Documentation for the Combine (focused) crawling system

Anders Ardö

November 29, 2006

Contents

Part I
Overview

1 Introduction

The Combine system is an open, free, and highly configurable system for focused crawling of Internet resources.

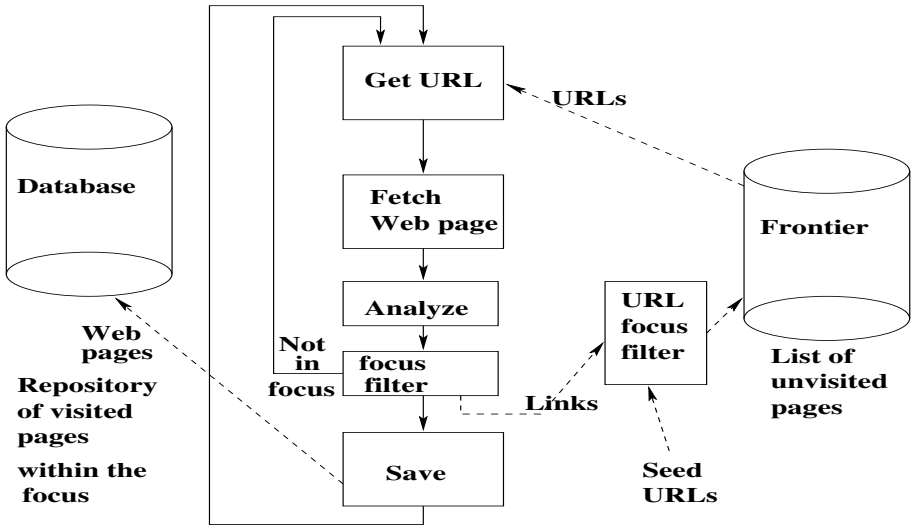

Figure 1: Overview of Combine focused crawler.

Main features include

- part of the SearchEngine-in-a-Box[1] system
- extensive configuration possibilities
- integrated topic filter (automated topic classifier) for focused crawling mode
- possibility to use any topic filter (if provided as a Perl Plug-In module[2]) in focused crawling mode
- crawl limitations based on regular expression on URLs both include and exclude (URL focus filter)
- character set detection/normalization
- language detection
- HTML cleaning
- metadata extraction

[1] http://combine.it.lth.se/SearchEngineBox/
[2] http://combine.it.lth.se/PlugIns/

- structured records for each crawled page
- supports many document types (text, HTML, PDF, PostScript, MsWord, PowerPoint, Excel, RTF, TeX, images)
- SQL database for data storage and administration

Naturally it obeys the Robots Exclusion Protocol[3] and behaves nice to Web-servers. Besides focused crawls (generating topic specific databases) Combine supports configurable rules on what's crawled based on regular expressions on URLs (URL focus filter). The crawler is designed to run continuously in order to keep the topic-specific database as up to date as possible.

The operation of Combine (overview in figure 1) as a focused crawler is based on a combination of a general Web crawler and an automated subject classifier. The topic focus is provided by a focus filter using a topic definition implemented as a thesaurus, where each term is connected to a topic class.

Crawled data are stored as a structured records in a local relational database.

Section 2 outlines how to download, install and test the Combine system and includes use scenarios.

Section 3 discuss configuration structure and highlights a few important configuration variables.

Section 4 describes policies and methods used by the crawler.

The system has a number of components (see section 7), the main user visible ones being `combineCtrl` which is used to start and stop crawling, and view crawler status and `combineExport` that extracts crawled data from the internal database and exports it as XML records.

Further details (lots and lots of them) can be found in part II 'Gory details' and in Appendix A.

2 Open Source distribution, Installation

The focused crawler have been restructured and packaged as a Debian package in order to ease distribution and installation. The package contains dependency information to make sure that all software that is needed to run the crawler is installed at the same time. In connection with this we have also packaged a number of necessary Perl-modules as Debian packages.

All software and packages are available from the Combine focused crawler Web-site[4].

2.1 Installation

This distribution is developed and tested on Linux systems. It is implemented entirely in Perl and uses the MySQL[5] database system, both of which are supported on many other operating systems. Porting to other UNIX dialects should be easy.

The system is distributed either as source or as a Debian package.

[3] http://www.robotstxt.org/wc/exclusion.html
[4] http://combine.it.lth.se/
[5] http://www.mysql.com/

2.1.1 Installation from source for the impatient

Unless you are on a system supporting Debian packages (in which case look at Automated installation (section 2.1.3)) you should download and unpack the source. The following command sequence will then install Combine:

```
perl Makefile.PL
make
make test
make install
mkdir /etc/combine
cp conf/* /etc/combine/
mkdir /var/run/combine
```

Test that it all works (run as root)
`./doc/InstallationTest.pl`

2.1.2 Porting to not supported operating systems - dependencies

In order to port the system to another platform, you have to verify the availability, for this platform, of the two main systems:

- Perl[6]
- MySQL version ≥ 4.1[7]

If they are supported you stand a good chance to port the system.

Furthermore the external Perl modules (listed in 10.3) should be verified to work on the new platform.

Perl modules are most easily installed using the Perl CPAN automated system (`perl -MCPAN -e shell`).

Optionally these external programs will be used if they are installed on your system.

- antiword (parsing MSWord files)
- detex (parsing TeX files)
- pdftohtml (parsing PDF files)
- pstotext (parsing PS and PDF files, needs ghostview)
- xlhtml (parsing MSExcel files)
- ppthtml (parsing PowerPoint files)
- unrtf (parsing RTF files)
- tth (parsing TeX files)
- untex (parsing TeX files)

[6] http://www.cpan.org/ports/index.html
[7] http://dev.mysql.com/downloads/

2.1.3 Automated Debian/Ubuntu installation

- add the following lines to your /etc/apt/sources.list
  `deb http://combine.it.lth.se/ debian/`
- give the commands
  `apt-get update`
  `apt-get install combine`

This also installs all dependencies such as MySQL and a lot of necessary Perl modules.

2.1.4 Manual installation

Download the latest distribution[8].

Install all software that Combine depends on (see above).

Unpack the archive with `tar zxf`
This will create a directory named `combine-XX` with a number of subdirectories including `bin, Combine, doc, and conf`.

'`bin`' contains the executable programs.

'`Combine`' contains needed Perl modules. Should be copied to somewhere Perl will find them, typically `/usr/share/perl5/Combine/`.

'`conf`' contains the default configuration files. Combine looks for them in `/etc/combine/` so they need to be copied there.

'`doc`' contains documentation.

The following command sequence will install Combine:

```
perl Makefile.PL
make
make test
make install
mkdir /etc/combine
cp conf/* /etc/combine/
mkdir /var/run/combine
```

2.1.5 Installation test

Harvest 1 URL by doing:

```
sudo combineINIT --jobname aatest --topic /etc/combine/Topic_carnivor.txt
combine --jobname aatest --harvest http://combine.it.lth.se/CombineTests/InstallationTest.html
combineExport --jobname aatest --profile dc
```

and verify that the output, except for dates and order, looks like

```
<?xml version="1.0" encoding="UTF-8"?>
<documentCollection version="1.1" xmlns:dc="http://purl.org/dc/elements/1.1/">
<metadata xmlns:dc="http://purl.org/dc/elements/1.1/">
<dc:format>text/html</dc:format>
```

[8] http://combine.it.lth.se/#downloads

```
<dc:format>text/html; charset=iso-8859-1</dc:format>
<dc:subject>Carnivorous plants</dc:subject>
<dc:subject>Drosera</dc:subject>
<dc:subject>Nepenthes</dc:subject>
<dc:title transl="yes">Installation test for Combine</dc:title>
<dc:description></dc:description>
<dc:date>2006-05-19 9:57:03</dc:date>
<dc:identifier>http://combine.it.lth.se/CombineTests/InstallationTest.html</dc:identifier>
<dc:language>en</dc:language>
</metadata>
```

Or run - as root - the script `./doc/InstallationTest.pl` (see Appendix A.1) which essentially does the same thing.

