Websuche

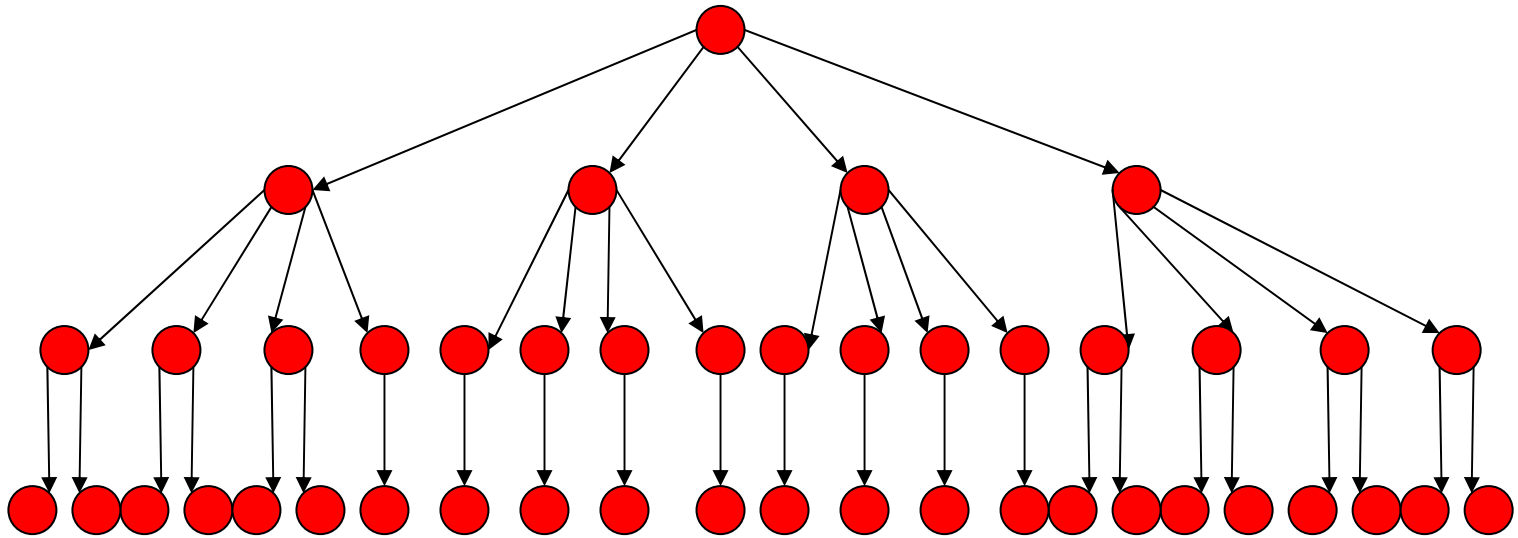
Spidering

Spiders (Roboters/Bots/Crawlers)

- Beginne mit einer umfassenden Menge von Start-URLs, von denen aus die Suche zu beginnen ist.
- Folge rekursiv allen Links auf diesen Seiten, um weitere Seiten zu finden.
- Füge alle **neu** gefundenen Seiten zum invertierten Index hinzu.
- Benutzer können ggf. selbst Seiten zur Indizierung und/oder als Start-URLs anmelden.

