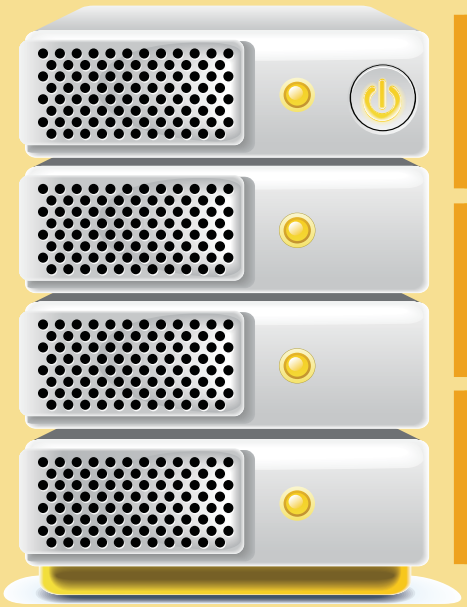
Mario Fischer

OPEN THE BOX!

WIE SUCHMASCHINEN ARBEITEN



SUCHINDEX

Hitlist

DocID\ Wort\ Kennzahl (Position-Ausprägung)
#1234\ toll\ 017-02
#1234\ webseite\ 104-16
#1234\ toll\ 211-96
#...

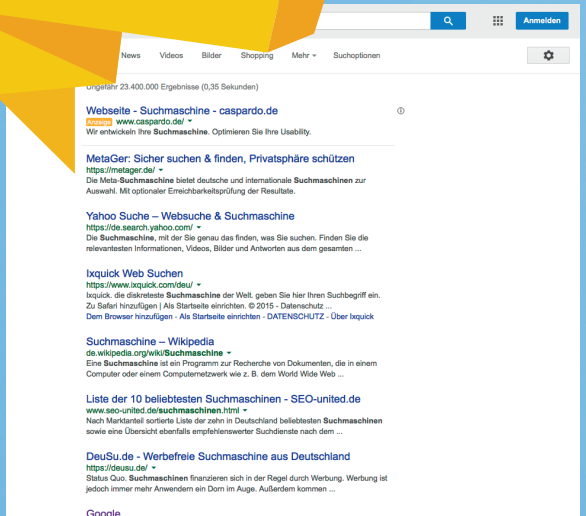
Direkter Index

DocID\ Worte\ Hitlist
#1234\ toll\ 017-02; 211-96
#1234\ webseite\ 104-16
#1234\ westernstiefel\ 003-01; 081-48; 093-64
#1235\...

Wortindex (invertiert)

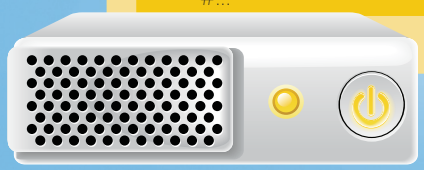
Wort\ DocIDs (Häufigkeit; Relevanz)
toll\ #1234(12;8); #86654(14;3); #4711(3;10); ....
troll\ #68834(18;5); #12545(11;8); #487771(2,1);..
trolley\ #...

Suchanfragen
QUERY–PROZESSOR

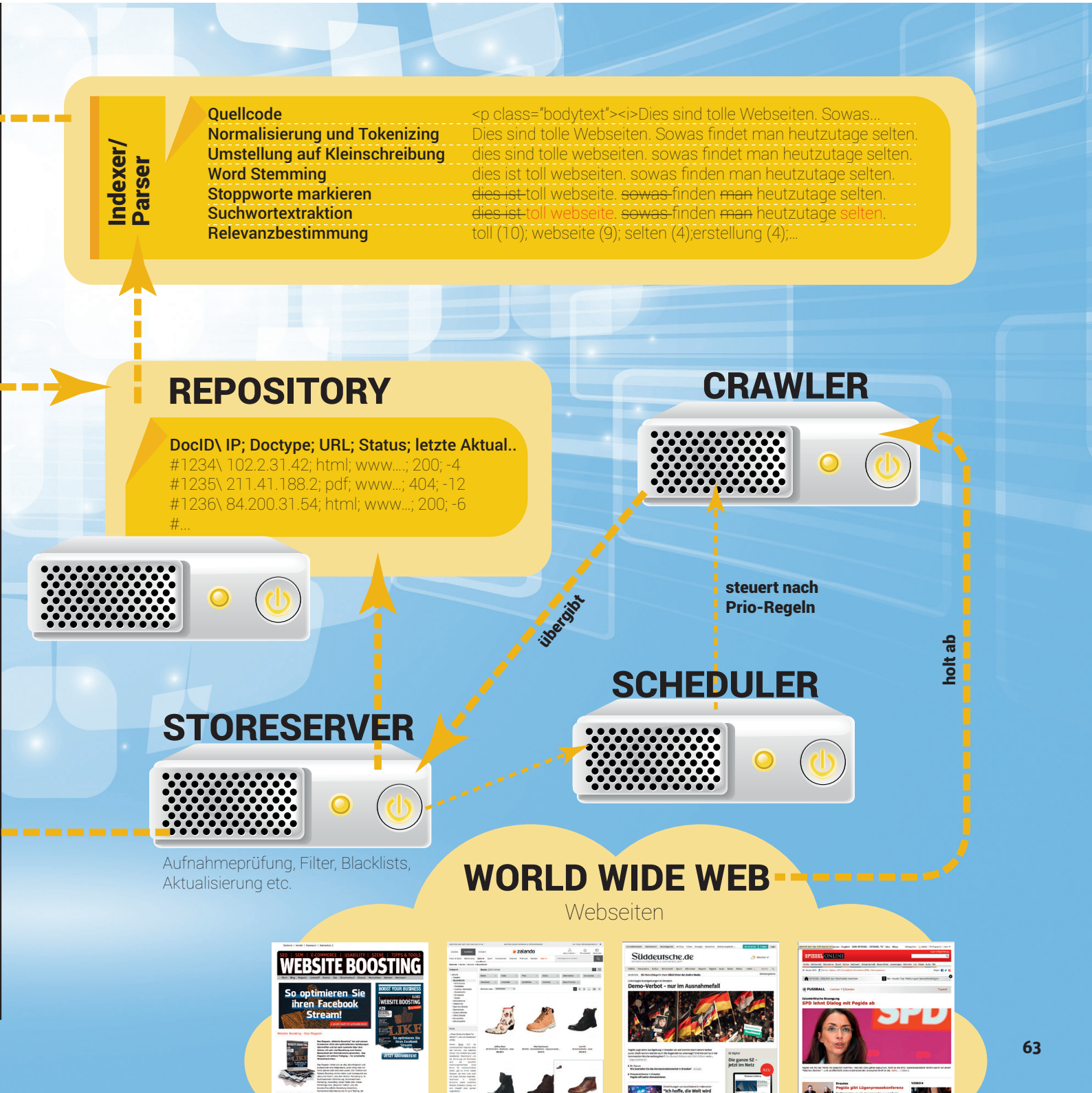


DOKUMENTINDEX

DocID\ IP; Doctype; URL; Status; letzte Aktual..
#1234\ 102.2.31.42; html; www...; 200; -4
#1235\ 211.41.188.2; pdf; www...; 404; -12
#1236\ 84.200.31.54; html; www...; 200; -6
#...



