6. Suchen im Web

FiXme Note: Generische Einführung zur Websuche.

In diesem Kapitel behandeln wir, wie man das Internet nach Schlagworten durchsucht. Als Nutzer einer Suchmaschine geben wir ein paar Wörter wie „Katzen Photos“ in ein Eingabefeld ein und wollen eine Liste von Webseiten erhalten, die sowohl das Wort „Katzen“ als auch das Wort „Photos“ enthalten.

Außerdem möchten wir, dass die Liste der Webseiten so geordnet ist, dass *wichtige* Ergebnisse vorne in der Liste stehen. Wie genau wir dabei *wichtig* definieren, ist eine interessante Frage, auf die wir im Detail eingehen werden.

Wir verlangen kein Textverständnis von unserer Suchmaschine, die Anfrage „Wie weit ist es von der Erde zum Mond?“ liefert also auch nur Webseiten, die gerade diese Wörter enthalten,[1] und auch Tippfehler korrigiert unsere Suchmaschine nicht.

FiXme Note: Etwas über den Trend hin zu semantischem Suchen.

6.1. Webseiten Finden

Es ist natürlich nicht praktikabel bei jeder Suchanfrage das ganze Internet zu durchforsten. Die erste Aufgabe ist also die Erstellung einer Art Index des Internets, in dem wir leicht nachschlagen können, welche Webseiten welche Wörter enthalten.

Dazu laden wir uns als erstes eine Kopie von jeder Webseite herunter. Wir benutzen einfache Programme, genannt *Crawler* (von engl. *crawl*, kriechen oder krabbeln) um alle Webseiten zu besuchen. Dem Crawler gibt man eine oder mehrere Webseiten vor, wo sie loslegen sollen, zum Beispiel wikipedia.org. Der Crawler lädt die Webseite herunter und schaut sich dann an, auf welche andere Webseiten von wikipedia.org aus verlinkt werden.[2] Zum Beispiel findet der Crawler einen Link zu de.wikipedia.org, der deutschen Wikipedia. Diesem Link folgt er einfach und verfährt dann mit de.wikipedia.org wie zuvor mit wikipedia.org. Wenn wir unsere Crawler lange genug arbeiten lassen, erhalten

[1] Google versteht diese Frage mittlerweile und liefert gleich die richtige Antwort, 384400km.

[2] Das war zur Zeit der ersten Webcrawler in den 90’er Jahren noch deutlich einfacher als heute. Mittlerweile bestehen viele Webseiten zu großen Teilen aus dynamischen Inhalten und welche Links vorhanden sind hängt vom Verhalten des Nutzers auf der Seite ab.

wir so eine Kopie des gesamten erreichbaren Internets. Seiten, die nirgends verlinkt sind, oder für deren Zugriff man ein Passwort benötigt, speichern wir aber natürlich nicht.[3]

Zur Vereinfachung nehmen wir an, dass alle Webseiten durchnummeriert sind. Man kann sie zum Beispiel nach ihrer Adresse sortieren und dann die Position in der sortierten Liste nehmen. Für unsere Zwecke ist es aber am besten, wenn man die Webseiten nach dem mysteriösen „Wichtigkeitsmaß“ sortiert, auf das wir erst im nächsten Abschnitt genauer eingehen. Dann entspricht die Zahl, die wir einer Webseite zuordnen, genau ihrem Wichtigkeitsrang.

Jetzt müssen wir die diese Kopie des Internets zu einem Index umbauen. Für jedes Wort möchten wir wissen, auf welchen Webseiten es vorkommt. Wenn wir uns die Webseiten ansehen, wissen wir, welches Wort zu jeder Webseite gehört. Wir möchten jetzt alle Webseiten nach Worten gruppieren.

Ein erster Ansatz wäre es, für jedes Wort eine Liste anzulegen, dann über alle Webseiten zu laufen und i an die Liste des Wortes w anzuhängen, wenn das Wort w auf der Webseite i vorkommt.[4]

Das funktioniert gut, solange der Index noch in den Speicher eines einzelnen Computers passt. Man kann sich aber leicht vorstellen, dass der Index des Internets dafür zu groß ist. In diesem Fall werden Speicherzugriffe sehr teuer, weil man womöglich mit einem entfernten Rechner kommunizieren muss um an die Liste des Wortes w heranzukommen. Was man mit großen Datenmengen aber sehr effizient machen kann ist sie zu sortieren (man benutzt Sortieren durch Mischen, siehe 3.2.2 auf Seite 56). Versuchen wir also das Problem der Indexerstellung auf ein Sortierproblem zurückzuführen.

