. Der Nutzer kann zu verschiedenen Themenbereichen wie z.B. Verlagsort, Jahr, Bemerkungen, Verfasser und Titel Suchbegriffe eingeben und diese entweder mittels „und“, „oder“, sowie „und nicht“ verbinden. Bei diesem Verfahren werden Suchresultate, die mindestens einen Begriff pro Themenbereich beinhalten höher gerankt als Suchresultate, die nur mit Suchwörtern aus wenigen Themenbereichen übereinstimmen.

6.2 Web-based Improvements to Boolean Query Specification

Mitte der neunziger Jahre wurden syntaktische Suchanfragen von Suchmaschinen unterstützt, weil diese eine einfachere und intuitivere Syntax bieten, als die bisherigen Operatoren. Die Suchmaschine AltaVista führte z.B. den „+“-Operator ein, dieser wurde als Präfix vor einen Suchwort verwendet, um damit anzugeben, dass dieser Suchbegriff in den Suchergebnissen vorhanden sein muss. Zu dieser Zeit wurden

statistische Algorithmen verwendet, um Suchergebnisse in einer Rangliste einzuordnen. Demzufolge konnte es vorkommen, dass das beste Suchresultat bei einer Suchanfrage mit drei Suchwörtern nicht alle enthalten hat. Statistische Algorithmen gewichten nämlich ein Dokument, dass 100-mal ein Suchwort enthält höher, als Dokument, in dem nur jedes der drei Suchwörter einmal vorkommt. Für den Nutzer war die Einführung des „+"-Operators ein Vorteil, um mehr Kontrolle über die gewünschten Suchresultate zu den eingegebenen Suchwörtern zu erlangen. Gleichwohl waren aber viele Nutzer mit der Verwendung des „+"-Operators überfordert, da er als AND-Verknüpfung zwischen zwei Wörtern gesehen wurde. Eine Suchanfrage mit „Hamster + Käfig" wurde falsch interpretiert, in dem Glauben, dass sowohl das Wort Hamster, als auch das Wort Käfig in den Suchergebnissen vorkommen muss. Jedoch bedeutet diese Suchanfrage, dass der Begriff Käfig vorkommen muss, der Begriff Hamster jedoch optional ist.[18]

6.3 Advanced Search Feature Statistics

Wie bereits vorgestellt, gibt es eine Vielzahl von Operatoren mit denen Suchanfragen spezifiziert werden können. Präzise Statistiken, die die Nutzung von Operatoren aufzeigen, sind von den großen Suchmaschinenanbietern nicht zu eruieren. Nachfolgende Zahlen zur Nutzung von Operatoren durch Nutzer sind deshalb von Web-Logs freier Suchmaschinenanbieter. Einer älteren Studie von Spink et. al. [27] zeigt, dass bei über einer Million ausgewerteten Suchanfragen weniger als fünf Prozent der Suchanfragen einen Booleschen Operator enthielten. Bei drei Prozent der Anfragen wurde die AND-Verknüpfung, bei einem Prozent die OR-Verknüpfung gewählt. Die AND NOT Verknüpfung wurde nur in 0,0003% der Fälle verwendet. Der „+"-Operator wurde in fünf Prozent der Anfragen verwendet, jedoch in drei Prozent der Fälle falsch eingesetzt. Bei dem „-"-Operator zeigte sich die gleiche Statistik mit einer Nutzungsrate von fünf Prozent und einer Fehlerrate von zwei Prozent. Die doppelten Hochkommata weisen ebenfalls eine Häufigkeit von fünf Prozent auf. Fehleingaben traten jedoch nicht nur bei der Verwendung der „+" und „-" Operatoren auf, sondern es wurde festgestellt, dass auch die Booleschen Operatoren falsch verwendet und zum Teil klein geschrieben wurden.

Eine neuere Studie von Jansen et al. [28] bei der über 1,5 Millionen Suchanfragen ausgewertet wurden zeigt, dass 2,1% der Suchanfragen Boolesche Operatoren enthielten und 7,6% andere Operatoren, zum Großteil doppelte Hochkommata.

Erweiterte Suchformulare werden einer Studie von Machill et al. [29] noch seltener benutzt, als die Booleschen Operatoren. Eine weitere Studie von Spink und Jansen [30 S. 79] kommt zudem zu der Schlussfolgerung, dass keine signifikante Entwicklung in der Nutzung von Operatoren ersichtlich ist. Dies wird wahrscheinlich durch die immer besseren Suchalgorithmen gefördert, die dem Nutzer gute Suchresultate, trotz nicht verwendeter Operatoren, zurückgeben.

---

[18] http://doc.utwente.nl/66886/1/glot.pdf, D. Hiemstra, F. de Jong: Statistical Language Models and Information Retrieval: natural language processing really meets retrieval, aufgerufen am 24.08.2012

7 Query Specification Using Command Languages

In einer Kommandosprache wird eine Syntax mit Befehlen, meist auch Verben genannt, verwendet, gefolgt von Argumenten. Ein Beispiel einer Kommandosprache aus dem System Melvyl der University of California sieht folgendermaßen aus: COMMAND ATTRIBUTE value {BOOLEAN-OPERATOR ATTRIBUTE value}.

"FIND PA darwin AND TW species OR TW descent" ist ein konkretes Beispiel in dem nach dem Personal Author (PA) Darwin und nach den Title Word (TW) Species oder Descent gesucht wird. Der Befehl FIND am Anfang der Suchanfrage drückt aus, dass nach bibliothekarischen Einträgen gesucht werden soll. Der Nachteile einer Kommandosprache sind, die Befehle für die Suche sowie die Attribute und deren Bedeutung zu vergessen. Des Weiteren verwenden auch unterschiedliche Suchmaschinen, oder in diesem Fall, Bibliothekssysteme verschiedene Kommando und Syntaxsprachen. Diese Kommandosprachen wurden auf die Anforderungen des Systems, nicht auf die des Nutzers angepasst und sind somit wenig flexibel.[19] [31]

Heute sind solche Befehlssprachen bei den gängigsten Suchmaschinen nicht mehr sehr gebräuchlich und wurden zum Teil durch s.g. Shortcuts ersetzt. Die Suchmaschine Google bietet z.B. bei der Eingabe von „Wetter Bamberg" das aktuelle Wetter inklusive Vorhersagen für die nächsten Tage für die Stadt Bamberg, in grafischer Form. Eine Währungsumrechnung lässt sich einfach mit einer Eingabe „150 USD in EUR" mit dem aktuellen Wechselkurs berechnen. Die Suchmaschine DuckDuckGo bietet mit den Befehlen „b:", „t:" und „f:" dem Nutzer die Möglichkeit explizit im Body einer Internetseite, im Titel einer Internetseite oder nach bestimmten Dateitypen, wie z.B. PDFs zu suchen. Die Suchmaschine Wolfram Alpha ist vor allem im mathematisch naturwissenschaftlichen Bereich führend. Eine Funktion wie „2x^3+4x^2-8x-5" wird sowohl grafisch aufgezeichnet, alternative Formen abgeleitet, die Nullstellen berechnet, das Minimum und Maximum der Funktion berechnet und viele weitere Operationen durchgeführt.

8 Outlook

Die Spezifikation von Suchanfragen wird weiterhin ein wichtiges Thema im Information Retrieval bleiben, dass ständig optimiert werden muss. Eine Suchanfrage stellt die Brücke zwischen dem Informationsbedürfnis des Nutzers und den Informationszugangssystemen dar. Für den Nutzer einer Suchmaschine muss eine Suchanfrage möglichst einfach zu formulieren sein, trotzdem aber Ergebnisse zurückliefern, die der Nutzer in Erfahrung bringen wollte. Für die Verbesserung der Ergebnisse müssen mehr explizite Informationen mitgegeben werden, um Präferenzen und Interessen des Nutzers besser deuten zu können, diese Daten könnten aus zurückliegenden Abfragen des Nutzers gewonnen werden. Durch die Ortsangabe des Nutzers kann die Suchmaschine den Nutzer bei der Formulierung seiner Suchanfragen zusätzlich unterstützen, indem die Suche bei bestimmten Themengebieten auf den geographischen Umkreis

[19] http://www.humc.edu/library/MELVYL/CHAP1.TXT, aufgerufen am 24.08.2012

des Nutzers eingeschränkt wird. Vor allem im mobilen Bereich von Suchanfragen müssen sich Suchmaschinen konzentrieren, der in den nächsten Jahren nochmals deutlich zunehmen wird und andere Anforderungen an den Nutzer sowie den Suchmaschinen stellen. Diese Anforderungen können neben der sprachlichen Eingabe einer Suchanfrage, auch eine Unterstützung bei der Eingabe über das mobile Gerät sein. Des Weiteren kann von einer Zunahme von Suchanfragen mit natürlicher Sprache ausgegangen werden, sowie von Fragestellungen in ganzen Sätzen. Eine dauerhafte Interaktion mit einer Suchmaschine, die nicht nur übliche Vorschlagslisten und auf optionale Suchbegriffe aufmerksam macht bei einer Suchanfragen, ist ebenso vorstellbar.

References

1. N.J. Belkin, R.N. Oddy, H.M. Brooks: Ask for information retrieval: Part I. Background and theory, Journal of Documentation, Vol. 38 Iss: 2, 1982, pp. 62
2. Picot, A.; Reichwald, R.; Wigand, R.T. (2001): Die grenzenlose Unternehmung. Information, Organisation und Management. 4. Aufl., Gabler, Wiesbaden. Seite 81
3. A. Henrich: Information Retrieval 1, Grundlagen, Modelle und Anwendungen, Version: 1.2 (Rev: 5727, Stand: 7. Januar 2008), Otto-Friedrich-Universität Bamberg
4. C.L. Borgman: Why are online catalogs still hard to use? Journal of the American Society for Information Science, 47(7), 1996, pp. 493–503
5. S.A. Cousins: In their own words: an examination of catalogue users' subject queries, Journal of Information Science, 18(5), 1992, pp. 329
6. A. Aula, N. Jhaveri, M. Käki: Information search and re-access strategies of experienced web users. In: Proceedings of the 14th International Conference on World Wide Web, ACM, New York, 2005, pp. 583–592
7. S. Chakrabarti: Mining the web: Discovering knowledge from hypertext data. Amsterdam (u.a.): Morgan Kaufmann, 2003
8. J. Muramatsu, W. Pratt: Transparent Queries: Investigation Users' Mental Models Of Search Engines. Proceedings of the 24th Annual International ACM SIGIR Conference on Research and development in information retrieval (SIGIR'01), 2001, pp. 217–224
9. T. Lewandowski: Linguistisches Wörterbuch. 4., neu bearbeitete Aufl. Quelle & Meyer, Heidelberg, Stichwort: natürliche Sprache, 1985
10. A. Pollock, A. Hockley: What's Wrong with Internet Searching. D-Lib Magazine, www.dlib.org, 1997
11. D. Bilal: Children's use of the Yahooligans! Web search engine. I. Cognitive, physical, and affective behaviors on fact-based search tasks. Journal of the American Society for Information Science, 51(7):646–665, 2000
12. J. Schacter, G.K.W.K. Chung, A. Dorr: Children's internet searching on complex problems: Performance and process analyses. Journal of the American Society for Information Science, 49(9):840–849, 1998

13. A. Broder: A taxonomy of web search. SIGIR Forum 36(2). http://www.acm.org/sigir/forum/F2002/broder.pdf, 2002. aufgerufen am 21.08.2012
14. D. Booth, B. Jansen, A. Spink: Determining the informational, navigational and transactional intent of web queries. Information Processing and Management, 2008, 44(3), pp. 1251-1266
15. D. Lewandowski: Web Information Retrieval, Technologien zur Informationssuche im Internet, DGI Schrift Informationswissenschaft 7, Frankfurt am Main, 2005
16. W.B. Frakes: Stemming Algorithms. In: Frakes, W. B.; Baeza-Yates, R. (eds.): Information Retrieval: Data Structures & Algorithms. Prentice Hall, New Jersey, 1992, Seite 131-160
17. W. Stock: Information Retrieval, Informationen suchen und finden, Oldenbourg Wissenschaftsverlag GmbH, München, 2007, Seite 201-204
18. S. Chakrabarti: Mining the Web: Discovering Knowledge from Hypertext Data. Amsterdam (u.a.): Morgan Kaufmann, 2003
19. H. P. Luhn: The Automatic Creation of Literature Abstracts. IBM Journal of Research and Development 2(2), 1958, pp. 159-165
20. K. Franzen, J. Karlgren: Verbosity and interface design. Technical report, Technical Report T2000, 2000
21. N.J. Belkin, D. Kelly, G. Kim, J.Y. Kim, H.J. Lee, G. Muresan, M.C. Tang, X.J. Yuan, C. Cool: Query length in interactive information retrieval. Proceedings of the 26th Annual International ACM SIGIR Conference on Research and development in information retrieval (SIGIR'03), 2003, pp. 205–212
22. S.A. Cousins: In their own words: an examination of catalogue users' subject queries. Journal of Information Science, 18(5):329, 1992
23. M.A. Hearst: Improving full-text precision using simple query constraints. In Proceedings of the Fifth Annual Symposium on Document Analysis and Information Retrieval (SDAIR'96), Las Vegas, NV, 1996
24. C.L.A. Clarke, G.V. Cormack, F.J. Burkowski: Shortest substring ranking (multitext experiments for TREC-4), In Donna Harman, editor, Proceedings of the Fifth Text Retrieval Conference (TREC-5), 1996
25. T. Tao, C. Zhai: An exploration of proximity measures in information retrieval. In Proceedings of the 30th Annual International ACM SIGIR Conference on Research and development in information retrieval (SIGIR'07), ACM Press, 2007, pp. 295–302
26. C.T. Meadow, B.A. Cerny, C.L. Borgman, D.O. Case: Online access to knowledge: System design. Journal of the American Society for Information Science, 40(2):86–98, 1989
27. A. Spink, D. Wolfram, B.J. Jansen, T. Saracevik: Searching the web: the public and their queries. Journal of the American Society for Information Science 53(2):pp. 226-234, 2001
28. B.J. Jansen, A. Spink, S. Koshman: Web searcher interaction with the Dogpile.com metasearch engine. Journal of the American Society for Information Science and Technology, 58(5):744–755, 2007
29. M. Machill, C. Neuberger, W. Schweiger, W. Wirth: Wegweiser im Netz: Qualität und Nutzung von Suchmaschinen. In: Machill, M.; Welp, C. (Hrsg.): Wegweiser im Netz: Qualität und Nutzung von Suchmaschinen. Gütersloh: Verlag Bertelsmann Stiftung, 2003, Seite 289-297
30. A. Spink, B.J. Jansen: Web Search: Public Searching of the Web. Dordrecht, Kluwer Academic Publishers, 2004
31. C. Lynch: The next generation of public access information retrieval systems for research libraries – lessons from 10 years of the melvyl system. Information Technology and Libraries, 11(4):405–415, 1992

Popular Destinations in Kombination mit anderen Methoden der Unterstützung von Query Reformulation

Felix Wiedemann

Abstract. Query Reformulation ist ein wichtiger Bestandteil des Prozesses des Information Retrieval. Eine Art der Unterstützung von Query Reformulation stellt die Anzeige von Popular Destinations dar, welche nach einer an das IR-System getätigten Anfrage Endpunkte von ähnlichen Suchvorgängen anzeigen. Dies kann Suchvorgänge effektiver und effizienter machen. Die positive Wirkung der Bereitstellung von Popular Destinations kann durch die Kombination mit anderen Arten der Query Reformulation-Unterstützung wie Rechtschreibhilfen, automatischen Termvorschlägen, implizitem Relevance Feedback und der Anzeige verwandter Artikel noch verbessert werden.

Keywords: query reformulation, popular destination, information retrieval, query destination, session destination.

1 Einleitung

Normalerweise durchläuft der Benutzer eines Information Retrieval Systems folgenden Vorgang iterativ:

Der Sucher, welcher ein Informationsbedürfnis zu befriedigen sucht, entscheidet, ob er die von ihm gewünschten Informationen auf einer ihm bekannten Webseite finden kann, oder ob er die Hilfe eines IR-Systems in Anspruch nehmen muss. Ist letzteres der Fall, so stellt er eine Anfrage an ein IR-System. Anschließend wägt er ab, ob die ihm präsentierten Suchtreffer dieses Bedürfnis hinreichend erfüllen. Trifft dies zu, so wird die Suche beendet. Andernfalls kommt es entweder zu einer Abänderung oder Neuformulierung der Suchterme, was sowohl manuell, als auch maschinell unterstützt erfolgen kann, oder aber zu einem erfolglosen Abbruch des Suchvorgangs. Die genannte Abänderung und Neuformulierung der Suchterme wird wiederholt und das Ergebnis idealerweise dadurch iterativ verbessert, bis der Sucher die gewünschten Informationen gefunden hat oder aber aufgibt. [1] (siehe Abbildung 1)

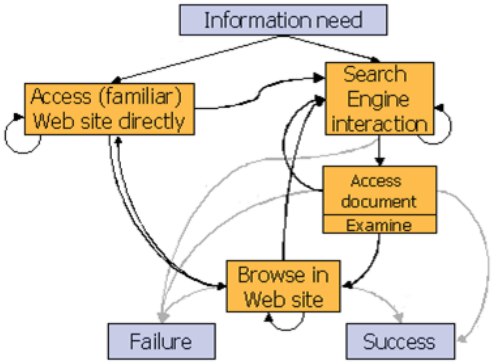


Abbildung 1: Übersicht über den Information Retrieval Prozess Prozess. [Abbildung im Original mit Prozentangaben] [1]

Dieser Prozess kann durch die Hypothese des *Anomalous State of Knowledge* erklärt werden: Der Benutzer hat Schwierigkeiten, sein Informationsbedürfnis auszudrücken. Daher verläuft die initiale Anfrage oft nicht erfolgreich. Allerdings erfasst er meist intuitiv sehr gut, welche der ihm angebotenen Treffer der Ergebnisliste für ihn relevant sind und wägt auf dieser Grundlage ab, ob eine Neuformulierung der Suche notwendig ist oder nicht. [2]

Ein Ziel von IR-Systemen sollte es sein, den Benutzer bei der Reformulierung zu unterstützen. Einerseits sollte dabei versucht werden, die Formulierungsschwierigkeiten des Benutzers durch Korrekturen und Vorschläge auszugleichen. Außerdem sollte sich ein IR-System die Fähigkeit des Nutzers zur Bewertung von Suchtreffern zu Nutzen machen. Finales Ziel muss eine effektivere und effizientere Behebung des Informationsdefizites sein.

Die Reformulierung der Suchbegriffe ist quantitativ bedeutsam: In einer Studie über die Websuchmaschine AltaVista wurde die Ursprungsanfrage von mehr als 50% aller Sucher abgeändert und beinahe ein Drittel tätigte drei oder mehr Anfragen [3]. Daher ist eine wichtige Aufgabe bei der Gestaltung von IR-Systemen die Bereitstellung von den Benutzern bei der Reformulierung von Anfragen unterstützenden Mechanismen. Einer davon ist die Fokussierung auf *query destinations.* Dieser Ansatz soll im Folgenden vorgestellt werden und anschließend anderen Methoden gegenübergestellt werden.

2 Funktionsweise der Benutzung von popular destinations zur Erweiterung von Suchergebnissen und damit verbundene Problemfelder

Wie können beliebte *destinations* zur Unterstützung des Benutzers eines IR-Systems genutzt werden? Hier müssen zuerst die Begriffe der *session destinations* und *query destinations* geklärt werden.

Session destinations bezeichnen die Domains, welche am der Ende der „Such-Sitzung“ besucht werden. Das Ende eines *session trail* wird laut White et. al. erreicht, wenn der Benutzer zu seiner Homepage zurückkehrt, seinen E-Mail-Posteingang überprüft, sich in einen Online-Dienst einloggt, eine Seite länger als 30 Minuten ohne Aktivität betrachtet oder das aktive Browser-Fenster schließt. [4]

Mit *query destinations* meinen White et al. die Domains, welche am Ende der aktiven Suchvorgänge oder ähnlicher Suchvorgänge stehen. Ein *query trail* endet bei Erreichen eines der Indikatoren des Endes eines *session trail* und zusätzlich dann, wenn eine neue Suchanfrage gestartet wird. [4]

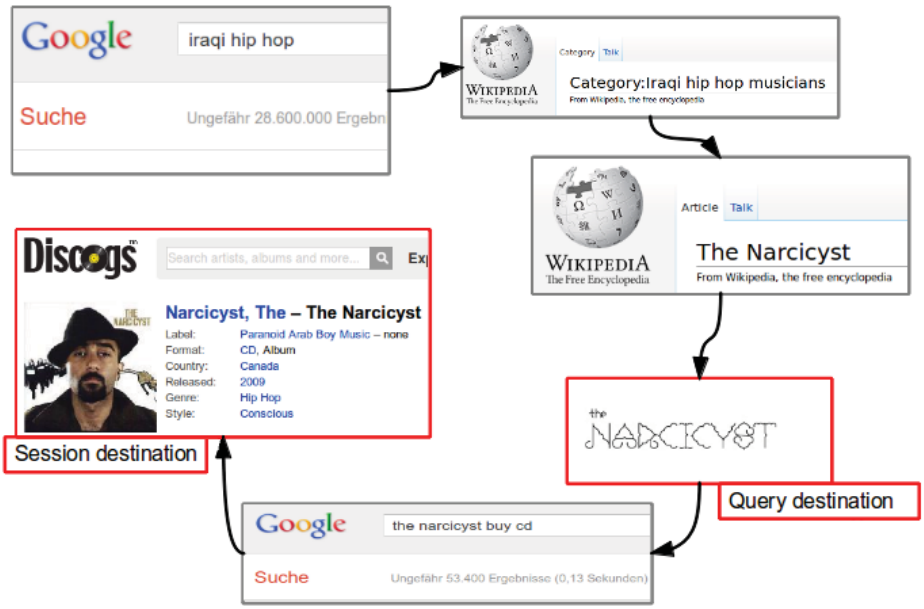


Abbildung 2: Beispiel einer Such-Sitzung und der darin enthaltenen destinations. Quelle: eigene Grafik erstellt aus Screenshots von am 25.06.2012 abgerufenen Webseiten

Abbildung 2 verdeutlicht, was unter *query destinations* und *session destinations* gemeint ist. Die initiale in Google getätigte Suche nach „iraqi hip hop“ führt den Suchenden über die sich mit dem Thema befassende wikipedia-Seite auf die dort verlinkte Seite des Künstlers The Narcicyst. Anschließend klickt der Benutzer auf einen Link zu der persönlichen Homepage des Rappers.[1] Dies stellt den Endpunkt des *query trails* und damit die *query destination* dar. Beim Anschauen der Seite stellt der Benutzer fest, dass er den Künstler durch den Kauf einer CD unterstützen will. Deswegen startet er eine neue Websuche und erhält unter anderem eine Seite als Ergebnis, auf der es ihm möglich ist, eine solche CD zu erwerben.[2] Anschließend bricht er seine Such-Sitzung ab. Aus diesem Grund wird die zuletzt geöffnete Seite als *session destination* registriert.

Wenn *popular destinations*, seien es nun *session destinations* oder *query destinations* zur Unterstützung des Benutzers eingesetzt werden, funktioniert dies folgendermaßen: Der Benutzer tätigt eine Suchanfrage an das IR-System. Das IR-System ermittelt die *destinations* ähnlicher Suchanfragen durch Vergleich der Anfrageterme zum Beispiel durch Summe der Termgewichte, welche das Produkte von *term frequency* und *inverse document frequency* darstellen. Die ermittelten *destinations* werden anschließend neben den regulären Suchtreffern angezeigt.

Ein Ziel dieses Verfahrens ist es, dem Benutzer den „Umweg” über die Seiten zu ersparen, die er sich im Laufe des Suchprozesses ansieht, bis er am Ende der *session* oder der *query* bei einer *destination* anlangt. Man erhofft sich dadurch einen Effizienzzugewinn. Außerdem werden dem Benutzer eventuell neue Ansätze zur Resolution seines Informationsbedürfnisses angeboten, da ihm Seiten angeboten werden, die er vielleicht ansonsten nicht gefunden hätte. Dies kann zu einer Erhöhung der Effektivität des IR-Systems führen.

Manche Websuchmaschinen bieten bereits für einige Suchanfragen Funktionalitäten an, die der von *popular destinations* ähneln. So wird zum Beispiel bei einer Suchanfrage nach „bamberg wetter“ von Google noch vor der eigentlichen Trefferliste eine Visualisierung der Wetteraussichten angezeigt. Dies erspart dem Benutzer den Umweg über die Webseiten von Wetterdiensten oder ähnlichem. (siehe Abbildung 3)

1 http://www.iraqisthebomb.com/

2 http://www.discogs.com/Narcicyst-The-Narcicyst/release/2076453

Abbildung 3: Wetter-Anfrage in Google als Beispiel für in der Praxis bestehende *popular destinations*-Implementierung (Webseite abgerufen am 04.07.2012)

Ein weiteres Beispiel für in der Praxis bestehende Umsetzungen des *popular destinations*-Ansatzes ist die Anzeige von Fußballergebnissen, wie sie während der Fußball-Europameisterschaft 2012 bei einer entsprechenden Suchanfrage zu sehen war. Hier wurde angenommen, dass Benutzer oft kein Interesse daran haben, auf die Suchtrefferseiten zu gelangen, sondern dass ihnen eine schlichte Anzeige der Spielergebnisse völlig ausreiche. (siehe Abbildung 4)

Abbildung 4: Fußballergebnisanzeige bei Google als Beispiel für bereits in der Praxis vorhandene *popular destination*. Quelle: Screenshot übernommen aus Schreiner, Alexander: "Search User Interfaces: Presentation of Search Results".

Bevor beliebte *destinations* dem Benutzer als zusätzliches Angebot bereitgestellt werden können, müssen diese identifiziert weden. Dies kann durch Auswertung von großen Mengen an Nutzerdaten geschehen. Insbesondere *query logs* und Aufzeichnungen des Browsens während der Such-*sessions* können hierbei als Grundlage hergenommen werden [4]. White et al. heben die Wichtigkeit des Nutzerverhaltens nach erfolgter Interaktion mit dem IR-System hervor. Die oben erwähnten Indikatoren für Endpunkte von *session trails* und *query trails* müssen durch Auswertung des Nutzerverhaltens erkannt werden und die damit erhaltenen *destinations* aufgezeichnet werden. Hier ergibt sich im Bereich der Websuche ein Praktikabilitätsproblem: Die Aufzeichnung der getätigten Suchanfragen, sowie der benutzten Links kann ohne Probleme getätigt werden. Sobald sich ein Benutzer jedoch in einer Distanz bewegt, die den initialen Link-Klick von der Suchmaschine zu einer weiteren Seite überschreitet, so kann nur mehr beispielsweise durch Browser-Plugins oder *proxy server logs* das Nutzerverhalten notiert werden.

Session und *query trails* enthalten folgende Seiten: Suchtreffer, Suchmaschinenseiten und Seiten, die durch Folgen von Links mit Suchtreffern verbunden sind [4]. Eine Schwierigkeit bei der Aufzeichnung von *trail*-Enden kann durch parallel gestartete Suchanfragen eines Benutzers entstehen, wenn dieser in mehreren *browser tabs* oder -Fenstern sucht. [4]

Ein weiteres Problem, welches sich bei der Ermittlung von *popular destinations* stellen könnte, ist, dass auch oft angesteuerte „Sackgassen“ als *destinations* gespeichert werden könnten. Gibt eine ausreichend große Anzahl an Personen ähnliche oder gleiche Suchen am gleichen Punkt auf, so wird die Domain als *destination* gespeichert. Benutzer, die ähnliche Informationsbedürfnisse haben, und die den Link zu dieser *destination* nutzen, landen in der gleichen „Sackgasse“.

Dazu kommt noch ein Problem welches sich durch *query mission change* ergeben könnte. Werden bei der Bearbeitung einer *query mission*, die sich als eine “sequence of reformulated queries that express a same need, usually an information need” [5] definieren lässt, zusätzliche Erkenntnisse gewonnen, so beeinflussen diese den Suchvorgang und können unter Umständen die *query mission* verändern. Wird nun die *query mission* innerhalb eines *session trails* verändert, so ergeben sich interessante Möglichkeiten: Im Idealfall ergibt sich eine *session destination*, welche für andere Benutzer nützlich ist, da sie ihnen neue Denkansätze zur Lösung ihres Informationsbedürfnisses bietet. Andernfalls können *session destinations*, welche nach einem *mission change* liegen, aber auch abwegig für andere Benutzer erscheinen, wenn der Zusammenhang mit der initialen Suche nicht oder nur schwer erkennbar ist.

2. Popular Destinations im Vergleich mit anderen Query Suggestion-Methoden

3.1 Rechtschreibkorrekturen und -vorschläge

3.1.1 Rechtschreibkorrekturen und -vorschläge in der Websuche

Da ungefähr 10 bis 15% aller Anfragen an Websuchmaschinen Fehler enthalten [6], liegt es nahe, dass Rechtschreibkorrekturen und -vorschlägen eine große Bedeutung zufällt. Rechtschreibhilfen können auf Editierdistanzen wie der Damerau-Levenshtein-Distanz aufbauen, welche die Zahl an Veränderungen der orthographisch falschen Anfragebegriffe zu einem fehlerlos geschriebenen Term darstellen [7]. Das IR-System baut anschließend eine Rangfolge an möglichen Verbesserungen auf, wählt diejenige mit der geringsten Editierdistanz aus und schlägt sie dem Benutzer vor. Falls statt der exakten Orthographie phonetische Ähnlichkeit als Grundlage für die Distanzberechnung verwendet werden soll, kann beispielsweise der Soundex-Code benutzt werden [7]. Im Gegensatz zur in der Textverarbeitung traditionellen Wörterbuch-basierten Korrektur, setzen Websuchmaschinen auf Algorithmen, welche sich auf *query logs* stützen [8].

3.1.2 Popular Destinations in Kombination mit Rechtschreibkorrekturen und -vorschlägen

Der Ansatz, beliebte *destinations* dem Benutzer als Vorschläge zu präsentieren und Rechtschreibkorrekturen sowie -vorschläge stehen sich in keiner Weise im Weg und können sich gut ergänzen. Es ist vermutlich vorteilhaft, Rechtschreibänderungen zuerst ablaufen zu lassen und alle anderen *query reformulations* anschließend zu erbringen. Dadurch kann eine größere Präzision erreicht werden.

Wird beispielsweise nach dem Begriff “Bismark” gesucht, so bietet Google beispielsweise nicht nur Treffer an, welche sich auf die Stadt Bismark in Sachsen-Anhalt beziehen, sondern unter anderem auch Seiten, die sich mit Otto von Bismarck beschäftigen. Diese Korrektur läuft automatische ab. Außerdem wird dem Sucher die Option geboten, den ursprünglichen Suchbegriff durch Klicken auf die korrigierte Variante des Terms durch diesen Term zu ersetzen (siehe Abbildung 5).

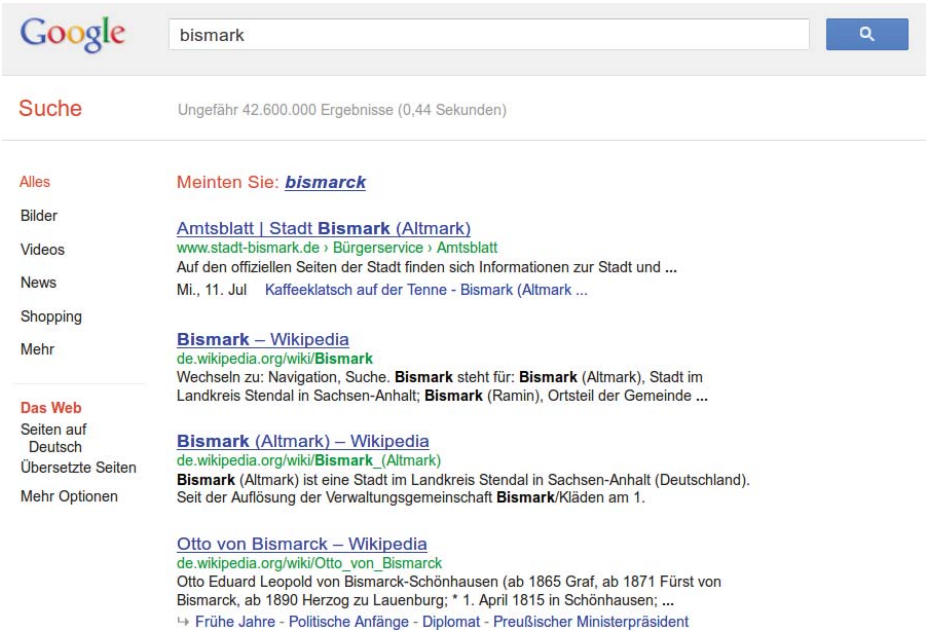


Abbildung 5: Beispiel für automatische und benutzergestützte Rechtschreibkorrektur bei Google (Webseite abgerufen am 25.06.2012)

Würde die Rechtschreibkorrektur nicht sofort erfolgen, so würde das Arbeiten mit *popular destinations* verwirrend für den Benutzer. Suchte er beispielsweise nach Otto von Bismarck und gäbe "Bismark" ein (vergäße also das "c") so würden ihm als *destinations* Seiten angeboten, welche sich in irgendeiner Weise mit der Stadt Bismark beschäftigen.

Bei *query destinations* dürfte dieses Problem schon relativ groß sein, bei *session destinations* lässt sich jedoch erwarten, dass der Bezug der *destination* zum nicht korrigierten Suchbegriff für den Benutzer, der sich seines Fehlers nicht bewusst ist, nicht mehr zu durchschauen ist.

Es ist also von großer Wichtigkeit, Methoden zur Rechtschreibkorrektur und Unterbreitung von Vorschlägen vor Erstellung von Verweisen auf *destinations* anzuwenden.

3.2 Automatische Termvorschläge

3.2 .1 Automatische Termvorschläge in der Websuche

Automatische Vorschläge von Suchtermen können die ursprünglichen Suchbegriffe erweitern, neu gewichten, [1] oder sie ersetzen. Sie sollten dem Benutzer erst mit der Anzeige der ersten Suchergebnisse präsentiert werden und nicht als Zwischenschritt zwischen Suchanfrage und Ergebnisdarstellung. Des Weiteren ist darauf zu achten, dem Benutzer nicht zu viele unterschiedliche Vorschläge zu unterbreiten, um ihn nicht zu überfordern.

Bei einer Eingabe des Suchbegriffs „bismarck“ in die Suchmaschine dogpile stellt diese neben den Suchtreffern eine Liste an Begriffen dar, mit welchen die Anfrage ergänzt werden kann. Diese Terme werden dabei an den ursprünglichen Begriff angehängt. Wird beispielsweise „battleship“ ausgewählt, so erfolgt die anschließende Suche nach „bismarck AND battleship“. (siehe Abbildung 6)

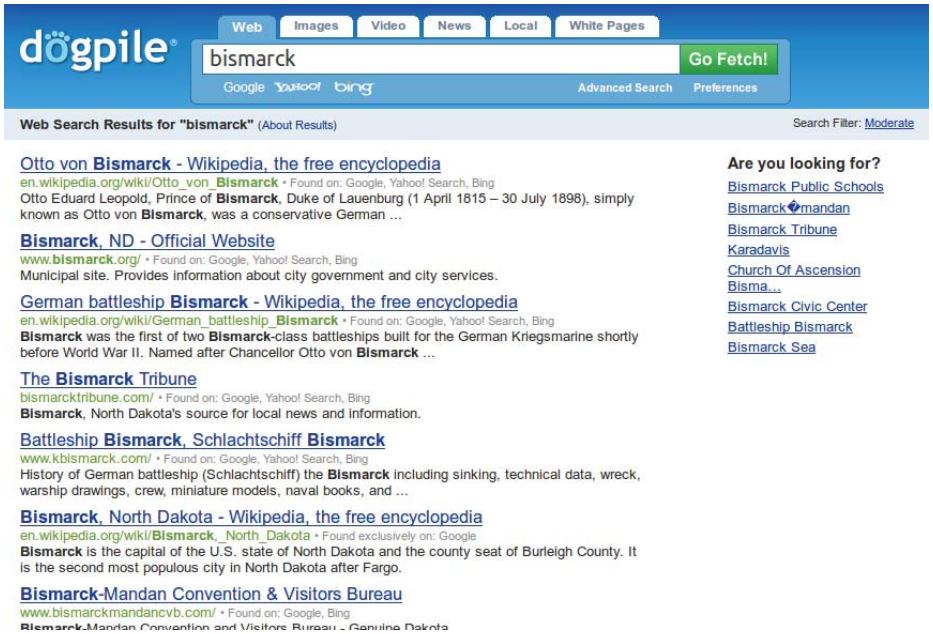


Abbildung 6: Automatische Termvorschläge der Suchmaschine dogpile für den Suchbegriff "bismarck" (Webseite abgerufen am 25.06.2012)

3.2 .2 Query Destinations im Vergleich mit automatischen Termvorschlägen

Eine Studie von White, Bilenko und Cucerzan [4] vergleicht die Bereitstellung von *popular destinations* mit dem Anbieten von *query refinements*:

Vier verschiedene Suchmaschinen wurden miteinander verglichen: Suche ohne Unterstützung (*Baseline*), Suche mit Bereitstellung verwandter Suchterme (*QuerySuggestion*), Suche mit Bereitstellung von *query destinations* (*QueryDestination*) und Suche mit Anbieten von *session destinations* (*SessionDestination*). Diese vier Suchmaschinen wurden auf ihre Tauglichkeit zur Lösung von *exploratory tasks* und *known-item tasks* getestet. *Known-item tasks* sind hierbei *lookup tasks*, welche "discrete and well-structured objects such as numbers, names, short statements, or specific files of text or other media" zurückliefern [9]. Unter *exploratory tasks* versteht man Suchaufgaben, die über reines Nachschlagen von Fakten hinausgehen. Lern- und Rechercheaufgaben, die auf Wissenserwerb, Verständnis und Bewertung von Sachverhalten basieren, sind *exploratory tasks* [9].

Als Ergebnis der Studie lässt sich festhalten, dass *QueryDestination* für *exploratory tasks* bevorzugt wird, während *QuerySuggestion* bei *known-item tasks* die erste Wahl der Nutzer darstellt. Eine stufenweise iterative Verfeinerung der laufenden Suche wie sie *QuerySuggestion* bietet wird bei nur knapper Verfehlung der Zielinformation präferiert. Bei anspruchsvolleren Suchaufgaben werden Vorschläge bevorzugt, die die Richtung einer Suche deutlich verändern können und das gesamte Themenfeld besser abdecken. [4]

Falls nun sowohl *query destinations*, als auch verwandte Suchterme angeboten werden, so müsste darauf geachtet werden, dass die Ergebnisdarstellung sehr umfangreich und eventuell unübersichtlich werden könnte. Dies kann für komplizierte Suchanfragen und erfahrene Benutzer hingenommen werden, ist aber für gewöhnliche Websuche eher kontraproduktiv.

Um eine Überlastung des Benutzers durch zu viele Optionen zu vermeiden, könnte eine Unterscheidung in *exploratory tasks* und *known-item tasks* getroffen werden bevor die Suche durchgeführt wird. Dies kann durch eine manuelle Auswahl des Benutzers geschehen oder aber durch Algorithmen zur Erkennung der Aufgabenart.

Eine manuelle Unterscheidung reduziert die *usability* eines IR-Systems. Gewöhnliche Benutzer kennen weder die Unterschiede zwischen den genannten Aufgabenarten, noch dürfte ihre Bereitschaft dafür, Zeit und Klicks in zusätzliche Entscheidungen zu investieren, sehr hoch sein.

Aus diesen Gründen bleibt nur die maschinelle Unterscheidung zwischen *exploratory tasks* und *known-item tasks*. Als Grundlage könnte hier angenommen werden, dass *known-item tasks* vermutlich meist weniger Reformulierungen benötigen, als *exploratory tasks*. Setzte man nun die Anzeige verwandter Suchterme bis zur x. Reformulierung ein und ersetzte diese ab der x+1. Reformulierung durch die Anzeige von *session destinations*, so könnte man diesem Phänomen Rechnung tragen. Ein Nachteil dieser Vorgehensweise wäre, dass der Effizienzgewinn des *session destination*-Ansatzes gemindert würde. Eine auf anderen Prämissen wie zum Beispiel Suchtermen basierende Unterscheidung müsste untersucht werden. Zu vermuten ist, dass allein durch Suchterme oder durch Kombinationen von Suchtermen noch nicht geschlossen werden kann, ob es sich um eine *known-item task* oder eine *exploratory*

task handelt. Eine Suche nach “VoIP + provider” könnte im Rahmen einer *known-item task* getätigt werden, wenn der Benutzer nach einer Liste von VoIP-Anbietern sucht. Es könnte sich aber ebenso um eine *exploratory task* handeln, wenn das Ziel des Benutzers ist, das für ihn beste Angebot im Bereich des VoIP zu finden.

3.3 Relevance Feedback

Relevance feedback kann in drei verschiedenen Arten stattfinden. Es kann explizit, implizit und als *pseudo relevance feedback* erfolgen. (siehe Abbildung 7) Diese drei Arten von *relevance feedback* sollen nun vorgestellt werden und auf ihre Vereinbarkeit mit der Nutzung von *popular destinations* untersucht werden.

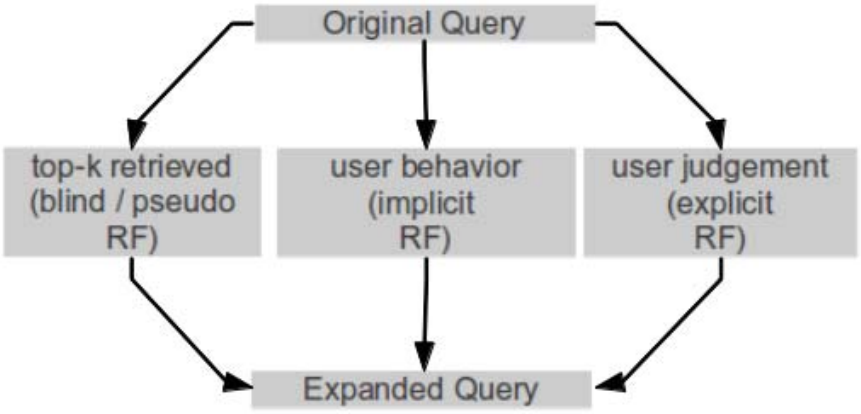


Abbildung 7: Arten des relevance feedback. Quelle: eigene Grafik nach https://www.cs.cmu.edu/~chenminl/images/RelevanceFeedback.bmp (abgerufen am 20.06.2012)

3.3.1 Explizites Relevance Feedback

Explizites *relevance feedback* setzt aktives Mitwirken des Benutzers zur Verbesserung der Suchergebnisse voraus: Nach der initialen Anfrage an das IR-System wird dem Benutzer die Möglichkeit gegeben, die Ergebnisse auf ihre Relevanz hin zu bewerten. Dies kann sowohl binär als reine Unterscheidung zwischen relevanten und irrelevanten Dokumenten geschehen, als auch durch Zuordnung eines Relevanzgrades zu jedem der Treffer. Auf dieser Grundlage verändert das IR-System anschließend zum Beispiel das Ranking der Treffer oder die Suchterme [8]. In einem mit Vektoren arbeitenden Modell würde der die initiale Anfrage repräsentierende Vektor weg von den gefundenen irrelevanten hin zu den gefundenen relevanten

Vektoren verschoben [10]. Dadurch können Anfrageterme eine neue Gewichtung erhalten und neue Terme zur Erweiterung der Anfrage eingeführt werden [1].

Explizites *relevance feedback* ist unter Laborbedingungen sehr gut geeignet, Suchergebnisse zu verbessern. Sowohl hinsichtlich der *precision*, als auch hinsichtlich des *recall* können Steigerungen erreicht werden [1]. Eine Studie, die sich mit der Suchmaschine Excite befasst, widerspricht diesem Befund jedoch und spricht von einer bis zu 37 prozentigen Versagensrate [11]. Explizites *relevance feedback* hat sich jedenfalls in der Praxis nicht durchgesetzt. Dies liegt unter anderem an der schwankenden Qualität der Vorschläge und zum anderen am benötigten Aufwand, der vom Benutzer gefordert wird [8]. Dies sind zwei Gründe, welche explizites *relevance feedback* vermutlich auch in der Zukunft nicht an Bedeutung gewinnen lassen werden.

3.3.2 Implizites Relevance Feedback

Eine andere Art des *relevance feedback* stellt implizites *relevance feedback* dar. Dieses geschieht durch Beobachten des Suchers während des Suchvorgangs. Es können unter anderem Lesezeit, *scroll*-Vorgänge und weitere Interaktionen beim *browsing* durch die Suchergebnisse aufgezeichnet und ausgewertet werden. Laut einer Studie von White, Ruthven und Jose liefere implizites *relevance feedback* besonders gute Ergebnisse bei komplexen Suchaufgaben, Außerdem nähme die Akzeptanz der Vorschläge durch implizites *relevance feedback* offenbar mit Fortschreiten der Such-Sitzung zu. Es muss jedoch auch erwähnt werden, dass die Ergebnisse der Nutzung impliziten *relevance feedbacks* wechselhaft sind hinsichtlich ihrer Effektivität. [12]

3.3.3 Implizites Relevance Feedback

Bei komplexen Suchaufgaben könnte der positive Effekt der Bereitstellung von *destinations* für eben diese Art von Suche durch Miteinbezug von implizitem *relevance feedback* erhöht werden, da beide Arten der Unterstützung bei der Anfragereformulierung hier ihre Stärken haben. Durch das implizite *relevance feedback* könnte es gelingen, speziell für die als relevant erkannten Suchtreffer *destinations* zu ermitteln. Es könnten also als weniger relevant erachtete Startpunkte von *query trails* weniger berücksichtigt werden und relevantere Startpunkte mehr. Somit ergäben sich vermutlich *destinations*, welche für die Sucher schlüssiger wirken.

3.3.4 Pseudo Relevance Feedback

Es soll nun näher auf die Möglichkeiten eingegangen werden, welche *pseudo relevance feedback* bietet. Dieses erfordert keine weiteren Eingaben des Benutzers, da das IR-System selbst annimmt, die ersten Suchtreffer seien relevant. Auf Basis dieser Treffer wird die Suche durch entsprechende Algorithmen angepasst. Laut Hearst hat sich *pseudo relevance feedback* bei manchen Anfragen als erfolgreich erwiesen. Bei anderen, bei welchen die topgerankten Dokumente jedoch irrelevant waren, erwies sich die Methode als unzuverlässig. [8]

3.3.5 Query Destinations in Verbindung mit Pseudo Relevance Feedback

Durch query logs lässt sich feststellen, wie häufig Anfragen auftreten. Bei sehr häufigen Anfragen lässt sich vermuten, dass ein IR-System brauchbare Ergebnisse liefert. Ab einem durch Tests zu ermittelnden Schwellwert könnte man *pseudo relevance feedback* anwenden und der dadurch bereits verbesserten Trefferliste *popular destinations* als Vorschläge beifügen.

Würde man *pseudo relevance feedback* auf jegliche Suchanfragen anwenden, so lässt sich annehmen, dass topgerankte irrelevante Dokumente zuerst zu einer Verschlechterung der Ergebnisliste führen würden. Anschließend würde wiederum diese verschlechterte Ergebnisliste zu noch unbrauchbareren *destinations*-Vorschlägen führen.

3.4 Verwandte Artikel

3.4.2 Anzeige verwandter Artikel in der Websuche

Die Anzeige verwandter Artikel kann als eine Unterart des *relevance feedback* betrachtet werden [13]. Suchmaschinen wie Pubmed[3] des US-amerikanischen National Center for Biotechnology Information bieten die Option, zu einem Suchtreffer, den der Nutzer als relevant kennzeichnet, *related citations* anzeigen zu lassen. Eine Suche nach “plastic surgery” gab an vierter Stelle einen Treffer zu “Burn treatment framework in Israel” aus. Klickte man auf den Link zu *related citations*, so veränderte sich die Trefferliste und es wurden Artikel angezeigt, welche sich beispielsweise mit Kriegsverletzungen, der Gesundheitsversorgung in Gaza und Brandverletzungen beschäftigen (siehe Abbildung 8).

3 http://www.ncbi.nlm.nih.gov/pubmed

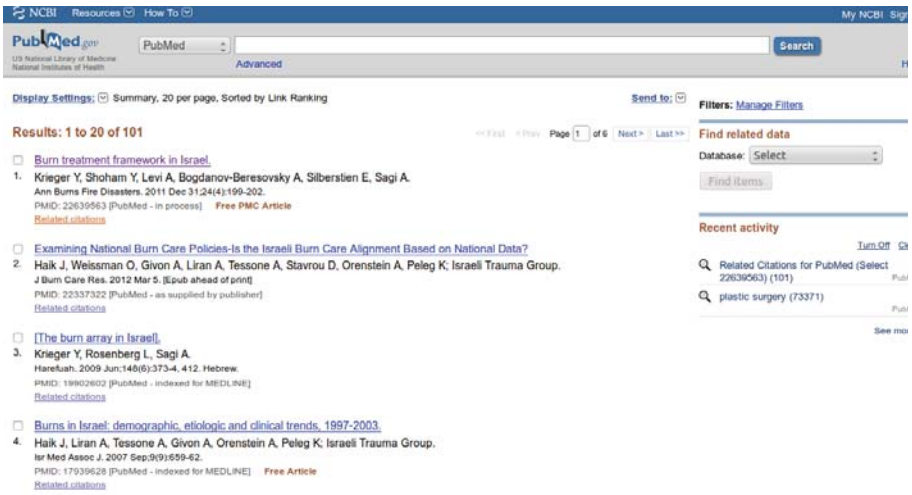


Abbildung 8: Suche in Pubmed nach "plastic surgery" und Auswahl von related citations zu "Burn treatment framework in Israel" (abgerufen am 25.06.2012)

3.4.2 Query Destinations in Kombination mit der Anzeige verwandter Artikel

Gegen eine Kombination von *popular destinations* und *related articles* spricht die kognitive Belastung des Benutzers durch zu viele Optionen nach der Anzeige der Suchtreffer. Außerdem ist der Algorithmus, der zur Erstellung der Liste verwandter Artikel verwendet wird, nicht für den Benutzer ersichtlich und es ist hierdurch nicht ersichtlich, nach welchen Kriterien die Verwandtschaft von Artikeln miteinander begründet wird. [8]

Ein Argument für die Verwendung beider Ansätze ist, dass *popular destinations*-Methoden präziser wirken, wenn der Benutzer vorher einen relevanten Treffer gekennzeichnet hat. Solange die Bedienoberfläche ausreichend übersichtlich gestaltet wird und die Wirkungsweise der einzelnen bereitgestellten Optionen für den Benutzer ausreichend nachvollziehbar ist, dürfte es von Vorteil sein, *popular destinations*-Vorschläge mit *related articles* zu kombinieren.

5. Fazit

Welche Schlussfolgerungen können gezogen werden? Der Ansatz, *popular destinations* für die Reformulierung von Anfragen bereitzustellen ist viel versprechend, da im Idealfall effektivitäs- und effizienzsteigernd, wenn auch noch unausgereift. *Destinations* sind besonders hilfreich bei *exploratory tasks*. Falls *destinations* für diese verwendet werden, so darf dies die Bearbeitung von *known-item tasks* nicht negativ beeinflussen. Von oberster Priorität für die *usability* eines IR-

Systems ist eine übersichtliche und verständliche Benutzeroberfläche, die nicht durch zu viele Möglichkeiten zur *query reformulation* beeinträchtigt werden darf. Falls dieser Grundsatz beachtet wird, gilt, dass die Kombination von *destinations*-Vorschlägen mit anderen *query reformulations* im Allgemeinen gut praktikabel ist. Die Anzeige verwandter Artikel und Rechtschreibhilfen verbessern die Präzision von *destinations*. *Relevance feedback* kann als insbesondere als implizites *relevance feedback* in Verbindung mit *destinations* angewendet werden. Zusätzliche Termvorschläge und *destinations* sind jeweils erfolgversprechend für verschiedene Suchaufgaben und ergänzen sich deswegen gut.

References

1. Eigenstuhler, G., Hubmann, A., Wischounig, D.: Vorlesungsblock 05: Query Reformulation, AI in Information Retrieval (2004), http://www.iicm.tu-graz.ac.at/isr/vo/inhalte/block_05/block05.htm
2. Belkin, N.J., Oddy, R., Brooks, H.: ASK for Information Retrieval. Journal of Documentation. 38(2), 61-71 (1982)
3. Jansen, B., Spink, A., Pedersen, J.: A Temporal Comparison of AltaVista Web Searching. Journal of the American Society for Information Science and Technology. 58(6), 559-570 (2005)
4. White, R. W., Bilenko, M., Cucerzan, S.: Studying the Use of Popular Destinations to Enhance Web Search Interaction. SIGIR 2007 Proceedings. 159-166 (2007)
5. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval: the Concepts and Technology Behind Search. Pearson. Harlow (2011)
6. Cucerzan, S., Brill, E.: Spelling correction as an iterative process that exploits the collective knowledge of web users. Proceedings Of Conference On Empirical Methods In Natural Language Processing (Emnlp'04). S. 293-300 (2004)
7. Henrich, A.: Information Retrieval 1. Modul MI-IR1-M. Skript zur Vorlesung (2011)
8. Hearst, M. A.: Search User Interfaces (2009), http://searchuserinterfaces.com/book/
9. Marchionini, G.: Exploratory Search: From Finding to Understanding. Communications of the ACM. 49(4), 41-46 (2006)
10. Henrich, A.: Information Retrieval: Grundlagen, Modelle und Anwendungen (2008), www.uni-bamberg.de/fileadmin/uni/fakultaeten/wiai_lehrstuehle/medieninformatik/Dateien/Publikationen/2008/henrich-ir1-1.2.pdf
11. Spink, A., Jansen, B.J., Ozmultu, H.C.: Use of Query Reformulation and Relevance Feedback by Excite Users. Internet Research: Electronic Networking Applications and Policy. 10(4), 317-328 (2000)
12. White, R. A., Ruthven, I., Jose, J.M.: A Study of Factors Affecting the Utility of Implicit Relevance Feedback. Proceedings of the Twenty-Eighth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM Press. 35-42 (2005)
13. Spink, A., Wolfram, D., Jansen, Major B.J., Saracevic, Tefko.: Searching the Web: The Public and Their Queries. Journal of the American Society for Information Science and Technology. 52(2), 226-234 (2001)

Information Visualization for Search Interfaces

Dietlinde Flavia Lump

Otto-Friedrich-Universität Bamberg, Lehrstuhl für Medieninformatik
dietlinde-flavia.lump@stud.uni-bamberg.de

Zusammenfassung. Informationsvisualisierung beschäftigt sich allgemein mit der Idee, Informationen über graphische Darstellungsformen so zu visualisieren, dass sie vom Menschen viel intuitiver und schneller aufgenommen werden können. Schließlich finden die Methoden der Informationsvisualisierung langsam aber Sicher auch Eingang in dem Bereich der Suchmaschinen. Diese Arbeit soll einen Einblick geben, auf welchen Ebenen einer Suchoberfläche Methoden der Informationsvisualisierung eingesetzt werden können. Es wird nicht nur auf die damit verbundenen potenziellen Mehrwerte, sondern auch auf die eventuellen Schwierigkeiten näher eingegangen. Dabei werden aktuell online verfügbare, aber auch lediglich in der Literatur beschriebenen Konzepte bzw. Systeme vorgestellt.

Schlagworte: Informationsvisualisierung, Visualisierungsprozess, Referenzmodell, Venn-Diagramm, Block-oriented-Diagramm, Clusterdarstellung, TileBars

1 Motivation und Aufbau der Arbeit

Das Sprichwort: „Ein Bild sagt mehr als tausend Worte“ verdeutlicht die Tatsache, dass die visuelle Wahrnehmung für den Menschen von großer Bedeutung ist. Wieso sollte man also das menschliche visuelle System nicht auch bei der Ausgestaltung von Suchmaschinen ausnutzen?

Ein Mensch kann Informationen durch visuelle Repräsentationen viel schneller und effektiver aufnehmen, als beispielsweise durch eine Aneinanderreihung von Wörtern (z.B. in Form eines Textes) oder Zahlen (z.B. in Form einer Tabelle). Der Mensch würde sicherlich diese schnelle und effektive Art der Informationswahrnehmung auch bei seiner Suche im WWW präferieren. Denn gerade im Bereich von Suchmaschinen ist man einer kontinuierlich wachsenden Informationsmenge und -komplexität ausgesetzt. Hierbei stehen Informationen in verworrenen Beziehungen zueinander und klare Strukturen sind schwer erkennbar.

Eine sinnvolle Informationsvisualisierung könnte den Nutzer bei seiner Suche unterstützen und die Suche an sich beschleunigen und verbessern. Der Einsatz von Techniken der Informationsvisualisierung im Bereich der Suche könnte nicht nur

dazu beitragen spezifische Informationen in großen Datenbeständen zu finden, man könnte außerdem auch Relationen und Strukturen erkennbar machen. Desweiteren wäre es Möglich Informationen im Kontext zu anderen Informationen darzustellen und verschiedene Sichten auf identische Datenbestände aufzuzeigen. Der Einsatz von Informationsvisualisierung im Bereich der Suche könnte also einen erheblichen Vorteil gegenüber traditionellen Suchoberflächen bieten. [Dä99, S.3ff]

Das nächste Kapitel widmet sich einer genaueren Definition des Begriffs „Informationsvisualisierung“ und einigen grundlegenden Gedanken. In Kapitel 3 wird darauf eingegangen, auf welchen Ebenen einer Suchmaschine konkret Techniken der Informationsvisualisierung eingesetzt werden können. Auf Basis dieses Ebenenmodells wird in den Kapiteln 3.1, 3.2 und 3.3 näher auf die einzelnen Visualisierungsebenen eingegangen. Das abschließende 4. Kapitel widmet sich einem kurzen Fazit und einem Ausblick.

2 Grundsätze der Informationsvisualisierung

2.1 Begriffsdefinition

Für den Begriff „Informationsvisualisierung“ gibt es in der Literatur keine einheitliche Definition. Nach Card/ Mackinlay/ Shneidermann versteht man unter „Informationsvisualisierung“ beispielsweise:

> *“The use of computer-supported, interactive, visual representations of abstract data to amplify cognition” [CaMaSh99, S.17]*

Demnach werden bei der Informationsvisualisierung computergestützte, interaktive, visuelle Repräsentationen eingesetzt, um die Erkenntnis zu fördern. In anderen Worten werden bei der Informationsvisualisierung abstrakte Informationen und komplexe Sachverhalte strukturell aufbereitet und grafisch dargestellt, so dass der Anwender Zusammenhänge erkennen kann. Dies ist eine der ersten und prägnantesten Definitionen des Begriffs.

Im Jahr 2008 erweitert Card diese Definition und behauptet:

> *“The purpose of information visualization is to amplify cognitive performance, not just to create interesting pictures. Information visualizations should do for the mind what automobiles do for the feet” [Ca08, S. 539]*

Die Absicht der Informationsvisualisierung besteht also ganz klar darin die kognitive Leistung zu erweitern und nicht lediglich interessante Bilder zu erzeugen. Es geht nicht darum Grafiken zu generieren, es soll vielmehr dem Verstand des Nutzers dieselbe Stütze bieten, wie das Auto den Füßen. Desweiteren soll Informationsvisualisierung, gerade in Anbetracht des wachsenden Informationsvolumens, dem Menschen helfen sich in der Informationsflut zurechtzufinden [Ca08, S. 539].

Die Definitionsansätze des Begriffs stellen ganz klar die menschliche visuelle Wahrnehmung und die in Kapitel 1 genannten Ziele bzw. Absichten der Informationsvisualisierung in den Vordergrund. Außerdem wird betont, dass durch Informationsvisualisierung ganz klar ein Mehrwert geschaffen werden muss.

2.2 Der Visualisierungsprozess

Bei der Informationsvisualisierung versucht man Informationen aufzubereiten und visuell darzustellen, um dem Nutzer eine leicht verständliche visuelle Repräsentation der Information zur Verfügung zu stellen. Wie dieser Visualisierungsprozess erfolgt, wird in der Literatur oft anhand des Referenzmodells erläutert (siehe Abbildung 1).

Im Mittelpunkt dieses Modells steht der Nutzer mit seinen zielgerichteten Aufgaben. Die Zwischenschritte bzw. Transformationsstufen des Modells sind durch die Beeinflussung der Daten in Form der menschlichen Interaktion gekennzeichnet. Insgesamt verläuft der Prozess in drei Transformationsschritten ab: Der erste Schritt, die Datentransformation (Filtering), überführt die Rohdaten beispielsweise über Filtertechniken in eine Datentabelle (diese enthält zum Beispiel relationale Beschreibungen der Daten inklusive Metadaten). Der nächste Schritt, die Visualisierungstransformation (Mapping) transformiert Datentabellen in visuelle Abstraktionen (beispielsweise geometrische Primitive). Der letzte Schritt, die Abbildungstransformation (Rendering), ermittelt aus den visuellen Abstraktionen die letztendliche bildliche Darstellung, indem graphische Parameter, wie beispielsweise Größe oder Positionen spezifiziert werden. Der entscheidende Teil im Modell ist das Mapping der Datentabellen auf die visuellen Abstraktionen, wobei hier eine Überführung aus dem Datenraum in den Bildraum stattfindet und die Wahl einer geeigneten geometrischen Darstellungsform sehr wichtig ist. [CaMaSh99, S.17 und Ch00, S.44]

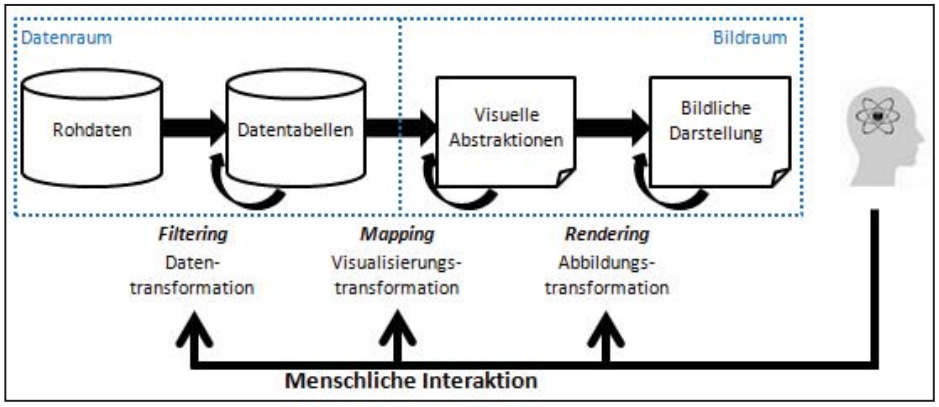


Abbildung 1: Referenzmodell der Informationsvisualisierung [1]

[1] Abgeändert entnommen aus [CaMaSh99, S.17] und [Ch00, S.44]

2.3 Techniken der Informationsvisualisierung

Zu den Techniken der Informationsvisualisierung gehören neben dem Einsatz von Symbolen (icons) und Farbhervorhebungen unter anderem auch das sogenannte „brushing and linking“, „panning and zooming“, „fokus and kontext“ und „overview and detail“.

„Brushing and linking“ beschreibt die Verbindung zwischen mindestens zwei Ansichten auf dieselben Daten. Dabei wirkt sich eine Selektion oder Farbhervorhebung der Repräsentation in einer Ansicht auf die anderen Ansichten aus. [He99]

Die Technik „panning and zooming“ übernimmt die Funktionen einer Filmkamera und kann ein Bild bzw. eine Darstellung sowohl seitlich nach links und nach rechts abtasten (panning), als auch in die Darstellung rein oder raus zoomen (zooming). [He99]

Bei „fokus and kontext“ wird in der Darstellung von Informationen zwischen einem zentralen Fokus und einem peripheren Darstellungsbereich unterschieden. In ein und derselben Ansicht/Darstellung gibt es also Stellen die besonders interessant sind und auf die ein Fokus gesetzt wird. Wobei auf die weniger interessanten Kontextbereiche kein Fokus gesetzt wird und keine Details dargestellt werden. [ScMü04, S.136]

„Overview and detail“ stellt eine Technik dar, die Informationen zum einen in einer detaillierten Teilansicht und zum anderen in einer übersichtlichen Gesamtansicht darstellt. Man hat also zwei Ansichten mit unterschiedlichen Detaillierungsgraden. Ein Beispiel wo diese Technik eingesetzt wird wäre „Google Maps“. [ScMü04, S.136]

3 Ebenen der Informationsvisualisierung

Im Folgenden stellt sich die Frage, auf welchen Ebenen einer Suchmaschine Informationsvisualisierung eingesetzt werden kann. Hierbei habe ich versucht die unterschiedlichen Ansätze der Informationsvisualisierung bei Suchmaschinen anhand eines Ebenenmodells zu systematisieren bzw. zu klassifizieren. (siehe Abbildung 2)

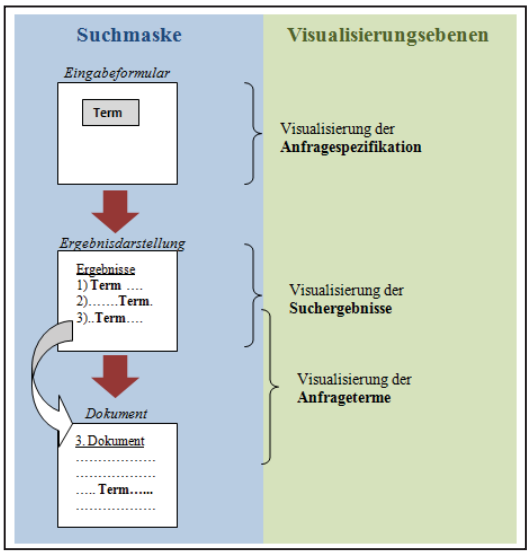


Abbildung 2: Ebenen der Informationsvisualisierung bei Suchmasken[2]

Auf der Seite der Suchmaske wird ein einfacher Suchprozess dargestellt: Eingabeformular => Ergebnisdarstellung => Dokument. Es stellt sich schließlich die Frage, auf welchen Ebenen dieser Suchmaske Informationsvisualisierung eingesetzt werden kann. Die erste Visualisierungsebene korrespondiert mit dem ersten Schritt im Suchprozess, und beschäftigt sich demnach mit der Visualisierung der Anfragespezifikation. Die nächste Visualisierungsebene entspricht dem zweiten Schritt im Suchprozess und betrachtet die Möglichkeiten der Visualisierung von Suchergebnissen. Die dritte Visualisierungsebene kommt bei der Ergebnisdarstellung und im Dokument zum Vorschein und betrachtet die Visualisierung der Anfrageterme. In den nachfolgenden Kapiteln werden die einzelnen Visualisierungsebenen näher erläutert.