Wir nutzen sie meist mehrmals täglich, freuen uns über gute Treffer und verschwenden keinen Gedanken daran, wie es möglich ist, aus mehreren Millionen Dokumenten zehn oder mehr passende Ergebnisse innerhalb von Sekundenbruchteilen zu finden. Und mehrfach tausendmal gleichzeitig pro Sekunde. Suchmaschinen – wie funktionieren diese gigantischen Datensammlungen eigentlich und wie schaffen sie es, aus schnöder Prosaabfragesprache à la „Wie groß ist Justin Biber?“ zu erkennen, dass hier niemand etwas über die durchschnittliche Größe eines selten gewordenen Baustamm-Abnagers wissen möchte?



Wie man eine Suchmaschine baut bzw. welche einzelnen Systeme dazu notwendig sind, ist prinzipiell bekannt. Das Geheimnis liegt in der Effizienz und damit in den unendlichen Details der Algorithmen, die Daten sammeln, aufbereiten, ablegen und auf Anforderung wieder heraussuchen und dabei noch nebenbei manipulierte Dokumente herausfiltern. Diese Algorithmen entstanden meist in vielen Tausend Personenjahren Forschung und Entwicklung von Unternehmen wie Google, Bing, Yahoo!, Yandex, Baidu und anderen und es ist allzu verständlich, dass die Details geheim gehalten werden. Die realen Abläufe bei Suchmaschinen sind naturgemäß extrem komplex, und sie darzustellen, würde bei Weitem den Rahmen sprengen, der für einen solchen Beitrag zur Verfügung steht. An vielen Stellen werden Dinge daher stark vereinfacht und in prinzipieller Weise dargestellt. Die Grundprinzipien des Information Retrieval (IR) sind allerdings im Wesentlichen gleich und für das notwendige Grundverständnis über die Arbeitsweise einer Suchmaschine genügt dies daher. Dieses Wissen ist natürlich nicht nur für SEO nützlich.

Eine Suchmaschine besteht vereinfacht gesagt im Wesentlichen aus drei großen Verarbeitungsmodulen:

» Dem **Crawler** (auch Bot oder Robot), der auf Anweisung vom Scheduler Dokumente holt zur Vorkategorisierung und zur Speicherung (im Repository) weiterreicht.
» Dem **Indexer**, der die unstrukturierten Daten aus dem Repository holt, aufbereitet und zur Speicherung an den Suchindex übergibt.
» Dem **Query-Prozessor** (auch Query Engine genannt), der über die Benutzerschnittstelle die Suchworte aufnimmt, zerlegt (parst) und die Ergebnisse aus dem Wortindex holt.

ZAHLENSPIELE

Pro Sekunde gehen etwa 1,4 Millionen neue Webseiten (einzelne Seiten) erstmals ans Netz. Pro Sekunde! Für einen Laien ist es kaum vorstellbar, wie eine solche Masse an neuen Informationen praktisch in Echtzeit aufgenommen und verarbeitet werden kann. Zudem müssen die Crawler ja auch regelmäßig den bereits vorhandenen Datenbestand von meist mehreren Billionen Seiten aktualisieren. Würde Google mit seinen 60 Billionen indexierter Seiten auch nur die Hälfte davon wenigstens einmal pro Jahr aktualisieren, kämen pro Sekunde noch mal ca. 57 Mio. zu crawlende Seiten dazu. Experten wissen, dass nach Bekanntgabe z. B. einer neuen Blogseite via „Ping“ diese Seite bereits nach wenigen Sekunden im Index vorhanden und auffindbar ist. Eine selbst für Fachleute unvorstellbare Leistung. An dieser Stelle wird auch klar, warum man nicht mal eben eine neue Suchmaschine bauen kann, wie es Jacques Chirac und Gerhard Schröder fast großmannssüchtig und fachlich eher naiv mit „Quaero“ versuchten. Von dem Projekt blieb am Ende nicht viel mehr übrig als viele versenkte Steuermillionen aus EU-Fördertöpfen.

Crawler und Scheduler

Der Crawler ist für die Beschaffung von Informationen aus dem Web zuständig. Die aufzurufenden Adressen bekommt er vom Scheduler zugeteilt. Dort wird bestimmt, wann eine Webadresse (URL) erneut auf Veränderungen geprüft wird. Je nachdem, ob ein Dokument gegenüber der letzten Version verändert wurde, werden die Besuchsintervalle immer weiter verkürzt oder ggf. auch verlängert. So lässt sich sicherstellen, dass z. B. ein Impressum, das meist inhaltlich seltener verändert wird, nicht so häufig gecrawlt werden muss wie z. B. eine Seite mit einem Blogbeitrag und wachsenden Kommentaren. Die Herausforderung besteht darin, diese Intervalle möglichst genau zu ermitteln, damit einerseits keine wertvollen Ressourcen verschwendet werden, aber andererseits der Index aktuell genug ist. Probleme machen dabei Seiten mit doppelten Inhalten oder Dokumente, die aus der technischen Sicht des Crawlers wenig oder keinen Text enthalten (z. B. Flash- oder Ajax-Programmierung). Generiert ein Content-Management- oder Shop-System dynamische Webadressen, muss der Crawler seine Arbeit abbrechen. Er findet nämlich ständig „neue“ URLs, die in Wirklichkeit aber immer den gleichen Inhalt haben. Solche ungünstigen Konstellationen nennt man „Spider Trap“, also eine Falle – auch wenn diese in der Regel nicht mit Absicht aufgestellt werden. Grundsätzlich versucht der Scheduler alle neuen Adressen, die über Links gefunden werden, ebenfalls aufzurufen und somit ständig den Index zu vergrößern.

Storeserver

Der Storeserver koordiniert das Handling der gecrawlten Seiten und die Weitergabe zur weiteren Verarbeitung. Grundsätzlich ist anzumerken, dass Suchmaschinen dokumentorientiert arbeiten. Dass verschiedene Dokumente unter einer Domain erreichbar sind, ist ein Indiz für die Zusammengehörigkeit, das ist aber nicht immer zwingend so. Erhält er beispielsweise als Rückmeldung einen Fehlercode vom Webserver, wird dies vermerkt und über den Scheduler wird versucht, die Seite zu späteren Zeitpunkten nochmals zu crawlen. Sind Seiten dauerhaft nicht erreichbar oder verzogen (wird per 301-Weiterleitung vom Server bekannt gegeben, sofern das eingestellt wurde), werden die aktiven Einträge entfernt. Auch Dubletten (Duplicate Content) werden hier über die Bildung sog. Hashcodes erkannt und als solche markiert. Blacklisten, also Aufstellungen unerwünschter Webseiten oder -sites, werden ebenfalls hier verwaltet. Dort können aber auch Begriffe hinterlegt sein, die in bestimmten Ländern aus rechtlichen Gründen zensiert werden müssen.