2.2 Getting started

A simple example work-flow for a trivial crawl job name 'aatest' might look like:

1. Initialize database and configuration (needs root privileges)
   `sudo combineINIT --jobname aatest`
2. Load some seed URLs like (you can repeat this command with different URLs as many times as you wish)
   `echo 'http://combine.it.lth.se/' | combineCtrl load --jobname aatest`
3. Start 2 harvesting processes
   `combineCtrl start --jobname aatest --harvesters 2`
4. Let it run for some time. Status and progress can be checked using the program '`combineCtrl --jobname aatest`' with various parameters.
5. When satisfied kill the crawlers
   `combineCtrl kill --jobname aatest`
6. Export data records in the ALVIS XML format
   `combineExport --jobname aatest --profile alvis`
7. If you want to schedule a recheck for all the crawled pages stored in the database
   `combineCtrl reharvest --jobname aatest`
8. go back to 3 for continuous operation.

Once a job is initialized it is controlled using `combineCtrl`. Crawled data is exported using `combineExport`.

2.3 Detailed documentation

The latest, updated, detailed documentation is always available online[9].

[9] http://combine.it.lth.se/documentation/

2.4 Use scenarios

2.4.1 General crawling without restrictions

Same procedure as in section 2.2. This way of crawling is not recommended for the Combine system since it will generate really huge databases without any focus.

2.4.2 Focused crawling - domain restrictions

Create a focused database with all pages from a Web-site. In this use scenario we will crawl the Combine site and the ALVIS site. The database is to be continuously updated, ie all pages have to be regularly tested for changes, deleted pages should be removed from the database, and newly created pages added.

1. Initialize database and configuration
   `sudo combineINIT --jobname focustest`

2. Edit the configuration to provide the desired focus
   Change the `<allow>` part in `/etc/combine/focustest/combine.cfg` from

   ```
   #use either URL or HOST: (obs ':') to match regular expressions to either the
   #full URL or the HOST part of a URL.
   <allow>
   #Allow crawl of URLs or hostnames that matches these regular expressions
   HOST: .*$
   </allow>
   ```

   to

   ```
   #use either URL or HOST: (obs ':') to match regular expressions to either the
   #full URL or the HOST part of a URL.
   <allow>
   #Allow crawl of URLs or hostnames that matches these regular expressions
   HOST: www\.alvis\.info$
   HOST: combine\.it\.lth\.se$
   </allow>
   ```

   The escaping of '.' by writing '\.' is necessary since the patterns actually are Perl regular expressions. Similarly the ending '$' indicates that the host string should end here, so for example a Web server on `www.alvis.info.com` (if such a one exists) will not be crawled.

3. Load seed URLs
   ```
   echo 'http://combine.it.lth.se/' | combineCtrl load --jobname focustest
   echo 'http://www.alvis.info/' | combineCtrl load --jobname focustest
   ```

4. Start 1 harvesting process
   `combineCtrl start --jobname focustest`

5. Daily export all data records in the ALVIS XML format
   `combineExport --jobname focustest --profile alvis`
   and schedule all pages for re-harvesting
   `combineCtrl reharvest --jobname focustest`

2.4.3 Focused crawling - topic specific

Create and maintain a topic specific crawled database for the topic 'Carnivorous plants'.

1. Create a topic definition (see section 4.5.1) in a local file named `cpTopic.txt`. (Can be done by copying `/etc/combine/Topic_carnivor.txt` since it happens to be just that.)

2. Create a file named `cpSeedURLs.txt` with seed URLs for this topic, containing the URLs:

   ```
   http://www.sarracenia.com/faq.html
   http://dmoz.org/Home/Gardening/Plants/Carnivorous_Plants/
   http://www.omnisterra.com/bot/cp_home.cgi
   http://www.vcps.au.com/
   http://www.murevarn.se/links.html
   ```

3. Initialization
   `sudo combineINIT --jobname cptest --topic cpTopic.txt`

   This enables topic checking and focused crawl mode by setting configuration variable `doCheckRecord = 1` and copying a topic definition file (`cpTopic.txt`) to `/etc/combine/cptest/topicdefinition.txt`.

4. Load seed URLs
   `combineCtrl load --jobname cptest < cpSeedURLs.txt`

5. Start 3 harvesting process
   `combineCtrl start --jobname cptest --harvesters 3`

6. Regularly export all data records in the ALVIS XML format
   `combineExport --jobname cptest --profile alvis`
   and schedule all pages for re-harvesting
   `combineCtrl reharvest --jobname cptest`

3 Configuration

3.1 Configuration files

All configuration files are stored in the `/etc/combine/` directory tree. All configuration variables have reasonable defaults (section 9).

job_default.cfg is job specific defaults. It is copied to a subdirectory named after the job by `combineINIT`.

SQLstruct.sql Structure of the internal SQL database used both for administration and to hold data records. Details in section A.4.

Topic_* contains various contributed topic definitions.

Global configuration files These files are used for global parameters for all crawler jobs.

default.cfg is the global defaults. It is loaded first. Consult section 9 and appendix A.3 for details. Values can be overridden from the job-specific configuration file `combine.cfg`.

tidy.cfg configuration for Tidy cleaning of HTML code

Files in job specific sub-directories The program `combineINIT` creates a job specific subdirectory in `/etc/combine` and populates it with some files including `combine.cfg` initialized with a copy of `job_default.cfg`. The job-name have to be given to all programs when started using the `--jobname` switch.

combine.cfg the job specific configuration. It is loaded secondly and overrides the global defaults. Consult section 9 and appendix A.3 for details.

topicdefinition.txt contains the topic definition for focused crawl if the `--topic` switch is given to `combineINIT`. The format of this file is described in section 4.5.1.

stopwords.txt a file with words to be excluded from the automatic topic classification processing. One word per line. Can be empty but must be present.

config_exclude contains more exclude patterns. Optional, automatically included by `combine.cfg`. Updated by `combineUtil`.

config_serveralias contains patterns for resolving Web server aliases. Optional, automatically included by `combine.cfg`. Updated by `combineUtil`.

sitesOK.txt optionally used by the built in automated classification algorithms (section 4.5) to bypass the topic filter for certain sites.

Configuration files use a simple format consisting of either name/value pairs or complex variables in sections. Name/value pairs are encoded as single lines formated like '`name = value`'. Complex variables are encoded as multiple lines in named sections delimited as in XML, using '`<name> ... </name>`'. Sections may be nested for related configuration variables. Empty lines and lines starting with '`#`' (comments) are ignored.

The most important configuration variables are the complex variables `<url><allow>` (allows certain URLs to be harvested) and `<url><exclude>` (excludes certain URLs from harvesting) which are used to limit your crawl to just a section of the Web, based on the URL. Loading of URLs to be crawled into the system checks each URL first against the Perl regular expressions of `<url><allow>` and if it matches goes on to match it against `<url><exclude>` where it's discarded if it matches, otherwise it's scheduled for crawling. (See section 4.3 'URL filtering').

You should always change the value of the variable `Operator-Email` in the file `/etc/combine/aatest/combine.cfg` and set it to something reasonable. It is used by Combine to identify you to the crawled Web-servers.

Further details are found in section 9 'Configuration variables'.

4 Crawler operation

The system is designed for continuous operation. The harvester processes an URL in several steps as is detailed in figure 2. As a start-up initialization the frontier has to be seeded with some relevant URLs. All URLs are normalized before they are entered in the database. Data can be exported in various formats including the ALVIS XML document format[10] as well as Dublin Core[11] records.

