

Übung Netzbasierte Informationssysteme

Termin 7: Web Crawling



Prof. Dr. Adrian Paschke

Arbeitsgruppe Corporate Semantic Web (AG-CSW)
Institut für Informatik, Freie Universität Berlin
paschke@inf.fu-berlin.de
<http://www.inf.fu-berlin.de/groups/ag-csw/>

- **Lernziele des 6. Übungsblattes waren:**
 - Einfache Lucene-basierte Web Suche
- **Wie war`s?**
 - Fehlerquellen
 - Probleme?

- Lucene für XML Suche
 - Parsing indexing and searching XML with Digester and Lucene
 - <http://www.ibm.com/developerworks/java/library/j-lucene>
- LuceneXMLIndexer in Apache Cocoon
 - Solr: Indexing XML with Lucene and REST
- Lucene für Bildsuche
 - LuceneIndexTransformer: Beispiel für IMG ALT indexen
- Lucene Ergebnisse besser ranken
 - LuceneBD
 - Full documentation at java/docs/scoring.html

- Übungsblatt 7 ist online auf der Website
- Thema: Web Crawling
- Aufgabe: Web Crawl durchführen und analysieren
- Inhalte heute
 - Rückblick: Crawlers
 - Hintergrund: Crawl Policies
 - Aufgabe: Nutch
 - Aufgabe: Plugins
 - Aufgabe: Link Suche

- A crawler is an *„automated script which browses the World Wide Web in a methodical manner“*
- The process of Web crawling has many uses
 - It is used by search engines as a means to acquire up-to-date Web data; a copy of all visited pages is stored for later indexing and faster search by use of a cache
 - Also for automating maintenance tasks on a website such as checking links or validating HTML code
 - It can be used for gathering specific types of information from Web pages such as harvesting e-mail addresses (spambots)

There are three important characteristics of the Web that generate a scenario in which Web crawling is very difficult:

- its large volume,
- its fast rate of change, and
- dynamic page generation

- Starting point is an URL or list of URLs – the **seed**
- As the crawler visits these URLs, it identifies all hyperlinks in the page and adds them to a list of URLs to visit – the **crawl frontier**
- URLs from the crawl frontier are recursively visited according to a set of policies

Types of Crawl Behaviour

Four types of policies

- A *selection policy* that states which pages to download.
- A *re-visit policy* that states when to check for changes to the pages.
- A *politeness policy* that states how to avoid overloading websites.
- A *parallelization policy* that states how to coordinate distributed web crawlers.

- Requires a metric of importance for prioritizing web pages
- Importance can be measured as a function over characteristics of a web page
 - Its intrinsic quality
 - Its popularity (in terms of links or visits)
 - Its URL (in terms of domain or path)
- Selection may be restricted on the basis of:
 - MIME Type, either by HTTP GET or checking the suffix
 - Dynamic produced pages (with ? in the URL)
- **Focused crawling** (or topical crawling)
 - Measures relevance in terms of similarity of the text of a page to a query
 - To predict relevance prior to selection, one strategy is to use the anchor text on the page

Re-visit policy

- There is a cost associated with not detecting a change in a web page that has been crawled
 - Freshness (whether the local copy is accurate or not)

$$F_p(t) = \begin{cases} 1 & \text{if } p \text{ is equal to the local copy at time } t \\ 0 & \text{otherwise} \end{cases}$$

- Age (how outdated the local copy is)

$$A_p(t) = \begin{cases} 0 & \text{if } p \text{ is not modified at time } t \\ t - \text{modification time of } p & \text{otherwise} \end{cases}$$

- Two re-visiting policies can be followed
 - Uniform: re-visiting all pages at same rate of frequency
 - Proportional: re-visiting more often the pages that change more frequently

Re-visit policy (2)

- Which performed better?
 - Uniform! (in terms of average freshness)
 - Why? When a page changes too often, the crawler wastes time trying to re-crawl it while its copy is still not fresh
- We should penalize the pages which change too often
 - Keep freshness high by ignoring pages which change too often
 - Keep age low by raising frequency of re-visit of pages monotonically with the rate of change of those pages
- No perfect re-visiting policy possible
 - Depend on the distribution of page changes
 - Exponential distribution is a good fit

- Crawlers carry a cost for the web site too, should always ask before crawling „May I?“
 - Network resources, as crawlers require considerable bandwidth and operate with a high degree of parallelism during a long period of time.
 - Server overload, especially if the frequency of accesses to a given server is too high.
 - Poorly written crawlers, which can crash servers or routers, or which download pages they cannot handle.
 - Personal crawlers that, if deployed by too many users, can disrupt networks and Web servers.
- The server provides some rules in the robots.txt file
 - Some search engines support an additional „Crawl-delay:“ parameter for the time to wait between requests