Fällt die nach den hinterlegten Regeln durchgeführte „Aufnahmeprüfung“ positiv aus, reicht der Storeserver die Seite weiter an das Repository und den Dokumentenindex.

```
<div id="ttcont35">
        <div class="csc-header csc-header-n1">
                <h1 class="csc-firstHeader">Selbstbezügliche Sätz
        </div>
        <p class="bodytext"><i>Die nachfolgenden Sätze haben alle
t vermischt sich mit einer Aussage über den Sachverhalt. </i> </p
        <p class="bodytext">Dies ist ein Satz mit &amp;quot;Zwiebelri

        <p class="bodytext">Dies ist ein Hamburger mit Vokalen, K
        <p class="bodytext">Ich bin nicht das Subjekt dieses Satz
```

Abb. 2: Bei der Normalisierung wird lesbarer Text extrahiert

Dokumentenindex

Hier werden zu jeder URL eindeutige IDs (DocID) erzeugt und wichtige Kennzahlen gespeichert wie z. B. die IP-Adresse, der Zeitpunkt der letzten Aktualisierung, der Status des Dokuments, der Dokumenttyp und vieles andere. Auch hier gilt, dass aus Platz- und Performancegründen natürlich keine Klartextinformationen abgelegt werden, sondern dass alle Ausprägungen in einfache Kennziffern umgewandelt werden. So könnte statt des Statuscodes 200 an der entsprechenden Stelle einfach eine 1 hinterlegt werden, für einen 404-Code eine 2 und so weiter. Über eine weitere Umwandlung in Hexadezimalcode wird nochmals Speicherplatz eingespart.

Repository

In dieser Datenbank werden die vom Storeserver gelieferten Dokumente als Kopie abgespeichert. Dazu bekommt jedes Dokument eine eindeutige URL bzw. eine ID zugewiesen und die Version wird mit einem Zeitstempel (Timestamp) versehen. Wird eine neue Version eines Dokuments geliefert, wird der Eintrag aktualisiert und die Veränderungen zur Vorversion werden archiviert. Wie viele Versionen einer Webseite wie lange aufbewahrt werden, ist nicht bekannt, aber man kann mit großer Sicherheit davon ausgehen, dass es Obergrenzen und spezielle Algorithmen gibt, die speichermäßig gesehen ein Ausufern bei sich häufig ändernden Dokumenten begrenzen bzw. in solchen Fällen ggf. nur Kennzahlen erzeugen und ablegen, statt wirklich alle Versionen einer Seite aufzuheben.

Indexer/Parser

Der Indexer repräsentiert das Herzstück der Informationsverarbeitung einer Suchmaschine. Er greift auf das Repository zurück und errechnet aus dem abgespeicherten Quellcode eines Dokuments eine Liste relevanter Suchworte. Diese legt er dann im Suchindex ab. Die meisten Suchmaschinen greifen an dieser Stelle auch auf Lexika aller Art zurück. Damit lassen sich z. B. die Rechtschreibung leichter abgleichen, Synonyme verketten, Homonyme erkennen (gleiche Wörter, die eine unterschiedliche Bedeutung haben, wie z. B. „Golf“ als Auto oder als Sport). Weitere Lexika erhalten gebräuchliche Vornamen, Orts- und Länderbezeichnungen und weitere spezielle Einträge. Damit ist bestimmbar, dass „Helmut“ mit hoher Wahrscheinlichkeit ein männlicher Vorname und damit Teil einer Namensnennung ist oder „München“ eine Ortsbezeichnung. Die Realität im Datenmodell ist natürlich ungleich komplexer, als es hier dargestellt werden kann. Führende Suchmaschinen arbeiten auch mit sog. Tripel Stores und speichern damit Beziehungen zwischen Entitäten (Objekten) ab. Über diese Komponenten „lernt“ eine Suchmaschine. Wurde erkannt, dass in einem „shop“ in der Regel „produkte“ vorhanden sind, dass ein „produkt“ einen „value“, also Wert/Preis hat und ein „shop“ immer mit einer realen Adresse verknüpft ist, kann man die semantische Beziehung zwischen Wörtern besser verstehen. Daher wissen Suchmaschinen wie Google oder Bing, dass in Paris der Eiffelturm steht, wer ihn wann erbaut hat, wie hoch er ist – oder wer der aktuelle Lebenspartner eines Prominenten ist. Diese Informationen helfen ganz enorm, die geparsten Begriffe besser zu klassifizieren und am Ende noch relevantere Suchergebnisse über die aufgebauten Ontologien liefern zu können.

Normalisierung

Der erste Schritt des Indexers besteht darin, die gefundenen Informationen zu „normalisieren“ bzw. zu homogenisieren, damit man sie in Datenbanken in einem einheitlichen Format vernünftig speichern kann. Dokumente im Web bestehen aus Sicht von Maschinen aus relativ unstrukturierten Daten. In einem HTML-Dokument gibt es neben lesbarem Text noch Textanweisungen, die der Formatierung und Programmierung dienen. Lesbare Texte liegen zudem auch unstrukturiert in Navigationsbereichen, als Ankertexte, in Aufzählungen oder in Fließtexten etc. vor. Bei der Normalisierung schneidet der Indexer alle Nicht-Textinformationen ab und extrahiert somit den reinen Content.

Tokenizing

Im nächsten Schritt sorgt der sog. Tokenzier (Token = Element) dafür, dass einzelne Wörter identifizierbar werden. Das ist gar nicht so trivial, wie sich das vielleicht anhört. Nicht immer sind Leer- oder Satzzeichen Worttrenner, wie die folgenden Beispiele zeigen.

- Wir trafen uns um 7.15 p. m.
- Wir trafen uns um 7.15 p. m. in St. Augustin.
- „Warum“, fragte Dr. O´Hara-Meyer aus New York, „ist E-Commerce so *kompliziert*?“

Sind „251.23“ zwei Zahlen oder handelt es sich im ein im amerikanischen üblichen Dezimalpunkt, also unser Komma, der die beiden Zahlen in einen wichtigen Zusammenhang setzt? Macht es Sinn, die Wörter „New“ und

„York“ getrennt zu behandeln? Es geht also beim Tokenizing darum, semantisch zusammenhängende Wörter bzw. Elemente zu identifizieren. Da die Fehlerquote mit einfachen Regeln („Ein Punkt beendet einen Satz“) zu hoch wäre, bedienen sich moderne Suchmaschinen sog. neuronaler Netze, um die Erkennungsraten zu verbessern. Der einfacheren Vergleichbarkeit wegen werden alle Wörter auf Kleinschreibung umgestellt.