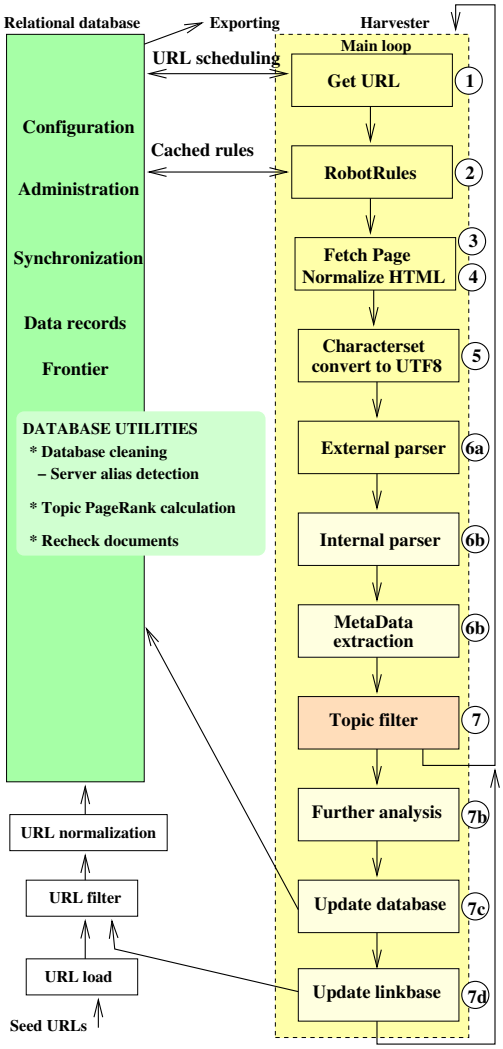

Figure 2: Architecture for the Combine focused crawler.

The steps taken during crawling (numbers refer to figure 2):

1. The next URL is fetched from the scheduler.

2. Combine obeys the the Robots Exclusion Protocol[12]. Rules are cached locally.

[10] http://www.alvis.info/alvis/architecture
[11] http://dublincore.org/
[12] http://www.robotstxt.org/wc/exclusion.html

3. The page is retrieved using a GET, GET-IF-MODIFIED, or HEAD HTTP request.
4. The HTML code is cleaned and normalized.
5. The character-set is detected and normalized to UTF-8.
6. (a) The page (in any of the formats PDF, PostScript, MS Word, MSExcel, PowerPoint, RTF and TeX/LaTeX) is converted to HTML (or plain text) by an external program.
   (b) Internal parsers handles HTML, plain text and images. This step extracts structured information like metadata (title, keywords, description, ...), HTML links, and text without markup.
7. The document is sent to the topic filter, (see section 4.5). If the Web-page is relevant with respect to the focus topic, processing continues with:
   (a) Heuristics like score propagation
   (b) Further analysis, like genre and language identification.
   (c) Update the record database
   (d) Update the frontier database with HTML links and URLs extracted from plain text.

Depending on several factors like configuration, hardware, network, workload, etc, the crawler normally processes between 50 and 250 URLs per minute. In general focused crawling using 'reasonable' topic definitions can do 100 - 150 URLs per minute.

4.1 URL selection criteria

In order to successfully select and crawl one URL the following conditions have to be meet:

1. The URL have to be selected by the scheduling algorithm (section 4.4).

   Relevant configuration variables: WaitIntervalHost (section 9.1.32), WaitIntervalHarvesterLockRobotRules (section 9.1.29), WaitIntervalHarvesterLockSuccess (section 9.1.30)
2. The URL have to pass the allow test

   Relevant configuration variables: allow (section 9.2.1)
3. The URL is not be excluded by the exclude test (see section 4.3).

   Relevant configuration variables: exclude (section 9.2.4)
4. The Robot Exclusion Protocol have to allow crawling of the URL
5. Optionally the document at the URL location have to pass the topic filter (section 4.5).

   Relevant configuration variables: classifyPlugIn (section 9.1.3), doCheckRecord (section 9.1.6).

4.2 Document parsing

The system selects a document parser based on the Mime-Type together with available parsers and converter programs

1. For some mime-types an external program is called in order to convert the document to a format handled internally (HTML or plain text).

   Relevant configuration variables: converters (section 9.2.3)

2. Internal parsers handle HTML, plain text, TeX, and Image.

   Relevant configuration variables: converters (section 9.2.3)

Supporting a new document format is as easy as providing a program that can convert a document in this format to HTML or plain text. Configuration of the mapping between document format (Mime-Type) and converter program is done in the complex configuration variable 'converters' (section 9.2.3).

4.3 URL filtering

Before an URL is accepted for scheduling (either by manual loading or recycling) it is normalized and validated. This process comprises a number of steps:

- Normalization
  - General practice: host-name lowercasing, port-number substitution, canonical URL
  - Remove fragments (ie '#' and everything after that)
  - Clean CGI repetitions of parameters
  - Collapse dots ('./', '../') in the path
  - Remove CGI parameters that are session ids, as identified by patterns in the configuration variable sessionids (section 9.2.6)
  - Normalize WWW-server names by resolving aliases. Identified by patterns in the configuration variable serveralias (section 9.2.5). These patterns can be generated by using the program `combineUtil` to analyze a crawled corpus.
- Validation: A URL have to pass all three validation steps outlined below.
  - URL length have to be less than configuration variable maxUrlLength (section 9.1.14)
  - Allow test: one of the Perl regular expressions in the configuration variable allow (section 9.2.1) must match the URL
  - Exclude test: none of the Perl regular expressions in the configuration variable exclude (section 9.2.4) must match the URL

  Both allow and exclude can contain two types of regular expressions identified by either '`HOST:`' or '`URL`' in front of the regular expression. The '`HOST:`' regular expressions are matched only against the WWW-server part of the URL while the '`URL`' regular expressions are matched against the entire URL.

4.4 Link selection/scheduling policy

All links from a relevant document are extracted, normalized and stored in the structured record. Those links that pass the selection/validation criteria outlined below are marked for crawling.

To mark a URL for crawling requires

- The URL should be from a page that is relevant (ie pass the focus filter)
- The URL scheme must be one of HTTP, HTTPS, or FTP
- The URL must not exceed the maximum length (configurable, default 250 characters)
- It should pass the 'allow' test (configurable, default all URLs passes)
- It should pass the 'exclude' test (configurable, default excludes malformed URLs, some CGI pages, and URLs with file-extensions for binary formats)

At each scheduling point one URL from each available (unlocked) host is selected to generate a ready queue, which is then processed completely before a new scheduling is done. Each selected URL in the ready queue thus fulfills these requirements:

- URL must be marked for crawling (see above)
- URL must be unlocked (each successful access to a URL lock it for a configurable time WaitIntervalHarvesterLockSuccess (section 9.1.30))
- Host of the URL must be unlocked (each access to a host locks it for a configurable time WaitIntervalHost (section 9.1.32))

4.5 Built in topic filter - automated subject classification

The built in topic filter is a library science approach to automated classification, using a topic definition with a pre-defined controlled vocabulary of topical terms, to determine relevance judgement. Thus it does not rely on a particular set of seed pages, or a collection of pre-classified example pages to learn from. It does require that some of the seed pages are relevant and contain links into the topical area. One simple way of creating a set of seed pages would be to use terms from the controlled vocabulary as queries for a general purpose search engine and take the result as seed pages.

The system for automated topic classification (overview in figure 3), that determines topical relevance in the topical filter, is based on matching subject terms from a controlled vocabulary in a topic definition with the text of the document to be classified [1]. The topic definition uses subject classes in a hierarchical classification system (corresponding to topics) and terms associated with each subject class. Terms can be single words, phrases, or boolean AND-expressions connecting terms. Boolean OR-expressions are implicitly handled by having several different terms associated with the same subject class (see section 4.5.1).

