Heinrich-Heine-Universität Düsseldorf
Philosophische Fakultät
Informationswissenschaft
Professor Stock
Hauptseminar Google tanzt – Zur Anatomie einer Suchmaschine

Web-Crawler im allgemeinen und verbesserte Web-Crawler auf der Basis der Konzepte von Page, Molina und Cho, sowie eigene Ansätze zur Verbesserung der Qualität von Crawlern.

Vorgelegt von
Daniel Ritter
Matrikelnummer: 905396
daniel@daniel-ritter.de

1.1 Einführung

In dieser Arbeit werde ich die Funktionsweise von Web-Crawlern erläutern. Web-Crawler dienen dazu, den Datenbestand von Web-Indizes automatisiert zu erweitern und zu pflegen. Da der Crawl-Vorgang die Basis für jeden automatisiert generierten Web-Index liefert, ist eine hohe Qualität der Crawling-Algorithmen essentiell. Schliesst man bereits beim Crawlen unerwünschte Seiten aus, gelangen sie gar nicht erst in den Index, was den Pflegeaufwand des Indexes später erheblich minimieren kann. Auch das frühere crawlen von relevanten Dokumenten erhöht die Qualität des Indexes bereits beim Crawl-Vorgang. Da die Gesamtanzahl der zu crawlenden Dokumente inzwischen im World Wide Web so weit angewachsen ist, dass ein komplettes Erfassen der Dokumente unmöglich geworden ist, ist besonders auf das selektive Erfassen von „wichtigeren" Dokumenten Wert zu legen. Algorithmen zur Erfassung von Web-Dokumenten müssen weiterhin möglichst ohne grössere menschliche Eingriffe gute Ergebnisse liefern, da ein menschliches Nachindexieren der gigantischen Datenmengen nicht zu bezahlen wäre. Es wird zunächst auf grundlegende Konzepte des Web-Crawlings eingegangen werden und anschliessend werden weiterführende Ideen, zur Verbesserung der Suchergebnisse von Crawlern, vorgestellt werden. Abschliessend werde ich einige eigene Ideen vorstellen, um die Qualität von Web-Indizes bereits beim Crawl-Vorgang zu verbessen.

1.2 Begriffsfindung

To „crawl" ist ein englisches Verb, das folgende Bedeutungen haben kann:
krabbeln, kriechen, kraulen (Wassersport)

Ein Web-Crawler „krabbelt" oder „kriecht" also durch das Web. Wie in vielen englischen Begriffen aus der Informatik ist hier eine eingängige, bildhafte Repräsentation des zu beschreibenden Vorgangs gewählt worden.

Google selbst bietet einige Definitionen für den Begriff „crawler":

google.com – define: crawler
A crawler is much like a spider except it is programmed to constantly surf the web, following any and all links it comes across. As it visits new websites, it checks its own database to see if the site is listed. If the site is already listed, it makes note of any changes and calculates a search engine ranking for the site. If the site has not been previously listed, the crawler will record all important information, add the website to the database, and assign a ranking to it.

A component of a search engine that roams the Web, storing the URLs and indexing the keywords and text of each page encountered. Also referred to as a robot or spider.

A Web Crawler (or Spider) is a piece of software that scans the World Wide Web finding pages to add to the index of a search engine.

A class of robot software that explores the World Wide Web by retrieving a Web document and following the links within that document. Based on the information gathered, a crawler creates indices for search engines.

A bot that visits websites and collects data in order to create entries for search engine indexes.

A software program which visits websites to create indexes for search engines. Also known as spiders, bots, and intelligent agents.

A component of a search engine that roams the web, storing the URLs and indexing the keywords and text of each page encountered. Also referred to as a robot or spider.

A "crawler" (also referred to as a bot, robot, wanderer, or spider) is a piece of software that "crawls" the Web collecting URLs and other information from Web pages. Search engines are built around the information that crawlers retrieve.

Search engine crawlers or crawlers for short mechanically try to read each page on a web site for adding them to the data base of a search engine such as Google or AltaVista. Other varieties exist: there are crawlers which try to extract email addresses for spamming purposes.