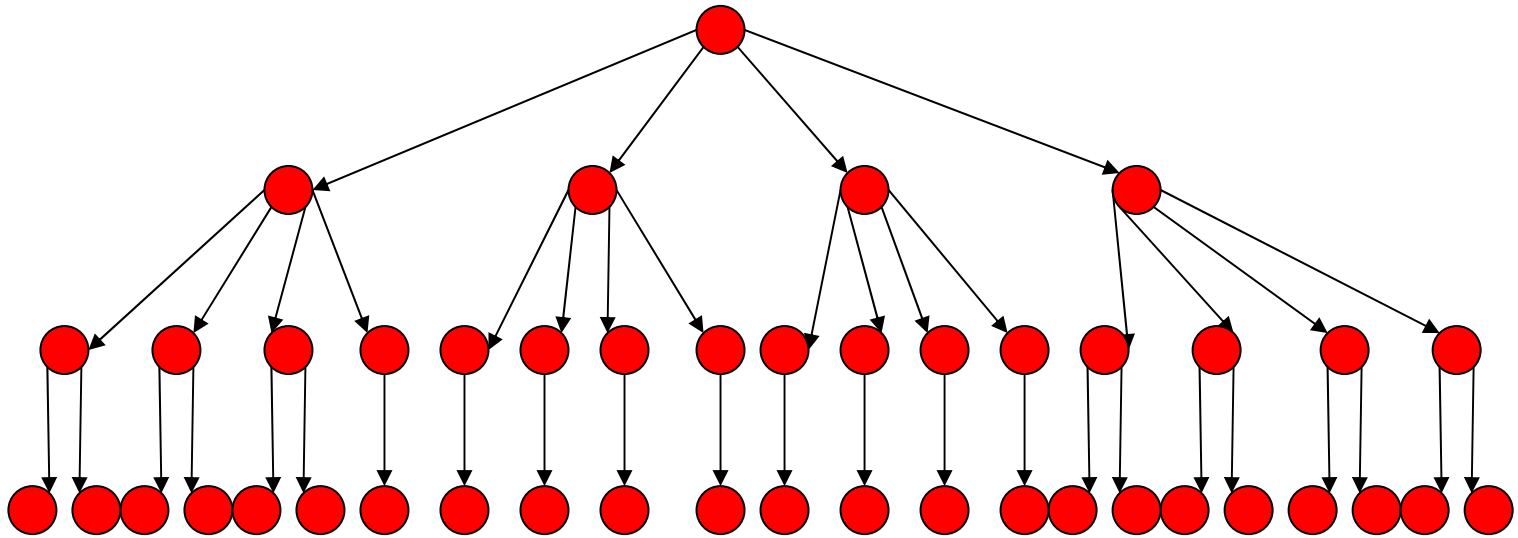
Suchstrategien

Breitensuche



Forts. Suchstrategien

Tiefensuche



Vor-/Nachteile der Suchstrategien

- Die Breitensuche sammelt jeweils alle Knoten, die gleich weit von der Ursprungsseite entfernt sind.
  - Erfordert Speicherung aller Knoten der vorhergehenden Ebene, d.h. der Speicherbedarf wächst exponentiell mit der Tiefe.
  - Dies ist der Standard-Crawling-Ansatz.
- Die Tiefensuche erfordert nur die Speicherung der Knoten ab der letzten Verzweigung, d.h. ist linear in der Tiefe.
  - Verfahren geht aber bei der Verfolgung eines einzigen Threads “verloren”.
- Beide Strategien können mit einer Warteschlange für URLs implementiert werden.

Vermeidung von Seiten-Duplikaten

- Man muss beim erneuten Besuch einer Seite erkennen, dass sie bereits besucht wurde, denn das Web ist ein Graph und kein Baum.
- Man muss besuchte Seiten effizient indizieren, um einen schnellen Widererkennungstest zu ermöglichen.
  - Baumindexierung (z.B. Trie)
  - Hashtabelle
- Indiziere Seiten mit URL als Schlüssel.
  - URLs müssen normalisiert werden (z.B. “/”-Endung entfernen)
  - Kein Erkennen duplizierter oder gespiegelter Seiten.
- Indiziere Seite mit textlichem Inhalt als Schlüssel.
  - Erfordert zunächst das Herunterladen der Seite.

Spider-Algorithmus

Initialisiere eine Warteschlange (Q) mit der Menge der bekannten URL's.
Bis Q leer oder das Seiten- bzw. Zeitlimit erschöpft ist:
    Hole URL *L* vom Anfang von *Q*.
    Wenn *L* keine HTML-Seite ist (.gif, .jpeg, .ps, .pdf, .ppt, etc.)
        gehe zum Schleifenanfang.
    Wenn *L* bereits besucht wurde,
        gehe zum Schleifenanfang.
    Lade Seite *P* mit URL *L* runter.
    Wenn *P* nicht runtergeladen werden kann (z.B. 404 Fehler, Roboter ausgeschlossen),
        gehe zum Schleifenanfang.
    Indiziere *P* (z.B. zum invertierten Index hinzufügen oder
    speichere Zwischenkopie).
    Analysiere *P*, um eine Liste neuer Links *N* zu erhalten.
    Füge *N* an das Ende von *Q* an.

Warteschlangen-Strategien

- Die Art des Anfügens von *N* an *Q* bestimmt die Suchstrategie.
- FIFO (*N* an das Ende von *Q* angefügt) ermöglicht Breitensuche.
- LIFO (*N* an den Anfang von *Q* angefügt) ermöglicht Tiefensuche.
- Das Anordnen von *Q* nach einer Heuristik ermöglicht einen“fokussierten Crawler”, der seine Suche auf “interessante” Seiten fokussiert.