3.1 Visualisierung der Anfragespezifikation

Am Anfang jeder Suche steht das Informationsbedürfnis eines Nutzers. Dieses Informationsbedürfnis muss der Nutzer als Anfrage spezifizieren und in der Regel in eine Suchmaske eingeben. Oft kommt es dabei zu Neuformulierungen der Anfrage, die durch textuelle Begriffsvorschläge unterstützt werden können. Problematisch ist generell, dass die gestellte Anfrage oft den eigentlichen Informationsbedarf schlecht abbildet. Wie eine solche Anfragespezifikation durch Visualisierung unterstützt werden kann, wird in den nächsten Unterkapiteln erläutert.

[2] Eigene Darstellung

3.1.1 Visualisierung einer booleschen Anfragespezifikation

Die boolesche Anfragespezifikation beruht auf Anfragesprachen, die auf bool'sche Logik basieren. Aufgrund der Komplexität dieser Art der Anfragespezifikation ist sie für die Mehrzahl der Nutzer nicht geeignet. Denn oft machen unerfahrene Nutzer bei ihrer booleschen Anfragespezifikation auf der einen Seite mehr Fehler und auf der anderen Seite benötigen sie mehr Zeit für ihre Anfrageformulierung [He09, S107ff]. Um den beschriebenen Problemen entgegenzuwirken und Nutzern bei ihrer booleschen Anfragespezifikation zu unterstützen könnte man einige graphische Darstellungsformen einsetzten. Darunter zählen beispielsweise das Venn Diagram Interface und das Block-Oriented Diagram. Diese Visualisierungsformen werden im Folgenden kurz erläutert [He09, S247ff]:

Durch das Venn Diagram Interface kann der Nutzer seine Anfrageterme nach Belieben anordnen und damit boolesche Anfragen generieren. Dabei steht je ein Kreis symbolisch für je einen Anfrageterm und der Nutzer kann recht intuitiv die Kreise in einem so genannten „AnfrageWorkspace" anordnen (siehe Abbildung 3).

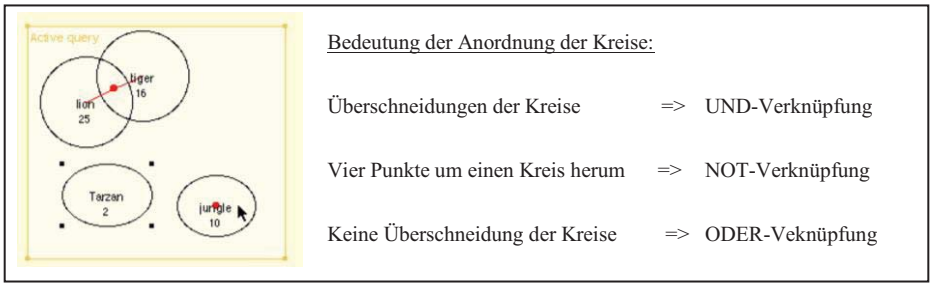


Abbildung 3: Beispielhafte Darstellung einer bool'schen Anfrage bei einem Venn Diagram Interface[3]

In den Kreisen steht neben dem Anfrageterm auch die Anzahl der Dokumente, die diesen Term beinhalten. Neben einem „AnfrageWorspace" steht bei einem Venn Diagram Interface auch ein „Eingabebereich" zur Verfügung, in dem neue Terme eingegeben und die entsprechenden Kollektionen ausgewählt werden können. In der „Anfrageansicht" wird dem Nutzer seine Anfrage in natürliche Sprache angezeigt, sodass er auf diesem Weg prüfen kann, ob seine Anordnung der Kreise im „AnfrageWorkspace" auch seine ursprüngliche Anfrage abbildet. [He09, S247ff]

Bei einem Block-Oriented Diagram gibt der Nutzer seine Anfrage in natürlicher Sprache ein und diese wird automatisch in ein „Block-Oriented Diagram" umgewandelt (siehe Abbildung 4). Hierbei wird jeder Anfrageterm durch ein Block repräsentiert. Diese Blöcke werden in Spalten und Zeilen angeordnet. Bei mehr wie zwei Blöcken pro Zeile werden die Anfrageterme UND verknüpft und bei mehr wie zwei Blöcken pro Spalte werden die Anfrageterme ODER verknüpft. [He09, S247ff]

[3] Entnommen aus [JoMc98]

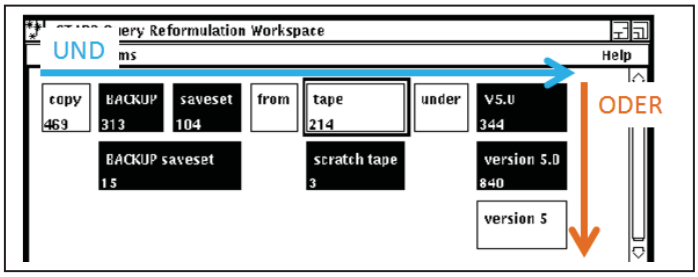


Abbildung 4: Darstellung einer bool'schen Anfrage bei einem Block-Oriented Diagram[4]

3.1.2 Visualisierung zur Unterstützung der Suchanfrageformulierung

Die Suchanfrageformulierung bei Google und Yahoo wird über eine einfache Listendarstellung der Term-Vorschläge unterstützt. Hierbei klappt bei Google schon nach Eingabe eines Buchstaben eine Liste auf, die die Anfrage vervollständigen bzw. verfeinern soll.

Bei Quintura.com werden die Term-Vorschläge einer Anfrage als Wortwolke dargestellt. Hierbei werden Begriffe mit einem thematischen Zusammenhang dementsprechend auch visuell nah beieinander gelegt, obwohl die Anordnung der Begriffe dennoch ein wenig willkürlich erscheint.

Diese Darstellung ist sicherlich für Kinder sehr sinnvoll, da diese durch die Eingabe eines Begriffs auf spielerische Art auf weitere Anfragen und interessante Themengebiete geleitet werden (quinturakids.com). Wenn man aber als Nutzer zielgerichtet nach bestimmten Informationen sucht, ist eine solche willkürliche Begriffswolke nicht sehr sinnvoll, da diese eine längere Sichtung erfordert.

Eine ähnliche Visualisierung zur Unterstützung der Suchanfrageformulierung bietet mnemo.org an. Hier werden die Term-Vorschläge sogar in Synonyme, Tags, benachbarte Begriffe und Übersetzungen klassifiziert. Vor allem zur Ideenfindung ist eine solche Darstellung sehr sinnvoll. So erscheinen beispielsweise Begriffe zu einer Anfrage, an die der Nutzer anfangs überhaupt nicht gedacht hat. Leider ist aber die letztendliche Suche bei mnemo.org aktuell sehr schlecht, da auch bei sehr einfachen Begriffen keine Ergebnisseiten dargestellt werden.

3.2 Visualisierung der Suchergebnisse

Nach Eingabe eines Terms, wird man im nächsten Schritt mit einer sehr großen Trefferzahl (oftmals im Hundertrausenderbereich) konfrontiert. Der Nutzer ist jedoch auch schon bei einer viel kleineren Trefferzahl damit überfordert aus den

[4] Entnommen aus [AnBrFl90]

Ergebnissen, die für ihn relevanten Seiten aufzufinden. Versucht man Techniken der Visualisierung bei der Ergebnisdarstellung einzusetzen, so ist es vor allem wichtig einen Mehrwert dahingegen zu bieten, dass der Nutzer bei seiner Entscheidung über relevante und nicht relevante Dokumente unterstützt wird. [BeHe07, S. 149]

Die nächsten Unterkapitel betrachten die unterschiedlichen Darstellungsformen der Suchergebnisse.

3.2.1 Einfache Darstellung der Suchergebnisse

Die Darstellung der Suchergebnisse bei Google und Yahoo ist eher einfach gehalten. Es kommt keine großartige Visualisierung zum Einsatz, denn es wird in Textform lediglich ein Titel, eine farblich hervorgehobene URL und ein Snippet dargestellt. Der Nutzer muss also Suchergebnis für Suchergebnis durchgehen und den Text der einzelnen Suchergebnisseiten durchlesen, um abschätzen zu können, ob es sich bei der Seite mit dem jeweiligen Titel, dem jeweiligen Snippet und der jeweiligen URL um eine für ihn relevante Seite handelt. Dieser Vorgang scheint auf den ersten Blick sehr aufwändig zu sein, jedoch sind die meisten Nutzer einer Suchmaschine mit genau diesem Vorgang vertraut und können es als störend empfinden, wenn sich diese recht einfache Darstellung in gewisser Weise ändert.

Ein Versuch die Nutzer auch visuell bei ihrer Suche zu unterstützen, bietet die additive Darstellung der Suchergebnisse in Form von kleinen Vorschaubilder (Screenshots/Thumbnails) der einzelnen Suchergebnisseiten. Beispielsweise bietet mister-wong.de genau diese Darstellungsform an, bei der in der Ergebnisliste auch automatisch Screenshots der Ergebnisseiten angezeigt werden. Diese Visualisierung ist vor allem zur Wiedererkennung von Seiten gut geeignet. Eine Beurteilung der Relevanz der jeweiligen Ergebnisseiten durch die Screenshots kann nur bedingt erfolgen, wobei ganz klar die Vorschaubilder lediglich einen visuellen Eindruck vermitteln. [WeBeHi09, S.253 und He09, S.265]

3.2.2 Chronologische Darstellung der Suchergebnisse

Bei der chronologischen Darstellung werden Suchergebnisse nach einem zeitlichen Kriterium geordnet und auf einer eindimensionalen Zeitleiste dargestellt. Eine Suchmaschine, die die Suchergebnisse chronologisch darstellt ist beispielsweise OneTimeLine. Schaut man sich die Ergebnisseite von OneTimeLine genauer an (siehe Abbildung 5), so erkennt man, dass die Darstellung einer großen Ergebnismenge auf einer horizontalen Zeitachse schnell unübersichtlich werden kann. Daher ist eine solche Darstellungsform wirklich nur dann geeignet, wenn eine chronologische Reihenfolge der Suchergebnisse vom Nutzer selbst gewünscht wird. Außerdem ist es bei vielen Dokumenten schwierig korrekte Zeitbezüge herzustellen, da nur die wenigsten Webseiten aussagekräftige Metadaten haben und somit semantische Inhalte sehr schwer erkennbar sind. [WeBeHi09, S.267]

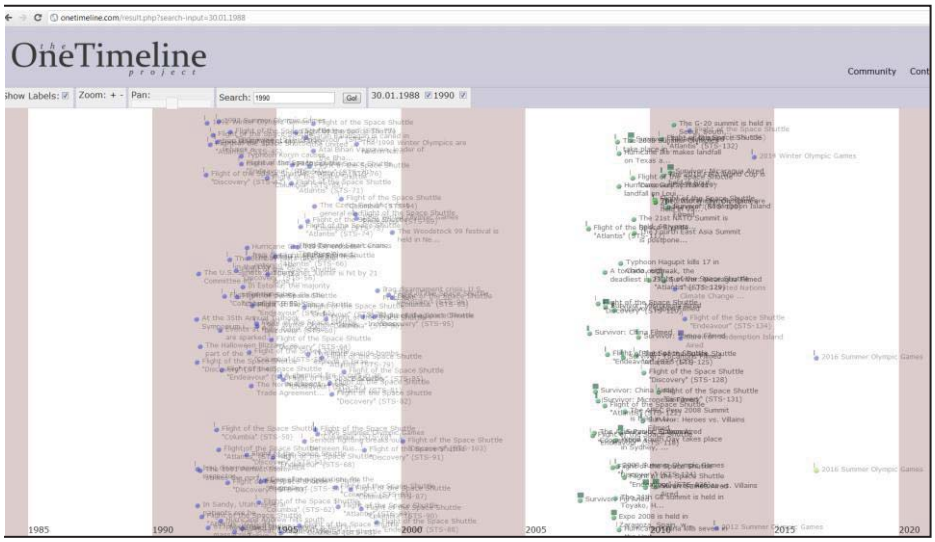


Abbildung 5: Ergebnisdarstellung bei OneTimeLine auf einer horizontalen Zeitachse[5]

3.2.3 Geografische Darstellung der Suchergebnisse

Eine geografische Darstellung der Suchergebnisse sieht es vor, die Suchergebnisse auf einer Karte zu visualisieren. Jedoch hängt der Mehrwert einer solchen geografischen Darstellung stark vom Kontext und der Anwendung ab [WeBeHi09, S.269]. Wenn man beispielsweise nach Dokumenten zu dem Thema „Informationsvisualisierung“ suchen will, so ist der geografische Bezug der einzelnen Ergebnisseiten für den Nutzer in der Regel uninteressant. Sucht der Nutzer jedoch nach „Museen“, so will er vermutlich auch wissen wo sich diese Museen genau befinden. Genau diese Branchensuche wird bekanntermaßen von Google Maps unterstützt. So wird die Ergebnismenge bei Google Maps auf der einen Seite als Ergebnisliste und auf der anderen Seite, repräsentiert durch Icons, auf einer geografischen Landkarte dargestellt.

3.2.4 Darstellung der Suchergebnisse in Cluster

Will man Suchergebnisse in Cluster darstellen, so muss man die Ergebnismenge in unterschiedliche Kategorien unterteilen. So hat der Nutzer einen guten Überblick über die verschiedenen Themenbereiche und kann, die für ihn relevanten Trefferbereiche selektieren. [BeHe07, S: 152 und WeBeHi09, S.256]

[5] Screenshot von `www.onetimeline.com` besucht: Juni 2012

Die Clusterdarstellung kann zum einen in textbasierter Listenform erfolgen (siehe clusty.com oder iboogie.com). Zum anderen gibt es auch die visuelle Variante, bei der die Cluster beispielsweise als Kreise und die Dokumente als kleine Bilder oder Symbole dargestellt werden. Bei der visuellen Darstellung ist die räumliche Nachbarschaft der Symbole oft ein Indiz für die thematische Ähnlichkeit der Dokumente. [He09, S.273] Suchmaschinen, die Clustervisualisierung nutzen, sind beispielsweise Grokker, Ujiko oder meX-Search.

Bei Grokker findet man neben der klassischen Listendarstellung der Suchergebnisse auch eine Unterteilung der Ergebnismenge in Cluster vor. Die Cluster/Kategorien werden durch größere Kreise, die Unterkategorien durch kleinere Kugeln/Kreise und die Dokumente durch Rechtecke mit abgeknickter Ecke repräsentiert. Außerdem kann man sich die Ergebnismenge genauer anschauen oder auch Vorabinformationen einblenden lassen, indem man die Zooming- und Mouse-Over-Funktionen nutzt. Als schwierig erweist sich bei der Clustervisualisierung die Bestimmung der Clusterbezeichnungen. Bei Grokker erfolgt diese Bestimmung über das „Latent Semantic Indexing“, dass die zentralen Konzepte der Dokumente herausfiltert und daraus die Clusterbezeichnungen bestimmt. [WeBeHi09, S.257]

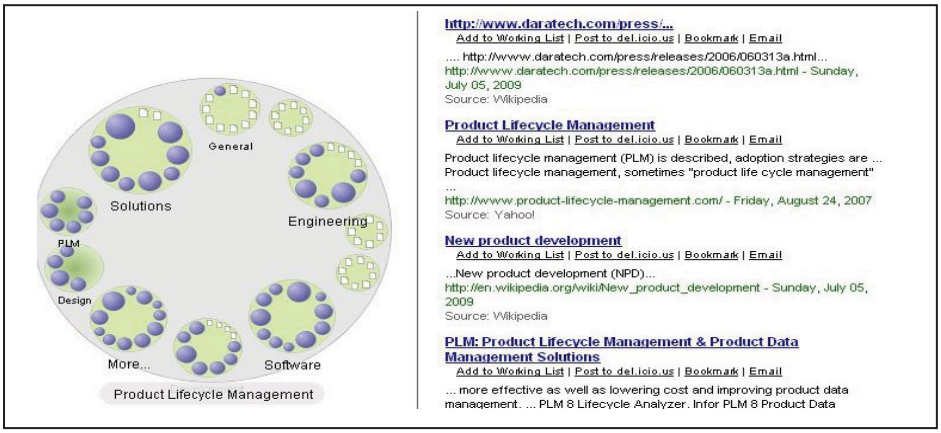


Abbildung 6: Clusterdarstellung bei Grokker zur Suchanfrage „Product Lifecycle Management“[6]

Die Metasuchmaschine MeX-Search bietet eine ähnliche Darstellung der Suchergebnisse wie Grokker. Dabei werden beispielsweise die Ergebnisse von Yahoo in thematische Cluster unterteilt, diese Cluster werden als Kreise dargestellt und kreisförmig angeordnet. In einem solchen Kreis befinden sich die Clusterbezeichnung und die zu dem Cluster zugehörigen Dokumente, die wiederum durch kreisförmige Linksymbole repräsentiert werden. Die Größe dieser Kreise repräsentiert die Größe

[6] Entnommen aus `http://perspectives.3ds.com/wp-content/uploads/grokker.jpg` besucht: Juni 2012

der Cluster und somit die Anzahl der dahinter stehenden Dokumente. Auch hier können über die Mouse-Over-Funktion Hintergrundinformationen eingeblendet werden. Abbildung 7 bildet beispielsweise das Cluster mit der Clusterbezeichnung „Information Visualization“ ab, welches 10 Dokumente beinhaltet. [WeBeHi09, S.257]

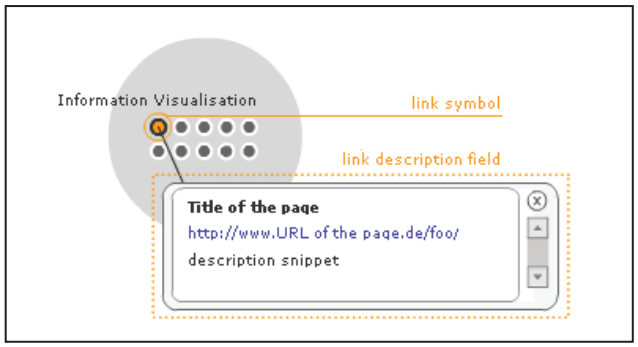


Abbildung 7: Darstellung eines Clusters bei meX-Search[7]

Eine, im Vergleich zu Grokker und meX-Search, völlig andere Clustervisualisierung nutzt die Suchmaschine Ujiko. Zur Unterscheidung der Cluster verwendet die Suchmaschine keine Kreise sondern ausschließlich Farbcodierungen. Auf einer Ergebnisseite werden maximal 12 Dokumente kreisförmig dargestellt. In der Mitte der Darstellung befinden sich die Clusterbezeichnungen. Mittels einer Circular Map wird die Zugehörigkeit eines Dokuments zu einem Cluster über die Farben der einzelnen Clusterbezeichnungen dargestellt (siehe Abbildung 8). [WeBeHi09, S.258-259]

[7] Entnommen aus `http://www.mex-search.com/pop.php?cat=help` besucht: Juli 2012

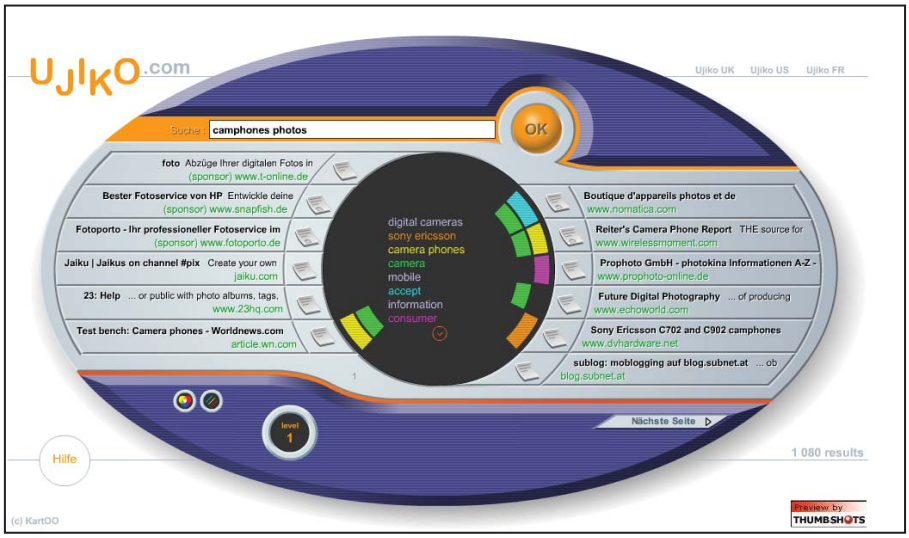


Abbildung 8: Farbcodierte Clusterdarstellung bei Ujiko[8]

Das große Problem bei allen genannten visuellen Suchmaschinen ist, dass sie leider entweder nicht mehr online verfügbar sind (Grokker.com, Ujiko.com) oder die Suche an sich nicht mehr funktioniert (meX-Search.com). Dahingegen ist die Clusterdarstellung in textbasierter Listenform viel weiter verbreitet und die aktuell verfügbaren Seiten wie beispielsweise clusty.com oder iboogie.com sind zumindest nutzbar.

Warum es keine aktuell brauchbare Websuchmaschine mit visuell dargestellten Cluster gibt, ist fraglich. Vielleicht stehen viele Nutzer den visuellen Repräsentationen der Cluster kritisch gegenüber, weil eine kurze Einarbeitung bzw. Erklärung der Symbole notwendig ist, um sich beim Sichten der Suchergebnisse zurecht zu finden. Demgegenüber kommen Nutzer sofort ohne große Einarbeitungszeit mit der textbasierten Listenform klar und finden sich schätzungsweise relativ schnell zurecht.

Ob man nun die Cluster in textbasierter Listenform präferiert oder doch eher eine visuelle Darstellungsform bevorzugt ist sehr subjektiv. Wichtig ist dass die Darstellung der Ergebnisseiten in Cluster einen großen Mehrwert bringen kann. Denn die Clustervisualisierung im allgemeinen kann dem Nutzer nicht nur einen guten Überblick über die Ergebnismenge bieten, die inhaltliche Gruppierung in Cluster stellt dem Nutzer auch wichtige Kontextinformationen zur Verfügung (Bei „Maus“ => Das Tier oder die Computermaus?) [WeBeHi09,S.259 und He09, S.273].

[8] Entnommen aus `http://lhivic.org/atelier/wp-content/ujiko-2.png` besucht: August 2012

3.2.5 Darstellung von Relationen zwischen Dokumenten

Suchergebnisse werden in diesem Fall nicht nur in Cluster unterteilt, es werden auch Beziehungen zueinander dargestellt. Dabei ist das Ziel Zusammenhänge zwischen Dokumenten sichtbar zu machen. Diese Zusammenhänge können aus sehr unterschiedlichen Merkmalen resultieren, beispielsweise aus begrifflichen Ähnlichkeiten oder aus geografischen oder thematischen Zusammenhängen. Problematisch ist jedoch, dass das Erkennen von Relationen eine semantisch-lexikalische Analyse der Inhalte voraussetzt. [WeBeHi09, S.260 und BeHe07, S.154]

Ein großer Vorteil einer solchen Darstellungsform ist, dass nicht sofort erkennbare Zusammenhänge zwischen Dokumenten, sichtbar werden und auf diese Weise auch eine Ähnlichkeitssuche erleichtert wird. So kann der Nutzer nicht nur viel leichter vergleichbare Dokumente finden, er kann auch seine Suchanfrage viel besser bzw. „kreativer“ anpassen, sodass diese letztendlich sein Informationsbedürfnis deckt. [WeBeHi09, S.262]

Die Suchmaschinen Kartoo und Tianamo versuchten diese Darstellungsform umzusetzen, sind aber aktuell nicht online verfügbar. [WeBeHi09, S.260]

3.2.6 Darstellung von Ergebnismengen mehrerer Suchmaschinen

Bei dieser Darstellungsform versucht man die Treffermengen unterschiedlicher Suchmaschinen gegenüberzustellen. Man betrachtet neben den verschiedenen Rankings auch die Schnittmengen der unterschiedlichen Suchmaschinen. [BeHe07, S.151]

Eine Suchmaschine, die die Ergebnismengen von Google und Yahoo gegenüberstellt ist beispielsweise Langreiter (Siehe Abbildung 9). Hierbei werden die Treffer der beiden Suchmaschinen als Punkte auf zwei parallel verlaufenden Geraden dargestellt. Jede Gerade steht für je eine Suchmaschine und jeder Punkt für je einen Treffer. Das Ranking der einzelnen Ergebnisseiten entspricht der Position auf der Geraden, wobei Dokumente mit dem höchsten Ranking sich ganz links befinden. Die Schnittmenge zwischen den beiden Suchmaschinen wird dadurch gekennzeichnet, dass die entsprechenden Punkte (die ja Ergebnisseiten repräsentieren) blau gekennzeichnet sind und zwischen den beiden Punkten auf den zwei Geraden eine Verbindungslinie eingezeichnet wird. [BeHe07, S.151 und WeBeHi09, S255-256]

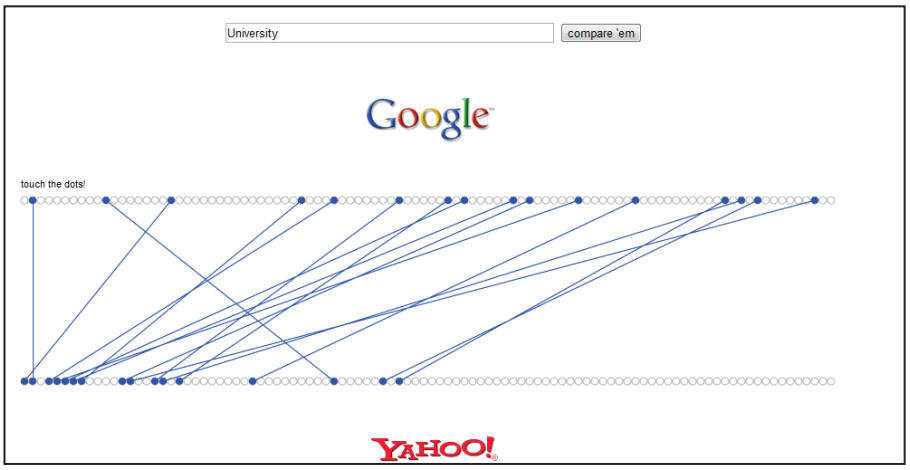


Abbildung 9: Vergleich der Suchergebnismengen von Google und Yahoo bei Langreiter [9]

3.3 Visualisierung der Anfrageterme

Die vom Nutzer eingegebenen Anfrageterme in einer Suchmaske stehen im Optimalfall repräsentativ für das Informationsbedürfnis des Nutzers. Der Nutzer hat also mit Bedacht entsprechende Anfrageterme gewählt und ordnet diesen bei seiner Suche eine hohe Bedeutung zu. Gerade bei der Beurteilung der Relevanz der Ergebnisdokumente spielen die Anfrageterme eine wichtige Rolle. Daher ist es sicherlich sinnvoll gerade diese Anfrageterme in gewisser Weise zu visualisieren, sodass der Nutzer beispielsweise deren Vorkommen quantifizieren oder auch lokalisieren kann. Hierbei können die Anfrageterme bei der Anzeige der Ergebnisse oder auch direkt im Dokument visualisiert werden.

3.3.1 Anfrageterme in Retrieval Ergebnissen visualisieren

Betrachtet man die Visualisierung der Anfrageterme auf den Google Ergebnisseiten, so erkennt man, dass die Anfrageterme im Ergebnistext (Titel, URL und Snippet) lediglich in Fettschrift dargestellt und damit hervorgehoben werden. Diese Darstellung sagt aber leider nichts darüber aus, wie oft der Term im Dokument vorkommt, geschweige denn wie wichtig der Term für das gesamte Dokument ist. Um diese Aspekte zu berücksichtigen wäre beispielsweise die Darstellung von TileBars sinnvoll.

[9] Entnommen aus `http://www.langreiter.com/exec/yahoo-vs-google.html?q=University` besucht: Juni 2012

TileBars versuchen den Zusammenhang zwischen den Anfragetermen und den Retrievalergebnissen zu visualisieren. Betrachtet man ein TileBar, so symbolisiert ein Rechteck jeweils ein Dokument und die Länge des Rechtecks lässt auf die Anzahl der Textsegmente im Dokument schließen (siehe Abbildung 10). In den Rechtecken stehen die Karos für die einzelnen Textsegmente in dem jeweiligen Dokument. Je dunkler so ein Karo ist, desto häufiger kommt der Anfrageterm in diesem jeweiligen Textsegment vor. Daraus kann man für jeden betrachteten Anfrageterm schließen wo, in welchem Dokument und in welchem Textsegment dieser am häufigsten vorkommt. Dabei gibt ein TileBar Hinweise auf die Länge des Dokuments und auf die Vorkommenshäufigkeit der Anfrageterme in den einzelnen Textsegmenten. [He09, S.254f]

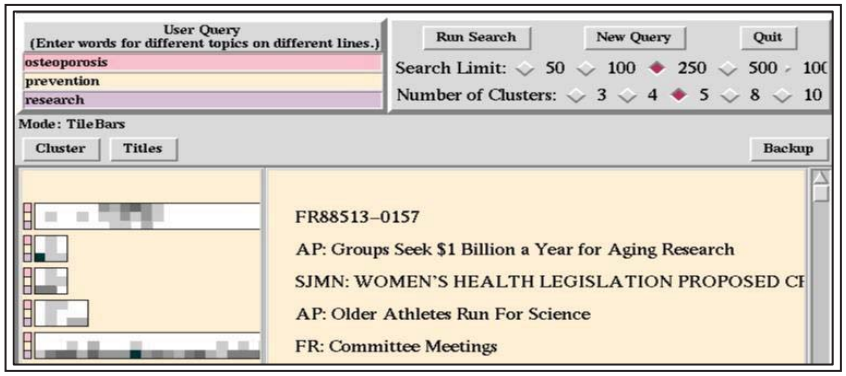


Abbildung 10: TileBar Visualisierung [10]

Betrachten wir beispielsweise in Abbildung 10 das zweite Dokument, so kann man aus der TileBar ablesen, dass das Dokument drei Textsegmente hat und in dem ersten Textsegment die ersten beiden Anfrageterme nicht vorkommen, dafür aber der letzte Anfrageterm sehr häufig vorkommt. Diese Darstellung bietet also auf einen Blick sehr viele Informationen über das Dokument und das Vorkommen der darin enthaltenen Anfrageterme. Problematisch ist lediglich, dass man sich vor der ersten Nutzung mit der Oberfläche und der Bedeutung der Rechtecke auseinandersetzen muss und das ablesen der Informationen eine kurze Einarbeitung benötigt. Wesentlich einfacher erscheint die Darstellung der vereinfachten TileBars. [He09, S.254ff]

Bei den vereinfachten TileBars hat man je Anfrageterm nur ein Karo pro Dokument (siehe Abbildung 11). Dieses eine Karo ist ein Indikator für die Vorkommenshäufigkeit des jeweiligen Anfrageterms in dem Dokument (sehr häufig = dunkelrot, sehr selten = hell/gelb). Diese Darstellung enthält keine Informationen über einzelne Textsegmente oder die Länge des Dokuments.

Betrachtet man das Beispiel in Abbildung 11, so sieht man auf den ersten Blick, dass für die entsprechende Anfrage das 8. Dokument sehr interessant erscheint, da dort alle drei Begriffe eine hohe Vorkommenshäufigkeit aufweisen. [He09, S.256f]

[10] Entnommen aus [He95]

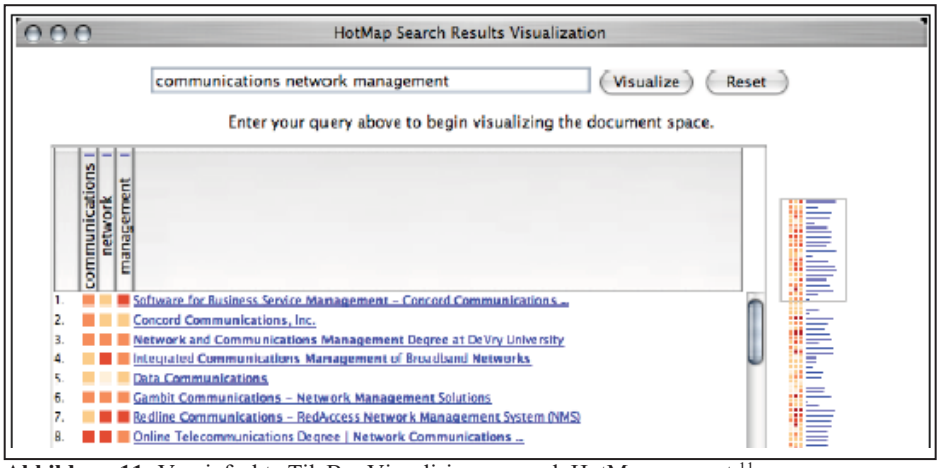


Abbildung 11: Vereinfachte TileBar Visualisierung, auch HotMap genannt [11]

Eine Studie zeigte, dass es bei Probanden zwischen der Nutzung von Google und den vereinfachten TileBars bei der Suche keine signifikanten Zeitunterschiede gab, die Probanden aber dennoch die vereinfachte TileBar Darstellung gegenüber Google bevorzugten. [He09, S.256f]

3.3.2 Anfrageterme in Dokumenten visualisieren

Man kann Anfrageterme nicht nur in den Ergebnislisten darstellen, sondern auch direkt in den Dokumenten visualisieren. Hierbei gibt es beispielsweise die Möglichkeit den Scrollbalken dafür einzusetzen, um die genaue Position der Anfrageterme anzuzeigen (siehe Abbildung 12). Nutzt man für jeden Anfrageterm unterschiedliche Farben, so kann auch die genaue Position mehrerer Anfrageterme dargestellt werden. [He09, S.252f]

[11] Entnommen aus [HoYa06]

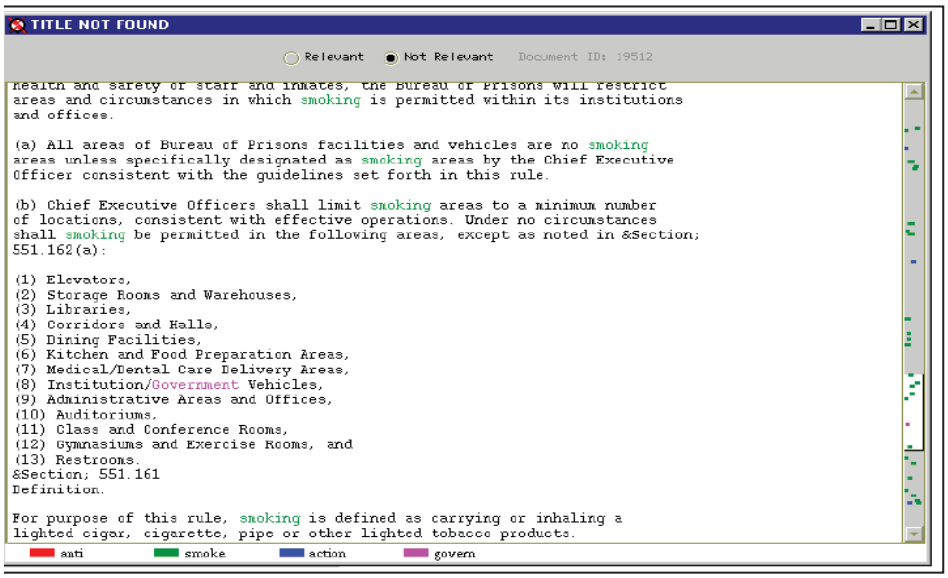


Abbildung 12: Position der Anfrageterme wird über Scrollbalken angezeigt [12]

4 Fazit und Ausblick

Die vorgestellten Ebenen der Informationsvisualisierung geben einen guten Einblick, an welchen Stellen Informationsvisualisierung bei Suchinterfaces eingesetzt werden kann. Aus den zahlreichen, oft auch sehr nützlichen Darstellungsformen wird ersichtlich, dass die allgemeine Idee, Informationsvisualisierung im Bereich des Information Retrieval einzusetzen schließlich sehr sinnvoll erscheint. Sicherlich würden sich viele Studenten, bei ihrer Suche nach relevanten Dokumenten für Seminare oder ähnliches, beispielweise über eine TileBar Darstellung der Suchergebnisse freuen. Auch eine Darstellung der Suchergebnisse in Cluster wäre vermutlich zur ersten Ideensammlung sehr hilfreich.

Dennoch wurde bisher der empirische Nachweis weder erbracht noch wiederlegt, dass Suchmaschinen mit Visualisierungen wirklich zu einer Steigerung der Nutzerzufriedenheit oder zu besseren oder effizienter ermittelten Suchergebnissen führen [Va04 und ArWo05]. Bei solchen Studien ist aber bei der Interpretation der Ergebnisse auch Vorsicht angebracht, da vor allem solche Faktoren, wie beispielsweise Hintergrundwissen oder Erfahrungen der Testpersonen die Ergebnisse

[12] Entnommen aus [By99]

stark beeinflussen. Je nach Auswahl der Testkandidaten können also komplett unterschiedliche Ergebnisse rauskommen. Ein Personenkreis mit nur sehr wenig Erfahrung im Bereich der Suche könnte zum Beispiel mit einer noch so kleinen Änderung der Suchoberfläche überfordert sein. Wohingegen eine erfahrene Gruppe an Testpersonen vielleicht gerade mit diesen Änderungen in der Suchoberfläche viel effektiver arbeiten könnte.

Betrachtet man aktuelle Suchmaschinen erkennt man sofort, dass die wenigsten Suchmaschinen Techniken der Informationsvisualisierung einsetzen. Ganz klar ist aber auch, dass große Suchmaschinen, wie Google oder Yahoo für die breite Masse ausgelegt sind und sicherlich gerade deswegen nur sehr zaghaft Änderungen in der Suchoberfläche vorgenommen werden. Mit aufwendigen Visualisierungsversuchen wird bei so großen Suchmaschinen eher nicht experimentiert. Doch auch visuelle Suchmaschinen bieten keine ausgereiften Lösungen an. Viele visuelle Suchmaschinen, die in der Literatur beschrieben werden sind außerdem oft nicht mehr online verfügbar oder es gibt nur noch Beta-Versionen. Warum ist also die Idee, Informationsvisualisierung in Suchinterfaces einzubringen, in der Praxis so schwer umsetzbar?

Ein Grund dafür könnte sein, dass es schwierig ist gleichzeitig eine intuitive Bedienbarkeit und einen Mehrwert gegenüber textbasierten Darstellungsformen zu bieten [WeBeHi09, S.281].

Außerdem wird man im Internet oft mit unstrukturierten Daten und komplexen Zusammenhängen konfrontiert. Desweiteren fehlen auch oft aussagekräftige Metadaten. All diese Aspekte erschweren die einheitliche Strukturierung der Daten, die schließlich notwendig ist um die Daten visualisieren zu können. Denn letztendlich kann die Visualisierung nur so gut sein, wie die Informationsstrukturierung, die der Visualisierung zugrunde liegt. [Dä99, S.19]

Will man nun einen Blick in die Zukunft wagen, so könnte es doch sein, dass gerade die Entwicklung in Richtung Semantic Web, auch den Einsatz der Informationsvisualisierung im Bereich von Suchmaschinen vorantreiben könnte. Im Semantic Web stehen semantische Annotationen (z. B. mittels RDF) zur Verfügung und genau diese könnten das Erstellen einer graphischen Repräsentation erleichtern.

References

[AnBrFL90] Anick, P., Brennan, J., Flynn, R., Hanssen, D., Alvey, B., Robbins J.: A direct manipulation interface for boolean information retrieval via natural language query. In Proceedings of the 13th Annual International ACM SIGIR Conference on Research and development in information retrieval, S. 135–150. ACM Press, New York (1990)

[ArWo05] Arnold, C., Wolff, C.: Evaluierung von Visualisierungsformaten bei der webbasierten Suche. In: Moes, Roswitha (eds.), Knowledge eXtended: die Kooperation von Wissenschaftlern, Bibliothekaren und IT-Spezialisten, S. 275-286. Reihe Bibliothek, Jülich (2005)

[BeHe07] Bekavac, B., Herget, J., Hierl, S. und Öttl, S.: Visualisierungskomponenten bei Web-basierten Suchmaschinen: Methoden, Kriterien und ein Marktüberblick. In: IWP - Information Wissenschaft & Praxis 58, S. 149-158. Dinges & Frick GmbH, Wiesbaden (2007)

[By99] Byrd., D.: A Scrollbar-based Visualization for Document Navigation. Proceedings of the Fourth ACM International Conference on Digital Libraries, S. 122-129. ACM Press, New York (1999)

[CaMaSh99] Card, S. K., Mackinlay, J. D., Shneiderman, B.: Readings in Information Visualization. Using Vision to think, San Francisco (1999)

[Ca08] Stuart Card: Information visualization. In A. Sears and J.A. Jacko (eds.), The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications, Lawrence Erlbaum Assoc Inc (2007)

[Ch00] Chi, E. H.: A Taxonomy of Visualization Techniques using the Data State Reference Model. Website (2000) Online unter `http://www-users.cs.umn.edu/~echi/papers/infovis00/Chi-TaxonomyVisualization.pdf`; besucht: August 2012.

[Dä99] Däßler, R.: Informationsvisualisierung. Stand, Kritik und Perspektiven. Website (1999) Online unter Online unter `http://fabdp.fh-potsdam.de/daessler/paper/InfoVis99.pdf`; besucht: August 2012.

[He09] Hearst, M.: Search user interfaces. Cambridge University Press, New York (2009)

[He99] Hearst, M.: User Interfaces and Visualization. In: Baeza-Yates, Ricardo A.; Ribeiro-Neto, Berthier (eds.): Modern Information Retrieval, S. 257-324. Addison Wesley, New York (1999).

[He95] Hearst, M.: TileBars: Visualization of Term Distribution Information in Full Text Information Access. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, S.59-66. ACM Press, New York (1995)

[HoYa06] Hoeber, O., Yang., X. D.: A comparative user study of web search interfaces: HotMap, Concept Highlighter, and Google. IEEE/WIC/ACM International Conference on Web Intelligence, S.866-874. IEEE Computer Society, Washington DC (2006)

[JoMc98] Jones, S., McInnes, S.: Graphical query specification and dynamic result previews for a digital library. In Proceedings of the 11th annual ACM symposium on User Interface Software and Technology, S.143-151. ACM Press, New York (1998)

[ScMü04] Schumann, H., Müller, W.: Informationsvisualisierung: Methoden und Perspektiven. In: it-Information Technology, Vol 46, Nr.3, S.135-141. Oldenbourg Verlag, München (2004)

[Va04] Vaughan, L.: New measurements for search engine evaluation proposed and tested. In: Information Processing and Management 40, S. 677-691. Elsevier, Canada (2004)

[WeBeHi09] Weinhold, T., Bekavac, B., Hierl, S. und Öttl, S.: Visualisierung bei Internetsuchdiensten. In: Dirk Lewandowski (eds.). Handbuch Internet-Suchmaschinen, S. 249-284. Akademische Verlagsgesellschaft, Heidelberg (2009)

Information Visualization for Text Analysis

Johannes Rettig

Otto-Friedrich-Universität Bamberg
johannes-david-reinhold.rettig@stud.uni-bamberg.de

Zusammenfassung. Die Grundbausteine einer jeden fundierten Entscheidungen bilden Informationen. Diese liegen in der Grundform jedoch lediglich als Datensätze vor, die in der Regel nur schwer interpretiert werden können. Insbesondere Beziehungen innerhalb von Daten sind oftmals komplex und nur unter großen Anstrengungen ergründbar.

Visualisierungssysteme bieten hier einen Mehrwert, in dem sie Daten grafisch aufbereiten. Diese Form der Darstellung entspricht der natürlichen menschlichen Wahrnehmung und unterstützt so den Benutzer in seinem Bestreben, Informationen zu gewinnen und Zusammenhänge aufzudecken.

Diese Arbeit befasst sich mit den Grundprinzipien der Visualisierung von Informationen und stellt das Potential entsprechender Systeme für die Analyse von Texten heraus.

Schlagworte: Information Visualization, Data Visualization, Visualisierungssysteme, Text Analysis, Text Mining, Data Mining,

1 Zielsetzung und Aufbau der Arbeit

'Ein Bild sagt mehr als tausend Worte'
(Sprichwort, Fred R. Barnard zugeschrieben)

Diese Lebensweisheit kann wohl jeder Mensch aus eigener Erfahrung bestätigen. Der Mensch ist ein visuell veranlagtes Wesen.

In Bruchteilen von Sekunden entscheiden Personen rein durch die äußere Erscheinung einer anderen Person, ob sie diese sympathisch finden oder nicht - der berühmte 'erste Eindruck' ist dann nur noch schwer umzupolen. Bei Internet Auktionen wird einem Angebot grundsätzlich mehr Erfolg eingeräumt, wenn ein Foto hinterlegt ist. In beiden Fällen möchte sich eine Person gerne erst 'ein Bild machen'. Die optische Präsentation ist hier in hohem Maße dafür verantwortlich, ob eine Person oder eine Sache akzeptiert wird oder nicht.

Insbesondere wenn es darum geht, einen schwierigen Sachverhalt für den Menschen verständlicher darzustellen, kann dies ein gutes Bild oder eine gute Grafik besser bewerkstelligen als ein umfangreicher Text.

Menschen sind für Informationen in visueller Form besonders zugänglich. Diesen Effekt möchte sich das Forschungsgebiet der Visualisierung zu Nutze machen. Es befasst sich mit der Veranschaulichung von abstrakten Daten durch Bilder oder Grafiken.

Abstrakte Daten können zum Beispiel Informationen über Dokumente und deren Inhalte sein. Welche Wörter tauchen besonders häufig auf? Mit welchen weiteren Dokumenten könnte es Verknüpfungen geben? Lassen sich aus diesen Abhängigkeiten wissenschaftliche Hypothesen ableiten? Dies sind nur beispielhafte Fragestellungen, denen sich das Forschungsgebiet der Textanalyse verschrieben hat. In diesem Kontext hat Information Visualization das Ziel, die Ergebnisse der Textanalyse abzubilden.

Diese Arbeit soll einen Überblick über die Methoden von Information Visualization geben, sowie deren Potenzial für die Textanalyse aufzeigen und Ziele definieren.

Kapitel zwei widmet sich den Grundprinzipien der Textanalyse. Zunächst wird dieser Begriff definiert und gegen andere Begrifflichkeiten abgegrenzt. Anschließend erfolgt eine Spezifizierung der zugrunde liegenden Analyseprozesse.

Textanalyse, Datenanalyse, Data Mining, Data Text Mining, Knowledge Mining - in der Fachliteratur werden diese Begriffe in vielen Fällen synonym behandelt. Insbesondere der Begriff des Data Minings ist jedoch vorherrschend. **Kapitel drei** eröffnet zunächst mit einer Einordnung der Textanalyse in das Data Mining und grenzt die weiteren Begrifflichkeiten dagegen ab. Im zweiten Teil werden die zwei extremen Ansätze des Data Minings, das automatisierte Data Mining und Information Visualization vorgestellt.

Im **vierten Kapitel** wird Information Visualization in der Textanalyse anhand konkreter Anwendungsformen betrachtet. Zunächst findet eine Definition der Ziele der Visualisierung in diesem Kontext statt. Anschließend werden verschiedene existierender Systeme beleuchtet.

Das **fünfte Kapitel** widmet sich den Einsatzmöglichkeiten und Grenzen von Information Visualization in der Textanalyse, während das **sechste Kapitel** ein Fazit zieht und einen Ausblick auf künftige Entwicklungen und Forschungsansätze wirft.

2 Grundprinzipien der Textanalyse

Das Zunehmen von sehr großen Datensammlungen steigert unser Bedürfnis danach, diese Daten zu analysieren und Informationen daraus zu gewinnen. Die Wurzeln dieser Analysen liegen in der statistischen Auswertung von gesammelten Datensätzen. Durch das Aufkommen von Computern konnte eine komplexe und automatische Visualisierung dieser Datensätze vorgenommen werden (Fayyad et al.

2002, S.1 ff.). Das folgende Kapitel bietet einen Überblick über die Prinzipien der Textanalyse, die die Auswertung von Dokumenten beschreibt.

2.1 Begriffsdefinition

Den Begriff **Textanalyse** versteht SPSS als "eine Methode, um verwendbares Wissen aus unstrukturierten Textdaten zu gewinnen anhand der Identifizierung von Kernbegriffen, -gesinnungen und -trends und anhand dieses Wissens Entscheidungen zu fällen." (SPSS 2008, S.3).

Feldman definiert Textanalyse als einen Prozess, der die "[Vor]verarbeitung von Dokumentenkollektionen [...], die Speicherung der Zwischendarstellungen, Techniken zur Analyse dieser Darstellungen sowie die Visualisierung der Ergebnisse" umfasst (Feldman 2004). Anhand dieser Ergebnisse werden Entscheidungen getroffen.

Während SPSS eine allgemeingültige Beschreibung der Textanalyse abgibt, verfolgt Feldman einen prozessorientierten Ansatz. Beide Definition ergänzen sich aber symbiotisch, da beide **die analysebasierte Gewinnung von Wissen aus Dokumenten** als wesentlichen Bestandteil erkennen, ebenso wie die darauf folgende **Entscheidungsfindung**. Die Prozesse der Textanalyse werden in Abschnitt 2.3. noch genauer betrachtet.

Textanalyse darf nicht mit einer Suche in Dokumenten oder Texten verwechselt werden. Ein Suchfunktion folgt einem Top-Down Ansatz um Informationen in Dokumenten zu finden: Der Benutzer gibt die gewünschten Suchbegriffe in ein Eingabefeld ein und erhält Dokumente, die diese Begriffe enthalten oder referenzieren. Dies setzt beim Benutzer voraus, dass dieser in der Lage ist, seine Anfragen entsprechend zu strukturieren. So fällt er die Entscheidung, welche Begriffe für seine Suche relevant sind.

Die Textanalyse folgt dagegen einem Bottom-Up Ansatz und stellt die genannten Voraussetzungen nicht an den Benutzer. Stattdessen stellt sie Beziehungen zwischen Dokumenten her und deckt so Zusammenhänge auf (SPSS 2008, S.3).

In diesem Kontext erfolgt ebenso eine Definition des Begriffes **Dokument**. Darunter werden viele verschiedene Dinge verstanden, wie zum Beispiel wissenschaftliche Artikel, Antworten auf eine Marktstudie, Datensätze in einer Datenbank (beispielsweise Aufzeichnungen eines Call Centers oder E-Mails von Kunden) und viele andere/weitere textuelle Inhalte (SPSS 2008, S.3). Kennzeichnend ist in jedem Fall der Inhalt in Textform, der ein Dokument ausmacht. Darüber hinausgehende Datensätze, wie etwa mathematische Daten, stellen kein Dokument dar und werden daher in dieser Arbeit nur der Vollständigkeit halber erwähnt.

2.2 Textanalyse und Text Mining

In der Fachliteratur existieren konkurrierende Begrifflichkeiten zur Textanalyse. Ein Begriff der häufig genannt wird, ist Text Mining, der oftmals synonym verwendet. An

dieser Stelle erfolgt zunächst eine Definition des Begriffs **Text Mining** sowie anschließend eine Abgrenzung zur Definition von Textanalyse.

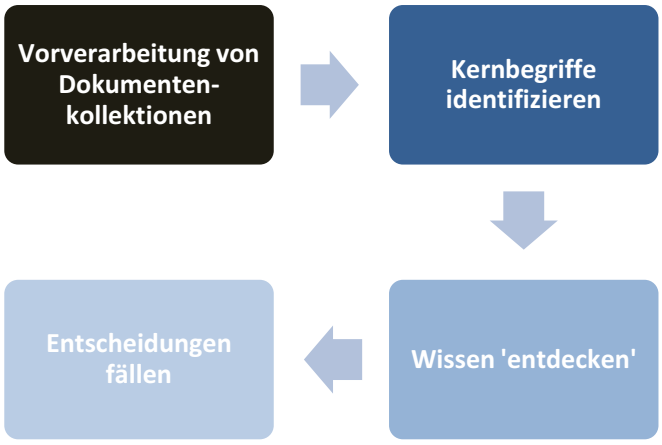
Text Mining definieren Feldman und Sanger als "einen wissensintensiven Prozess, in dem ein Benutzer mit einer Dokumentensammlung [...] interagiert, unter Verwendung von Analyse Tools". Text Mining versucht, **sinnvolle Informationen aus Datenquellen** zu **gewinnen** (Feldman & Sanger, 2006, S. 1). Solche Daten können einen großen Schatz an Wissen enthalten (Nasukawa & Nagano, 2001, S. 516).

Es wird offensichtlich, dass **Text Mining** und **Textanalyse** in der hier dargelegten Form als **Synonyme** die gleichen Aufgaben beschreiben. Unter beiden Begriffen wird letztendlich die Informationsgewinnung aus textuell basierten Dokumenten unter Verwendung analytischer Werkzeuge verstanden. Daher werden die Begriffe des Text Minings und der Textanalyse im Folgenden als Synonyme verwendet.

Ferner ist auch der Begriff des **Document Mining**s als **Synonym** zu verstehen, da ein Dokument als Träger von Text gilt. Document Mining betont definitorisch also die Darreichungsform, hat aber ebenso zum Ziel, Informationen aus diesen Dokumenten zu gewinnen (Gee & Light 2002, S. 345 ff.).

2.3 Analyseprozesse der Textanalyse

Die zentralen **Analyseprozesse der Textanalyse** bestehen wie in Abbildung 1 dargestellt aus der Vorverarbeitung von Dokumentenkollektionen, der Identifikation von Kernbegriffen, dem 'Entdecken' von Wissen und dem Fällen von Entscheidungen.



(Abb.1: Analyseprozesse der Textanalyse - Eigene Darstellung basierend auf SPSS 2008, Feldman 2004)

Die einzelnen Teilprozesse werden nachfolgend genauer betrachtet:

Vorverarbeitung von Dokumentenkollektionen - Zunächst wird eine Dokumentenkollektion benötigt. Diese kann aus Dokumenten unterschiedlichster Sprachen und Dateitypen bestehen. Zur Vorbereitung der Kollektion gehört zunächst die Identifikation der verwendeten Sprachen, um Begriffe richtig zuordnen zu können. Desweiteren könnte hier zum Beispiel die Konvertierung in ein einheitliches Format oder die Unterteilung der Dokumente in einzelne Bestandteile stattfinden (SPSS 2008, S. 7).

Kernbegriffe identifizieren - Jedes Dokument wird durch eine ausreichende Zahl an Informationen dargestellt, die den Inhalt repräsentieren (Gee & Light 2002, S. 348). Diese Informationen sind Kernbegriffe bzw. -aussagen als Repräsentanten des Dokumentes (SPSS 2008, S.3). Die Reduzierung des Dokumentes auf einige wenige Schlüsselwörter vermeidet, dass Begriffe, die im Dokument zwar erwähnt werden aber nicht das eigentliche Thema umreißen, dennoch als repräsentativ gelten.

Wissen 'entdecken' - Die Repräsentanten können vielfältig genutzt werden. Es könnte etwa an dieser Stelle eine Linkanalyse erfolgen, die Beziehungen zwischen Begriffen aufdeckt. Dies soll an folgendem Beispiel verdeutlicht werden. Es handelt sich um exemplarische Aussagen aus drei verschiedenen Texten:

- Firma A wurde von Firma B übernommen.
- Firma B erwarb Firma A.
- Firma B's Übername der Firma A ist abgeschlossen.

Eine Analyse dieser Aussagen sollte dabei zu dem Ergebnis kommen, dass sie alle die gleiche Bedeutung haben. Die Beziehung zwischen Firma A und Firma B kann also in mehreren verschiedenen Dokumenten festgestellt werden.

Entscheidungen fällen - Anhand der durchgeführten Analysen kann ein sogenanntes Prognosemodell entwickelt werden, auf dessen Basis Entscheidungen getroffen werden. Die Prognose besteht darin, aus den vorliegen Daten Rückschlüsse auf künftige Entwicklungen zu ziehen.

Eine Möglichkeit besteht darin, die Beziehungen zwischen Begriffen grafisch abzubilden und in Berichte aufzunehmen. Im Anschluss werden diese Berichte interpretiert und entsprechende Entscheidungen abgeleitet.

Eine weitere Möglichkeit besteht darin, die Prognosemodelle in Anwendungssysteme zu integrieren. Denkbar ist etwa die automatische Generierung von Verkaufsangeboten, die Identifizierung der Kreditwürdigkeit von Kunden oder die Aussortierung extremer Feedbackpositionen von Kunden (SPSS 2008, S. 16).

Zusammenfassend lässt sich feststellen, dass die Textanalyse ein enorm mächtiges Instrument darstellt um Entscheidungen zu treffen. Die Anwendungsgebiete sind dabei vielfältig:

- Pharma Unternehmen könnten die Effektivität einer Behandlungsform analysieren, indem Kommentare der behandelten Patienten, wie sie sich vor, nach und während der Behandlung gefühlt haben ausgewertet werden.
- Trends und Entwicklungen in der Wirtschaft können bereits frühzeitig vorausgesagt werden, ohne dass alle potentiell relevanten Dokumente auch wirklich gelesen werden müssen.
- Geheimdienste könnte anhand verschiedener Dokumente beispielsweise Verbindungen zwischen terroristischen Zellen aufdecken.
  (SPSS 2008, S. 17).

3 Data Mining

Eine weitere sehr weit verbreitete Terminologie ist Data Mining. Die **Textanalyse** beziehungsweise das Text Mining kann als **Teil des Data Minings** angesehen werden. Das folgende Kapitel ordnet die Textanalyse daher zunächst in das Data Mining ein und stellt anschließend dessen Ansätze vor. Information Visualization wird im Abschluss des Kapitels näher beleuchtet.

3.1 Einordung der Textanalyse in das Data Mining

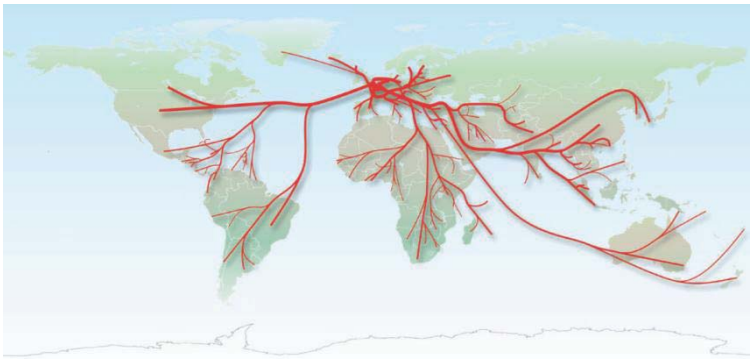
Unter **Data Mining** versteht man die automatische Entdeckung von Trends und Mustern aus sehr großen Datensätzen, um anhand dieser Informationen Entscheidungen zu treffen (Hearst 1999, S.3). Fayyad und Uthurusamy definieren Data Mining allgemeiner als "Knowledge Discovery in Databases" (Fayyad & Uthurusamy 1996, S.1).

Es wird deutlich, dass Data Mining und Text Mining eine große Ähnlichkeit aufweisen: Beide Ansätze möchten **unter Verwendung analytischer Werkzeuge Informationen aus einem Datenträger gewinnen**. Der Unterschied besteht lediglich in der Art des Datenträgers. Während **Text Mining** sich explizit auf **textuell basierte Dokumente** konzentriert, ist **Data Mining** auf **Datensätze jedweder Art** ausgerichtet. Text Mining stellt daher ein Teilgebiet des Data Minings dar. In der Literatur wird daher unter anderem auch der Begriff des Text Data Minings verwendet, der eben diesen Umstand betont (Hearst 1999, S.3).

Textuelle Daten sind als solche jedoch nicht immer eindeutig zu erkennen, eine Abgrenzung ist im Einzelfall daher schwierig. Abbildung 3 zeigt Stromlinien, die den Import von Ressourcen jeder Art in das Vereinigte Königreich darstellen.

Auf den ersten Blick verbinden wir mit dieser Darstellung keine textuellen Inhalte, da wir lediglich einige Linien sehen. Um zu entscheiden, ob diese Darstellung und die

zugrunde liegenden Daten Teil des Data Minings und/oder des Text Mining sind, ist letztendlich die Frage zu beantworten: Was ist eigentlich ein 'Text'?



(Abb.3: Import von Ressourcen in das Vereinigte Königreich - Simmons 2006)

Die Brockhaus Enzyklopädie definiert **Text** als eine fixierte sprachliche Äußerung, die schriftlich erfolgen kann, aber nicht muss. Ferner werden einem Text bestimmte Kriterien zugesprochen, wie beispielsweise die Zuordnung zu einer sich äußernden Person oder kulturelle, soziale oder persönliche Annahmen, die sich aus dem Kontext ergeben. Einzelne Kriterien können deutlicher auftreten, während andere wegfallen, so dass eine klar Abgrenzung nicht immer möglich erscheint. Letztendlich liegt die Entscheidung beim Leser, ob er eine Sammlung von Schriftzeichen als Text auffasst. Dies hängt maßgeblich davon ab, ob er darin einen **Sinn** erkennen kann oder nicht (Brockhaus 2012).

Diese Definition widerspricht teilweise der Definition von Dokumenten im Punkt 2.1. Dort werden unter anderem auch "Datensätze in einer Datenbank" als textuelle Inhalte verstanden (SPSS 2008, S.3). Natürlich ergeben auch Datensätze einen Sinn, jedoch erfordern sie etwa im Vergleich zu einem Zeitungsartikel die individuelle Verknüpfung von vielen Informationen und deren Interpretation, während bei einem Artikel die Intention bereits durch die Überschrift deutlich wird. Auch können Datensätze nicht zwangsläufig als sprachliche Äußerung angesehen werden, sie sind auch nicht unbedingt einer sich äußernden Person zugeordnet. Wichtige Merkmale der oben genannten Text Definition fehlen hier.

In der einschlägigen IT-Literatur wird textueller Inhalt als eine Anordnung von Zeichen, also eine Zeichenkette angesehen. Diese definiert Ullenboom als "eine Folge von Buchstaben, Ziffern oder Sonderzeichen" (Ullenboom, S. 100). Der Datentyp String fungiert hier als Speicherplatz für solche Zeichen.

Zu beachten ist jedoch, dass Ziffern, die als Zeichenkette (String) gespeichert werden, in der Regel nicht oder nur unter Aufwand von Anwendungssystemen auch als Zahl wahrgenommen werden können. Um mit solchen Ziffern zu rechnen wird in der Regel der Datentyp Integer (Ganze Zahl) oder Double (Fließkommazahl) verwendet.

Daher wird aus diesen Zusammenhängen **Text** als eine **Zeichenkette** (String) definiert, die nicht als rechnerische Zahl wahrgenommen werden soll. Die Betrachtung von **rein rechnerische**n **Zahlen** ist daher **kein Text** und wird in dieser Arbeit nicht betrachtet. Entsprechend werden in dieser Arbeit nur Dokumente betrachtet, deren Inhalt rein textueller Natur nach dieser Definition entspricht.

Abbildung 3 präsentiert uns Linien, die im Vereinigten Königreich münden. Diese Linien basieren auf Importzahlen (Simmons 2006). Basierend auf der aufgestellten Definition von Text ist diese Auswertung also dem Data Mining zuzuordnen, nicht jedoch dem Text Mining.

Im folgenden Punkt werden nun die beiden extremen Ansätze des Data Minings vorgestellt. Anschließend erfolgt eine Einschätzung, in wie weit diese Ansätze auch für das Teilgebiet des Text Minings Relevanz erlangen können.

3.2 Ansätze des Data Minings

Die beiden Extremfälle im weiten Spektrums des Data Minings stellen automatisiertes Data Mining auf der einen Seite, sowie Information Visualization auf der anderen Seite dar (Pilote & Fillion 2002, S. 115). Beide Extreme werden im Folgenden vorgestellt und ihre Relevanz für die Textanalyse aufgezeigt.

3.2.1 Automatisiertes Data Mining

Unter automatisiertem Data Mining versteht man im Extrem die Automatisierung aller Teilschritte des Data Minings (Abbildung 1 gilt hier analog), also von der Vorverarbeitung der Dokumente über die Identifizierung von Kernbegriffen und die 'Entdeckung' von Wissen bis hin zum Fällen von Entscheidungen. **Visualisierung** wird hier eingesetzt um **Ergebnisse anzuzeigen** (Pilote &Fillion 2002, S. 115 f.).

Automatisiertes Data Mining erfordert erheblichen Aufwand an das Design, die Programmierung und das Testen eines solchen Systems. Auch muss dieses System immer wieder dem aktuellen Bedarf angepasst werden. Dabei nimmt die Performance mit der Zeit automatisch ab, da signifikante Entwicklungen in der Regel so lange ignoriert werden, bis sie von Hand in das System eingefügt werden (Pilote &Fillion 2002, S. 115).

Sofern die Nutzer mit Visualisierungssystemen vertraut sind, können diese Probleme aber teilweise umgangen werden. Die Nutzer besitzen ein Grundgefühl, ob die gelieferten Ergebnisse komplett und korrekt sind, sie entwickeln ein Gefühl für das große Ganze. Wurde eine signifikante Entwicklung in das System noch nicht eingepflegt, kann ein Nutzer diesen Umstand durch seine Erfahrung aufdecken und das System oder den Data Mining Prozess selbstständig anpassen (Pilote, Fillion 2002, S. 115 f.).