A program that automatically fetches Web pages. Crawlers (also called spiders) are used to feed pages to search engines. Crawlers might request all Web pages at a site during a search; therefore, requests by crawlers are typically excluded from request counts. (Usage Import) 2. A program that collects files by following links contained in those files (usually over HTTP) or by following directory trees in a file system. (Search, Content Analyzer)

Zusammenfassend kann gesagt werden, dass ein Web-Crawler (andere Arten von Crawlern werden in dieser Arbeit von mir nicht betrachtet werden) durch folgende Eigenschaften zu beschreiben ist:

-Ein Crawler ist ein Softwareprogramm
-Ein Crawler ruft automatisiert Webseiten auf und findet auf ihnen Hypertext-Links zu weiteren Webseiten, welche danach ebenfalls in einer festgelegten Reihenfolge aufgerufen werden.
-Dies findet nach vorher festgeschriebenen Regeln statt.
-Ein Crawler ist an einen Index gekoppelt, in welchen er die gefundenen Seiten

einspeist.

-Ein Crawler kann auch zur Index-Pflege benutzt werden, indem er nicht mehr existierende Seiten aus dem Index entfernt oder geänderte und umgezogene Seiten im Index aktualisiert.

2.1 Der übergeordnete Crawling-Algorithmus

Die grundlegende Vorgehensweise aller Crawler ist gleich und kann leicht in einem Ablaufdiagramm mit wenigen Schritten dargestellt werden:

Schritt 1	Übergabe einer Start-URL an den Crawler
Schritt 2	Einlesen des Inhaltes der Start-URL. Schreiben der gefundenen Links in einen Speicher. Weitere individuelle Schritte für den spezifischen Crawler
Schritt 3	Einlesen des Inhaltes einer der URL's aus dem URL-Speicher. Schreiben der gefundenen Links in den Speicher. Weitere individuelle Schritte für den spezifischen Crawler
Schritt 4	Überprüfen, ob der Speicher leer ist oder eine andere Abbruchbedingung erfüllt wurde. Falls weiter gecrawlt werden soll, Rücksprung zu Schritt 3. Ansonsten Sprung zu Schritt 5.
Schritt 5	Ende des Crawlvorgangs

Die Schritte entsprechen dem Ablauf, dem man folgen würde, wenn man die undankbare Aufgabe hätte, eine Website oder gar das Web von Hand ohne automatisierte maschinelle Hilfe zu indizieren. Der kritische Punkt, welcher am meisten Potential für eine Optimierung des Crawlers bietet ist Schritt 3. Der grösste Teil dieser Arbeit wird sich damit befassen, auf welche Weise die nächste zu crawlende URL aus dem URL-Speicher ausgewählt werden sollte, um bestmögliche, spam-freie und relevante Dokumente in den Index zu übernehmen. Da der Crawler der Datenlieferant für den Index ist, ist es sinnvoll an dieser Stelle sehr viel Arbeit zu investieren, da dadurch automatisch der Index sauberer und relevanter wird. Was gar nicht erst in den Index gelangt, kann später den Benutzer, der den Index durchsucht, auch nicht belästigen. Ausserdem ist die Chance relevante Dokumente zu einer Suche zu finden natürlich höher, je mehr relevante Dokumente sich in

dem Index befinden. Die Definition, ob ein Dokument relevant für seinen Index ist, muss also im besten Fall schon während des Crawlens getroffen werden. Dadurch können Resourcen, wie Rechenzeit und Speicherplatz, aber auch aufwendiges Nachindexieren und Bereinigen des Indexes durch Menschenhand gespart werden.