Einschränken des Crawlens

- Begrenze das Spidern auf eine bestimmte Site.
  - Entferne Links anderer Sites von *Q*.
- Begrenze das Spidern auf ein bestimmtes Verzeichnis.
  - Entferne Links, die nicht im spezifizierten Verzeichnis vorhanden sind.
- Befolge die Beschränkungen von Seiteninhabern (Crawler-Ausschluss).

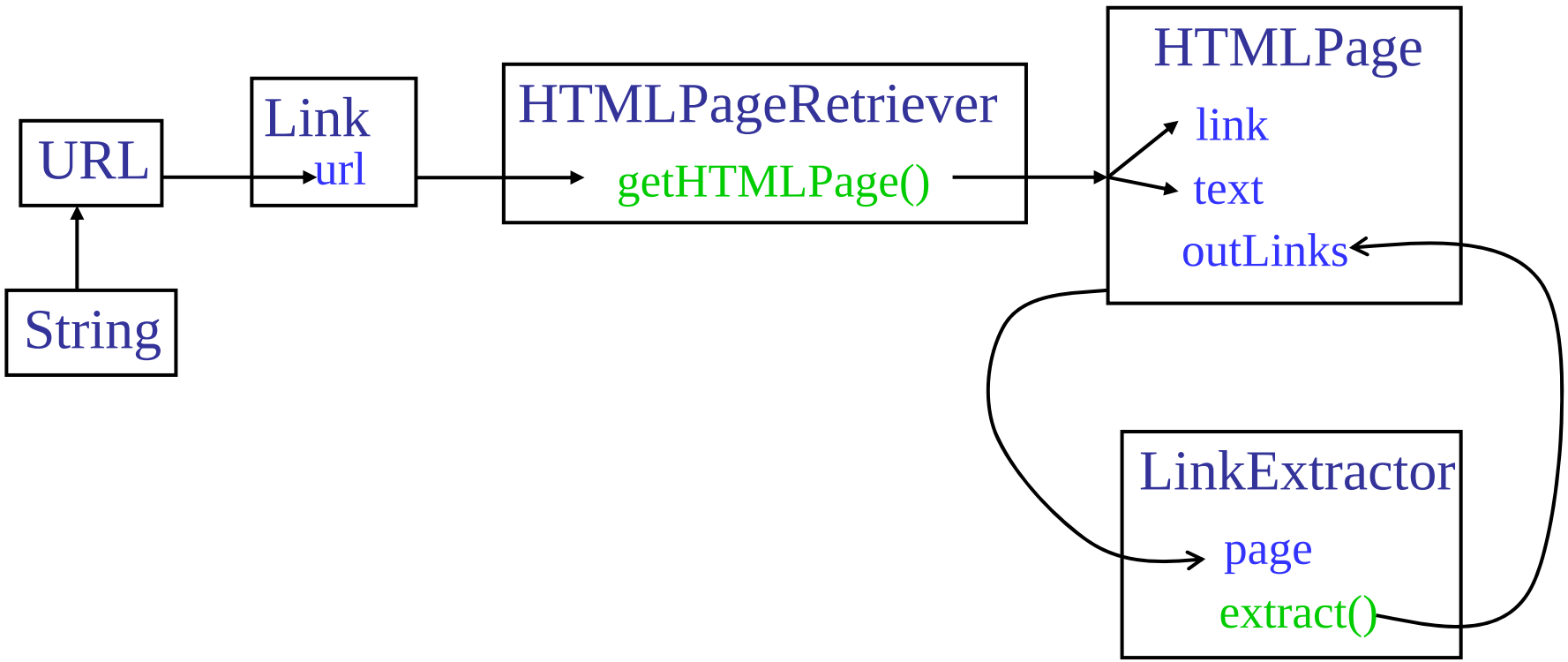
Linkextraktion

- Muss alle Links auf einer Seite finden und die URLs extrahieren.
  - <a href="http://www.cs.utexas.edu/users/mooney/ir-course">
  - <frame src="site-index.html">
- Muss relative URLs vervollständigen unter Verwendung der URL der aktuellen Seite:
  - <a href="proj3"> wird zu http://www.cs.utexas.edu/users/mooney/ir-course/proj3
  - <a href="../cs343/syllabus.html"> wird zu http://www.cs.utexas.edu/users/mooney/cs343/syllabus.html

URL-Syntax

- Eine URL hat die folgende Syntax:
  - <Schema>://<Autorität><Weg>?<Anfrage>#<Fragment>
- Eine *Autorität* hat die Syntax:
  - <Host>:<Port-Nummer>
- Eine *Anfrage* durchläuft variable Werte in HTML- Form und hat die Syntax:
  - <Variable>=<Wert>&<Variable>=<Wert>…
- Ein *Fragment* wird auch als eine *Referenz* oder ein *ref* bezeichnet und ist ein Pointer innerhalb des Dokuments auf einen Punkt, der durch einen Anker-Tag folgender Form spezifiziert ist:
  - <A NAME="<Fragment>">