- A **parallel crawler** runs multiple crawls in parallel
 - Maximize the download rate
 - Minimize the overhead from parallelization
 - Avoid repeated downloads of the same page
- How are URLs crawled in parallel?
 - Dynamic assignment: a central server allocates URLs to a crawl process ensuring load balance
 - Static assignment: an assignment function is given before the crawl which maps URLs to a crawl process (e.g. a hash function)
- All commercial search engines parallel crawl the Web
 - Grub lets you be a part of this distributed crawling

Grub – P2P distributed crawler



► home log in download news tools stats wiki help

Grubby Lives!

Search is part of the fundamental infrastructure of the Internet. And, it is currently broken.

Why is it broken? It is broken for the same reason that proprietary software is always broken: lack of freedom, lack of community, lack of accountability, lack of transparency. Here, we will start to change all that.

Grub started back in 2000 with a simple concept of distributing part of the search process pipeline: crawling. In a way, we were a bit ahead of our time, but our intention then was what it is now. We want to help fix search.

Now, with the help of [Wikia](#), community members, contributors, and Open Source developers our time has come again. Come be part of something greater. Come help us change the World.

NOTICE: The Windows client is running in **TEST** mode right now, watch the [Grub wiki](#) for updates and to help us port this thing :)

[Sign up](#) today and start helping us fix search.

Start Crawling the Web



Start Using the Results

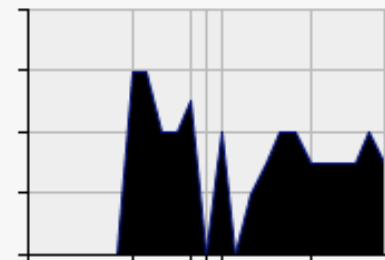


GRUB STATS

► Top 10 Users Today | [More](#)

User	URLS
1 di vin	0.4M
2 max0913	0.3M
3 KAMCOBILL	0.3M
4 godcandy	0.2M
5 generic_toaster	0.2M
6 www.artprojectgroup.com	0.1M
7 www.beeqroup.de	0.1M
8 retrotoys	0.1M
9 Robert Hotchkiss	0.1M
10 miesales	0.1M

► Millions of URLs per Day



- open source web-search software
- builds on [Lucene Java](#), adding web-specifics, such as a crawler, a link-graph database, parsers for HTML and other document formats
- Consists roughly of three components
 - 1. The Crawler, which discovers and retrieves web pages
 - 2. The 'WebDB', a custom database that stores known URLs and fetched page contents
 - 3. The 'Indexer', which dissects pages and builds keyword-based indexes from them

- Requisites from Nutch 0.9.x Tutorial
 - Java, either from [Sun](#) or [IBM](#) on Linux is preferred. Set NUTCH_JAVA_HOME to the root of your JVM installation.
 - Apache's [Tomcat](#)
 - On Win32, [cygwin](#), for shell support. (If you plan to use Subversion on Win32, be sure to select the subversion package when you install, in the "Devel" category.)
- Release at <http://www.apache.org/dyn/closer.cgi/lucene/nutch/>
- Tutorial for Nutch 0.9 at <http://peterpuwang.googlepages.com/NutchGuideForDummies.htm>

Getting Started (Windows)

- For Linux under Windows you can use cygwin. Quick guide:
 - `mount` maps Win32 drives into the local directory tree
 - `cygpath -W` shows you the current Windows directory (expect it to be `C:\WINNT`)
 - Go to the Nutch top level directory e.g. `cd ../nutch-0.9`
 - Try `bin/nutch` to see if it works.
- Make sure `NUTCH_JAVA_HOME` is set
 - In Environmental Variables
 - For me, a relative path to the JDK top level directory worked

```
E /cygdrive/c/nutch-0.8.1
Administrator@PINNE ~
$ mount
C:\cygwin\bin on /usr/bin type system (binmode)
C:\cygwin\lib on /usr/lib type system (binmode)
C:\cygwin on / type system (binmode)
c: on /cygdrive/c type user (binmode,noumount)
d: on /cygdrive/d type user (binmode,noumount)
u: on /cygdrive/u type user (binmode,noumount)
z: on /cygdrive/z type user (binmode,noumount)

Administrator@PINNE ~
$ cygpath -W
/cygdrive/c/WINNT

Administrator@PINNE ~
$ cd ..

Administrator@PINNE /cygdrive/c/Documents and Settings
$ cd ..

Administrator@PINNE /cygdrive/c
$ cd nutch-0.8.1

Administrator@PINNE /cygdrive/c/nutch-0.8.1
$ bin/nutch
Usage: nutch COMMAND
where COMMAND is one of:
  crawl          one-step crawler for intranets
  readdb         read / dump crawl db
  mergedb       merge crawl-db's, with optional filtering
```