2.2 Der Einlesevorgang von URL's durch den Crawler

Der eigentliche Crawl-Vorgang ist informationstechnisch nicht schwer zu realisieren. Ein Crawler führt im Endeffekt während des Crawlvorgangs keine anderen Schritte aus, als ein menschlicher Internetbenutzer, der mit seinem Browser Seiten aufruft. Sowohl der vom Menschen bediente Browser, als auch der Crawler sind HTTP-Clients, die nach festgelegten Regeln Seiten von Webservern anfordern und danach weiterverarbeiten. Der Datenverkehr im World Wide Web ist durch das Hypertext Transfer Protocol (HTTP) standardisiert. WWW-Browser und auch Server halten sich an die HTTP-Konventionen. Dieser offene, allgemein akzeptierte Standard macht das WWW erst möglich. So verstehen Server Befehle von Clients. Um eine Webseite abzurufen, genügt bereits ein einziger HTTP-Befehl, der Befehl GET. Fordert ein Client eine Seite von einem Server an, so verbindet er sich zunächst per TCP/IP mit dem Server und setzt danach den GET-Befehl für das gewünschte Dokument ab. Der Server antwortet mit einem „Response Header" und dem gewünschten Dokument, falls dieses verfügbar ist und der Benutzer die Rechte besitzt es anzufordern. Die Eingabe einer URL in die Adresszeile des Browsers führt also auch intern zu einem GET-Befehl an den gewünschten Webserver. Während Browser im nächsten Schritt den erhaltenen HTML-Code für den Benutzer einer Webseite lediglich zur Ansicht rendern, müssen Crawler diesen auf weitere Links untersuchen, durch eventuelle Filter schicken und im Index abspeichern oder verwerfen.

Das Aufrufen einer Webseite aus „Crawlersicht" ist leicht mit wenigen Befehlen auf allen gängigen Betriebssystemen nachzuvollziehen. Hier ein Beispiel unter Linux

```
# Verbindungsaufbau zum Webserver der Uni Düsseldorf auf Port 80
#(Port 80 ist der Standard TCP/IP-Port auf dem Webserver laufen sollten. Alle
```

Webbrowser gehen #von einer Verbindung zu Port 80 aus, wenn nicht explizit ein anderer Port an die URL angehängt #wird)

> telnet www.uni-duesseldorf.de 80 > ausgabe.txt

> GET / HTTP/1.1 [EINGABETASTE]

Host: www.uni-duesseldorf.de

In der Datei ausgabe.txt finden wir danach:

Hier beginnt der Verbindungsaufbau des Programmes Telnet mit dem Webserver der Uni-#Düsseldorf.

Trying 134.99.128.40...

Connected to www.uni-duesseldorf.de.

Escape character is '^]'.

HTTP-Response Heder

Diesen Header bekommt ein „normaler" Web-Surfer nie zu Gesicht, trotzdem wird er als Antwort #auf jede Anfrage an einen Webserver mitgeschickt, bevor der Document-Body, also die eigentliche #HTML-Seite übertragen wird.

HTTP/1.1 200 OK

Date: Tue, 25 May 2004 13:57:02 GMT

Server: Apache/2.0.47 (Unix) PHP/4.3.2 DAV/2 mod_ssl/2.0.47 OpenSSL/0.9.6g

Last-Modified: Tue, 25 May 2004 13:55:51 GMT

ETag: "1bac87-b9c0-1d97a7c0"

Accept-Ranges: bytes

Content-Length: 47552

Content-Type: text/html; charset=ISO-8859-1

Document Body, die eigentliche Webseite. Diesen Code sieht man auch, wenn man im Browser

die Funktion für das Anzeigen des Quelltextes wählt.

```
<center><html>
<head>
<base href="http://www.uni-duesseldorf.de/HHU/" />

  <title>
    Heinrich-Heine-Universit<E4>t D<FC>sseldorf
  </title>
  <meta name="description" content="Heinrich-Heine-Universit<E4>t D<FC>sseldorf">
  <meta name="keywords" content="">
  <meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charse
   ..... Rest der HTML-Ausgabe wurde von mir gekürzt .....
```

2.3 Das Extrahieren von URL's aus den empfangenen Dokumenten

Nach dem Einlesen eines Dokuments muss der Inhalt auf Links zu anderen Dokumenten durchsucht werden. Diese gefundenen Links stellen die Grundlage für den weiteren Crawlingvorgang dar. Diese Links werden, wie oben beschrieben, danach in den URL-Speicher geschrieben. Da Textextraktion aus Dokumenten zu den grundlegenden Aufgaben in der elektronischen Datenverarbeitung gehört, sind hierfür bereits seit Jahrzehnten mächtige Werkzeuge wie zum Beispiel die Regulären Ausdrücke vorhanden. Auch dieser Schritt ist kein grosses Problem im Crawlingprozess. Das Auslesen von URL's aus Dokumenten leistet zum Beispiel das UNIX-Kommando grep in wenigen Millisekunden. Trotzdem möchte ich noch auf ein Problem hinweisen, was beim Auslesen von URL's aus Web-Dokumenten leider nicht die Ausnahme, sondern die Regel ist. Für einen Index ist eine URL nur zu gebrauchen, wenn sie in absoluter Form abgelegt wird. In HTML können Links jedoch absolut und relativ gesetzt werden.