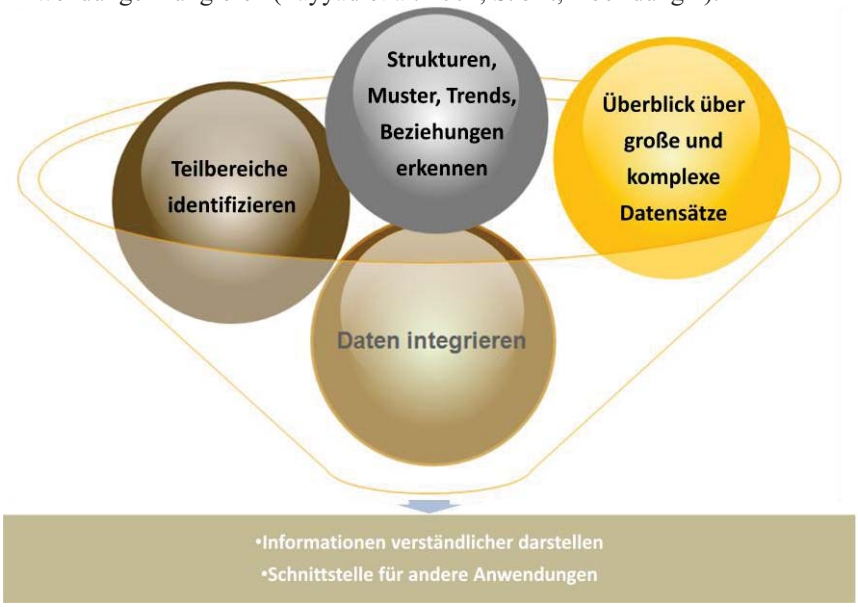
3.2.2 Information Visualization

Das andere Extrem des Data Minings stellt Information Visualization dar. Es ist gekennzeichnet von der Interaktion mit dem Benutzer, der das System steuert.

Grinstein und Ward definieren **Visualization** als "Die grafische (im Gegensatz zur textuellen oder verbalen) Übermittlung von Informationen" (Grinstein& Ward 2002, S. 23). Eine ähnliche Definition bietet auch Jung an, der Visualisierung als "das Umwandeln symbolischer oder nummerischer Information in visuelle Information wie Bilder, Graphiken und Karten" (Jung 1998, S. 2) beschreibt. Die Brockhaus Enzyklopädie definiert: "[Visualisierung beschreibt die] bildliche Aufbereitung, Darstellung und Kommunikation von Information[en]" (Brockhaus 2012).

Alle Definitionen stimmen darin überein, Visualisierung als die **grafische beziehungsweise visuelle Übermittlung von Informationen** anzusehen.

Visualisierung unterstützt Menschen darin, **Strukturen, Muster, Trends, oder Beziehungen in Daten** zu **erkennen**. Sie hilft, einen **Überblick über große und komplexe Datensätze** zu bekommen, sie kann **Daten integrieren** oder bestimmte **Bereiche** für weitere Analysen **identifizieren**. In einem idealen Anwendungssystem unterstützt die Visualisierung die Wahrnehmung des Menschen (Grinstein & Ward 2002, S. 21). Sie kann dabei sehr bedeutend sein, um Informationen für Menschen verständlicher darzustellen. Ferner kann sie aber auch als eine Schnittstelle für andere Anwendungen fungieren (Fayyad et al. 2002, S. 5 f., Abbildung 4).



(Abb. 4: Warum Visualisierung? - Eigene Darstellung nach Grinstein & Ward 2002, S. 21 und Fayyad et al. 2002, S. 5 f.)

In der Literatur konkurrieren die Begriffe **Data Visualization** und **Information Visualization** miteinander. Die weiter oben angeführten Definitionen der Visualisierung umfassen jeweils bereits den Begriff der Information. Der Begriff der Information Visualization stellt diesen Umstand noch einmal deutlich heraus, während Data Visualization sich bewusst auf Daten anstatt Informationen bezieht.

Die Begriffe **'Informationen' und 'Daten'** sind voneinander abzugrenzen. Dies soll an einem Beispiel verdeutlicht werden. Während die Aussagen "Dies ist ein Buch" und "This is a book" zwar die gleiche Information repräsentieren (ein Buch liegt vor), handelt es sich dennoch um unterschiedliche Daten (vgl. Punkt 3.3, die Information wird durch unterschiedliche Zeichenketten repräsentiert). Eine **Information** kann in diesem Kontext als **Interpretation von Daten** angesehen werden (Ferstl & Sinz 2006, S. 131 f.).

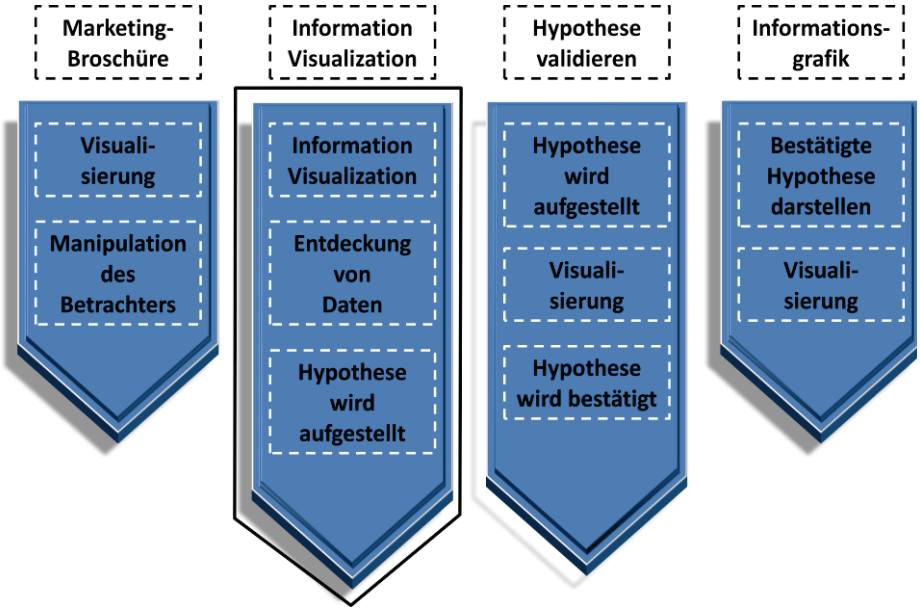
Diese Interpretation erfolgt dabei **durch den Leser**. Er zieht aus beiden Aussagen jeweils die gleiche Information. Auch der Benutzer zieht aus den angezeigten Daten am Bildschirm Schlüsse und legt die Ergebnisse entsprechend aus. Dies übernimmt nicht der Computer, selbst beim automatisierten Data Mining werden die Schlüsse immer noch durch den Betrachter gezogen.

Die Begriffe 'Information' und 'Daten' stellen daher im Kontext der Visualisierung lediglich **unterschiedliche Abstraktionsebenen** dar: Data Visualization beschreibt den Prozess der Visualisierung von Daten auf einem Bildschirm und vergleichbaren Geräten. Information Visualization ergänzt dies um die menschliche Interpretation der Daten. Da die Interpretation jedoch nicht Teil des Systems ist, können beide Begriffe als **Synonyme** angesehen werden, wenn es um die Eigenschaften der damit beschriebenen Anwendungssysteme geht.

Im folgenden Abschnitt wird eine kurze Übersicht zu den Arten der Visualisierung gegeben, ehe anschließend die Komponenten eines Information Visualization Systems detailliert dargestellt werden.

3.2.2.1 Visualisierungsarten

Visualisierung im Allgemeinen kann zur Entdeckung von Daten, zur Bestätigung einer Hypothese oder auch zur Manipulation des Betrachters (wie etwa in einer Marketing Broschüre) eingesetzt werden.



(Abb. 5: Visualisierungsarten - Eigene Darstellung nach Grinstein & Ward 2002, S. 22)

Nutzer suchen im Fall von ***Information Visualization*** als Teil des Data Minings in den Daten nach Strukturen oder Trends und leiten davon Hypothesen ab (Grinstein & Ward 2002, S. 22, Abbildung 5).

Dies unterscheidet sich von einer Visualisierung wie beispielsweise einer ***Informationsgrafik***. Diese transportiert zwar ebenso Informationen, jedoch hat hier der Nutzer bereits eine Hypothese aufgestellt und überprüft diese anhand der Grafik. Die Systemparameter sind daher oft schon festgelegt, Analysewerkzeuge sind notwendig, um die Hypothese zu testen (Grinstein& Ward 2002, S. 22, Abbildung 5).

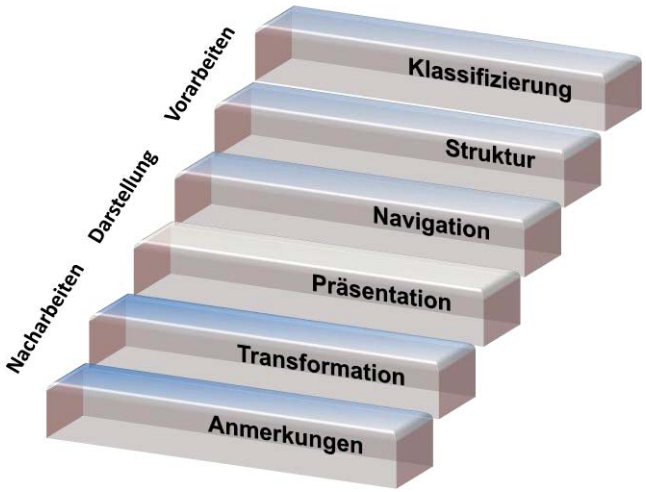
Ferner kann dem Nutzer bereits eine ***validierte Hypothese*** vorliegen. Er weiß daher, was dargestellt werden soll und kann sich auf Feinheiten konzentrieren, um die Darstellung zu optimieren. Dies stellt die stabilste und am besten vorhersagbare Visualisierungsart dar (Grinstein& Ward 2002, S. 22, Abbildung 5).

Im Falle einer ***Marketingbroschüre*** (oder einem vergleichbaren Druckerzeugnis) dient die Visualisierung ausschließlich dazu, einen bestimmten Zusammenhang an den Nutzer zu transportieren (Grinstein& Ward 2002, S. 22 f.). Die Art der Darstellung, wie auch gewählte Farben, Schriften usw. manipulieren den Betrachter und sollen ihn beispielsweise zu einer Kaufentscheidung bewegen.

Der folgende Abschnitt betrachtet die Komponenten eines Visualisierungssystems und stellt heraus, dass Visualisierung mehr ist als die reine Präsentation von Grafiken.

3.2.2.2 Komponenten

Ein Visualisierungssystem ist durch einen komponentenweisen Aufbau gekennzeichnet.



(Abb. 6: Komponenten der Visualisierung - Eigene Darstellung nach Pilote & Fillion 2002, S. 104)

Pilote und Fillion (Pilote & Fillion 2002, S. 104) erkennen folgende semantischen **Komponenten**:

1. ***Klassifizierungsmechanismus***: Dieser Mechanismus unterteilt den Datensatz zunächst in Klassen. Alle folgenden Operationen finden auf diesen Klassen statt. Die Anwendung der Operationen auf die Klassen erwies sich als besonders nützlich.
2. ***Strukturelle Operationen***: Sie arbeiten auf Basis der Attribute der Klassen (auf Meta-Ebene) um die Anzahl und die Mächtigkeit von Unterklassen zu variieren. Diese Operationen können weiter unterteilt werden in Produkte, Vereinigen und Zuordnungen.
    a. *Produkt-Operationen* kombinieren Attribute.
    b. *Vereinigungs-Operationen* (Generalisierungen) gruppieren Werte von ausgewählten Attributen.
    c. *Zuordnungs-Operationen* verbinden Unterklassen miteinander, oder mit separaten Datensätzen nach einem festgelegten Schlüssel
3. ***Navigations-Operationen***: Sie lassen die Filterung der Unterklassen zu, so können Unterklassen hinzugefügt oder weggelassen werden; Sie können verborgen, ausgewählt, gelöscht und wiederhergestellt werden, oder es gibt die Möglichkeit in feinere Level der Darstellung zu zoomen.
4. ***Präsentations-Operationen***: Ein gegebener Datensatz kann aus verschiedenen Blickwinkeln betrachtet und auf unterschiedliche Arten dargestellt werden. Dies umfasst auch das Hinzufügen von Gesamtzahlen,

Durchschnitten oder Prozentangaben. Auch können die darzustellenden Attribute je nach Bedarf ausgewählt werden.

5. ***Daten Transformation***: Berechnete Attribute können hinzugefügt werden. Deren Werte werden so berechnet, wie sie gebraucht werden.
6. ***Anmerkungs-Operationen***: Sie erlauben es, Aufzeichnungen durch den Nutzer zu speichern. Damit können beispielsweise Fehler in den Daten markiert werden.

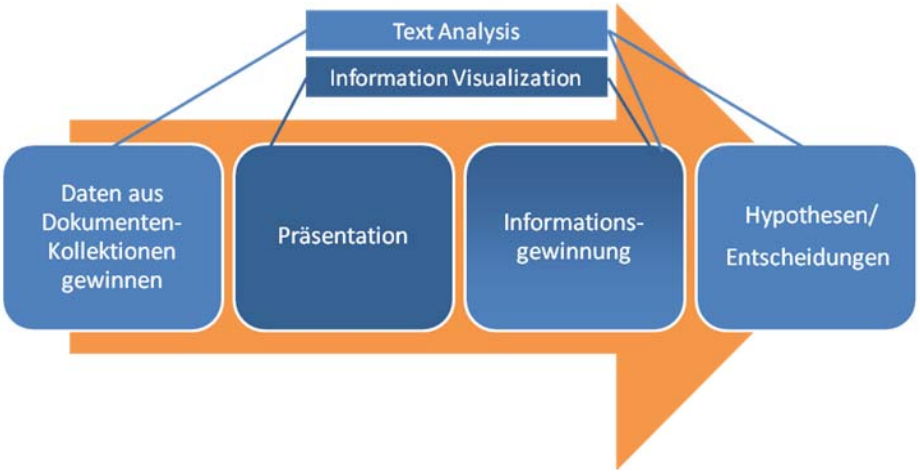
Pilote und Fillion kommen zu dem Ergebnis, dass die Verwendung von qualitativ hochwertigen Grafiken kein essentieller Bestandteil von Data Visualization ist, da viele Komponenten des Visualisierungssystems unabhängig von der grafischen Darstellung einsetzbar sind. Die genannten Kernmechanismen unterstützen jedoch jedes Visualisierungssystem unabhängig vom Grad der Komplexität. Einfache, begrenzte Grafiken können Zeitvorteile mit sich bringen, vor allem dann, wenn sie im Rahmen des Data Minings weiterverarbeitet werden. (Pilote & Fillion 2002, S. 103).

4 Information Visualization in der Textanalyse

4.1 Ziele

Das Ziel der Textanalyse stellt das **Treffen von Entscheidungen** bzw. das Aufwerfen von zugrundeliegenden Hypothesen dar. Hierfür werden zunächst **Daten aus** großen **Dokumentenkollektionen** gewonnen. Die Kombination dieser Daten führt zu **Wissen**/Informationen, die als Basis für Entscheidungen fungieren (vgl. Kapitel 2.3.).

Information Visualization setzt dazwischen an und präsentiert die gewonnenen Daten zweckorientiert. Die sinnvolle **Präsentation** erleichtert die Wissens- bzw. Informationsgewinnung und unterstützt die Entscheidungsfindung (vgl. Abbildung7).



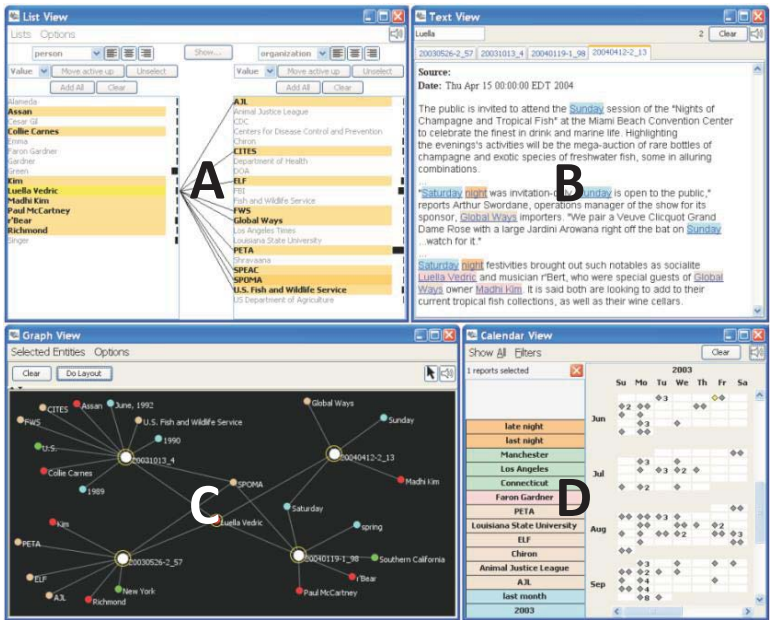
(Abb. 7: Ziele von Information Visualization in der Textanalyse - Eigene Darstellung basierend auf SPSS 2008, Feldman & Sanger 2006, sowie Grinstein & Ward 2002)

4.2 Anwendungsformen

Information Visualization wird in der Textanalyse vielfältig eingesetzt. Dabei ist zu unterscheiden zwischen der Visualisierung von Wortverbindungen, von Häufigkeiten, sowie von Zitat-Beziehungen. Die folgenden Unterpunkte stellen die jeweilige Anwendungsform anhand praktischer Systeme vor.

4.2.1 Visualisierung von Verbindungen zwischen Wörtern

Dieser Abschnitt beschreibt, wie die Verbindung von Einheiten zwischen Dokumenten visualisiert werden kann (Hearst 2009, S. 281).



(Abb. 7: Die Jigsaw Oberfläche - Görg et al. 2007, S. 2)

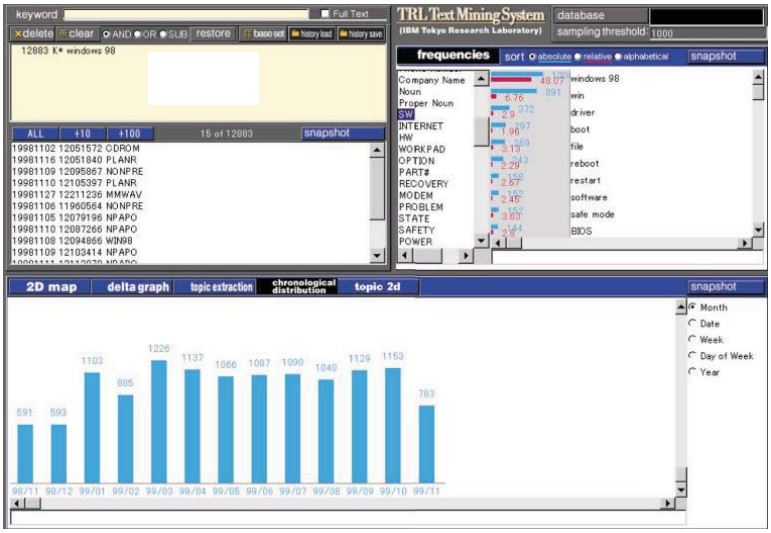
Das **Jigsaw** System soll hier als Beispiel dienen. Es unterstützt verschiedene Ansichten der Verbindungen zwischen Wörtern aus verschiedenen Dokumenten. Hierfür werden verschiedenen Sichten bereitgestellt (vgl. Abbildung 7):

- **List View (A)**: Hier werden Verbindungen zwischen Wörtern gezeichnet und koloriert sie. Der Autor Vedric hat hier zum Beispiel besonders häufig Artikel mit dem Begriff PETA veröffentlicht.
- **Text View (B)**: Das aktuell betrachtete Dokument und darin enthaltene Wörter werden angezeigt.

- **Graph View (C)**: Die Verbindungen zwischen Wörtern und Dokumenten werden in einem Diagramm dargestellt. Die größeren, umrandeten Knoten stellen dabei Dokumente dar, anhängig sind entsprechende Kernbegriffe.
- **CalenderView (D)**: Ein Überblick über alle relevanten Dokumente und die darin enthaltenen Wörter, abhängig von deren Publikationsdatum, wird geboten.

Ein weiteres Beispiel stellt das **TAKMI** System dar (vgl. Abbildung 8).

TAKMI wurde unter anderem eingesetzt, um in einem Call Center Fragen und Antworten zu Computerproblemen zu erfassen. Abbildung 8 zeigt, dass im Jahre 1998/1999 in diesem Call Center der Begriff 'Windows 98' ein sehr großes Thema darstellte (Nasukawa& Nagano, 2001, S. 972 ff.).



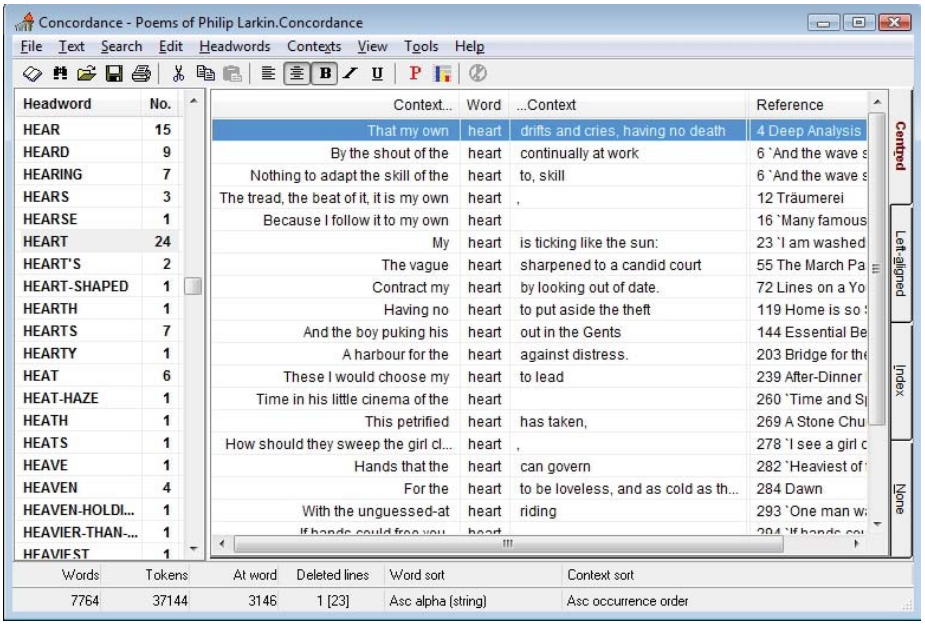
(Abb. 8: Die TAKMI Oberfläche - Nasukawa & Nagano, 2001, S. 973)

4.2.2 Visualisierung von Worthäufigkeiten und Übereinstimmungen

Ein weiteres interessantes Anwendungsgebiet besteht im Bereich von literarischen Werken. Vielfältige Analysen sind hier denkbar, wie beispielsweise die Untersuchung von Dokumenten hinsichtlich von Übereinstimmungen zur Erkennung von Plagiaten führen kann.

Ein System, dass solche Übereinstimmungen abbilden kann, ist das **Concordance System** (vgl. Abbildung 9).

Im Beispiel wurde der Begriff 'Heart' ausgewählt. Jedes Auftreten des Begriffs wird in seinem Kontext dargestellt. Unter 'Reference' sind Informationen zu dem jeweiligen Dokument und der Seite dargestellt.



(Abb. 9: Concordance Oberfläche - Watt 2011)

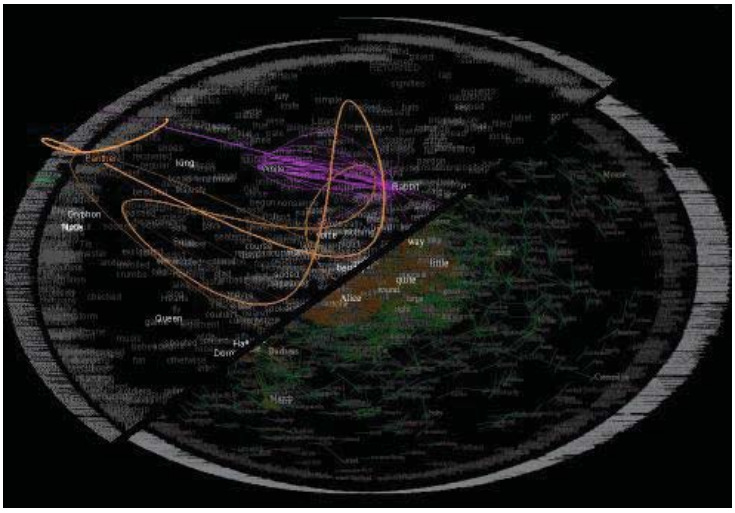
Auch die Visualisierung von Worthäufigkeiten kann für Autoren interessant sein, um etwa Wiederholungen von Begriffen zu erkennen und diese zu umgehen.

(Abb. 10: Wordle Wolke basierend auf einer Rohfassung dieser Arbeit - Feinberg 2011)

Wordle stellt eine Web-Anwendung dar (http://www.wordle.net). Nach der Eingabe eines Textes lässt sich eine Wordle Cloud erzeugen. Je häufiger ein Begriff auftaucht, desto größer wird dieser dargestellt. Diese Applikation ist durch einen geringen

Umfang und den fehlenden Vergleich zwischen Dokumenten gekennzeichnet, dennoch wird eine Grundfunktionalität bereitgestellt (vgl. Abbildung 10).

Ein Visualisierungstool, das sowohl Worthäufigkeiten als auch Übereinstimmungen darstellt, ist **TextArc**.



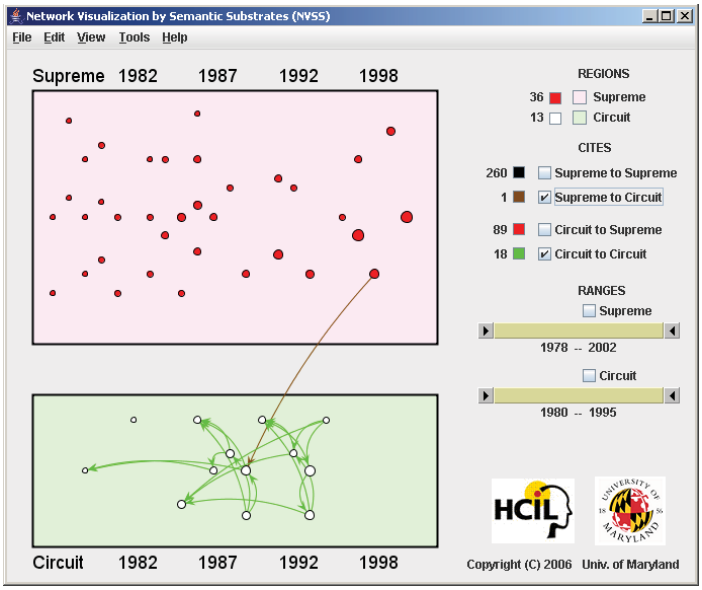
(Abb. 11: Die TextArcOberfläche - Paley, 2002)

TextArc stellt alle Textzeilen in einer Spiralform dar. Häufig auftretende Begriffe werden in der Mitte der Spirale platziert. Die Auswahl eines dieser Begriffe zeigt an, in welchen Zeilen der Begriff auftritt und mit welchen häufig auftretenden Begriffen Beziehungen bestehen. So taucht im Text 'Alice im Wunderland' von Carroll etwa der König fast ausschließlich nur mit der Königin auf, da er lediglich eine Randfigur darstellt. Auch auf den entsprechenden Textabschnitt kann direkt zugegriffen werden.

4.2.3 Visualisierung von Zitations-Beziehungen

Ein letztes Anwendungsgebiet stellen Zitations-Beziehungen dar, also wie Autoren aufeinander verweisen oder sich gegenseitig zitieren. Anhand dieser Daten kann die Wichtigkeit von Autoren oder Themen dargestellt werden. Eine populäre Anwendung stellen Web-Ranking-Algorithmen dar, da sich Web-Dokumente durch Hyperlinks letztendlich gegenseitig zitieren (Hearst 2009, S. 286).

Das **NVSS System** visualisiert, wie sich Gerichte in den USA gegenseitig auf Urteile beziehen (vgl. Abbildung 12). In diesem Beispiel zitierten 89-mal Gerichte auf einer unteren Eben Instanzen der gleichen Ebene. Die Zitierung einer untergeordneten Instanz durch ein übergeordnetes Gericht fand hingegen nur einmal statt (vgl. 'Cites'), (Aris & Shneiderman 2006, o.S.).



(Abb. 12: NVSS Oberfläche - Aris &Shneiderman 2006, o.S.)

Mittels Linien kann visualisiert werden, welche Gerichte genau betroffen sind. Hier werden beispielhaft die Beziehungen innerhalb der untergeordneten Instanzen dargestellt, sowie die Zitierung der übergeordneten auf untergeordnete Instanzen (Aris & Shneiderman 2006, o.S., Abbildung 12).

(Abb. 13: PaperLens Oberfläche - Lee et al., 2005)

Während das NVSS System speziell für derartige Einsatzszenarien konzipiert wurde, geht das **PaperLens System** darüber hinaus und ist vielfältiger einsetzbar. Das PaperLens System kann anhand einer Dokumentenkollektion unter anderem die Beliebtheit von Themen vergleichen (vgl. Abbildung 13, A) oder die Top 10 der zitierten Autoren (vgl. Abbildung 13, B).

5 Einsatzmöglichkeiten und Grenzen

Die vielfältigen Einsatzmöglichkeiten von Information Visualization in der Textanalyse sind bereits durch die hier vorgestellten Anwendungsformen deutlicher geworden. Die unter Punkt 4.2. aufgestellten Ziele sind durch diese Systeme erreichbar, jedoch müssen gewisse Einschränkungen beachtet werden.

Das **TAKMI** System aus Punkt 4.2.1 wurde 1998 in einem Call Center eingesetzt und ermittelte, dass Fragen rund um Windows 98 zu dieser Zeit am Bedeutsamsten waren. In diesem Fall bereitete das Center Antworten zu häufigen Fragen vor und stellte diese Informationen auf der eigenen Webseite zur Verfügung, um so die Zahl der Anrufer zu verringern und den Mitarbeitern das individuelle Erstellen einer Antwort zu diesem Thema zu ersparen. Ein Rückgang der Anrufe zu diesem Thema konnte in der Folgezeit verzeichnet werden. Ferner ließ TAKMI in Rückschlüsse über den Lebenszyklus eines Produktes zu. Dies führte zu einer Entlastung der Mitarbeiter und verbesserte die Qualität ihrer Ratschläge (Nasukawa & Nagano 2001, S. 972 ff.).

Das **Concordance** System in Punkt 4.2.2. kann beispielsweise eingesetzt werden, um Duplikate wie auch Beinahe-Duplikate zu finden, da hier nicht nur Wörter, sondern auch das konkrete Umfeld angezeigt werden.

Selbst ein System wie **Wordle** aus Punkt 4.2.3., dass eher als Spielerei bekannt ist, kann sehr sinnvoll eingesetzt werden, zum Beispiel um häufige Wiederholungen zu erkennen und diese sprachlich besser zu umgehen.

Die Erstellung der Wordle-Wolke basierend auf einer früheren Fassung dieser Arbeit machte deutlich, dass der Begriff 'beispielsweise' sehr häufig verwendet wurde (vgl. Abbildung 10). Die Arbeit wurde entsprechend angepasst und diese Wiederholung gezielt umgangen.

Die Einsatzmöglichkeiten und -szenarien von Information Visualization in der Textanalyse sind vielfältig und sinnvoll. Ein gutes Visualisierungssystem lässt Zusammenhänge erkennen und bereitet diese grafisch so auf, dass der Benutzer relativ einfach Rückschlüsse ziehen und fundierte Entscheidungen treffen kann.

Jedoch ist jedes System immer auch von seinem Benutzer abhängig. Selbst das bereits angesprochene automatisierte Data Mining erfordert Aufmerksamkeit in Form von Wartung. Darüber hinaus sind Rechner auf dem heutigen Stand aber nicht zu den gleichen Denkleistungen in der Lage wie Menschen, welche Entscheidungen immer

aus dem Kontext eigener Erfahrungen, ihrer Umwelt, sowie den Gegebenheiten heraus treffen. Systeme sind hierzu nicht in der Lage. Insbesondere das Fällen von komplexen Entscheidungen sowie das Erkennen komplexer Zusammenhänge ist daher eine Aufgabe des Menschen, die ihm ein Visualisierungssystem auf dem heutigen Stand der Technik nicht abnehmen kann.

6 Fazit und Ausblick

Die vorliegende Arbeit gibt einen Überblick über die Aufgaben und den Sinn der Textanalyse als Teil des Data Minings und stellt Information Visualization als geeignetes Instrument vor, um die Textanalyse zu unterstützen.

Die Kombination dieser beiden Techniken bietet enorme Potentiale und kann Zusammenhänge aufdecken, die vorher nicht bekannt waren. Entscheidungen werden so besser, nachhaltiger und fundierter getroffen.

Um die Ergebnisse von Visualisierungssystemen zu testen, müssen jedoch geeignete Evaluationssysteme entwickelt werden. Momentan besteht Evaluation größtenteils noch darin, die erzielten Ergebnisse qualitativ selbst zu beurteilen und durch Trial-and-Error zu verbessern. Um eine objektive Beurteilung zu ermöglichen, müssen entsprechende Tests entwickelt werden. (Pickett & Grinstein 2002, S. 83).

Desweiteren existiert kein System, dass ein Datenbankmanagementsystem mit vollem Funktionsumfang und ein komplettes Visualisierungstool integriert, da bisher immer nur einer dieser beiden Faktoren als Basis dient und der weitere Faktor als Add-On oder Ähnliches bereitgestellt wird. Die Integration eines Systems mit beiden Faktoren auf Augenhöhe könnte daher noch weitergehende und tiefergreifende Analysen erlauben und würde das derzeit noch vorherrschende parallele Verwenden zweier Systeme redundant machen (Wierse 2002, S. 123 f.).

Einige sehr umfangreiche und weit entwickelte Systeme wie JIGSAW oder TAKMI zeigen bereits, welche Potentiale Information Visualization in der Textanalyse birgt und wohin der Trend geht. Der Fokus künftiger Entwicklungen sollte daher weiter darauf liegen, noch gezielter Zusammenhänge finden zu können.

Gleichzeitig darf die **Visualisierung** aber nicht vernachlässigt werden, da sie den Weg zur Erkennung von Zusammenhängen überhaupt erst ebnet. TextArc oder Wordle bieten hier bereits heute sehr einfache, dennoch ansprechende und überschaubare Oberflächen, während beispielsweise JIGSAW aus Gründen der Komplexität eine Einarbeitungsphase benötigt.

Daher sollten derartige Systeme neben dem Fokus auf noch besserer Darstellung von Informationen auf gleicher Ebene die Weiterentwicklung der Visualisierung betreiben.

Literaturverzeichnis

Aris, Aleks und Shneiderman Ben (2012): NVSS: Network Visualization by Semantic Substrates. University of Maryland. Online verfügbar unter http://www.cs.umd.edu/hcil/nvss/, zuletzt geprüft am 09.09.2012.

Brockhaus AG (2012): Text. Online verfügbar unter http://www.brockhaus-enzyklopaedie.de/be21_article.php.

Fayyad, Usama und Uthurusamy, Ramasamy (1996): Data mining and knowledge discovery in databases. In: Commun. ACM, 39, 11. New York, S. 24–26.

Feinberg, Jonathan (2011): wordle. Online verfügbar unter http://www.wordle.net/, zuletzt geprüft am 09.09.2012.

Feldman, Ronen (2004): Text Analytics: Theory and Practice. Online verfügbar unter http://www.ir.iit.edu/cikm2004/tutorials.html#T2, zuletzt geprüft am 09.09.2012.

Feldman, Ronen und Sanger, James (2006): The Text Mining Handbook. Advancad Approaches in Analyzing Unstructured Data. Cambridge.

Ferstl, Otto K. und Sinz Elmar J. (2006): Grundlagen der Wirtschaftsinformatik. München.

Gee, Alexander G. und Light John (2002): Document Mining and Visualization. In: Grinstein, Georges G.; Wierse, Andreas und Fayyad, Usama (Hg.): Information Visualization in Data Mining and Knowledge Discovery. London, S. 345–353.

Görg, Carsten; Liu, Zhicheng; Parekh, Nee; Singhal, Kanupriya und Stastko John (2007): Visual Analytics with Jigsaw. Georgia.

Grinstein, Georges G.; Wierse, Andreas und Fayyad, Usama (Hg.) (2002): Information Visualization in Data Mining and Knowledge Discovery. London.

Grinstein, Georges G. und Ward Matthew O. (2002): Introduction to Data Visulization. In Grinstein, Georges G.; Wierse, Andreas und Fayyad, Usama (Hg.): Information Visualization in Data Mining and Knowledge Discovery, S. 21–46.

Hearst, Marti A. (1999): Untangling text data mining. In: ACL '99 Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics. Stroudsburg, S. 3–10.

Hearst, Marti A. (2009): Search User Interfaces. Cambridge.

Jung, Volker (1998): Integrierte Benutzerunterstützung für die Visualisierung in Geo-Informationssystemen. Darmstadt.

Lee, Bongshin; Czerwinski, Mary; Robertson, George und Bederson Ben (2012): PaperLens. University of Maryland. Online verfügbar unter http://www.cs.umd.edu/hcil/paperlens/, zuletzt geprüft am 09.09.2012.

Nasukawa T. und Nagano T. (2001): Text analysis and knowledge mining system. In: IBM System Journal (40, 4), S. 967–984.

Paley, Bradford (2002): TextArc. Online verfügbar unter http://www.textarc.org/, zuletzt geprüft am 09.09.2012.

Pilote, Michel und Fillion Madeleine (2002): Character-Based Data Visualization for Data Mining. In Grinstein, Georges G.; Wierse, Andreas und Fayyad, Usama (Hg.): Information Visualization in Data Mining and Knowledge Discovery. London, S. 103–120.

Pickett, Ronald M. und Grinstein, Georges G. (2002): Evaluation of Visualization Systems. In Grinstein, Georges G.; Wierse, Andreas und Fayyad, Usama (Hg.): Information Visualization in Data Mining and Knowledge Discovery. London: Academic Press, S. 83–86.

R. J. C. Watt (2011): Concordance. Dundee. Online verfügbar unter http://www.concordancesoftware.co.uk/, zuletzt geprüft am 09.09.2012.

Simmons, Andrew; Moran, Dan und Chowla, Peter (2006): The UK Interdependence Report. How the world sustains the nation's lifestyles and the price it pays.

SPSS Inc. (2008): Mastering New Challenges in Text Analytics.

Ullenboom, Christian (2011): Java ist auch eine Insel. Bonn.

Uramoto, N.; Matsuzawa, H.; Nagano, T.; Murakami, A.; Takeuchi, H. und Takeda K. (2004): A text-mining system for knowledge discovery from biomedical documents. In: IBM System Journal (43, 3), S. 516–533.

Wierse, Andreas (2002): What Can Visualization Do for Data Mining? In: Grinstein, Georges G.; Wierse, Andreas und Fayyad, Usama (Hg.): Information Visualization in Data Mining and Knowledge Discovery. London, S. 123–124.

A Survey of Personalization Techniques for Online Search and Recommender Systems

Ralf Strobel

Otto-Friedrich-Universität Bamberg
ralf-michael.strobel@stud.uni-bamberg.de

Abstract. The World Wide Web has long outgrown human ability to browse for information without aid, so that automated search and recommendation systems have become indispensable and omnipresent. Personalization represents the next stage of evolution for these automated search assistants. By analyzing user input over time, a system can learn about the individual needs and preferences of each person and generate more suitable customized results. This paper aims to be a comprehensive introduction to common personalization techniques, as well as an overview of current research trends. Chapter 1 presents different example applications of personalization. Chapter 2 compares strategies for user data collection while Chapter 3 illustrates how to coalesce this raw feedback into user models and generate custom recommendations. Finally, Chapter 4 addresses current practical and ethical challenges for web personalization.

Keywords: Personalization, web search, search engines, recommender systems, implicit feedback, user modeling, collaborative filtering, matrix factorization.

1 Introduction

The World Wide Web has grown exponentially, ever since it became a mainstream phenomenon in the mid 1990s. Rapidly decaying prices for bandwidth, storage and processing power have given rise to large-scale online services, focused on media distribution, e-commerce and user-generated content. A downside of this fast technological evolution is the increasing "information overload" of users [30, p. 2]. The vast amount of content available is making it more and more difficult to find the most relevant information, most suitable products, or most enjoyable media items.

Search engines, using automated web crawlers, proved able to keep up with the growing number of web pages. However, early ranking mechanisms, based only on term matching and page popularity, now frequently fail to identify relevant pages as both web content and user interests have become more complex and diverse (see 1.1).

Personalized search is an attempt to provide more suitable results to users by incorporating background information about their interests and previous actions. The same principles can be applied to filter incoming content and create customized news streams (see 1.2) or to make automated recommendations about other existing content items a user might also enjoy (see 1.3).

1.1 Personalization in Ad Hoc Search

Ad hoc web search, as provided by Google, Yahoo or Microsoft Bing, continues to be the most popular method of information retrieval for the majority of internet users. The input interface can consist of as little as a single line text field to enter space-separated search terms. The output is a list of web pages which contain the specified terms, ranked by relevance to the current query as well as general page popularity. All major web search engines also support advanced syntax rules, such as boolean operators and multi-word phrases. However, field studies have shown that these options are only used in a minority of searches (typically less than 10% of queries) and that the average number of terms per query usually lies in between 2.5 and 3.5 across various search services [27] [37].

This weak specification of search queries is regarded as the main reason why ad hoc web search engines often display poor success rates. Coyle and Smyth, 2007 analysed 12 000 Google searches from more than 70 software company employees and found that only 48% of search sessions resulted in at least one result being clicked [9]. They increased this rate to 62% by highlighting results which co-workers had found helpful in similar search situations. In a previous study, the team had already established that there is a significant overlap of queries as well as result utility within a “search community”, i.e. people who share similar information needs [34].

Dou et al., 2007 analysed search engine logs and simulated the effects of five different personalisation algorithms (user-level and group-level click feedback ranking; long-term and short-term abstract user interest profiling) to see if these would have lead to higher ranking positions for clicked results in known queries [12]. The team found that all five techniques outperformed naïve web search for queries which displayed at least one of the following three characteristics:
a) Ambiguity, e.g. “miami dolphins”, which can refer to a football team or a species.
b) High click entropy, i.e. strong variations in the results which were clicked by users.
c) Repetition, i.e. users or user groups executing the same query more than once.

The importance of repeated queries for personalisation is also supported by Teevan et al, 2007, who determined that 40% of investigated Yahoo Search queries were in fact attempts by the respective user to re-find previously discovered web pages [39].

1.2 Personalization in Filter and Notification Systems

To satisfy ongoing information interests of web users, various subscription mechanisms have emerged. Newsletters and RSS feeds were originally often unspecific and simply delivered all new content available on a web site. Now, many large sites such as YouTube begin to offer feeds which can be customized to contain items from contexts each individual user is specifically interested in. Google Alerts is a popular generic service, which allows users to register standing web search queries and receive newly discovered matching pages periodically via email.

So far, all of these notification system require active subscription and maintenance by the recipient. Statistics show that only a minority of expert users make use of such features. In a small survey among Google staff members, Yang and Jeh, 2006 found

that only 27% use their company's own alert system – although more than 60% could name search queries for which they would appreciate notifications whenever new results became available [43]. Explanations commonly included laziness, as well as dissatisfaction with the quality of results delivered by the current alert system.

Yang and Jeh subsequently implemented an algorithm to automatically identify relevant standing queries for users from their individual search query history. Results matched very well with the user's explicit identification of their continuous interests.

Das et al, 2007 had the similar goal of generating personalized Google News streams [10]. They analysed click logs to discover topic interests of individual users. During an evaluation period of six months, their customized news listings received 38% more clicks than the baseline control using general popularity ranking.

1.3 Personalization in Recommender Systems

Search and filter systems help users to discover content they are explicitly interested in and actively looking for. Discovery of new unrelated content and new interests still occurs mostly through social and public channels, like recommendations by friends, blogs or forums. However, since the mid 1990s, researchers have also been looking into ways of presenting users with suitable automated recommendations [30, pp. 2-3].

In the recent years, this idea received further attention from companies behind large commercial web sites such as YouTube or Amazon, who have a strong interest in generating page views. Popular forms of presentation include customized start pages with product or content recommendations for each user, as well as contextual recommendations of similar items on every content page the user visits. (The implementation of such systems is presented in detail in chapter 3.)

Other sites even offer automated content recommendations as their main service to customers. MovieLens, for example, suggests films a user might also enjoy, based on ratings given to films he already knows. Additional research into the subject also originates from the field of targeted advertisement, as offered by Google AdSense, which largely relies on the same principles of user profiling.

In 2006, Netflix, currently the world's largest movie rental and video-on-demand service, challenged the computer science community to develop a system which could predict movie ratings by their customers at a root-mean-square error of 0.85, which was 10% less than their proprietary system [4]. Netflix released a data set, comprised of more than 100 million movie ratings from 500 000 anonymous users. Based on this training data, the competing algorithms had to predict a second set of 3 million ratings from the same users, which was not available to the scientists. In 2009, a joint research group between AT&T Labs and Yahoo Research was awarded a one million dollar reward for their “BellKor” solution, which was the first to produce predictions at the desired level or accuracy [21].

2 User Data Acquisition

The effectiveness of personalized search and recommendation systems depends largely on the quality of available user profiles. Existing methods to acquire information about users can generally be divided into two categories: Explicit feedback methods rely on the users to actively specify their preferences through selection or rating interfaces. Implicit feedback methods passively monitor user behavior to discover evidence of content preference.

2.1 Explicit Feedback

One of the most common forms of explicit user feedback is per item rating on a distinct scale, for example zero to five stars, as implemented by Amazon or MovieLens. Social news platforms like Digg or Facebook often implement an even simpler binary preference score in the form of a “like” toggle-button.

An alternative (or complementary) approach is to allow users to create explicit profiles by specifying their interests and orientations through selection interfaces, for example a field of check boxes to select content categories the user is interested in. Demographic details such as age, gender and location are also often part of profiles. This information is specifically interesting for simple rule-based recommendations as well as targeted advertisement.

Research shows that predictions based on explicit feedback outperform those derived from indirect metrics in the majority of cases [20, p. 146]. Annika Waern, 2004 investigated hybrid solutions and found that adaptive news filtering consistently delivered better results when initialized with an explicit user profile and then trained using implicit feedback, compared to using only the implicit metrics [42].

On the other hand, Waern's comparison between implicit-only and explicit-only filters showed that implicit profiling did in fact match and even surpass the prediction quality of explicit profiles, when large amounts of training data were available. This demonstrates a general advantage of implicit feedback, namely that there is usually much more data to work with, compared to explicit input. The initial advantage of more valid explicit data will often diminish over time, as more implicit metrics become available. This effect is further increased by the fact that users are often too lazy to maintain their profiles, so that their preferences may be out of date [26, p. 93].

It should further be noted that in the above study, participants were required to create an extensive profile for themselves. Real users are commonly reluctant to create and maintain personal profiles. The same can be said about their willingness to explicitly rate content, even if it requires only a single click. This becomes obvious, for instance, when looking at YouTube videos, where the number of views commonly exceeds the number of feedback votes by a factor of more than 100 (Table 1).

The relatively low number of explicit responses additionally makes such feedback systems prone to abuse by attackers who wish to artificially inflate the score of their own content or harm scores of competitors. If only few genuine votes exist, as is for example often the case with Amazon products, it also requires very few forged votes to dominate the total score (see further discussion in section 4.1).

Table 1. Page views and explicit feedback votes for most watched YouTube videos of 2011. (Data retrieved from counters on respective pages, July 23, 2012.)

Video ID	Views	Votes	Views/Vote
nGeKSiCQkPw	112270014	460365	243,87
GI6CfKcMhjY	85311355	474631	179,74
QH2-TGUlwu4	80724753	680563	118,61
_JmA2ClUvUY	70498351	198569	355,03
khCokQt--l4	69812599	499497	139,77

2.2 Implicit Feedback

Page views are the most widely used metric for implicit relevance feedback. The assumption is that users will only look at content they are interested in. In reality, this may not always be true, but it is generally assumed that the "noise" from irrelevant page views does not hurt overall results significantly [16, p. 221].

Fox et al., 2005 additionally investigated more than 30 other implicit signals in web search behaviour, to determine which were most correlated with explicit relevance feedback [14]. They found that page view duration and page exit type (close browser, back to search results, new query...) were the strongest indicators of user preference. This demonstrates how analysis on a user session level can provide valuable extra information compared to simple view counts on a page level. More complex session analysis has even been used to provide personalized navigation in multiple studies, such as Mobasher et al., 2001, by matching the current user activity to common page transition patterns [25].

To determine topical interests of users, search queries also represent a key source of information. Yang and Jeh, 2006 investigated which query patterns were most likely to indicate a long-term interest of the user [43]. Their interest score algorithm derives user preferences from query logs, by taking into account the numbers of query refinements, number of clicks on results, as well as similarities with older searches.

However, many simple forms of personalization do not even require detailed user preference data. Typical cases are language and location filters for search results. The necessary data can be obtained directly from the HTTP request headers, sent by the browser [26]. The "accept-language" header gives explicit information about the primary interface language of the visitor, while the "user-agent" header indicates the browser and device type used for access. The user's geographic location may be available via GPS or can be approximated from the remote IP address. Lastly, trivial context parameters like the current time and date can also be utilized to improve results, as some content may be searched for more often during office hours than in the evening, or more popular around specific holidays or anniversaries.

Other forms of simple rule-based personalisation operate based on demographic data, such as user age, gender, marital status, education, income level, religious or political orientation. This is often the case for online advertisement, where companies can choose which target audience they want their products to be presented to. If the

necessary user data is not available from explicit profiles, it can be inferred from implicit signals using statistical data mining methods [1, p. 228].

The principle behind this “inferential context” is to compare content viewed and searched for by the user with known statistical patterns of content preference for different demographics such as age groups and genders. The resulting profiles are only probabilistic, but achieve more than adequate precision on average for their main fields of application, such as targeted advertisement.

3 Implementation

The general purpose of every personalized search or recommender system is to identify “items” (products, pages, content...) which are of interest to the current user [30, p. 8]. Therefore, a key aspect during the design and implementation of such systems is to define models for both items and users. These data structures then become the basis of a similarity measure algorithm, which determines for each combination of user and item the likelihood of a match between item properties and user preferences. The output is then commonly a simple list of items, ranked by this similarity scale [30, p. 2].

This chapter will discuss the three most common approaches to match items against user preferences. The first relies solely on item-to-item comparison and tries to match new items against items already confirmed to be preferred by the user. The second method attempts to compile these known user preferences into an abstract profile and then matches items against it. The third approach, known as “collaborative filtering”, is to match users against other users with similar preferences and obtain recommendations from their feedback.

The first and third method both belong to the group of memory-based strategies, which operate on large numbers of often pre-computed entity relationships and are implemented strongly based on database management systems. The second approach is a model-based strategy, which focusses more on computation-intensive analysis in realtime to make complex predictions. The following subsections will investigate both strategies separately, to present their strengths and weaknesses. However, it should be mentioned that they are by no means mutually exclusive and many existing recommender systems in fact rely on hybrid algorithms for increased performance.

3.1 Item-to-Item Matching

The most basic form of recommender system is simply a list of related items, displayed in context of every item the user chooses to examine. This is often used by content-rich web sites such as YouTube or Amazon to enable explorative navigation. This approach uses no explicit user profiles, but simply assumes that the visitor is interested in what he chooses to look at and suggests more of the same kind. The only data structure necessary for this method is the item model, which is a set of attributes that describes each item and allows for comparisons between items.

For weakly structured items, such as texts, the model is usually a set of contained words, extracted by a text tokenizer, or keywords associated by semantic analysis [23, p. 75]. The similarity measure between each two items is then generated by the same kinds of algorithms employed by web search engines, most commonly a vector space model, comparing term frequency and inverse document frequency [23, p. 81] [2, pp. 41-42]. Discriminative algorithms (categorization, clustering) are also in use for specific applications [2, pp. 48-66].

Well-structured items like products or jobs have common attributes (e.g. for mobile phones: screen size, operating system, battery life...). They can be compared in their specific vector space, with similar items defined by a high number of identical values.

A different approach to similarity detection is not to analyse the content itself but user interaction with content. For instance, Amazon regards two items as similar if they are often purchased in the same orders [22, p. 78]. The similarity measure used in such correlation-based systems is usually the Pearson correlation coefficient, which measures the tendency of users to treat two items similarly [20, p. 162].

The advantage of this approach is that it does not face semantic gap problems, i.e. the inability of automated systems to analyse and compare the meaning of content items (typically images, sounds or videos). Disadvantages are its dependence on available usage metrics, as well as false-positive correlation detection due to random user behaviour. Worst case computation complexity for correlation-based similarity is also greater, at $O(n^2m)$, with n as the number of items and m as the number of users, as each item pair has to be compared for each user. However, the sparsity of item-user relationships reduces this effort to roughly $O(nm)$ in practice [22, p. 79].

In many applications, items do not change their properties over time or only do so at a very slow rate. For this reason, item-to-item recommender systems commonly pre-compute similarities and store them in a similarity matrix, implemented in the form of a database table, linking each item to a set of most similar other items [22, p. 79]. This reduces the computation complexity during output generation to $O(1)$. The complexity for the insertion of a new item is $O(n)$, with n as the number of existing items, as each has to be compared to the new addition. Adequate pre-filters can further reduce this cost.

Item-to-item matching can also be used to generate true personalized recommendations. This requires a very basic user model, which is simply a list of items the user has liked before. If these item-user relationships are stored in a database table and a similarity table is available as described above, then generating a list of personal recommendations can be achieved directly by a join query, which compiles a list of similar items based on the user's previous preferences [22, p. 79].

A similar approach is also applicable to web search. In this case, the user model is a weighted list of keywords the user has often searched for before. Each new search query is then expanded by these known preferred terms in order to generate a personalized result list [23, pp. 82-85] [35, p. 586].

3.2 Item-to-User Matching

Basing recommendations on general item similarity is a very crude simplification for many applications. It does not take into account that different users may prioritize different attributes when comparing items. In a movie recommendation system, for example, one user may like all movies on the same subject and treat them as similar, while another user likes all movies directed by the same person.

In order to generate more accurate recommendations, which take into account the user's individual aspect-preferences, it is necessary to create abstract user models. These models commonly reside in the same attribute vector space as the item. For example: Each movie is described by a genre vector, indicating how much it matches the characteristics of each genre on a scale between 0 and 1. Then each user profile is also a genre vector, indicating how much the user prefers content of each genre on the same scale. The similarity function between an item and a user is therefore simply the dot product of both vectors [19, p. 44].

The main challenge for this approach is that the individual attribute preferences of users are not known explicitly, which is why they are referred to as “latent factors” or “hidden factors”. It is known which items a user prefers (from either implicit or explicit feedback), but not why. This raw user-item feedback can be represented in the form of a matrix with users as lines, items as columns and feedback as inner values. The process of deriving latent factor user profiles from this raw feedback data is therefore called “matrix factorization”. The requirement of such a distinct learning and data aggregation phase, before the system is ready to make predictions, is a key characteristic of model-based strategies.

Matrix factorization is essentially a linear optimization problem. It is assumed that each existing feedback value is the dot product of the corresponding item and user attribute vectors, as described above. While the item vector is fixed, the optimal user vector to explain all known feedback has to be determined. (Note that the matrix is always sparse, as the missing values are exactly the desired preference predictions). This problem can be solved mathematically through singular value decomposition (SVD). However, numerous improved algorithms have been developed to avoid the shortcomings of SVD, such as performance issues and over-fitting for matrices which only contain very little feedback data [20, pp. 151-161].

Advanced systems like the BellKor solution (Netflix) further improve predictions by compensating bias and temporal distortion in feedback data [21] [20, pp. 151-161]. A common bias effects is caused by users who vote extremely positive on every item, which needs to be addressed by input normalization. Temporal effects on the user side include drifts in preferences over time as well as distinction between long-term and short-term interests. Items may also undergo temporal changes, such as decline in popularity over time, or periodic popularity peaks in context of specific holidays. To address these effects, a third dimension has to be added to the feedback matrix, indicating the time of user-item interaction. In user and item models, each vector component is then no longer a single scalar value, but a parametric function of time.

3.3 User-to-User Matching (Collaborative Filtering)

Whenever items must appeal to a user's "taste", for example music or artwork, recommender systems based on item attributes may not be able to deliver satisfying results, as human esthetic perception is difficult to parametrize. An alternative approach is to assume that users who have liked similar items before share the same taste and will also judge future items similarly. This method does not require an explicit item model and is commonly referred to as "collaborative filtering" (CF).

Most CF recommender systems use the k-nearest neighbour algorithm (k-NN) or one of its variations [11, pp. 114-120]. The idea is to identify k "neighbours" for each user, i.e. users with comparable taste. Recommendations are then compiled from the preferred items of neighbours and ranked by their average rating of these items. The value for k can vary between less than 10 and several hundred, depending on quality and performance goals. Systems with high numbers of items generally also require higher k-values to achieve sufficient coverage by neighbour feedback [26, p. 95].

Neighbour associations are commonly built offline and stored in a similarity table. For a system with m users and n items, this process has a worst case complexity of $O(m^2n)$, as each user has to be compared to all others to determine which k of them prefer the most similar item sets. However, average runtime complexity is much lower due to highly sparse user-item preference and low user-user preference overlap.

Compared to recommendation building from item-item similarity, with a complexity of $O(n^2m)$, collaborative filtering is still more expensive, because the number of users is typically higher then the number of items in most systems [20, p. 178]. Neighbours also have to be rebuilt periodically, as user preferences change over time, while item similarity is commonly static. Furthermore, the online process of CF is slower than that of item-item recommendations: While both are essentially a join between a similarity- and the user-item preference table, the CF query entails an aggregation over k separate preference sets.

While content-based recommender systems work instantly, collaborative filtering first requires a critical mass of user interactions to make valid predictions. Even in a well populated system, the introduction of new users and items remains a problem. Each user has to give a sufficient amount of feedback before receiving good suggestions. Each item has to be liked by a sufficient amount of users before appearing in suggestions frequently. Therefore, many systems only use CF in a complementary fashion along with other recommender systems [31, p. 300].

Still, collaborative filtering is one of the most popular recommender system types, not just for its independence from item modelling, but also for the quality of results. One major advantage over content-based and particularly model-based systems is that CF is very insensitive to over-fitting and often produces serendipitous results, i.e. suggestions the user did not expect but still found useful [11, p. 112] [33].

In recent years, the emergence of online social communities has lead to the development of "social filtering" as a new sub-form of CF. In this type of system, neighbours are not assigned by similarity algorithms, but directly taken from the user's explicit friend or contact list. This drastically decreases computation cost and also makes suggestions more intelligible to users, as the system may indicate that an

item was listed because a specific friend liked it. On the other hand, experience has shown that the assumption “we like what our friends like” does not always hold (see section 3.4) [24, p. 288]. Cai et al., 2011 further remark that algorithmic neighbour detection can still be useful in social filtering, as a way to recommend new like-minded people to users [6].

3.4 Asymmetric Matching: Trails, Trust and Dialogues

The techniques described in the three previous subsections make up the bulk of commercial recommendation systems today. They all share *similarity* (essentially a symmetric graph between items or users) as the key source of recommendations. However, in recent years, the fields of artificial intelligence and social science have driven new ideas for recommendation systems, which instead rely on directed graphs.

For personalized navigation assistance, several research groups have used machine learning techniques to discover common user navigation trails on web sites, in order to suggest pages to new visitors who display similar patterns. Taghipour et al., 2007 trained a system from web usage logs by reinforcement learning to predict next page views of uses, based on their preceding actions [38]. On simulated test data, their system displayed significantly superior precision, compared to similarity-based collaborative filtering. Eirinaki and Vazirgiannis, 2005 achieved similar results with an algorithm based on markov models, trained using combined information from web usage graphs and page link structure graphs [13].

Product recommendations have also been shown to benefit from more complex awareness of item-item-similarities. Dialogue- or critique-based systems allow users to iteratively refine the suggestions they receive by specifying which direction they should go into (e.g. “cheaper”, “higher resolution”, “less weight”) [36, pp. 358-365]. The entire graph is typically not pre-built, as it would be very extensive, but partially computed in real time by faceted search.

Research on social filtering has shown that asymmetric graphs can be a more adequate model for interpersonal relationships. Many existing online communities, such as Facebook, work by the model of “friendships” (a symmetric relationship) which form a symmetric “social network” graph. However, “trust” may be a better metaphor to use, whenever the goal is to inform users about new content produced or endorsed by like-minded individuals, as is the case for social news applications.

Trust is an asymmetric relationship, usually limited to a specific knowledge domain [24, p. 287]. One person may trust another regarding movie recommendations, but not necessary vice versa and not regarding any other subject. Trust relationships also don't imply that two people know each other in real life. Especially the trusted person does not even need to be aware of the existence of its follower. Twitter is currently the largest online service built on such asymmetric relationships. Recommender systems using this model are called “trust-aware” or “trust-enhanced” [41]. However, focussed research on this subject has only just begun in the past few years.

4 Critique and Challenges

Consumer studies have shown that online users highly appreciate personalized content presentation, and that it can lead to a significant increase in profit for web vendors and content producers [17, p. 629]. This seemingly obvious win-win situation has given rise to a boom in personalization across the internet. While the previous chapters have focused mainly on the advantages of this development, the following chapter will highlight two common points of criticism regarding the legal and cultural consequences of an increasingly personalized web.

4.1 Security Issues

As shown in chapters two and three, the key to good personalization lies in the acquisition of personal information on users. Many online services are now collecting and aggregating large amounts of user data, far beyond the level of traffic analysis, in order to offer “smarter” web sites. First concerns that these practices could be seen as a violation of privacy were voiced even before the existence of the world wide web.

In a 1989 conference paper, Alfred Kobsa already proposed a scenario in which data from a travel booking system was leaked to an insurance company and used for unwanted targeted advertising towards people who like to visit potentially dangerous countries [18]. He concluded that handling of user-related data was to become an important issue in the context of national privacy legislation.

Twenty years later, these theoretical concerns have turned into the very realistic fears of many internet users. Publicly known cases of security breaches among internet companies, including major players like Google, have made users aware of the risks of carelessly providing personal information online [3]. Commonly expressed fears include the use of private information in the context of employment or credit background checks, price discrimination based on alleged product interest, aggressive telemarketing, as well as surveillance by government or law enforcement agencies [17, pp. 628-629].

Particularly the last aspect is quite real, not only in nations like China, but also in western countries, most of which have already authorized their agencies to request private user data from companies in order to prosecute criminals. According to the Google Transparency Report of 2011, during one year the company released information from more than 23 000 user accounts to U.S. authorities, and data from nearly 3800 user accounts to German authorities [15].

Across numerous surveys, between 80% and 90% of polled internet users stated concerns about the privacy of their personal information and that they had refused to provide data to a web site before [17, pp. 632-636]. On the other hand, the studies also indicated little correlation between a person's stated concerns and actual behaviour. Specifically, most users were readily available to submit personal data whenever they saw a valid purpose to it, like in the case of online shopping.

A common technique to provide basic data privacy in online systems is the use of pseudonymous user names instead of identifying personal details like real name or address [17, p. 652]. The effectiveness of this solution is diminished, however, by the

fact that most online systems still require a valid email address, which may be traced back to a real person using other data sources. Identification may even be possible from the gathered user preferences alone, if they can be matched against other known profile data from the same person [17, p. 653].

Another suggested approaches to increased privacy is to store personal data in the web client or encrypted in a distributed peer-2-peer network, rather than on centralized servers [17, pp. 654-656]. However, these techniques are not applicable to most systems which rely on heavy data processing (see chapter 3).

In lack of perfect technological solutions, online privacy still relies on behavioural regulation of businesses. Legal requirements force companies to declare their use of personal data and ask for user consent in advance, as well as offer opt-out mechanisms for users who wish for their data to be deleted [17, pp. 647-651]. Thus, the responsibility for the privacy of their data mostly rests with the users themselves.

Spoof Detection. A different thread to the security of search and recommender systems is the manipulation of feedback data by attackers, for example to artificially boost the rating of an item they own (product push), or decrease the rating of competitors (product nuke) [41, p. 656]. The most common type of attack, known as “profile injection”, generates a large amount of user profiles, often in a partially automated process, which all rate the targeted items high or low, while also featuring a number of votes on “filler items”, either random or based on actual people's votes, to disguise the attack profile [5, pp. 810-814]. Concrete strategies and sophistication of such attacks may vary depending on the attacker's knowledge of the system.

There is an ongoing arms raise between attackers and developers of detection algorithms, which seek to identify possibly spoofed ratings through statistical or semantic analysis. In essence, this boils down to a binary classification problem (spam or no spam) [5, p. 820]. It has further been shown that model-based recommender systems are generally more robust to attacks than memory-based systems, as the latter rely strongly on item or user similarity, which can be easy to imitate for attackers [5, pp. 828-829].

4.2 Overspecialization

The general premise of recommender system is that users have a defined set of interests, and that the goal is to find items matching these preferences. But this view may be an over-simplification, compared to some of the real world models for recommender systems, like a friend's advice or suggestions by a movie-store clerk.

Pattie Maes, Gavin Potter and others summarize the issue in an Interview with the New York Times [40]: “You don’t want to see a movie just because you think it’s going to be good”, Maes says, and Potter adds: “If you go with the movie-store clerk, you will get more unpredictable but potentially more exciting recommendations.” Every person probably knows from experience that there can be many reasons to go see a movie besides individual interest – for example peer pressure because friends have already seen it, or simply the desire to try out something new.

Algorithmic attempts have been made to boost serendipitous and diverse results in search and recommender systems. The common approach is to define that items which receive a high ranking score must be both relevant according to the core search criteria, while at the same time being as different as possible from items the user has already seen [32, pp. 286-289].

Evaluation of serendipity in a practical user study is likely to be difficult, however. The typical evaluation metric of recommender systems is prediction accuracy, which is easy to calculate as the error between predicted and real item ratings, or one could say the absence of false-positive results [32, pp. 273-277]. A system with high serendipity, on the other hand, would be defined by a low number of false-negative results. But in order to calculate this metric, it would be necessary for the test users to investigate and rate every available item, in order to detect those which the system did not recommend although they were in fact of interest to the user.

Redundancy. The principle described above for a user's taste in items can also be translated to information needs. Zhang et al., 2002 have suggested that an adaptive search and filtering system should ideally return items “that are similar to previously delivered relevant documents in the sense of having the same topic, but also dissimilar (...) in the sense of containing new information.” [44] These goals are of course contradictory in many systems, especially for web search engines with their high percentage of re-finding queries (see section 1.1). Redundancy filtering is more likely to play an important role in sophisticated search assistant systems, which maintain an extended model of the user's information state and needs.

Radicalization. A potentially dangerous cultural side-effect of search personalization has recently been pointed out by Eli Pariser: Since adaptive search and filter systems tend to deliver more items similar to those a user has preferred before, they are also more likely to highlight results which represent the users pre-existing knowledge and opinions, rather than contradicting facts.