Crawling: Configuration

- Configuration for crawl needs three things
 - The **seeds** (starting URL or URLs) are placed in a flat file in a new subdirectory of the Nutch installation
 - Edit the file `conf/crawl-urlfilter.txt` replacing `MY.DOMAIN.NAME` with the name of the domain you wish to crawl. Regular expressions are possible, e.g. all sites in the `apache.org` domain:


```
+^http://([a-z0-9]*\.)*apache.org/
```
 - Edit the file `conf/nutch-site.xml` providing at least the following properties (see the tutorial)


```
http.agent.name
http.agent.description
http.agent.url
http.agent.email
```

Crawling: Execution

- The `crawl` command takes following parameters
 - `-dir dir` names the directory to put the crawl in.
 - `-threads threads` determines the number of threads that will fetch in parallel.
 - `-depth depth` indicates the link depth from the root page that should be crawled.
 - `-topN N` determines the maximum number of pages that will be retrieved at each level up to the depth.

e.g. `crawl urls -dir crawl -depth 4 -topN 100`

- To start, choose shallow depths and a small topN: check that the right pages are being crawled. For a full crawl a depth around 10 is generally appropriate.

Crawling: Execution

```
Administrator@PINNE /cygdrive/c/nutch-0.8.1
$ bin/nutch crawl urls -dir crawl -depth 4 -topN 100
crawl started in: crawl
rootUrlDir = urls
threads = 10
depth = 4
topN = 100
Injector: starting
Injector: crawlDb: crawl/crawlDb
Injector: urlDir: urls
Injector: Converting injected urls to crawl db entries.
Injector: Merging injected urls into crawl db.
Injector: done
Generator: starting
Generator: segment: crawl/segments/20071121102454
Generator: Selecting best-scoring urls due for fetch.
Generator: Partitioning selected urls by host, for politeness.
Generator: done.
Fetcher: starting
Fetcher: segment: crawl/segments/20071121102454
Fetcher: threads: 10
fetching http://citeseer.ist.psu.edu/hunter01adding.html
Fetcher: done
CrawlDb update: starting
CrawlDb update: db: crawl/crawlDb
CrawlDb update: segment: crawl/segments/20071121102454
CrawlDb update: Merging segment data into db.
CrawlDb update: done
Generator: starting
Generator: segment: crawl/segments/20071121102508
Generator: Selecting best-scoring urls due for fetch.
Generator: Partitioning selected urls by host, for politeness.
Generator: done.
Fetcher: starting
Fetcher: segment: crawl/segments/20071121102508
Fetcher: threads: 10
fetching http://citeseer.ist.psu.edu/check/2017607
fetching http://citeseer.ist.psu.edu/coblitz/utility.txt
fetching http://citeseer.ist.psu.edu/context/1122384/0
fetching http://citeseer.ist.psu.edu/729681.html
fetching http://citeseer.ist.psu.edu/update/459015
```

- Simplest way to verify the integrity of your crawl is to launch NutchBean from command line:
`org.apache.nutch.searcher.NutchBean searchterm`
- Now you can place the Nutch WAR file into your servlet container (Tomcat)
 - It is designed to run in the ROOT context
 - It finds the index by default at `.\crawl`
 - Override this by editing `WEB-INF\classes\nutch-site.xml`

```
<property>  
  <name> searcher.dir </name>  
  <value>/path_to_index </value>  
</property>
```



Über

FAQ



Suchen

[Hilfe](#)

[ca](#) | [de](#) | [en](#) | [es](#) | [fi](#) | [fr](#) | [hu](#) | [it](#) | [jp](#) | [ms](#) | [nl](#) | [pl](#) | [pt](#) | [sh](#) | [sr](#) | [sv](#) | [th](#) | [zh](#)



[help](#)

Treffer 1-2 (von insgesamt 45 gefundenen Seiten):

[Adding Multimedia to the Semantic Web - Building an MPEG-7 Ontology - Hunter \(ResearchIndex\)](#)

... author = "J. Hunter", title = "Adding **Multimedia** to the Semantic ... Ontology", text = "Hunter, J.: Adding **Multimedia** to the Semantic ...
<http://citeseer.ist.psu.edu/hunter01adding.html> ([Im Cache](#)) ([Erklärung](#)) ([Referenzen](#)) ([Mehr von citeseer.ist.psu.edu](#))