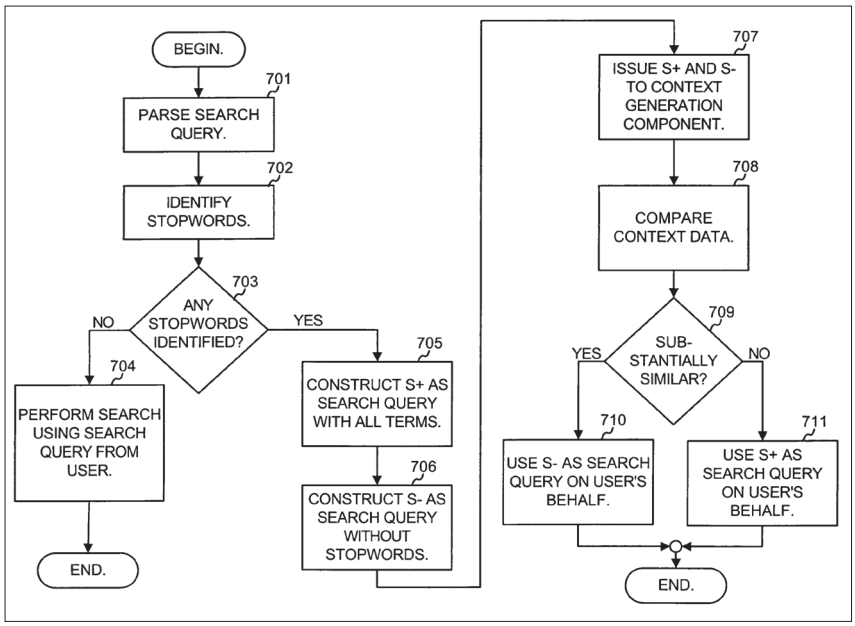


Abb. 3: Google vergleicht sogar, ob die Ergebnisse mit oder ohne Stoppwörter ermittelt werden sollten *(Quelle: Google, US Patent US7409383)*

Sprachidentifikation

Die Sprache, in der ein HTML-Dokument abgefasst ist, kann der Betreiber im Meta-Tag „language“ codieren. Da dies allerdings bezogen auf die Masse der Seiten vergleichsweise nur wenige Webmaster machen und zudem durch die Verwendung ungenügend angepasster Vorlagen falsche Deklarationen nicht unwahrscheinlich sind, können sich Suchmaschinen nicht auf diese Selbstauskunft verlassen. In der Regel genügt es, mit einem Set eindeutiger Tokens einen Abgleich mit Wörterbüchern vorzunehmen, um eine relativ zuverlässige Erkennung der tatsächlich verwendeten Sprache zu gewährleisten. Dies ist nicht nur für die Filterung nach der Sprache eines Suchenden wichtig, sondern auch, um die nachfolgenden Schritte korrekt durchführen zu können – denn sie variieren zum Teil erheblich.

Wordstemming

Die Wissenschaft spricht hier von der Lemmatisierung von Lexemen (die sprachliche Grundform). Dabei geht es darum, ein Wort auf seine Grundform zurückzuführen. So lässt sich z. B. „gelobtes“ auf das Verb „loben“ oder das Substantiv „Lob“ zurückführen. Diese Reduzierung hat mehrere Vorteile. Zum einen reduziert sich die Anzahl der Wörter, die im (invertierten) Suchindex abgelegt werden müssen. Zum anderen lassen sich über die gemeinsame Grundform auch mehr Webseiten zu einem Thema als für die Suche möglicherweise relevant identifizieren. Wenn auf einer Webseite steht „das Alter von Justin Biber ist 14“ und auf einer anderen „Justin Biber ist 14 Jahre alt“, könnten beide eine Antwort auf eine entsprechende Suchanfrage geben. Würde eine Suchmaschine nur tumb das Wort „alt“ in einem Suchstring „Wie alt ist Justin Biber?“ mit dem Vorkommen der Wörter „Justin Biber“ und „alt“ auf Webseiten prüfen, wäre das Ergebnis jeweils auf eine exakte Übereinstimmung von Begriffen eingeschränkt. Google verwendet daher bereits seit 2004 Verfahren des Wordstemming. Auch Yahoo!, Bing und andere Suchmaschinen setzen diese Verfahren ein, der genaue Umfang ist allerdings wie meist unbekannt. Die Kunst beim Stemming ist, das System zwischen der einfachsten Art, Plural- in Singularformen zu bringen, und einer ausufernden Übertreibung und einer damit einhergehenden Erhöhung von Fehlern auszubalancieren. Selbstverständlich geht die Information der tatsächlichen Wortverwendung beim Stemmen nicht verloren und kann dem Lexem, also der Grundform, beim Abspeichern mitgegeben werden.

Beim Stemmen würde z. B. das Wort „auffinden“ der Grundform „finden“ zugeordnet und dort mit einer Kennziffer hinterlegt:

> **WordID: 4712 /finden/** gefunden(1); findet(2); findbar(3); findend(4); auffinden(5); … #fundort; #fundstück; #fundsache …

Nach diesem (völlig fiktiven) Schema muss man, statt speicherintensiv einzelne Wörter zu speichern, z. B. nur 4712-5 ablegen und kann dies bei Bedarf über den Datenbankeintrag wieder zurückverwandeln. 4712 beschreibt die Zeile und die Flexionsziffer 5 die Ausprägung „auffinden“. Da man weiß, dass alle Einträge mit gleicher Wort-ID (hier 4712) auf denselben Wortstamm zurückgehen, lassen sich Vergleiche sehr viel besser und vor allem deutlich schneller durchführen. Einer der bekanntesten und oft verwendeten Stemming-Algorithmen ist sicherlich der von Porter (weitere Informationen unter *http://einfach.st/port1*).

Stoppwörter ermitteln

Nicht alle Begriffe sind für Suchanfragen relevant. Je häufiger ein

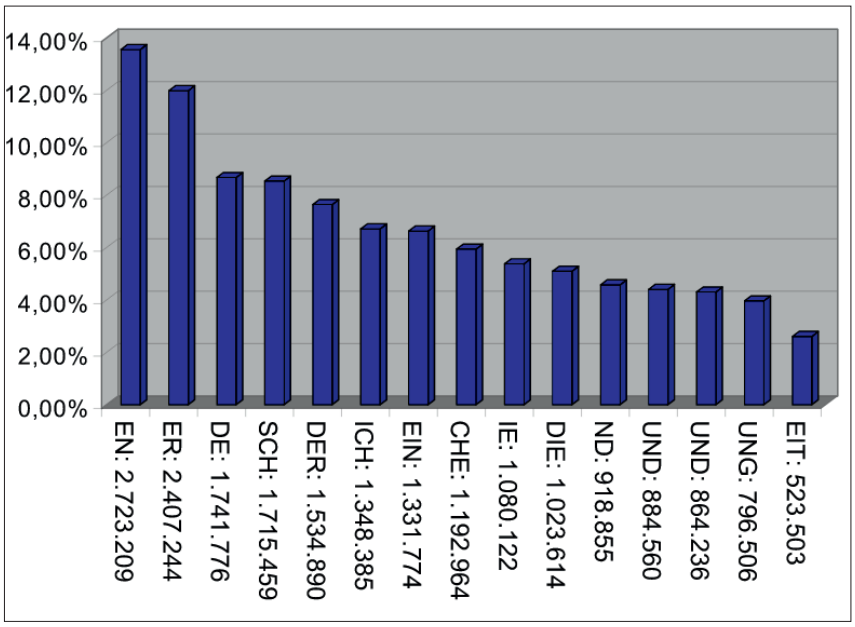


Abb. 4: Trigrammanalyse – wie häufig kommen welche Kombinationen von drei Buchstaben vor? *(Stichprobe 20,1 Mio. Texte - Quelle: Wikipedia, http://einfach.st/wp6)*