Java-Klassen Spider



Link-Normalisierung

- Äquivalente Verzeichnis-Endungen werden durch Entfernung des End-Schrägstrichs normalisiert:
  - http://www.cs.utexas.edu/users/mooney/
  - http://www.cs.utexas.edu/users/mooney
- Interne Seitenfragmente (ref's) werden entfernt:
  - http://www.cs.utexas.edu/users/mooney/welcome.html#courses
  - http://www.cs.utexas.edu/users/mooney/welcome.html

Link-Extraktion in Java

- Java Swing enthält einen HTML-Parser. (Alternativen sind JTidy und NekoHTML.)
- Der Swing-Parser verwendet “call-back”-Methoden.
- Parser bekommt ein Objekt überreicht, dass diese Methoden hat:
  - HandleText(char[] text, int position)
  - HandleStartTag(HTML.Tag tag, MutableAttributeSet attributes, int position)
  - HandleEndTag(HTML.Tag tag, int position)
  - HandleSimpleTag (HTML.Tag tag, MutableAttributeSet attributes, int position)
- Wenn der Parser auf ein Tag oder einen eingebetteten Text trifft, wird die für dieses Objekt geeignete Methode aufgerufen.

Forts. Link-Extraktion in Java

- Nehme im HandleStartTag, falls es ein “A”-Tag ist, den HREF-Attributwert als ursprüngliche URL.
- Vervollständige die URL durch Verwendung der Basis-URL:
  - new URL(URL baseURL, String relativeURL)
  - Scheitert, falls Basis-URL in einem Verzeichnisnamen endet, aber dies wird nicht durch ein abschließendes “/” angezeigt.
  - Füge ein “/” zur Basis-URL, wenn diese nicht in einem Dateinamen mit Erweiterung endet (und demzufolge vermutlich ein Verzeichnis ist).

Zwischenspeichern mit Basis-URL

- Speichere die Kopie einer Seite in einem lokalen Verzeichnis für das eventuelle Indexieren zwecks Retrieval.
- BASE-Tag im Kopfbereich einer HTML Datei verändert die Basis-URL für alle relativen Pointer:
  - <BASE HREF="<base-URL>">
- Dies ist ein spezielles HTML-Konstrukt zur Verwendung in Dokumenten, die von ihrem ursprünglichen Ort entfernt wurden.

Ankertext-Indizierung

- Extrahiere Ankertext (zwischen <a> und) jedes verfolgten Links.
- Ankertext ist gewöhnlich beschreibend für das Dokument, auf den er verweist.
- Füge Ankertext zum Inhalt der Bestimmungsseite hinzu, um ggf. neue Schlüsselwörter zu erhalten.
- Wird von Google verwendet:
  - Evil Empire
  - IBM

Forts. Ankertext-Indizierung

- Hilft, wenn der beschreibende Text in einer Bestimmungsseite eher in Bildern/Logos als in zugänglichen Text eingebettet ist.
- Oftmals ist Ankertext nicht nützlich:
  - “klicke hier”
- Vergrößert die Beschreibung insbes. für beliebte Seiten mit vielen eingehenden Links, führt zu erhöhtem Recall bei diesen Seiten.
- Man kann sogar dem Ankertext höhere Gewichte geben als dem Inhalt der Originalseite, da er schwerer zu manipulieren ist.

Roboter-Ausschluss

- Webseiten und Seiten können spezifizieren, dass Roboter gewisse Bereiche nicht crawlen/indizieren sollen.
- Zwei Komponenten:
  - Roboter-Ausschluss-Protokoll: Seitenweise Spezifizierung ausgeschlossener Verzeichnisse.
  - Roboter META Tag: Individueller Dokument-Tag zum Ausschluss bei der Indexierung oder der Weiterverfolgung von Links.