Beispiel für einen absoluten Link:
<a href="http://www.phil-fak.uniduesseldorf.de/infowiss/content/downloads.php">
InfoWi Downloadcenter</A>

Ein absoluter Link enthält die komplette Adresse eines Dokuments mit Host, Pfad und Dateiname.
Solche URL's stellen für Crawler kein Problem da. Die URL kann extrahiert werden und danach direkt für den Index übernommen werden.

Web-Clients unterstützen jedoch auch (standardkonform) den Aufruf von Dokumenten, die durch eine relative URL in der HTML-Seite bezeichnet werden. Angenommen wir befänden uns mit dem Webbrowser auf der Webseite
http://www.meineseite.de/home.html
auf dieser Seite befindet sich folgender Link:
<A HREF="gaestebuch.php">Zu meinem Gästebuch</A>

Das Problem wird schnell klar. Der Client setzt diesen zur aktuellen URL relativen Link um in http://www.meineseite.de/gaestebuch.php, um an den Server einen kompletten Request senden zu können. Dasselbe muss auch ein Crawler leisten, denn URL's wie „gaestebuch.php" wären im Index nicht zu gebrauchen. Ohne Angaben zu Server, Pfad und Datei ist ein Eintrag im Index nutzlos, da das Dokument später nicht wieder aufgefunden werden könnte. Der Crawler muss also nach der Extraktion der URL's aus einem Dokument diese noch auf relative Verweise überprüfen und diese in absolute umwandeln, was ohne weiteres möglich ist, da dem Crawler ja bekannt ist von welchem Host und aus welchem Verzeichnis er das aktuelle Dokument geladen hat.

2.4 Das Ablegen der Seiten im Index

Das Ablegen der Seiten im Index wird von Index zu Index verschieden gehandhabt und hat mit den Aufgaben des Crawlers nichts mehr zu tun. Der Crawler übergibt zu speichernde Seiten an andere Programme, die diese Aufgabe leisten. Deshalb wird dieser Punkt in der Arbeit nicht weiter behandelt werden.

3.1 Vorteil von gezielter URL-Wahl aus dem URL-Speicher

Nun kommen wir zum zentralen Punkt in der Optimierung eines Crawlers.
Es ist von grossem Vorteil die URL's, welche sich bereits im URL-Speicher für spätere Begutachtung befinden, nicht einfach in Eingangs- oder Zufallsreihenfolge zu crawlen. Durch gezielte Wahl der nächsten zu crawlenden URL können Rescourcen gespart werden während gleichzeitig die Relevanz und damit die Qualität des Indexes gesteigert werden kann.