[Towards Second and Third Generation Web-Based Multimedia - van Ossenbruggen, Geurts, Cornelissen, Ru](#)

... Comparison of two **Multimedia** Presentation Sys.. (context) - Andre 2 ... et al. 1 Device Independent **Multimedia** Authoring (context) - Hardmar Ossenbruggen ...
<http://citeseer.ist.psu.edu/vanossenbruggen01towards.html> ([Im Cache](#)) ([Erklärung](#)) ([Referenzen](#)) ([Mehr von citeseer.ist.psu.edu](#))



[ca](#) | [de](#) | [en](#) | [es](#) | [fi](#) | [fr](#) | [hu](#) | [it](#) | [jp](#) | [ms](#) | [nl](#) | [pl](#) | [pt](#) | [sh](#) | [sr](#) | [sv](#) | [th](#) | [zh](#)

- All of the parsing, indexing and searching that Nutch does is actually accomplished by various plugins
 - Based on plugin system used in Eclipse
 - In the Web app under WEB-INF/classes/plugins
- To use a given plugin
 - edit the conf/nutch-site.xml file, adding the name of the plugin to the list of plugin.includes
- E.g. text and HTML crawling are enabled by default
 - How to add PDF document crawling?
 - The plugin `parse-pdf` handles PDF documents

- A plugin is an extension of an extension point
- The core extension points are defined in the NutchExtensionPoints plugin (see the plugin.xml file)
- A list with details is at <http://wiki.apache.org/nutch/AboutPlugins>
- Each extension point defines an interface which must be implemented by the extension

Plugins: creating your own

- You can check the source code of the plugins distributed with Nutch
 - In src/plugin
- Developing your plugin source you should use your own directory/package structure, e.g. nbi/gruppe0/nutch
- Deploying your plugin you need to create a new directory in the plugins directory with the name of your plugin
 - Containing a plugin.xml file which describes your plugin to Nutch
 - Containing a src directory with your plugin source code

```
<?xml version="1.0" encoding="UTF-8"?>
  <plugin id=„nbi-plugin“
    name=„NBI Übung 7 Plugin“
    version="0.0.1"
    provider-name=„ag-csw.de">

    <runtime>
      <!-- As defined in build.xml this plugin will
      end up bundled as nbi-plugin.jar -->
      <library name=„nbi-plugin.jar">
        <export name="*" />
      </library>
    </runtime>
  </plugin>
```

Plugin.xml (2)

```
<!--
```

The NBI Parser extends the HtmlParseFilter to do something more during the HTML parse -->

```
<extension id=„nbi.gruppe0.nutch.parse.nbi-  
plugin.nbifilter“  
name=„NBI Parser“  
point="org.apache.nutch.parse.HtmlParseFilter">
```

```
  <implementation id=„NBIParser“  
  class=„nbi.gruppe0.nutch.parse.nbi-  
  plugin.NBIFilter"/>
```

```
</extension>
```

```
</plugin>
```

- In the simplest form

```
<?xml version="1.0"?>  
<project name="nbi-plugin" default="jar">  
<import file="../build-plugin.xml"/>  
</project>
```

- Save in the plugin directory

Plugin deployment

- To build the plugin, you need the Ant tool
 - <http://ant.apache.org/>
- In the directory where you find build.xml simply type ant
- To deploy the plugin in Nutch, add this line to src/plugin/build.xml (not your plugins build.xml, nor the build.xml in the Nutch top level directory!)

```
<ant dir=„nbi-plugin“ target="deploy" />
```
- Run `ant` from the top level directory to compile all and `ant war` to get a new Web archive file for Tomcat

Link Suche with Nutch

- Your task is to edit the crawl so that
 - Both the inlinks and outlinks (URL lists) are indexed with the page being crawled
 - Nutch Search supports queries like „which pages have outlinks to x“ or „which pages have inlinks from x“
- Edit or extend the existing Nutch plugins
- Build Nutch and deploy the Web search in Tomcat

Link Suche with Nutch (2)

- Outlinks
 - **org.apache.nutch.parse.html.DOMContentUtils** has a `getOutlinks()` method
 - **org.apache.nutch.parse.Outlink** has a `getToUrl()` method
- Inlinks
 - **org.apache.nutch.crawl.Inlink** has a `getFromUrl()` method