Wort in einem Text vorkommt, desto ungeeigneter ist er genaugenommen. Eine Suche nach „der" oder „dieses" erscheint wenig sinnvoll. Daher spielen sie bei der Identifikation eines Themas und der Bedeutung einer Webseite für ein treffendes Suchwort eine untergeordnete Rolle. Suchmaschinen sortieren solche Stoppwörter wie „und", „ein", „das" etc. daher vor der Weiterverarbeitung in einem gesonderten Schritt aus. Damit sinkt die Zahl der Wörter für die weitere Analyse und spätere Speicherung noch weiter.

Die nachfolgenden 100 Wörter stellen nach dem Vorkommen in deutschen Texten fast die Hälfte (47,1 %) der jeweils verwendeten Wörter:

> der, die, und, in, zu, den, das, nicht, von, sie, ist, des, sich, mit, dem, dass, er, es, ein, ich, auf, so, eine, auch, als, an, nach, wie, im, für, man, aber, aus, durch, wenn, nur, war, noch, werden, bei, hat, wir, was, wird, sein, einen, welche, sind, oder, zur, um, haben, einer, mir, über, ihm, diese, einem, ihr, uns, da, zum, kann, doch, vor, dieser, mich, ihn, du, hatte, seine, mehr, am, denn, nun, unter, sehr, selbst, schon, hier, bis, habe, ihre, dann, ihnen, seiner, alle, wieder, meine, Zeit, gegen, vom, ganz, einzelnen, wo, muss, ohne, eines, können, sei

Es ist unmittelbar klar, dass sich solche Wörter nicht zur Suche in Dokumenten eignen, weil sie keinen eigenen Sinn tragen (bis auf das Wort „Zeit") und praktisch im jedem Text mehrmals und damit viel zu häufig vorkommen. Sie sind typische Kandidaten für Stoppwörter. Eine Liste englischer Stoppwörter findet man beispielsweise bei van Rijsberg unter *http://einfach.st/esw1*.

Suchwortextraktion und Relevanzbestimmung

Nachdem alle störenden und weniger zur Beschreibung geeigneten Wörter eliminiert wurden, können die Kernalgorithmen die verbliebenen Wörter einer Relevanzprüfung für das Dokument unterziehen. Man kann sich das in etwa so vorstellen, dass den Wörtern nach verschiedenen Kriterien Punktwerte zugewiesen werden, die im Folgenden natürlich nur beispielhaft verwendet werden – schließlich zählen solche Details zu den großen Geheimnissen der Suchmaschinenbetreiber. Nicht komplett unrealistisch könnte das folgende Bewertungsschema sein: Ein Dokument enthält neben anderen Begriffen das Wort „Westernstiefel". Dieses kommt im Title des Dokuments vor, und zwar gleich zu Anfang. Dafür könnte man dem Wort die ersten 5 Punkte zuweisen. Zudem erscheint es noch in einem mit H1 (H steht für Headline, also Überschrift) ausgezeichneten Text (3 Punkte) und in drei von sechs H2-Textzeilen (2 Punkte). Im Fließtext wird das Wort einmal fett dargestellt (1 Punkt) und einmal in einer Aufzählung verwendet (1 Punkt). Zudem kommt es häufiger am Anfang von Textabschnitten vor als am Ende (1,5 Punkte). Ein wohl wichtiger Baustein ist die Relevanzmessung über WDF*IDF-Faktoren (siehe hierzu ausführlich den Titelbeitrag in Ausgabe 18). Dazu wird die Häufigkeit z. B. des Wortes „Westernstiefel" gegenüber allen anderen Wörtern ermittelt, jeweils über einen Logarithmus gestaucht und mit der Fähigkeit, ein gutes, trennscharfes Suchwort zu sein, in Beziehung gesetzt (IDF-Wert). Substantive spielen bei der Relevanzbestimmung übrigens eine wichtigere Rolle als Verben und Adjektive.

Bei Letzterem geht es darum, wie häufig das Wort „Westernstiefel" auch in allen anderen Dokumenten im Index auftaucht. Kommt es in sehr vielen Dokumenten vor, ist es für eine genaue Relevanzbestimmung für das vorliegende Dokument eher ungeeignet. Ein Beispiel: Das Wort „rot" kommt laut einer Google-Abfrage ungefähr in 241 Mio. Dokumenten vor, „Westernstiefel" nur in 366.000. „Lebensmittelsaugring" erscheint dagegen nur auf 74 Webseiten weltweit. Die zunehmende Eignung als relevantes Wort für ein Dokument bzw. eine Webseite erscheint mit abnehmender Häufigkeit des Erscheinens auf Webseiten durchaus logisch. WDF*IDF setzt nun

die relative Häufigkeit innerhalb eines Dokuments mit der Häufigkeit in allen anderen Dokumenten in Beziehung. Man darf getrost davon ausgehen, dass bei dem Stand der Technik auch Synonyme oder stark begriffsverwandte Wörter mit in diese Berechnung eingehen, sodass die Algorithmen nicht einfach nur stur Wörter zählen.

Ein Beispiel zur Erkennung der textlichen „Normalität" eines Textes stellt das Zipfsche Gesetz dar, das häufig in der sog. Korpuslinguistik eingesetzt wird. Es beruht darauf, dass bestimmte Wörter in einer Sprache häufiger vorkommen als andere und man dies über eine mathematische Formel abbilden kann. Über sog. N-Gramme lässt sich die Wahrscheinlichkeit berechnen, mit welcher auf ein bestimmtes Wort ein anderes folgt. Für Google Books gibt es übrigens einen Ngram Viewer (*https://books.google.com/ngrams*), mit dem sich N-Gramme gefiltert nach Zeit und Sprache aus Büchern berechnen lassen.