Das Problem besteht nicht bei begrenzten Crawl-Vorgängen, die beispielsweise nur eine einzige Website umfassen. Hier ist die Anzahl der zu crawlenden Seiten endlich und auch eine andere Crawlreihenfolge würde den Vorgang nicht beschleunigen. Zwar wären auch hier relevatere Dokumente früher im Index verfügbar, wirklich wichtig wird diese Ordnung jedoch erst bei Crawlern, die nicht komplett erfassbare Datenbestände durchforsten sollen. Für Web-Crawler, die sich durch das gesamte Web bewegen, ist solche eine unbekümmerte und ungeordnete „Kriecherei" keine Option. Da die Anzahl der Seiten so enorm gross ist, dass nicht alle besucht und erfasst werden können, macht ein Ranking hier viel Sinn. Festplattenplatz und Internetbandbreiten sind begrenzt. Deswegen kann nur eine Auswahl des Inhalts des Webs besucht und indiziert werden. Eben deshalb ist es so wichtig zu versuchen den Crawler die relevantesten Seiten zuerst besuchen zu lassen. Durch jede relevante Seite die ein Crawler früher besucht als eine nicht relevante betreibt er Kostenreduzierung und Qualitätsverbesserung in einem Schritt. Ausserdem besteht bei nicht relevanten Seiten eher die Gefahr, dass sie zu weiteren nicht relevanten Seiten linken. Relevante Seiten bieten in der Regel gute Informationen und könnten wahrscheinlich zu vielen anderen relevanten Informationen linken, falls sie Informationshubs darstellen. Durch frühes crawlen einer nicht relevanten URL kann so viel Zeit des Crawlvorgangs damit verschwendet werden, von dieser nicht relevanten URL aus zu weiteren nicht relevanten URL's verwiesen zu werden. Das Problem ist auch dem normalen Surfer bekannt. Erreicht man gewollt oder ungewollt eine Erotikseite im Web, kann man sich vor Links zu ähnlichen Angeboten kaum noch retten. Ein langer Crawlvorgang könnte so in riesigen Datenmengen an nicht brauchbarem Material enden, falls solch eine ungewünschte URL am Anfang des Vorgangs im URL-Speicher landet.

Genau mit dieser Problematik haben sich Page, Molina und Cho beschäftigt. Mit ihren Crawlingkonzepten versuchen sie einen Crawler zu erstellen, der relevante URL's zuerst besucht, aus den bereits oben genannten Gründen.

3.2 Exkurs: Das begriffliche Problem der Relevanz

Ob ein Dokument relevant ist oder nicht, liegt leider grösstenteils im Auge des Betrachters. Ein Dokument ist für eine suchende Person dann relevant, wenn es Informationen bietet, nach denen gesucht wurde oder direkt zu diesen leitet. Google hat auf den ersten Blick kleinere Probleme bei der Auswahl von Seiten für seinen Index als spezialisiertere Suchdienste, da zunächst einmal alles für wenigstens einen der Millionen von Benutzer interessant sein könnte. Selbst Werbeseiten sind für viele Nutzer interessant, da sie offizielle Informationen zu Produkten direkt vom Hersteller bieten können. Kaum jemand würde zum Beispiel die offiziellen Web-Auftritte von McDonalds oder Nesté als „Spam" klassifizieren. Spezialisiertere Suchdienste, wie zum Beispiel eine Suchmaschine für Kinder, können viel klarer umreissen, welche Dokumente für ihre Benutzer relevant sind oder nicht. Es können viel engere Relevanzkriterien für die Crawler gesteckt werden, so können pornographische, politische und gewaltverherrlichende Seiten relativ leicht durch simple Wortfilter umgangen werden. Google hat die undankbare Aufgabe eine Linie zu ziehen zwischen global eventuell interessanten Inhalten und echtem Spam. Als echten Spam möchte ich Seiten bezeichnen, die z.B. „query hijacking" betreiben, also Google vortäuschen zu einem bestimmten Thema informationen bereitzuhalten, aber eigentlich nur auf eine ganz andere Seite weiterleiten. Beliebt sind auch Doorway-Pages, Seiten welche zu 10.000den in den Index eingespeist werden um hohe Positionen in den Suchergebnissen zu erhalten und welche weitere möglicherweise interessante Treffer nach unten verdrängen. Sicherlich können auch so genannte „Dialer-Seiten", welche dem Benutzer Teure Einwahlprogramme mit betrügerischer Absicht „unterjubeln" wollen als echter Spam bezeichnet werden. Die phantastischen Ergebnisse, die Google durch seinen Pagerank noch vor einigen Jahren lieferte sind heute leider weitestgehend Geschichte. Die Spammer haben Möglichkeiten gefunden, selbst riesige automatisiert generierte Linkblasen zu erstellen, die den hochgelobten Page-Rank Google's ins Wanken bringen. Es

besteht eine dringende Notwendigkeit weitere nicht oder nur schwer betrügbare Relevanzkriterien zu implementieren. Da Google es mit einer gewaltigen Datenflut zu tun hat, können diese Systeme nur automatisiert, dass heisst mit keiner oder nur kleiner menschlicher Interaktion funktionieren.