The algorithm works by string-to-string matching of terms and text in documents. Each time a match is found the document is awarded points based on which term is matched and in which structural part of the document (location) the match is found [5]. The points are summed to make the final relevance score of the document. If the score is above

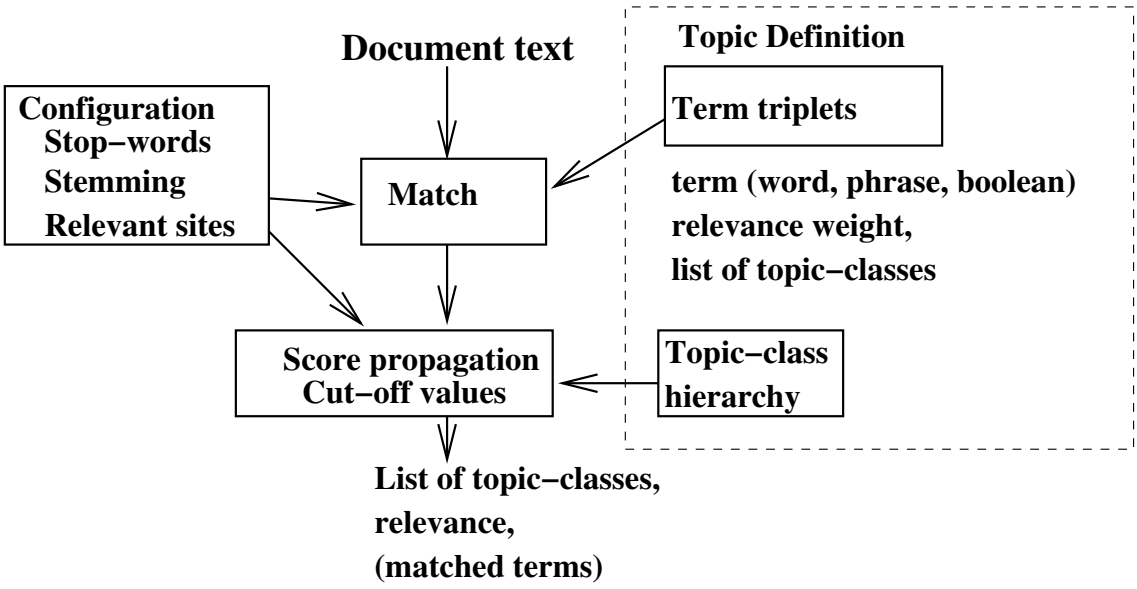

Figure 3: Overview of the automated topic classification algorithm

a cut-off value the document is saved in the database together with a (list of) subject classification(s) and term(s).

By providing a list of known relevant sites in the configuration file `sitesOK.txt` (located in the job specific configuration directory) the above test can be bypassed. It works by checking the host part of the URL against the list of known relevant sites and if a match is found the page is validated and saved in the database regardless of the outcome of the algorithm.

4.5.1 Topic definition

Located in `/etc/combine/<jobname>/topicdefinition.txt`. Topic definitions use triplets (term, relevance weight, topic-classes) as its basic entities. Weights are signed integers and indicate the relevance of the term with respect to the topic-classes. Higher values indicate more relevant terms. A large negative value can be used to exclude documents containing that term.

Terms can be:

- single words
- a phrase (ie all words in exact order)
- a boolean AND-expression connecting terms (ie all terms must be present but in any order). The boolean AND operator is encoded as '@and'

A boolean OR-expression has to be entered as separate term triplets. The boolean expression "`polymer AND (atactic OR syndiotactic)`" thus have to be translated into two separate triplets, one containing the term "`polymer @and atactic`", and another with "`polymer @and syndiotactic`".

Terms can include (Perl) regular expressions like:

- a '?' makes the character immediately preceding optional, ie the term "`coins?`" will match both "`coin`" and "`coins`"
- a "`[^\s]*`" is truncation (matches all character sequences except space ' '), "`glass art[^\s]*`" will match "`glass art`", "`glass arts`", "`glass artists`", "`glass articles`", and so on.

It is important to understand that each triplet in the topic definition is considered by itself without any context, so they must **each** be topic or sub-class specific in order to be useful. Subject neutral terms like "use", "test", "history" should not be used. If really needed they have to be qualified so that they become topic specific (see examples below).

Simple guidelines for creating the triplets and assigning weights

- Phrases or unique, topic specific terms, should be used if possible, and assigned the highest weights, since they normally are most discriminatory.
- Boolean AND-expressions is the next best.
- Single words can be to general and/or have several meanings or uses that make them less specific and those should thus be assigned a small weights.
- Acronyms can be used as terms if they are unique
- Negative weights should be used in order to exclude concepts.

4.5.2 Topic definition (term triplets) BNF grammar

TERM-LIST :== TERM-ROW '<**cr**>' || '#' <**char**>+ '<**cr**>' || '<**cr** >'
TERM-ROW :== WEIGHT ': ' TERMS '=' CLASS-LIST
WEIGHT :== ['-']<**integer**>
TERMS :== TERM [' **@and** ' TERMS]*
TERM :== WORD ' ' [WORD]*
WORD :== <**char**>+||<**char**>+<**perl-reg-exp**>
CLASS-LIST :== CLASSID [',' CLASS-LIST]
CLASSID :== <**char**>+

A line that starts with '#' is ignored as are empty lines.

<**perl-reg-exp**> is only supported by the plain matching algorithm described in section 4.5.4.

"CLASSID" is a topic (sub-)class specifier, often from a hierarchical classification system like Engineering Index[13].

4.5.3 Term triplet examples

```
50: optical glass=A.14.5, D.2.2
30: glass @and fiberoptics=D.2.2.8
50: glass @and technical @and history=D.2
50: ceramic materials @and glass=D.2.1.7
-10000: glass @and art=A
```

[13] http://www.ei.org/

The first line says that a document containing the term "`optical glass`" should be awarded 50 points for each of the two classes A.14.5 and D.2.2.

"`glass`" as a single term is probably too general, qualify it with more terms like: "`glass @and fiberoptics`" , or "`glass @and technical @and history`" or use a phrase like "`glass fiber`" or "`optical glass`".

In order to exclude documents about artistic use of glass the term "`glass @and art`" can be used with a (high) negative score.

An example from the topic definition for 'Carnivorous Plants' using regular expressions.

```
#This is a comment
75: D\.?\s*californica=CP.Drosophyllum
10: pitcher[^\s]*=CP
-10: pitcher[^\s]* @and baseball=CP
```

The term "`D\.?\s*californica`" will match `D californica, D. californica, D.californica` etc.

The last two lines assures that a document containing "`pitcher`" gets 10 points but if the document also contains "`baseball`" the points are removed.

4.5.4 Algorithm 1: plain matching

Selected by setting the configuration parameter `classifyPlugIn = Combine::Check_record`

The algorithm produces a list of suggested topic-classes (subject classifications) and corresponding relevance scores using the algorithm:

$$\text{Relevance\_score} =$$

$$\sum_{\text{all locations}} \left(\sum_{\text{all terms}} (hits[\text{location}_j][\text{term }_i] * weight[\text{term}_i] * weight[\text{location}_j]) \right)$$

term weight ($weight[\text{term}_i]$) is taken from the topic definition triplets.

location weight ($weight[\text{location}_j]$) are defined ad hoc for locations like title, metadata, HTML headings, and plain text. However the exact values for these weights does not seem to play a large role in the precision of the algorithm [5].

hits ($hits[\text{location}_j][\text{term}_i]$) is the number of times term_i occur in the text of location_j

The summed relevance score might, for certain applications, have to be normalized with respect to text size of the document.

One problem with this algorithm is that a term that is found in the beginning of the text contributes as much as a term that is found at the end of a large document. Another problem is the distance and thus the coupling between two terms in a Boolean expression might be very large in a big document and this is not taken into account by the above algorithm.

4.5.5 Algorithm 2: position weighted matching

Selected by setting the configuration parameter `classifyPlugIn = Combine::PosCheck_record`

In response to the problems cited above we developed a modified version of the algorithm that takes into account word position in the text and proximity for boolean terms. It also eliminates the need to assign ad hoc weights to locations. The new algorithm works as follows.