Auch die Häufigkeit von Buchstaben in Texten ist gut mathematisch beschreib- und damit prüfbar. Während das E mit Abstand am häufigsten (> 17 %) vorkommt, ist der häufigste Anfangsbuchstabe das D, gefolgt vom S. Der häufigste Endbuchstabe ist das N (21 %). Noch aussagekräftiger werden solche Berechnungen, wenn nicht nur die Häufigkeit einzelner Buchstaben verglichen wird, sondern das Auftreten mehrerer Buchstaben. Wie in Abbildung 4 zu sehen ist, kommen bestimmte Dreierkombinationen (Trigramme) sehr häufig vor. Die augenscheinlichen Zweierpaare „EN", „ER", „DE" und „IE" enthalten in Texten davor oder danach ein Leerzeichen, das mitgezählt wird.

Weicht die Art der Begriffs- und Buchstabenverwendung weit ab vom durchschnittlich Erwartbarem, handelt es sich möglicherweise um einen sinnfreien Text, wie er nicht selten von Spammern automatisiert erzeugt wird. Über solche Filter lassen sich auch sog. automatisiert „gespinnte" Texte erkennen, die durch Begriffsvariationen versuchen, das Duplicate-Content-Problem zu umgehen. Kurzum – mittels moderner Linguistik und speziell entwickelter Algorithmen können aus einem Text maschinell sehr viele nützliche Informationen zur Einschätzung gewonnen werden. Selbst eine Zuordnung zum Schulabschluss eines Autors ist heutzutage automatisiert möglich – und dabei handelt es nicht um einfache Verfahren, die nur Rechtschreibfehler zählen.

Selbstverständlich wird von den meisten Suchmaschinen zur Relevanzbestimmung nicht nur der Text eines Dokuments herangezogen, sondern auch Signale wie z. B. die Anzahl und Qualität der Backlinks, der Vertrauensfaktor für eine Domain (Trust), das Besucherinteresse (siehe den Beitrag von Andre Alpar in dieser Ausgabe ab S. 58), deren Verweildauer, oder auch Faktoren, ob Suchende nach einem Klick auf ein Suchergebnis schnell wieder zurückkehren und auf ein anderes Ergebnis klicken.

Dr. Jan Pedersen von Bing erklärte erst kürzlich in einem Beitrag, dass die Qualität des Contents beim Ranking dort eine sehr große Bedeutung hat:

„Content Quality is a Primary Factor in Ranking" (*http://einfach.st/bblog)*.

Seinen Ausführungen nach ist das Ranking bei Bing generell eine Funktion von

a) themenspezifischer Relevanz („Topical Relevance"),
b) Context und
c) Qualität des Contents.

Die Qualität wird durch Autorität (kann man dem Inhalt trauen?), Nützlichkeit *(„... When considering the utility of the page, our models try to predict whether the content is sufficiently useful for the topic it is trying to address. Does the page provide ample supporting information? Is it at the appropriate level of depth for the intended audience? We prefer pages with relevant supporting multimedia content: instructional videos, images, graphs, etc. Another important criterion in evaluating utility is gauging the effort and level of expertise required to generate the content. Websites serving unique content are preferred to those recycling existing data or widely available materials ...")* und die Darstellung bzw. wie gut der gesuchte Inhalt auf der Seite dargestellt und gefunden wird, bestimmt. Suchmaschinen sind also mittlerweile sehr wohl in der Lage, Texte und Wörter nicht nur zu vergleichen, sondern sind mitten in den Versuchen, diese tatsächlich zu verstehen – soweit man bei Computern wirklich von Verständnis im üblichen Wortsinn sprechen kann.

Da es in diesem Beitrag aber im Kern nicht um Rankingfaktoren und deren Einsatz geht, bleiben diese hier weitgehend unberücksichtigt. Für das generelle Verständnis der Funktion einer Suchmaschine ist es auch nicht notwendig und würde die Betrachtungen deutlich verkomplizieren.

Der Suchindex

Hitlist

Da Suchende mittels der Eingabe eines oder mehrerer Wörter nach Ergebnissen suchen, muss eine Suchmaschine aus den Texten wichtige Wörter extrahieren und so aufbereiten, dass ein performanter Zugriff darauf möglich ist. In der sog. Hitlist werden daher alle in einem Dokument vorkommenden relevanten Wörter nacheinander abgelegt. Aus dem Satz „Das ist eine tolle Webseite und hier können Sie tolle Westernstiefel kaufen" stehen nach der Eliminierung von Stoppwörtern, dem Stemming und der Fixierung der Position im Text:

» toll (an Position 004)
» webseite (an Position 005)

Kategorie	Wertziffer	Beispiel 1	Beispiel 2	Beispiel 3	Beispiel 4
Title	1				x (1)
H1	2				
H2-1	4	x (4)			
H2-2	8				
fett/kursiv	16	x (16)	x (16)	x (16)	
Fließtext	32		x (32)		
Aufzählung	64			x (64)	
...					
WZ-Summe		20	48	80	1

Abb. 5: Fiktives Beispiel einer Hitlist-Berechnung

DocID\ Wort\ Kennzahl (Position-Ausprägung)
#1234\ toll\ 004-01; 010-02
#1234\ webseite\ 104-16
#1234\ westernstiefel\ 003-01; 081-48; 093-64
#1235\ ...

Abb. 6: Prinzipieller Aufbau des direkten Index

Suchwort	DocID (Häufigkeit; Relevanz; ...)
...	
toll	#4532(12;8); #86654(14;3); #4711(3;10); ....
troll	#68834(18;5); #12545(11;8); #487771(2;1); ....
trolley	#755541(8;8); #652654(10;10); #620412(18;9); ....
...	
webseite	#78443(22;8); #44522(18;7); #4711(1;7); ...
website	#87531(34;9); #54363(8;7); #322241(15;9); ....
...	

Abb. 7. Schematische Darstellung des invertierten Index

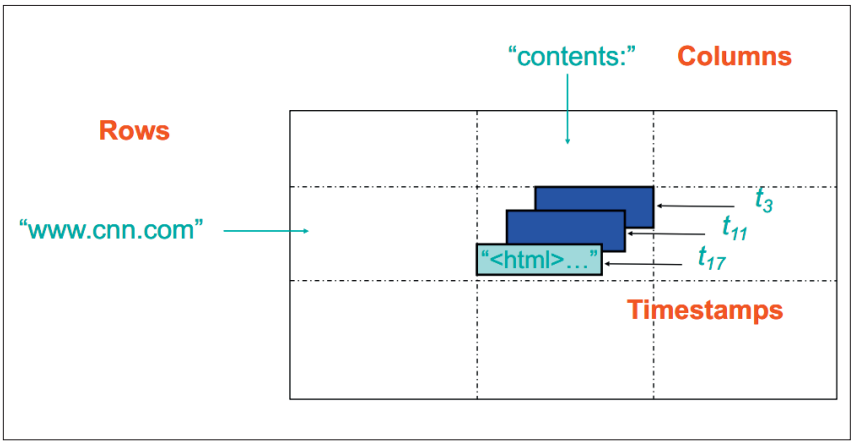


Abb. 8: Das „Basic Data Model" von Google *(Quelle: Dean Ladis; http://einfach.st/ladis)*