Unser Problem ist weiterhin, dass wir von den gespeicherten Webseiten wissen, welche Worte auf jeder Webseite vorkommen und jetzt die Webseiten gruppieren müssen. Gleiche Dinge zusammen zu führen kann man aber leicht durch Sortieren bewerkstelligen. Für die Webseite i erstellen wir für jedes Wort w, das auf i vorkommt, ein Paar (w, i). Wenn wir das für alle Webseiten gemacht haben, haben wir eine riesige Liste solcher (*Wort*, *Seite*) Paare, die wir jetzt einfach sortieren. Das gruppiert alle Paare mit dem gleichen Wort, so dass wir nur einmal über die sortierte Liste rüber gehen müssen, um für jedes Wort die Liste der Webseiten zu erhalten, die es enthalten. Es ist wichtig, dass die Liste der Webseiten durch diesen Prozess automatisch auch sortiert ist. Durch die Wahl der Nummerierung der Webseiten, stehen also in jeder Liste die wichtigsten Webseiten zuerst.

Anmerkung: Da das Internet sehr viele Webseiten hat, circa 4 Milliarden[5], muss man sich

[3] Außerdem können Webseitenbetreiber über eine spezielle Textdatei, `robots.txt`, steuern, ob sie indiziert werden möchten oder nicht. Crawler schauen sich per Konvention, ob die Webseite eine `robots.txt` hat und stellen an Hand ihres Inhalts fest, ob die Seite indiziert werden darf.

[4] Damit man leicht die zu einem Wort w gehörige Liste findet, ist es hilfreich die Wörter auch durchzunummerieren.

[5] Die Webseite worldwidewebsize.com schätzt die Größe des durchsuchbaren Internets an Hand der Anzahl der Suchergebnisse ab.

fragen, ob es überhaupt praktikabel ist, einen Index des Internets abzuspeichern. Die deutsche Sprache enthält laut Duden etwa eine halbe Millionen Wörter. Wenn wir also annehmen, dass wir insgesamt etwa 10^7 Schlagworte haben und jedes Wort auf etwa 10^7 Webseiten steht, müssen wir etwa 10^{14} Zahlen speichern können. Einen normalen Desktop Computer kann man heutzutage leicht mit einem Terabyte Speicher ausstatten, das entspricht $2.5 \cdot 10^{11}$ Zahlen. Demnach reichen schon 400 solche Rechner aus, um den ganzen Index zu speichern.

Wenn wir den Index vorberechnet haben, ist es sehr leicht Suchanfragen zu beantworten, die nur ein einziges Wort enthalten. Man muss einfach im Index nachschlagen, auf welchen Webseiten das Wort vorkommt; die Liste ist automatisch richtig sortiert und man kann sie dem Nutzer einfach geben. Zum Nachschlagen benutzt man natürlich Binärsuche (vgl. 3.1.2 auf Seite 49), weil der Index sortiert ist.

Spannender ist es, Suchanfragen zu beantworten, die mehrere Worte $w_1, \ldots, w_n$ enthalten. Dazu schlägt im Index die Listen $\ell_1, \ldots, \ell_n$ aller Worte nach und muss sie nach Webseiten durchsuchen, die in allen Listen vorkommen. Die Antwort ist also $\ell_1 \cap \ldots \cap \ell_n$. In einer Übungsaufgabe werden wir diskutieren, wie man das effizient macht, auch wenn die Listen sehr lang sind.

6.2. Webseiten nach Relevanz Sortieren

Die Sortierung nach Wichtigkeit ist der eigentliche Beitrag der Suchmaschinen für das World Wide Web. Für praktisch jede Anfrage gibt es tausende oder hunderttausende Webseiten, die alle Schlagworte enthalten, es ist unmöglich für einen Menschen sich alle anzusehen, um die am besten passende Seite zu finden.

Webseiten nach Relevanz zu sortieren scheint ein unglaublich schwieriges Problem zu sein, das viel Textverständnis voraussetzt. Computer haben auch heute noch große Probleme damit Texte semantisch zu verstehen, ist es also hoffnungslos Webseiten nach ihrer Relevanz sortieren zu wollen?

Die ersten Suchmaschinen haben Menschen eingesetzt, um Webseiten nach Kategorien zu ordnen und ihre Wichtigkeit zu bestimmen. Das hat mit dem rasanten Wachstum des Internets zunehmend schlechter funktioniert. Die Gründer von Google, Sergey Brin und Larry Page, hatten eine revolutionäre Einsicht, die eine Automatisierung dieses Vorgangs erlaubt hat.

Woher weiß man, ob ein Mensch „wichtig“ oder „unwichtig“ ist? Ein Mensch ist genau dann wichtig, wenn wichtige Menschen ihn für wichtig halten! Das gleiche Prinzip kann man auch auf Webseiten anwenden. Eine Webseite ist dann wichtig, wenn andere wichtige Webseiten sie für wichtig halten. Im Folgenden werden wir uns ansehen, wie man aus dieser zirkulären Definition etwas konkretes herleiten kann.