First all text from all locations are concatenated (in the natural importance order title, metadata, text) into one chunk of text. Matching of terms is done against this chunk. Relevance score is calculated as

$$\text{Relevance\_score} =$$

$$\sum_{\text{all terms}} \left(\sum_{\text{all matches}} \frac{weight[\text{term}_i]}{\log(k * position[\text{term}_i][\text{match}_j]) * proximity[\text{term}_i][\text{match}_j]} \right)$$

term weight ($weight[\text{term}_i]$) is taken from the topic definition triplets

position ($position[\text{term}_i][\text{match}_j]$) is the position in the text (starting from 1) for match_j of term_i. The constant factor k is normally 0.5

proximity ($proximity[\text{term}_i][\text{match}_j]$) is

$$\begin{cases} 1 & \text{for non boolean terms} \\ \log(distance\_between\_components) & \text{for boolean terms} \end{cases}$$

In this algorithm a matched term close to the start of text contribute more to the relevance score than a match towards the end of the text. And for Boolean terms the closer the components are the higher the contribution to the relevance score.

4.6 Topic filter Plug-In API

The configuration variable classifyPlugIn (section 9.1.3) is used to find the Perl module that implements the desired topic filter. The value should be formatted as a valid Perl module identifier (ie the module must be somewhere in the Perl module search path). Combine will call a subroutine named 'classify' in this module, providing a XWI-object as in parameter. The subroutine must return either 0 or 1, where
0: means record fails to meet the classification criteria, ie ignore this record
1: means record is OK and should be stored in the database, and links followed by the crawler

A XWI-object is a structured object holding all information from parsing a Web-page.

More details on how to write a Plug-In can be found in the example Plug-In classifyPlugInTemplate.pm (see Appendix A.2).

4.7 Analysis

Extra analysis is enabled by the configuration variable doAnalyse (section 9.1.5). Among other things analysis tries to determine the language of the text in the page. The URL is used to extract an indication of the category (University, Education, Research, Publication, Product, Top page, Personal page) of a page.

4.8 URL recycling

URLs for recycling comes from 3 sources:

- Links extracted during HTML parsing
- Redirects (unless configuration variable UserAgentFollowRedirects (section 9.1.23) is set)
- URLs extracted from plain text (enabled by the configuration variable extractLinksFromText (section 9.1.8)).

Automatic recycling of URLs is enabled by the configuration variable AutoRecycleLinks (section 9.1.1). It can also be done manually with the command
`combineCtrl -jobname XXXX recyclelinks`

The command `combineCtrl -jobname XXXX reharvest` marks all pages in the database for harvesting again.

4.9 Complete application - SearchEngine in a Box

The SearchEngine-in-a-Box[14] system is based on the two systems Combine Focused Crawler and Zebra text indexing and retrieval engine[15]. This system allows you build a vertical search engine for your favorite topic in a few easy steps.

The SearchEngine-in-a-Box web-site contains instructions and downloads to make this happen. Basically it makes use of the ZebraHost (see section 9.1.37) configuration variable which enables direct communication between the crawler and the database system and thus indexes records as soon as they are crawled. Which also means that they are directly searchable.

5 Evaluation of automated subject classification

Automated classification approach used for evaluation was string-matching of terms (section 4.5) from an engineering-specific controlled vocabulary Engineering Index, used in Elsevier's Compendex database. One reason for choosing this approach is that it does not require training documents that are often not available especially on the World Wide Web. For a discussion on different approaches to automated subject classification, see [2].

In [6] a machine-learning and a string-matching approach to automated subject classification of text were compared as to their performance on a test collection of six classes. It was shown that SVM on average outperforms the string-matching approach: our hypothesis that SVM yields better recall and string-matching better precision was confirmed only on one of the classes. The two approaches being complementary, we investigated different combinations of the two based on combining their vocabularies. SVM performs best using the original set of terms, and string-matching approach also has best precision when using the original set of terms. Best recall for string-matching is achieved when using descriptive terms. Reasons for these results need further investigation, including a larger data collection and combining the two using predictions.

Another reason for choosing the string-matching approach is that Engineering Index is a well-developed vocabulary with an average of 88 manually selected terms designating

[14] http://combine.it.lth.se/SearchEngineBox/
[15] http://www.indexdata.dk/zebra/

one class [4]. In this study, it has been explored to what degree different types of terms in Engineering Index influence automated subject classification performance. Preferred terms, their synonyms, broader, narrower, related terms, and captions were examined in combination with a stemmer and a stop-word list. The results showed that preferred terms perform best, whereas captions perform worst. Stemming in most cases showed to improve performance, whereas the stop-word list did not have a significant impact. The majority of classes is found when using all the terms and stemming: micro-averaged recall is 73

In [5] the was to determine how significance indicators assigned to different Web page elements (internal metadata, title, headings, and main text) influence automated classification. The data collection that was used comprised 1000 Web pages in engineering, to which Engineering Information classes had been manually assigned. The significance indicators were derived using several different methods: (total and partial) precision and recall, semantic distance and multiple regression. It was shown that for best results all the elements have to be included in the classification process. The exact way of combining the significance indicators turned out not to be overly important: using the F1 measure, the best combination of significance indicators yielded no more than 3

Other issues specific to Web pages were identified and discussed in [3]. The focus of the study was a collection of Web pages in the field of engineering. Web pages present a special challenge: because of their heterogeneity, one principle (e.g. words from headings are more important than main text) is not applicable to all the Web pages of a collection. For example, utilizing information from headings on all Web pages might not give improved results, since headings are sometimes used simply instead of using bold or a bigger font size. A number of weaknesses of the described approach were identified, and ways to deal with those were proposed for further research. These include enriching the term list with synonyms and different word forms, adjusting the term weights and cut-off values and word-sense disambiguation.

6 Performance

Performance evaluation of the automated subject classification component is treated in section 5. Performance in terms of number of URLs treated per minute is of course highly dependent on a number of circumstances like network load, capacity of the machine, the selection of URLs to crawl, configuration details, number of crawlers used, etc. In general, within rather wide limits, you could expect the Combine system to handle up to 200 URLs per minute. Handle here means everything from scheduling of URLs, fetching of pages over the network, parsing the page, automated subject classification, recycling of new links, to storing the structured record in a relational database. This holds for small simple crawls starting from scratch to large complicated topic specific crawls with millions of records.

The prime way of increasing performance is to use more than one crawler for a job. This is handled by the `-harvesters` switch used together with the `combineCtrl start` command (for example

```
combineCtrl -jobname MyCrawl -harvesters 5 start
```

will start 5 crawlers working together on the job 'MyCrawl'. The effect of using more than one crawler on crawling speed is illustrated in figure 4 below.

Configuration also have an effect on performance. In figure 5 performance improvements based on configuration changes are shown. The choice of algorithm for automated classification turns out to have biggest influence on performance, where algorithm 2 - section 4.5.5 - (`classifyPlugIn = Combine::PosCheck_record` - Pos in figure 5) is much

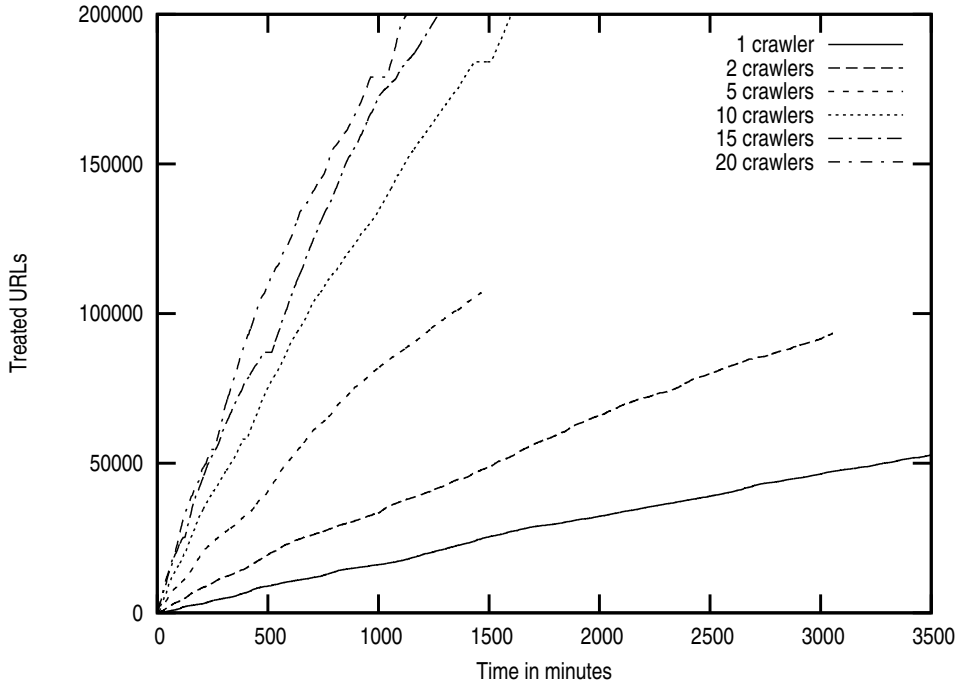

Figure 4: Combine crawler performance.

faster than algorithm 1 - section 4.5.4 - (`classifyPlugIn = Combine::Check_record` - Std in figure 5). Tweaking of other configuration variables also have an effect on performance but to a lesser degree. Tweaking consisted of not using Tidy to clean HTML (`useTidy = 0`) and not storing the original page in the database (`saveHTML = 0`).