» toll (an Position 010)
» Westernstiefel (an Position 011)
» kaufen (an Position 012)

Neben der Position sind natürlich noch viele andere Dinge wichtig, um letztlich die Relevanz berechnen bzw. beurteilen zu können. Wurde das Wort fett gedruckt, mit relativ gesehen größerer Schriftart, wird es im Title genannt, in einer H1-Überschrift etc.? Diesen Ausprägungen können dann Wertziffern zugeordnet und bei Erfüllung für das Wort aufsummiert werden. Abbildung 5 zeigt dies exemplarisch. Kommt ein Wort in der zweiten Überschrift (H2-1) vor und ist es fett hervorgehoben, werden die Werte 4 und 16 addiert. Zu dem Wort wird also in der Hitlist neben der Position des Wortes (011) die Ausprägung 20 gespeichert. Der Eintrag wäre also:

» Westernstiefel; 011-20

Die Ausprägung „20" kann bei Bedarf natürlich über die entsprechende Tabelle wieder zurück in die Faktoren „H2-1" und „fett/kursiv" zerlegt werden. In der Realität ist dieser Ausprägungskatalog natürlich ungleich komplexer und die Speicherung wird über komprimierende Algorithmen vollzogen (Google konnte das Speichervolumen der Hitlist durch solche technischen Maßnahmen übrigens auf nur 4 Bit reduzieren). Durch diesen Prozess muss im Prinzip jedes Dokument im Web mit jedem Nicht-Stoppwort. Bei 60 Billionen Dokumente z. B. im Index von Google und der Annahme, dass eine Seite im Schnitt in etwa 200 Wörter enthält, bekommt man einen Eindruck, welche enorme Rechenpower allein für diesen fortlaufenden Prozess notwendig ist.

Direkter Index

Im direkten Index werden für jedes Dokument die dort vorhandenen Wörter abgelegt zusammen mit den Werten aus der Hitlist. Kommt ein Wort mehrfach vor, werden auch die unter-

schiedlichen Hitlists dazu gespeichert. Abbildung 6 ist zu entnehmen, dass im Dokument mit der ID #1234 das Wort „toll“ zweimal vorkommt, zuerst auf Position 4 (004) und im Title (01) und dann noch mal an zehnter Position (010) in der H1-Überschrift (02). Das Wort „westernstiefel“ kommt dreimal vor, auf den Positionen 3 (im Title), 81 (fett im Fließtext) und 93 (in einer Aufzählung). Die Ausprägungen sind wie erwähnt über die Tabelle in Abbildung 5 nachvollziehbar.

Invertierter Index – Wortindex

Der sog. invertierte, also umgekehrte Index enthält eine komplette Liste aller Suchworte in ihrer Normalform. Jedes Wort bekommt wie auch die Dokumente eine eindeutige ID. Kommt ein Wort in einem Dokument vor und wurde es vom Indexer als relevant erachtet, wird zu dem betreffenden Wort die Dokumenten-ID (DocID) abgespeichert. Somit enthält dieser Index also alle identifizierten und gespeicherten Suchworte und in den jeweiligen Zeilen die Referenznummern für alle Dokumente, für die es bei einer Suche als möglicher Treffer infrage kommt. Man kann davon ausgehen, dass, wie in Abbildung 7 gezeigt wird, zusätzlich zu jeder DocID noch weitere klassifizierende Kennzahlen abgespeichert werden, die der Indexer ermittelte und die über die Hitlists und den direkten Index berechnet wurden. In unserem Beispiel wird die Häufigkeit der Verwendung dieses Wortes im Dokument sowie eine Relevanzkennzahl (z. B. von 1-10) aufgeführt. Was Suchmaschinen hier genau ablegen und wie detailliert diese Informationen sind, ist natürlich nicht öffentlich bekannt.

Der Grund für die Verwendung eines solchen invertierten Index liegt klar auf der Hand. Wird eine Suchanfrage gestellt, müssen nicht Billionen von Dokumenten danach durchsucht werden, sondern man kann vergleichsweise schnell über parallel laufende Prozesse zunächst auf das Suchwort zugreifen und hat eine vorverarbeitete Liste an relevanten Dokumenten. Diese müssen dann „nur“ noch entsprechend gefiltert und sortiert werden. Diese Art der Datenablage bringt gegenüber traditionellen Speichermethoden exorbitante Geschwindigkeitsvorteile.

ZAHLENSPIELE

Google hält diese Daten in über 500 großen Datenbanktabellen, die größte davon mit allen Domaineinträgen ist angeblich über 70 Petabyte groß und enthält über 70 Billionen Zellen. Das gesamte Speichervolumen wird auf eine dreistellige Zahl im Exabyte-Bereich geschätzt. Das System ist darauf ausgerichtet, Daten mit einem Volumen von einem Zettabyte (10 hoch 21 Byte) handeln zu können. Zur besseren Einschätzung: 1.000 Megabyte = 1 Gigabyte, 1.000 Gigabyte = 1 Terabyte, 1.000 Terabyte = 1 Petabyte, 1.000 Petabyte = 1 Exabyte, 1.000 Exabyte = 1 Zettabyte. 2003 betrieb Google noch ein Cluster von etwas über 15.000 handelsüblichen PCs mit einem eigenen Betriebssystem. 2009 waren es schon über eine halbe Million Server. Um Daten in dieser großen Menge wirtschaftlich und performant zu handhaben, entwickelte man bei Google frühzeitig u. a. ein eigenes Filesystem (GFS), den MapReduce-Algorithmus (genauer laufen mehr als 10.000 spezialisierte MapReduce-Programme gleichzeitig) und das BigTable-Datenbanksystem. In „BigTable“ werden Daten dreidimensional gespeichert: Eine Datenzelle wird über Zeilen und Spalten sowie in einer dritten Dimension über unterschiedliche Zeitstempel (Timestamps) adressiert. Die gesamte Entwicklungszeit für „BigTable“ betrug sieben Jahre. Unter dem Namen „Hbase“ und „Hypertable“ sind übrigens kostenlose Open-Source-Versionen von BigTable verfügbar, die auf Hadoop-Clustern aufsetzen. Die chinesische Suchmaschine Baidu setzt Hypertable (Download unter *www.hypertable.org*) ein.

Der Query-Prozessor

Dieser Teil einer Suchmaschine ist für die Beantwortung von Suchanfragen zuständig. Nutzer geben ihre Suchanfrage in ein entsprechendes Feld ein und innerhalb von Bruchteilen einer Sekunde ist ein Ergebnis da. Die größte Suchmaschine Google beantwortet nach eigenen Angaben etwa 40.000 Suchanfragen pro Sekunde. Für die Beantwortung einer einzigen Suchanfrage sind bei Google mehrere Tausend Server gleichzeitig beschäftigt mit insgesamt mehr als 10 Mrd. Prozessorzyklen und in Summe ca. 100 MB Daten, die bewegt werden müssen. Als Richtlinie müssen 99 % aller Suchanfragen in weniger als 50 Millisekunden beantwortet werden. Dazu werden überall auf der Welt große Rechenzentren betrieben, die mit verteilten Daten arbeiten.