Roboter-Ausschluss-Protokoll

- Site-Administrator erstellt eine “robots.txt”-Datei an der Wurzel des HostWeb-Verzeichnisses.
  - http://www.ebay.com/robots.txt
  - http://www.cnn.com/robots.txt
- Datei enthält eine Liste aller für einen gegebenen Robot ausgeschlossenen Verzeichnisse.
  - Schließt alle Bots von der kompletten Site aus:

```
User-agent: *
Disallow: /
```

Roboter-Auschluss-Protokoll: Beispiele

- Schließt spezifische Verzeichnisse aus:

```
User-agent: *
Disallow: /tmp/
Disallow: /cgi-bin/
Disallow: /users/paranoid/
```

- Schließt einen speziellen Bot aus:

```
User-agent: GoogleBot
Disallow: /
```

- Läßt einen speziellen Bot zu:

```
User-agent: GoogleBot
Disallow:

User-agent: *
Disallow: /
```

Roboter-Ausschluss-Protokoll: Details

- Verwende nur Leerzeilen, um für verschiedene Benutzeragenten die jeweils nicht erlaubten Verzeichnisse zu trennen.
- Ein Verzeichnis pro “Disallow”-Zeile.
- Keine regulären Ausdrücke in Verzeichnisnamen erlaubt.

Roboter-META-Tag

- Schließe META-Tag in den HEAD-Bereich eines spezifischen HTML-Dokuments ein.
  - <meta name="robots" content="none">
- Inhalt ist eine Wertepaar für zwei Dimensionen:
  - index | noindex: Zulassen/nicht Zulassen der Indexierung dieser Seite.
  - follow | nofollow: Verfolgen/nicht Verfolgen der Links auf dieser Seite.

Forts. Roboter-META-Tag

- Spezielle Werte:
  - all = index,follow
  - none = noindex,nofollow
- Beispiele:

<meta name="robots" content="noindex,follow">

<meta name="robots" content="index,nofollow">

<meta name="robots" content="none">

Roboter-Ausschluss: Diskussion

- META Tag ist neuer und weniger gut angepasst als “robots.txt”.
- Standards sind Konventionen, die von “guten Robotern” befolgt werden.
- Firmen sind für die “Nichtbeachtung” dieser Konventionen und das “unerlaubte Betreten” von privatem virtuellem Raum belangt worden.
- “Gute Roboter” versuchen auch, individuelle Sites nicht mit vielen, schnellen Anfragen zu “bombardieren”.
  - keine “Denial of Service”-Angriffe.

Multi-Threaded Spidering

- Engpass ist die Netzwerkverzögerung beim Herunterladen von individuellen Seiten.
- Das beste ist, multiple Threads parallel laufen zu lassen, wobei jeder eine Seite von einem anderen Host anfordert.
- Verteile URLs auf Threads, um die gerechte Verteilung von Anfragen über die verschiedenen Hosts zu gewährleisten, den Durchsatz zu maximieren und die Überlastung einzelner Server zu vermeiden.
- Frühe Google-Spider hatten zahlreiche koordinierte Crawler mit je ca. 300 Threads, die zusammen in der Lage waren, mehr als 100 Seiten pro Sekunde herunterzuladen.

Geführtes/fokussiertes Spidering

- Sortiere die Warteschlange, um zunächst mehr “interessante” Seiten zu erforschen.
- Zwei Arten von Fokussierung:
  - Themen-gerichtet
  - Link-gerichtet

Themen-geleitetes Spidering

- Nehme an, dass die gewünschte Themenbeschreibung oder interessante Musterseiten gegeben sind.
- Sortiere die Linkreihe nach der Ähnlichkeit (z.B. mit Kosinus-Maß) ihrer Ursprungsseiten und/oder Ankertext zu dieser Themenbeschreibung.
- Lädt vorzugsweise Seiten herunter, die sich auf das gegebene Thema beziehen.

Link-gerichtetes Spidering

- Überwache die Links und speichere die Ein- und Ausgrade jeder angetroffenen Seite.
- Sortiere die Warteschlange, um beliebte Seiten mit vielen eingehenden Links vorzuziehen. (*Autoritäten*).
- Sortiere die Warteschlange, um zusammenfassende Seiten mit vielen ausgehenden Links (*hubs*) zu bevorzugen.

Gecrawlte Seiten auf dem neusten Stand halten

- Das Web ist sehr dynamisch: viele neue Seiten, aktualisierte Seiten, entfernte Seiten, etc.
- Periodische Prüfung aller gespiderten Seiten hinsichtlich Aktualisierungen und Beseitigungen:
  - Schau nur auf Kopfinfo (z.B. META-Tags bei der letzten Aktualisierung), um zu bestimmen, ob die Seite sich geändert hat. Lade die gesamte Seite erst wieder, wenn notwendig.
- Verfolge, wie oft jede Seite aktualisiert wurde und kehre vorzugsweise zu Seiten zurück, die dynamischer sind.
- Aktualisiere vorzugsweise Seiten, auf die öfter zugegriffen wird, um die Aktualität der beliebteren Seiten zu optimieren.