7 System components

All executables take a mandatory switch `-jobname` which is used to identify the particular crawl job you want and as well the job-specific configuration directory.

Briefly `combineINIT` is used to initialize SQL database and the job specific configuration directory. `combineCtrl` controls a Combine crawling job (start, stop, etc) as well as printing some statistics. `combineExport` export records in various XML formats and `combineUtil` provides various utility operations on the Combine database.

Detailed dependency information (section 10) can be found in the 'Gory details' section.

7.1 combineINIT

Creates a Mysql database, database tables and initializes it. If the database exists it is dropped and recreated. A job specific configuration directory is created in `/etc/combine/` and populated with a default configuration file.

If a topic definition filename is given, focused crawling using this topic defintion is enabled per default. Otherwise focused crawling is disabled, and Combine works as a general crawler.

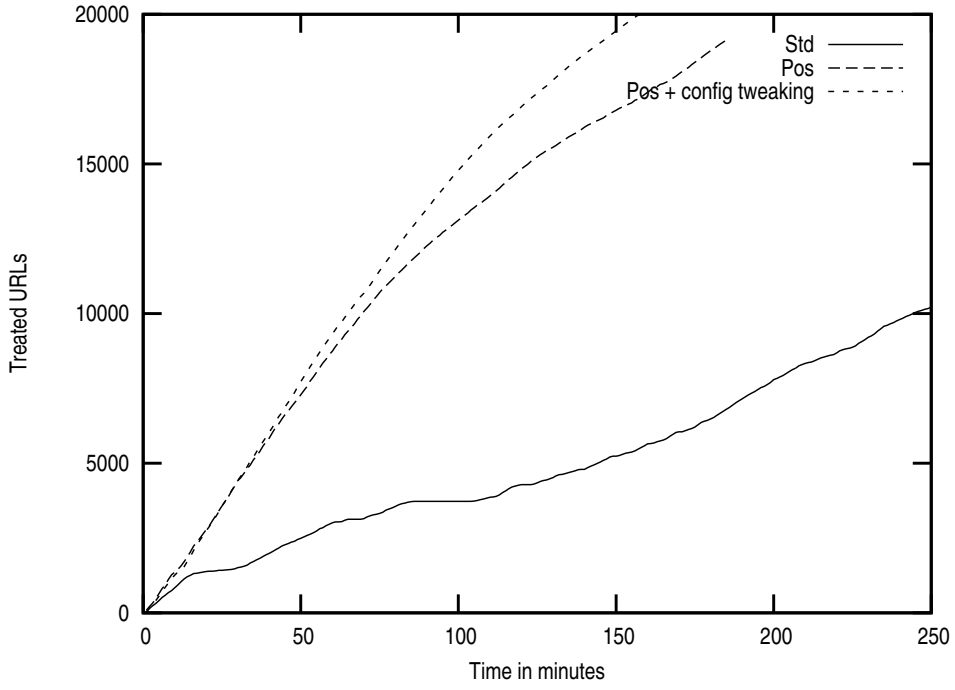

Figure 5: Effect of configuration changes on focused crawler performance.

7.2 combineCtrl

Implements various control functionality to administer a crawling job, like starting and stoping crawlers, injecting URLs into the crawl que, scheduling newly found links for crawling, controlling scheduling, etc.

This is the preferred way of controling a crawl job.

7.3 combineUtil

Does various statistics generation as well as performing sanity checks on the database.

7.4 combineExport

Export is done according to one of three profiles: `alvis`, `dc`, or `combine`. `alvis` and `combine` are very similar XML formats where `combine` is more compact with less redundancy and `alvis` contains some more information. `dc` is XML encoded Dublin Core data.

The `alvis` profile format is defined by the Alvis Enriched Document XML Schema[16].

For convinience a switch `-xsltscript` adds the possibility to filter the output using a XSLT script. The script is feed a record according to the `combine` profile and the result is exported.

7.5 Internal executables and Library modules

`combine` is the main crawling machine in the Combine system and `combineRun` starts, monitors and restarts `combine` crawling processes.

[16] http://www.miketaylor.org.uk/tmp/alvis/d3.1/enriched-document.xsd

7.5.1 Library

Main, crawler specific, library components are collected in the `Combine::` Perl namespace.

References

[1] A. Ardö and T. Koch. Automatic classification applied to the full-text Internet documents in a robot-generated subject index. In *Online Information 99, Proceedings*, pages 239–246, Dec. 1999. http://www.it.lth.se/anders/online99/.

[2] K. Golub. Automated subject classification of textual Web documents. *Journal of Documentation*, 62(3):350–371, 2006.

[3] K. Golub. Automated subject classification of textual web pages, based on a controlled vocabulary: challenges and recommendations. *New review of hypermedia and multimedia*, 12(1):11–27, June 2006. Special issue on knowledge organization systems and services.

[4] K. Golub. The role of different thesauri terms in automated subject classification of text. In *IEEE/WIC/ACM International Conference on Web Intelligence*, Dec. 2006.

[5] K. Golub and A. Ardö. Importance of HTML Structural Elements in Automated Subject Classification. In A. Rauber, S. Christodoulakis, and A. M. Tjoa, editors, *9th European Conference on Research and Advanced Technology for Digital Libraries - ECDL 2005*, volume 3652 of *Lecture Notes in Computer Science*, pages 368 – 378. Springer, Sept. 2005. Manuscript at: http://www.it.lth.se/knowlib/publ/ECDL2005.pdf.

[6] K. Golub, A. Ardö, D. Mladenic, and M. Grobelnik. Comparing and Combining Two Approaches to Automated Subject Classification of Text. In J. Gonzalo, C. Thanos, M. F. Verdejo, and R. C. Carrasco, editors, *10th European Conference on Research and Advanced Technology for Digital Libraries - ECDL 2006*, volume 4172 of *Lecture Notes in Computer Science*, pages 467–470. Springer, Sept. 2006.

Part II
Gory details

8 Frequently asked questions

1. What does the message 'Wide character in subroutine entry ...' mean?

2. What does the message 'Parsing of undecoded UTF-8 will give garbage when decoding entities ...' mean?

3. I can't figure out how to restrict the crawler to pages below '`http://www.foo.com/bar/`'?

   Put an appropriate regular expression in the <allow> section of the configuration file. Appropriate means a Perl regular expression, which means that you have to escape special characters. Try with
   `URL http:\/\/www\.foo\.com\/bar\/`

4. I have a simple configuration variable set, but Combine does not obey it?

   Check that there are not 2 instances of the same simple configuartion variable in the same configuration file. Unfortunately this will break configuration loading.

5. If there are multiple <allow> entries, must an URL fit all or any of them?

   A match to any of the entries will make that URL allowable for crawling. You can use any mix of HOST: and URL entries

6. It would also be nice to be able to crawl local files.

   Presently the crawler only accepts http, https, and ftp as protocols.