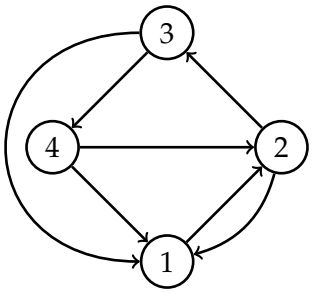

Abbildung 6.1.: Ein gerichteter Graph auf vier Webseiten.

Für Webseiten werden wir Links benutzen, um ihre „Meinung“ über andere Webseiten festzustellen. Wenn Webseite A auf Webseite B verweist, dann hält Webseite A Webseite B für wichtig. Wir erhalten so einen großen gerichteten Graphen, der alle Webseiten als Knoten enthält, mit einer gerichteten Kante $A \to B$, wann immer eine Webseite A auf eine Webseite B verweist. Abbildung 6.1 zeigt ein Beispiel.

Jetzt müssen wir uns überlegen, wie wir aus einem Graphen wie Abbildung 6.1 die Wichtigkeit einer Webseite ablesen können. Wir denken vom Ergebnis her und nehmen an, wir wüssten schon für jede Webseite A ihre Wichtigkeit b_A. Zunächst legen wir fest, wie die Wichtigkeit weitergegeben wird. Am einfachsten ist es, wenn jede Webseite die gleiche Menge Wichtigkeit an jeden ihrer Nachbarn weitergibt. Wenn eine Webseite A also auf fünf andere Webseiten verweist, dann erhält jede von ihnen $b_A/5$ Wichtigkeitseinheiten von A. Für die Zahl der ausgehenden Links einer Webseite A schreiben wir $\mathrm{out}(A)$.

Woher kommt aber nun b_A? Gerade von den eingehenden Links nach A! Wir summieren die Wichtigkeit der Webseiten, die auf A verweisen, geteilt durch die Zahl ihrer ausgehenden Verweise auf. Es gilt

$$b_A = \sum_{\text{alle auf A zeigende Webseiten } B_i} \frac{b_{B_i}}{\mathrm{out}(B_i)}$$

Wir erhalten so eine Gleichung für die Wichtigkeit jeder Webseite. Für den Webgraphen aus Abbildung 6.1 sieht das Gleichungssystem wie folgt aus.

$$b_1 = \frac{b_4}{2} + \frac{b_3}{2} + \frac{b_2}{2} \tag{6.1}$$

$$b_2 = b_1 + \frac{b_4}{2} \tag{6.2}$$

$$b_3 = \frac{b_2}{2} \tag{6.3}$$

$$b_4 = \frac{b_3}{2} \tag{6.4}$$

Ein solches System von Gleichungen kann man lösen, zum Beispiel mit dem Algorithmus von Gauss. Damit die Lösung eindeutig ist, müssen wir die Gesamtwichtigkeit aller Webseiten festlegen, sonst kann man in einer Lösung jede Wichtigkeit mit dem Gleichung Faktor multiplizieren um eine weitere Lösung zu erhalten. Normieren wir die Gesamtwichtigkeit also auf 1 durch Hinzufügen einer weiteren Gleichung

$$1 = b_1 + b_2 + b_3 + b_4 \tag{6.5}$$

Für das Beispiel hat das Gleichungssystem die Lösung

$$b_1 = \frac{7}{21} \approx 0.33 \quad b_2 = \frac{8}{21} \approx 0.38 \quad b_3 = \frac{4}{21} \approx 0.19 \quad b_4 = \frac{2}{21} \approx 0.1,$$

da gilt

$$b_4 \overset{(6.4)}{=} \frac{b_3}{2} \overset{(6.3)}{=} \frac{b_2}{4} \overset{(6.2)}{=} \frac{b_1}{4} + \frac{b_4}{8}$$

und somit

$$b_1 = \frac{7}{2} b_4 \quad b_2 = 4b_4 \quad b_3 = 2b_4.$$

Einsetzen in (6.5) ergibt dann $b_4 = 2/21$, womit die anderen Variablen ausgerechnet werden können.

Wie man aber schon an diesem kleinen Beispiel sieht, ist das Lösen von Gleichungssystemen eine relativ komplizierte Aufgabe. Die besten Algorithmen brauchen $O(n^{2.38})$ Operationen um ein System mit n Variablen zu lösen. Wenn n die Anzahl der Webseiten im Internet ist, sieht man leicht, dass dieser Ansatz viel zu aufwendig ist.