3.3 Crawler Konzepte von Page, Molina und Cho

Nachfolgend möchte ich nun kurz die drei grundlegenden Versuchskonzepte beschreiben, nach denen ein „URL-ordering"-Crawler sich bewegen könnte.

3.4 Der „Crawl & Stop Crawler"

In diesem Konzept besucht der Crawler eine festgelegte Anzahl von K Seiten mit einer beliebigen Start-URL. Ein ideal funktionierender Crawler hätte nach dem Besuch von K Seiten aus allen verfügbaren Seiten nur die K relevantesten extrahiert. Wären also N Seiten durch Links (K < N) verfügbar gewesen, hätte der Crawler aus diesen N verfügbaren Seiten die K relevantsten Seiten extrahiert und den Vorgang beendet. Einfach ausgedrückt könnte man sagen, dass der Crawler eine gewissen Menge von Seiten besuchen darf, die Menge der verfügbaren Seiten jedoch grösser ist und er nur die relevantesten Seiten aus der Gesamtmenge besuchen sollte.

3.5 Der „Limited Buffer Crawler"

Auch bei diesem Vorgang besucht der Crawler eine festgelegte Anzahl von K Seiten. Sein Speicherplatz in den er erfasste Seiten ablegen kann ist jedoch nur S gross (S < K). Der Crawler hat also nicht genug Platz um alle gecrawlten Dokumente unterzubringen. Hier entfernt der Crawler während des Crawlingvorgangs Dokumente aus seinem Speicher um Platz für neue zu schaffen. Bei einem ideal funktionierenden Crawler wären nach dem Erfassen von S Dokumenten nur noch die relevantesten aus der Gesamtmenge K im Speicher. Da dem Crawler die Relevanz von neue erfassten Dokumenten nicht bekannt ist muss er versuchen sie auf der Basis der bereits vorhandenen Dokumente zu „erraten".

3.6 **Crawl & Stop with Threshold**

Hier wird vorgegangen wie bei der ersten Methode. Jedoch wird ein Ziel-Qualitätskriterium für das Crawlen vergeben. Nach dem Crawvorgang von K Dokumenten muss ein perfekter Crawler alle Dokumente mit dem höchsten Qualitätswert erfasst haben.

3.7 Kriterien nach denen die nächste zu crawlende URL ausgesucht werden kann

Wie bereits weiter oben beschrieben, muss solch ein Crawling-Vorgang ein auf irgendeine Art und weise geartetes Relevanzkriterium besitzen. Der Crawler muss also die Möglichkeit haben entweder aus dem Inhalt der Seite selbst, oder aus der URL der Seite oder einem Link auf der Seite selbst heruaszufinden, ob die Seite, bzw. die Links dem gegebenen Relevanzkriterium entsprechen oder nicht. Page, Molina und Cho führen in ihrer Arbeit die 4 folgenden Relevanzkriterien an, die alle miteinander kombinierbar sind und wie wir wissen über Jahre sehr gute Ergebnisse erzielt haben:

1. Ähnlichkeit zu einem Suchwort für den Crawler (Similarity to driving query)
Seiten werden zuerst gecrawlt, auf die Links zeigen, die dem Suchwort ähnlich sind. Beim aktualisieren von Seiten werden solche zuerst gecrawlt, die das Suchwort (oft) enthalten. Diess Kriterium funktioniert leider bei einem Index, der das ganze Web erfassen will nicht. Für das crawlen on Seiten einer thematischen Auswahl bringt dieses Kriterium jedoch gute Ergebnisse. Google verwendet dieses Relevankriterium wahrscheinlich bei den Crawlvorgängen für seine spezialisierten Linux- und Windowssuchen.

2. Backlink Anzahl
Die Anzahl der Links die auf eine URL zeigen generieren ihren Backlinkwert. Dieser Wert wird immer genauer, je länger der Crawlvorgang bereits läuft. Hier wird's angenommen, dass häufig verlinkte Seiten als interessant einzustufen sind. Genau dieses Relevanzkriterium, dass auch einen grossen Teil des PageRanks ausmacht, wird von den automatisch generierten Linkblasen der Spammer angegriffen.