7. Crawling of a single host is VERY slow. Is there some way for me to speed the crawler up?

   Yes it's one of the built-in limitations to keep the crawler beeing 'nice' It will only access a particular server once every 60 seconds by default. You can change the default by adjusting the following configuration variables, but pleas keep in mind that you increase the load on the server.
   WaitIntervalSchedulerGetJcf=2
   WaitIntervalHost = 5

8. How can I crawl a fixed number of link steps from a set of seed pages? (Is it for example possible to crawl only one single webpage? Or one webpage and all local links on that webpage (and not any further)?)

   Initialize the database and load the seed pages. Turn of automatic recycling of links by setting the simple configuration variable 'AutoRecycleLinks' to 0.

   Start crawling and stop when '`combineCtrl -jobname XXX howmany`' equals 0.

   Handle recycling manually using 'combineCtrl, with action 'recyclelinks'. (Give the command `combineCtrl -jobname XXX recyclelinks`')

   Iterate to the depth of your liking.

9. I run combineINIT but the configuration directory is not created?

   You need to run combineINIT as root, due to file protection permissions.

10. Where are the logs?

    They are stored in the SQL database <jobname> in the table `log`.

11. What are the main differences between Std and PosCheck algorithms for automated subject classification?

12. I don't understand what this means. Can you explain it to me ? Thank you !

    ```
    40: sundew[^\s]*=CP.Drosera
    40: tropical pitcher plant=CP.Nepenthes
    ```

    It's part of the topic definition (term list) for the topic 'Carnivorous plants'. It's well described in the documenentation, please see section 4.5.1. The strange characters are Perl regular expressions mostly used for truncation etc.

13. I want to get all pages about "icecream" from "www.yahoo.com". And I don't have clear idea about how to write the topic definition file. Can you show me an example?

    So for getting all pages about 'icecream' from 'www.yahoo.com' you have to:

    (a) write a topic definition file according to the format above, eg containing topic specific terms. The file is essential a list of terms relevant for the topic. Format of the file is "numeric_importance: term=TopicClass" e.g. "`100: icecream=YahooIce`" (Say you call your topic 'YahooIce'). A few terms might be:

    ```
    100: icecream=YahooIce
    100: ice cone=YahooIce
    ```

    and so on stored in a file called say TopicYahooIce.txt

    (b) Initialization

    ```
    sudo combineINIT -jobname cptest -topic TopicYahooIce.txt
    ```

    (c) Edit the configuration to only allow crawling of www.yahoo.com Change the <allow> part in /etc/combine/focustest/combine.cfg from

    ```
    #use either URL or HOST: (obs ':') to match regular expressions to either the
    #full URL or the HOST part of a URL.
    <allow>
    #Allow crawl of URLs or hostnames that matches these regular expressions
    HOST: .*$
    </allow>
    ```

    to

    ```
    #use either URL or HOST: (obs ':') to match regular expressions to either the
    #full URL or the HOST part of a URL.
    <allow>
    #Allow crawl of URLs or hostnames that matches these regular expressions
    HOST: www\.yahoo\.com$
    </allow>
    ```

(d) Load some good seed URLs

(e) Start 1 harvesting process

14. Why load some good seeds URLs and what the seeds URLs mean.

    This is just a way of telling the crawler where to start.

15. My problem is that the installation there requires root access, which I cannot get. Is there a way of running Combine without requiring any root access?

    The are three things that are problematic

    (a) Configurations are stored in /etc/combine/...

    (b) Runtime PID files are stored in /var/run/combine

    (c) You have to be able to create MySQL databases accessible by combine Apart from that nothing else needs root access.

    If you take the source and look how the tests (make test) are made you might find a way to fix 1. Though this probably involves modifying the source - maybe only the Combine/Config.pm

    2. is strictly not necessary and it will run even if /var/run /combine does not exist, although not the command 'combineCtrl –jobname XXX kill'

    3. is necessary and I can't think of a way around it except making a local installation of MySQL and use that.

16. What does the following entries from the log table mean?

    (a) `| 5409 | HARVPARS 1_zltest | 2006-07-14 15:08:52 | M500; SD empty, sleep 20 second..`

    This means that there are no URLs ready for crawling (SD empty). Also you can use combineCtrl to see current status of ready queue etc

    (b) `| 7352 | HARVPARS 1_wctest | 2006-07-14 17:00:59 | M500; urlid=1; netlocid=1; http:/`

    Crawler process 7352 got a url (http://www.shanghaidaily.com/) to check (1_wctest is a just a name non significant) M500 is a sequence number for an individual crawler starting at 500 and when it reaches 0 this crawler process is killed and another is created. urlid and netlocid are internal identifiers used in the MySQL tables.

    (c) `| 7352 | HARVPARS 1_wctest | 2006-07-14 17:01:10 | M500; RobotRules OK, OK`

    Crawler process have checked that this url (identified earlier in the log by pid=7352 and M500) can be crawled according to the Robot Exclusin protocol.

    (d) `| 7352 | HARVPARS 1_wctest | 2006-07-14 17:01:10 | M500; HTTP(200 = "OK")  => OK`

    It has fetched the page (identified earlier in the log by pid=7352 and M500) OK

    (e) `| 7352 | HARVPARS 1_wctest | 2006-07-14 17:01:10 | M500; Doing: text/html;200;0F0610`

    It is processing the page (in the format text/html) to see if it is of topical interest 0F061033DAF69587170F8E285E950120 is the MD5 checksum of the page

17. In fact , I want to know which crawed urls are corresponding to the certain topic class such as CP.Aldrovanda . Can you tell me how can I know ?

    You have to get into the raw MySQL database and perform a query like

SELECT urls.urlstr FROM urls,recordurl,topic WHERE urls.urlid=recordurl.urlid AND recordurl.recordid=topic.recordid AND topic.notation='CP.Aldrovanda';

Table urls contain all URLs seen. Table recordurl connect urlid to recordid. recordid is used in all tables with data from the crawled Web pages.

18. what is the meaning of the item "ALL" in the notation column of the topic table ?

    If you use multiple topics in your topic-defintion (ie the string after '=') then all the relevant topic scores for this page is summed and given the topic notation 'ALL'.

    Just disregard it if you only use one topic-class.

19. Combine should crawl all pages underneath, but not go outside the domain (i.e. going to www.yahoo.com) but also not going higher in position (i.e. www.geocities.com/boulevar/atlanta/index.h

    Is it possible to set up Combine like this?

    Yes, change the <allow>-part of your configuration file combine.cfg to select what URL's should be allowed for crawling (by default everything is allowed). See also section 4.3.

    So change

```
<allow>
#Allow crawl of URLs or hostnames that matches these regular expressions
HOST: .*$
</allow>
```

    to something like

```
<allow>
#Allow crawl of URLs or hostnames that matches these regular expressions
URL http:\/\/www\.geocities\.com\/boulevard\/newyork\/
</allow>
```

    (the backslashes are needed since these patterns are in fact Perl regular expressions)

9 Configuration variables

9.1 Name/value configuration variables

9.1.1 AutoRecycleLinks

Default value = 1

Description: Enable(1)/disable(0) automatic recycling of new links

Used by: SD_SQL.pm

9.1.2 baseConfigDir

Default value = /etc/combine

Description: Base directory for configuration files; initialized by Config.pm

Used by: FromHTML.pm; combineExport

Set by: Config.pm

9.1.3 classifyPlugIn

Default value = Combine::Check_record

Description: Which topic classification PlugIn module algorithm to use
Combine::Check_record and Combine::PosCheck_record included by default
see classifyPlugInTemplate.pm and documentation to write your own

Used by: combine

9.1.4 configDir

Default value = NoDefaultValue

Description: Directory for job specific configuration files; taken from 'jobname'

Used by: Check_record.pm; combineUtil; PosCheck_record.pm

Set by: Config.pm

9.1.5 doAnalyse

Default value = 1

Description: Enable(1)/disable(0) analysis of genre, language

Used by: combine

9.1.6 doCheckRecord

Description: Enable(1)/disable(0) topic classification (focused crawling)
Generated by combineINIT based on –topic parameter