Hints for the Exercise

- `HTMLParser.getParse` returns a `Parse` object
 - `Parse.getData()` returns a `ParseData` object
 - `ParseData.getOutlinks()` returns the page `Outlinks`
- Compare `index-basic` and `index-more` plugins
 - The latter shows how to index further data about a page
 - Page `Inlinks` are passed in the `Document` object
- Enable search using the new fields
 - `Outlinks` and `inlinks` should both be indexed as new fields of the `Document` object
 - Field search is supported already in `query-basic` plugin

- A very useful document is <http://wiki.apache.org/nutch/WritingPluginExample>
- Index-more plugin source showed how to extend indexing to include more document fields
- Query-more plugin source showed how to extend querying to support new document fields
- A single new plugin can be written which extends both the indexing and querying to support in- and outlinks
`nbi-plugin`

- Example: MoreIndexingFilter.java

```
public Document filter(Document doc, Parse parse,
    UTF8 url, CrawlDatum datum, Inlinks inlinks)
    throws IndexingException {
String contentLength =
    parse.getData().getMeta(Response.CONTENT_LENGTH
    );
if (contentLength != null)
    doc.add(new Field("contentLength",
        contentLength, Field.Store.YES,
        Field.Index.NO));
return doc;
}
```

- Important to set the field parameters correctly
e.g. the anchors field in BasicIndexingFilter.java
`Field.Store.NO` means that the field content is not stored and hence not returned in the results
`Field.Index.TOKENIZED` means that the field contents will be split by a tokenizer (it is a String array)

- Example: NBIIndexer.java

```
public Document filter(Document doc, Parse parse,
    UTF8 url, CrawlDatum datum, Inlinks inlinks)
    throws IndexingException {
    .. Use Iterator to get inlink from inlinks ..
    .. For each inlink, getFromUrl() returns a string
    .. Concatenate these URLs in a string ..
    doc.add(new Field("in", in_urls, Field.Store.NO,
        Field.Index.TOKENIZED));
```

```
Outlink[] outlinks =
    parse.getData().getOutlinks()
.. Get array length and loop over all outlinks
.. Outlink.getToUrl() returns URL as string ..
Concatenate these URLs in a string ..
doc.add(new Field("out", out_urls,
    Field.Store.NO, Field.Index.TOKENIZED));

return doc;
}
```

- Example: URLQueryFilter.java

```
/**
 * Handles "url:" query clauses, causing them to search the
 * field indexed by
 * BasicIndexingFilter.
 */
public class URLQueryFilter extends FieldQueryFilter {

    public URLQueryFilter() {
        super("url");
    }

    public void setConf(Configuration conf) {
        super.setConf(conf);
    }

}
```

- Example: NBIInlinkQueryFilter.java

```
/**
 * Handles „in:“ query clauses, causing them to search the
 * inlinks field indexed by
 * NBIIndexer.
 */
public class NBIInlinkQueryFilter extends FieldQueryFilter {

    public NBIInlinkQueryFilter() {
        super(„in“);
    }

    public void setConf(Configuration conf) {
        super.setConf(conf);
    }

}
```



- The plugin has a folder `nbi-plugin`
 - With `plugin.xml` defining three extensions:
 - `NBIIndexer` extends `IndexingFilter`
 - `NBIInlinkQueryFilter` and `NBIOutlinkQueryFilter` extends `QueryFilter`
 - With `build.xml`
 - With the correct package structure e.g. `de/ag-nbi/nutch`
- Use `ant` to compile the plugin (-> `nbi-plugin.jar`)
- Add the plugin to `nutch-site.xml` (see next slide)
- Use `ant` to recompile Nutch (now our indexer will be used)
- Use `ant` to recompile the Nutch Web application (now our query filters will be used)
- Redeploy Nutch on Tomcat and try it out...

- Add plugin.includes in nutch-site.xml

```
<property>
<name>plugin.includes</name>
<value>nutch-extensionpoints|protocol-
  http|urlfilter-regex|parse-(text|html)|index-
  basic|query-(basic|site|url)|nbi-plugin</value>
<description>Regular expression naming plugin
  directory names to include. Any plugin not
  matching this expression is excluded. In any case
  you need at least include the nutch-
  extensionpoints plugin. By default Nutch includes
  crawling just HTML and plain text via HTTP, and
  basic indexing and search plugins. </description>
</property>
```

- Problems with plugin.includes?

- Change it directly in nutch-default.xml

- **Termine**

- Ausgabe: 6.1.2009 Abgabe bis: 20.1.2009, 16:00 Uhr
- 2-wöchiges Übungsblatt
- 2er-Teams

- Fragen?

Viel Spaß und Erfolg !