Pariser has dubbed this phenomenon of users being locked in an environment based on their own opinions a “filter bubble”. “What I seem to like may not be (...) what I need to know to be an informed member of my community or country.”, he says [29, p. 18]. This phenomenon may not be new, as political or religious extremists have always had a tendency to select publications which further validate their existing opinions. But online personalization may support this process of self-radicalization on a whole new level, by taking the decision away from the user in the first place.

5 Conclusions and Outlook

Recommender systems are already quite successful and prevalent wherever the goal is to anticipate the individual “taste” of a user. The Netflix Prize challenge demonstrated how sophisticated hybrid solutions between model-based and memory-based methods can deliver high quality personal recommendations at feasible computation times.

Pandora Internet Radio is another such example where a system based on complex user preference modelling (in this case music) was able to create a very popular personalized online service [28].

The great challenge to all existing recommender systems is still the semantic gap. Model-based approaches only work well when items can be easily reduced to and compared using descriptive parameters. That is why sites with high numbers of diverse items such as Amazon or YouTube still rely mostly on less complex item-to-item comparisons (tags, usage correlations). Simple listings of related items are already a widespread practice across many web sites and have proven extremely useful for explorative navigation. Some sites like Amazon try to offer actual personalized recommendations, but their predictive quality is still often dissatisfying.

Lack of semantic context is also a problem for the personalization of search engine results. Research has proven that taking into account the past search behaviour of users can in fact improve both retrieval and precision (see 1.1). However, the effect is only significant for ambiguous or repetitive queries, which can easily be improved by the user himself through reformulation [16, p. 232]. Smart personal news filters (see 1.2) have shown promising results in experimental studies but have not been implemented on a large scale, possibly due to their height computational cost.

An interesting development for the upcoming years may be the integration of social web technologies into search engines. Surveys indicate that around 90% of internet users already engage in cooperative search, for instance by looking over the shoulder of another search engine user and suggesting query reformulations, or by sharing queries and results via email or chat [35, p. 588]. The experiment of Coyle and Smyth, 2007 demonstrates how inbuilt facilitation of result sharing can strongly improve search effectiveness [9]. The emergence of trust-aware recommender systems (see 3.4) and social search interfaces may foster the spontaneous formation of what researchers have dubbed “micro search communities” [35, p. 607] to reduce the current isolation and redundant activities between search engine users.

Looking into the far future, the big goal will be to gradually bridge the semantic gap and develop “intelligent personal assistants”, which are able to truly support a person in complex information retrieval tasks. These artificial intelligence systems will need have an abstract awareness of what the user is currently trying to achieve, which information he already has and which he still requires. They will also need to access and integrate user information from more sources than ever, including personal files, contacts, communication, schedules and web search history [8]. A first glimpse into the potential of this technology was given by the CALO project between 2003 and 2008 [7]. Part of its findings later became part of Apple's “Siri” service.

It seems logical that all advances towards more intelligent personalization will always require even more user information to be collected and processed. Debates regarding privacy concerns are therefore sure to intensify further throughout the next years. Regarding Eli Pariser's concerns about filter bubbles (see 4.2), future search assistants should hopefully be intelligent enough to help us cope with daily information overload in a neutral way, without taking over thinking for us entirely.

References

1. Adomavicius, G., Tuzhilin, A.: Context-Aware Recommender Systems. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P. (eds.): Recommender Systems Handbook, pp. 217--253. Springer, New York (2011)
2. Amatriain, X., Jaimes, A., Oliver, N., Pujol, J.M.: Data Mining Methods for Recommender Systems. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P. (eds.): Recommender Systems Handbook, pp. 39--71. Springer, New York (2011)
3. Armerding, T.: The 15 worst data security breaches of the 21st Century (2012), http://www.csoonline.com/article/700263.
4. Bennett, J., Lanning, S.: The Netflix Prize. In: Proceedings of KDD Cup and Workshop, http://www.netflixprize.com/assets/NetflixPrizeKDD_to_appear.pdf, (2007)
5. Burke, R., O'Mahony, M.P., Hurley, N.J.: Robust Collaborative Recommendation. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P. (eds.): Recommender Systems Handbook, pp. 805--835. Springer, New York (2011)
6. Cai, X., Bain, M., Krzywicki, A., Wobcke, W., Kim, Y., Compton, P., Mahidadia, A.: Collaborative Filtering for People to People Recommendation in Social Networks. In: Li, J. (eds.): Advances in Artificial Intelligence, LNCS, vol. 6464, pp. 476--485. Springer, Heidelberg (2011)
7. CALO Project, http://www.ai.sri.com/project/CALO
8. Chaudhri, V. K., Cheyer, A., Guili, R., Jarrold, B., Myers, K. L., Niekrasz, J.: A Case Study in Engineering a Knowledge Base for an Intelligent Personal Assistant. In: Proceedings of the 2006 Semantic Desktop Workshop. ISWC, Athens, USA (2006)
9. Coyle, M., Smyth, B.: Information recovery and discovery in collaborative web search. In: Amati. G., Carpineto, C., Romano, G. (eds.): ECIR 2007. LNCS, vol. 4425, pp. 356--367. Springer, Heidelberg (2007)
10. Das, A., Datar, M., Ashutosh, G.: Google News Personalization. Scalable Online Collaborative Filtering. In: Proceedings of the 16th international conference on World Wide Web, pp. 271--280. ACM, Banff, Alberta (2007)
11. Desrosiers, C., Karypis, G.: A comprehensive survey of neighborhood-based recommendation methods. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P. (eds.): Recommender Systems Handbook, pp. 107--144. Springer, New York (2011)
12. Dou, Z., Song, R., Wen, J.: A Large-scale Evaluation and Analysis of Personalized Search Strategies. In: Proceedings of the 16th international conference on World Wide Web, pp. 581--590. ACM, Banff, Alberta (2007)
13. Eirinaki, M., Vazirgiannis, M.: Usage-based PageRank for web personalization. In: Proceedings of the fifth IEEE international conference on data mining, pp. 130–137. IEEE, Washington DC (2005)
14. Fox, S., Karnawat, K., Mydland, M., Dumais, S., White, T.: Evaluating Implicit Measures to Improve Web Search. In: ACM Transactions on Information Systems 23 (2), pp. 147-–168 (2005)
15. Google Transparency Report, http://www.google.com/transparencyreport/userdatarequests/
16. Hearst, M. A.: Search user interfaces. Cambridge University Press, New York (2009)
17. Kobsa, A.: Privacy-Enhanced Web Personalization. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.): The Adaptive Web. LNCS, vol. 4321, pp. 628--670. Springer, Heidelberg (2007)

18. Kobsa, A.: User Modeling in Dialog Systems: Potentials and Hazards. In: AI & Society 4, pp. 214--240 (1990).
19. Koren, Y., Bell, R., Volinsky, C.: Matrix Factorization Techniques for Recommender Systems. IEEE Computer 42(8), 30--37 (2009)
20. Koren, Y., Bell, R.: Advances in collaborative filtering. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P. (eds.): Recommender Systems Handbook, pp. 145--186. Springer, New York (2011)
21. Koren, Y.: The BellKor Solution to the Netflix Grand Prize. Technical report, Netflix Inc. http://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf (2009)
22. Linden, G., Smith, B., York, J.: Amazon.com Recommendations. Item-to-Item Collaborative Filtering. In: IEEE Internet Computing 7(1), pp. 76--80 (2003)
23. Lops, P., de Gemmis, M., Semeraro, G.: Content-based recommender systems: State of the art and trends. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P. (eds.): Recommender Systems Handbook, pp. 73--105. Springer, New York (2011)
24. Ma, H., Zhou, D., Liu, C., Lyu, M., King, I.: Recommender systems with social regularization. In: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, pp. 287–296. ACM, Hong Kong (2011)
25. Mobasher, B., Dai, H., Luo, T., Nakagawa, M.: Effective personalization based on association rule discovery from web usage data. In: Proceedings of the 3rd ACM Workshop on Web Information and Data Management, pp. 9--15. ACM, Atlanta (2001)
26. Mobasher, B.: Data Mining for Web Personalization. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.): The Adaptive Web. LNCS, vol. 4321, pp. 90--135. Springer, Heidelberg (2007)
27. Moukdad, H., Large, A.: Users' perceptions of the Web as revealed by transaction log analysis. In: Online Information Review 25 (6), pp. 349--359 (2001).
28. Pandora Internet Radio, http://www.pandora.com/
29. Pariser, E.: The Filter Bubble. What the Internet Is Hiding from You. Penguin Press, New York (2011)
30. Ricci, F., Rokach, L., Shapira, B.: Introduction to Recommender Systems Handbook. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P. (eds.): Recommender Systems Handbook, pp. 1--35. Springer, New York (2011)
31. Schafer, J.B., Frankowski, D., Herlocker, J., Sen, S.: Collaborative Filtering Recommender Systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.): The Adaptive Web. LNCS, vol. 4321, pp. 291--324. Springer, Heidelberg (2007)
32. Shani, G., Gunawardana, A.: Evaluating Recommendation Systems. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P. (eds.): Recommender Systems Handbook, pp. 257--297. Springer, New York (2011)
33. Sinha, R., Swearingen, K.: Comparing Recommendations Made by Online Systems and Friends. In: Proceedings of the Joint DELOS-NSF Workshop on Personalisation and Recommender Systems in Digital Libraries, pp. 73–78. ERCIM, Dublin (2001)
34. Smyth, B., Balfe, E., Freyne, J., Briggs, P., Coyle, M., Boydell, O.: Exploiting Query Repetition and Regularity in an Adaptive Community-Based Web Search Engine. In: User Modeling and User-Adapted Interaction 14 (5), pp. 383--423 (2004).
35. Smyth, B., Coyle, M., Briggs, P.: Communities, Collaboration, and Recommender Systems in Personalized Web Search. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P. (eds.): Recommender Systems Handbook, pp. 579--614. Springer, New York (2011)

36. Smyth, B.: Case-Based Recommendation. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.): The Adaptive Web. LNCS, vol. 4321, pp. 342–376. Springer, Heidelberg (2007)
37. Spink, A., Wolfram, D., Jansen, B.J., Saracevik, T.: Searching the web. The public and their queries. In: Journal of the American Society for Information Science 53(2), pp. 226--234 (2001).
38. Taghipour, N., Kardan, A., Ghidary, S.S.: Usage-based web recommendations: a reinforcement learning approach. In: Proceedings of the 2007 ACM Conference on Recommender Systems, pp. 113–120. ACM, Minneapolis (2007)
39. Teevan, J., Adar, E., Jones, R., Potts, M. A. S.: Information Re-Retrieval. Repeat Queries in Yahoo's Logs. In: In Proceedings of the 30th Annual International ACM SIGIR Conference. ACM, Amsterdam (2007)
40. Thompson, C.: If You Liked This, You're Sure to Love That (2008), http://www.nytimes.com/2008/11/23/magazine/23Netflix-t.html
41. Victor, P., De Cock, M., Cornelis, C.: Trust and Recommendations. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P. (eds.): Recommender Systems Handbook, pp. 645--675. Springer, New York (2011)
42. Wærn, A.: User Involvement in Automatic Filtering. An Experimental Study. In: User Modeling and User-Adapted Interaction, 14 (2), pp. 201--237 (2004).
43. Yang, B., Jeh, G.: Retroactive Answering of Search Queries. In: Proceedings of the 15th international conference on World Wide Web, pp. 457--466. ACM, Edinburgh (2006)
44. Zhang, Y., Callan, J., Minka, T.: Novelty and redundancy detection in adaptive filtering. In: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 81--88. ACM, New York (2002)

Die Evaluierungsinitiative INEX

Christoph Jungnickl

Otto-Friedrich-Universität Bamberg, Lehrstuhl für Medieninformatik
Christoph-philipp@stud.uni-bamberg.de

Zusammenfassung. Ziel dieser Arbeit ist es, die Evaluierungsinitiative INEX vorzustellen. Hierfür werden die Tracks geschildert, die zeitliche Entwicklung dargestellt, sowie Ziele und wesentliche Ergebnisse & Entwicklungen präsentiert. Die Einleitung fast den Sinn hinter INEX und die Entstehungsgeschichte zusammen. Um sich die Art und Weise wie die Initiative arbeitet besser vorzustellen, werden Beteiligte und der Ablauf von INEX beschrieben. Für ein besseres Verständnis des XML Retrievals wird zudem die Auszeichnungssprache XML kurz erläutert. Aufgrund der Wichtigkeit, wird der Ad-Hoc Track ausführlicher beschrieben. Dazu werden die Document Collections, Topics, das Relevanzkriterium und die Metrics behandelt. Abschließend sollen eine Trackübersicht zu INEX, INEX 2012, ein Fazit und ein Ausblick in die Zukunft die Arbeit abrunden.

Schlagworte: INEX, Teilnehmer, , XML, Ad-Hoc, Document Collections, Topics, Relevanz, Metrics, Tracks, Trackübersicht, INEX 2012

1 Zur Arbeit

Diese Arbeit bedient sich zum Großteil der Quelle „INEX 2002 - 2006: Understanding XML Retrieval Evaluation", geschrieben von Mounia Lalmas and Anastasios Tombros Queen Mary University of London, Mile End Road, London, UK, was an der Stelle ausdrücklich erwähnt werden soll.
Sie fast INEX von 2002 bis 2006 zusammen, was genau die Anfänge von INEX darstellt. Damit wird gleichzeitig auch ein Grundverständnis für INEX geschaffen bzw. hilft zu verstehen, worum es bei INEX geht. Das Kapitel Metrics wurde dabei stark gekürzt, da auch ohne die genauen Formeln ersichtlich wird, wie Evaluierung von XML Mehthoden funktioniert. Bei der Recherche hat sich gezeigt, dass die Quelle sehr gute Qualität hat, da sie wiederum die jährlich erscheinenden Reports zitiert. Hinderlich kann die englische Sprache sein, da man ohne Kenntnis des Information Retrieval Wortschatzes Probleme hat, folgen zu können.

Für die Jahre nach 2006 bis heute wird dann eine Grafik zeigen, in wie viele verschiedene Richtungen die Entwicklung von INEX geführt hat.

2 Einleitung[1]

Herkömmliche Suchmaschinen liefern als Ergebnis ganze, für die Anfrage relevante Dokumente zurück. Die nötige Information muss der Nutzer dann selbst versuchen im Dokument zu finden. Neue Suchmaschinen erledigen den Schritt mit, da sie relevante Teile aus relevanten Dokumenten zurückgeben. Diese Art des Retrievals bezeichnet man als Fokused Retrieval. Hauptziel von INEX ist es Focused Retrieval Methoden zu evaluieren.[2]
Vorstellen kann man sich das zum Beispiel wie einen Grafikkarten-Benchmark, mit dem man die Performance von Grafikkarten evaluiert. Anstatt Grafikkarten werden bei INEX Suchmaschinen, genauer die Methoden dahinter, evaluiert.
Der Name INEX steht für "Initiative for the Evaluation of XML Retrieval". Es wird nämlich explizit die Auszeichnungssprache XML berücksichtigt, genauer die Struktur, welche das XML Format auszeichnet. Diese macht es besonders interessant für IR, da diese berücksichtigt werden kann.
Neben XML wäre Html ein weiteres Format, besser gesagt „Markup Language", wobei sich INEX eben auf ersteres konzentriert.
Die Evaluierung geschieht einerseits anhand von immer größer werdenden Test Collections, welche aus strukturierten Dokumenten im XML Format bestehen und andererseits anhand von standardisierten Evaluierungsmaßnahmen, welche unter anderem aus Scoring Methoden und Themenstellungen, den sogenannten Topics, bestehen. Ein Forum bietet außerdem die Möglichkeit der Kommunikation für Organisationen, wo besonders an Testergebnisvergleich zu denken ist. Wenn man sich die Frage stellt, warum INEX im Jahr 2002 entstanden ist, so lässt sich dies dadurch begründen, dass zunächst das Format XML 1998[3] in seiner ersten Version erschienen ist und dann besonders für digitale Bibliotheken, Produkt Verzeichnissen und allgemein im Internet immer mehr Verwendung gefunden hat. Daraufhin wurden Information Retrieval Methoden entwickelt, die versucht haben, bei Suchanfragen XML Dateien bestmöglich zu finden und zurückzuliefern. Aus dem Verlangen nach der Evaluierung dieser Methoden ist INEX entstanden. Die Notwendigkeit und die Daseinsberechtigung solcher Evaluierungsinitiativen lässt sich auch anhand der Anzahl weiterer Initiativen zeigen. An der Stelle zu erwähnen sind TREC, CLEF und NTCIR. Gemeinsam haben solche unabhängigen Plattformen, dass sie Entwicklern von Information Retrieval Methoden Kosten sparen, welche durch eine eigene Evaluierung entstehen würden - was naheliegend ist, da eine Eigene Entwicklung einer Evaluierung viel Arbeit erfordert. Durch die Standardisierung der Evaluierung, kann man Suchmaschinen auch besser miteinander vergleichen, da immer unter gleichbleibenden Bedingungen getestet wird und keine Retrieval Methoden Vor-, bzw. Nachteile hat. Figure 1. zeigt die Internetpäsenz von INEX 2012, um sich über INEX zu informieren, die nächsten wichtigen Termine im Zusammenhang mit INEX zu erhalte und natürlich auch, um an INEX teilzunehmen.

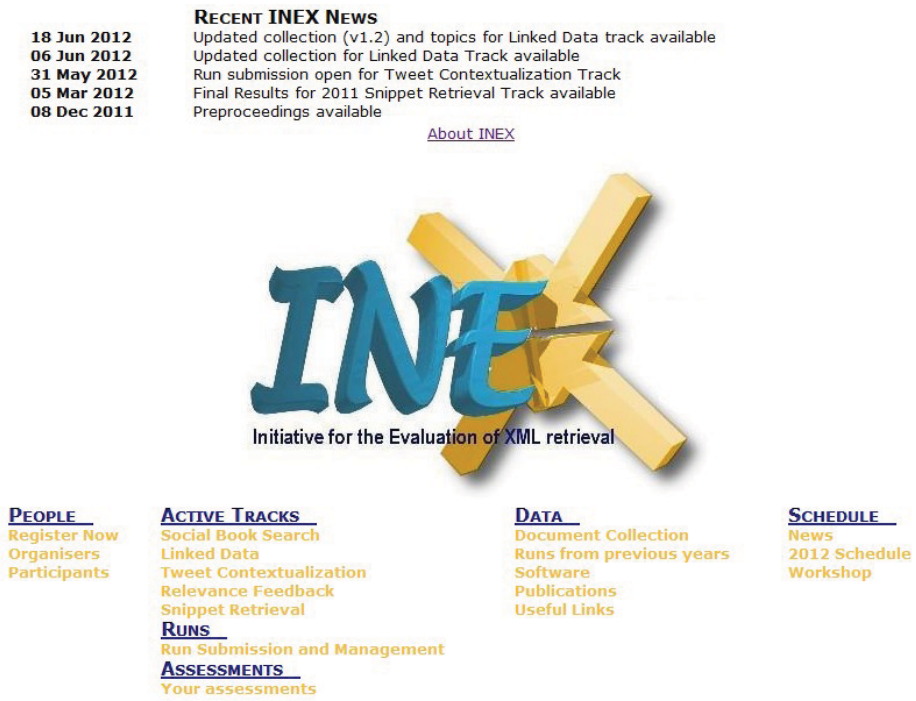


Fig. 1.

3 Workshop Veröffentlichungen Teilnehmer[4]

Im Vergleich zu einem Unternehmen, in dem Mitarbeiter tagtäglich an einem Ort auf ein Ziel hinarbeiten und das über einen mehr oder weniger festen Kreis von Mitarbeitern verfügt, läuft INEX seit Beginn der Initative anders ab. Man trifft sich einmal im Jahr zu einem Workshop, auf dem Ergebnisse des letzten Jahres präsentiert werden. Im Mittelpunkt davon stehen die verschiedenen Tracks, welche Aufgabenfelder darstellen, die gezielt Information Retrieval Probleme behandeln. Falls nötig wird über neue Tracks entschieden und bestehende Tracks weiterverfolgt. Der Workshop fördert in besonderer Weise den Wissensaustausch, da die Beteiligten sich gemeinsam an einem Ort befinden und so intensiver diskutiert werden kann. INEX ist kein festes Konstrukt, sondern erfährt ständig Veränderungen und muss angepasst werden, weswegen auch Vorschläge über Verbesserungen an INEX allgemein oder den Inhalten selber einen hohen Stellenwert bei dem Workshop haben. Der Workshop legt sozusagen die Marschroute für das nächste Jahr fest. Die Resultate dieses Workshops werden von Springer in den Lecture Notes in Computer Science veröffentlicht.
Generell kann jeder bei INEX mitwirken und die Initiative lebt von den Teilnehmern, welche zum Beispiel Topics vorschlagen und ihre Ergebnisse teilen. Die Teilnehmer

setzen sich zum Großteil aus Personen von Universitäten zusammen. Aus ursprünglich zwei Universitäten, hat sich die Anzahl teilnehmende Organisationen auf aktuell 97 mit 132 teilnehmenden Personen erhöht, welche sich jeweils mit den fünf unterschiedlichen Tracks beschäftigen. Als wichtige deutsche Person in Verbindung mit INEX, ist Norbert Fuhr von der Universität Duisburg-Essen zu nennen, der fast immer beteiligt gewesen ist und dadurch maßgeblichen Anteil an der Entwicklung von INEX hat. Interessant ist der Fakt, dass auch Unternehmen wie IBM oder GOOGLE teilnehmen, wodurch die Wichtigkeit von INEX deutlich wird. Welchen Zweck solche Unternehmen allerdings verfolgen, sollte an der Stelle kritisch hinterfragt werden und deren Einflussnahmen kontrolliert werden, wobei die Teilnahme an sich natürlich legitim ist.

4 Die Auszeichnungssprache XML[14]

XML steht für Extensible Markup Language und bedeutet erweiterbare Auszeichnungssprache. Sie ist eine Meta-Seitenbeschreibungssprache, die ein Format für die Beschreibung von strukturierten Daten bereitstellt, wodurch der Inhalt genauer deklariert werden kann. Dadurch dass Daten eine einheitliche Form bekommen erreicht man eine Unabhängigkeit vom Anbieter und Daten können einfach im großem Umfang implementiert werden. Für das XML Retrieval von besonderer Bedeutung sind XML Dokumente. Dabei ist die Sprache ein auf Text basierendes Format. Eine XML-Quelle besteht zunächst aus XML Elementen die von einem Starttag geöffnet und von einem Endtag geschlossen werden, zwischen denen sich Information, beziehungsweise der Inhalt befindet. Welche Elemente der Autor für das Dokument wählt, bleibt ihm überlassen. Er legt keine Darstellung, sondern nur die Bedeutung der Tags fest. Letztendlich erhält man eine Hierarchische Struktur, welche man sich am besten anhand eines einfachen Beispiel Artikels und die Baumstruktur dazu vor Augen hält:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Article>
  <Title>Buchliste</Title>
    <Section>
      <Paragraph>Buchinhalt</Paragraph>
    <Section>
</Article>
```

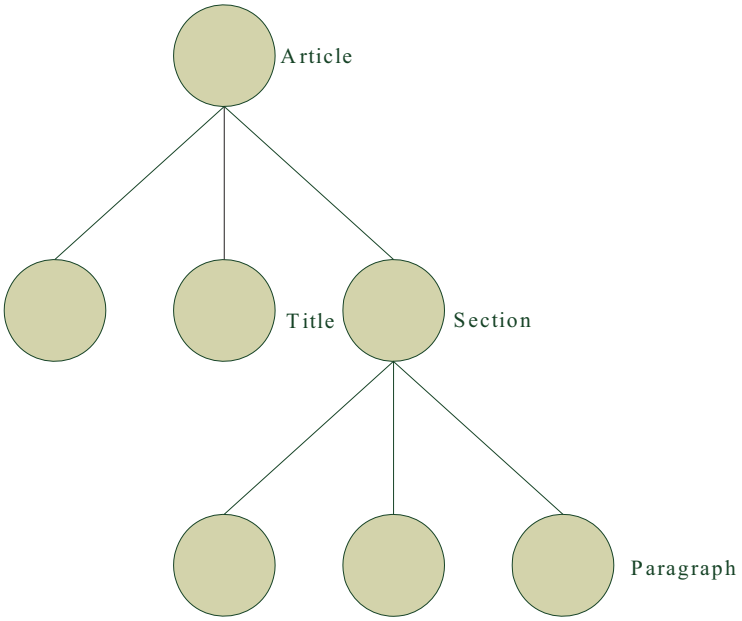


Fig. 2.

Diese Struktur von Dokumenten ist von zentraler Bedeutung für das XML Retrieval. So kann man bei der Anfrage gezielt Strukturbedingungen stellen. Voraussetzung hierbei ist, die Kenntnis über die zugrunde liegende Struktur. Hat man zum Beispiel eine große Sammlung von Artikeln im XML Format und weiß, dass alle Artikel einen Tag Titel haben, so kann man explizit nach Artikeltiteln suchen. Durch dieses Beispiel wird schnell klar, dass XML Retrieval besonders interessant ist, für digitale Büchereien, Produktkataloge oder andere große Sammlungen von Dokumenten, die über eine gleichbleibende und bekannte Struktur verfügen.

5 Der Ad hoc Track[1]

Der Ad hoc Track stellt die Standard XML Abfrage Simulation dar, bei dem Anfragen auf eine Dokumenten Sammlung gestartet werden und eine Ergebnisliste mit Dokumenten, beziehungsweise Dokumentelementen, entsprechend ihrer Relevanz zur Anfrage, gelistet zurückgegeben werden. Wegen seiner Wichtigkeit wird er im Folgenden genauer präsentiert. 2002 als erster Track eingeführt, wurde er bis 2010 weiterentwickelt, was ihn zum zeitlich längsten Track macht. Im Rahmen des Tracks entstanden die INEX test beds, welche aus Document Collections, Topics (Themenstellungen) und einer Relevanzbeschreibung bestehen. Über die Metrics kann man letztlich jeder Suchmaschine einen Wert für die Performance zuweisen. Die test beds definieren sozusagen das Testumfeld in dem evaluiert werden soll. Um möglichst reale Testbedingungen zu simulieren, hat man im Laufe der Zeit Veränderungen vorgenommen. Welche diese waren, wird in den nächsten Kapiteln geschildert.

5.1 Document Collections[1]

Am Anfang von INEX bestanden die Document Collections aus IEEE Computer Society´s Veröffentlichungen. Die verwendete Sprache hat man sich auf Englisch geeinigt. Sie waren 494 MB groß, besaßen 12,107 Artikel von 1995 bis 2002 und die Artikel besaßen im Durchschnnitt 1,532 XML Knoten. Die grobe Struktur von einem Artikel sieht so aus, dass man 3 Teile hat, front matter, body und back matter. Front matter untergliedert sich in Metadaten, wie title author publication information. Der Body, worin der eigentliche Inhalt steht, ist ebenfalls unterteilt in Sections, versehen mit title und Paragraphen. Im Back Matter steht unter anderem das Verzeichnis. Da die Struktur der Artikel fest war, war die Verwendung speziell für XML Retrieval Evaluierungszwecke naheliegend, da man jetzt in einer Anfrage gezielt jene Struktur einbeziehen konnte und eine große Menge von Dokumenten durchsuchen konnte. Anschließende Veränderungen der Document Collections gingen immer in die Richtung, die Sammlung zu vergrößern und so zu gestalten, dass man immer realere XML Retrieval Szenarios erhält. So hat man zunächst weitere IEEE Artikel hinzugefügt und mit der Aufnahme einer Speicherkopie von Wikipedia, welche sogar Bilddateien enthielt, die Sammlung extrem erweitert. Für die Buchsuche wurden zudem gescannte Bücher aufgenommen und der Lonely Planet Guide befindet sich ebenfalls innerhalb der Collection. Anzumerken ist, dass INEX die Artikel nicht besitzt, sondern für ihre Evaluierung benutzen darf. Zum Schluss ist festzuhalten, dass die Erweiterung der Document Collections Sinn macht, da man mit mehr Artikeln realer testen kann, auch wenn dadurch das Problem entsteht, dass man Retrieval Methoden, welche mit unterschiedlichen Document Collections evaluiert wurden, nicht vergleichen kann.

5.2 Topics

Beim Evaluieren sind Topics Themenstellungen, mit denen man Abfragen auf die Document Collections durchführt. Topics werden, genau wie die Document Collections in der englischen Sprache verfasst. Bevor allerdings Topics in INEX zu Evaluierung aufgenommen werden, durchlaufen diese einen Prozess. Jene Topics, von Teilnehmern vorgeschlagen, werden gesammelt, geprüft, Duplikate entfernt und dann an die Teilnehmer als Menge von Topics zurückgeschickt. Die Teilnehmer führen mit den Topics und ihren Suchmaschinen die Anfragen durch und schicken die Ergebnismenge zurück an INEX. Heraus kommt dabei eine Menge von Ergebnismengen, woraus mittels einer Methode, genannt Pooling, eine Menge aus Dokumenten gebildet wird, die für die Evaluierung genommen werden. Diese Dokumente werden dann wiederum an die ursprünglichen Verfasser der Topics geschickt, welche nun urteilen ob die Dokumente für die Abfrage relevant sind oder nicht.

Neue Topics sind nötig, damit die sich ständig verändernde Document Collections berücksichtigt wird. Des Weiteren stell dieser aufwändige Ablauf sicher, dass INEX nicht manipuliert werden kann oder sich keine Teilnehmer Vorteile verschaffen können, indem sie zum Beispiel Abfragen einbringen, von denen sie wissen, dass ihre Suchmaschine besonders gute Performance Werte liefert. Dies würde dazu führen, dass Teilnehmer sich benachteiligt fühlen, da die Evaluierung unseriös wäre. Wie bereits eingangs erwähnt wurde, sind kommerzielle Unternehmen wie IBM im Teilnehmerkreis kritisch zu betrachten, jedoch besteht von Seiten der Erstellung von Topics ein geringeres Risiko für Beeinflussung.

Im Laufe der Zeit haben sich verschiedene Arten von Topics gebildet, welche jeweils eine fest definierte Struktur aufweisen müssen. Zunächst zu erwähnen sind die content-only (CO) Topics, welche reine Inhaltsbedingungen enthalten und die Strukturbedingungen eines XML Artikels ignorieren. Zu vergleichen sind diese mit klassischen Information Retrieval Abfragen. Obwohl die Struktur bei den Topics keine Rolle spielt, spielt sie eine große beim Suchergebnis. Hier muss die richtige Granularität für die einzelnen Ergebnisse gefunden werden, mit dem Ziel die am relevantesten Dokumentelemente zurückzuliefern. Daneben gab es content-and-structure (CAS) Topics, welche explizit die hierarchische Dokumentstruktur eines XML Dokuments berücksichtigen.

```
<inex_topic topic_id="76" query_type="CAS">
<title>
//article[(./fm//yr = ’2000’ OR ./fm//yr = ’1999’) AND
about(.,
’"intelligent transportation
system"’)]//sec[about(.,’automation
+vehicle’)]
</title>
<description>
Automated vehicle applications in articles from 1999 or
```