Dean Ladis von Google rechnete 2009 vor, wie sich das Zeitverhalten für die SERP (Suchergebnisseite) bei klassisch seriellem und parallelem Zugriff darstellen würde (*http://einfach.st/ladis*):

„30 seeks * 10 ms/seek + 30 * 256K / 30 MB/s = 560 ms
10 ms/seek + 256K read / 30 MB/s = 18 ms“

Die Unterschiede zwischen 560 oder 18 Millisekunden sind essenziell und rechtfertigen nicht nur den großen Maschinenpark, sondern bedingen ihn.

Der Query-Prozessor verwendet prinzipiell die gleichen Mechanismen wie der Indexer. Aus der Sucheingabe

„ich möchte günstige Westernstiefel kaufen“

wird durch das Tokenizing, Stemming und die Stoppworteliminierung (vgl. auch Abbildung 3, bei Bedarf werden auch die Stoppwörter berücksichtigt)

„günstig westernstiefel kaufen“.

Mit diesen Suchworten kann dann auf den invertierten Index zugegriffen werden. Dort sind, wie schon beschrieben, alle Suchworte gelistet und in welchen Dokumenten sie jeweils als relevant erkannt wurden. So wird auch

der Grund für den vorherigen aufwendigen Prozess des Zerlegens von Text klar: Ohne diese Invertierung zu einzelnen Suchworten wäre der blitzschnelle Zugriff auf die notwendigen Informationen bzw. infrage kommenden Webseiten als Ergebnis nicht möglich.

Wird mehr als ein Suchbegriff eingegeben (unter Berücksichtigung von Stoppwörtern), wird in der Regel solchen Seiten mehr Relevanz zugeschrieben, auf denen alle diese Begriffe vorkommen. Je näher sie im Text zusammenstehen, desto besser für die Distanzberechnung. Steht im Text einer Webseite „tolle Webseite" direkt nacheinander, beträgt die Distanz für die umgewandelte Suchphrase „toll webseite" 1. Stünde dort „Eine tolle Idee wäre, eine benutzerfreundliche Webseite zu erstellen", wäre die Distanz 5 bzw. ohne Stoppwörter 3. Diesen Effekt machten sich übrigens vor Jahren Spammer zunutze, indem sie automatisiert Millionen von Webseiten mit genauen Mehrwortsuchphrasen (Long Tail) und meist unsinnigem Text erstellten. Durch die relativ genaue Übereinstimmung solcher Suchphrasen und das künstliche Vorkommen auf diesen Seiten konnten sie eine vergleichsweise gute Rankingberechnung erzeugen. Suchmaschinen schoben diesem Treiben allerdings einen Riegel vor und daher wurde die Dominanz einer nahen Begriffsdistanz zurückgenommen.

Nachdem die für die eingetippten Suchbegriffe relevanten Seiten gefunden wurden, können die notwendigen Sortierungen, Filterungen und Echtzeitprüfungen durchgeführt werden.

Berücksichtigt werden können unter anderem z. B.:

» Die Spracheinstellungen des Suchenden
» Das Land, von dem aus die Suche ausgelöst wurde
» Das Land, in dem der Webserver steht
» Die verwendete Sprache auf potenziellen Trefferseiten
» Besondere Aktualität bei Ergebnissen (Freshness) aufgrund bestimmter Suchworte oder plötzlich stark ansteigender Suchanfragen bei bestimmten Begriffen
» Persönliche Einstellungen des Suchenden, wenn er eingeloggt ist
  - Blockierte Sites
  - Suchhistorie
  - Spezielle Sucheinstellungen (z. B. safe search=on)
  - Präferenzen und Interessen
  - Ergebnisse/Empfehlungen/Vorlieben von Freunden aus sozialen Netzwerken
» Rechtlich notwendige Zensierungen in einzelnen Ländern (in D z. B. Nazi-Websites)
» Das Gerät, mit dem die Suche ausgelöst wurde (PC, Smartphone, Tablet etc.)
» Die Einstufung der Suche als informations-, navigations- oder transaktionsorientiert
» Als regional orientiert erkannte Suchworte (z. B. München) sind enthalten
» Erkennung als Refined Search (mehrere inhaltlich zusammenhängende Suchanfragen nacheinander)
» Notwendigkeit der Beimischung besonderer Suchboxen (Bücher, Bilder, Videos, News, Maps etc.) aufgrund bestimmter Suchphrasen
» Genügend „unterschiedliche" Suchergebnisse (z. B. nicht nur Ergebnisse aus Lexika oder nur aus Blogs)
» u. a. m.

Der Unterschied zwischen dem sog. Profile-Rank bei eingeloggten Usern und dem Generic-Rank kann durchaus nennenswert sein, wie man selbst gut testen kann, wenn man sich ein- oder ausloggt. In einem Patent von Google, „Personalization of Web Search Results Using Term, Category, and Link-Based User Profiles" (*http://einfach.st/gpat12*), das 2012 genehmigt wurde, wurde dies explizit dargelegt. Laut Marcus Tandler nutzt Google derzeit 57 Signale, die auf das persönliche Userverhalten Rücksicht nehmen.

Für häufige Suchanfragen können passende Trefferlisten in ausgelagerten Zwischenspeichern abgelegt werden (sog. Query Caching). Dies erspart, jedes Mal den kompletten Retrievalprozess durchlaufen zu müssen. Je nach Suchmaschine müssen diese vorgefertigten Listen dann nur noch nach den persönlichen Vorlieben der Suchenden (personalisierte Suche) gefiltert oder umsortiert werden.

Das messbare Verhalten der Suchenden (Klickrate auf Suchergebnisse, Rückkehrrate und -zeitraum nach einem Suchklick) fließt nach Meinung vieler Experten wieder zurück in die Berechnung der Relevanz und verändert damit wahrscheinlich auch die entsprechenden Kennzahlen im Index. Insofern korrigieren sich die Bewertungsalgorithmen fortlaufend selbst, indem die errechnete Relevanz mit der von den Usern empfundenen Relevanz verglichen wird.

Hier schließt sich am Ende der Kreis zwischen dem langen Weg von der ersten Aufnahme eines Dokuments aus dem Web, der Aufbereitung und Beurteilung und dem ständigen Abgleich, ob tatsächlich richtig gewichtet wurde. Lang ist der Weg allerdings nur bezogen auf die vielen Leitungen, Glasfasern, Prozessoren, Speicherchips und Festplatten, durch die diese Informationen geschleust werden. In zeitlicher Hinsicht passiert dies alles in nur wenigen Sekunden. ¶