Webseiten sind wichtiger, wenn sie viele eingehende Links haben. Umsomehr, wenn die auf sie verweisenden Seiten ihrerseits viele eingehende Links haben, und nur wenige ausgehend. Das legt die Vermutung nahe, dass man sich, wenn man den Links zufällig folgend über den Webgraphen läuft, häufiger in wichtigen Webseiten wiederfindet als in unwichtigen. Der Fachbegriff für diese Art über Graphen zu laufen ist *Irrfahrt*. In Abbildung 6.2 haben wir eine solche Irrfahrt auf dem Webgraphen aus Abbildung 6.1 durchgeführt und uns in jedem Schritt notiert, wie oft wir eine Webseite schon besucht haben. Nach 99 Schritten haben wir folgende Besuchsstatistik:

Webseite 1	Webseite 2	Webseite 3	Webseite 4
33	39	18	9

Zumindest für dieses Beispiel scheint unsere Intuition richtig gewesen zu sein. Je länger unsere Irrfahrt andauert, desto mehr nähert sich die normierte Besuchszahl einer Webseite ihrer Wichtigkeit an. Tatsächlich kann beweisen, dass die Besuchsstatistik einer Irrfahrt sich immer schnell der Wichtigkeit der Knoten annähert.

Man kann also dich Wichtigkeit von Webseiten durch eine Irrfahrt über den Webgraphen bestimmen. Natürlich muss man sich nicht auf nur einen Irrfahrer beschränken, man

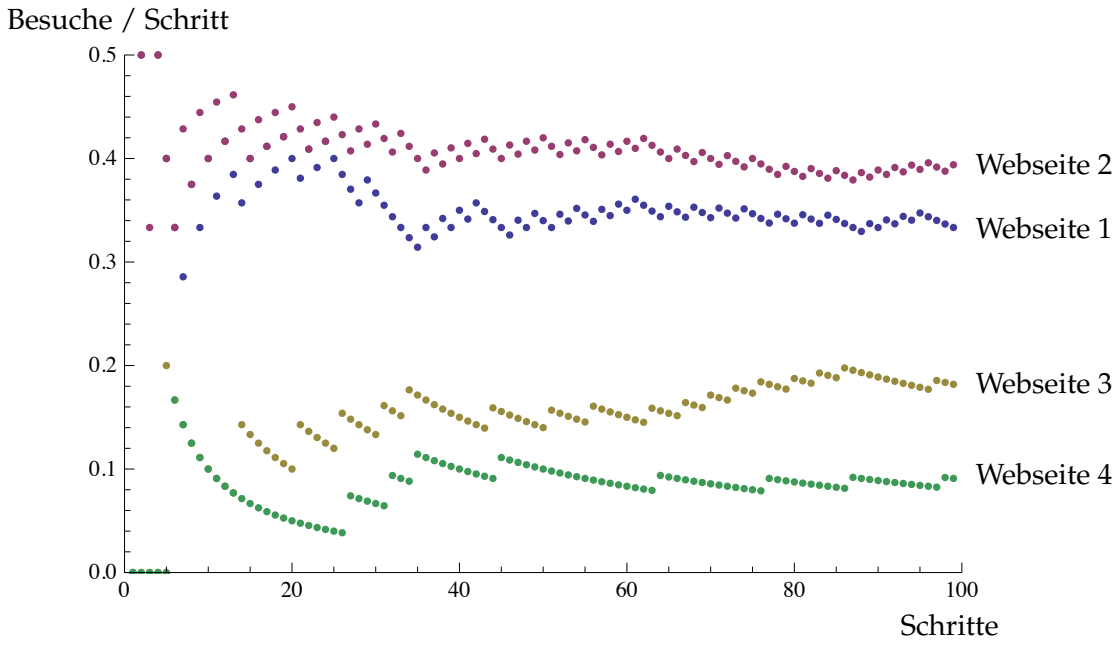

Abbildung 6.2.: Eine Irrfahrt auf dem Webgraphen aus Abbildung 6.1. Auf der y-Achse ist abgebildet, wie oft eine Webseite besucht wurde, geteilt durch die Anzahl der Schritte, die wir schon gemacht haben.

kann mit vielen Computern gleichzeitig viele Irrfahrten machen und so sehr schnell genaue Besuchsstatistiken erstellen. Die Besuchsstatistik näher sich noch schneller der Wichtigkeit an, wenn man gelegentlich die Irrfahrt abbricht und zu einer zufällig gewählten Webseite springt.

Dieses Verfahren funktioniert so gut, dass man auf ein aufwendiges Lösen von Gleichungssystemen verzichten kann. Durch die Möglichkeit viele Irrfahrten gleichzeitig zu machen, ist es auch für sehr große Graphen geeignet.