```
2000 about intelligent transportation systems.
</description>
<narrative>
To be relevant, the target component must be from an
article on intelligent transportation systems published
in 1999 or
2000 and must include a section which discusses
automated vehicle
applications, proposed or implemented, in an
intelligent
transportation system.
</narrative>
</inex_topic>
```

Fig. 3. [1]

In der **Abbildung** ist die feste Struktur eines CAS Topis zu erkennen. Diese enthält als erstes ein title Element, in dem man Strukturbedingungen mittels der NEXI Abfrageprache verfassen kann und entspricht der langen Erklärung, dem nächsten festen Strukturelement, genannt narrative. Die description zwischen title und narative ist eine auf wesentliche gekürzte Beschreibung.

Die beiden genannten Topics wurden weiterentwickelt. Aus CO Topics wurden CO+S Topics, die ein optionales Strukturbedingungselement cas title, ebenfalls mittels der NEXI Abfragesprache formuliert, enthalten, um dadurch die Nützlichkeit von Strukturbedingungen zu untersuchen. Man konnte nämlich einmal mit Strukturbedingung (CO+S) evaluieren und direkt dagegen ohne (CO), mit dem ansonsten gleichen Topic.

Ein weiterer interessanter Aspekt in Bezug auf Strukturbedingungen ist wie diese überhaupt verwendet werden. Begonnen hat man mit einer strikten Auslegung der Bedingung, jedoch hat man die Bedingungen später nur noch vage interpretiert und sie mehr als Hinweise, wo sich ungefähr relevante Information befindet, gesehen.

Im Kapitel vorher wurde bereits erwähnt, dass die Document Collection immer größer wird, parallel dazu wächst auch die Menge an Topics. Aber auch hier ist die Entwicklung eine logische, da eben im XML Retrieval Benutzer eine Vielzahl von Abfragen tätigen und man wirklichkeitsnäher evaluieren kann. Schwierig wird es auch hier wieder Resultate von Retrieval Systemen zu vergleichen, sobald unterschiedliche Topics verwendet wurden.

5.3 Relevanz[1]

Nachdem in den Kapiteln zuvor die Abfragen und die Sammlung von Dokumenten getrennt betrachtet wurden, wird jetzt die Verbindung der beiden Mengen erläutert. Als Relevanz bezeichnet man die Beziehung zwischen Abfrage und den einzelnen Ergebnisdokumente. Die einfache Frage zu klären, welches Dokument relevant für eine Abfrage ist, hat sich als besonders schwer erwiesen und zu ständiger

Diskussionen und Änderungen geführt. Wichtig zu erkennen ist an der Stelle, dass das Kriterium Relevanz das von höchster Bedeutung für die Evaluierung ist, da Suchmaschinen ja wissen müssen was genau denn überhaupt zurückgeliefert werden soll. Sucht man zum Beispiel ein Dokument mit dem Wort Baum, werden natürlich alle Dokumente, in denen Baum vorkommt zurückgeliefert. Bei 2 Dokumenten in denen das Wort Baum unterschiedlich oft vorkommt, erwartet man natürlich, dass das Dokument höherer Priorität hat, worin häufiger Baum vorkommt. So schlicht heutzutage solche Überlegungen scheinen, müssen solche Aspekte zunächst im Relevanzkriterium definiert werden. Hinzu kommt, dass bei XML Retrieval die nötige Granulariät des zurückgelieferten Dokumentelements besonders wichtig ist um Fokused Retrieval zu ermöglichen. Man stellt sich nicht nur die Frage, welches Dokument relevant ist, sondern da man eine hierarchische Struktur im XML Dokument hat, auch welche Elemente des Dokuments relevant sind. Falls ein Parent Element und ein darin enthaltenes Child Element relevant sind, welches wird zurückgegeben?

Mit dem Relevanzkriterium versucht man Suchmaschinen zu lenken dem Nutzer die bestmögliche Information für sein Informationsbedürfnis zu liefern. Die Evaluierung soll ja nicht fern der Realität sein und letzlich die Suchmaschine am besten abschneiden, welche auch im alltäglichen Gebrauch die besten Ergebnisse für Benutzer liefert und nicht nur versucht werden, bei der Evaluierung gut abzuschneiden. Eine klare Formulierung der Relevanz hat sich auch für nötig erwiesen, da sonst immer wieder Fehlinterpretationen aufgetreten sind. Wenn ein Entwickler meint er habe seine Suchmaschine entsprechend des Relevanzkriteriums programmiert und schließt schlechter ab als erwartet, so kann das daran liegen, dass er das Relevanzkriterium falsch verstanden hat.

Begonnen hat man, die Relevanz zunächst anhand von zwei Dimesionen zu definieren. Zum einen die Topical Relevance welche besagt, inwiefern ein Ergebniselement das Informationsbedürfnis eines Suchenden befriedigt. Man misst die exhaustivity, also Ausgiebigkeit. Je mehr Bedingungen erfüllt werden, die im Topic gestellt werden, desto relevanter ist ein Element für das Topic, hat also eine höhere Topical Relevance. Als zweites hat man die Dimension Component Coverage eingeführt, die speziell für das Fokused Retrieval von Bedeutung ist, da die specifity, also die Genauigkeit, gemessen wird. Im Dokumentelement können nämlich neben den relevanten Informationen für den Suchenden auch nicht relevante Informationen stecken, wobei ein Dokumentenelement relevanter für eine Anfrage ist, das weniger überflüssige Information enthält, also eine höhere Component Coverage besitzt.

INEX hat die beiden Dimensionen später noch in 4 Teile unterteilt, um feinere Abstufungen zu definieren. Die Topical Relevance bestand danach aus:

- Irrelevant: The element does not contain any information about the topic of request.
- Marginally relevant: The element mentions the topic of request, but only in passing.
- Fairly relevant: The element discusses the topic of request, but not exhaustively.
- Highly relevant: The element discusses the topic of request exhaustively.

Fig. 4. [1]

Jetzt konnte man zum Beispiel abschätzen ob eine Section mit zwei Paragraphen relevanter ist, als jeder der Paragraphen für sich alleine. Um möglichste passende, gezielte Elemente von der richtigen Größe zurückzuliefern, also die richtige Granularität zu bestimmen, hat man die Dimension Topical Relevance ebenfalls in 4 Teile zerlegt:

- No coverage (N): The topic, or an aspect of the topic, is not a theme of the element.
- Too large (L): The topic, or an aspect of the topic, is only a minor theme of the element.
- Too small (S): The topic, or an aspect of the topic, is the main or only theme of the element, but the component is too small to act as a meaningful unit of information.
- Exact coverage (E): The topic, or an aspect of the topic, is the main or only theme of the element, and the element acts as a meaningful unit of information.

Fig. 5. [1]

An der Stelle der Entwicklung traten Probleme auf, weil die Abstufungen too small und too large zu Fehlinterpretationen geführt haben. Für CAS topics wurden die Kategorien als Relation zwischen der tatsächlichen Größe des Ergebniselements und des Zielelements verstanden, was falsch war, denn gemeint war die die Relation zwischen relevanter und nicht relevanter Information. Aus dieser Problematik heraus hat man die beiden Dimensionen neu formuliert:

- **Exhaustivity**, which measures how exhaustively an element discusses the topic of the user's request.
- **Specificity**, which measures the extent to which an element focuses on the topic of request (and not on other, irrelevant topics).

Fig. 6. [1]

In der Dimension Exhaustivity, welche Topical Relevance ersetzt hat, wurde in den Abstufungen das Wort relevant durch exhaustive ersetzt. Das im Absatz vorher besprochene Problem hat dazu geführt, dass man in den Abstufungen der Specifity, welche Component Coverage ersetzt, nun folgendermaßen unterschieden hat:

- Not specific: the topic of request is not a theme discussed in the element.
- Marginally specific: the topic of request is a minor theme discussed in the element.
- Fairly specific: the topic of request is a major theme discussed in the element.
- Highly specific: the topic of request is the only theme discussed in the element.

Fig. 7. [1]

Es gab aber auch Stimmen, die 1 Dimension alleine für sinnvoller hielten aber man hielt 2 Dimensionen zunächst für richtig, da bei der Bewertung wahrscheinlich verschiedene Schwerpunkte von den Bewertern gesetzt werden, wenn man einen Wert für die Relevanz vergibt. Eine Person tendiert dazu highly specific Elemente für relevanter zu halten und eine andere hält higly exhaustive für relevanter, dafür kann die Specifity geringer ausfallen.

Unter anderem aus Kostengründen hat man die Zuordnung Bewertung für Elemente vereinfacht. Dies geschah, indem man erst relevante Passagen hervorgehoben hat und dann innerhalb dieser Passagen Elemente bewertet hat. Im Zuge dieser Simplifizierung wurde abermals die Specifity Dimension neu formuliert und man hat die Exhaustivity Dimension in eine 3 +1 Abstufung geändert.

Am Ende dieser Entwicklung über die Relevanz, hat man sich nach langer Diskussion dazu entschieden, die Exhaustivity Dimension wegfallen zu lassen. Wiederrum waren Kostengründe und die Notwendigkeit einer einfachen Nachvollziehbarkeit, der man durch die Reduzierung auf die leichter verständliche Specifity Dimension gerecht wurde, ausschlaggebend.

Wichtig ist, dass durch diese Vereinfachung keine Qualitätseinbußen für die Evaluierung und Vergleichbarkeit von Suchmaschinen entstanden sind, was anhand von ausgiebigen Studien belegt wurde. Abschließend bleibt noch zu sagen, dass dieses Kapitel über die Entwicklung der Relevanz zeigt, wie schnell sich Inhalte und Ansichten verändern können und INEX eben nie als festes Konstrukt anzusehen ist. Man wird die richtige Definition für die Relevanz vielleicht nie hundertprozentig finden, jedoch verfolgt auch hier INEX das Ziel möglichst nahe am realen XML Retrieval zu sein.

5.4 Metrics[1]

Das Hauptziel von INEX ist das Messbarmachen der Performance von Suchmaschinen, um diese anschließend vergleichen zu können. Dieses Kapitel stellt dabei eine frühe Vorgehensweise grob da, ohne genauer auf Formeln und Funktionen einzugehen. Stattdessen werden Probleme und Besonderheiten die beim XML Retrieval in Verbindung mit den Metriken entstanden sind aufgeführt. Wieder ist die Struktur eines XML Dokument dafür ausschlaggebend gewesen.

Ähnlich wie bei andere Evaluierungsinitiativen, wird die Effektivität einer Suchmaschine bei INEX mittels Precision und Recall Metriken oder Varianten davon ermittelt. Der Name dafür lautet inex_eval metric und basiert auf einem Zählmechanismus, der wiederrum die Anzahl zurückgegebener Elemente und

relevanten Elementen einbezieht. Später wurde dieser Zählmechanismus fallen gelassen und Metriken, welche einerseits auf einen aufsummierten Mehrwert und andererseits auf effort-precission/gain-recall basierten, angewandt. Es wurde dabei in user-oriented und system-oriented unterschieden, wobei hier nur ersteres erklärt werden soll.

Bei User-oriented wurde für jedes zurückgegebene Element ein Mehrwert errechnet von null bis eins. Dieser Mehrwert ist abhängig von den Dimensionen Exhaustivity und Specifity, welche im vorherigen Kapitel behandelt wurden. Mittels Quantisierungsfunktionen erhielt man aus diesen beiden Dimensionen schließlich einen Wert von eins bis null. Diese Quantisierungsfunktionen haben sich entsprechend der Entwicklung der Relevanzdimensionen ebenfalls im Laufe der Zeit verändert. Als die Exhaustivity Dimension wegfiel, hat die Quantisierungsfunktion lediglich einem Element einen Specifity Wert zugeordnet. Die Werte der Elemente aus der Abfrageergebnisrangliste werden aufsummiert und mit einer idealen Ergebnisrangliste verglichen. Der Wert 1 steht dann für optimale Performance, da die Ergebnisliste genau der idealen Ergebnisliste entspricht.

Es gab jedoch zwei Problematiken, welche zu Veränderungen geführt haben. Als erstes die Overlapping elements, die Elemente in einem XML Dokument sind, welche einander enthalten. Zum Beispiel ein Parent Element und sein Child Element getrennt betrachtet, sind Overlapping Elements, da im Parent Element ja bereits das Child Element steckt. Hilfreich ist es an der Stelle, sich die hierarchische Baumstruktur eines XML Dokuments ins Gedächtnis zu rufen. Zum Probleme wurden solche Elemente dadurch, dass Suchmaschinen die sehr viele relevante Overlapping Elements als Suchergebnis auf eine Abfrage zurückgeliefert haben, bei der Evaluierung besser abgeschnitten haben als Suchmaschinen, die gezielt versucht haben redundante Information zu vermeiden. Jemand der einen Text über Schuhe sucht, möchte als Suchergebnis schließlich nicht unzählige Treffer aus einem Dokument mit immer den gleichen Informationen, sondern lieber verschiedene Stellen im Dokument oder aus anderen Dokumenten als Ergebniselement erhalten.

Die zweite Problematik waren near-misses. Als Near-misses bezeichnet man Elemente, von denen aus relevante Information erreicht wird. Die Struktur von XML Dokumenten macht es möglich, dass Benutzer Zugang zu anderer, strukturell in Verbindung stehender Information haben, indem sie browsen oder scrollen. Dieser Umstand führt dazu, dass man near-misses in die Metriken einbeziehen und beachten muss. Eine Suchmachine die zum Beispiel anstatt keinem Ergebnistreffer zu einer Buchsuche, wenigstens Treffer anzeigt, von denen man das gesuchte Buch findet, soll bei der Evaluierung besser abschneiden.

Beide Problematiken wurden erkannt und die Metriken angepasst. Am Schluss dieses Kapitels kann man das Fazit ziehen, dass man wie bereits beim Relevanzkriterikum versucht hat, das optimale Suchverhalten für einen Nutzer auch am besten zu bewerten. Dies geschieht über Funktionen und dem Relevanzkriterium und man bedient sich dabei der Mathematik. Die Metriken bilden das Herzstück der Evaluierung. Suchmaschinen, die in der Praxis gute Ergebnisse erzielen, müssen bei der Evaluierung noch lange nicht so gut abschneiden, da immer die Metriken ausschlaggebend sind, was durch die overlapping elements gezeigt wurde.

6 Trackübersicht

Außer dem Ad hoc Track hat es in der Zeit von INEX 2002 bis INEX 2012 noch 18 weitere Tracks gegeben. Neben einer Übersicht sollen die Tracks allgemein analysiert werden und die aktuellen Tracks anklingen.

Table 1. zeigt die Tracks, wobei auf der X-Achse die Zeit von 2002 bis 2012 angetragen ist und auf der Y-Achse die Namen der verschiedenen Tracks. Die Balken entsprechen dem Zeitintervall, in welchem die Tracks bei INEX entwickelt und bearbeitet wurden. Interessant sind einige Fakten, welche man anhand der Übersicht erkennen kann. Der längste Track war der neun Jahre dauernde Ad hoc Track, da dort im Rahmen der Entwicklung viele Änderungen stattgefunden haben und das Grundgerüst für die Evaluierung erarbeitet wurde. Die kürzesten Tracks waren der ein Jahr dauernde Web Service Discovery Track und der Use Case Track, bei dem versucht wurde, herauszufinden wer die wirklichen Benutzer von XML Retrieval sind, um dann entsprechend die Evaluierungsszenarios so zu gestalten dass sie möglichst real sind. Die Dauer eines Tracks lässt nicht auf die Wichtigkeit schließen, denn jeder Track hat seine Darseinsberechtigung, sonst würde er nicht ins Leben gerufen werden, sondern vielmehr wird ein Track nicht mehr weiterverfolgt, sobald man meint der aktuelle Stand ist für die Evaluierung richtig. Nimmt man alle Jahre der Tracks zusammen, erhält man 48 Jahre, wobei diese ja zum Teil nebeneinander bearbeitet wurden und teilt man diese durch die Anzahl der bis INEX 2011 stattgefundenen 14 Tracks, so kommt man auf eine durchschnittliche Zeit eines Tracks von 3,42 Jahren. Addiert man die Anzahl der Tracks von INEX 2002 bis INEX 2012, die jeweils in einem Jahr stattgefunden haben, und teilt sie durch die elf Jahre, welche INEX besteht, so kommt man auf durchschnittlich 5,54 Tracks pro Jahr. Schön verdeutlicht die Grafik dabei, dass zunächst nur der Ad hoc Track entwickelt wurde, dann sprunghaft 2004 fünf Tracks entstanden sind, um vielleicht in mehrere Richtungen weiter zu forschen und verschiedene Probleme getrennt voneinander zu bearbeiten.. Von da an stieg die Anzahl der Tracks pro Jahr von INEX, bis 2010 der Höhepunkt, mit 9 Tracks pro Jahr, erreicht wurde. INEX 2011 und INEX 2012 hat man die Tracks dann wieder deutlich reduziert, vielleicht um dadurch gezielter Aufgabenstellungen zu lösen und sich darauf zu konzentrieren anstatt zu viele offene Baustellen zu haben. Der letzte Punkt, welcher aus der Grafik ersichtlich ist, ist der Relevance Feedback Track, welcher 2004 bis 2005 und dann 2010 neu aufgegriffen wurde und seitdem weiterentwickelt wird, weshalb man nie sagen kann, dass ein beendeter Track eine Aufgabenstellung zu XML Retrieval für immer richtig gelöst hat, sondern Änderungen ständig möglich sind.

INEX 2012 hat neben dem Relevance Feedback Track noch den Social Book Search Track, welcher sich aus dem Book Track entstanden ist, den Linked Data Track, den Tweet Contextualization Track und den Snippet Retrieval Track. Letzterer befasst sich mit der Darstellung eines Snippets, also eines Schnipsels für jedes Dokument aus der Suchergebnismenge. Dadurch soll ein Benutzer schnell und einfach urteilen können, ob das Dokument relevant für ihn ist oder nicht.

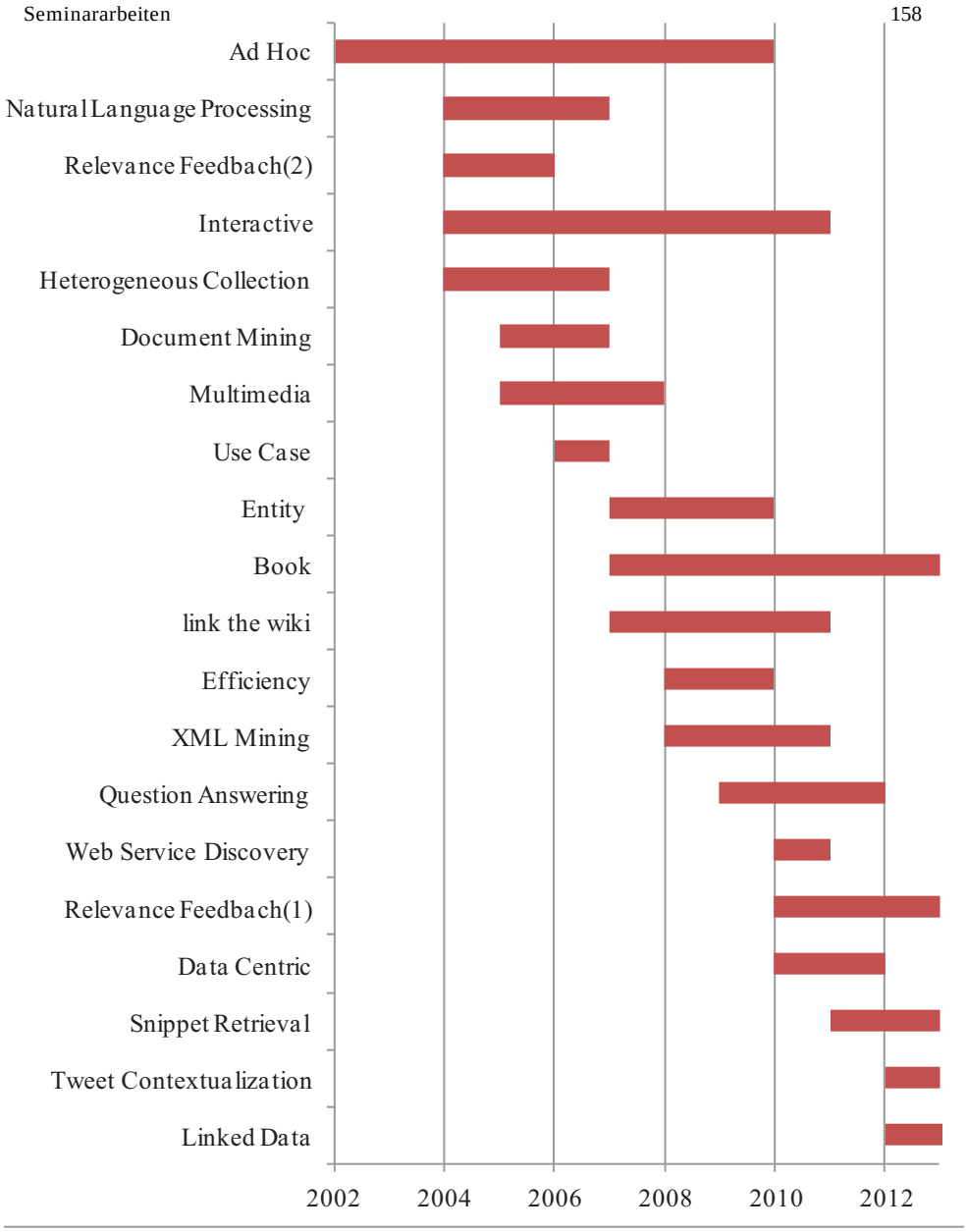

Ad Hoc
Natural Language Processing
Relevance Feedbach(2)
Interactive
Heterogeneous Collection
Document Mining
Multimedia
Use Case
Entity
Book
link the wiki
Efficiency
XML Mining
Question Answering
Web Service Discovery
Relevance Feedbach(1)
Data Centric
Snippet Retrieval
Tweet Contextualization
Linked Data
2002
2004
2006
2008
2010
2012

Table 1.[1, 5-13]

7 Zusammenfassung, Fazit und Ausblick

XML ist eine Auszeichnungssprache, die sehr verbreitet ist und sich vor allem für große Datenmengen wie Büchern, Artikeln oder Dokumenten eignen, welche eine gleichbleibende Struktur aufweisen und die man bei der Such berücksichtigen kann. Daraus entstanden sind eine Vielzahl von Suchmaschinen für XML Retrieval. Das Hauptziel von INEX ist die Evaluierung von diesen Suchmaschinen und deren Methoden. Die Evaluierungsinitiative hat seit seine Entstehung die Evaluierung für XML Retrieval ständig verbessert und bietet mittlerweile umfangreiche und passende Testbedingungen, in Form von Document Collections, Topics und Relevanzkriterium an. So kann man ein möglichst reales XML Retrieval Szenario simulieren. Besonderheiten des XML Retrievals, die durch die Struktur eines XML Dokuments entstehen, wie near-misses und overlapping elements, werden im Relevanzkriterium und den Metrics berücksichtigt. Eine sinnvolle Arbeitsweise von INEX stellt dabei die Aufteilung in Tracks da, wodurch gezielt XML Information Retrieval Probleme benannt, die es dann zu lösen gilt. Bisher gab es 19 Tracks unterschiedlicher Dauer. INEX 2012 wird im Wesentlichen den Weg mit 5 aktuellen Tracks weiter beschreiten wie bisher, wobei nach 10 Jahren alleinigen Arbeitens, das erste Mal mit CLEF zusammengearbeitet wird. CLEF steht für Cross-Language Evaluation Forum und dient ebenfalls der Evaluierung im Information Retrieval, jedoch mit europäischen Sprachen, im Gegensatz zu INEX, wo Englisch verwendet wird.(**4**) Spannend ist die Frage, wie diese Teamarbeit zweier so großer und bisher eigenständigen Initiativen funktionieren wird.

8 Quellen

[1]"INEX 2002 - 2006: Understanding XML Retrieval Evaluation", geschrieben von Mounia Lalmas and Anastasios Tombros Queen Mary University of London, Mile End Road, London, UK

[2]https://inex.mmci.uni-saarland.de/about.html Aufrufdatum:22.07.2012

[3]http://de.wikipedia.org/wiki/Extensible_Markup_Language Aufrufdatum:22.07.2012

[4]https://inex.mmci.uni-saarland.de/people/participants.jsp

[5]Overview of the Initiative for the Evaluation of XML retrieval (INEX) 2002
Norbert Gövert University of Dortmund Germany,
Gabriella KazaiQueen Mary University of London United Kingdom

[6]Report on the INEX 2003 Workshop
Schloss Dagstuhl,
15-17 December 2003
Norbert Fuhr, Mounia Lalmas

[7]Overview of INEX 2004
Mounia Lalmas and Norbert Fuhr

[8]Overview of INEX 2005
Gabriella Kazai Mounia Lalmas and Norbert Fuhr
Information Systems, University of Duisburg-Essen, Duisburg, Germany
Department of Computer Science, Queen Mary, University of London, LondonUK

[9]Overview of INEX 2006
Andrew Trotman, Mounia Lalmas, Norbert Fuhr
University of Duisburg-Essen, Duisburg, Germany
University of Otago, Dunedin, New Zealand
Queen Mary, University of London, London, UK

[10]Report on INEX 2008
Gianluca Demartini Ludovic Denoyer Antoine Doucet Khairun Nisa Fachry
Patrick Gallinari Shlomo Geva Wei-Che Huang Tereza Iofciu
Jaap Kamps Gabriella Kazai Marijn Koolen Monica Landoni
Ragnar Nordlie Nils Pharo Ralf Schenkel Martin Theobald
Andrew Trotman Arjen P. de Vries Alan Woodley Jianhan Zhu

[11]Report on INEX 2009
T. Beckers P. Bellot G. Demartini L. Denoyer C.M. De Vries
A. Doucet K.N. Fachry N. Fuhr P. Gallinari S. Geva
W.-C. Huang T. Iofciu J. Kamps G. Kazai M. Koolen
S. Kutty M. Landoni M. Lehtonen V. Moriceau R. Nayak
R. Nordlie N. Pharo E. SanJuan R. Schenkel X. Tannier
M. Theobald J.A. Thom A. Trotman A.P. de Vries

[12]Report on INEX 2010
D. Alexander P. Arvola T. Beckers P. Bellot T. Chappell C.M. De Vries
A. Doucet N. Fuhr S. Geva J. Kamps G. Kazai M. Koolen
S. Kutty M. Landoni V. Moriceau R. Nayak R. Nordlie N. Pharo
E. SanJuan R. Schenkel A. Tagarelli X. Tannier J.A. Thom A. Trotman
J. Vainio Q. Wang C. Wu

[13]Report on INEX 2011
P. Bellot T. Chappell A. Doucet S. Geva J. Kamps
G. Kazai M. Koolen M. Landoni M. Marx V. Moriceau
J. Mothe G. Ram__rez M. Sanderson E. Sanjuan F. Scholer
X. Tannier M. Theobald M. Trappett A. Trotman Q. Wang
[14]http://msdn.microsoft.com/de-de/library/cc431269.aspx

Techniken zur Identifikation typischer Wikipedia Anfragen

Florian Gundelsheimer

Otto-Friedrich-Universität Bamberg, Lehrstuhl für Medieninformatik

Florian-manfred.gundelsheimer@stud.uni-bamberg.de

Zusammenfassung Diese Seminararbeit beschäftigt sich mit Methoden zur Identifikation von typischen Wikipedia Anfragen. Nach der Einleitung wird in Kapitel 2 die Vorgehensweise einer Log-Dateien-Analyse erklärt und die Ergebnisse aufgezählt, die diese liefern kann. Anschließend wird näher auf die Techniken und Möglichkeiten des Web Minings eingegangen. In Kapitel 4 werden verschiedene Studien dargestellt, welche Suchanfragen kategorisieren. Danach werden zwei weitere Studien betrachtet, die sich mit der Klassifizierung von Suchanfragen in verschiedenen Themengebieten beschäftigen. Abschließend erfolgt in Kapitel 6 ein Fazit der Arbeit.

Schlagworte: Log-Dateien-Analyse, Web Mining, informationsorientierte Suchanfragen, navigationsorientierte Suchanfragen, Wikipedia

1 Einleitung

Wikipedia, die freie Online Enzyklopädie, wer kennt sie nicht? Der Erfolg dieses Internet Lexikons ist erstaunlich. Zunächst einige Zahlen, die den Stellenwert der Wikipedia klarstellen. Die deutsche Wikipedia enthält über 1,4 Millionen Artikel und ist damit hinter der Englischen, die über 4 Millionen Artikel enthält, die zweitgrößte (vgl. ww.wikipedia.org Stand August 2012). Die Qualität der Artikel weist dabei große Unterschiede auf. Eine Studie, die im Auftrag von Nature erstellt wurde [Giles 2005], wird besonders kontrovers diskutiert. Diese Studie bescheinigt Wikipedia eine Qualität, ähnlich der bekannten *Encyclopedia Britannica.* (vgl. Rainer Hammwöhner, Karl-Peter Fuchs, Markus Kattenbeck, Christian Sax.*)*Wikipedia kann somit einen guten ersten Einblick in die meisten Themengebiete geben, ist allerdings als Primärquelle einer wissenschaftlichen Arbeit wohl eher ungeeignet.

Ungeachtet dessen, wird die Online Enzyklopädie meist nicht direkt besucht, sondern über Suchergebnisse, welche durch eine Suchmaschine erzeugt wurden. 2/3 der Wikipedia Besucher gelangen über Suchmaschinen auf die Website, wobei Google mit über 93% den größten Anteil ausmacht (vgl Vivienne Waller).

Interessant ist es nun zu identifizieren, welche Suchanfragen typischerweise auf Wikipedia führen.

Diese Seminararbeit wird dabei zunächst auf die Vorgehensweise bei einer Log-File Analyse, sowie dem Web Mining eingehen, da diese zur späteren Kategorisierung von Suchanfragen benötigt werden. Anschließend werden verschiedene Studien vorgestellt, die Suchanfragen kategorisieren und damit erkennen lassen, welche Anfragen, typischerweise auf den Internetauftritt von Wikipedia verweisen.

2 Vorgehensweise bei einer Log-File Analyse

Die übliche Vorgehensweise zur Auswertung von Internet-Nutzungsdaten besteht in der Erstellung einer Log-File Analyse. Bei einer Log-File Analyse werden Log Dateien von einem Internet Provider oder Proxy Server erworben, um diese dann mit Hilfe von Programmen zu analysieren *(*vgl Vivienne Waller).Aus einer Log Datei können folgenden Daten entnommen werden:

183.121.143.32 - - [18/Mar/2003:08:04:22 +0200] "GET /images/logo.jpg HTTP/1.1" 200 512 "http://www.wikipedia.org/" "Mozilla/5.0 (X11; U; Linux i686; de-DE;rv:1.7.5)

Bedeutung:	**Beispiel:**
IP oder DNS -Adresse des zugreifenden Computers	183.121.143.32
Zeitpunkt des Zugriffs	[18/Mar/2003:08:04:22 +0200]
Kommando, das an den Server gestellt wird - meistens wird mit dem Befehl (GET) eine Datei angefordert	GET /images/logo.jpg
das benutzte Übertragungsprotokoll, z.B. HTTP 1.1	HTTP/1.1
die Antwort des Servers	200 bei erfolgreicher Anfrage, 404 bei Seite nicht gefunden

Größe der angeforderten Datei	512 KB
Browser	Mozilla/5.0
Ursprungsseite des Users	"http://www.wikipedia.org/"
Betriebssystem	Linux i686; de-DE;rv:1.7.5

Log-File Statistiken müssen sorgfältig geprüft und vorsichtig interpretiert werden (vgl. Heindl, 2003, http://de.wikipedia.org/wiki/Logdatei)

2.1 Probleme bei einer Log-File Analyse

Die beiden Hauptprobleme bei der Auswertung von Log Dateien sind auf der einen Seite, die Unterscheidung der einzelnen Nutzer und zum anderen, die unvollständigen Daten aufgrund von Caching. Weiterhin kann es vorkommen, dass mehrere Nutzer mit der gleichen IP Adresse auf eine Website zugreifen und somit im Log-File als ein einzelner Nutzer gewertet werden. Außerdem kann auch ein einzelner User, aufgrund von dynamischer IP-Vergabe, mehrfach gezählt werden. Ein weiterer Faktor, der die Zugriffszahlen von Webseiten verfälscht, ist das Cachen durch einen Proxy Server, der zwischen Client und Server geschaltet sein kann. Dieser Proxy-Server tritt den Client gegenüber als Server auf, während er den Web-Server gegenüber als Client Auftritt(vgl. http://de.wikipedia.org/wiki/Logdateianalyse)

All diese Faktoren verfälschen die Zugriffszahlen auf Wikipedia und müssen bei der Auswertung von Log-Files beachtet werden, um geeignete Kennzahlen zu erhalten.

2.2 Auswertung der Log-Dateien

Bei der Auswertung von Log Dateien gibt es viele Programme, welche die Analyse vereinfachen. Folgende Auswertungsmöglichkeiten sind bei einer herkömmlichen Log-File Analyse gegeben.

2.2.1 Hits

- Jeder Aufruf der den Web-Server erreicht, erzeugt einen Hit. Hits erfassen somit alle Dateien, die vom Web-Server zum Nutzer übertragen werden. Sind in einer HTML-Seite weitere Formate eingebunden, wie zum Beispiel PDF-Dateien, oder Bilder, erzeugen diese weitere Hits auf dem Server. Um den tatsächlichen Erfolg eines Internetauftritts einzuschätzen, sind Hits somit nicht geeignet. Man kann jedoch ermitteln, wie sich der Traffic auf der

Webseite über Zeiträume hinweg entwickelt (vgl. http://dbs.uni-leipzig.de/files/abstr/wum-DB-Spektrum_02-02.pdf

2.2.2 Pageviews

Mit Pageviews kann die Anzahl von Aufrufen einzelner Webseiten gemessen werden. Bei Wikipedia gibt es Aufschlüsse darüber, welche Artikel besonders beliebt sind (vgl. http://www.e-teaching.org/didaktik/qualitaet/logfile/).

2.2.3 Sessions

Sessions geben Auskunft darüber, welche Seiten ein Nutzer während seines Aufenthalts auf dem Interangebot besucht. Dabei erhält er beim erstmaligen Besuch der Website eine Session-ID, über welche sämtliche Aufrufe, die dieser User tätigt, identifiziert werden können (vgl. http://www.e-teaching.org/didaktik/qualitaet/logfile/).

2.2.4 Anzahl der Besucher

Die Zahl der Besucher zeigt, wie viele einzelne Besucher auf das Internetangebot zugegriffen haben. Über die Session-ID können Besucher identifiziert werden, die die Website bereits besucht haben. Somit können Schlussfolgerungen gezogen werden, wie attraktiv das Internetangebot für Besucher ist und ob diese regelmäßig wiederkommen (vgl. http://www.e-teaching.org/didaktik/qualitaet/logfile/).

2.2.5 Klickpfade

Klickpfade zeigen den Weg, den ein User durch das Webangebot genommen hat. Sie geben Aufschluss, wie der Nutzer durch das Onlineangebot navigiert und welche Wege er genommen hat, bis er zu seinem vermeintlichen Ziel kam, beziehungsweise wo er die Homepage verlassen hat. Diese Daten können Aufschluss darüber geben, welche Artikel bei Wikipedia überarbeitet werden müssten, da viele User die Seite bei schlechten Artikeln zügig verlassen (vgl. http://www.e-teaching.org/didaktik/qualitaet/logfile/).

3 Web Mining

Eine weitere Möglichkeit Log File Analysen zu automatisieren und noch mehr Informationen aus Log Dateien zu gewinnen, beruht auf den Methoden der Übertragung des Data Minings auf das Internet. Diese, vom Data Mining auf das Web übertragene Verfahren, werden Web Mining genannt (vgl. http://de.wikipedia.org/wiki/Web_Mining). Im Folgenden werden die drei verschiedenen Arten von Web Mining genannt und kurz erklärt. Abschließend wird noch etwas genauer auf das Web Usage Mining eingegangen.

3.1 Web Structure Mining

Das Web Structure Mining beschäftigt sich mit der Untersuchung der Anordnung von einzelnen Elementen innerhalb einer Website, sowie mit der Anordnung verschiedener Webseiten zueinander. Dabei wird besonderes Augenmerk auf den Verweisen von einer Website auf die nächste gelegt(vgl. http://de.wikipedia.org/wiki/Web_Mining).

3.2 Web Content Mining.

Das Web Content Mining widmet sich der Analyse von Inhalten von Websites. Ziel ist es, die Suche nach Informationen im Internet zu erleichtern. Dabei werden Online-Dokumente gruppiert und klassifiziert. Außerdem wird das Auffinden von Dokumenten nach Eingabe bestimmter Suchbegriffe analysiert. Hierbei werden vor allem Verfahren des Text Mining verwendet(vgl. http://de.wikipedia.org/wiki/Web_Mining).

3.3 Web Usage Mining

Das Web Usage Mining beschäftigt sich mit der Analyse des Nutzerverhaltens von Besuchern einer Website und ist daher für uns besonders interessant. Bei dieser Ausprägungsform des Web Minings, werden Data Mining Methoden auf die Log Files des Web Servers angewandt, um Aufschlüsse über Verhaltensmuster und Interessen der Wikipedia Nutzer zu erhalten und stellt somit eine Methode zur Identifikation von typischen Wikipedia Anfragen dar (Srivastava et al. 2000). Das Web Usage Mining besteht aus drei Phasen(vgl. Markus Bischoff):

- Data Preprocessing
- Pattern Discovery
- Pattern Analysis

Der erste Schritt, die Vorverarbeitung, ist relativ schwierig, da die vielen Daten der Serverlogs die eindeutige Identifizierung eines jeden Nutzers erschweren. Hier liegen die gleichen Probleme, wie bei einer Log-File Analyse vor. Beim Preprocessing werden nicht relevante Felder aus den Daten entfernt. Verschiedene Datenquellen, wie z. B. Server Log Dateien, oder Daten aus einer Datenbank, werden zu einer Datenbank zusammengefügt und in eine Form gebracht, die eine Analyse vereinfachen. Bei dieser Bearbeitung der Daten werden bspw. Benutzer identifiziert oder Sessions eines Nutzers ermittelt. Wenn dies erfolgt ist, wird eine Musterkennung eingeleitet, wobei verschiedene Methoden zum Einsatz gebracht werden. Es können statische Analysen durchgeführt werden, welche Informationen über die Besucher einer Webseite liefern(vgl. Markus Bischoff). Mögliche Analyseergebnisse sind die Anzahl der Pageviews, mit deren Hilfe die am häufigsten besuchten Seiten auf Wikipedia identifiziert werden können.

Weiterhin kann die View Time analysiert werden. Diese zeigt auf, wie lange ein User eine Seite besucht hat. Dies kann Aufschluss darüber geben, ob der Nutzer den Artikel gelesen hat, oder die Seite schnell wieder verlassen hat. Außerdem kann der Navigationspfad innerhalb des Webauftritts betrachtet werden. Desweiteren können mit Hilfe von Clustering-Verfahren, Nutzergruppen ermittelt und mittels Klassifikation Nutzer bestimmt werden. Der letzte Schritt des Web Usage Mining ist die Musteranalyse. Dabei wird nach interessanten Mustern, Regeln und Statistiken gesucht, mit deren Hilfe wiederum typische Wikipedia Anfragen identifiziert werden können.

Letztendlich kann man sagen, dass in nachfolgenden Studien Methoden der Log-File Analyse und des Web Usage Mining genutzt wurden, um Studien anzufertigen. In den Studien wurden Suchanfragen kategorisiert. Dazu ist es notwendig Log Dateien von Suchmaschinen zu analysieren, um erkennen zu können, welche Anfragen informationsorientiert sind und somit zumeist typische Wikipedia Anfragen darstellen.[1]

[1] http://i-san.de/lexikon/suchmaschinenoptimierung-seo/23-suchverhalten, http://www.seopt.de/seoblog/allgemein/wikipedia-eintraege-bei-google.html

4 Studien zur Kategorisierung von Suchanfragen

Eine Studie, die Broder 2002 veröffentlicht wurde, untersuchte 400 Suchanfragen und ordnete die Suchanfragen drei verschiedenen Kategorien zu (vgl. BRODER 2002, S.5). Die Log Dateien Analyse der Suchmaschine Altavista kam zu folgender Kategorisierung von Suchanfragen:

- Navigationsorientierte Suchanfragen
- Transaktionsorientierte Suchanfragen
- Informationsorientierte Suchanfragen

Type of Query	User Survey	Query Log Analysis
Navigational	24.5%	20%
Informational	??(estimated 39%)	48%
Transactional	> 22% (estimated 36%)	30%

Suchanfragenverteilung(vgl. Broder 2002

4.1 Navigationsorientierte Suchanfragen

Ein Benutzer erstellt eine navigationsorientierte Anfrage, um eine Seite zu besuchen, die er bereits kennt oder die er bereits besucht hat (vgl. Broder 2002).

4.2 Transaktionsorientierte Suchanfragen

Transaktionsorientierte Suchanfragen haben die Intension, nach dem Auffinden der Seite, eine Transaktion durchzuführen, wie zum Beispiel den Kauf eines Buches oder den Download einer Datei (vgl. Broder 2002).

4.3 Informationsorientierte Suchanfragen

Informationsorientierte Suchanfragen entsprechen am ehesten dem im Information Retrieval typisch gestellten Anfragen. Informationsorientierte Anfragen zielen darauf ab, Information über ein bestimmtes Thema zu bekommen. Wurde die Information, die man gesucht hat, gefunden wird der Artikel gelesen und anschließend ist die Suchanfrage beendet(vgl. Mach Sonja S.10-11). Die Suchanfragen bestehen dabei aus

einzelnen Wörtern oder aus ganzen Sätzen. Weiterhin ist eine Verbindung zwischen informationsorientierten Suchanfragen und typischen Wikipedia Anfragen zu erkennen, da Wikipedia viele informationsorientierte Suchanfragen „beantworten“ kann (vgl. http://www.seopt.de/seoblog/allgemein/wikipedia-eintraege-bei-google.html, Vivienne Waller).

Somit kann man mit Hilfe der Methoden zur Identifikation von informations-orientierten Anfragen auch typische Wikipedia-Anfragen identifizieren. User, die gezielt Wikipedia suchen geben meist die Zusätze (*Wikipedia*, *wiki*, *Wikipedia encyclopaedia*, *site:en.Wikipedia.org*, und *www.Wikipedia.org*) ein. In diesem Fall ist es eine navigationsorientierte Anfrage mit dem Ziel, eine Wikipedia Webseite zu besuchen (Broder 2002). Hat der User zu Beginn der Suche nicht die Intention Wikipedia zu besuchen, sind es meist informationsorientierte Suchanfragen, die auf Wikipedia führen. In folgenden Studien wird näher auf Methoden zur Identifikation von informationsorientierten und navigationsorientierten Suchanfragen eingegangen.

4.4 Vorstellung verschiedener Studien

Im Folgendem werden verschiedene Studien vorgestellt, die versuchen Suchanfragen zu kategorisieren. Die Grundlage der Studien bilden zumeist die Kategorien die Broder 2002 aufgezählt hat.

4.4.1 Kang und Kim 2003

Kang und Kim berechnen in ihrer Studie, mit welcher Wahrscheinlichkeit eine Anfrage einer Kategorie zugeordnet werden kann. Dabei kann es der Fall sein, dass eine Anfrage mehreren Kategorien zugeordnet werden kann. Die Kategorien entsprechen denen von Broder 2002.

Bei dem Versuch Anfragen zu kategorisieren, werden in der Studie Inhaltsinformation (z.B. Titel und Texte der Homepage), Linkinformationen (Links die zu Website führen) und URL- Informationen (Name der Organisation, Anzahl der Slashes als Hinweis, ob es sich um eine Hauptseite oder Unterseite handelt) analysiert.

Informationsorientierte Anfragen enthalten meist Schlagworte oder eine nähere Erklärung, was der User sucht. Die Studie geht davon aus, dass beispielsweise das Wort Fanclub eine navigationsorientierte Suchanfrage ist, während andere Anfragen

beispielsweise Primzahl oder die Frage „Was ist eine Primzahl?“ auf informationsorientierte Anfragen hinweisen (vgl. KANG UND KIM 2003, S. 64 und 67 ff).

Außerdem sind Kang und Kim der Meinung, die Kategorisierung von Suchanfragen nicht nur mittels Analyse der einzelnen Schlagwörter durchzuführen, sondern den gesamten Verlauf der Suchanfrage eines Nutzers zu betrachten (vgl. KANG UND KIM 2003, S. 70).

Den unterschiedlichen Bedürfnissen der Nutzer muss Beachtung geschenkt werden. Algorithmen für informationsorientierte Suchanfragen sind für navigationsorientierte Suchanfrage eher ungeeignet (vgl. KANG UND KIM 2003, S. 71.)

4.4.2 Rose und Levinson 2004

Auch Rose und Levinson bauen ihre Studie auf den Kategorien von Broder 2002 auf. Auch sie versuchen das Informationsbedürfnis hinter einer Suchanfrage zu analysieren, ohne jedoch die bisher verwendeten Faktoren, wie z.B. Anzahl der Worte in einer Suchanfrage, zu nutzen. Für sie ist wichtig, die Intension hinter einer Suchanfrage zu verstehen. Bank kann beispeilsweise für ein Geldinstitut oder eine Sitzbank stehen. Informationsorientierte Suchanfragen sehen Rose und Levinson als Informationssuche zu einem bestimmten Thema. Diese Suche zielt auf keine bestimmte Webseite ab, sondern nur auf die Erfüllung ihres Informationsbedürfnisses (vgl. ROSE UND LEVINSON 2004, S.14). Die Studie untersuchte Log Dateien von der Suchmaschine Alta Vista. Dabei wurden 1500 Suchanfragen analysiert. Die Studie kam zu dem Ergebnis, dass 40% der Suchanfragen nicht informationsorientiert waren. Weiterhin hatte ein Großteil der informationsorientierten Suchanfragen die Absicht, ein Produkt oder Dienstleistungen zu finden. Suchanfragen nach Produkten, oder Dienstleistungen sind typischerweise keine Wikipedia Anfragen. Etwas mehr als 35% der Anfragen hatten die Intention, sich über ein bestimmtes Thema zu informieren und sind somit typische Infomation Retrieval Anfragen, also Anfragen die primär auch Wikipedia Einträge als Ergebnis liefern (vgl. ROSE UND LEVINSON 2004, S.17). Bei Rose und Levinson treten Unterschiede zu Broder auf. Dies kann daran liegen, dass die Suchmaschine Alta Vista nicht repräsentativ für den Großteil von Suchanfragen ist, sondern meist bei komplizierten Sachverhalten verwendet wird

4.4.3 Lee et al. 2005

Auch in dieser Studie geht es darum, wie und ob man Suchanfragen klassifizieren kann. Die Intension einer Anfrage wurde mittels der Befragung von 28 Studierenden der „Departments Computerwissenschaften“ der Universität Los Angeles ermittelt.

Die Studenten ordneten die 50 beliebtesten Suchanfragen bei Google, den Kategorien informationsorientiert oder navigationsorientiert zu. Das Ergebnis der Befragung ist, dass die Intension, die hinter einer Suchanfrage steckt, vorhersagbar, also entweder der Kategorie informationsorientiert oder navigationsorientiert zuordenbar ist (vgl. LEE ET AL. 2005, S. 392). Einige Suchanfragen können jedoch nicht so leicht vorhergesagt werden. Dazu gehören Suchanfragen, wie z.B. Anfragen über Personen oder Software. Diese Anfragen müssen gesondert betrachtet werden und können nicht automatisch kategorisiert werden. Klicks, die in der Vergangenheit getätigt wurden und die Analyse der Anchor Link Verteilung, werden bei Lee et al als Möglichkeit gesehen, die Absicht hinter Suchanfragen vorherzusagen.

Bei navigationsorientierten Anfragen ist zu erkennen, dass die Suchenden eine bestimmte Website im Blick haben, weshalb bei den Suchergebnissen meist nur eine oder wenige Seiten angeklickt werden. Bei der informationsorientierten Anfrage hingegen, hat der Suchende meist keine bestimmte Seite im Blick, sondern besucht mehrere Internetauftritte bevor er sein Informationsbedürfnis befriedigt hat (vgl. LEE ET AL 2005, S. 394). Das heißt, auf unseren Fall übertragen, klickt der User sofort nach seiner Suche auf Wikipedia, handelte es sich wohl um eine navigations-orientierte Anfrage. Klickte der User jedoch mehrere Suchergebnisse an, bevor er auf Wikipedia landete, war die Suchanfrage wohl eher informationsorientiert. Das Ergebnis, nämlich der Besuch der Website Wikipedia, ist jedoch das Gleiche.

5 Kategorisierung von Suchanfragen nach Themen

In folgenden Studien werden Suchanfragen nach thematischen Gesichtspunkten unterteilt und untersucht, ob es Unterschiede bei Suchanfragen je nach Thema gibt.

5.1 Spink et al. 2001

Bei dieser Studie wurden mehr als eine Million Suchanfragen analysiert und ein Themengebiet zugewiesen. Dazu wurden Log Files der Suchmaschine Excite untersucht, die an einem Tag getätigt wurden(vgl. SPINK ET AL. 2001, S. 3). Neben den typischen Merkmalen, wie z.B. Länge der Anfrage, Anzahl der Wörter oder Nutzung von Operatoren, wurde den Suchanfragen auch ein Themengebiet zugewiesen (vgl. SPINK ET AL. 2001, S. 8 und S.11). Es wurde eine qualitative und

quantitative Untersuchung durchgeführt. Die Zuordnung in die einzelnen Klassen erfolgte durch Studenten der „Library and Information Science“.

Folgende elf Hauptklassen konnten identifiziert werden:

1. Entertainment, recreation
2. Sex, pornography, preferences
3. Commerce, travel, employment, economy
4. Computers, the Internet
5. Health, the sciences
6. People, places, things
7. Society, culture, ethnicity, religion
8. Education, the humanities
9. The performing and fine arts
10. Government
11. Unknown, incomprehensible

Es konnten weitere Unterklassen gebildet werden, die hier allerdings nicht beachtet wurden. Die meisten Anfragen konnten der Klasse „Entertainment recreation“ zugeordnet werden, gefolgt von der Kategorie „Sex, pornography, preferences“. Die wenigsten Anfragen wurden den Bereichen „Government“ und „Unknown“ zugeordnet (vgl. SPINK ET AL. 2001, S. 13).

5.2 Paper: „The search queries that took Australien Internet users to Wikipedia“(vgl. Vivienne Waller)

Dieses Papier analysierte Suchanfragen, die Nutzer auf Wikipedia führen. Hierfür wurden Daten analysiert, die vom australischen Web Anbieter Hitwise zur Verfügung gestellt wurden. Der Vorteil der Nutzung von Transaktion Log Dateien ist, dass eine große Anzahl von Suchanfragen analysiert werden kann und das Userverhalten wird durch die Analyse nicht beeinflusst. Transaktions Logs zeichnen auf, was die Nutzer wirklich tun, nicht das was Nutzer sagen was sie tun. Suchanfragen die offensichtlich beabsichtigen auf Wikipedia zu führen (*Wikipedia*, *wiki*, *Wikipedia encyclopaedia*, *site:en.Wikipedia.org*, and *www.Wikipedia.org*) gelten als navigationsorientierte Anfragen (vgl. Broder 2002). Weniger als zwei Prozent von allen Suchanfragen waren navigationsorientiert. All die anderen Suchanfragen über bestimmte Themen werden als informationsorientiert angesehen, auch wenn sie das Wort wiki enthalten. Diese Studie unterscheidet sich von der Studie Spoerri´s (2007), welche die 100 meistbesuchten Wikipedia Seite untersuchte, da hier die „Long Tail“ Verteilung beachtet wurde. Die „Long Tail“ Verteilung ist eine Zipf Verteilung. Diese besagt, dass beispielsweise 1,7% der Suchanfragen, Suchbegriffe verwenden, die 500 mal

oder öfter vorkommen. 61% der Suchanfragen machen Suchbegriffe aus, die nur einmal vorkommen. Die nachfolgende Grafik veranschaulicht diese Theorie.

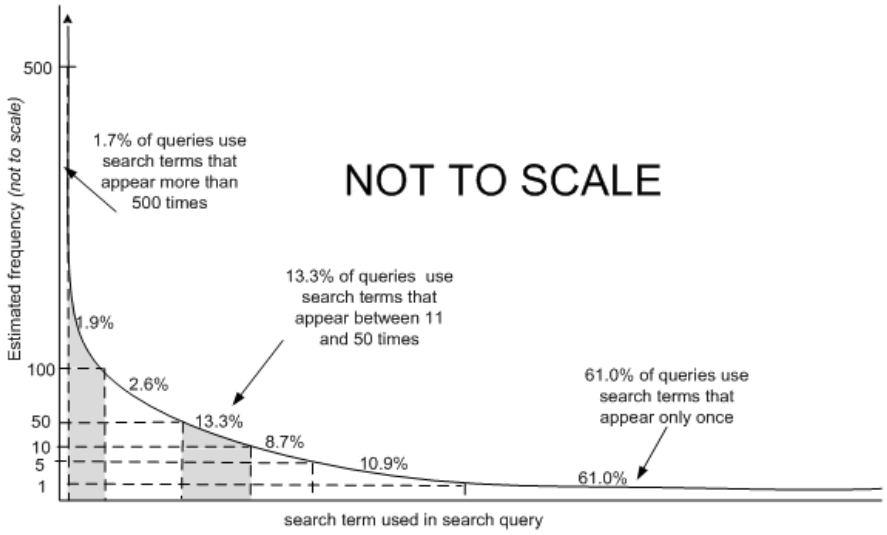


Abbildung 1: „Long Tail“ Verteilung der Suchanfragen auf Wikipedia

Nachfolgende Grafik zeigt die Themenverteilung an, welche die Nutzer auf Wikipedia führten:

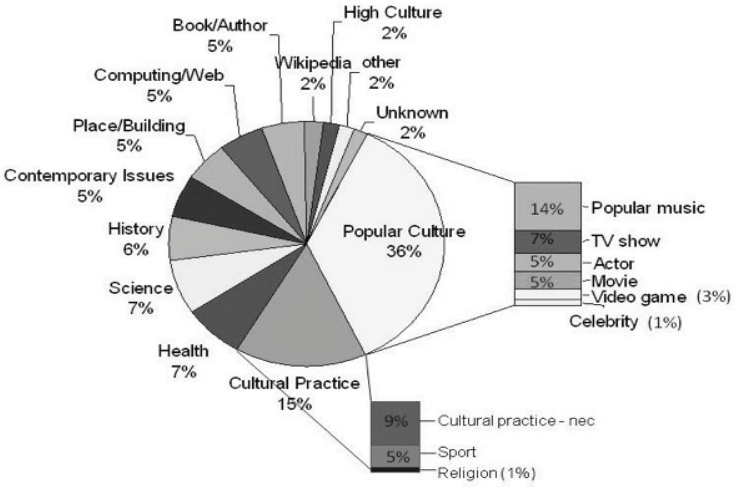


Abbildung 2:Verteilung der Themen die Suchmaschinen Nutzer zu Wikipedia führten

Bei Abbildung 2 wurde zwischen „High Culture“ und „Popular Culture“ unterschieden. Hoch Kultur umfasst nach Gans (1999) Anfragen nach Kunst und klassischer Musik. Der Begriff Populäre Kultur wurde für Anfragen nach, beliebter Musik, Fernsehsendungen, Schauspieler, Filme, Spiele und Prominente benutzt. Weniger als 2% der Anfragen, die User auf Wikipedia brachten, waren „High Culture“ Anfragen. Mehr als ein Drittel der Anfragen nahmen Bezug auf „popular Culture“. Danach folgen mit jeweils 7% Gesundheit, Wissenschaft und Geschichtsabfragen.

Weiterhin untersuchte die Studie, wer die Besucher von Wikipedia waren und aus welcher Gesellschaftsschicht diese kommen. Abbildung 3 zeigt die verschiedenen Besucherschichten an.

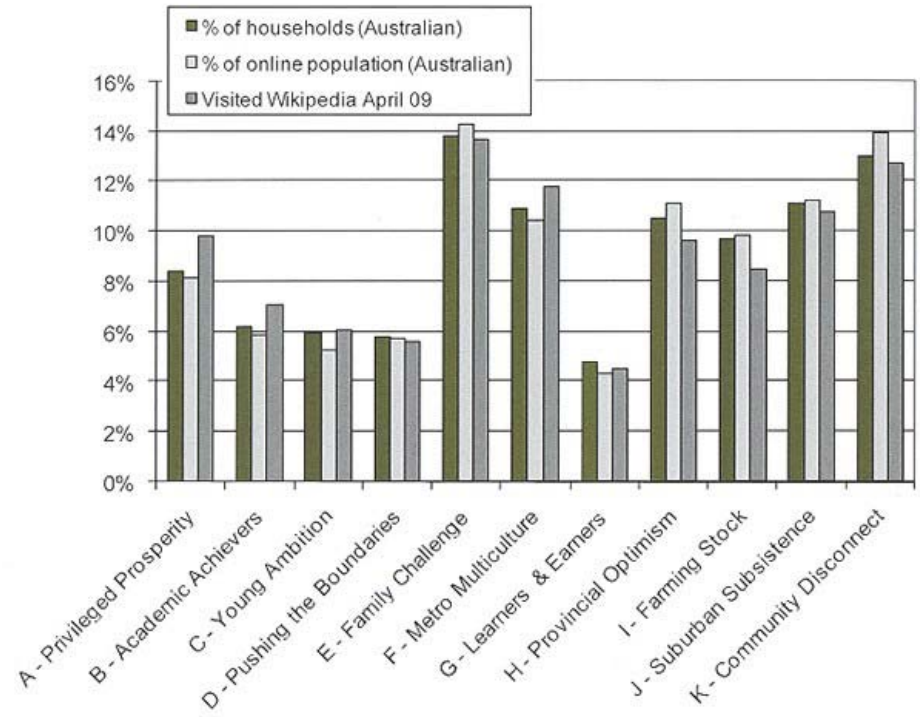


Abbildung 3: Lifestyle Profil der Besucher von Wikipedia, verglichen mit ihrer Repräsentation in der australischen Bevölkerung. (Quelle: Hitwise)

6 Qualitative Analyse Methoden

Die bisher angewandten Methoden bezogen sich meist auf die Analyse von Log Dateien. Diese Methoden lassen allerdings die emotionalen und kognitiven Aspekte eines Users außer Acht. Dies kann jedoch durch den zusätzlich Einsatz von qualitativen Methoden, wie beispielsweise Interviews, oder Befragungen ausgeglichen werden. Befragungen können helfen, statistisch erworbene Erkenntnisse zu stärken und möglicherweise neue Wege für die Log Dateien Analyse aufzeigen. Durch die Nutzung elektronischer Umfragen können die Kosten der qualitativen Methoden gering gehalten werden (vgl. Mach Sonja S.39)

7 Fazit

Es gibt verschiedene Wege typische Wikipedia Anfragen zu identifizieren. Die meisten Methoden stützen sich auf die Analyse von Log Dateien. Auf diesen Log Dateien können verschiedene Analyseverfahren angewandt werden. Eine Ausweitung der Log File Analyse stellt das sogenannte Web Mining dar, welches noch mehr Möglichkeiten der Analyse bietet. Mit Hilfe dieser zwei Verfahren kann versucht werden, Suchanfragen zu klassifizieren. Es gibt verschiedene Arten der Klassifikation. Entweder werden die Anfragen nach dem Informationsbedürfnis der Nutzer eingeteilt, oder man teilt den Suchanfragen verschiedene Klassen zu. Weiterhin gibt es qualitative Methoden, die ebenfalls beachtet werden müssen. Bei diesen Methoden werden User interviewt oder Online befragt. Auch diese Methode kann in Zusammenspiel mit quantitativen Methoden hilfreich sein, die Intension hinter Suchanfragen zu identifizieren. Alles in allem bieten sich in diesem Gebiet weitere Studien an, da die genaue Nutzung der Wikipedia noch nicht ausreichend erforscht ist (vgl. Vivienne Waller).

Literaturverzeichnis:

- **Vivienne Waller**
  The search queries that took Australian Internet User to Wikipedia
  http://informationr.net/ir/16-2/paper476.html; Zuletzt geprüft: 14.09.2012

- **Mach Sonja**
  Anforderungsanalyse für die Bereitstellung von Suchanfragedaten: Literaturanalyse und Empfehlungen für die Aufbereitung von Logfiledaten für weitere Studien; S.5-47; http://opus.haw-hamburg.de/volltexte/2011/1307/pdf/Mach_Sonja_von_110219.pdf
  Zuletzt geprüft: 13.09.2012

- **Broder 2002**
  Broder, Andrei: *A taxonomy of web search*. In: SIGIR Forum 36 (2002), Nr. 2., S. 3-10 – Quelle: http://www.acm.org/sigir/forum/F2002/broder.pdf
  Zuletzt geprüft: 14.09.2012

- **Wikipedia:**
  - http://de.wikipedia.org/wiki/Logdateianalyse
  - http://de.wikipedia.org/wiki/Web_Mining
  - http://de.wikipedia.org/wiki/Logdatei

- **Kang und Kim 2003**
  Kang, In-Ho ; Kim, GilChang: *Query Type Classification for Web Document Retrieval*.http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.58.5400&rep=rep1&type=pdf Zuletzt geprüft: 14.09.2012

- **LEE ET AL. 2005**
  Lee, Uichin; Liu, Zhenyu; Cho, Junghoo : *Automatic Identification of User Goals in Web Search*. In: WWW '05 Proceedings of the 14th international conference on World Wide Web (2005), S. 391 – 400 Online Quelle: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.59.3810&rep=rep1&type=pdf Zuletzt geprüft: 14.09.2012

- **Rose and Levison 2004**
  Rose, Daniel E. ; Levinson, Danny: *Understanding User Goals in Web Search* Zuletzt geprüft: 14.09.2012

- **SPINK ET AL. 2001**
  Spink, Amanda ; Wolfram, Dietmar ; Jansen, Bernhard J. ; Saracevic, Tef-ko: *Searching the web* : Thepublic and their queries. Quelle: http://comminfo.rutgers.edu/~tefko/JASIST2001.pdf Zuletzt geprüft: 14.09.2012

- Rainer Hammwöhner, Karl-Peter Fuchs, Markus Kattenbeck, Christian Sax, Qualität der Wikipedia, http://epub.uni-regensburg.de/15565/1/isi_2007.pdf Zuletzt geprüft: 14.09.2012

- http://www.e-teaching.org/didaktik/qualitaet/logfile/

- Markus Bischoff, Visualisierung von Ontologienutzungsdaten http://www.umingo.de/doku.php?id=paper:visualisierung_ontologienutzungsdaten:section022 zuletzt geprüft: 14.0.2012

Sphinx - Open search server

Manuel Leithner

Otto-Friedrich-Universität Bamberg, Lehrstuhl für Medieninformatik

Zusammenfassung. Diese Seminararbeit zum Search Server Sphinx gliedert sich zu Beginn in eine Einführung zu dessen Geschichte und Grundlagen, wobei bereits hier auf die Architektur des Suchsystems eingegangen wird. Eine exemplarische Anwendung, anhand der die Installation und Grundkonfiguration von Sphinx veranschaulicht werden soll, führt tiefer in das Thema. Weitere Beispiele erläutern zudem die Definition von Datenquellen und Suchindizes sowie deren verschiedene Typenausprägungen. Anschließend widmet sich ein Kapitel der Suche mit Sphinx. Hierbei kommt es vor allem auf unterschiedliche Suchmodi, Operatoren und das Resultset-Processing an. Eine Einführung in die Java-API sowie eine Bewertung des Ganzen schließen diese Arbeit ab.

Schlagworte: Sphinx, Installation, Konfiguration, Search Server, Java

1 Google und die Suchserver

Spätestens seit dem Start der Onlinesuchplattform Google im Jahr 1998 stellt sich vielen Entwicklern die Frage, wie man effizient in großen Datenmengen suchen soll. Zwar helfen bei kleinen Datenmengen noch die aus Datenbanksystemen bekannten SQL-Suchfunktionen wie *LIKE* oder *MATCH*. Sobald es aber um Volltextsuche im Terrabyte-Bereich geht, stoßen auch diese schnell an ihre Grenzen bzw. sind auf Grund der langen Berechnungsdauer unzureichend. Deshalb wurden sogenannte Searchframeworks wie beispielsweise das Apache-Projekt Lucene[1] ins Leben gerufen. Jene generieren aus einem bekannten Datenstamm einen sogenannten Index, auf dem relativ effizient Suchmethoden ausgeführt werden können.

Apache Lucence und *Apache Solr*[2] sind die wohl bekanntesten freien Suchserver. Mit *Sphinx* betritt nun ein weiterer Vertreter dieser Gattung die Bühne der Suchframeworks.

2 Sphinx Search Server

Diese Seminararbeit stellt das Open-Source Projekt Sphinx vorgestellt werden. Eine Einführung in das Thema erläutert einige Grundlagen, bevor eine Anleitung

[1] http://lucene.apache.org/core
[2] http://lucene.apache.org/solr

die Installation unter Windows veranschaulicht. Im Folgenden werden die einzelnen Bestandteile von Sphinx genauer erklärt.

2.1 Geschichte und Grundlagen

Mit Sphinx stellte *Andrew Aksyonoff* im Jahr 2001 erstmalig den von ihm entwickelten *Open Search Server*[3] der Öffentlichkeit vor. Elf Jahre später wurde im Juli 2012 die bislang letzte Version 2.0.5 zum Download[4] angeboten. Auch diese ist mit der *GPLv2-Lizenz* geschützt, welche es ermöglicht Sphinx in nichtkommerziellen Projekten kostenfrei zu verwenden. Erst beim kommerziellen Einsatz ist der Kauf einer Lizenz notwendig. Hierfür wurde im Jahr 2007 von Entwickler *Andrew Aksyonoff* und dem Datenbank-Experten *Peter Zaitsev* die Firma *Sphinx Technologies Inc.* gegründet. Diese ist erster Ansprechpartner für Lizenz- bzw. Support-Angelegenheiten und beschäftigt derzeit ca. 10 Mitarbeiter [1].

Entwickelt in C++, ist Sphinx derzeit auf den verschiedensten Plattformen verfügbar. Darunter finden sich Microsoft Windows, Linux, FreeBSD, Solaris und Apples MacOS. Für den Datenzugriff und -import wurden eine Vielzahl an Schnittstellen geschaffen. Neben einem Interface für Datenbanken wie *MySQL*[5] und *PostgreSQL*[6] unterstützt Sphinx ebenfalls Datenbanken, welche ODBC-kompatibel sind. Hier wären z.B. die Datenbank-Server von Microsoft bzw. Oracle zu nennen. Aber auch andere Datenquellen bleiben nicht außen vor. Mit dem Sphinx-eigenen XML-Format *xmlPipe* können auch unstrukturierte Daten wie MP3s, Bilder oder Office-Dateien importiert werden.

Für den Zugriff auf den Sphinx-Server stehen dem Entwickler mehrere Möglichkeiten zur Verfügung. *SphinxAPI* bezeichnet für eine native Programmbibliothek verschiedenster Klassen und Methoden, die den direkten Zugriff erlaubt und eine Kommunikation mit Sphinx herstellt. Anfangs in PHP[7] entwickelt, steht die API in der aktuellen Version in mehr als 10 Programmiersprachen zur Nutzung bereit. Darunter gängige Sprachen wie C/C++, Java, C#, Ruby oder Python.

SphinxQL ist die SQL-Schnittstelle der Search-Engine. Hiermit wird es möglich, direkt per SQL-Befehl eine Suche zu starten. Dazu wurde der SQL-Standard implementiert und durch Sphinx-eigene Befehle erweitert, um eine vollständige Unterstützung aller Features per SQL-Syntax zu gewährleisten [2].

Seit einiger Zeit ist nun mit *SphinxSE* eine weitere Schnittstelle hinzugekommen. Hierbei handelt es sich um ein Plugin für den MySQL-Server, das es erlaubt Anfragen an Sphinx direkt aus MySQL heraus zu stellen, um die erzeugten Ergebnisse darauffolgend im MySQL-Server weiter zu verarbeiten bzw. abzulegen.

[3] abgeleitet aus den Begriffen Open-Source und Search-Server
[4] http://sphinxsearch.com/downloads
[5] http://www.mysql.com
[6] http://www.postgresql.org
[7] http://www.php.net

Der von Sphinx erstellte Index basiert auf einem im Information-Retrieval bekanntem *Inverted File*. Dieses besteht aus einem Wörterbuch aller vorkommenden Keywords, einer Liste von Dokumenten-Ids sowie einer Liste der Positionen aller Keywords. Zusätzlich werden in diesem die vom Nutzer vorgegebenen Attribute eines jeden Dokuments gespeichert. Der erstellte Index ist etwa auf 70-80% der Originalgröße komprimiert. Zwar wäre auch eine höhere Komprimierung möglich, jedoch entschied man sich aufgrund der längeren Dekomprimierung bei Suchvorgängen für eine eher geringer Dichte.

2.2 Architektur und Leistungsfähigkeit

Zur Veranschaulichung des Search-Servers zeigt Abbildung 1 den Aufbau und die Kommunikationswege eines Suchsystems. Wie man erkennen kann, greift eine Anwendung sowohl auf die Sphinx-Instanzen als auch auf andere Datenbanksysteme zu. So werden z.B. bei einem Foren-System per Sphinx alle zu einem Suchbegriff passenden Beiträge herausgefunden und über die Datenbanksysteme zusätzliche Informationen wie Beitragsautor, Datum oder Querverlinkungen angefordert.

Die Suche über Sphinx selber ist in zwei Schritte aufgeteilt. Zum einen die Indizierung, die vom sogenannten *indexer* durchgeführt wird. Hierbei fordert jener alle notwendigen Daten aus den jeweiligen Datenquellen an und bildet daraus einen Index. Zum anderen erledigt *searchd*[8] die eigentliche Suche auf dem Index. Sobald eine Anfrage an *searchd* gestellt wird, bildet dieser ein *Resultset* und überträgt es an die aufrufende Instanz [3].

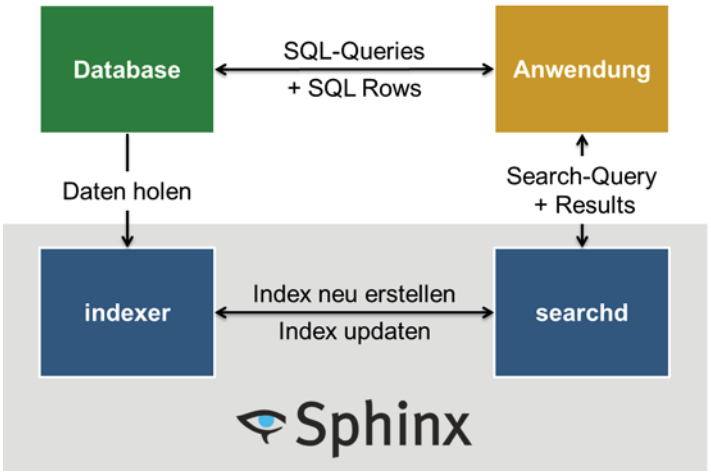


Abb. 1. Kommunikation zwischen den Schichten

[8] Das "d" am Ende steht für daemon

Die Bildung von Clustern, sowohl beim *indexer* als auch auf *searchd*-Seite, ermöglicht eine sehr hohe Indizierungs- bzw. Suchperformance. So sind Indizierungsgeschwindigkeiten von über 60MB pro Sekunde pro Indizierungsserver möglich. Mit über 500 Queries pro Sekunde werden bei einem Datenbestand von 1.5 GB ebenfalls hohe Suchgeschwindigkeiten erreicht. Beispiele aus der Praxis zeigen zusätzlich die Steigerung der Performance durch die Bildung von Clustern. Die amerikanische Top10-Webseite *Craigslist.com* betreibt ein Sphinx-Cluster mit 15 Instanzen und verarbeitet so 200 Millionen Queries pro Tag, was wiederum über 2000 Queries pro Sekunde entspricht. Am Beispiel *Broadreader.com* lässt sich erkennen, dass Sphinx auch riesige Datenbestände keine Probleme bereiten. So verarbeiten hier 37 Sphinx-Instanzen über *fünf Terrabyte* Daten[1].

2.3 Beispiel-Anwendung Film-Datenbank

Im Folgenden wird nun anhand eines Beispiels die Installation und Konfiguration eines Sphinx-Servers erläutert. Die Konsolenanwendung soll die Suche durch eine Filmdatenbank ermöglichen. Hierzu wurde ein Datenbank-Dump der bekannten Webseite IMDB[9] herangezogen und so bearbeitet, dass die Daten wieder in einer MySQL-Datenbank abgelegt werden können. Abbildung 2 zeigt

```
Sphinx-IMDB Search-Client started ...
Type in -h or -help for more information
-h
################################## Sphinx-Help ######################################
#                                                                                   #
# -exit, -quit                  Quits program                                       #
# -help, -h                     Shows this help-messages                            #
#                                                                                   #
# -search <searchTerm*>         Searches for a spezific term                        #
# -custom <searchExpr>          Search using a custom expression                    #
# -match <mode>                 Matching Mode (Phrase, Boolean, All, Any)           #
# -index <index*>               Sets spezific Indizes to be searched                #
# -limit <offset> <limit>       Limits the result item count                        #
# -reset                        Resets default search configuration                 #
#                                                                                   #
#####################################################################################
-limit 0 20
Set new limits to 0 - 20
-match All
Set new matching mode 'All'
-search James Bond 007
Searching for 'James Bond 007 ' on indices '*'...
Query retrieved 23 of 23 matches in 0.0 sec.
Query stats:
        'james' found 4169 times in 2675 documents
        'bond' found 1459 times in 1147 documents
        '007' found 81 times in 39 documents

Matches:
    1. id=1307612, weight=3, title="Troldspejlet" (1989) {{#29.1}}
    2. id=1421694, weight=3, title=007: Agent Under Fire (2001) (VG)
    3. id=1539792, weight=3, title=Casino Royale (1967)
    4. id=1599765, weight=3, title=Die Another Day (2002)
    5. id=1615639, weight=3, title=Dr. No (1962)
```

Abb. 2. Konsolenanwendung zur Filmsuche

die Anwendung und deren Funktionen. So wird es möglich sein nach verschiedenen Termen zu suchen, verschiedene Indizes zu durchsuchen, die Anzahl der

[9] http://www.imdb.com

Suchergebnisse einzuschränken und den Treffer-Modus einzustellen. Das Ergebnis stellt die einzelnen Resultate mit Film-Id, Gewichtung und Titel in Textform dar. Zunächst soll jedoch die Installation des Sphinx-Servers unter Windows aufgezeigt werden.

2.4 Installation unter Windows

Für die Installation unter *Microsoft Windows*[10] stehen verschiedene Binaries im Download-Bereich der offiziellen Webseite[11] zur Verfügung. In dieser Beispielinstallation wird das 64-Bit Paket mit MySQL, PostgreSQL und Stemmer-Bibliothek verwendet. Nach dem erfolgreichen Download werden die Dateien beispielsweise in das Verzeichnis

D:/Sphinx

extrahiert[12]. Um Sphinx ausführen zu können, müssen nun die Pfade in der Config-Datei angepasst werden. Hierzu wird die Datei

D:/Sphinx/sphinx-min.conf.in

geöffnet und alle Vorkommnisse des Strings

@CONFDIR@ durch *D:/Sphinx*

ersetzt. Das abschließende Speichern und Schließen der Datei beendet vorerst die Konfiguration von Sphinx. Im späteren Verlauf werden weitere Einstellungen in dieser Datei vorgenommen. Für den erstmaligen Start des Servers genügen jedoch diese Grundanpassungen.

Es empfiehlt sich Sphinx als Windows-Service zu registrieren. Um dies durchzuführen, muss die Kommandozeile als Adminsitrator aufgerufen und der folgende Befehl ausgeführt werden:

D:/Sphinx/bin/searchd.exe –install –config D:/Sphinx/sphinx-min.conf.in –servicename SphinxSearch

Die Basis-Installation und Konfiguration ist damit abgeschlossen. Als Voraussetzung für das folgende Beispiel muss ebenfalls ein MySQL-Server[13] installiert sein. Die Datenbank-Setupscripte finden sich im Anhang A. Der komplette SQL-Dump benötigt bei diesem Beispiel ca. 1.5 Gigabyte Speicher und kann somit nicht direkt in dieser Seminararbeit abgelegt werden. Es können jedoch beliebige Datensätze verwendet werden, solange das SQL-Schema nicht verändert wird.

[10] Im Beispiel wird Windows 7 verwendet
[11] http://sphinxsearch.com/downloads/release
[12] z.B. mit WinZip oder 7Zip
[13] Ein vorkonfigurierter Server ist auf der Seite http://apachefriends.org herunterladbar

2.5 Configuration-File

Für die Erzeugung eines Indizies und die Suche in jenem wird die bereits angesprochene Config-Datei erweitert. Zunächst soll jedoch der Aufbau einer solchen Datei erläutert werden.

Aufbau Die komplette Konfiguration einer Sphinx-Instanz geschieht über das Config-File (siehe Anhang B) im Installations-Ordner. Jenes ist in vier Abschnitte eingeteilt. Bereich 1 und 2 konfiguriert die Datenquellen und Indizes. Die Bereiche 3 und 4 sind für Einstellungen des *indexers* und des *searchd* verantwortlich. Im Testsystem sind die Standardwerte ausreichend. Deshalb wird im weiteren Verlauf nicht weiter auf die verschiedenen Konfigurationsmöglichkeiten eingegangen. Für weitere Informationen steht die offizielle Seite[1] mit einer ausführlichen Dokumentation zur Verfügung.

Vererbung und Scripting Zur einfachen Erweiterung und Wartung ist es möglich, die aus der Objektorientierung bekannte Vererbung in der Konfigurationsdatei einzusetzen. Hierzu werden zunächst allgemeingültige Einstellungen, wie z.B. eine Verbindung zu einer Datenbank, definiert. Per Vererbung können dann mehrere Datenquellen von dieser Verbindung erben. Somit es ist nicht nötig die Verbindungsdaten für jede Datenquelle zu definieren. Dies erspart Zeit und vermindert die Fehleranzahl.

Ein weiteres Feature ist die Scriptunterstützung. So kann zum Beispiel ein Script in das Config-File eingebunden werden, welches beim Parsen durch Sphinx ausgeführt wird. Dadurch ist es möglich Datenquellen oder Indizes dynamisch zu genieren. Listing 1.1 zeigt die Erzeugung von fünf Datenquellen per PHP-Script.

```
1 #!/usr/bin/php
2 <?php for($i=1; $i<=4; $i++) { ?>
3   source dataSource<?=$i=?> {
4   ...
5   }
6 <?php } ?>
```

Listing 1.1. Config-File - Scripting

Zeile 1 definiert in diesem Fall die Ausführung des PHP-Interpreters auf einem Linux-Server. Es sind jedoch auch andere Interpreter oder Programme (z.B. ShellScripte, Python, ...) verwendbar, sofern der jeweilige Interpreter installiert ist.

2.6 Datenquellen

In der Grundkonfiguration wird für das Beispiel eine Datenquelle mit dem Namen *localDatabase* angelegt. Listing 1.2 zeigt die Definition einer MySQL-Datenquelle. Neben dem Datenbanktyp *mysql* werden *Host*, *User*, *Passwort*, *Datenbank* und *Port* angegeben. Optional wäre es auch möglich die IP des Servers

anzugeben. Da es sich beim Testsystem jedoch um eine lokale Installation handelt, kann dieser Eintrag vernachlässigt werden. Sphinx nimmt bei fehlenden Einträgen die Defaultwerte (hier localhost bzw. 127.0.0.1) an.

```
1 source localDatabase {
2   type          = mysql
3   sql_host      = localhost
4   sql_user      = root
5   sql_pass      =
6   sql_db        = seminar2012
7   sql_port      = 3306
8 }
```

Listing 1.2. Config-File - Source-Definition

Somit ist eine Verbindung zur Beispieldatenbank hergestellt, mithilfe der nun eine Datenquelle über das Konstrukt der Vererbung für den Film-Index definiert werden kann. Hierfür stellt Sphinx verschiedene Query-Methoden bereit, die zu verschiedenen Zeitpunkten der Indizierung zum Einsatz kommen.

Main data-fetching Query Alle Datenreihen, die über diesen Query empfangen werden, sind Bestandteil des von Sphinx generierten Indizes. Somit stellt der *Main data-fetching Query* den wichtigsten Teil der Indizierung dar und muss in der Konfigdatei angegeben sein. Sphinx unterstützt hierbei jede Art von SQL-Abfragen. So können auch komplexe Anfragen mit *JOINS* oder *LIMITS* angegeben werden. Dabei ist nur zu beachten, dass die erste zurückgegebene Spalte jeweils die *ID* des indizierten *Dokuments* beschreibt. Somit sollte dieses Element eindeutig sein bzw. eine fortlaufende Nummer haben. Für das Beispiel zeigt Listing 1.3 in Zeile 4 eine mögliche SQL-Abfrage.

Pre-Query, Post-Query & Post-Index Query Mit Hilfe dieser Queryarten können vor- bzw. nachbereitende Aufgaben erledigt werden. Im *Pre-Query* können z.B. Einstellungen wie beispielsweise UTF-8 als Ausgabeformat durchgeführt werden. Ebenso ist es möglich hier Markierungen der zu indizierenden Datensätze zu setzen. Dies ist z.B. nötig, wenn nicht immer die komplette Datenbank neu indiziert werden soll, sondern ausschließlich die neuen Elemente. Listing 1.3 zeigt bereits wie eine solche *Delta-Indizierung* konfiguriert wird. Kapitel 2.8 geht auf dieses Thema jedoch noch einmal genauer ein.

```
1 source movieSource : localDatabase
2 {
3   sql_query_pre  = SET @maxts:= (SELECT NOW())
4   sql_query      = SELECT * FROM movies WHERE ts<@maxts
5   sql_query_post = REPLACE INTO movies_helper
                     VALUES ("movies_tmp", @maxts)
6   sql_query_post_index = DELETE FROM movies_helper
                           WHERE tablename="movies"
```

```
7    sql_query_post_index = UPDATE movies_helper
                            SET tablename="movies"
                            WHERE tablename="movies_tmp"
8    sql_query_post_index = DELETE FROM movies_klist WHERE
                             ts<(SELECT maxts FROM
                             movies_helper WHERE
                             tablename="movies")
9    # Document attribute
10   sql_attr_string   = title
11 }
```

Listing 1.3. Config-File - Query-Definition

Post-Query bzw. *Post-Index Query* werden ausgeführt, sobald die Daten vom *Main data-fetching Query* empfangen, bzw. sobald die Indizierung abgeschlossen wurde. Bei der Konfiguration der verschiedenen Queries ist zu beachten, dass *Pre-Query* und *Post-Query* in derselben Datenbank-Verbindung wie der *Main data-fetching Query* ausgeführt werden. *Post-Index Query* hingegen wird in einer neuen seperaten Verbindung ausgeführt. Dies ist durch mögliche Verbindungs-Timeouts begründet und sollte bei der Konfiguration im Auge behalten werden.

Attribute bezeichnen Zusatzinformationen, die direkt im Index abgelegt werden können. Listing 1.3 definiert als einziges Attribut den *Title* des Films als Typ *String*. Somit kann dieser bei der Suche direkt im Result-Set mit ausgegeben werden. Dies hat den Vorteil, dass jene Information direkt verfügbar und keine zusätzliche Anfrage an die originale Datenbank nötig ist. Prinzipiell wäre hier eine 1:1 Kopie der Datenquelle möglich, da mehrere verschiedene Datentypen definiert werden können. Als Datentypen stehen beispielsweise Nummernformate wie *Integer*, *BigInt*, oder *Float* bereit. Neben *Bits*, *Boolean* oder *Timestamps* können auch multidimensionale *Multivalues* abgelegt werden. Es ist jedoch ratsam nur wirklich wichtige Daten eines Dokumentes als Attribute abzulegen, um den Index nicht unnötig zu vergrößern, da darunter natürlich auch die Suchperformance leidet.

2.7 Index-Typen

Um mit Sphinx suchen zu können, müssen neben der nun definierten Datenquelle auch ein oder mehrere Indizes konfiguriert werden. Hierfür stehen dem Nutzer drei verschiedene Arten zur Auswahl.

Disk-based Index definiert einen *On-Disk* Index, dessen Daten auf der Festplatte abgelegt werden. Updates erfordern einen Neuaufbau des Index (*Non-Incremental*). Zur Definition eines solchen Indizes müssen verschiedene Einstellungen in der Konfigurationsdatei vorgenommen werden. Listing 1.4 zeigt eine Beispieleinstellung.

```
index movieIndexDisc
{
  source        = movieSource
  path          = D:/Sphinx/data/movies
  charset_type  = sbcs
  stopwords     = D:/Sphinx/stopword.txt
  min_word_len  = 3
}
```

Listing 1.4. Config-File - Disk-based Index

Hierbei wird zunächst die Datenquelle und der Pfad auf der Festplatte definiert. Zusätzlich kann noch ein Zeichenformat (hier Single Byte Character Set), eine Liste von Stopwords und eine minimale Wortlänge angegeben werden. Die letzten beiden Einstellungen limitieren die zu indizierenden Wörter. Alle Vorkommnisse der Stopwords-Liste sowie Wörter kürzer als drei Zeichen werden nicht indiziert.

Des weiteren könnten noch zusätzliche Features, wie der Support für HTML[14], erweitertes Stemming[15] oder SubString-Indexing eingefügt werden. Diese sind jedoch für das Beispiel nicht notwendig. Auch hier sei wieder auf die offizielle Dokumentation verwiesen.

RT-Index Sofern sich die Datenmenge in Grenzen hält, ist auch ein sogenannter *Real-Time Index* möglich. Dieser besitzt keine Datenquelle, sondern wird per *Push-Prinzip* von der Anwendung gefüllt. Daher ist jener auch *incrementell* aktualisierbar und erfordert keine Neugenerierung bei einem Update.

Wie der Name vermuten lässt, ist der *RT-Index* im Gegensatz zum *Disk-based Index* hauptsächlich im RAM des Servers abgelegt (*In-Memory*). Sofern die Größe des RAMs nicht ausreicht, werden die restlichen Daten ebenfalls auf der Festplatte abgelegt. Deshalb spricht man beim *RT-Index* auch von einer hybriden Speicherung. Listing 1.5 zeigt die etwas komplexere Definition eines RT-Indizes.

```
index movieIndexRT
{
  type          = rt
  rt_mem_limit  = 32M
  path          = D:/Sphinx/data/moviesRT
  charset_type  = utf-8
  rt_field      = title
  rt_field      = content
  rt_attr_uint  = gid
}
```

Listing 1.5. Config-File - RT-Index

[14] ermöglicht die Indizierung von HTML-Seiten ohne HTML-Tags
[15] bezeichnet die Findung von Wortursprüngen

Neben dem Typ, dem Pfad und dem Zeichenformat müssen beim RT-Index zusätzlich noch eine *Junk-Größe* sowie die zu indizierenden Felder und Attribute angegeben werden. Der Junk-Wert definiert die Größe des Indexteils, der im Speicher gehalten wird. Bei Indexgrößen von mehreren Giga- bzw. Terrabyte sollte diese Größe natürlich im dreistelligen Megabyte- bzw. im Gigabyte-Bereich liegen, da der Index in Stücke der hier angegeben Größe aufgeteilt wird. Wie bereits angesprochen generiert Sphinx den RT-Index nicht aus einer Datenquelle. Deshalb müssen die Felder und Attribute der Dokumente hier zwingend angegeben werden, damit Sphinx eine optimale Struktur erstellen kann [4]

Distributed Index bezeichnet einen hybriden Index, der eine Kombination der beiden bereits vorgestellten Typen darstellt. So können damit verschiedene Indizes zusammengefügt werden, welche evtl. über mehrere Server-Instanzen verteilt sind. Listing 1.6 stellt die Zusammenführung der bereits erläuterten Indizes mit einem, auf einem entfernten Server existierenden Index dar.

```
1 index movieIndexDist
2 {
3   type  = distributed
4   local = movieIndexRT
5   local = movieIndexDisc
6   agent = 192.168.2.20:9312:serverIndex
7 }
```

Listing 1.6. Config-File - Distributed Index

Sofern eine Anfrage an diesen Index abgeschickt wird, sucht Sphinx parallel sowohl auf den beiden lokal definierten Indizes (movieIndexRT und movieIndexDisc) als auch auf dem Index des Server mit der IP 192.168.2.20.

2.8 Indizierung

Als Indizierungsmöglichkeiten stehen dem Nutzer zwei verschiedene Arten zur Auswahl. Zum einen die *Real-Time-Indizierung*, wie sie beim *RT-Index* anzutreffen ist. Für diese sind keine zusätzlichen Einstellungen in der Konfiguration notwendig und sie wird im Folgenden nicht weiter behandelt. Zum anderen die *Batch-Indizierung*, welche beim *Disk-based Index* zum Einsatz kommt. Diese Indizierungsart kann ebenfalls in zwei Alternativen ausgeführt werden. *Full-Reindexing* generiert den Index von Grund auf neu. Diese Vorgehensweise ist jedoch nur bei kleinen Datenbeständen empfehlenswert, da dies trotz der hohen Geschwindigkeit von ca. 60MB pro Sekunde bei Datenbeständen im Terrabyte-Bereich mehrere Stunden in Anspruch nimmt. Deshalb stellt Sphinx auch die Möglichkeit des *Delta-Reindexing* bereit. Hierbei werden nur die neuen Daten in einen zusätzlichen Index aufgenommen und anschließend mit dem Originalindex gemerged. Listing 1.3 zeigt die Konfiguration des Hauptindex. Für die Aktivierung des Delta-Index muss in der Konfigurationsdatei eine weitere

Datenquelle, wie sie in Listing 1.7 zu sehen ist, eingetragen werden. Hierfür wird der Zeitpunkt der letzten Indizierung in einer Variable gespeichert und nur Datensätze, welche nach diesem Zeitstempel eingetragen wurden, indiziert. Mit Hilfe einer *Killlist* werden gelöschte oder aktualisierte Elemente aus dem Originalindex gelöscht.

```
1 source movieSourceDelta : localDatabase
2 {
3   sql_query_pre   = SET @maxts:= (SELECT maxts FROM
        movies_helper WHERE tablename="movies")
4   sql_query     = SELECT * FROM movies WHERE ts>=@maxts
5   sql_query_killlist  = SELECT id FROM movies WHERE ts>=
        @maxts UNION SELECT id FROM movies_klist
6 }
```

Listing 1.7. Config-File - Distributed Index

Zur Integration des Deltaindex in den Hauptindex kommen nach dem Aufbau des Deltaindex die Konzepte des *Index-Mergings* und der *Index-Rotation* zum Einsatz. Ersteres verschmilzt eine Kopie des Hauptindex mit dem neu gebauten Delta. Die Index-Rotation ist notwendig, um einen Stopp von Sphinx zu verhindern. Da Sphinx den Index, der zur Suche verwendet wird, beim Start mit einem *Filelock* versiegelt, müsste beim Wechsel auf einen neuen Index die Anwendung beendet werden. Durch das Prinzip der Rotation baut der *Sphingx-indexer* einen neuen zweiten Index auf und speichert diesen unter dem Indexnamen mit der Dateiendung *.new.sp*. Sobald die Indizierung abgeschlossen ist, sendet dieser ein *SIGHUP* Signal an den *searchd*. Je nach Konfiguration wartet Sphinx, bis keine Suchanfragen mehr an den aktuellen Index gestellt werden und wechselt dann den Index per Umbenennung und Löschen des alten Index (*Non-Seamless rotation*). Zwischenzeitliche Anfragen an Sphinx generieren eine Fehlermeldung, dass aktuell der Dienst nicht verfügbar ist.

Seamless rotation hingegen läd parallel den neuen Index und routed neue Anfragen auf diesen. Bestehende Anfragen auf den alten Index werden abgearbeitet und erst danach wird jener Index gelöscht.

Um nun das Gezeigte in die Praxis umzusetzen, muss zunächst ein Index erstellt werden. Hierfür wird der Sphinx-Indexer per Konsole aufgerufen:

indexer movieIndexDisc

Nach erfolgreicher Indizierung steht der Index zur Suche bereit. Sofern sich nun im Laufe der Zeit Daten ändern, kann in regelmäßigen Abständen (im Livesystem meistens per Cronjob) ein Deltaindex über den selben Befehl (Indexname: movieSourceDelta) generiert werden. Das Merging und die Rotation des Index binden jenen in das Live-Suchsystem ein. Hierfür muss der folgende Befehl ausgeführt werden:

indexer —merge <movieIndexDisc><movieSourceDelta>—rotate

2.9 Exceptions, Stopwords & Shortwords

Bevor nun die eigentliche Suche begonnen werden kann, sollen noch weitere hilfreiche Features vorgestellt werden. Um auch beispielsweise Informatiktexte indizieren zu können, unterstützt Sphinx sogenannte *Exceptions*. Hierbei handelt es sich um eine Liste von Ausnahmebehandlungen für Wörter wie "C++" oder ähnlichem, welche ohne diese nicht in den Index finden würden, da "++" Sonderzeichen sind. Sobald Exceptions definiert sind, können auch jene Texte korrekt indiziert werden.

Zur Beschleunigung und Verkleinerung des Index gibt es die Konzepte der *Stop-* und *Shortwords*. Die Konfiguration der beiden Features ist bereits im Listing 1.4, dort in den Zeilen 6 und 7 aufgezeigt. Um die Liste der Stopwords jedoch nicht manuell schreiben zu müssen, bietet Sphinx den Befehl *"indexer –buildstops stopword.txt N"* zur automatischen Stopwordgenerierung an. Die Angabe N=10 würde hierbei die 10 häufigsten Wörter des Index in die Datei *stopword.txt* schreiben.

2.10 Suche

Da die Basis-Konfiguration abgeschlossen ist, können die ersten Suchanfragen an den *searchd* abgegeben werden. Folgendes Kapitel stellt auch hier die Möglichkeiten des Anwenders zur Eingrenzung der Suchergebnisse dar.

Matching-Modes bezeichnen die Alternativen, wie ein Treffer gefunden wird. Sphinx definiert hierbei vier verschiedene Hauptarten, welche jedoch nur per SphinxAPI und SphinxSE angesteuert werden könnnen. SphinxQL bleibt hier aufgrund der SQL-Syntax außen vor.

Als Standard-Modus ist *ALL* festgelegt, womit nur Ergebnisse gefunden werden, welche *ALLE* übergebenen Schlüsselwörter enthalten. Beispielsweise werden beim Suchstring "James Bond 007" nur Datensätze geliefert, die alle drei Wörter enthalten. Das finale Ranking, also die Reihenfolge der Ergebnisse, hängt hier von der Ähnlichkeit sowie von der Häufigkeit der Schlüsselwörter ab.

Eine weitere Möglichkeit wäre das *ANY*-Matching. Wie der Name vermuten lässt, müssen hier nicht alle Worte vorkommen. Es genügt wenn mind. ein Wort enthalten ist. Auch hier wird der Treffer mit der höchsten Ähnlichkeit als erstes gelistet.

Als *PHRASE*-Matching bezeichnet Sphinx die Suche nach der genauen Phrase. Im Suchbeispiel muss das Dokument die Zeichenfolge "James Bond 007" in dieser exakten Ausformulierung enthalten. Als Reihenfolgekriterium ist hier die Position des ersten Vorkommens entscheidend. Kommt der Suchterm beispielsweise am Anfang des Dokuments vor, so wird dieses Dokument auch als erstes zurückgegeben.

Als letzten Modi stellt Sphinx den sogenannte *BOOLEAN*-Modus zur Verfügung. Hierbei kann der Nutzer verschiedene Suchbegriffe mit Boolschen-Operatoren verknüpfen. Zum Beispiel kann die Suche nach "James AND (BOND OR

007)" abgegeben werden. Die Ergebnismenge listet Dokumente, welche sowohl "James" als auch eines der beiden Begriffe "Bond" und "007" enthält. Ist dieser Modus eingestellt, werden die Ergebnisse keinem Ranking unterzogen.

Mit *EXTENDED* gibt es nun auch einen Modi, der die Verwendung verschiedenster Operatoren direkt im Suchterm ermöglicht. Deren Einsatz wird im Folgenden vorgestellt.

Operatoren dienen zur sinnvollen Verknüpfung bzw. Limitierung der Suchergebnisse.

Im *EXTENDED*-Modus unterstützt Sphinx ebenfalls die boolschen Operatoren. Sollen zwei Wörter per *AND* verknüpft werden, werden diese einfach der Reihe nach aufgezählt *James Bond*. Ein senkrechter Strich verknüpft die beiden Terme per *ODER* (*James |Bond*). Ein Ausschluss von Suchtermen ist mit der *NOT*-Annotation möglich. Hierfür wird dem Begriff ein Ausrufezeichen vorgeschoben (z.B. *James !007*). Alle Operatoren können natürlich auch auf Phrasen angewandt werden und beliebig geklammert werden, um Begriffe oder Phrasen zu gruppieren.

Soll beispielsweise ein Wort nur im Filmtitel vorkommen, so ermöglicht der *Field*-Operator eine Eingrenzung auf ein Dokumentenfeld. Per *@Feldname* können so Einschränkungen auf ein oder mehrere Felder vorgenommen werden. *@(Title) James Bond* sucht nach den beiden Termen nur im Titel des Dokuments. Filmbeschreibungen oder sonstige Felder werden ignoriert.

Mit Hilfe des *Proximity*- und *Near*-Operators können Abstände zwischen Phrasen und Schlüsselwörtern definiert werden. Mit dem Suchterm *"James Bond 007" 3* dürfen zwischen den drei Wörtern insgesamt nur drei zusätzliche Keywords vorkommen (Stopwords werden ignoriert). *James NEAR/3 Bond NEAR/3 007* definiert die selbe Suche per *Near*-Operator. Hierbei ist auch der Unterschied zwischen beiden Operatoren sichtbar. Der *Near*-Operator bezieht sich jeweils nur auf zwei Wörter und deren Abstand. Beim *Proximity*-Operator werden die Gesamtabstände aller Wörter in die Berechnung einbezogen.

Der *Quorum*-Suchbefehl ermöglicht es die Suche zu limitieren. *"James Bond 007"/2* würde Ergebnisse liefern, die mind. zwei der drei angegebenen Schlüsselwörter enthalten. Somit können untere Limits der Übereinstimmungen angegeben werden, um vermeintlich schlechte Ergebnisse auszusortieren.

Reihenfolgebeziehungen realisiert der Sphinx-Nutzer mit dem *Strict Order*-Befehl. *James <<Bond* definiert, dass James im Dokument *vor* Bond auftauchen muss. Um weitere Einschränkungen vorzunehmen kann der *Sentence*- oder *Paragraph*-Operator eingesetzt werden, um nur Ergebnisse zu erhalten, bei denen die Suchbegriffe im selben Satz bzw. im selben Paragraphen vorkommen.

Abschließend gibt es für die Unterstützung von XML bzw. HTML den *Zone*-Befehl. Mit diesem können Dokumente gefunden werden, bei denen ein Term in einer bestimmten Zone vorkommt. Eine solche wird über ein Anfangs- und ein Endtag, beispielsweise * ... *, definiert.

Resultset-Processing verarbeitet die von Sphinx berechnete Ergebnismenge. Jenes *Resultset* besteht aus den gefundenen Dokumenten und Metadaten. Die Liste der Dokumente enthält sowohl die Dokumenten-IDs sowie das Gewicht des Treffers als auch verschiedene zusätzliche Eigenschaften, wie definierte Attribute oder zusätzlich berechnete Werte. In den Metadaten finden sich beispielsweise die Anzahl der gefundenen Treffer, Statistiken über die Keyword-Verteilung im Resultset oder auch Angaben zu aufgetretenen Fehlern und Warnungen.

Das *Resultset-Processing* verarbeitet Sphinx-intern die durch den *SELECT*-Operator gefundenen Ergebnisse. Dies ermöglicht eine bereits durch Sphinx optimierte Ergebnisstruktur, ohne dass unnötige Treffer an den Suchclienten zurückgegeben werden. Ähnlich wie bei SQL stehen hier *Expressions*, *Filter*, *Grouping*, *Sorting* und verschiedene andere Funktionen zur Verfügung. Mit *Expressions* können weitere Ergebnisse errechnet werden, wie zum Beispiel die Umrechnung einer Währung. Hierzu stehen sowohl arithmetische, vergleichende als auch boolsche Operatoren zur Auswahl. Zusätzlich können auch mathematische Funktionen wie Sinus, Cosinus oder *Geodist* (zur Berechnung von Distanzen) eingesetzt werden.

Filter dienen der Einschränkung des Resultsets. Wie auch bei SQL wird dies bei SphinxQL mit der *WHERE*-Klausel realisiert. So können beispielsweise nur Filme zwischen den Jahren 2000 und 2012 abgerufen werden. Da die Filterung die Menge der Ergebnisse einschränkt, ist sie somit auch für die Performance der Suche verantwortlich. Sphinx optimiert intern die Suchqueries und kann so bereits frühzeitig nicht relevante Ergebnisse streichen.

Mithilfe des Groupings und Sortings können die Suchergebnisse weiter strukturiert und in die richtige Reihenfolge gebracht werden. Ebenfalls als Anlehnung an den SQL-Syntax wird dies mit *GROUP BY* bzw. *ORDER BY* durchgeführt [3].

2.11 Java-API

Neben zahlreichen APIs für verschiedene Programmiersprachen, stellt Sphinx ebenso eine solche für Java zur Verfügung. Vor der erstmaligen Verwendung muss diese noch mithilfe des Java-Compilers *javac* zu einer *Jar-Datei* gebaut werden. Der folgende Aufruf übernimmt dies für den Entwickler:

D:/Sphinx/api/java/mk.cmd

Dies setzt jedoch die Installation des aktuellen *Java-SDKs*[16] voraus. Eine dazu gehörige Dokumentation kann ebenfalls mit Hilfe des *mkdoc.cmd*-Batchfiles erzeugt werden. Da generell die Dokumentation der Java-Schnittstelle eine Schwachstelle ist, sollte sich der Entwickler zuvor die ausführlichen Beispiele der PHP-API zu Gemüte führen [5]. Da die Java-API bis auf den Syntax identisch zu dieser ist, sollte die Nutzung daraufhin kein Problem mehr darstellen.

[16] http://www.oracle.com/technetwork/java/javase/downloads/index.html

Nach der Registrierung der Jar-Datei im *Classpath*, können die Klassen und Methoden im eigenen Code verwendet werden. Als Ergebnisdatenstruktur kommt bei der Java-API die Klasse *SphinxResult* zum Einsatz, welche alle notwendigen Daten enthält.

Listing 1.8 zeigt ein Minimalbeispiel zur Suche nach einem James Bond Film.