3. Pagerank

Der Einsatz des Pageranks, ist besonders interessant für bereits erfasste Seiten die aktualisiert werden sollen, bei nicht bekannten Seiten muss der PR geraten werden. Bei einer bereits vorhandenen Datenbank von Seiten könnte man einem Crawler so zum Beispiel dazu anweisen bei seinen aktualisierungsvorgängen zunächst Seiten mit einem hohen PageRank zu besuchen.

4. Location Metric

Hier wird als Relevanzkriterium der Aufbau der URL eines Dokumentes herangezogen. So können zum Beispiel Dokumente mit einer kurzen URL ohne Unterverzeichnisse als relevanter eingestuft werden. Hier kann auch auf verschiedene Dateitypen eingenagen werden. Man könnte beispielsweise davon ausgehen, dass pdf-Dateien von Universitäten grundsätzlich höher gewichtet werden als andere Dokumente, etc.

4.1 Eigene Ideen zur Verbesserung der Ergebnisse von Crawlingvorgängen

Leider kann auch ich nicht das „Ei des Kolumbus" zu dieser Thematik beisteuern, dennoch habe ich einige Ideen, welche zwar nicht komplett ohne, jedoch mit nur minimalem menschlichen Arbeitsaufwand die Ergebnisse von Crawlvorgängen erheblich verbessern könnten.

4.2 Idee #1: Der whois Ansatz

Die Hauptmethodik, mit der das PageRank-System untergraben wird, sind die automatisch generierten Linkblasen von Spammern, die eine riesiege Anzahl von Seiten erstellen, welche sich alle selbst untereinander verlinken. Dies muss von verschiedenen Domains aus geschehen, da das Vorgehen sonst leicht zu unterbinden wäre. Offizielle Top-Level Domains sind im Internet jedoch nicht anonym zu bekommen. Jedermann hat die Möglichkeit sich detailliert über den Inhaber eine Domain per whois-Abfrage zu

informieren. So leifert die Anfrage zu der Domain gallview.com, einer Erotikseite, zum Beispiel folgende Informationen:

Registrant:
Don H
9245 Reseda Blvd
Ste 415
Northridge
CA
US
91324
(gekürzt)

Die Domain gehört also einem gewissen Don H aus Kalifornien. Da Spammer ihre Linkblasen auf verschiedene Domains verteilen müssen, um erfolgreich durch diese Methode ihren PageRank zu heben, ist der whois-Eintrag ein sehr wirkungsvoller Ansatzpunkt, der allerdings minimale menschliche Interaktion erfordert. Menschliche „Web-Scouts" müssten dem System nur eine einzige Seite aus einer Spam-Linkblase manuell mitteilen. Das System könnte danach selbstständig alle Dokumente von Domains, welche denselben Besitzer haben entfernen, bzw. beim Crawlvorgang garnicht erst annehmen. Durch diese Methode könnten natürlich auch ungewollt „gute" Seiten aus dem Index entfernt werden, dennoch denke ich, dass die Vorteile die Nachteile hier klar überwiegen. Natürlich müsste man diese These in einem Versuch belegen. Als weiteren Nachteil könnte man hier anführen, dass höhere Rechenzeit benötigt wird, da das System vor Zugriffen auf Seiten von noch nicht bekannten Domains whois-Abfragen durchführen müsste.

4.3 Idee #2: Der standardisierte Spam-Ansatz.

Auch bei dieser Idee wäre minimale menschliche Interaktion nötig. Dennoch wäre die Umsetzung meiner Meinung nach sehr erfolgreich. Der deutsche Gesetzgeber sieht inzwischen für den Download von Dialer-Software standardiesierte Benutzermasken vor.