```
1  SphinxClient c = new SphinxClient("localhost", 9312);
2  c.SetMatchMode(SphinxClient.SPH_MATCH_ALL);
3  c.SetWeights(new int[] { 100, 1 });
4  c.SetLimits(0, 100);
5  c.SetSortMode(SphinxClient.SPH_SORT_RELEVANCE, "");
6  SphinxResult result = c.Query("James Bond", "*");
```

Listing 1.8. Sourcecode - Suchbeispiel

Zunächst wird ein Sphinx-Client erzeugt und verschiedene Einstellungen gesetzt. Die *Query-Methode* sucht nach den Begriffen *James* und *Bond* in allen (*) definierten Indizes. Das erhaltene Resultset kann daraufhin in der Anwendung weiterverarbeitet werden.

3 Bewertung

Sobald es um schnelle Volltextsuche geht, führt im nicht-kommerziellen Opensource-Bereich eigentlich kein Weg an Sphinx vorbei. Denn während die MySQL-Suche bereits bei ca. 100.000 Einträgen spürbar langsamer wird, stößt Sphinx selbst bei mehreren Millionen Datensätzen nicht an seine Leistungsgrenzen. So ist Sphinx laut Angaben des Entwicklers bei der Suche zwischen 10 und 1000 mal schneller als eine MySQL-Suche und ca. 2-4 mal schneller als die Konkurrenzengine *Lucene* [4]. Auch eigene Tests bei deaktiviertem Caching haben dies gezeigt. Beim hier angesprochenen Beispiel dauerte die durchschnittliche Suche mit MySQL ca. 0.2 Sekunden. Sphinx hingegen erledigte die gestellten Aufgaben in unter 0.0001 Sekunden. Zusätzliche Performanceverbesserungen können durch gezielte Clusterbildung erzielt werden. Ein weiterer Vorteil ist die breite Unterstützung verschiedenster Programmiersprachen, sowie die einfache Installation und Wartung. Sollte beispielsweise aktuell im Projekt nur MySQL eingesetzt werden, ist die Integration von Sphinx sehr einfach, denn die bestehenden SQL-Query können beibehalten werden, da SphinxQL ebenfalls mit diesem Query-Syntax arbeitet.

A Anhang - SQL-Setup

```
1  --
2  -- Datenbank: `seminar2012`
3  -- Tabellenstruktur für Tabelle `movies`
4  --
5  CREATE TABLE IF NOT EXISTS `movies` (
6    `id` int(11) NOT NULL AUTO_INCREMENT,
7    `title` text NOT NULL,
8    `summary` text NOT NULL,
9    `year` text NOT NULL,
10   `ts` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON
        UPDATE CURRENT_TIMESTAMP,
11   PRIMARY KEY (`id`)
12 ) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT
     =2502908 ;
13
14 --
15 -- Tabellenstruktur für Tabelle `movies_helper`
16 --
17 CREATE TABLE IF NOT EXISTS `movies_helper` (
18   `tablename` text NOT NULL,
19   `maxts` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON
         UPDATE CURRENT_TIMESTAMP
20 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
21
22 --
23 -- Tabellenstruktur für Tabelle `movies_klist`
24 --
25 CREATE TABLE IF NOT EXISTS `movies_klist` (
26   `id` int(11) NOT NULL,
27   `title` text NOT NULL,
28   `ts` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON
        UPDATE CURRENT_TIMESTAMP
29 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Listing 1.9. SQL-Setup

B Anhang - Config-File

```
# Bereich 1
# base datasource
source localDatabase
{
  type           = mysql
  sql_host       = localhost
  sql_user       = root
  sql_pass       =
  sql_db         = seminar2012
  sql_port       = 3306
}

# movieDataSource
source movieSource : localDatabase
{
  sql_query_pre    = SET @maxts:= (SELECT NOW())
  sql_query      = SELECT * FROM movies WHERE ts<@maxts
  sql_query_post     = REPLACE INTO movies_helper VALUES
      ("movies_tmp", @maxts)
  sql_query_post_index = DELETE FROM movies_helper WHERE
      tablename="movies"
  sql_query_post_index = UPDATE movies_helper SET
      tablename="movies" WHERE tablename="movies_tmp"
  sql_query_post_index = DELETE FROM movies_klist WHERE
      ts<(SELECT maxts FROM movies_helper WHERE tablename=
      "movies")
  # Document attribute
  sql_attr_string   = title
}

source movieSourceDelta : localDatabase
{
  sql_query_pre   = SET @maxts:= (SELECT maxts FROM
      movies_helper WHERE tablename="movies")
  sql_query     = SELECT * FROM movies WHERE ts>=@maxts
  sql_query_killlist  = SELECT id FROM movies WHERE ts>=
      @maxts UNION SELECT id FROM movies_klist
}

# Bereich 2
# Disk-based Index
index movieIndexDisc
{
  source         = movieSource
  path           = D:/Sphinx/data/movies
  docinfo        = extern
  charset_type   = sbcs
```

```
41    stopwords      = D:/Sphinx/stopwords.txt
42    min_word_len   = 3
43  }
44
45  # Real-Time Index
46  #index movieIndexRT
47  #{
48  # type           = rt
49  # rt_mem_limit   = 32M
50  # path           = D:/Sphinx/data/moviesRT
51  # charset_type   = utf-8
52  # rt_field       = title
53  # rt_field       = content
54  # rt_attr_uint   = gid
55  #}
56
57  # Distributed Index
58  #index movieIndexDist
59  #{
60  # type  = distributed
61  # local = movieIndexRT
62  # local = movieIndexDisc
63  # agent = 192.168.2.20:9312:serverIndex
64  #}
65
66  # Bereich 3
67  indexer
68  {
69    mem_limit    = 32M
70  }
71
72  # Bereich 4
73  searchd
74  {
75    listen           = 9312
76    listen           = 9306:mysql41
77    log              = D:/Sphinx/log/searchd.log
78    query_log        = D:/Sphinx/log/query.log
79    read_timeout     = 5
80    max_children     = 30
81    pid_file         = D:/Java/Sphinx/log/searchd.pid
82    max_matches      = 1000
83    seamless_rotate  = 1
84    preopen_indexes  = 1
85    unlink_old       = 1
86    workers          = threads # for RT to work
87    binlog_path      = D:/Sphinx/data
88  }
```

Listing 1.10. Config-File

References

1. Sphinx Tech. Inc.: Sphinx Open Search Server,
   unter *http://sphinxsearch.com*,
   zuletzt geprüft am 10.09.2012
2. Sphinx Tech. Inc.: OpenSQL Camp 2009, "Sphinx 2009",
   unter *http://sphinxsearch.com/files/talks/sphinx2009-opensqlcamp2009.pdf*,
   zuletzt geprüft am 10.09.2012
3. Aksyonoff, A.: Introduction to Search with Sphinx: From Installation to Relevance Tuning. O'Reilly Media Inc., Sebastopol, USA
4. Sphinx Tech. Inc.: MySQL UC 2010, "Sphinx: full-text search in 2010",
   unter *http://sphinxsearch.com/files/talks/sphinx-mysqluc2010.pdf*,
   zuletzt geprüft am 10.09.2012
5. Aksyonoff, A.: Introduction to Search with Sphinx: From Installation to Relevance Tuning. O'Reilly Media Inc., Birmingham, U.K

Apache Solr

Johannes Döring

Otto-Friedrich-Universität Bamberg, Lehrstuhl für Medieninformatik

Zusammenfassung Diese Arbeit soll einen Überblick über Apache Solr geben. Apache Solr ist eine Volltextsuchmaschine, die viele Funktionen bietet Informationen aufzubereiten und dem Nutzer zu präsentieren. Die notwendige Konfiguration und unterstützte Funktionalität werden ebenfalls behandelt. Abschließend wird gezeigt, welche Möglichkeiten es gibt, Apache Solr in eine Webseite einzubinden.

1 Einführung

Das Internet bietet eine immer größer werdende Menge an Informationen an. Damit steigen auch die Anforderungen an Suchmaschinen, diese Informationen so aufzubereiten, um dem Nutzer die gewünschten Informationen bereit zu stellen. Die *Apache Software Foundation* bietet als eine Lösung das Projekt *Solr* an. Apache Solr ist eine Volltextsuchmaschine, d. h. es werden Texte (nicht nur einzelne Worte) als Suchbegriff angenommen und der Index danach abgesucht.[1]
Um Apache Solr besser verstehen zu können wird im nächsten Abschnitt Apache Lucene vorgestellt, der zugrunde liegende Kern von Apache Solr.

1.1 Apache Lucene

Apache Lucene ist eine Programmbibliothek zur Volltextsuche.[2] Der Fakt, dass Lucene nur eine Bibliothek ist, bedeutet auch, dass es kein Server oder Webcrawler[1] ist. So gibt es beispielsweise keine Konfigurationsdateien. Um Lucene zu verwenden ist es notwendig eigenen Code zu schreiben, der die Lucene API verwendet.
Nichtsdestotrotz bringt Lucene bereits einige Funktionen mit sich. Um Dokumente auffindbar zu machen, müssen diese indexiert werden. Lucene verwendet hierfür einen invertierten Index. Dies bedeutet, dass im Cache nur der Verweis auf den Ort des Dokuments oder der Datenbank gespeichert wird.[3] Der Vorteil liegt darin, dass nicht alle Dokumente oder die Datenbank durchsucht werden muss, sondern nur der Index. Während der Indexierung bietet Lucene noch weitere Möglichkeiten den Text zu analysieren. Hierfür gibt es sogenannte *tokenizer*, welche Texte in einzelne Wörter oder Zahlen zerlegt, und *stemmers*, die ein Wort auf den Wortstamm zurückführt. Die Textanalyse dient dazu bei einer Suchanfrage mehr Treffer liefern zu können.

[1] Ein Computerprogramm, das automatisch das World Wide Web durchsucht und Webseiten analysiert

Weitere Funktionen von Lucene sind eine Abfragesprache, um bessere Ergebnisse zu erhalten, sowie ein Scoring-Algorithmus zur Bewertung von Suchergebnissen. Die Bewertung dient der Anordnung der Suchergebnisse, um die Relevanten als Erstes anzuzeigen.

1.2 Apache Solr

Apache Solr hingegen ist eine Webapplikation für Volltextsuchen, basierend auf Lucene. Es baut auf Lucene auf und erweitert dieses mit Konfigurationsdateien und zusätzlicher Funktionalität. Dies erleichtert beispielsweise das Definieren der Textanalyse, welche zur Indexierung durchgeführt werden soll. Lucene ist somit die zugrunde liegende Bibliothek und Solr eine Webapplikation.
Die Konfigurationsmöglichkeiten und andere Funktionen werden in den weiteren Abschnitten noch genauer erläutern und werden an dieser Stelle deshalb nur kurz aufgezählt:

1. Ein Server, der über HTTP via XML und JSON kommuniziert
2. Eine webbasierte Administration-Oberfläche
3. Facettierung von Suchergebnissen
4. Eine Analysetool zum Testen und Debuggen der Textanalyse
5. Solritas: Ein Anwendungsbeispiel von Apache Solr

1.3 Start von Apache Solr

Um die Funktionen von Solr benutzen zu können sind ein paar Schritte zur Vorbereitung notwendig. Zu erst muss die standalone Version von Solr heruntergeladen werden. Diese steht zum kostenlosen Download auf der Solr Webseite zur Verfügung (http://lucene.apache.org/solr/). Die aktuelle Version beim Verfassen dieser Arbeit ist Version 3.6.0. Liegt das Programm auf der Festplatte, kann es mittels des mitgelieferten Tools „start.jar“ über die Kommandozeile ausgeführt werden. Der Befehl lautet:

```
java -jar start.jar
```

Dieser startet automatisch einen integrierten Jetty Server mit dem Standardport 8983. Da der Jetty Server nur zu Vorführzwecken gedacht ist und nur selten im Produktiveinsatz ist, ist eine Installation auf einen anderen Servlet Container sinnvoll. Ein Beispiel ist „Tomcat“, ebenfalls ein Apache Projekt. Wie man Solr auf Tomcat installiert ist ausführlich auf der Solr Webseite beschrieben.
Damit ist die Installation von Solr abgeschlossen und die Administrationsseite kann aufgerufen werden über: [2]

```
http://localhost:8983/solr/admin/
```

[2] Wenn Solr lokal installiert wurde

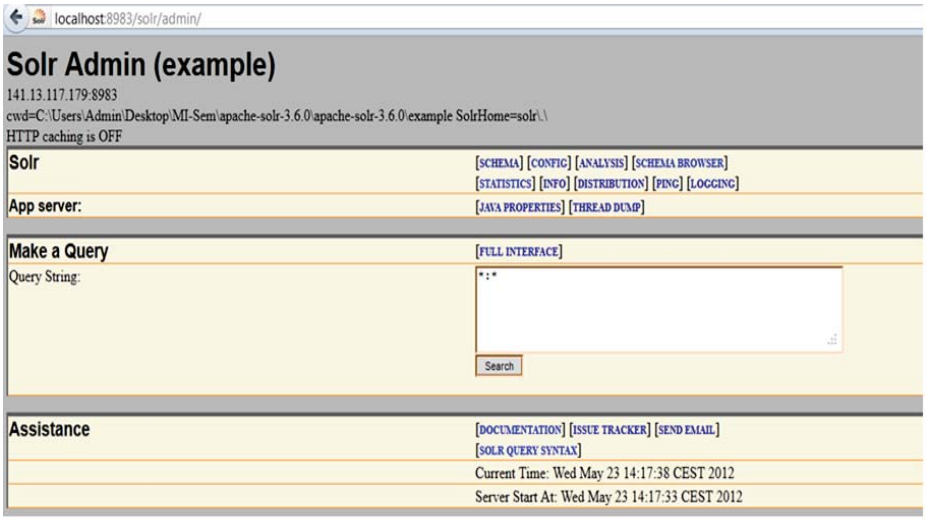


Abbildung 1. Administrationsseite

Die Administrationsseite ist in Abbildung 1 dargestellt. Über diese Seite können alle wichtigen Informationen über Solr aufgerufen werden. Es können beispielsweise die *Schema.xml* und die *Konfigurationsdatei* über den Browser angezeigt werden. Des weiteren können hier Suchanfragen eingeben und der Index durchsucht werden. An dieser Stelle würde die Suchanfrage noch keine Ergebnisse liefern, da noch keine Daten im Index zur Verfügung stehen. Um Dokumente auffindbar zu machen, müsse diese zuerst indiziert werden.

1.4 Indizierung

Für die Indizierung von Dokumenten kann die mitgelieferten *post.jar* verwendet werden. Durch Ausführen dieser Datei können Dokumente einfach zum Index hinzugefügt werden. Möchte man beispielsweise alle XML-Dateien aus einem Verzeichnung zum Index hinzufügen wird lediglich folgender Aufruf benötigt:

```
java -jar post.jar *.xml
```

Der Ausdruck setzt nur voraus, dass die *post.jar* im selben Verzeichnis liegt wie die XML-Dateien. Abbildung 2 zeigt den Aufruf der *post.jar* in der Kommandozeile.
Durch Aufruf der *post.jar* können Dokumente zum Indizieren noch genauer eingeschränkt werden und beispielsweise ganze Dateinamen eingegeben werden, um nur ein bestimmtes Dokument zu indizieren. Zudem können alle Befehle auch in XML-Format eingegeben werden. Ein Beispiel für das Hinzufügen eines Dokuments sieht wie folgt aus:

```
<add><field name=‚manu‘>name:Radeon</add>
```

```
SimplePostTool: POSTing files to http://localhost:8983/solr/update..
SimplePostTool: POSTing file gb18030-example.xml
SimplePostTool: POSTing file hd.xml
SimplePostTool: POSTing file ipod_other.xml
SimplePostTool: POSTing file ipod_video.xml
SimplePostTool: POSTing file mem.xml
SimplePostTool: POSTing file money.xml
SimplePostTool: POSTing file monitor.xml
SimplePostTool: POSTing file monitor2.xml
SimplePostTool: POSTing file mp500.xml
SimplePostTool: POSTing file sd500.xml
SimplePostTool: POSTing file solr.xml
SimplePostTool: POSTing file utf8-example.xml
SimplePostTool: POSTing file vidcard.xml
SimplePostTool: COMMITting Solr index changes..
```

Abbildung 2. Indizierung über post.jar

Dieser Befehl sagt aus, dass alle Dokumente des Herstellers (engl. manufacturer) „Radeon“ hinzugefügt werden sollen. Entsprechend können dieses Dokumente auch wieder aus dem Index gelöscht werden mit:

```
<delete><field name=‚manu‘>name:Radeon</delete>
```

Verwendet man die *post.jar* werden die Dokumente nach dem Hinzufügen auch automatisch indiziert. Bei Aufruf über XML-Format ist noch ein „commit“ notwendig um Dokumente zu indizieren. Hintergrund ist, dass der Indizierung Prozess bei Verwendung eines invertierten Indexes sehr zeit- und leistungsintensiv sein kann bei entsprechend hoher Anzahl an Dokumenten. Deshalb können Dokumente zur Laufzeit bereits hinzugefügt, aber zu einem späteren Zeitpunkt indiziert werden, z. B. in der Nacht. Sind nun Dokumente im Index, können diese anschließend durch Suchanfragen gefunden und angezeigt werden.

1.5 Unterstütze Datenformate

Apache Solr unterstützt eine Vielzahl von Datenformaten, welche indiziert werden können. Dieser Abschnitt soll deshalb einen groben Überblick über die wichtigsten Datenformate geben.
Ein wichtiges Datenformat ist das XML-Format. Der Vorteil von XML-Dateien zur Kommunikation ist, dass diese plattform- und implementationsunabhängig sind. Ein Beispiel für den Aufbau einer XML-Datei zeigt Abbildung 3. Die Datei enthält verschiedene Felder (engl. fields) mit Bezeichnungen. Diese Felder dienen der genaueren Beschreibung der Datei bzw. Dokuments. Im Beispiel enthält die Datei eine ID, einen Namen, eine Kategorie usw. Der Zweck der Einteilung ist die Möglichkeit eine Vielzahl von Dokumenten anhand der Merkmale, wie die Kategorie, ordnen zu können. Nutzen kann diese Funktion der Ersteller einer Webseite für beispielsweise einen Online-Shop und kann alle Produkte mit der Kategorie „graphics card“ filtern lassen. Der Nutzer, welcher eine Suchanfrage

startet, kennt normalerweise den Aufbau einer solchen Datei nicht und kann deshalb nicht spezifisch nach der Kategorie suchen, sondern nur nach den Begriff wie „graphics card“. Wie dies umgesetzt werden kann, wird in Kapitel 3 näher erläutert.

Das nächste wichtige Dateiformat ist *JSON*. JSON steht für „JavaScript Object

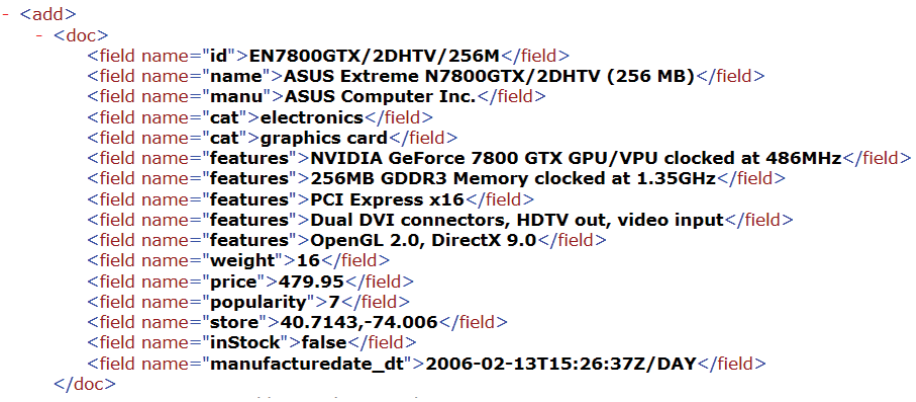


Abbildung 3. Beispieldatei in XML-Format

Notation“ und hat einen ähnlich strukturierten Aufbau wie das XML-Format. Ein Buch in JSON-Format könnte folgendermaßen aussehen:

```
{
    "id" : "978-0641723445",
    "cat" : ["book","hardcover"],
    "name" : "The Lightning Thief",
    "author" : "Rick Riordan",
    "series_t" : "Percy Jackson and the Olympians",
    "sequence_i" : 1,
    "genre_s" : "fantasy",
    "inStock" : true,
    "price" : 12.50,
    "pages_i" : 384
  }
```

Auch hier wird das Dokument mit Merkmalen versehen wie eine eindeutige ID, einen Autor etc.

Für Solr spielen diese beiden Dateiformate, XML und JSON, eine besondere Rolle, da Solr ausschließlich über diese beiden Formate kommuniziert. Suchergebnisse werden standardmäßig als XML-Format zurückgeliefert und müssen anschließend aufbereitet werden, um den Nutzer präsentiert werden zu können. Wie dies umgesetzt werden kann ist in Kapitel 3 genauer beschrieben.

Da normalerweise nicht alle Dokumente in XML- oder JSON-Format vorliegen

unterstützt Solr auch die Verarbeitung anderer Dateiformate. Es können beispielsweise Office Dokumente wie *Microsoft Word* indiziert werden. Ein weiteres sehr weit verbreitetes und häufig verwendetes Datenformat ist *PDF* (Portable Document Format). Hier bedient sich Solr eines weiteren Projekts von Apache namens *Tika*, welches aus PDF-Dokumenten Metadaten extrahiert.[4]So kann bei PDF-Dokumenten während der Indizierung Informationen über beispielsweise den Autor, Erscheinungsjahr, Titel usw. herausgefiltert werden. Dadurch können diese Dokumente auch für spezielle Suchen, wie die Suche nur nach Autoren, verwendet werden.
Zur Verwaltung größerer Mengen an Daten werden in der Regel keine einzelnen Dokumente indiziert sondern Datenbanken verwendet. Zur Unterstützung dafür bietet Solr den *DataImportHandler*.

DataImportHandler Der mitgelieferte DataImportHandler von Apache Solr vereinfacht das Importieren von Daten.Dabei werden folgende Datenquellen unterstützt:

- Import von Daten aus einer Datenbank über JDBC (Java Database Connectivity)
- Import von Daten von einer URL
- Import von e-Mails von einem IMAP [3] Server einschließlich Anhänge

1.6 Solr konfigurieren

Welche Datenformate Solr indizieren kann, wurde im vorherigen Abschnitt gezeigt. Welche Felder solche Dokumente enthalten wird in der *Schema.xml* definiert. Dort wird festgelegt wie diese Felder behandelt werden, wenn Dokumente zum Index hinzugefügt bzw. Suchanfragen auf diese gestartet werden. Die Konfigurationsdatei *Schema.xml* enthält zwei wichtige Sektionen: Datentypen und Felder.

Datentypen Die *<types>* -Sektion erlaubt das Definieren einer Liste von *fieldType* -Deklarationen, welche im Schema benutzt werden soll. Es legt zudem die zugrunde liegende Solr-Klasse für den Datentyp fest. Solr beinhaltet hierfür alle gängigen Datentypen wie *bool* , *int* , *date* , Textfelder usw. Die Definition eines Datentyps sieht in der *Schema.xml* folgendermaßen aus:

```
<fieldType name="text_general" class="solr.TextField">
```

Der Datentyp hat demnach die Bezeichnung „text_general“ und nutzt die Klasse „solr.TextField“.
Dem Datentyp können nun spezielle *analyzer* zugeordnet werden. Diese sind spezielle Operationen, welche auf Felder mit diesen Datentyp angewandt werden.

[3] Internet Message Access Protocol. Textbasiertes Protokoll zum Zugriff auf E-Mails, die sich auf einem Mailserver befinden.

Analyzer können sogenannte *tokenizer*, die den Inhalt in *tokens* (Worte) zerteilen, und Filter sein. Dabei kann noch definiert werden, wann die *analyzer* ausgeführt werden sollen: Bei der Indizierung des Dokuments oder bei einer Suchanfrage. Dabei kann jedem *analyzer* nur ein *tokenizer* zugeordnet werden, da ansonsten Konflikte auftreten würden. Bei Textfeldern zerlegt ein *tokenizer* beispielsweise Sätze in die einzelnen Wörter.
Filter hingegen können beliebig viele für einen *analyzer* definiert werden. Gängige Filter bei Textfelder sind zum Beispiel:

- **StopFilterFactory** Hier werden Wörter, welche in einer speziellen Textdatei definiert wurden, entfernt. Diese Wörter sind sprachabhängig und meisten ohne Bedeutung für die Suchanfrage, da sie sehr häufig vorkommen. Beispiele im Deutschen sind Worte wie: und, die, der, das, ein etc.
- **SynonymFilterFactory** Dieser dient bei der Suchanfrage Synonyme der eingegebenen Wörter ebenfalls zu berücksichtigen. Synonyme müssen auch vorher in einer speziellen Textdatei definiert werden. Eine Zeile in einer solchen Textdatei könnte folgendermaßen aussehen: GB,gib,gigabyte,gigabytes. Dies bedeutet, dass alle dieser Worte als gleich angesehen werden.
- **LowerCaseFilterFactory** Wandelt alle großgeschriebenen Wörter in Kleingeschriebene um, damit auch diese gefunden werden können.

Eine beispielhafte Definition eines Datentyps zeigt Beispiel 1.1

Beispiel 1.1. Beispiel einer Datentyp-Definition

```
<fieldType name="text_general" class="solr.TextField"
    positionIncrementGap="100">
  <analyzer type="index">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" ignoreCase="
        true" words="stopwords.txt"
        enablePositionIncrements="true" />
    <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
  <analyzer type="query">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" ignoreCase="
        true" words="stopwords.txt"
        enablePositionIncrements="true" />
    <filter class="solr.SynonymFilterFactory" synonyms="
        synonyms.txt" ignoreCase="true" expand="true"/>
    <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
</fieldType>
```

Um den einzelnen Felder eines Dokumentes nun die Datentypen zuzuordnen, müssen diese ebenfalls in der *schema.xml* definiert werden. Felder müssen hierfür auf die entsprechenden Datentypen referenzieren.

Felder Ein XML-Dokument besteht wie Abbildung 3 zeigt aus verschiedenen Feldern, wie beispielsweise „id, name, manu, ...“ . Um nun den gewünschten Datentyp zu referenzieren, muss dies in der *schema.xml* definiert werden. Ein Beispiel zeigt 1.2:

Beispiel 1.2. Definition eines Feldes

```
<field name="name" type="text_general" indexed="true" stored="
    true" required="true"/>
```

Dem Feld mit der Bezeichnung „name“ wurde mit diesem Eintrag der Datentyp „text_general“ zugeordnet. Wird ein Dokument indiziert, welches einen Feld mit dieser Bezeichnung beinhaltet, so werden alle im Datentyp definierten *analyzer* ausgeführt.
Es können außerdem noch weitere Optionen festegelegt werden, welche für das entsprechende Feld gelten. Die zusätzlichen Optionen in dieser Zeile haben folgende Funktionen:

- **indexed** „true“, wenn dieses Feld indiziert werden soll. Nur indizierte Felder sind durchsuchbar und sortierbar.
- **stored** „true“, falls der Wert des Feldes wieder-auffindbar sein soll während der Suche.
- **required** legt fest, ob ein Dokument dieses Feld beinhalten muss, ansonsten wird dieses nicht indiziert.

Zur Überprüfung, ob die definierten Datentypen und Felder korrekt funktionieren, bietet Solr auf der Administrationsseite das Analyse-Werkzeuge. Dieses Werkzeuge kann auch benutzt werden, um die Funktionsweise der *analyzer* zu demonstrieren. Eine beispielhafte Anwendung des Analyse-Werkzeugs zeigt Abbildung 4 .
Im Folgenden wird nun das Vorgehen zur Nutzung des Analyse-Werkzeugs beschrieben:
Als erstes muss der Datentyp ausgewählt werden. In diesem Beispiel ist dieser der vorher beschriebene Datentyp text_general. Anschließend wird eingegeben, welche Daten indiziert werden sollen, sprich der Wert für den bestimmten Datentyp. Hier dient als einfaches Beispiel der Satz „Kann GB sein“ . Anschließend wird ein Wert für eine beispielhafte Suchanfrage eingeben. Im vorliegendem Beispiel wurde der Satz „Kann aber auch gigabyte sein“ eingegeben. Das Ergebnis zeigt die Funktionsweise der einzelnen *analyzer* für die Indizierung und Suchanfrage. Diese werden Zeilenweise dargestellt. Als erstes soll das Resultat für den „Index Analyzer“ näher betrachtet werden. Die zugehören *analyzer* sind in Beispiel 1.1 beschrieben.
Als Erstes wird der *tokenizer* auf den Eingabetext angewandt. Daraus ergibt sich die erste Zeile, in der der Satz in seine einzelnen Wörter zerlegt wurde (dargestellt durch die senkrechten Trennstriche). Als nächstes wird der Text durch Filter weiter analysiert. Der erste Filter ist die *StopFilterFactory* . Diese "löscht" alle vorher in der entsprechenden Textdatei definierten Stopwörter aus dem Text. In diesem Fall wurden dort die zwei Wörter „kann“ und „sein“ definiert. Diese werden dementsprechend herausgefiltert und es bleibt „GB“ übrig.

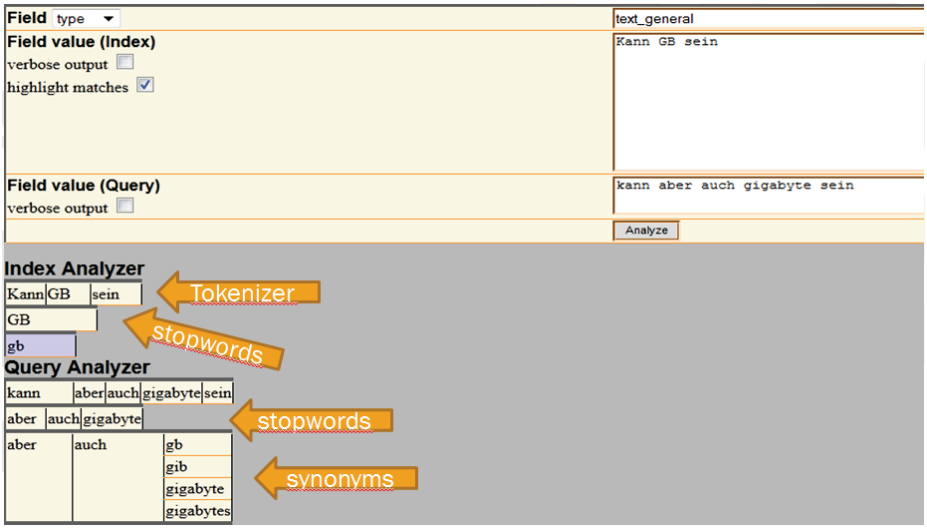


Abbildung 4. Beispielhafte Anwendung des Analyse-Werkzeugs

Als Letztes kommt die *LowerCaseFilterFactory* zum Einsatz. Diese wandelt das großgeschriebene „GB“ in „gb“ um, damit das Großgeschriebene auch bei einer Suchanfrage mit der kleingeschriebenen Variante berücksichtigt wird.
Das selbe Prinzip wird nun bei der Analyse des Suchabfragetextes angewandt. Der einzige Unterschied ist, dass bei der Suche der Filter *SynonymFilterFactory* ausgeführt wird. Hierfür werden für ein Wort Synonyme in einer Textdatei hinterlegt und ebenfalls berücksichtigt. In diesem konkreten Fall bedeutet dies, dass eine Suche nach „gb“ ebenfalls Ergebnisse mit den Begriffen „gib, gigabyte“ oder „gigabytes“ liefert.
Nachdem alle *analyzer* ausgeführt wurden, können beide Ergebnisse verglichen und nach Übereinstimmungen gesucht werden. Liegen Übereinstimmungen vor, würde durch diese Suchanfrage das Dokument gefunden und zurückgeliefert werden. In diesem einfachen Beispiel würde sich das Wort „gb“ decken und wird deshalb farblich hinterlegt.[4]

2 Funktionalität von Apache Solr

In Kapitel 1 wurde gezeigt wie Apache Solr eingerichtet und richtig konfiguriert wird. Als nächster Schritt kann Apache Solr für dessen eigentliche Aufgabe benutzt werden: das Suchen. Auch dafür steht im Interface der Administrationsseite ein Feld zur Verfügung um Suchabfragen zu testen. Abbildung 5 zeigt diese Eingabefeld.Als einfachstes Beispiel wird ein Suchbegriff eingegeben (beispielsweise „Radeon“) und anschließend die Suchabfrage gestartet.
Apache Solr liefert anschließend alle Ergebnisse zurück, die dieses Wort enthal-

[4] Dafür muss der Haken bei „highlight matches“ gesetzt werden

Abbildung 5. Eingabe einer Suchabfrage

ten und stellt diese in XML-Format dar. Das Ergebnis dieser Suchabfrage zeigt Abbildung 6 [5] . In diesem Fall wurde genau ein passendes Dokument gefunden, welches die geforderten Suchkriterien erfüllt.

Nicht alle Suchabfrage sind so simpel wie im vorherigen Beispiel. Deshalb un-


```
− <response>
  + <lst name="responseHeader"></lst>
  − <result name="response" numFound="1" start="0">
    − <doc>
      − <arr name="cat">
          <str>electronics</str>
          <str>graphics card</str>
        </arr>
      − <arr name="features">
          <str>ATI RADEON X1900 GPU/VPU clocked at 650MHz</str>
          <str>512MB GDDR3 SDRAM clocked at 1.55GHz</str>
          <str>PCI Express x16</str>
          <str>dual DVI, HDTV, svideo, composite out</str>
          <str>OpenGL 2.0, DirectX 9.0</str>
        </arr>
        <str name="id">100-435805</str>
        <bool name="inStock">false</bool>
        <str name="manu">ATI Technologies</str>
```


Abbildung 6. Ergebnis der Suchabfrage

terstützt Solr eine Vielzahl von möglichen Befehlen, um noch genauere Suchergebnisse zu liefern und diese entsprechend aufzubereiten. Wie diese Befehle zu verwenden sind, um von Solr verstanden zu werden, ist in der Solr Syntax beschrieben. Diese wird anhand von Solritas erläutert. Solritas ist ein von Solr mitgeliefertes Online-Vergleichsportal und dient vor allem der Veranschaulichung

[5] Das entsprechende Dokument muss vorher indiziert worden sein

der Syntax. Die Startseite von Solritas ist in Abbildung 7 dargestellt. Die Syntax wird in den folgenden Abschnitten anhand von Solritas erläutert.

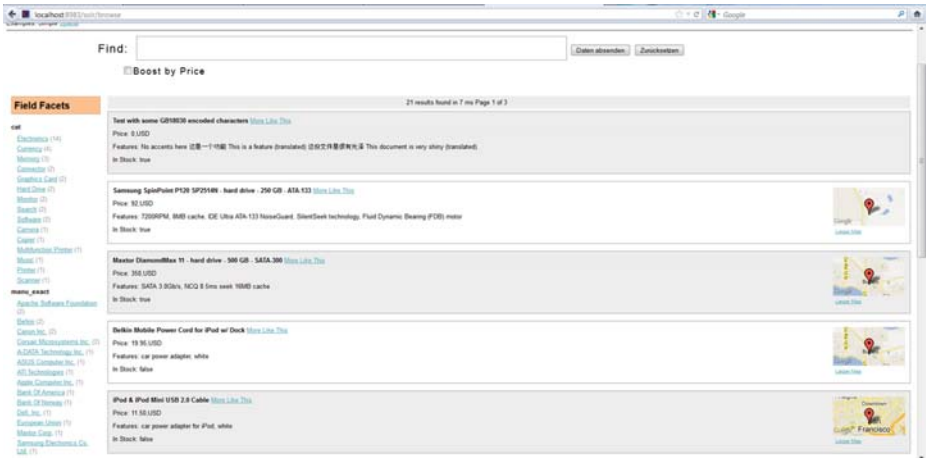


Abbildung 7. Startseite von Solritas

2.1 Sortierung

Werden bei einer Suchabfrage mehrere Ergebnisse gefunden, so kann es bei der Präsentation hilfreich sein, diese zu sortieren. Der Nutzer kann dementsprechend angeben, nach welchen Attribut die Ergebnisse sortiert werden sollen. Es kann zudem unterschieden werden, ob auf- oder absteigend sortiert werden soll. Zur Veranschaulichung ein Beispiel, wie ein solcher Befehl aussieht:

```
q=iPod&sort=price asc
```

Nun zu der Bedeutung dieses Befehls. Der Nutzer der Suchmaschine sucht nach dem Begriff „iPod“ und möchte die Ergebnisse anschließend nach dem Preis (*price*) sortieren. Es soll der geringste Preis zuerst angezeigt werden. Dies wird durch das Hinzufügen des Bergriffs *asc* erreicht. Entsprechend ist eine absteigende Sortierung mit dem Schlüsselwort *desc* möglich.
Für den Aufruf dieser Suchabfrage könnte die komplette URL wie folgt aussehen:

```
http://localhost:8983/solr/browse?q=iPod&sort=price desc
```

Der erste Teil der URL beschriebt den Pfad zur Installation von Solritas. Dieser könnte auch eine Webadresse sein.

2.2 Facettierte Suche

Ein weiteres, sehr wichtiges Feature bei Apache Solr ist die facettierte Suche. Durch diese ist ein einfaches Verfeinern der Suchabfrage möglich. Gerade bei Anwendungen wie Online-Vergleichsportalen oder Shops ist diese Funktion sehr nützlich. Da hier oft sehr viele verschiedene Produkte angeboten werden kann es für den Nutzer sehr hilfreich sein die Ergebnisse immer weiter einzuschränken. Zur Verdeutlichung wird das wird wieder nach dem Schlüsselwort „iPod“ gesucht und anschließend die Suchergebnisse immer weiter eingeschränkt.

```
q=iPod&fq=cat:“electronics“&fq=price:[0.0+TO+50.0]
```

Es werden somit nur alle Ergebnisse mit der Kategorie (*cat*) „electronics“ angezeigt. Außerdem sind nur die Ergebnisse relevant bei denen der Preis (*price*) zwischen 0 und 50 liegt.

2.3 Highlighting

Eine weitere Funktion, vor allem um Suchergebnisse besser präsentieren zu können, ist das Hervorheben bestimmter Ausschnitte. Im Normalfall handelt es sich hierbei um die gesuchten Schlüsselwörter, um sie im Kontext des Dokuments einordnen zu können. Als Beispiel dient folgender Befehl:

```
?q=Radeon&hl=true&hl.q=Radeon
```

Dieser Befehl zeigt drei Anweisung. Der Erste ist die Suche nach dem Schlüsselwort „Radeon“. Danach wird das Highlighting auf *true* gesetzt mit „hl=true“ . Dies dient der Aktivierung dieser Funktion. Zum Schluss kann noch genauer definiert werden, was hervorgehoben werden soll mit „hl.q“ . Ohne diese Anweisung wird automatisch das gesuchte Schlüsselwort (In diesem Fall Radeon) hervorgehoben. Dieses kann jedoch auch überschrieben werden. So kann beispielsweise nach Radeon gesucht, aber das Wort „ATI“ hervorgehoben werden. Als Ergebnis erhält man eine XML-Datei, die zusätzlich einen Highlight-Sektion beinhaltet, wie Beispiel 1.3 zeigt. In dieser werden nochmals alle Felder, welche das hervorgehobene Schlüsselwort enthalten, aufgelistet und mit einem -Tag versehen. Dieser kann z. B. auf einer Webseite verwendet werden, um das Suchwort individuell darzustellen.

Beispiel 1.3. Die Highlight-Sektion

```
<lst name="highlighting">
        <lst name="100-435805">
        <arr name="features">
                <str>ATI <em>RADEON</em> X1900 GPU/VPU
                    clocked at 650MHz</str></arr>
        <arr name="name">
                <str>ATI <em>Radeon</em> X1900 XTX 512 MB PCIE
                     Video Card</str>
        </arr></lst>
</lst>
```

2.4 More Like This

Möchte der Nutzer über ein Thema mit Hilfe einer Suchmaschine recherchieren, so kann die Funktion *More Like This* nützlich sein. Dabei werden neben gefunden Dokumenten bei gegebenen Schlüsselwörtern auch ähnliche Suchergebnisse zurückgeliefert. So können Dokumente des gleichen Autors oder gleicher Kategorie mit angezeigt werden.
Diese Funktion kann auch in Online-Shops sehr nützlich sein. Ein bekanntes Beispiel ist Amazon, welches bei der Detailansicht von Artikel eine Sektion „Verwandte Artikel“ auflistet, um den Kunden potenziell interessante Artikel anzuzeigen.
Die Umsetzung dieser Funktion in Apache Solr erfolgt einerseits über den Befehl *mlt=true*, d. h. die More Like This-Funktion wird zuerst aktiviert. Danach kann über *mlt.fl* definiert werden, welche Felder hierfür relevant sind. Diese können direkt über die URL eingegeben oder in der Konfigurationsdatei definiert werden.
Ein Beispiel aus Solritas ist erneut die Suche nach einer Grafikkarte von Radeon. Wird hier nun zusätzlich der Befehl *mlt=true* gesetzt, werden weitere ähnliche Artikel angezeigt. Abbildung 8 zeigt dieses Beispiel. Die Darstellung wird vom Ersteller der Webseite festgelegt und kann individuell gestaltet werden. So kann nur ein Link auf den ähnlichen Artikel angezeigt oder, wie in dieser Abbildung, noch weitere Informationen dargestellt werden (unter „Similar Items“) .

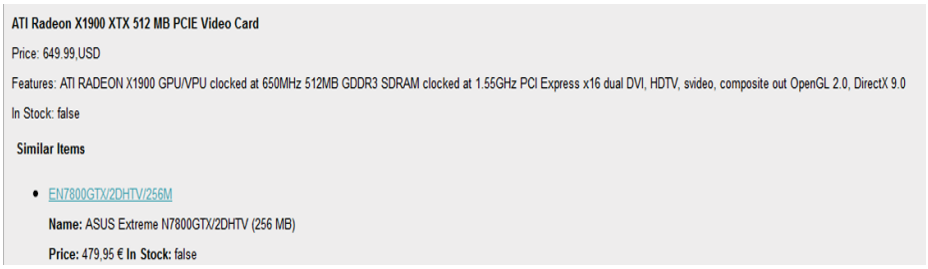


Abbildung 8. Beispiel für More Like This

2.5 Besondere Sucheingaben

Neben den in diesem Kapitel vorgestellten Funktionen für Suchabfragen, kann Solr noch mit besonderen Eingaben von Schlüsselworten umgehen.

Fuzzy Search Darunter wird das Suchen mit einer gewissen Unschärfe verstanden. Wurde ein Schlüsselwort nicht korrekt geschrieben, so können trotzdem Ergebnisse geliefert werden. Explizit kann die Fuzzy Search mit „~“ durchgeführt werden, dieses wird am Ende des Suchwortes angehängt. Als Beispiel soll wieder

nach „Radeon“ gesucht werden. Es wird allerdings das Suchwort „Radoen“ eingeben und deshalb keine Ergebnisse geliefert. Benutzt man allerdings Fuzzy Search und gibt explizit „Radoen~“ ein, werden alle Dokumente mit dem Schlüsselwort „Radeon“ gefunden.

Wildcards Unter Wildcards werden Platzhalter verstanden und durch „*“ dargestellt. Dadurch müssen Suchworte nicht komplett eingegeben werden. Durch die Eingabe von „Rad*“ werden als Ergebnis alle Dokumente geliefert, die Worte beginnend mit „Rad“ enthalten, beispielsweise „Radeon“ .

Kombination von Suchworten Zudem ist es möglich Suchworte zu kombinieren. Apache Solr unterstützt hierfür zwei Operatoren: *AND* und *OR* . Wird *AND* benutzt, so müssen beide Suchworte im Dokument vorhanden sein, während bei *OR* nur eines der Beiden enthalten sein muss.

3 Integration von Apache Solr in Webseite

In zweiten Kapitel wurde gezeigt, welche Funktionalität durch Apache Solr unterstützt wird. Die Nutzer einer Suchmaschine wissen meist nicht, wie diese Funktionen aufgerufen werden können. Die Syntax der notwendigen Befehle ist in der Regel unbekannt. Dafür muss der Ersteller einer Webseite die Funktionalität so bereitstellen, dass diese einfach genutzt werden kann. In diesem Kapitel wird beschrieben, wie eine solche Webseite aufgebaut sein kann, um möglichst einfach und flexibel Apache Solr zu integrieren.

3.1 Einbindung statischer URLs

Eine Möglichkeit ist die Einbindung statischer URLs, d. h. die im vorherigen Kapitel gezeigten Befehle statisch durch eine URL in die Webseite einbinden. Als Beispiel soll die Facetten-Übersicht, wie sie in Abbildung 9 dargestellt ist, in eine Webseite aufgenommen werden. Durch einen Klick auf den ersten Eintrag „Electronics“ werden nur Ergebnisse dieser Kategorie angezeigt. Durch welchen Befehl dies erreicht werden kann, wurde in Abschnitt 2.2 erläutert. Soll dieser nun fest in der Webseite integriert werden, kann dies durch folgenden HTML-Code erreicht werden.

```
<li><a href="/solr/browse?&q=&fq=cat:""electronics""">
    Electronics</a></li>
```

Dadurch wird ein neues Listenelement mit der Bezeichnung „Elecetronics“ erzeugt.
Was jedoch wenn es irgendwann keine solche Kategorie mehr gibt? Sie würde trotzdem angezeigt werden, aber keine Produkte mehr enthalten. Und was wenn eine neue Kategorie hinzukommt? Diese müsste wieder manuell eingetragen werden. Das zeigt, dass diese Variante nicht zu empfehlen ist, da es sehr viel manuelle Arbeit benötigt. Eine bessere Möglichkeit dies umsetzten wird im nächsten Abschnitt erläutert.

3.2 VelocityResponseWriter

VelocityResponseWriter ermöglicht es Solr auf von Velocity Templates erstellten Inhalt zu reagieren.[5] Velocity ist eine einfache, auf Java-basierende Template-Engine, die Daten von Java-Objekten in beispielsweise HTML darstellt.[6]
Mit solchen Templates lassen sich einfach Webseiten aufbauen. Der VelocityResponseWriter greift hierfür auf Parameter im Cache zu und ersetzt diese durch vorher definierte Werte. Diese können in der *solrconfig.xml* eingetragen werden. Soll z. B. die Facetten-Übersicht aus Abbildung 9 erzeugt werden, so werden nur zwei Facettenfelder benötigt: cat und manu_exact. Diese müssen in der Konfigurationsdatei angegeben werden:

```
<str name="facet.field">cat</str>
<str name="facet.field">manu_exact</str>
```

Damit sind die beiden Facettenfelder definiert und der VelocityResponseWriter kann darauf zugreifen. Als nächstes muss das Velocity Template geschrieben werden. Ziel ist es für alle Facetten der Facettenfelder einen Link zu erstellen, um nur diese Facette anzuzeigen. Für jedes einzelne Facettenfeld müssen dafür alle Facetten durchlaufen werden. Dafür werden zwei *foreach*-Schleifen benötigt. Die erste durchläuft alle Facettenfelder, gibt deren Namen aus und versieht diese mit einer *css*[6] -Klasse, um sie entsprechend darzustellen. Im Template sieht dies wie folgt aus:

```
#foreach($field in $response.facetFields)
      <span class="facet-field">$field.name</span>
```

Anschließend wird eine Liste erstellt, die alle Facetten enthält. Jedes Listenelement enthält den Link für den Filter, die Bezeichnung der Facette und die Anzahl, wie oft diese vorkommt. Dazu dient die zweite *foreach*-Schleife, welche nun alle Facetten des Feldes durchläuft und die Informationen beschafft.

```
<ul>
   #foreach($facet in $field.values)
      <li><a href="#url_for_facet_filter($field.name, $facet.
          name)">$facet.name</a> ($facet.count)</li>
   #end
</ul>
```

Beide Teile des Templates zusammen ergeben den HTML-Code, wie er in Abbildung 10 dargestellt ist. So lässt sich mit ein paar Zeilen Code schnell die Facetten-Übersicht erstellen. Diese ist zudem noch sehr flexibel und reagiert sofort auf Veränderungen, wenn beispielsweise eine neue Facette hinzukommt.
Der VelocityResponseWriter wurde verwendet um alle Inhalte von Solritas zu erstellen. Es ist also kein HTML-Coding notwendig um eine solche Seite zu erstellen. Es werden nur die Templates und css-Klassen zur Darstellung verwendet.

[6] Cascading Style Sheets. Formatierungssprache zum Formatieren von strukturierten Dokumenten

Abbildung 9. Solritas Facetten-Übersicht

```
<h2 >Field Facets</h2>
    <span class="facet-field">cat</span>

<ul>
            <li><a href="/solr/browse?&q=&fq=cat:%22electronics%22">electronics</a> (14)</li>
            <li><a href="/solr/browse?&q=&fq=cat:%22currency%22">currency</a> (4)</li>
            <li><a href="/solr/browse?&q=&fq=cat:%22memory%22">memory</a> (3)</li>
            <li><a href="/solr/browse?&q=&fq=cat:%22connector%22">connector</a> (2)</li>
            <li><a href="/solr/browse?&q=&fq=cat:%22graphics+card%22">graphics card</a> (2)</li>
            <li><a href="/solr/browse?&q=&fq=cat:%22hard+drive%22">hard drive</a> (2)</li>
            <li><a href="/solr/browse?&q=&fq=cat:%22monitor%22">monitor</a> (2)</li>
            <li><a href="/solr/browse?&q=&fq=cat:%22search%22">search</a> (2)</li>
            <li><a href="/solr/browse?&q=&fq=cat:%22software%22">software</a> (2)</li>
            <li><a href="/solr/browse?&q=&fq=cat:%22camera%22">camera</a> (1)</li>
            <li><a href="/solr/browse?&q=&fq=cat:%22copier%22">copier</a> (1)</li>
            <li><a href="/solr/browse?&q=&fq=cat:%22multifunction+printer%22">multifunction printer</a> (1)</li>
            <li><a href="/solr/browse?&q=&fq=cat:%22music%22">music</a> (1)</li>
            <li><a href="/solr/browse?&q=&fq=cat:%22printer%22">printer</a> (1)</li>
            <li><a href="/solr/browse?&q=&fq=cat:%22scanner%22">scanner</a> (1)</li>
        </ul>
    <span class="facet-field">manu_exact</span>

<ul>
            <li><a href="/solr/browse?&q=&fq=manu_exact:%22Apache+Software+Foundation%22">Apache Software Foundation</a> (1)</li>
            <li><a href="/solr/browse?&q=&fq=manu_exact:%22Belkin%22">Belkin</a> (2)</li>
            <li><a href="/solr/browse?&q=&fq=manu_exact:%22Canon+Inc.%22">Canon Inc.</a> (2)</li>
            <li><a href="/solr/browse?&q=&fq=manu_exact:%22Corsair+Microsystems+Inc.%22">Corsair Microsystems Inc.</a> (2)</li>
            <li><a href="/solr/browse?&q=&fq=manu_exact:%22A-DATA+Technology+Inc.%22">A-DATA Technology Inc.</a> (1)</li>
            <li><a href="/solr/browse?&q=&fq=manu_exact:%22ASUS+Computer+Inc.%22">ASUS Computer Inc.</a> (1)</li>
            <li><a href="/solr/browse?&q=&fq=manu_exact:%22ATI+Technologies%22">ATI Technologies</a> (1)</li>
            <li><a href="/solr/browse?&q=&fq=manu_exact:%22Apple+Computer+Inc.%22">Apple Computer Inc.</a> (1)</li>
            <li><a href="/solr/browse?&q=&fq=manu_exact:%22Bank+of+America%22">Bank of America</a> (1)</li>
            <li><a href="/solr/browse?&q=&fq=manu_exact:%22Bank+of+Norway%22">Bank of Norway</a> (1)</li>
            <li><a href="/solr/browse?&q=&fq=manu_exact:%22Dell%2C+Inc.%22">Dell, Inc.</a> (1)</li>
            <li><a href="/solr/browse?&q=&fq=manu_exact:%22European+Union%22">European Union</a> (1)</li>
            <li><a href="/solr/browse?&q=&fq=manu_exact:%22Maxtor+Corp.%22">Maxtor Corp.</a> (1)</li>
            <li><a href="/solr/browse?&q=&fq=manu_exact:%22Samsung+Electronics+Co.+Ltd.%22">Samsung Electronics Co. Ltd.</a> (1)</li>
            <li><a href="/solr/browse?&q=&fq=manu_exact:%22U.K.%22">U.K.</a> (1)</li>
            <li><a href="/solr/browse?&q=&fq=manu_exact:%22ViewSonic+Corp.%22">ViewSonic Corp.</a> (1)</li>
        </ul>
```

Abbildung 10. Durch Template generierter HTML-Code

4 Fazit und Ausblick

Apache Solr ist eine sehr mächtige Volltextsuchmaschine, die dazu noch einfach zu konfigurieren ist. Um alle Möglichkeiten voll ausschöpfen zu können ist eine Einarbeitung dennoch unerlässlich. Als große Hilfe dienen allerdings die vielen Beispieldateien, die dem Installationspaket beiliegen. Diese helfen nicht nur dem besseren Verständnis von Solr, sondern können auch verwendet werden, um diese in eine eigene Webseite zu integrieren. Die Konfigurationsdateien beinhalten dafür schon eine Vielzahl von Konfigurationsmöglichkeiten, die bei Bedarf nur auskommentiert werden müssen. Auch die vorgefertigten Templates des VelocityResponseWriter können helfen, eigene Webseiten zu kreieren. Diese müssen hierfür nur angepasst werden.
Ein weiterer großer Vorteil ist, dass Apache Solr eine Open-Source-Software ist und damit kostenfrei heruntergeladen werden kann. Zudem erfreut sich Solr einer großen Community, die die Entwicklung vorantreibt. In dieser Arbeit wurde die Version 3.6.0 von Solr vorgestellt. Die neueste Version ist 4.0 und wird bald veröffentlicht. Das Release enthält neue und überarbeitete Features, die zukünftig Solr-Suchen mächtiger, schneller und komfortabler machen sollen.[7]
Eine Übersicht aller neuen Funktionen findet sich auf der Solr Webseite .[8] Die größte Neuerung ist die *Solr Cloud* . Durch diese Technik soll die Konfiguration und Verwaltung verteilter Systeme vereinfacht werden. Damit lässt sich ein hochverfügbares und fehlertolerantes Cluster von Solr-Servern einrichten.[9]
Die Entwicklung von Apche Solr ist damit noch lange nicht abgeschlossen und lässt auf viele weitere Verbesserungen und Funktionen hoffen.

Literatur

1. http://lucene.apache.org/solr
2. http://lucene.apache.org/
3. Smiley, D., Pugh, E.: Apache Solr 3 Enterprise Search Server -. Packt Publishing Ltd, Birmingham (2011)
4. http://tika.apache.org/
5. http://wiki.apache.org/solr/VelocityResponseWriter
6. http://http://velocity.apache.org/
7. http://www.heise.de/developer/artikel/Die-Neuerungen-von-Apache-Solr-4-0 html
8. http://wiki.apache.org/solr/Solr4.0
9. http://wiki.apache.org/solr/SolrCloud

MapReduce mit Apache Hadoop

Philipp Ott

Otto-Friedrich-Universität Bamberg, Lehrstuhl für Medieninformatik
philipp-bernhard-rudolf.ott@stud.uni-bamberg.de

Zusammenfassung. MapReduce sowie Apache Hadoop werden in diesem Paper kurz vorgestellt und anhand mehrerer Beispiele und Java-Implementierungen erläutert. Dabei wird zunächst MapReduce untersucht und dessen Teilphasen diskutiert. Anschließend findet ein Vergleich mit relationalen Datenbankmanagementsystemen statt und die wesentlichen Unterschiede werden aufgezeigt. Eine Vorstellung von Apache Hadoop als Open-Source-MapReduce-Implementierung leitet den zweiten Hauptteil ein. Im Paper werden dann der konkrete Aufbau und Ablauf einer MapReduce-Aufgabe durch Apache Hadoop beschrieben. Außerdem umreißt es die wesentlichen Eigenschaften von Apache Hadoop und gibt einen knappen Überblick über dessen Subprojekte.

Schlagworte: MapReduce, Apache Hadoop, Einführung, HDFS

1 Motivation

„Ich glaube, dass es auf der Welt einen Bedarf von vielleicht fünf Computern geben wird“, soll angeblich Thomas Watson, der damalige Vorsitzende von IBM, im Jahre 1943 gesagt haben. Um zu erkennen, wie falsch er mit dieser Aussage liegen sollte, mussten keine 70 Jahre bis heute vergehen. Aber erst in den letzten 10 Jahren ist (und daran sind sicherlich mitunter auch das Internet und dessen globale Vernetzungsmöglichkeiten für jedermann verantwortlich) eine wahre Datenflut über uns hereingebrochen. Allein das soziale Netzwerk Facebook muss über 500 Terabytes [1] an neuen Inhalten, hochgeladen durch seine Nutzer, verarbeiten. Was auf der einen Seite ein gewaltiges Problem für die Aufnahmemöglichkeiten und das Fassungsvermögen des menschlichen Geistes darstellt, bringt auch viele neue Herausforderungen für Unternehmen mit solchen riesigen Datenmengen umzugehen. Viele Algorithmen sind schlichtweg nicht für die Verarbeitung von zu vielen Inhalten ausgelegt und benötigen viel zu viel Zeit bis zur Berechnung des Ergebnisses. Es besteht wohl kein Zweifel, dass kaum jemand (um das oben genannte Beispiel aufzugreifen) Facebook nutzen würde, wenn die im Hintergrund ablaufenden Berechnung so lange dauern würden, dass der Nutzer bei jeder Interaktion viele Sekunden lang warten müsste, bis eine Antwort vom Server kommt.

Gehen wir beispielsweise von folgender Aufgabenstellung aus, für die es gilt, einen geeigneten Algorithmus zu entwerfen und blenden dabei zunächst die Berücksichtigung der Größe der Datenmengen aus: Gegeben sei eine Anzahl von

Dokumenten, von denen die Worthäufigkeiten über alle Dokumente hinweg bestimmt werden soll[1]. Ein naheliegender Algorithmus zur Lösung des Problems würde (in Pseudo-Code) wohl in etwa so aussehen:

```
Für jedes Dokument d:
  Für jedes Wort w in d:
    Wenn w in Ergebnismenge m:
      Erhöhe Anzahl von w in m um 1.
    Sonst:
      Lege neue Anzahl für w mit Wert 1 in m fest.
  Ende für.
Ende für.
Gib m als Ergebnis aus.
```

Sicherlich würde der soeben dargestellte Algorithmus seinen Zweck erfüllen und Worthäufigkeiten bestimmen. Leider gibt es aber zwei Probleme, die vor allem deswegen nicht auf dem ersten Blick auffallen, da sie nur unter bestimmten Rahmenbedingungen eintreten:

Zum einen muss die gesamte Ergebnismenge ständig abrufbereit sein und sich damit im Systemspeicher befinden. Handelt es sich – wie im obigen Fall – um Wörter, so sind diese natürlich sprachbedingt begrenzt. Werden allerdings auch Wortvarianten wie „arbeitet“ statt „arbeiten“, Plurale wie „Seminare“ statt „Seminar“ oder einfach nur Rechtschreibfehler mit berücksichtigt, so nimmt der Ergebnisraum exponentiell zu. Sind dann die Anzahl und der Umfang der Dokumente entsprechend groß, kann selbst die Berechnung von Worthäufigkeiten eine enorme Speicherauslastung produzieren. Entsprechend komplexere Aufgaben mit mehr Speicherbedarf sind dann kaum mehr mit einem derartigen Algorithmus zu handhaben.

Zum anderen spielt aber häufig auch die Geschwindigkeit bis zur Berechnung des Ergebnisses – wie bereits erwähnt – eine wichtige Rolle. Verteilte Bearbeitung ist daher unumgänglich, bringt aber automatisch auch einige Probleme mit sich, die eine Anpassung des vorgestellten Algorithmus erfordern. Ein kleineres (und meist glücklicherweise noch recht gut lösbares) Problem bringt die Verteilung der Aufgaben bzw. eine sinnvolle Auslastung aller beteiligten Computer zu schaffen. Würde beispielsweise jeder zur Verfügung stehende Rechner nach der Formel „Anzahl der Dokumente / Anzahl der Rechner“ Dokumente erhalten, so würde die Wortanzahl innerhalb der Dokumente unberücksichtigt bleiben. Ein Rechner, der zufällig viele kurze Dokumente erhält, wäre mit seiner Aufgabe viel schneller fertig und würde die übrige Zeit ungenützt bleiben. Das größere Problem entsteht jedoch aufgrund der Tatsache, dass jeder Computer den kompletten Lösungsweg durchlaufen muss. Wie können die vielen unterschiedlichen Teilhäufigkeiten zu einer gesamten Ergebnismenge zusammengefasst werden? Natürlich kann Computer am Ende seiner

[1] Dabei handelt es sich im Übrigen um eine Aufgabenstellung, die durchaus im Themengebiet des Information Retrieval für Suchmaschinenbetreiber einer Rolle spielt. Natürlich müssten in diesem Fall weiterführende Herausforderungen wie beispielsweise eine Stammformreduktion der einzelnen Wörter berücksichtigt werden. Für das oben genannte Beispiel sind diese Aspekte jedoch nicht relevant.

Berechnung das Ergebnis an eine zentrale Instanz schicken. Damit sind allerdings ab diesem Zeitpunkt die Vorteile der verteilten Verarbeitung passé, da die aufwendige Endberechnung aller erhaltenen Zwischenergebnisse wiederum nur von einem Computer durchgeführt werden kann.

Unabhängig von der Verarbeitung existiert aber mit der vergangenen Entwicklung der Festplatten [2] noch eine weitere Schwierigkeit, die dazu führt, dass verteiltes Arbeiten unerlässlich ist. Die Kapazität der Festplatten in den letzten zehn Jahren etwa von ca. 40 Gigabytes pro Festplatte auf derzeit etwa 1 Terabyte pro Festplatte gestiegen, was einen Faktor 25 bei gleichem Preis bedeutet. Problematisch ist allerdings, dass die Leseraten für Festplatten nicht in dem gleichen Verhältnis gestiegen sind. Vor zehn Jahren lag diese bei ungefähr 25 Megabytes pro Sekunde, was bei 40 Gigabytes ca. 21 Minuten benötigt, um die gesamte Festplatte auszulesen. Inzwischen sind wir bei etwa 125 Megabytes pro Sekunde (ein Zuwachs um den Faktor 5), benötigen aber aufgrund der viel größeren Festplatte ca. 136 Minuten. Leider müssen aber heute auch viel mehr Daten verarbeitet werden, wie eingangs mit den erwähnten 500 Terabytes von Facebook verdeutlicht wurde. Dies hat zur Folge, dass der deutliche Kapazitätszuwachs auch benötigt wird, ein ausgeklügeltes verteiltes System, um nicht an den im Vergleich deutlich zeitaufwendigeren Leseraten zu scheitern, aber ebenso. Selbiges gilt auch für die Transferraten zwischen Computern, die auch deutlich schneller zugenommen haben, als es die Lesegeschwindigkeit einer Festplatte tat.

Das nachfolgende Kapitel über MapReduce gibt einen Eindruck über dessen Funktionsweise und geht anschließend auf ausgewählte Aspekte detaillierter ein. Das anschließende Kapitel zu Apache Hadoop erläutert den Aufbau und Ablauf eines MapReduce Jobs, einige Merkmale der Apache Hadoop Implementierung, beschreibt die weiteren Apache Subprojekte und zeigt konkrete Code-Beispiele zur Implementierung von MapReduce in Java.

2 MapReduce

Um diese und weitere Probleme bei einer parallelen Verarbeitung zu umgehen, veröffentlichten Jeffrey Dean und Sanjay Ghemawat, Mitarbeiter von Google, im Jahr 2004 ein Paper [3] über das sogenannte MapReduce-Verfahren. Es handelt sich dabei um ein Programmiermodell, welches auch riesige Datenmengen verarbeiten kann und bei Google seit 2003 eingesetzt wurde. Die Autoren legten auch viel Wert auf dessen Einfachheit, so dass MapReduce auch von unerfahrenen Anwendern im Bereich paralleler Verarbeitung genutzt werden kann. Der Suchmaschinenkonzern hat im September 2009 ca. 544 Petabytes [13, S. 41] Daten mit MapReduce verarbeitet. Derzeit wird MapReduce von einer Vielzahl bekannter Unternehmen wie Yahoo, Microsoft oder Facebook verwendet. Google hat sich dagegen 2010 [4] dazu entschlossen, auf MapReduce zu verzichten und ein neues System (Google Caffeine) einzusetzen, dass noch genauer auf die Aufgaben im Bereich Information Retrieval abgestimmt ist.

2.1 Die Funktionsweise von MapReduce an einem Beispiel

Die Funktionsweise von MapReduce wird zunächst an einem Beispiel erläutert. Dazu nehmen wir die folgende exemplarische Aufgabenstellung an: Wir haben die Messergebnisse von unterschiedlichen Wetterstationen erhalten und wollen eine Auflistung der höchsten gemessenen Temperatur pro Jahr. Dabei liegen die gegebenen Messergebnisse als einfache Textdaten vor, wobei jede Zeile einen Eintrag mit Informationen über die jeweilige Messstation, den Messzeitpunkt und den Messwert enthält.

Als Vorbereitung müssen die vorliegenden Daten zunächst ins System eingelesen werden. Nehmen wir dazu die folgenden Werte an:

```
Messstation A, 12.10.1999, 20.3 Grad Celsius
Messstation A, 15.12.2000, 15.9 Grad Celsius
Messstation C, 22.11.1999, 21.5 Grad Celsius
Messstation A, 10.12.1999, 12.4 Grad Celsius
Messstation B, 22.12.2000, 16.0 Grad Celsius
Messstation B, 30.10.2000, 15.9 Grad Celsius
```

Anschließend kann in der Map-Phase eine Zuordnung zwischen Zeitpunkt und Messwert erfolgen. Dazu wird für jeden Eintrag ein Paar aus Jahreszahl und Temperaturangabe bestimmt:

```
1999, 20.3
2000, 15.9
1999, 21.5
1999, 12.4
2000, 16.0
2000, 15.9
```

Der folgende Shuffle-Schritt ist das Herzstück von MapReduce und muss nicht vom Nutzer definiert werden. Es findet dabei eine Gruppierung auf Basis der Jahreszahlen statt. Im konkreten Fall könnte die Gruppierungsregel in Pseudo-Code in etwa so lauten:

```
Für jede Jahreszahl j:
  Sammle alle zugehörigen Temperaturangaben t.
  Gib alle t für j als Ergebnis aus.
Ende für.
```

Als Ergebnis liegen demnach eine Liste aller vorkommenden Jahreszahlen und deren zugehörigen Temperaturangaben vor. Diese könnte beispielsweise folgendermaßen aussehen:

```
1999, [20.3, 21.5, 12.4, usw.]
2000, [15.9, 16.0, 15.9, usw.]
usw.
```

Dieses Zwischenergebnis stellt wiederum die Ausgangsbasis für den kommenden Reduce-Schritt dar. Dieser kann nun die anfänglich gewünschte Operation – in unserem Beispiel die Ermittlung des Maximalwertes pro Jahr – ausführen und somit das gewünschte Ergebnis bestimmen. Im Beispiel würde dies lauten:

```
1999, 21.5
2000, 16.0
```

Die nachfolgende Abbildung fasst die dargestellten Schritte in einer Grafik zusammen. Dabei wird zwischen dem Einlesen der Daten (erster Kasten) und dem Ergebnis des Mapping (dritter Kasten) ein weiter Schritt durchgeführt, dessen Ziel ein effizienter Zugriff auf die Eingabedatei ist. Jeder Eintrag erhält eine Positionsangabe, um diesen im Map-Schritt schnell erreichen zu können.

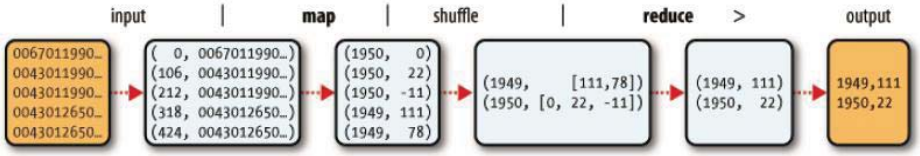


Abb. 1. Die beschriebenen Schritte des Beispiels mit Nennung der Phasen

2.2 Die MapReduce-Phasen im Detail

Wie das soeben vorgestellte Beispiel zeigt, muss sich der Nutzer um zwei Phasen des Frameworks kümmern: den Map-Schritt und den Reduce-Schritt. Diese sollen nun im Detail erläutert werden.

In der *Map-Phase* müssen alle relevanten Dokumente untersucht werden. Es werden dabei stets eine Vielzahl von Paaren, bestehend aus einem Schlüssel, der für die Gruppierung im anschließenden Shuffle-Schritt relevant ist, und einem Wert, der dem Schlüssel in diesem Paar zugeordnet ist, generiert.

Die entsprechende mathematische Funktion [5] lautet:

$$\text{Map: } K \times V \rightarrow (L \times W)^*$$

$$(k, v) \rightarrow [(l_1, x_1), \ldots, (l_{rk}, x_{rk})]$$

K und L entsprechen dabei den Schlüsseln, V und W die zugehörigen Werte. Die entsprechenden kleinen Buchstaben sind Instanzen vom jeweiligen Typ, beispielsweise Zahlen.

In der *Reduce-Phase* werden dann alle Werte, die in der Map-Phase für einen Schlüssel eingegeben wurden, ausgegeben. Der Nutzer hat also nur noch mit einem konkreten Schlüssel und der Verarbeitung von dessen Ausprägungen zu tun.

Hier lautet die mathematische Funktion wie folgt, wobei die Variablendefinition der oben genannten entspricht:

$$\text{Reduce: } L \times W^* \rightarrow W^*$$

$$(l, [y_1, \ldots, y_{sl}]) \rightarrow [w_1, \ldots, w_{ml}]$$

Zwischen den genannten beiden, durch den Nutzer zu implementierenden Schritten, findet die sogenannte *Shuffle-Phase* statt. Dabei werden jeweils für alle in der Map-Phase verwendeten Schlüssel die entsprechenden Inhalte zusammengetragen. Im Prinzip kann auch von einer Art Umstrukturierung gesprochen werden, da das anfängliche Strukturmerkmal der zugehörige Input (beispielsweise das zugehörige Dokument) ist. Durch die Shuffle-Phase geht diese Information verloren, wird aber durch den Schlüssel als neues Strukturierungsmerkmal ersetzt.

Die nachfolgende Abbildung [6] zeigt diese Zusammenhänge für das eingangs erwähnte Beispiel nochmals in übersichtlicher Form:

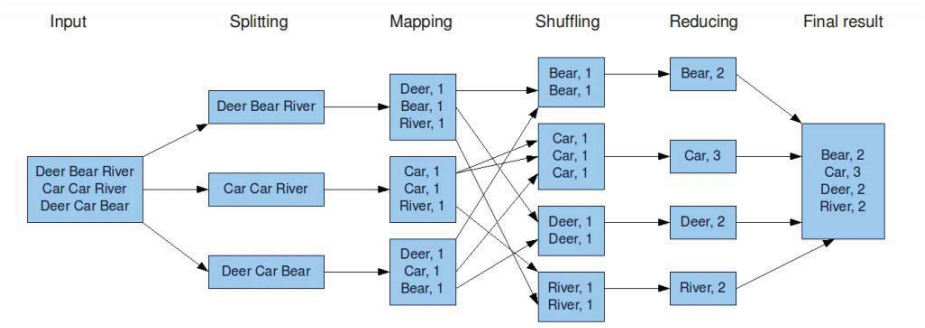


Abb. 2. Der gesamte MapReduce Prozess zum Zählen von Wörtern

Zusätzlich besteht noch die Möglichkeit, eine *Combine-Phase* zu definieren. Wie sich wahrscheinlich bereits vermuten lässt, ist diese Phase jedoch nicht verpflichtend. Es handelt sich vielmehr um eine Optimierungsmöglichkeit, die jedoch nicht auf jeden Anwendungsfall übertragen werden kann. Es handelt sich dabei um einen lokalen Reduce-Vorgang, der auf dem Ergebnis der Map-Phase – vor dem Shuffling – stattfindet. Ziel dieses Vorgangs ist es, bereits erkennbare falsche Ergebnisse frühzeitig auszusortieren, um den Aufwand für später, wo potentiell deutlich mehr Inhalte und Komplexität herrscht, zu verringern. Auch verursacht die Übertragung eines Zwischenergebnisses deutlich weniger Netzwerkbelastung, als dies bei allen Ergebnissen der Fall wäre. Leider kann die Combine-Phase nur dann eingesetzt werden, wenn ein Zwischenergebnis sicher nicht als künftiger Ausgabewert in Frage kommt (oder der Nutzer diese Entscheidung bewusst so trifft). Ein Beispiel hierfür wäre die Berechnung eines Mittelwerts, da dessen Ergebnis sich verändert, wenn erst Teil-Mittelwerte berechnet werden und über diese dann erneut ein Mittelwert gebildet wird. Aufgabenstellungen, die für ein Combine in Frage kommen, werden als „distributive“ bezeichnet [12]. Dann gilt: func(a, b, c, d) = func(func(a, b), func(c, d))

Exemplarisch wird dies in der nachfolgenden Abbildung [7] verdeutlicht. Ziel in diesem Fall ist es, den minimalen Wert zu ermitteln. Dabei können alle lokalen Nicht-

Minima bereits nach der Map-Phase als falsch identifiziert und damit in der Combine-Phase aussortiert werden.

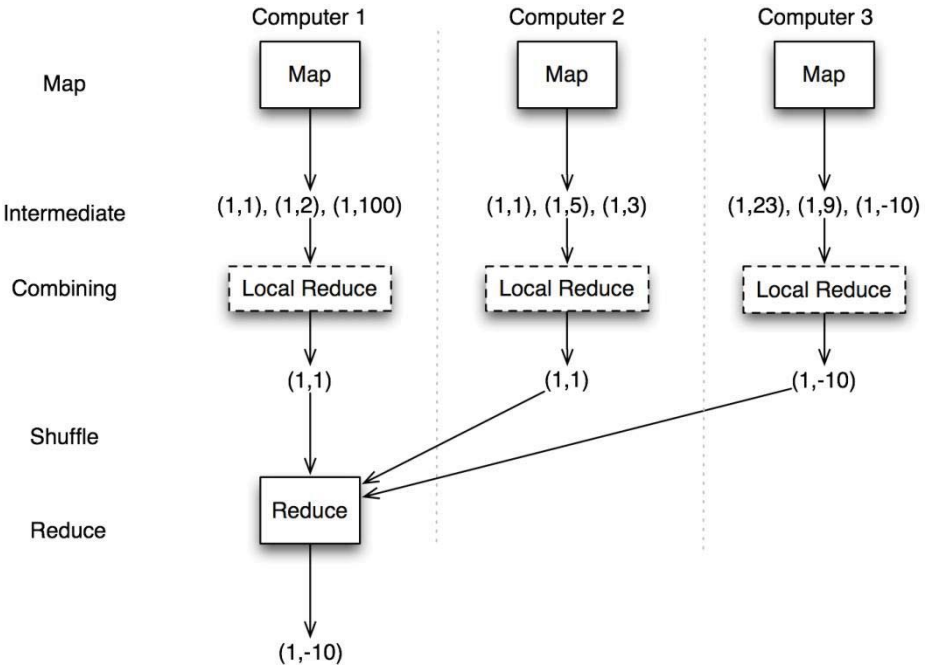


Abb. 3. Ein MapReduce Prozess einschließlich lokalen Reduce (Combine-Phase)

2.3 Abgrenzung zu relationalen Datenbankmanagementsystemen

Dem einen oder anderen mag die Frage aufkommen, warum für die parallele Verarbeitung nicht einfach eine (relationale) Datenbank samt der damit einhergehenden Abfragemöglichkeiten eingesetzt wird.

Die nachfolgende Tabelle [10, S. 5] zeigt die wesentlichen Unterscheidungsmerkmale, die nachfolgend kurz erläutert werden:

Tabelle 1 Vergleich zwischen traditionellen relationalen Datenbankmanagement-systemen und MapReduce

	Traditionelle RDBMS	**MapReduce**
Datengröße	Gigabytes	Petabytes
Zugriffart	Interaktiv und Batchverarbeitung	Batchverarbeitung
Änderungen	Mehrmaliges Schreiben und Lesen	Einmaliges Schreiben, mehrmaliges Lesen
Schema	Statisches Schema	Dynamisches Schema
Integrität	Hoch	Niedrig
Skalierbarkeit	Nicht immer linear	Linear

Wie die oben dargestellte tabellarische Zusammenfassung zeigt, bewegt sich MapReduce bei der Verarbeitung von Datenmengen im Bereich von Petabytes. Wie schon erwähnt, ist es speziell auf derartige Vielzahl und Größe an Daten ausgelegt. Natürlich können auch weniger Daten verarbeitet werden. Dies ist allerdings oftmals mit den gegebenen Mitteln einer Datenbank auch möglich, welche aber gleichzeitig zusätzliche Möglichkeiten anbietet, wie beispielsweise die verschiedenen Zugriffsmöglichkeiten. Diese können bei relationalen Datenbankmanagement-systemen zusätzlich interaktiv erfolgen, der Nutzer kann stets seine Anfrage binnen kürzester Zeit ändern. Auch können mehrfache Updates (Schreiboperationen) auf die Inhalte einer Datenbank erfolgen, was bei nur lesender Verarbeitung von MapReduce nicht vorgesehen ist. Die Datenstruktur unterscheidet sich ebenfalls. So werden die Inhalte bei relationalen Datenbanken in eine feste, statische Struktur gepasst, die vor dem Einspielen der Inhalte definiert werden muss. MapReduce benötigt keine derart feste Struktur, da es die Daten zur Laufzeit interpretiert, wenngleich auch durch den Nutzer definiert werden muss, wo MapReduce eben dann seine zu verarbeitenden Daten findet. Dies geschah im obigen Beispiel mit den Wetterdaten im vorbereitenden Schritt (Daten einlesen) und in der Map-Phase (eingelesene Daten verarbeiten). Damit einher geht auch eine unterschiedliche Integrität der Daten, da relationale Datenbanken diese häufig normalisiert abspeichern. Dies ist bei MapReduce weder vorgesehen noch, da zusätzliche Leseoperationen notwendig wären, um an die normalisieren Inhalte zu gelangen. Das wohl entscheidendste Argument für den Einsatz von MapReduce ist aber dessen lineare Skalierung (doppelte Datenmenge, doppelte Verarbeitungszeit, usw.), die auch eines der wesentlichen Ziele bei der Entwicklung von MapReduce darstellte. Relationale Datenbankmanagementsysteme können diese Werte leider nicht (einfach) erreichen, wobei es auch dort für einen Nutzer verschiedene Ansatzpunkte zur Verbesserung der Performance gibt.

Trotz der gerade eben vorgestellten Unterschiede nähern sich beide Systeme immer weiter an. Die verschiedenen Datenbankenhersteller versuchen das Konzept (oder bestimmte Eigenschaften) von MapReduce auf ihr Verarbeitungsmodell anzuwenden und mit Pig und Hive[2] sind Abfragesprachen geschaffen worden, die einem Datenbankentwickler den Umgang mit MapReduce erleichtern und vertrauter machen sollen.

3 Apache Hadoop

Mit der Entwicklung von Hadoop, einer Open-Source MapReduce Implementierung, wurde 2004 [8], nach der Veröffentlichung der Papers über MapReduce, begonnen. Es zielte darauf ab, die Skalierungsprobleme der zum damaligen Zeitpunkt weit verbreiteten Suchmaschine Nutch durch die mit dem Einsatz von MapReduce einhergehenden Vorteile zu lösen. Eine erste einsatzbare Version für Nutch lag 2005 vor. Im darauffolgenden Jahr bekam Hadoop Unterstützung durch Yahoo! und wurde unter Apache Software Foundation ein Lucene-Subprojekt [8]. 2008 stiegt Apache Hadoop sogar zum 2008 Top-Level-Projekt auf und beherbergt seitdem einige

[2] Es handelt sich hierbei um Subprojekte von Apache Hadoop, die in einzelnen auf den nachfolgenden Seiten vorgestellt werden.

Subprojekte, die nachfolgend kurz vorgestellt werden. Mit dem Hadoop-Framework gewann ein Cluster im Juli 2009 zweimal den Terabyte Sort Benchmark, der in unterschiedlichen Kategorien die leistungsfähigsten Systeme zur Datensortierung bestimmt. Außerdem konnte ein Hadoop-Cluster von Yahoo 100 Terabyte in zwei Stunden und 53 Minuten sortieren [8].

3.1 Aufbau eines MapReduce Jobs

Ein MapReduce Job besteht bei Apache Hadoop aus verschiedenen Teilen, die teilweise verpflichtend sind, andere aber auch freiwillig und damit zusätzlich definiert werden können.

Essentiell sind dabei die Angabe einer Mapper- und einer Reducer-Klasse, die jeweils ein entsprechendes Interface implementieren. Eine konkrete Beispielimplementierung findet sich als Abschluss dieses Kapitels. Die Angabe erfolgt über die Methoden *setMapperClass* und *setReducerClass* der entsprechenden Job-Instanz. Zusätzlich muss noch ein Pfad angegeben werden, wo die einzulesenden Dateien gefunden werden können, sowie ein Ausgabepfad für entsprechende Ergebnisse. Die Methoden hierfür lauten *setInputPath* und *setOutputPath*. Zum Schluss eines Jobs sollte der Aufruf *waitForCompletion(true)* erfolgen, um automatisch den Fortschritt solange anzufragen, bis der Job beendet ist.

Konkret könnte ein Apache Hadoop Job wie folgt [9] aussehen:

```
public static void main(String[] args) throws Exception
  {
    Job job = new Job(new Configuration());
    job.setJobName("MaxTemperature");

    job.setMapperClass(MaxTempMapper.class);
    job.setReducerClass(MaxTempReducer.class);

    job.setInputPath(new Path("in"));
    job.setOutputPath(new Path("out"));

    job.waitForCompletion(true);
  }
```

3.2 Ablauf eines MapReduce Jobs

Die nachfolgende Abbildung [10, S. 154] gibt einen groben Überblick über die unterschiedlichen Vorgänge während eines MapReduce Jobs. Die darin enthaltenen Begriffe sowie der genaue Ablauf werden anschließend an diese Grafik erläutert. Apache Hadoop entwickelt sich jedoch ständig weiter, so dass einzelne Teile der Abbildung inzwischen auf eine andere Weise realisiert sein können.

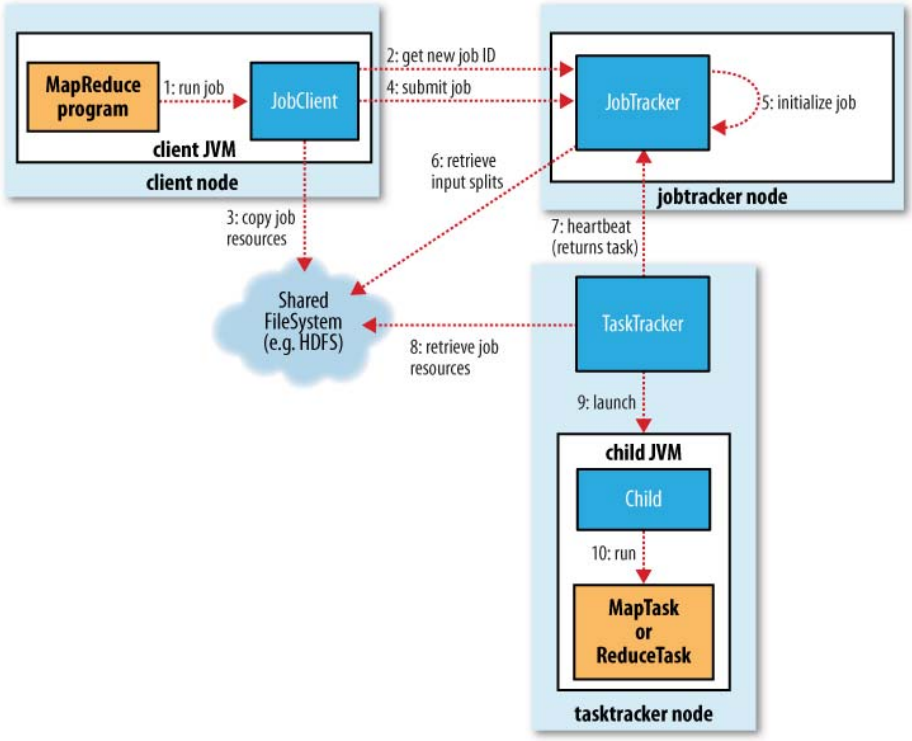


Abb. 4. Der Ablauf eines MapReduce Job in Apache Hadoop

Zunächst werden die einzelnen Begriffe der vorangehenden Abbildung erläutert:

Der *JobClient* enthält die im vorherigen Kapitel genannten Informationen bezüglich des Aufbaus eines Jobs. Bei einem *JobTracker* handelt es sich um eine Instanz, die benötigt wird, um die Ausführung der Jobs zu koordinieren. Dazu ist es notwendig, dass der JobTracker alle von ihm verwalteten Jobs kennt bzw. diese sich bei ihm registrieren. Er teilt zudem einen Job in verschiedene Arbeitsschritte auf. Diese werden von einem TaskTracker übernommen, wobei pro Task ein Prozessor-Kern läuft. Der *TaskTracker* führt die eigentlichen Map- und Reduce-Aufgaben aus. Alle Komponenten teilen sich dabei ein *gemeinsames Dateisystem (HDFS)*, welches speziell auf die Bedürfnisse von Apache Hadoop und MapReduce zugeschnitten ist. Details zu diesem Dateisystem können unter den weiteren Bestandteilen von Apache Hadoop nachgelesen werden.

Die nachfolgenden Nummern in Klammern verweisen auf die jeweilige Nummer in der Ablaufdarstellung. Um einen Job unter Apache Hadoop auszuführen, muss dieser zunächst erstellt werden. Der Aufbau des Jobs wurde im vorangegangenen Kapitel erläutert. Dieser wird nun an eine neue JobClient-Instanz übergeben (1). Anschließend wird eine eindeutige Job-ID beim JobTracker beantragt (2). Ist dies erledigt, werden alle Job-Dateien ein das gemeinsame Dateisystem übertragen, so dass diese der gesamten Hadoop-Umgebung zur Verfügung stehen (3). Dies beinhaltet neben dem Programm auch zusätzlich benötigte Bibliotheken und

Konfigurationsdateien. Im nächsten Schritt wird dem JobTracker durch den Aufruf von *submitJob* mitgeteilt, dass alle Daten bereitstehen und der Job bearbeitet werden kann (4). An dieser Stelle erfolgt eine Überprüfung hinsichtlich der logischen Korrektheit der Ein- und Ausgabeparameter des Jobs. Läuft diese positiv ab, wird der Job vom JobTracker initialisiert und in eine Job-Warteschlage registriert (5). Dabei wird der Job in mehrere Tasks aufgeteilt. Die zuvor vom JobClient im Dateisystem gespeicherten Informationen werden nun vom JobTracker abgeholt (6). Dies geschieht in Form von Input Splits, welche logische Referenzen auf einen entsprechenden Block im Dateisystem darstellen. Die TaskTracker erhalten nun einzelne Tasks und benachrichtigen den JobTracker Informationen in Form von Statusmeldungen (heartbeat), wobei ein JobTracker nur eine begrenzte Anzahl an Tasks gleichzeitig bearbeiten kann (7). Der TaskTracker erhält die zur Ausführung des Tasks relevanten Daten aus dem Dateisystem (8). Dabei gilt, dass jeder Map-Task genau einen Input-Split verarbeitet. Zur Ausführung des Tasks wird eine TaskRunner-Instanz in einer neuen Java Virtual Machine (JVM) gestartet (9). Somit läuft jeder Task in einer eigenen JVM. Dort wird der tatsächliche MapTask oder ReduceTask durchgeführt (10). Als Ergebnis wird der Statuswert *finished* oder *failed* an den TaskTracker kommuniziert. Wenn alle Tasks abgeschlossen sind, meldet dieser dem JobTracker, der er fertig ist und weitere Aufgaben übernehmen kann.

3.3 Merkmale von Apache Hadoop

Natürlich beruht Apache Hadoop auf einer Vielzahl an Merkmalen und Prinzipien. Einige ausgewählte Merkmale werden nun kurz erläutert:

Fehlertoleranz. Ein Grundprinzip, auf welchem Apache Hadoop basiert, ist die Annahme und Akzeptanz von Hardwareausfällen. Dies ist vor allem dadurch begründet, dass Apache Hadoop speziell auch mit einer Vielzahl von kleineren und leistungsschwächeren Maschinen umgehen kann. Die dort vorkommenden Probleme, z.B. Festplattendefekte, müssen bereits im Vorfeld mit berücksichtigt werden. Daher wurde Apache Hadoop mit einer hohe Fehlertoleranz versehen, was praktisch eine mehrfache Datenhaltung und Datenverarbeitung bedeutet.

Streaming-API. Obwohl Apache Hadoop in Java implementiert und zunächst nur dafür Jobs geschrieben werden konnten, wurde Apache Hadoop um eine sogenannte Streaming-API erweitert. Dadurch wird die Open-Source MapReduce Implementierung auch für eine Vielzahl anderer Programmiersprachen zugänglich. Dies wird durch die Nutzung primitiver Ein- und Ausgaben (z.B. Datei Zeile für Zeile einlesen) und zusätzlich ausführbaren Skripten in der jeweiligen Programmiersprache ermöglicht, die die Verarbeitung des MapReduce Jobs steuern.

HDFS. Zusätzlich nutzt Apache Hadoop ein eigenes Dateisystem (Hadoop Distributed File System), welches in früheren Versionen verpflichtend vorgeschrieben war und aufgrund der speziell in verteilten Systemen gegebenen Charakteristika auch immer noch für Produktivsysteme empfohlen wird. Es basiert auf den folgenden Prinzipien: Dateien, die dort verwaltet werden, können riesengroß (mehrere

Terabytes) sein. Dabei wird von einem einmaligen Schreibvorgang ausgegangen. Lesezugriffe finden selbstverständlich mehrfach statt. Außerdem wird als Hardwarebasis von vielen billigen Rechnern ausgegangen, so dass mit Fehlern und Ausfällen ständig gerechnet werden muss. Entsprechend dieser Eigenschaften ist HDFS nicht mit Systemen, die extrem schnelle Reaktions- und Lesezeiten erfordern, vielen kleinen Dateien enthalten oder vielfache Schreibvorgängen durchführen, kompatibel.

3.4 Subprojekte von Apache Hadoop

Als Apache-Top-Level-Projekt bietet Hadoop ein Dach für viele kleine Projekte, die Hadoop um weitere Funktionalitäten erweitern. Eine Auswahl dieser Subprojekte [11, S. 41ff. und 301ff.] [12, S. 212ff.] wird nun kurz erläutert[3]:

HBase. Bei HBase handelt es sich um eine verteilte, skalierbare Datenbank, die speziell auf sehr große Datenmengen auslegt ist und in einen Hadoop-Cluster integriert werden kann. Als Basis wird ebenfalls HDFS als Dateisystem verwendet. Dabei implementiert HBase die Prinzipien von Googles Datenbank BigTable, welche auf wenige Änderungen, aber viele Ergänzungen ausgelegt ist.

Hive. Mit diesem Subprojekt wird Apache Hadoop um Data Warehouse Funktionalitäten erweitert, die sich thematisch sehr gut in diese Landschaft integrieren lassen, da gerade im Bereich Data Warehouse Unmengen an Daten anfallen, die effizient verarbeitet werden müssen. Dabei ist die Verwendung von Hive ein wenig SQL-orientiert und zwingt den Nutzer eine gewisse Strukturierung in die vorgegebenen Daten zu bringen.

Pig. Pig bildet eine zusätzliche Abstraktionsebene zwischen dem Nutzer und Hadoop, um eine einfache, aber auch erweiterbare Basis zu schaffen, auf der der Nutzer operieren kann. Zugleich bietet dies die Möglichkeit, komplexe Operationen automatisch zu optimieren. Dem Nutzer steht für Interaktionen die für Pig eingeführte Sprache, *Pig Latin*, zur Verfügung.

Chukwa. Dieses Subprojekt dient der Verwaltung und Verarbeitung von riesigen verteilten Systemen. Es setzt auf HDFS und Apache Hadoop auf und sammelt Daten aus verschiedenen Systemen. Häufig wird Chukwa im Zusammenhang mit der Erstellung und Handhabung von Log-Dateien verwendet.

ZooKeeper. ZooKeeper unterstützt den Nutzer beim Entwickeln verteilter Anwendungen und ist daher nicht ausschließlich für Hadoop vorgesehen, wenngleich dessen Nutzung bei Hadoop aufgrund der verteilten Verarbeitung sinnvoll ist. Die folgenden Eckpunkte zeichnen ZooKeeper aus: Es soll hohe Verfügbarkeit

[3] Detaillierte Informationen zu den jeweiligen Projekten können unter dem Menüpunkt *Related Projects* auf der Website von Apache Hadoop (https://hadoop.apache.org/) einsehen werden.

sicherstellen, lose Kommunikation vereinfachen und will dies als Bibliothek mit einfachen, aber mächtigen ausdrücken erreichen.

Die nachfolgende Abbildung zeigt nochmals eine grobe Einordnung der soeben kurz dargestellten Subprojekte in Relation zu MapReduce und Apache Hadoop. So setzen Hive, Chukwa und Pig auf dem bestehenden Apache Hadoop auf und bieten neue oder umgestaltete Funktionalitäten an. Das Dateisystem HDFS bildet wiederum die Zugriffsschicht für Apache Hadoop auf die eigentliche Platte, was ebenfalls von Hbase genutzt wird. ZooKeeper bietet für alle Technologien interessante Querschnittsfunktionen.

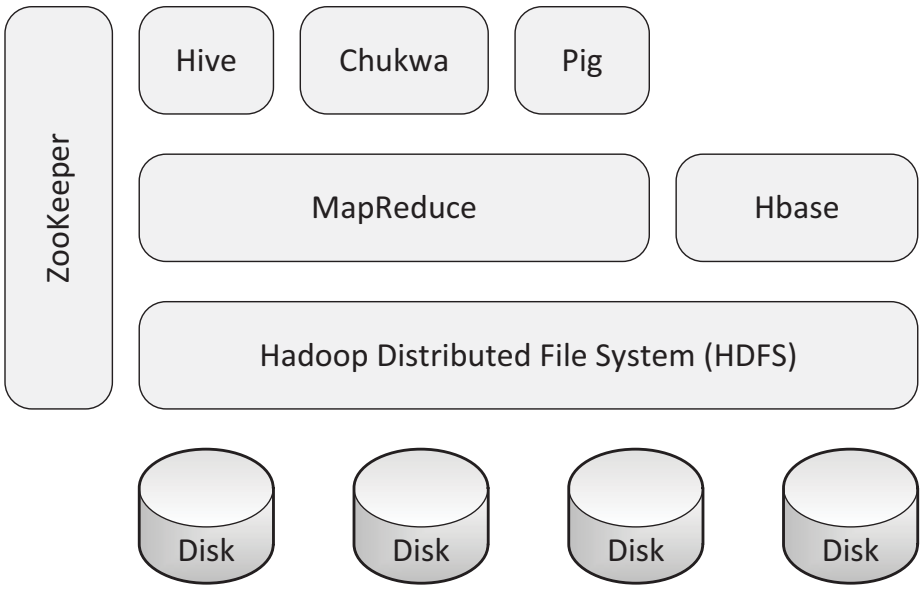


Abb. 5. Einordnung von Apache Hadoop (MapReduce) und dessen Subprojekten

3.5 Konkretes Codebeispiel mit Apache Hadoop

Nachdem nun auf den Aufbau, den Ablauf, weitere Merkmale und Subprojekte von Apache Hadoop eingegangen wurde, sollen nun der Map- und der Reduce-Schritt in einer konkreten Java-Implementierung gezeigt werden. Hierzu wird das oben vorgestellte Beispiel zur Ermittlung der höchsten Temperatur aufgegriffen, wobei für die Beispielimplementierungen ganzzahlige Temperaturangaben angenommen werden.

Map-Implementierung. Beim Mapping wird der aktuelle Datensatz zunächst in die Variable *line* geschrieben. Aus diesem wird dann die Jahreszahl (*year*) und die Temperaturangabe (*intTemperature*) extrahiert. Aus diesen beiden Angaben kann ein Schlüssel-Wert-Paar erzeugt werden, welches mittels *output.collect* dem nächsten Schritt übergeben wird.

```
public static class MaxTempMapper extends Mapper<Text,
  Text, Text, IntWritable> {

  public void map(Text key, Text value, Context
    context) throws IOException, InterruptedException {

      //Datensatz einlesen
      String line = value.toString();
      //Jahreszahlenangabe daraus extrahieren
      String year = line.substring(15, 19);
      //Temperatur daraus extrahieren
      int airTemperature = Integer.parseInt(line.
        substring(87, 92));
      //Weitergabe des erzeugten Schlüssel-Wert-Paares
      output.collect(new Text(year),
        new IntWritable(airtemperature));

  }
}
```

Reduce-Implementierung. Der nachfolgende Reduce-Schritt erstellt zunächst eine temporäre Variable namens *maxValue*, die bei der Initialisierung auf den minimalen Wert gesetzt wird, um keine Verfälschungen bei dem folgenden Vergleich zu verursachen. Anschließend wird über alle Werte des aktuellen Jahres bzw. Schlüssels iteriert und über die *Math.max*-Funktion jeweils der höchste Wert ermittelt, der wiederum in die temporäre Variable geschrieben wird. Am Ende enthält diese den höchsten Wert und kann über *output.collect* weitergereicht werden.

```
public static class MaxTempReducer extends
  Reducer<Text, Text, Text, IntWritable> {

  public void reduce(Text key, Iterable<IntWritable>
    value, Context context) throws IOException,
    InterruptedException {

      //Variable für die temporär höchste Temperatur
      int maxValue = Interger.MIN_VALUE;
      //Iteration über alle Werte des Jahres
      //(Schlüssels) und dabei Vergleich mit maxValue
      while (values.hasNext()) {
        maxValue = Math.max(maxValue, values.next()
          .get());
      }
      //Weitergabe des erzeugten Schlüssel-Wert-Paares
      output.collect(key, new IntWritable(maxValue));
  }
}
```

Beide Klassen implementieren dabei ein entsprechendes Interface *Mapper* oder *Reducer*, welche mit vier verschiedenen Typen parametrisiert wurden (in Java: *Generics*). Dabei handelt es sich um die Ein- und Ausgabetypen für Schlüssel und Wert. Die Syntax lautet: <*Schlüssel_Input, Wert_Input, Schlüssel_Output, Wert_Output*>.

Der zugehörige Job-Aufruf in Java wurde bereits bei der Beschreibung des Aufbaus von Apache Hadoop dargestellt.

References

1. http://news.cnet.com/8301-1023_3-57498531-93/facebook-processes-more-than-500-tb-of-data-daily besucht: September, 2012.
2. http://bigdatanerd.wordpress.com/2012/02/12/hadoop-vs-rdbms-where-hadoop-scores-over-rdbms besucht: September, 2012.
3. http://research.google.com/archive/mapreduce.html besucht: September, 2012.
4. http://googleblog.blogspot.de/2010/06/our-new-search-index-caffeine.html besucht: September, 2012.
5. https://de.wikipedia.org/wiki/MapReduce besucht: September, 2012.
6. http://devveri.com/hadoop/hadoop-mapreduce-ornek-uygulama besucht: September, 2012.
7. https://www.datadr.org/doc/introduction.html besucht: September, 2012.
8. http://research.yahoo.com/files/cutting.pdf
   http://www.heise.de/developer/artikel/Verarbeiten-grosser-verteilter-Datenmengen-mit-Hadoop-964755.html besucht: September, 2012.
9. https://hadoop.apache.org/docs/current/api/index.html besucht: September, 2012.
10. White, T. (2009): Hadoop: The Definitive Guide. Sebastopol, CA: O'Reilly. (2009)
11. Lam, C.: Hadoop in Action. Greenwich, Conn: Manning Publications. (2011)
12. Jim G., Adam B., Andrew L., Hamid P.: Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. Institute of Electrical and Electronics Engineers, Inc. (1995)
13. Jeff Dean J.: Designs, Lessons and Advice from Building Large Distributed Systems-einsehbar unter: http://www.scribd.com/doc/21244790/Google-Designs-Lessons-and-Advice-from-Building-Large-Distributed-Systems besucht: September, 2012.

Exploring Query Patterns in Email Search

Daniel Thomä

Otto-Friedrich-Universität Bamberg, Lehrstuhl für Medieninformatik,
daniel-felix.thomae@stud.uni-bamberg.de

Zusammenfassung Die klassische E-Mail hat nach einer langen Erfolgsgeschichte noch immer eine hohe Bedeutung als Kommunikationsmedium im geschäftlichen und privaten Bereich. Umso erstaunlicher erscheint es, dass bisher nur sehr wenig über das Suchverhalten der Anwender und deren Suchanfragen an Mail Clients bekannt ist. Zudem scheint sich bisher keine gemeinhin akzeptierte Darstellungs- und Suchfunktionalität für solche Anwendungen herauskristallisiert zu haben, welche vom Standard mit Listen und Ordnern abweicht. Im Folgenden wird also eine der jüngsten Veröffentlichungen in diesem Bereich von Harvey und Elsweiler[6] vorgestellt und kommentiert. Anschliessend werden einige alternative Ansätze für E-Mail Verwaltungssoftware vorgestellt und diskutiert, sowie einige mögliche Begründungen genannt, warum dieser Bereich bisher so sehr "vernachlässigt" wurde. Schliesslich wird ein eigenes Visualisierungskonzept vorgestellt und eine Prognose für die zukünftige Entwicklung gewagt.

Schlagworte: E-Mail Suche, Ontologien, E-Mail Darstellung, Concept Lattices, Semantic E-Mail Processing, Query-Expansion, Design-Vorschlag, Konzept

1 Besonderheiten der Mail- und Desktopsuche und Vorstellung des Basisartikels

Im Kontrast zur "klassischen" Mediensuche, bspw. der Suche nach Büchern in einer Bibliothek, der Suche nach Internetseiten auf Suchmaschinen wie Google[1] oder gar der Suche nach Informationen als solches in einer semantischen "Wissens-Engine" wie Wolfram[2] gibt es bei der Mail oder auch der Desktop Suche einige Unterschiede. So ist der allgemeine Ablauf in den genannten Fällen in etwa wie folgt: Auftreten eines Informationsbedürfnisses, Suche nach Seiten oder Büchern welche dieses Thema behandeln und somit im konkreten Fall weiterhelfen könnten, Studium der Ergebnisse und, sollte das Bedürfnis nicht befriedigt sein, ggf. eine Verfeinerung oder neue Anfrage. Sucht ein Nutzer nach E-Mails oder Dateien auf seinem Rechner, so sind dies persönliche Daten, d.h. Mails die vom oder an den User gesendet wurden, Dateien wie Dokumente und Bilder

[1] http://www.google.com/
[2] http://wolframalpha.com

die erstellt oder verwaltet werden. Diese Suchanfragen unterscheiden sich nicht nur durch die geringere Grösse der Kollektionen von der Web-Suche, sondern auch darin dass der Nutzer oft sehr genau weiss, welche Mail respektive Datei er sucht und gezielt selbige nochmals vorgehalten haben möchte, sei es um ein konkretes Informationsbedürfnis zu befriedigen (bspw. einen Termin oder Deadline nachlesen) oder um sich das Ganze nochmals anzusehen (bspw. Urlaubsbilder, Weihnachtsmails von Freunden etc.). Er sucht also nicht "ins Blaue hinein" nach Informationen, sondern meist bewusst nach bekannten Daten, was natürlich auch zu Unterschieden im Umgang mit und Anforderungen an die Suchfunktionalitäten solcher Systeme führt. Einige Zusammenhänge wurden von Morgan Harvey und David Elsweiler untersucht und sollen nachfolgend genauere Betrachtung finden. Weitere Ansätze, welche vor allem der Orientierung in diesen Datensätzen dienen werden in den abschliessenden Kapiteln beleuchtet.
Die Publikation von Harvey und Elsweiler: "Exploring Query Patterns in Email Search" [6] wurde im April 2012 auf der ECIR, der European Conference on Information Retrieval in Barcelona vorgestellt und ist somit sehr aktuell. Aufbauend auf früheren Veröffentlichungen der Erlangener und Regensburger Forscher ist sie in Form einer Kleinstudie gehalten. Über einen Zeitraum von vier Monaten wurde das Suchverhalten in Desktop-Mailsoftware von 47 Testpersonen, davon 37 Männer und zehn Frauen im Alter von 21-49 Jahren mit insgesamt 7 verschiedenen Nationalitäten untersucht. Dazu haben die Autoren eine Erweiterung für den bekannten, plattformübergreifenden open source Mail-Client Mozilla Thunderbird[3] geschrieben, welche automatisiert geöffnete Mails, Ordner- sowie Sortierungsklicks und Suchanfragen speichert. Die auf diese Weise gesammelten Informationen wurden in sogenannten *Query-Chains* organisiert. Eine Query-Chain enthält immer eine Suchanfrage und alle daraufhin geöffneten Mails und endet mit der Durchführung einer neuen Anfrage, einem Neustart des Programms oder nach einer fünf minütigen Phase der Inaktivität. Im Rahmen des vorliegenden Artikels wurden zunächst nur Anfragen und geöffnete Mails untersucht und jegliche Sortierungs- oder Filterklicks ausgeblendet. Eine Untersuchung selbiger soll in späteren Veröffentlichungen erfolgen. Zur Operationalisierung der Ergebnisse wurden zudem nur Anfragen erfasst, nach welcher auch tatsächlich eine Mail geöffnet wurde (was immerhin eine Reduktion auf 59% der Gesamtanfragen ausmacht). Somit kommen insgesamt knapp 1500 Query-Chains für eine weitere Analyse in Frage. Die beiden folgenden Kapitel beschäftigen sich detailliert mit den Ergebnissen des Artikels.

[3] http://www.mozilla.org/thunderbird/

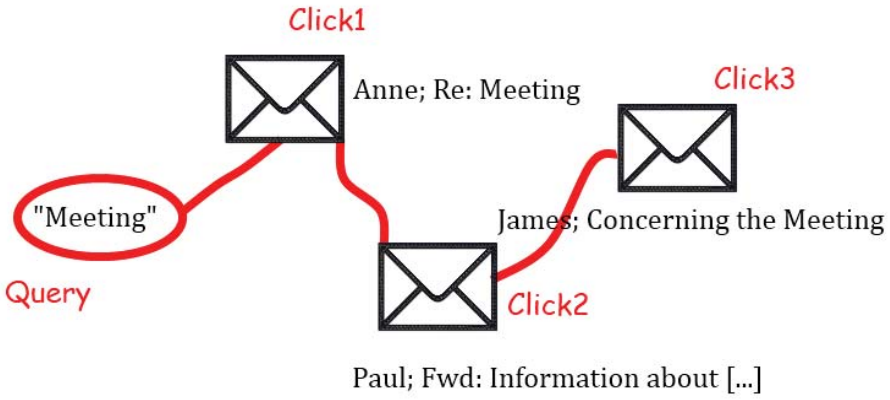


Abbildung 1. Veranschaulichung der Query-Chains als Grundlage der Analyse.

2 Erster Blick auf die Daten

Die Eckpunkte der Analysedaten geben schon einen ersten Überblick über die Messung und weisen bereits deutliche Diskrepanzen zu den für Web-Suche üblichen Werten auf. So enthält eine Anfrage oder Query im Schnitt lediglich 5.87 Zeichen, also meist auch nur Wortteile. Zudem besteht der Suchbegriff in 94% der Fällen aus einer einzigen Zeichenkette und nur 0.56% der Begriffe sind mehr als zwei Wörter, was insgesamt zu einem Schnitt von 1.07 Wörtern pro Anfrage führt. Dies ist erheblich kürzer als bisher veröffentlichte Ergebnisse aus Desktop-, Web- und Mail-Suche Studien welche die Autoren als durchschnittlich 1.6 Wörter[5], 12.1 Zeichen[2] bzw. 1.4 Wörter[13] nennen. Auch die Analyse der Anfragewiederholungen lässt einen ersten Eindruck zu, bevor im folgenden Kapitel näher darauf eingegangen wird. Insgesamt wird ein Anteil von 45.3% der Suchanfragen vom selben Anwender wiederholt, wobei in der Summe eine Quote von 7.4% nutzerübergreifend wiederholt wird - was sich beides wohl ganz gut mit den Daten für die Web-Suche deckt. Dies mag zunächst überraschend erscheinen, brechen doch im Vergleich zu klassischen Suchmaschinen Anfragen bezüglich aktueller Events, bekannten Persönlichkeiten oder auch einfach häufige Suchbegriffe weg: “EM 2012”, “Olympia London”, “Angela Merkel” oder “Grill Steaks” werden wohl kaum als gemeinsamer Nenner eines Grossteils der Mailnutzer auftreten. Andererseits jedoch, findet man durchaus häufige Vor- bzw. Nachnamen, Begriffe wie “Deadline”, “Presentation” oder auch “Powerpoint” bei einer Mehrheit der User, was diesen Wert wiederum plausibel macht. Ausserdem lässt sich bereits aus dieser groben Betrachtung, neben der Häufigkeit von Namen als Suchbegriffen, die grosse Bedeutung von Personen in der Mail-Suche erkennen: ein grosser Anteil der Klicks in den erfassten Query-Chains stammt von den Top k Kontakten der Nutzer. Hierzu wurde die Häufigkeit mit welcher die Kontakte dem jeweiligen User eine E-Mail schicken als Rangfolgekriterium genutzt. Das Ergebnis dieser Untersuchung zeigt den grossen Einfluss deutlich.

So sind in 40% aller Query-Chains Klicks auf Mails von mindestens einem der Top fünf Kontakte enthalten und über alle Chains hinweg verteilt macht die Anzahl der Klicks auf Mails der Top fünf Kontakte gar 25%, sowie des wichtigsten Kontakts immerhin 16%, der Gesamtklicks aus. Eine weitere statistische Analyse der Autoren zeigt, dass die Top Kontakte im Vergleich zum Anteil der Mails, welche sie ausmachen, eine komparativ wesentlich höhere "Klickwahrscheinlichkeit" aufweisen als bei der reinen Streuung über die Binominal- bzw. Multinominalverteilung zu erwarten wäre, was in Abb. 2 illustriert ist. D.h. sollte der Top Kontakt bspw. 10 der 100 Gesamtmails in der Kollektion geschrieben haben, so macht dieser Kontakt einen deutlich grösseren Anteil als 10% der Gesamtklicks in den Query-Chains aus. Auch lässt sich ein Punkt erkennen (in diesen Messdaten zwischen k=6 und k=10) an welchem die Kontakte unterdurchschnittlich interessant werden, d.h. seltener als statistisch erwartet in den Suchanfragen angeklickt werden. Diese Ergebnisse klingen nachvollziehbar, denn oft hat man eine handvoll enger Kontakte vor allem im persönlichen Arbeitsumfeld, welche sehr viele und sehr relevante Mails schreiben, die man häufig zu Referenzzwecken nochmals benötigt. Auf der anderen Seite gibt es eine Menge an Verteilern und peripheren Kontakten, welche meist nur "zur Kenntnis" genommen werden, aber deren Nachrichten typischerweise von keiner grösseren Relevanz sind - und somit auch in den Suchanfragen "unterrepräsentiert" erscheinen.
Schliesslich lässt sich auch eine erste Aussage zum Zeitverlauf der Wiederholungen von Anfragen machen. Vergleichbar mit anderen Ergebnissen zur Web-Suche[10] ist hier direkt nach der ersten Verwendung eines Suchbegriffes die Wahrscheinlichkeit einer Wiederholung am grössten. Danach fällt diese rapide ab, um nach ca. 20 Tagen in etwa konstant zu bleiben, was darauf hindeutet, dass manche Nachrichten auch nach sehr langer Zeit noch relevant bleiben und der Verteilung somit ein sog. Long-Tail Charakter[4] unterstellt werden kann (siehe Abb. 3).

[4] http://en.wikipedia.org/wiki/Long_tail

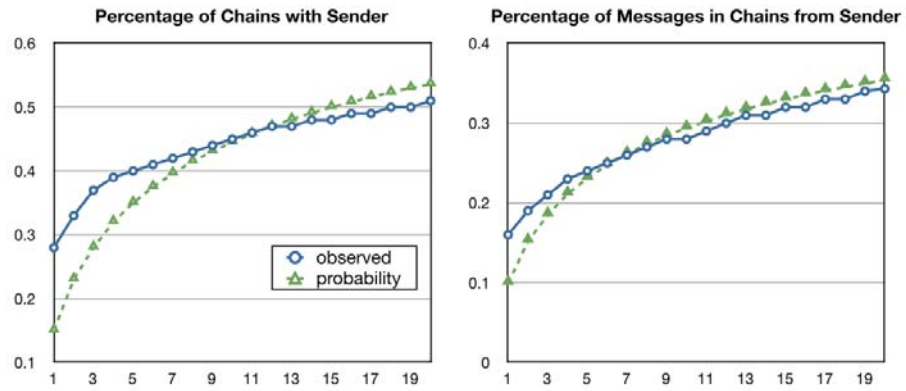


Abbildung 2. Tatsächliches Auftreten vs. Wahrscheinlichkeit des Auftretens nach statistischer Verteilung von Nachrichten der Top k Kontakte.

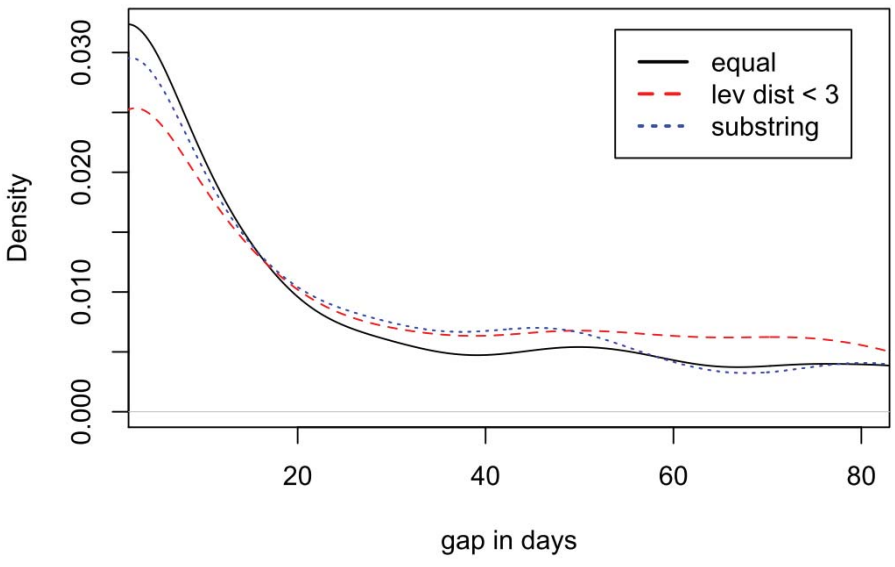


Abbildung 3. Wiederholung der gleichen Suchanfragen im Zeitverlauf.

3 Detail-Analyse der Wiederholungen und Überlappungen von Suchanfragen

Die weitere Analyse der Daten bezieht sich zu grossen Teilen auf Tabelle 1, welche aus dem Artikel von Harvey und Elsweiler übernommen wurde. Es lässt sich erkennen, dass gleiche Anfragen gegenüber unterschiedlicher Anfragen nur mit einer geringen Mehrwahrscheinlichkeit zu Überschneidungen in den Query-Chains führen. So gibt es bei identischen Anfragen in 59% der Fälle Überschneidungen in den Klicks (Anteil von Feld eins und Feld zwei an den gleichen Anfragen). Bei unterschiedlichen Anfragen sind es immerhin noch 49% (analog für Felder vier und fünf). Als Erklärung hierfür lassen sich zwei Sachverhalte anführen. Zum Einen sind, wie bereits erwähnt, E-Mail Datenbanken verhältnismässig klein mit einigen hunderten oder tausenden Einträgen, während zum Anderen die Query-Chains mit im Mittel 5.17 Klicks relativ lang sind, was automatisch zu mehr Überlappungen als bspw. bei der Web-Suche führt, wo laut Tyler[13] der Durchschnitt unter einem Klick pro Suchanfrage liegt - und zudem wohl einige Milliarden Einträge im Index stehen.
Auch der Umkehrfall Feld drei, also wiederholte Suchanfragen welche zu keinerlei Überschneidungen bei den Klicks geführt haben, weicht mit 18.7% deutlich von den 4% ab, welche Teevan et al.[12] für Web-Suchmaschinen angeben. Dies kann laut Aussage der Autoren über das Suchverhalten der Nutzer erklärt werden. So sei hier eine andere Nachricht des gleichen Absenders gesucht, was die fehlende Überlappung der Klicks bedingen soll. Unterstützt wird diese Vermutung dadurch, dass in diesen Fällen 39.4% der angeklickten Mails den gleichen Absender haben und wie in der groben Betrachtung bereits erwähnt wurde, Anwender häufig mit Namen bzw. Absendern als Suchbegriff arbeiten. Das ist eine Angewohnheit, welche sich auch auf die Klicküberschneidungen bei verschiedenen Suchanfragen auswirkt (Felder vier und fünf). Eine weitere Analyse der Autoren konnte zeigen, das es sich hierbei häufig nicht um verschiedene Anfragen "per se" handelt, sondern eigentlich um die selbe Suche, welche nur alternativ formuliert wurde. Es wurden hier sehr ähnliche Begriffe verwendet wie bspw. Abkürzungen von Sendernamen, vermutlich mit der Absicht die selbe Nachricht wie bei einer vorherigen Suche mit einem minimalen Aufwand wiederzufinden. Vor diesem Hintergrund konnten die Autoren zeigen, dass eine geringe Levenshtein-Distanz[5] zwischen zwei Suchbegriffen, d.h. $D \in \{0, 1\}$ eine Absicht anzeigt, alte Ergebnisse wiederzufinden (anhand proportional grösserer Überschneidungen) und dass ab einer Distanz von $D \geq 2$ wieder eher nach verschiedenen Inhalten gesucht wird. Dies kann gemeinsam mit der bereits angesprochenen Anfrage-Zeit Kurve (siehe Abb. 3) zur Erkennung der Nutzerabsicht verwendet werden um in einer zukünftigen Generation intelligenter User-Interfaces entsprechend Ergebnisse zu sortieren/präsentieren. Konkret bedeutet dies je geringer die Levenshtein-Distanz und je geringer der zeitliche Abstand zur ersten Anfrage, desto grösser ist die Wahrscheinlichkeit, dass der Anwender die Absicht verfolgt, ein Ergebnis wiederzufinden. Ein weiterer Unterschied zur konventionellen (Web-) Suche

[5] http://de.wikipedia.org/wiki/Levenshtein-Distanz bzw. auch Literaturverzeichnis[8]

welche Harvey und Elsweiler aus ihren Daten herauslesen, liegt im temporären Verlauf der Query-Chains bei identischen Suchanfragen. Während hier bei der Suchmaschinenanfragen ein Muster zu beobachten ist, in welchem eine geringe Zeitspanne (bis hin zu einigen Stunden) eine sehr geringe Überschneidung aufweist und bei Suchen, welche nach sehr langer Zeit (über 30 Tage) wiederholt werden wiederum eine sehr hohe Überschneidungswahrscheinlichkeit von $\geq 90\%$[12] auftritt, ist dies in den Daten der Studie nicht zu erkennen.
Ein möglicher Erklärungsansatz hierfür zielt auf die unterschiedlichen Anwendungsbereiche der beiden Medien ab: während bei der Websuche hauptsächlich nach dem eingangs bereits erwähnten *"suchen, sichten, erneut suchen"* Rythmus vorgegangen wird, gibt es selbigen bei der Suche nach Mails offenbar nicht. Konkret heisst das, dass wenn eine Anfrage bei einer Suchmaschine innerhalb einiger Stunden wiederholt wird weitere Quellen zum selben Thema gesucht werden und nach Tagen oder Wochen dann eben genau diese Seite die bei der ersten Suche schon geholfen hatte. Bei E-Mails wird hingegen anscheinend üblicherweise nach bestimmten Nachrichten gesucht, welche unabhängig vom Zeitverlauf eben das bestehende Informationsbedürfnis befriedigen - was dazu führt, dass neue Mails im Zeitverlauf "aktuell" für die Suche werden, was hier die Diskrepanz zur Web-Suche darstellt.

Query-Chains Gesamt 1467 (100%)	**Überlappende Ketten - 784**		**Keine Überlappung** 932 (63.5%)
	Identisch 138 (9.4%)	**Überschneidend** 646 (44%)	
Gleiche Anfragen 665(45.3%)	① 58 (4.0%)	② 333 (22.7%)	③ 274 (18.7%)
Verschiedene Anfragen 802(54.7%)	④ 82 (5.6%)	⑤ 313 (21.3%)	⑥ 407 (27.7%)

Tabelle 1. Detaillierte Übersicht über die Überlappungen bei Suchanfragen und Query-Chain Klicks.

4 Diskussion des Artikels

Die vorliegende Studie und der begleitende Artikel geben einen ersten Einblick in das Mail Suchverhalten der Nutzer und decken einige grundlegende Kausalzusammenhänge auf. Als wichtigste Erkenntnisse sind hier vor allem die Design-Implikationen zu nennen, welche man anhand der Ergebnisse ablesen kann. So zum Beispiel eine Berücksichtigung der Ähnlichkeit zwischen Suchanfragen sowie dem temporären Aspekten des "Alters" der Nachricht und der eventuell bereits getätigten, sehr ähnlichen Anfragen um eine verbesserte Priorisierung der Suchergebnisse in zukünftigen Versionen von intelligenten Nutzer-Interfaces zu realisieren. Auch eine Sortierfunktion welche nach Top k Sender oder der Anzahl der angeklickten Mails ordnet, würde in diesen Bereich fallen.
Als Kritikpunkt muss zunächst einmal die Datengrundlage der Studie genannt

werden. Eine Grundgesamtheit von weniger als 50 Personen, noch dazu mit knapp 80% Männeranteil welche wohl zudem noch einer gemeinsamen, klar abgrenzbaren Gruppe an Nutzern entstammen kann sicher nicht die Verallgemeinerbarkeit oder Validität der Ergebnisse unterstreichen. Selbiges war höchst wahrscheinlich auch nicht das Ziel dieser Untersuchung, es sei dennoch angemerkt, dass hier vermutlich ein Bias vorliegt und weitere grösser angelegte Studien nötig sind. Des Weiteren fehlen jegliche Beachtungen der ebenfalls erfassten Order- bzw. Sortierungsklicks. Mit Einbezug selbiger in die Untersuchungen liessen sich womöglich an manchen Stellen andere Schlüsse ziehen oder bestehende Vermutungen untermauern. Insgesamt fehlt der Studie eine wissenschaftliche Vorgehensweise mit Hypothesen, semantischem Modell, Variablen etc. - das soll natürlich nicht heissen, dass nicht wissenschaftlich korrekt gearbeitet wurde - es fehlt einfach ein grundlegendes Konzept, was eben weiterführende Untersuchungen und Studien dringend nötig macht um die hier vermuteten Erkenntnisse zu bestätigen. Inhaltliche Kritik gilt vor allem den teilweise unklaren, an einigen Stellen nicht mathematisch exakt abgegrenzten Begrifflichkeiten sowie der Tabelle. Selbige ist nicht unbedingt selbsterklärend und die von den Autoren gelieferten Definitionen der Begriffe helfen einem nicht dabei zu erkennen welche Zellen jetzt in Summe 100% ergeben sollten - und welche eben nicht. Ausserdem wird vorgeschlagen, Nachrichten in Zukunft auch nach dem Alter filtern zu können, was seit etlichen Generationen des verwendeten Mail-Klienten (Thunderbird) zumindest in der vollwertigen Suchansicht (Ctrl.+Shift+F) problemlos möglich ist.

5 Alternative Darstellungsformen

Um ein Information Retrieval in Mail Datenbanken zu erleichtern bzw. semantische Zusammenhänge zwischen verschiedenen Nachrichten klarer herauszustellen und eine andere Art der Suche zu ermöglichen existieren einige Ansätze alternativer Darstellungsformen für Mail-Anwendungen, welche vom heutigen Baumstruktur Standard mit Ordnern und Listen abweichen und meist auf Ontologien arbeiten. Komplementär dazu bestehen Systeme, welche quasi in die "andere Richtung" arbeiten und auf der Basis semantischer Kriterien u.a. Mailversendung an ad-hoc generierte Gruppen ermöglichen oder automatisiert diverse Prozesse anstossen, Stichwort: *Semantic E-Mail Processing*. Zu letzterem seien vor allem einige Theoriepaper von Luke McDowell et al. der University of Washington[9] genannt und als praktische Anwendung bspw. der noch ausbaufähige Ansatz der Stanford University mit dem SEAMail Konzept[7]. Während hier hauptsächlich versucht wird, ein Informationsverwaltungs-System zu erstellen welches "intelligent" Nachrichten an die richtigen Rezipienten zu schicken, mehrere Kommunikationskanäle gleichzeitig zu bedienen (va. Soziale Web-Services) und definierte Workflows zu nutzen um unter Nutzung des Mediums E-Mail Informationen nachzufragen oder Aufträge zu erteilen, wird ein klassisches Data-Mining in vorhanden Nachrichtensätzen hier nicht extra unterstützt. Die zuerst genannten Systemtypen, welche auf Basis von Gruppenzugehörigkeiten und Ontologien

arbeiten um Nachrichten auf andere Art und Weise zu speichern und zu verwalten bieten hierzu mehr Optionen an. So auch der *Conceptual E-Mail Manager, "CEM"* von Cole und Stumme[1], welcher als richtungsweisendes Konzept gesehen werden kann, um hier zunächst mal ein Beispiel zu nennen. Im CEM sind die Nachrichten anhand des formalen Kontextes von Stichworten in multidimensionalen Gitterstrukturen, den sog. *Concept Lattices* organisiert und graphisch dargestellt. Dies erlaubt eine Navigation durch die Sammlung über alle unterschiedlichen Pfade. Konkret bedeutet dies, dass beim speichern von Nachrichten gewisse Schlagworte angegeben werden und beim späteren Abrufen jeder dieser referenzierten Pfade verfolgt werden kann um die Mail in einer graphischen Oberfläche zu finden (siehe Abb. 4). Dies ist ein Ansatz wie man ihn auch aus dem klassischen Dokumentenmanagement kennt und welcher hier Anwendung in E-Mail Systemen findet.
Auch einen wichtigen Forschungsschwerpunkt in diesem Bereich stellt die automatische Erstellung solcher Ontologien für grössere Mail Datenbanken mit Hilfe lexikalischer Datenbanken wie George Millers WordNet[6] und eine Durchsuchung nach gewissen Themen dar. Aus diesem Bereich entstammt auch eine der wenigen Anwendungen aus dem praktischen Bereich [11], welche Forensikern eine Analyse grosser Mailbestände ermöglichen soll. Anders als beim klassischen Information Retrieval wie es auch bei Thunderbird eingesetzt wird, d.h. manuelle Filter und Suchbegriffe welche als boolescher Ausdrücke verknüpft werden und anhand von Mengenzugehörigkeiten dann ein Ergebnis-Set liefern, geschieht in diesem Falle die Erstellung der Suchanfrage automatisch via *Query-Expansion* auf linguistischer Basis. Zusätzlich wird hier ein Ranking-Algorithmus verwendet um die Relevanz der Suchergebnisse zu erkennen. Mit Hilfe dieses Systems ist es möglich mit sehr unspezifischen Anfragen gute Ergebnisse (va. in Relation zu klassischen, händischen Booleschen Anfragen) zu erzielen, also eine teilweise deutliche Verbesserung bei den beiden massgeblichen Kennzahlen für Recommender Systeme, *Precision* und *Recall* zu erreichen. Inwiefern die hier vorgestellten Lösungen und Ansätze einem "generischen Nutzer" beim alltäglichen Umgang mit dem Medium E-Mail helfen, ist nicht final geklärt. Damit beschäftigt sich das nächste Kapitel.

[6] http://wordnet.princeton.edu/

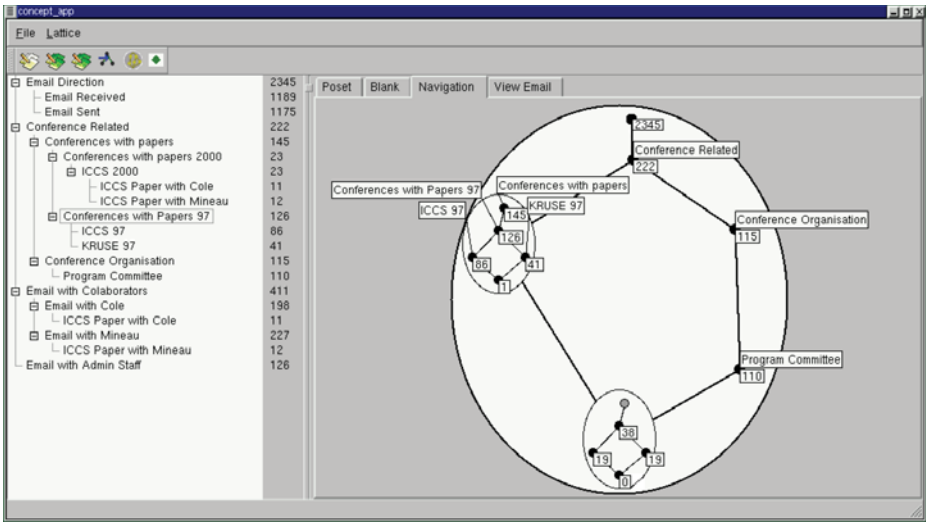


Abbildung 4. Screenshot des Conceptual E-Mail Managers (Stand: Jahr 2000). Linker Hand Menüdarstellung der Kontexte, rechter Hand graphische Interpretation.

Abbildung 5. Illustration der Nutzeroberfläche des forensisches Searchtools von Son et al. 2008. Erkennbar sind neben dem Anfrage Panel links auch die Ausgabe des Scoring Algorithmus (rechts oben) sowie eine Vorschau der Mails (rechts unten).