Die Teils kriminellen Praktiken der Dialeranbieter sind so softwaretechnisch besser zu identifizieren. Bei dem von mir ausgewählten Beispiel handelt es sich um eine Seite, auf der Malvorlagen für Kinder heruntergeladen werden können (http://www.malvorlagen.de.ms). Der Anbieter rechnet hier mit kleineren Kindern, welche sich kostenpflichtig in den sehr teuren Dienst einwählen um diese Ausmalbilder herunterladen zu können. Eine einmalige Einwahl schlägt hier mit 30€ zu Buche, ein Gegenwert für den man im Handel dutzende Malbücher erstehen könnte. Bei solchen Dialerangeboten ist immer ein Eingabefeld anzubringen, in welches das Wort „OK" eingegeben werden muss. Ausserdem müssen ein Hashwert für den Dialer sowie der Einwahlpreis angegeben werden. Alle diese Merkmale sind softwaretechnisch leicht zu erfassen. Ausserdem benutzt malvorlagen.de.ms für die Verteilung des Dialers eine externe Abrechnungsfirma (http://download.service-url.de/) welche ausschliesslich auf Gewinnbeteiligungsbasis Dialer für andere Seiten, wie eben auch die Malbuchseite, zum Download anbietet. So ergeben sich hier verschiedene Möglichkeiten für die Eindämmung dieses Problems.

-Man könnte eine Software entwickeln, welche Einwahlmasken, wie die oben gezeigte als solche identifiziert. Somit würde man automatisch viele der Dialer-Anbieter erkennen können.

-Daraufhin könnte man alle Dokumente, die auf eine entsprechende Dialer Downloadseite verlinken automatisiert aus dem Index entfernen und bei weiteren Crawlingvorgängen nicht erneut aufnehmen.

-In Kombination mit Idee #1 könnte hier einem grossen Teil der Dialer Anbieter das Handwerk gelegt werden.

5. Fazit

Zusamenfassend kann gesagt werden, dass die Ideen, welche Google gross gemacht haben das Internet und die Informationsverarbeitung mit Sicherheit positiv beeinflusst und verändert haben. Ich kann mich noch gut an die Faszination der ersten Google-Tage erinnern. Zu dieser Zeit fand man tatsächlich fast immer genau, was man suchte. Doch leider haben maschinell automatisierte Methoden den Nachteil von der „Gegenseite" erkannt und umgangen werden zu können. Dies zeigt sich auch schon seit Jahrzehnten bei der zur Zeit wieder sehr aktuellen Kopierschutzproblematik von Software. Es ist immer nur eine Frage der Zeit, bis auch noch so perfekt und trickreich geschütze Software „geknackt" und wieder frei kopierbar aus einschlägigen Kanälen zu beziehen ist. Meiner Einschätzung nach können automatisierte Vorgänge, solange es noch keine funktionierende künstliche Intelligenz gibt, nicht menschliche Arbeit leisten. Dennoch kann die Technologie in Verbindung mit minimaler menschlicher Geistesleistung sehr viel bessere Ergebnisse erzielen. Google hatte lange die Einstellung vertreten, dass die Aufgabe das Wissen der Welt zu archivieren und zugänglich zu machen nur auf komplett maschinellem Wege möglich wäre. Ich gehe davon aus, dass Google und andere Suchmaschinenbetreiber noch für eine lange Zeit nicht darum herum kommen werden meine und ähnliche Ideen einzusetzen um nicht in einem automatisch generierten Informationswust anderer Maschinen unterzugehen.

6. Quellennachweis

Cho J. , Garcia-Molina H. , Page L.

Efficient Crawling Through URL Ordering

Department of Computer Science

Stanford, CA 94305

http://www.csd.uch.gr/~hy558/papers/cho-order.pdf

Hypertext Transfer Protocol -- HTTP/1.1

Network Working Group

Juni 1999

http://www.w3.org/Protocols/rfc2616/rfc2616.html

GNU telnet - user interface to the TELNET protocol manual

GNU Software Foundation

http://www.mcsr.olemiss.edu/cgi-bin/man-cgi?telnet

GNU grep – manual for the GNU grep command

GNU Software Foundation

http://www.rt.com/man/egrep.1.html

Google – define:crawler

Google's definitions for the term „crawler"

http://www.google.de/search?hl=de&q=define%3Acrawler&btnG=Google-Suche&meta=

Googlebot: Google's Web Crawler

Google's FAQ-Pages for it's crawler „Googlebot"

http://www.google.com/bot.html