6 Fazit und Designvorschlag

Es wurden neben einer Studie über das Information Retrieval und Anfrageverhalten von Nutzern in E-Mail Systemen auch einige Konzepte für das IR Retrieval und die allg. Unterstützung bei der Nutzung von Mails zu Kommunikationszwecken vorgestellt. Diese sind jedoch ausserhalb von Fachkreisen und Konferenzen praktisch unbekannt bzw. haben zumindest noch keinen grossflächigen Einzug in Haushalte von Privatpersonen oder die Büros durchschnittlicher Angestellter gefunden. Die Ursachen dafür kann man in verschiedenen Richtungen suchen. Zum Einen sind die meisten der letztlich entstandenen Prototypen grafisch nicht unbedingt ansprechend, was in Zeiten des Web 2.0 und der "Apps" einfach nicht mehr konkurrenzfähig erscheint. Während der CEM von Cole und Stumme damals verständlicherweise noch grafisch besonders poliert erschien, haben hier erst die Nachfolgeprojekte wie *HierMail*[4] oder *Mail-Sleuth*[7] Abhilfe geschaffen (siehe Abb.6). Die Entwicklung dieser Projekte wurde dennoch eingestellt. Der Mehrwert, welcher durch die Nutzung solcher als Stand-Alone Anwendungen oder Plugins erhältlichen Programme entsteht, scheint nicht ausreichend zu sein um über ihre mangelnde visuelle Raffinesse hinwegzusehen bzw. sie werden schlicht nicht benötigt. Wenn man sich allein die Daten der vorliegenden Studie von Harvey und Elsweiler nochmals ansieht stellt man fest, dass alleine schon ohne die Nutzung von Filteroptionen (Anm.: es ist nicht ganz geklärt ob in der Studie Filter benutzt wurden, diese Information fehlt neben vielen weiteren im Artikel) Anfragen mit einem doch sehr kurzen Suchterm von im Schnitt unter sechs Zeichen und meist lediglich einem Wort in nur wenigen Klicks (øLänge der Query-Chains 5.17 Klicks) zum Ziel führen. Das unterstützt, in Verbindung mit dem schwachen Interesse an ontologischen Mail-Klienten, die bereits genannte Vermutung das solche Systeme wohl ein wenig über das Ziel hinausschiessen und für die breite Masse an Anwendern nicht interessant sind, da die typischen Anwendungsszenarien der Suchfunktion mit Filtern und vielleicht zwei Wörtern als Suchanfrage schon gut abgedeckt sind.
Vielleicht wäre eine Orientierung an praktikablen Lösungen, welche Verbesserungen auf Basis der bekannten Baumstruktur bieten, sinnvoller. Eine übersichtlichere, grafisch hochwertige Darstellung von laut Suchanfrage relevanten Mails im Zeitverlauf (bspw. als Strahl) unter Berücksichtigung der Top k Absender, wie es die Ergebnisse des Artikels nahe legen, wäre zumindest eine Alternative, über welche man einmal nachdenken könnte. Auch eine schlichte Repräsentation des Scoring-Wertes der Recommendation Engine in einem solchen andersartig (hier: zeitlich) sortierten Ergebnisfeld über Grösse oder Farbe der Nachrichten wäre denkbar. Damit entfernt man sich von eventuell "fremd" anmutenden Dingen wie Indexierung, Schlagworten und multidimensionalen Darstellungsformen und bleibt in für den Standard Nutzer greifbaren und bekannten Gebieten von Ordnern und Listen - ermöglicht trotzdem eine Beschleunigung oder Verbesserung der Arbeitsabläufe durch eine optimierte Darstellung. Aus diesem Grund wurde die nachfolgende Grafik entworfen. Sie zeigt eine mögliche Ausführung des

[7] http://www.mailsleuth.net/

beschriebenen Repräsentationsverfahren. Da dies nicht die primäre Zielsetzung dieser Arbeit war und diese Oberfläche lediglich als Anreiz und zur Illustration dienen soll, wird gebeten die mangelnde grafische Qualität zu entschuldigen. Diese Ansicht wird lediglich aufgerufen, sobald die Suchfunktion (alternativ auch bei Schnellsuche) betätigt wird. Sonst bleibt die normale Darstellungsform eines gängigen Mailprogrammes wie bspw. Microsoft Outlook[8] oder Mozilla Thunderbird erhalten, was eine Neuentwicklung der Standardfunktionalitäten unnötig macht und die Realisierung bspw. als Plugin oder Erweiterung attraktiver erscheinen lässt. Die visuell angedeuteten Funktionalitäten sollen hier kurz ausgeführt werden. Das Suchfeld ermöglicht eine Formulierung der Suchanfrage nach klassischem booleschen Muster, automatisierte Query Expansion und einige Funktionen wie Filter (Nachricht nicht älter als, usw.). Die Hauptausgabe besteht in einer Liste der relevantesten E-Mails, einsortiert in die Top k Sender des Nutzers. Hier erscheinen häufige Sender weiter oben in der Liste. Jede senderspezifische Unterbox kann entlang des Zeitstrahles durchsucht werden, wobei hier die Nachrichten nach ihrer Relevanz nicht räumlich sortiert sind sondern lediglich entsprechend eingefärbt (rot=sehr relevant, orange=relevant, grün=eventuell relevant). Die Orientierung wird durch eine komprimierende, vereinfachende Meta-Darstellung oberhalb des Scrollbalkens vereinfacht. Für jede Mail stehen hier auch Kurzübersichten bereit, welche sich durch einen Klick wie üblich im Vorschau Fenster rechter Hand unten öffnen lassen. Ausserdem steht eine Zusammenfassung bereit, welche einen ersten Überblick über die Resultate der Suche liefert.

[8] http://office.microsoft.com/en-us/outlook/

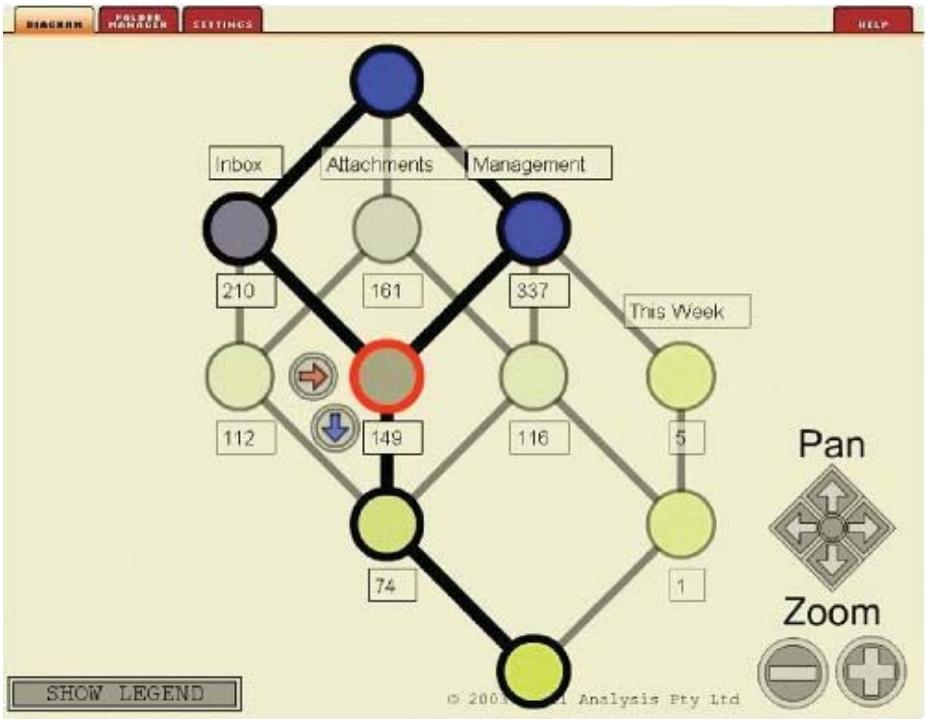


Abbildung 6. Eine neuere Version des CEM, unter dem Namen HierMail. Aus einer Darstellung in Eklund 2004[3].

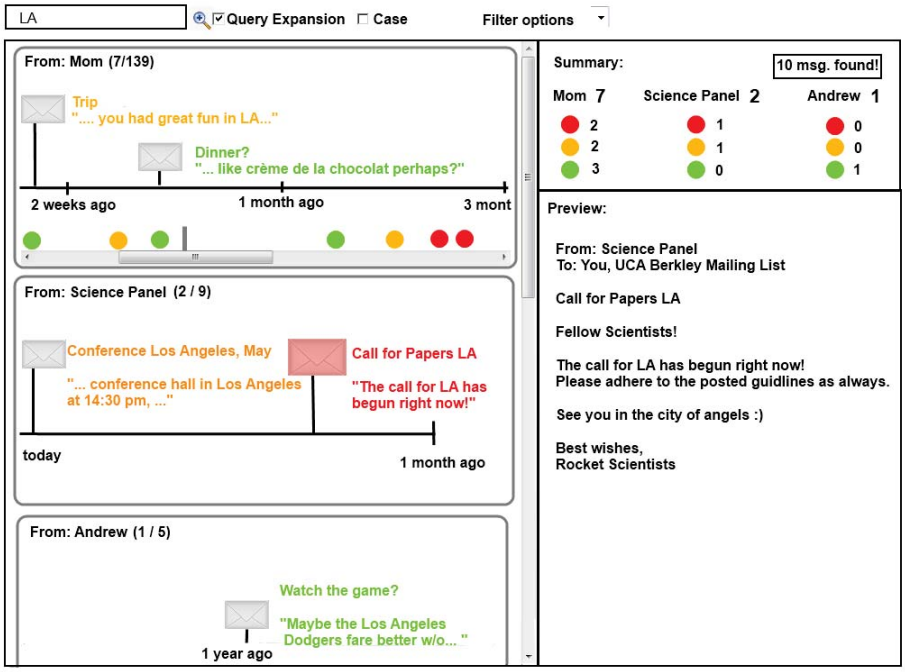


Abbildung 7. Grobentwurf für eine mögliche Darstellung eines Systems zur Mail-Suche, welches die Ergebnisse von Harvey und Elsweiler berücksichtigt.

7 Ausblick und zukünftige Arbeiten

Die künftige Forschung in diesem Bereich sollte zunächst E-Mail Suche als einzelnen, klar differenzierten Problemfall wahrnehmen, was anscheinend noch nicht überall geschieht. Konkret fehlt hier besonders eine Abgrenzung zur Desktop Suche, welche u.A. in dem Paper von Dumais et al. von 2003[2] bereits untersucht und konzeptionalisiert wurde. Die Mailsuche wird dementsprechend oft in gesamtheitlichen Studien oder Artikeln angesprochen, nicht jedoch spezifisch untersucht oder ihre Besonderheiten herausgestellt. So sind die Erkenntnisse von Harvey und Elsweiler, auch wenn sie noch in einer weiteren, grösseren Untersuchung validiert werden müssen, ein Indiz dafür, dass gerade der Personenbezug und der Einfluss des zeitlichen Verlaufs für die Suchanfragen einen massgeblichen Unterschied zur Web Suche darstellen. Es liegt ausserdem der Schluss nahe, dass Systeme für die Desktop Suche einen sehr viel geringeren personellen Einfluss, sowie einen unterschiedlichen Bezug zu zeitlichen Verläufen besteht. All dies führt zu unterschiedlichen, einzigartigen Anforderungen für diese drei Systeme. Während hier die web-basierte Suche durchaus umfassend untersucht wurde, müssen für die beiden verbleibenden Anwendungsfälle noch Schlüsselmerkmale herausgearbeitet und entsprechende Design-Implikationen gezogen werden. Zu-

künftige Werkzeuge für die Unterstützung der Suche in Maildatenbanken sollten sich eventuell am oben genannten Ansatz (siehe auch Abb.7) orientieren um den durchschnittlichen Anwendern dieser "Bürosoftware" einen leichten Einstieg sowie einen klar erkennbaren Vorteil zu bieten und nicht nur sehr spezifische Anwendungsfälle für (IT-)Experten abzudecken. Inwiefern sich die klassische E-Mail als solches d.h. als eigenes Medium behaupten kann bleibt abzusehen. Bisher ist sie trotz Instant-Messengern, sozialen Webportalen, günstigem oder kostenlosem Web-Hosting/Cloud Services und SMS gerade im geschäftlichen Alltag noch unabdingbar. Sie wird hier jedoch oftmals zweckentfremdet um bspw. grössere Datenmengen zu verschicken und wird von Konkurrenzprodukten wie dem "E-POSTBRIEF"[9] ins Visier genommen. Wie lange es also dauert, bis die E-Mail neue Wege bestreiten muss oder einfach in einheitliche Suiten verpackt wird, in welchen alle Kommunikationskanäle gebündelt verwaltet werden, lässt sich nicht absehen. Es ist aber durchaus möglich, dass etwas in dieser Richtung geschehen kann, was dementsprechend wiederum neue Anforderungen für die Suchmechanismen zur Folge hätte.

Literatur

1. Richard Cole and Gerd Stumme. Cem – a conceptual email manager. In Bernhard Ganter and Guy Mineau, editors, *Conceptual Structures: Logical, Linguistic, and Computational Issues*, volume 1867 of *Lecture Notes in Computer Science*, pages 438–452. Springer Berlin / Heidelberg, 2000.
2. Susan Dumais, Edward Cutrell, JJ Cadiz, Gavin Jancke, Raman Sarin, and Daniel C. Robbins. Stuff i've seen: a system for personal information retrieval and re-use. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 72–79, New York, NY, USA, 2003. ACM.
3. P. Eklund. *Concept Lattices: Second International Conference on Formal Concept Analysis, ICFCA 2004, Sydney, Australia, February 23-26, 2004, Proceedings*. Lecture Notes in Computer Science. Springer, 2004.
4. Peter W. Eklund and Richard Cole. Structured ontology and information retrieval for email search and discovery. In *Proceedings of the 13th International Symposium on Foundations of Intelligent Systems*, ISMIS '02, pages 75–84, London, UK, UK, 2002. Springer-Verlag.
5. David Elsweiler, Morgan Harvey, and Martin Hacker. Understanding re-finding behavior in naturalistic email interaction logs. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 35–44, New York, NY, USA, 2011. ACM.
6. Morgan Harvey and David Elsweiler. Exploring query patterns in email search. In *Proceedings of the 34th European conference on Advances in Information Retrieval*, ECIR'12, pages 25–36, Berlin, Heidelberg, 2012. Springer-Verlag.
7. Michael Kassoff, Charles Petrie, Lee-Ming Zen, and Michael Genesereth. Semantic email addressing: The semantic web killer app? *IEEE Internet Computing*, 13(1):48–55, January 2009.

[9] http://www.epost.de/privatkunden.html

8. Vladimir Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory*, 10(8):707–710, 1966. Original in *Doklady Akademii Nauk SSSR* 163(4): 845–848 (1965).
9. Luke McDowell, Oren Etzioni, and Alon Halevy. Semantic email: theory and applications. *Web Semant.*, 2(2):153–183, December 2004.
10. Mark Sanderson and Susan Dumais. Examining repetition in user search behavior. In *Proceedings of the 29th European conference on IR research*, ECIR'07, pages 597–604, Berlin, Heidelberg, 2007. Springer-Verlag.
11. Phan Thien Son, Lan Du, Huidong Jin, Olivier Vel, Nianjun Liu, and Terry Caelli. A simple wordnet-ontology based email retrieval system for digital forensics. In *Proceedings of the IEEE ISI 2008 PAISI, PACCF, and SOCO international workshops on Intelligence and Security Informatics*, PAISI, PACCF and SOCO '08, pages 217–228, Berlin, Heidelberg, 2008. Springer-Verlag.
12. Jaime Teevan, Eytan Adar, Rosie Jones, and Michael A. S. Potts. Information re-retrieval: repeat queries in yahoo's logs. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 151–158, New York, NY, USA, 2007. ACM.
13. Sarah K. Tyler and Jaime Teevan. Large scale query log analysis of re-finding. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 191–200, New York, NY, USA, 2010. ACM.

Nutzungsrechte in Enterprise Search Systemen - Secure Enterprise Search

Michael Großmann

Otto-Friedrich-Universität Bamberg, Lehrstuhl für Medieninformatik
michael-alfons.grossmann@stud.uni-bamberg.de

Zusammenfassung Ähnlich den Websuchmaschinen, wird auch bei der unternehmensinternen Suche - dem sogenannten Enterprise Search - ein zentraler Zugang zu allen Informationen des Unternehmens gewünscht. Neben unternehmenseigenen Datenbanken, internen Webseiten, sollen auch Emails, Word-, Excel-, Powerpoint- oder PDF-Dateien durchsucht werden können, um so den Mitarbeitern einen zentralen Zugang zum Wissen des Unternehmens gewährleisten zu können. Da jedoch nicht alle Informationen des Unternehmens für jeden Mitarbeiter gleich zugreifbar sein sollen, muss die Enterprise Search Lösung entsprechend abgesichert werden. Die entscheidende Frage dabei ist, wie sichergestellt werden kann, dass nur autorisierte Nutzer die sicherheitskritischen Informationen finden können. Diese Arbeit befasst sich mit solchen Secure Enterprise Search Systemen, mit dem Ziel dem Leser einen ersten Einblick in die Thematik zu bieten, indem unter anderem gängige Architekturen, Definitionen sowie typische Problemfelder aufgezeigt werden.

Schlagworte: Information Retrieval, Secure Enterprise Search, Enterprise Full-text Search, Sichere Suche, Nutzungsrechte in Unternehmenssuche

1 Einleitung und Problemstellung

Die Nutzung von Suchmaschinen, hat sich heutzutage fest im Alltag vieler Menschen verankert. Als Suchinstrumente schaffen sie einen zentralen Zugang zu den Informationen und Dokumenten im Internet und ermöglichen so eine einfache aber gezielte Suche auf den zahllosen Webservern weltweit. Eine derart zentrale Suchlösung wird häufig auch im Kontext einer unternehmensinternen Suchmaschine, dem sogenannten Enterprise Search, gewünscht. So sollen beispielsweise neben unternehmenseigenen Datenbanken, internen Websites auch Emails, Gesprächsprotokolle, Word-, Excel-, Powerpoint-, PDF-, Grafik- oder Videodateien durchsucht werden können, um damit den Mitarbeitern einen zentralen Zugang zum Wissen des Unternehmens anbieten zu können, ohne dass diese unterschiedliche Suchlösungen nutzen müssen. Problematisch hierbei ist, dass die Unternehmensdaten häufig in verschiedenen manchmal nicht miteinander kompatiblen IT-Systemen verteilt sind, oder als unstrukturierte Daten vorliegen, was eine integrierte Suche bzw. spätere Abfrage erschwert. Zudem können

ohne entsprechende Vorkehrungen eventuell vertrauliche oder personenbezogene Daten in die Abfrageergebnisse einbezogen werden, was aus Gründen der Vertraulichkeit bzw. des Datenschutzes mitunter problematisch ist. Da immer mehr Unternehmen immer mehr Daten speichern und indexieren bzw. durchsuchbar machen, erhöht sich die Gefahr, dass sicherheitskritische Informationen an die Öffentlichkeit gelangen. Jonny Long zeigt in seinem Buch "Google Hacking"[1] eindrucksvoll, wie vertrauliche Informationen eines Unternehmens relativ leicht über die normale Googlesuche abgerufen werden können, wenn die Dokumente nicht entsprechend gesichert, bzw. die Nutzungsrechte adäquat festgelegt wurden. Die entsprechenden Suchstrings auf interessante Unternehmensdateien können in seiner Google Hacking Database[1] abgerufen werden. Das Ausmaß solcher Datenpannen rückte in den letzten Monaten vermehrt in den Fokus der Öffentlichkeit, nachdem mehrere Einbrüche in die Computersysteme von Unternehmen bekannt wurden. Um derlei Datenpannen zu vermeiden muss eine sichere unternehmensinterne Suchmaschine die Zugriffs- und Nutzungsrechte des anfragenden Benutzers berücksichtigen, bevor es ihm die Suchergebnisse endgültig präsentieren kann. So gilt es die integrierte Suche, in welcher alle Datenquellen miteinander verbunden sind, mit den vielleicht in Widerspruch stehenden Sicherheitsanforderungen, nur bestimmten Nutzern einen präzisen und wohldefinierten Zugang zu gewährleisten, zu harmonisieren. Letztlich subsumiert sich alles in der Frage, wie sichergestellt werden kann, dass nur autorisierte Nutzer sicherheitskritische Informationen finden können.

Die vorliegende Arbeit befasst sich mit sicheren Enterprise Search Systemen, im folgenden als Secure Enterprise Search Systemen oder kurz SES bezeichnet und verfolgt das Ziel, dem Leser einen ersten Einblick in die Thematik zu gewähren, sowie auf Probleme bei der Implementierung solcher Lösungen hinzuweisen. Zunächst wird in *Kapitel 2* definiert, was unter Enterprise Search und Secure Enterprise Search (SES) zu verstehen ist. Weiterhin wird in *Kapitel 3* auf die wichtigsten Bestandteile von Secure Enterprise Search eingegangen. Die eingangs gestellte Frage, wie sichergestellt werden kann, dass nur autorisierte Nutzer sicherheitskritische Informationen finden können, dient hierbei als roter Faden und es gilt zu klären, was unter autorisierten Nutzern bzw. sicherheitskritische Informationen zu verstehen ist, bzw. bis zu welcher Granularität die Suche unterstützt werden soll. So sollen verschiedene Rollenmodelle für die Nutzer, sowie Ebenen für die Granularität der Suche vorgestellt werden. Der Schwerpunkt der Arbeit bildet schließlich *Kapitel 4*, in welchem zwei typische Architekturmuster - das Early und das Late-Binding - erklärt werden, sowie *Kapitel 5*, in welchem eine mögliche Implementierung von Early-Binding über virtuelle Tokens (virtual Tokens) beschrieben wird. Abschließend wird in *Kapitel 6* ein kurzes Fazit gezogen.

[1] siehe http://www.hackersforcharity.org/ghdb/

2 Definition (Secure) Enterprise Search

Enterprise Search Ausgehend von den heutigen Internet-Suchmaschinen, wie beispielsweise Google, gibt es schon länger die Bestrebung einen ebenso zentralen Informationszugang auch für die Inhalte in Unternehmen zu ermöglichen. Die Vision ist hierbei einen über alle Informationsquellen des Unternehmens integrierten Informationszugang zu gewährleisten. Das Forschungsfeld des Information Retrieval, dass sich mit integrierten Suchansätzen im Unternehmen beschäftigt wird als Enterprise Search bezeichnet. David Hawking [2, Seite 641] definiert Enterprise Search dabei ganz allgemein, als die Anwendung von Information Retrieval Technologien zur Suche von Informationen im Unternehmen. Sie kann als die Suche über externe Webseiten, Intranetseiten und anderen elektronischen Quellen wie Emails oder Datenbankeinträgen interpretiert werden.

Ziel einer internen Suchmaschine ist es laut Oliver Schonschek [3], strukturierte und unstrukturierte Daten, welche in den unterschiedlichsten Formaten vorliegen und an verschiedensten Speicherorten abgelegt wurden, mit einer zentralen Suchfunktion abfragen zu können. Neben strukturierten Datenbanken sollen insbesondere auch unstrukturierte bzw. teilweise strukturierte Daten wie Emails, Chatprotokolle, Word-, Excel-, PowerPoint- oder PDF-Dateien sowie MP3-, Audio-, Grafik- oder Videodateien einbezogen werden. Die Integration dieser unstrukturierten Datenquellen stellt häufig eine große Herausforderung dar. Laut einer Studie sollen immerhin rund 70 Prozent aller Daten im Unternehmen in unstrukturierter Form vorliegen. [3]

Als eigens formulierte Definition lässt sich zusammenfassend festhalten, dass man unter Enterprise Search die unternehmensinterne Suche nach Inhalten bzw. Daten in den unterschiedlichen Systemen des Unternehmens verstehen kann, mit dem Ziel die Mitarbeiter dabei bestmöglich bei der Suche nach für sie arbeitsrelevanten Informationen zu unterstützen.

Secure Enterprise Search Wie eingangs erläutert kommt es häufig vor, dass auch unternehmenskritische oder personenbezogene Daten in die Suche einbezogen werden sollen, wobei es hierbei gilt die Vertraulichkeit der Daten zu gewährleisten. Unter Secure Enterprise Search versteht man deshalb einerseits die Möglichkeit für Unternehmen seinen Mitarbeitern einen umfassenden Informationszugang zu Unternehmensdaten zu gewährleisten, während andererseits gleichzeitig sicherheitskritische Daten vor unauthorisiertem Zugriff geschützt werden [4, S.3]. Secure Enterprise Search bietet in erster Linie eine Art Zugriffskontrollmechanismus, welcher sicherstellt, dass Mitarbeiter, die eine Abfrage durchführen, nur die Ergebnisse angezeigt bekommen, welche sie auch berechtigt sind zu sehen [5, S. 2]. Wichtige Bestandteile einer Secure Enterprise Search Lösung sind dabei ein Rollenmodell für die Nutzer der Suchmaschine, welches Zugriffe regelt, sowie ein Modell für die Granularität der Suche allgemein. Rollenmodell der Nutzer sowie Granularität der Suche soll im nächsten Kapitel ausführlicher behandelt werden. Da Secure Enterprise Search eine Form von Nutzungsrechten impliziert, sollen im Weiteren Verlauf des Papers die Be-

grifflichkeiten *Nutzungsrecht in Enterprise Search Systemen* sowie *Secure Enterprise Search* synonym verwendet werden, wie es bereits auch im Titel der Arbeit angedeutet wurde.

3 Zentrale Bestandteile einer Secure Enterprise Search Lösung

Bevor es in *Kapitel 4* um mögliche Architekturen von SES-Lösungen gehen soll, werden in diesem Kapitel die zentralen Bestandteile einer SES-Lösung aufgezeigt. Wie bereits angedeutet gehören hierzu ein Rollenmodell der Nutzer, sowie die Festlegung der Granularität der Suche. Bevor diese jedoch sinnvoll festgelegt werden können gilt es eine Sicherheitsbedarfsanalyse des Unternehmens durchzuführen, in welcher es darum geht sicherheitskritische und damit schützenswerte Inhalte zu identifizieren und mögliche Risikobereiche frühzeitig aufzudecken. Hierzu sollte eine Analyse der Unternehmensinfrastruktur mit den verschiedenen Informationsquellen, sowie vorhandener Altsysteme (Legacy Systems) durchgeführt werden. Welche Inhalte sind besonders schützenswert und wer soll Zugang zu den Daten bekommen? Wie sieht die derzeitige Unternehmensinfrastruktur aus? Ist es möglich, und wenn ja wie, die verschiedenen Informationsquellen sicher zu integrieren, dass ein zentraler Zugang ermöglicht werden kann? Diese und weitere Fragen gilt es in einer Sicherheitsbedarfsanalyse zu klären. Dabei kann sich der Sicherheitsbedarf von Unternehmen zu Unternehmen stark unterscheiden. So hat eine Klinik vermutlich andere Sicherheitsanforderungen als ein Produktionsunternehmen. Die Erkenntnisse der Sicherheitsbedarfsanalyse bilden schließlich die Basis für die Festlegung des Rollenmodells der Nutzer sowie der Granularität der Suche.
Dass die Sicherheit einer Suchmaschine besonders wichtig und notwendig ist zeigt eine Studie von Price Waterhouse Cooper, in welcher es heißt, dass Enthüllung von vertraulichen Informationen die Fortune 1000 Unternehmen jährlich im Durchschnitt 15 Millionen Dollar pro Unternehmen kosten [4, S. 4]. Um zu verhindern dass bspw. Finanzdaten, Kundendaten, Mitarbeiterdaten oder dergleichen durch die Suchmaschine an die Öffentlichkeit gelangen, muss die Suchmaschine die Zugriffsrechte der Dokumente mit dem Berechtigungsnachweis der Nutzer abgleichen.

3.1 Rollenmodell mit Zugriffsrechten für Nutzer

Rollenmodelle, welche Zugriffsrechten auf bestimmte Dokumente für einzelne Nutzer festlegen, bilden die eine Seite der Sicherheitsgleichung. Aufgrund der Erfahrungen und der Ähnlichkeit zu anderen Systemen werden Rollenmodelle heutzutage häufig gut verstanden. Genannt seien hier z.B. Rollenmodelle innerhalb von Content Management Systemen, Datenbanken oder innerhalb des Linux Dateisystems. Die Herausforderung in Bezug auf Secure Enterprise Search liegt in den Implementierungsdetails, wie sich die Nutzerrechte bspw. innerhalb der Suche umsetzen lassen. Generell lassen sich Nutzer anhand folgender Kategorien klassifizieren [6, S. 5]:

- Globaler Status
- Gruppen / Rollen
- Spezifischer Nutzer

Die Kategorien sind dabei nach zunehmender Restriktion des Zugriffs absteigend geordnet, können sich jedoch je nach Festlegung mitunter auch überlappen, bspw. wenn eine Gruppe lediglich aus einem einzelnen Nutzer besteht.

Globaler Status Innerhalb dieser Klassifikation bekommen alle Nutzer, die sich erfolgreich am System verifizieren können, die selben Zugriffsrechte zugesprochen. Das heißt, dass ein Nutzer mit globalem Status „verifiziert“, vollen Zugriff auf *alle* Informationen des Unternehmens erhält. Schlägt die Verifizierung hingegen fehl, erhält er keinen Zugriff. Eine Abstufung zwischen einzelnen Mitarbeitern ist nicht möglich. Dieses „Alles oder Nichts“-Prinzip erscheint in der heutigen Zeit häufig als nicht mehr zweckmäßig und eine feinere Abstufung wird gewünscht. Dies kann bspw. durch Verfeinerung der Zugriffsrechte durch Gruppen/Rollen erreicht werden. [6, S. 5]

Gruppen / Rollen In dieser Klassifikation können beliebig viele und unterschiedliche Gruppen bzw. Rollen definiert und mit entsprechenden Zugriffsrechten ausgestattet werden. So ist es möglich manche Gruppen bspw. nach Rang oder Titel der Mitarbeiter zu formen (z.B. CEO, Ländermanager, Regionalmanager), während andere Gruppen nach Querschnittsfunktionen wie Personal, Buchhaltung, Finanzen usw. gebildet werden können. Die einzelnen Mitarbeiter können schließlich den verschiedenen Gruppen zugeordnet werden, was eine sehr flexible Rechtevergabe ermöglicht. Gruppen können sich dabei auch überlappen bzw. können im extremsten Falle auch nur aus einzelnen Personen bestehen. [6, S. 5]

Spezifischer Nutzer Als feinste Abstufung der Zugriffsrechte kann die Sicherheit auch auf Nutzerbasis in Form spezifischer Nutzer definiert werden. Hierzu muss für jeden Nutzer individuell festgelegt werden, auf welche Dokumente er Zugriff hat und auf welche nicht. Diese Klassifikation stellt praktisch das Gegenteil zum globalen Status dar. Wenn auch aus Sicherheitsgesichtspunkten die Rechtevergabe auf Nutzerbasis das Optimum darstellt, so ist sie häufig aufgrund des initialen und fortlaufenden Administrationsaufwands nicht praktikabel, sowie aufgrund der Komplexität der Abhängigkeiten häufig schwer zu implementieren. Weiterhin kann eine Suchmaschine durch die vielen notwendig werdenden Filterungen schnell überfordert werden. [6, S. 5]

Weil Begriffe wie ACL, LDAP und Active Directory häufig im Zusammenhang mit Zugriffsrechten von Nutzern auftauchen, sollen diese kurz erläutert werden. Eine Access Control List (ACL), zu deutsch Zugriffssteuerungsliste, legt fest, welcher Nutzer, welche Dienste und Dateien nutzen darf. ACLs sind dabei feiner einstellbar als bspw. die bekannten Zugriffsrechte des Linux Dateisystems, welche

die Rechtevergabe lediglich für einen Benutzer, eine Gruppe und den „Rest der Welt“ zulassen.[2] Das Leightweigt Directory Access Protocol (LDAP) und Active Directory sind vereinfacht gesprochen Standardprotokolle, um Informationen über Nutzer, Gruppen von Nutzern und anderen Unternehmensressourcen über Verzeichnisdiensten leicht abfragen zu können [6, S. 6].

3.2 Granularität der Suche

Während die Festlegung des Rollenmodells der Nutzer die eine Seite der Sicherheitsgleichung darstellt, stellt die Granularität, mit welcher Genauigkeit die Suche im einzelnen ermöglicht werden soll, die andere Seite dar. Sie kann auf unterschiedlichen Ebenen (Level) implementiert und gemanagt werden. Es ist notwendig die Granularität für die jeweils gewählte Implementierung gezielt festzulegen [5, S. 2]. Folgende Ebenen sind für die Granularität der Suche bekannt und sollen im weiteren kurz erläutert werden [6, S. 2f][5, S. 2]:

- Alles oder Nichts-Ebene (All or Nothing)
- Kollektions- bzw. Repository-Ebene
- Dokument-Ebene
- Feld-Ebene
- Sub-Feld-Ebene

Abbildung 1 veranschaulicht die unterschiedlichen Granularitätsebenen, sortiert nach zunehmendem Detaillierungsgrad der Suche.

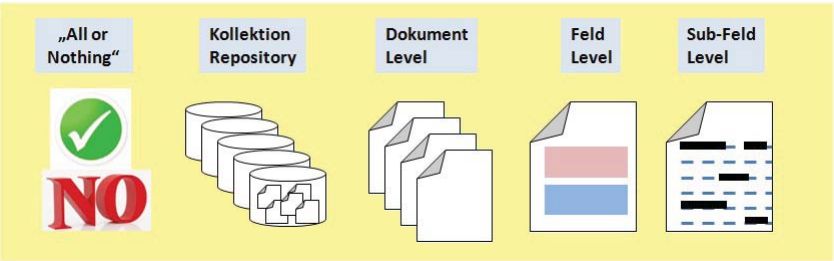


Abbildung 1. Verschiedenen Granularitätsebenen nach zunehmendem Detaillierungsgrad der Suche. Quelle: In Anlehnung an [6, S. 2ff]

Alles oder Nichts Ebene (All or Nothing) Analog dem globalen Status, bei Festlegung der Nutzerrechte, stellt die Alles oder Nichts Ebene das Gegenstück bei der Granularität der Suche dar. Die dahinter liegende Idee ist, dass wenn ein Mitarbeiter eingeloggt oder anderweitig verifiziert ist, er *alle* indexierten Datenquellen des Unternehmens sehen und durchsuchen kann. Wenn er nicht eingeloggt ist wird ihm der Zugriff verwehrt. [6, S. 2]

[2] `http://de.wikipedia.org/wiki/Access_Control_List`

Kollektions- bzw. Repository-Ebene Eine einfache und dennoch wirkungsvolle Technik, um die Sicherheit der Informationen zu gewährleisten, ist die Dokumente nach ihren Sicherheitsanforderungen in größere Kollektionen (manchmal auch als Repositories, Dokumentenindizes oder Dokumentensets bezeichnet) einzuteilen [6, S. 2]. So können z.B. verschiedene Kollektionen für Öffentlichkeit, Intranet, Partner, spezifische Abteilungen oder ähnlichem strukturiert werden, welche im Anschluss sehr einfach auf entsprechende Gruppen gemappt werden können [5, S. 2]. Die Suchmaschinen können schließlich recht leicht den Zugriff auf die einzelnen Kollektionen gewähren, sobald sich der Nutzer entsprechend angemeldet hat.

Dokument-Ebene Kommt es vor, dass der Zugriff innerhalb einer Kollektion weiter eingeschränkt werden soll, weil verschiedene Dokumente unterschiedlichen Mitarbeitern zugänglich gemacht werden sollen, kann der Zugriff auch auf Dokumentenebene spezifiziert werden. Dabei wird der Zugriff auf Ebene der einzelnen Dokumente mit entsprechend angefügten Annotationen geregelt. Diese Ebene ähnelt recht stark Datenbanksystemen, in welchen verschiedene Gruppen verschiedene Daten sehen können. Die Granularität der Suche auf Dokumentebene zu managen, ist heute die bevorzugte Methode vieler Secure Enterprise Search Lösungen [5, S. 2].

Feld-Ebene Reichen Zugriffsrechte auf Dokumentebene nicht aus, können diese auch auf Feld-Ebene ausgeweitet werden. Die Idee der Feld Ebene ist es, dass unterschiedliche Nutzer unterschiedliche Teile des selben Dokuments sehen können. So wäre bspw. eine weitere Abstufung zwischen Länder- und Regionalmanager möglich, in welcher die Ländermanager alle Verkaufszahlen des Landes einsehen können, wohingegen die Regionalmanager lediglich die für Sie relevanten Daten sehen können. Wie leicht vorzustellen, beginnt die Komplexität der Implementierung auf dieser Ebene schlagartig anzusteigen [6, S. 3].
Die Möglichkeit der Implementierung einer Feld-Ebenen Granularität ist auch von der Art der Strukturierung der zugrundeliegenden Dokumente abhängig. Für XML- und HTML-Dokumente, sowie Datenbankeinträge lassen sich z.B. leichter Zugriffskontrollen auf Felder bzw. Teile des Dokumentes anwenden, wie für völlig unstrukturierte Dokumente. [5, S. 2]

Sub-Feld-Ebene Die höchste Granularitätsstufe liegt auf Sub-Feld-Ebene und entspricht einer weiteren Verfeinerung der Feld-Ebenen Granularität. Manchmal kann es notwendig sein, Informationen auch auf einer Sub-Feld Ebene einzuschränken. Dies beinhaltet bspw. die Entfernung bzw. „Schwärzung“ bestimmter Wörter oder Phrasen eines Dokuments. Sub-Feld-Ebenen Retrieval gilt als am schwierigsten umzusetzen, speziell wenn die zugrundeliegenden Dokumente unstrukturiert sind. [6, S. 4]

Zusammenfassend lässt sich veranschaulichend festhalten, dass man sich die unterschiedlichen Möglichkeiten der Implementierung der Rechtevergabe der Suchmaschine schließlich als eine zweidimensionale Matrix mit dem Rollenmodell auf der einen, und der Granularität der Suche auf der anderen Seite, vorstellen kann. Die einzelnen Zugriffsrechte für die Nutzer, sowie die Sichtbarkeit der Suchergebnisse unterscheiden sich innerhalb der einzelnen Felder dieser Matrix.

4 Konzeptuelle Architekturen von Secure Enterprise Search Lösungen

Obwohl sich viele Implementierungen von Anbieter zu Anbieter stark unterscheiden können, haben sich doch zwei grundlegende Architekturmuster herausgebildet, um den Inhalt der Suchergebnisse zu filtern. Je nachdem zu welchem Zeitpunkt eine Filterung der Ergebnisse durchgeführt wird, unterscheidet man zwischen Frühem (Early-Binding) und Spätem (Late-Binding) filtern. Beide Ansätze sollen im Folgenden erklärt, sowie deren Vor- und Nachteile aufgezeigt werden. Um eine gemeinsame Ausgangsbasis zu schaffen soll jedoch zuvor der allgemeine Aufbau einer Enterprise Search Lösung kurz wiederholt werden. Für eine ausführlichere Darstellung der Feinheiten von Enterprise Search Systemen sei jedoch auf Spezialliteratur verwiesen z.B. [2, S. 641ff].

4.1 Grundsätzlicher Aufbau einer Enterprise Search Lösung

Abbildung 2 zeigt beispielhaft die Architektur einer allgemeinen Enterprise Search Lösung eingeteilt in die drei Phasen Sammeln (Gathering), Extraktion (Extraction) und Indexierung (Indexing), mit dem Ziel einen invertierten Index über die unterschiedlichen Datenquellen des Unternehmens zu erstellen [2, S. 648f].

Innerhalb der Sammelphase geht es darum die unterschiedlichen Datenquellen des Unternehmens wie Dokumente, Emails, usw. zu crawlen. Herausforderungen liegen hierbei ähnlich den Websuchmaschinen bspw. bei der Aktualität bzw. Aktualisierung der Daten oder der Erkennung von Duplikaten oder beinahe Duplikaten.

In der Extraktionsphase geht es darum, Text und Metadaten aus den Datenquellen auszulesen und entsprechend für die Indexierung aufzubereiten und in einer Dokumentenkollektion bzw. Metadatenkollektion zu speichern. Die Probleme bei der Extraktion liegen meist in der Natur der unstrukturierten Dokumenten begründet. So gestaltet es sich oft schwierig die richtigen Informationen zu filtern, was eine Suche später erschwert.

Schließlich wird in der Indexierungsphase durch den Indexer auf Basis der Dokument- und Metadatenkollektion ein invertierter Index gebildet.

Zur Vereinfachung wurde hierbei der Aspekt der Verteilung ausgeklammert und angenommen, dass über alle Dokumentenquellen hinweg ein Dokument- und Metadatenrepository angelegt und daraus lediglich ein invertierter Index generiert wurde. Eine solche monolithische Suchmaschine ist in manchen Organisationen aus technischen Gründen häufig nicht umsetzbar. So kann das Sammeln,

Extrahieren und Indexieren mancher Unternehmensquellen zu langsam sein, die Netzwerklast zu hoch oder die Datenmanegen zu umfangreich. Als Ausweg bietet es sich dann an, eine Art Federated Search (auch bekannt unter Begriffen wie search federation, metasearch, oder distributed information retrieval) zu implementieren, bei welchem eine Suchanfrage an mehrere Suchmaschinen weitergeleitet und die Ergebnisse schließlich aggregiert werden [2, S. 659f]. Da Federated Search folglich als Suche über mehrere monolithische Suchmaschinen verstanden werden kann, soll er im Weiteren nicht genauer betrachtet werden und der Fokus auf einer monolithischen Suchmaschine liegen.

Die Wahl der Architektur (Late- oder Early-Binding) einer Secure Enterprise Search Lösung hat schließlich einen großen Einfluss darauf, wie der invertierte Index letztlich gebildet wird, bzw. welche zusätzlichen Informationen darin gespeichert werden müssen. Nachdem nun die grobe Architektur eines Enterprise Search Systems vorgestellt wurde, stellt sich im weiteren die Frage, wie diese verfeinert werden muss, um eine sichere Suche gewährleisten zu können.

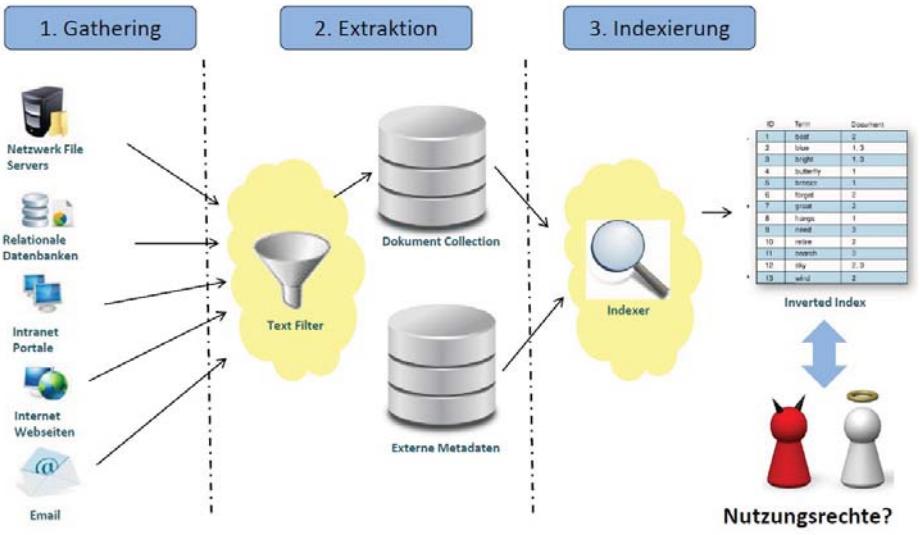


Abbildung 2. Funktionsmodell einer Enterprise Search Lösung. Quelle: [2, S. 649], leicht abgeändert

4.2 Spätes Filtern (Late Binding)

Beim Late-Binding findet die Filterung der Dokumente erst statt, *nachdem* die Anfrage des Nutzers an die Suchmaschine übermittelt wurde. Nach der Übermittlung der Suchanfrage werden alle Dokumente aus der Dokumentenkollektion zurückgeliefert, die mit dem Suchstring übereinstimmen. Bevor die

Ergebnisse dem Nutzer jedoch präsentiert werden, checkt die Suchmaschine, ob der Nutzer überhaupt die entsprechende Berechtigung besitzt, das jeweilige Dokument einzusehen (Annahme von Sicherheit auf Dokumentebene). Im Falle, dass der Nutzer keinen Zugriff auf das Dokument besitzt, wird es aus der Ergebnisliste gestrichen. [5, S. 3]

Abbildung 3 gibt einen Überblick über den Ablauf der Suche beim Late-Binding, wobei zusammenfassend folgende Schritte durchlaufen werden:

1. Erstellung des invertierten Index über die Datenquellen des Unternehmens
2. Nach Absetzung der Suchanfrage des Nutzers werden alle relevanten Dokumente zurückgeliefert, unabhängig der Zugriffsrechte des Nutzers
3. Im Zuge des Post-Processing (Nachverarbeitung) nimmt die Suchmaschine bspw. über LDAP- oder Active Directory Dienste eine Nutzer/Dokument Zugriffsvalidierung vor
4. Wenn der Nutzer keinen Zugriff auf das entsprechende Dokument besitzt, wird es aus der Ergebnisliste gestrichen
5. Die überprüfte Ergebnisliste wird schließlich dem Nutzer präsentiert

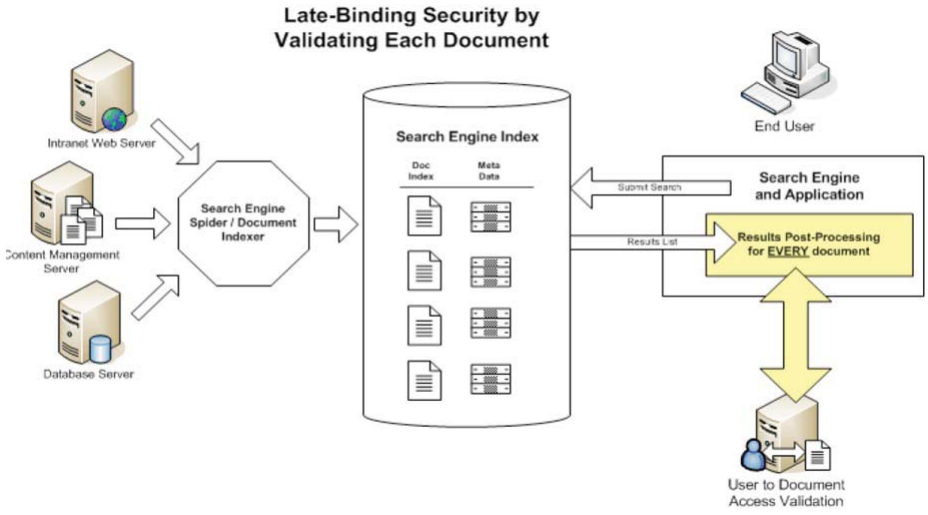


Abbildung 3. Late Binding. Quelle: [6, S. 7]

Eine Late-Binding Filterung kann möglicherweise recht langsam sein und Dienste wie LDAP und Active Directory des Unternehmens oder anderer Sicherheitssysteme stark belasten. Man bedenke dass jedes Dokument zur Anfragezeit mit den entsprechenden Nutzerrechten abgeglichen werden muss. Cachingmechanismen und parallele Verarbeitung werden verwendet um diesen Vorgang so

weit wie möglich zu optimieren, was jedoch manchmal weitere Sicherheitsprobleme aufwirft. Ein weiterer Nachteil des Late-Binding liegt darin, dass es unter Umständen auch recht lange dauern kann bis genügend relevante Dokumente gefunden wurden. Kann bspw. ein Nutzer durchschnittlich lediglich 10 Prozent der Inhalte sehen und sollen 10 Dokumente in der Ergebnisliste erscheinen, müssen im Durchschnitt 100 Dokumente durchsucht werden bis genügend Dokumente angezeigt werden können [6, S. 7].

Late-Binding war dennoch bis vor kurzen die bevorzugte Lösung, da sie tendenziell leichter und schneller zu implementieren war und weniger Konzeptionsarbeit benötigte. Vorhandene Systeme konnten weitgehend belassen werden und lediglich eine Filterung gegen Ende des Prozesses war zu implementieren. Leider wurde bei Einführung einer neuen Enterprise Search Lösung oft so vorgegangen, dass zunächst versucht wurde, eine erste lauffähige Suchmaschine über alle Unternehmensquellen zu implementieren und diese dann hinsichtlich Effektivität, Effizienz sowie Performanz zu optimieren. Sicherheitsaspekte wurden häufig ausgeblendet, bzw. sollten erst im Anschluss hinzugefügt werden [6, S. 1]. Dieses Vorgehen lies sich leichter mit Late-Binding umsetzen. Um das Sucherlebnis der Mitarbeiter weiter zu verbessern wurde nach neuen Wegen gesucht die Performanz zu steigern. Dies gelang durch Early-Binding.

4.3 Frühes Filtern (Early-Binding)

Beim Early-Binding werden die Dokumente bereits gefiltert *bevor* die eigentliche Anfrage an die Suchmaschine übermittelt wird [6, S. 6]. Während des Crawling und Indexierungsprozesses sammelt die Suchmaschine alle Zugriffs- und Sicherheitsrechte über die jeweiligen Dokumente und speichert diese innerhalb des Index mit (dabei unterscheiden sich die Implementierungsdetails von Anbieter zu Anbieter). Eine Möglichkeit die Zugriffsinformationen im Index zu speichern kann über sogenannte Virtual Tokens umgesetzt werden. Dies soll in *Kapitel 5* genauer betrachtet werden. Neben der Speicherung von Zugriffsinformationen im Index werden zudem auch genaue Informationen über die Zugriffsrechte des Nutzers gesammelt, nachdem dieser sich entsprechend am System authentifizieren musste. Wenn der Nutzer schließlich eine Anfrage sendet, werden die Zugriffsinformationen automatisch an die Anfrage angehängt, sodass die Suchmaschine lediglich die Dokumente zurückliefern kann, auf die der Nutzer auch tatsächlich Zugriff hat. Diese Filterung kann man sich ähnlich einer "Where"Bedingung innerhalb einer SQL Anweisung vorstellen. Diese Vorfilterung hat meist eine deutliche Steigerung der Performance zur Folge. [5, S. 3]

Abbildung 4 veranschaulicht den Ablauf des Early-Binding, wobei zusammenfassend folgende Schritte durchlaufen werden:

1. Erstellung des invertierten Index über die Datenquellen des Unternehmens. Im Unterschied zum Late-Binding werden dabei Zugriffsrechte auf einzelne Dokumente bereits mit in den Index aufgenommen. Dies geschieht durch ein Mapping von ACL Informationen auf die einzelnen Dokumente.

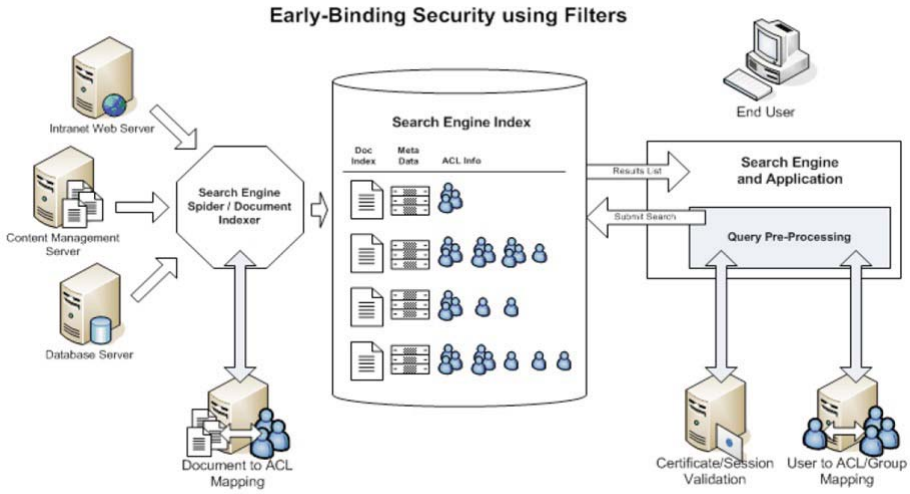


Abbildung 4. Early Binding. Quelle: [6, S. 6]

2. Innerhalb des Pre-Processing (Vorverarbeitung) findet sowohl eine Benutzerauthentifizierung statt, als auch wird der Nutzer auf verschiedene Gruppen gemappt. D.h. die Suchmaschine "kennt"den Nutzer und weiß, auf welche Gruppen der Nutzer Zugriff haben darf
3. Der Nutzer sendet schließlich eine Suchanfrage an die Suchmaschine. An diese Suchanfrage werden die zuvor bestimmten Zugriffsrechte des Nutzers angehängt.
4. Die Suchmaschine liefert schließlich lediglich die für diesen Nutzer relevanten Dokumente als Ergebnisliste zurück und präsentiert diese dem Nutzer

Im Vergleich zum Late-Binding ist Early-Binding häufig schwieriger zu designen und implementieren, da es häufig komplexer ist die einzelnen Zugriffsrechte der verschiedenen Datenquellen, welche sich von Quelle zu Quelle auch stark unterscheiden können oder gar unvereinbar sind, geeignet in den Index aufzunehmen. Zudem können geeignete Schnittstellen fehlen, was ein Mapping der Zugriffsinformationen aus ACL Listen auf die jeweiligen Dokumente je nach (Legacy-)System schwer oder unmöglich macht. Problematisch ist auch die Synchronisation der Zugriffsrechte im Index bzw. eine geeignete Updatepolitik, da ein Update der Zugriffsrechte eine Reindexierung bzw. Deltaindexierung notwendig macht [6, S. 7].
Nicht desto trotz werden Early-Binding Systeme aktuell bevorzugt, da sie schneller sind und dadurch ein angenehmeres Sucherlebnis bieten [5, S. 3]. Die Steigerung der Performanz wird dadurch erreicht, dass ACL Informationen über die Dokumente direkt im Index gespeichert werden, wodurch eine Abfrage wesentlich schneller erfolgen kann. Zudem müssen nicht alle Dokumente durchsucht werden.

Die Ergebnisliste wird derart vorgefiltert, dass lediglich diejenigen Dokumente enthalten sind, die der Nutzer auch sehen darf. Dass nicht alle Dokumente von der Suchmaschine geliefert werden kann im Vergleich zum Late-Binding, zudem als weiterer Sicherheitsmechanismus interpretiert werden. Offen bleibt an dieser Stelle, wie eine Early-Binding Lösung technisch umgesetzt werden kann.

5 Umsetzung von Early Binding mit Virtual Tokens

Im letzten Kapitel bleib die interessante Fragen offen, wie genau eine technische Implementierung des Early-Binding aussehen kann und welche Informationen bzw. wie diese im Index zu speichern sind. Im Folgenden soll deshalb dargestellt werden, wie mit Hilfe sogenannter Virtual Tokens laut Kasprzak at al. [7] ein Early-Binding umgesetzt werden kann.

Den Ausgangspunkt für eine spätere Volltextsuche, bildet ein invertierter Index. Der Index enthält ein Lexikon aller Wörter, welche in allen indexierten Dokumenten vorkommen. Diese Wörter werden jeweils mit den DokumentIDs gemappt, in welchen sie vorkommen (siehe Abbildung 5 links). Darüber hinaus werden jedoch auch Zugriffsinformationen der Dokumente, welcher Nutzer bzw. welche Gruppe das Dokument lesen darf, zusätzlich in den Index über sogenannte Virtual Tokens aufgenommen. Virtual Tokens unterscheiden sich von den "normalen"Wörtern dadurch, dass sie nicht in den Dokumenten vorkommen, sondern lediglich dazu dienen die Zugriffsinformationen zu codieren. Die Zugriffsrechte eines gegebenen Dokuments werden schließlich durch die Nutzer oder Gruppen repräsentiert, die das Dokument sehen dürfen und werden in der Form p:Nutzer bzw. p:Gruppenname angegeben. Das Präfix p: wurde deshalb hinzugefügt, dass es zu keinen Duplikaten mit Wörtern in den Dokumenten kommt. Es könnte aber auch jedes andere Präfix gewählt werden, solange die Einzigartigkeit des Virtual Token gewahrt wird (siehe Abbildung 5 rot markiert).

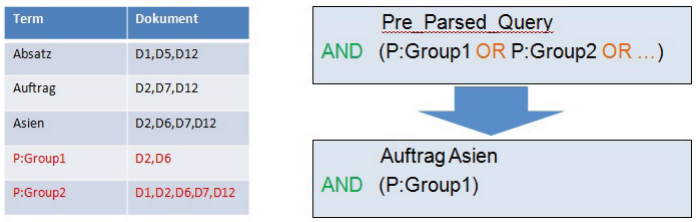


Abbildung 5. Links: Invertierte Liste; Rechts: Anfrageerweiterung. Quelle: Eigene Darstellung

Der Nutzen der Virtual Tokens erschließt sich unmittelbar nachdem der Ablauf einer eingehenden Suchanfrage betrachtet wird. Hat sich ein Mitarbeiter am System eingeloggt und eine Suchanfrage abgesetzt wird diese um seine Zugriffsrechte erweitert. Die Erweiterung erfolgt dadurch, dass an den ursprünglichen

Suchstring durch ein Boolsches AND die jeweiligen Gruppen in Form der zuvor festgesetzten Virtual Tokens (bei mehreren Gruppen jeweils mit OR verknüpft) angefügt werden (siehe Abbildung 5 rechts). Dokumente, auf die der Nutzer keinen Zugriff hat, werden frühzeitig ausgefiltert was tendenziell eine starke Performancesteigerung zur Folge hat. Im Beispiel in Abbildung 5 (rechte Seite unten) würde der Nutzer lediglich Dokument D2 und D6 von der Suchmaschine als Ergebnis präsentiert bekommen, nicht jedoch D7 oder D12 auch wenn in diesen Dokumenten die Suchbegriffe vorkommen, da er hierfür keine Zugriffsberechtigung besitzt. Da Suchmaschinen zudem auf invertierten Listen sehr schnell arbeiten können bzw. auf diese optimiert sind, können durch Virtual Tokens die Zugriffsrechte sehr elegant umgesetzt werden.

6 Probleme bei der Implementierung von Secure Enterprise Search

Im letzten Kapitel soll ein kurzer Überblick auf Probleme, die bei der Implementierung von Secure Enterprise Search Lösungen auftreten können, gegeben werden. Die einzelnen Beispiele erheben keinen Anspruch auf Vollständigkeit sondern verfolgen lediglich das Ziel dem Leser ein Gespür für mögliche Fehlerquellen zu vermitteln.

Mangelnde Struktur der Dokumente Eines der Hauptprobleme, um Zugriffsrechte adäquat zuweisen zu können, liegt in der mangelhaften Struktur vieler Dokumente begründet. Auch zu komplexe Dokumente erweisen sich als schwierig. Je nach geforderter Granularität der Suche kann die Komplexität bei der Rechtevergabe enorm ansteigen. Je höher die Komplexität, desto höher wird auch die Wahrscheinlichkeit das Fehler in den Zugriffen entstehen, welche zu Sicherheitslücken führen können.

Um Datenpannen zu vermeiden müssen die Datenquellen analysiert werden. Je nach Struktur der Quellen muss die Granularität der Suche angepasst werden. Demnach wäre bspw eine Sicherheit auf Dokumentenebene einer Sicherheit auf Sub-Feld Ebene vorzuziehen, wenn die Quellen stark unstrukturiert sind.

Mangelhafte Filterung der Ergebnislisten Eines großes Problem bei SES Lösung ergibt sich, wenn die Ergebnislisten der Suche nicht sorgfältig gefiltert wurden. So können bereits bei der Ergebnisdarstellung durch den Titel oder durch Snippets wesentliche Informationen herausgelesen werden, auch wenn vielleicht der Zugriff auf das Dokument als ganzes verweigert wurde. Ein Hinweis im Titel kann bereits aus dem Kontext heraus genug Informationen liefern um zu einem Datenleck zu führen.

Die Sicherheit sollte folglich nicht nur auf einzelne Teile oder Schlüsselworte ausgerichtet sein, sondern im optimalen Fall auf Basis des semantischen Inhalts des Dokuments erfolgen [8, S. 1]. Um die Semantik von Dokumenten zu erfassen, existieren unter dem Stichwort Web Semantics bereits erste Ansätze, welche

jedoch z.T. noch deutlichen Forschungsbedarf benötigen. Interessierte Leser seien auf das Buch Web Semantics [9] verwiesen.

Caching Um die Performance zu steigern wird meist auf Cachingmechanismen zurückgegriffen. Hierzu werden temporär Daten zwischengespeichert um die Verarbeitungsgeschwindigkeit zu erhöhen. Jegliche Zwischenspeicherung sicherheitskritischer Daten erhöht jedoch auch das Risiko, dass Unberechtigte Zugang zu kritischen Informationen bekommen.

Als Lösung muss von Fall zu Fall abgewogen werden und ein Kompromiss zwischen der Performance auf der einen und den Sicherheitsanforderungen auf der anderen Seite gefunden werden [8, S. 1].

Datenschutz verbietet Crawling und Indexierung Um die notwendigen Dokumente zugreifbar zu machen müssen diese indexiert werden. Dies gilt sowohl für die öffentlichen als auch für die privaten Dokumente. Um den invertierten Index erstellen zu können benötigt der Crawler umfassenden Zugang zu allen Dokumenten. Dies kann jedoch aus Datenschutzgründen verboten sein. Zudem kann es zur Gegenwehr in Abteilungen kommen, wenn diese nicht wollen, dass ihre Ergebnisse indexiert werden. Denn, was nicht indexiert wird, kann auch nicht gefunden werden. Die Notwendigkeit eines Crawlers vollen Zugang zu den Dokumenten zu bekommen und gleichzeitig aber den Zugang zu beschränken stellt häufig eine Herausforderung dar [6, S. 1].

Eine Möglichkeit ergibt sich hierbei durch den bereits kurz angesprochenen Federated Search. Herrschen enge Datenschutzbestimmungen muss eventuell von der monolithischen Suchmaschinenstruktur abgewichen werden und mehrere *kleinere* Suchmaschinen mit eigenem invertiertem Index umgesetzt werden.

Extraktion notwendiger Zugriffsinformationen aus Altsystemen häufig schwierig Aufgrund historisch gewachsener Altsysteme und Sicherheitsstandards im Unternehmen gestaltet sich die Extraktion von bspw. ACL Informationen aus diesen Systemen häufig als Herausforderung, wenn eine Suchlösung implementiert werden soll.

Bei der Auswahl eines Anbieters für eine Secure Enterprise Search Lösung ist deshalb besonders darauf zu achten, dass die SES Lösung geeignete Schnittstellen für die Integration mit existierenden (Legacy-) Systemen bereitstellt [8, S. 1].

7 Fazit

Die vorliegende Arbeit hatte zum Ziel einen ersten Überblick zum Thema Secure Enterprise Search zu vermitteln. Ein Schwerpunkt lag in der Beantwortung der Frage, wie sichergestellt werden kann, dass nur autorisierte Nutzer Zugriff auf sicherheitskritische Dokumente bekommen können. Neben Rollenmodellen für Nutzer und der Granularität der Suche, wurden die zwei gängigsten Architekturen - das Late-Binding und das Early-Binding - vorgestellt, sowie für letzteres

eine mögliche Implementierung auf Dokumentebene über Virtual Tokens im invertierten Index aufgezeigt.
Anbieter von fertigen Secure Enterprise Search Lösungen, wie Oracle, IBM oder Google unterstützen in ihren Lösungen häufig sowohl Late- als auch Early-Binding, wobei meist wenig Details über die zugrunde gelegte Implementierung offengelegt werden. Bis auf vereinzelte Whitepaper und den Manuals der kommerziellen Lösungen sind wenige Informationen bekannt. Interessierte Leser seien für weitere Informationen bspw. auf die Datenblätter zu Oracles Secure Enterprise Search [10] [11], und Googles Whitepaper zu Enterprise Search und Access Control [12] verwiesen. Die Informationen konzentrieren sich jedoch hauptsächlich auf Beschreibung von den Schnittstellen der Lösungen sowie auf die für die einzelne Suchmaschine spezifischen technischen Details. Im Open Source Bereich sind dem Autor keine Quellen bekannt, die eine fertige Secure Enterprise Search Lösung bereitstellen (von „kreativen“ Eigenkreationen mit Solr abgesehen).
Die Bereitstellung einer zentralen, über alle Datenquellen integrierten, Suche im Unternehmen wird in Zukunft vermutlich weiter an Bedeutung zunehmen. Aufgrund er zahlreichen Datenpannen der letzten Jahre gilt es einen guten Kompromiss zu finden, zwischen stärkerer Absicherung der Informationen einerseits und andererseits einer bestmöglichen Unterstützung der Mitarbeiter, bei der für sie arbeitsrelevanten Suche. Hierfür besteht für die Zukunft noch weiterer Forschungsbedarf

Literatur

1. Long, J.: Google Hacking for Penetration Testers. 2nd edition edn. Syngress, Burlington, MA (2007)
2. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval - The Concepts and Technology Behind Search. 2ed edition. edn. Addison Wesley, Amsterdam (2011)
3. Oliver, S.: Wie sich interne Suchmaschinen datenschutzkonform einsetzten lassen. Website (2009) Online verfügbar unter http://www.datenschutz-praxis.de/fachwissen/fachartikel/wie-sich-interne-suchmaschinen-datenschutzkonform-einsetzen-lassen/; besucht: September, 2012.
4. Coveo: The Value of Secure Enterprise Search. Website (2007) Online verfügbar unter http://www.infostratasolutions.com/docs/coveo/Coveo%20Value%20Of%20Enterprise%20Search%20Whitepaper.pdf; besucht: September, 2012.
5. Sawant, S.V.: Implementing Secure Enterprise Search. Website (2009) Online verfügbar unter http://www.mphasis.com/pdfs/WhitePapers/ImplementingSecureEnterpriseSearch_May%202009.pdf; besucht: September, 2012.
6. Bennett, M.: Mapping Security Requirements to Enterprise Search. Website (2007) Online verfügbar unter http://www.ideaeng.com/storage/Mapping-Security-Requirements-to-Enterprise-Search.pdf; besucht: September, 2012.
7. Kasprzak, J., Brandejs, M., Cuhel, M., Obsivac, T.: Access Rights in Enterprise Full-text Search. Website (2010) Online verfügbar unter http://www.fi.muni.cz/reports/files/2010/FIMU-RS-2010-08.pdf; besucht: September, 2012.
8. Autonomy (an HP Company): Enterprise Search: Addressing Security and Entitlement Issues. Website (2012) Online verfügbar unter http://publications.autonomy.com/docs/addressing-security-wp; besucht: September, 2012.
9. Hitzler, P., Krötzsch, M., Rudolph, S., Sure, Y.: Semantic Web - Grundlagen. Springer London, Limited, London (2008)
10. Oracle: Secure Enterprise Search - Version 11.1.2.2. Website (2011) Online verfügbar unter http://www.oracle.com/technetwork/search/oses/overview/ses-datasheet-11-129111.pdf; besucht: September, 2012.
11. Oracle: Secure Searching with Oracle Secure Enterprise Search 11g. Website (2011) Online verfügbar unter http://www.oracle.com/technetwork/search/oses/overview/ses-securesearch-wp-129549.pdf; besucht: September, 2012.
12. Google: Enterprise Search and Access Control with the Google Search Appliance. Website (2006) Online verfügbar unter http://www.bluepoint.net.au/google-search/google-enterprise-security; besucht: September, 2012.