Used by: combine

9.1.7 doOAI

Default value = 1

Description: Use(1)/do not use(0) OAI record status keeping in SQL database

Used by: MySQLhdb.pm

9.1.8 extractLinksFromText

Default value = 1

Description: Extract(1)/do not extract(0) links from plain text

Used by: combine

9.1.9 HarvesterMaxMissions

Default value = 500

Description: Number of pages to process before restarting the harvester

Used by: combine

9.1.10 HarvestRetries

Default value = 5

Used by: combine

9.1.11 httpProxy

Default value = NoDefaultValue

Description: Use a proxy server if this is defined (default no proxy)

Used by: UA.pm

9.1.12 LogHandle

Used by: Check_record.pm; FromHTML.pm; PosCheck_record.pm

Set by: combine

9.1.13 Loglev

Default value = 10

Description: Logging level (0 (least) - 10 (most))

Used by: combine

9.1.14 maxUrlLength

Default value = 250

Description: Maximum length of a URL; longer will be silently discarded

Used by: selurl.pm

9.1.15 MySQLdatabase

Default value = NoDefaultValue

Description: Identifies MySQL database name, user and host

Used by: Config.pm

9.1.16 MySQLhandle

Used by: combineUtil; LogSQL.pm; combine; RobotRules.pm; combineExport; SD_SQL.pm; XWI2XML.pm; MySQLhdb.pm

Set by: Config.pm

9.1.17 Operator-Email

Default value = "YourEmailAdress@YourDomain"

Description: Please change

Used by: RobotRules.pm; UA.pm

9.1.18 Password

Default value = "XxXxyYzZ"

Description: Password not used yet. (Please change)

9.1.19 saveHTML

Default value = 1

Description: Store(1)/do not store(0) the raw HTML in the database

Used by: MySQLhdb.pm

9.1.20 SdqRetries

Default value = 5

9.1.21 SummaryLength

Description: How long the summary should be. Use 0 to disable the summarization code

Used by: FromHTML.pm

9.1.22 UAtimeout

Default value = 30

Description: Time in seconds to wait for a server to respond

Used by: UA.pm

9.1.23 UserAgentFollowRedirects

Description: User agent handles redirects (1) or treat redirects as new links (0)

Used by: UA.pm

9.1.24 UserAgentGetIfModifiedSince

Default value = 1

Description: If we have seen this page before use Get-If-Modified (1) or not (0)

Used by: UA.pm

9.1.25 useTidy

Default value = 1

Description: Use(1)/do not use(0) Tidy to clean the HTML before parsing it

Used by: FromHTML.pm

9.1.26 WaitIntervalExpirationGuaranteed

Default value = 315360000

Used by: UA.pm

9.1.27 WaitIntervalHarvesterLockNotFound

Default value = 2592000

Used by: combine

9.1.28 WaitIntervalHarvesterLockNotModified

Default value = 2592000

Used by: combine

9.1.29 WaitIntervalHarvesterLockRobotRules

Default value = 2592000

Used by: combine

9.1.30 WaitIntervalHarvesterLockSuccess

Default value = 1000000

Description: Time in seconds after succesfull download before allowing a page to be downloaded again (around 11 days)

Used by: combine

9.1.31 WaitIntervalHarvesterLockUnavailable

Default value = 86400

Used by: combine

9.1.32 WaitIntervalHost

Default value = 60

Description: Minimum time between accesses to the same host. Must be positive

Used by: SD_SQL.pm

9.1.33 WaitIntervalRrdLockDefault

Default value = 86400

Used by: RobotRules.pm

9.1.34 WaitIntervalRrdLockNotFound

Default value = 345600

Used by: RobotRules.pm

9.1.35 WaitIntervalRrdLockSuccess

Default value = 345600

Used by: RobotRules.pm

9.1.36 WaitIntervalSchedulerGetJcf

Default value = 20

Description: Time in seconds to wait before making a new reschedule if a reschedule results in an empty ready que

Used by: combine

9.1.37 ZebraHost

Default value = NoDefaultValue

Description: Direct connection to Zebra indexing - for SearchEngine-in-a-box (default no connection)

Used by: MySQLhdb.pm

9.2 Complex configuration variables

9.2.1 allow

Description: use either URL or HOST: (obs ':') to match regular expressions to either the full URL or the HOST part of a URL.
Allow crawl of URLs or hostnames that matches these regular expressions

Used by: selurl.pm

9.2.2 binext

Description: Extensions of binary files

Used by: UA.pm

9.2.3 converters

Description: Configure which converters can be used to produce a XWI object
Format:
1 line per entry
each entry consists of 3 ';' separated fields
Entries are processed in order and the first match is executed
external converters have to be found via PATH and executable to be considered a match
the external converter command should take a filename as parameter and convert that file
the result should be comming on STDOUT
mime-type ; External converter command ; Internal converter

Used by: UA.pm; combine

9.2.4 exclude

Description: Exclude URLs or hostnames that matches these regular expressions
default: CGI and maps
default: binary files
default: Unparsable documents
default: images
default: other binary formats
more excludes in the file config_exclude (automatically updated by other programs)

Used by: selurl.pm

9.2.5 serveralias

Description: List of servernames that are aliases are in the file ./config_serveralias
(automatically updated by other programs)
use one server per line
example
www.100topwetland.com www.100wetland.com

means that www.100wetland.com is replaced by www.100topwetland.com during URL normalization

9.2.6 sessionids

Description: patterns to recognize and remove sessionids in URLs

9.2.7 url

Description: url is just a conatiner for all URL related configuration patterns

Used by: Config.pm; selurl.pm

10 Module dependences

10.1 Programs

10.1.1 combine

Uses: Combine::Config; Combine::XWI; Combine::UA; Combine::RobotRules; Combine::LogSQL; Combine::FromHTML; Combine::FromImage; Combine::FromTeX; Combine::DataBase; HTTP::Date; HTTP::Status; URI::URL; Getopt::Long; Combine::SD_SQL; Lingua::Identify;

10.1.2 combineCtrl

Uses: Getopt::Long; Combine::SD_SQL; Combine::Config;

10.1.3 combineExport

Uses: Combine::MySQLhdb; Combine::Config; Combine::XWI2XML; DBI; HTTP::Date; Encode; Getopt::Long; Alvis::Pipeline; XML::LibXSLT; XML::LibXML;

10.1.4 combineINIT

Uses: Getopt::Long; Combine::Config; DBI;

10.1.5 combineUtil

Uses: Getopt::Long; Combine::Config; Combine::SD_SQL; Combine::MySQLhdb; Combine::MySQLhdb; Net::hostent;

10.2 Library modules

10.2.1 Check_record.pm

Uses: Combine::XWI; Combine::LoadTermList; Combine::Matcher; Combine::Config;

Used by:

10.2.2 CleanXML2CanDoc.pm

Uses: Alvis::Canonical;

Used by: Combine::XWI2XML;

10.2.3 Config.pm

Uses: Config::General; DBI;

Used by: combineCtrl; combine; combineExport; combineINIT; combineUtil; Combine::Check_record; Combine::FromHTML; Combine::LogSQL; Combine::MySQLhdb; Combine::PosCheck_record; Combine::RobotRules; Combine::SD_SQL; Combine::UA; Combine::XWI2XML; Combine::selurl;

10.2.4 DataBase.pm

Uses: Combine::MySQLhdb; Combine::selurl;

Used by: combine;

10.2.5 FromHTML.pm

Uses: Combine::Config; HTTP::Date; URI; URI::Escape; HTML::Entities; Encode; HTML::Tidy; Combine::HTMLExtractor;

Used by: combine;

10.2.6 FromImage.pm

Uses: Image::ExifTool;

Used by: combine;

10.2.7 HTMLExtractor.pm

Uses: HTML::TokeParser; URI; Data::Dumper;

Used by: Combine::FromHTML;

10.2.8 LoadTermList.pm

Uses: DBI; Lingua::Stem;

Used by: Combine::Check_record; Combine::PosCheck_record;

10.2.9 LogSQL.pm

Uses: Combine::Config;

Used by: combine;

10.2.10 Matcher.pm

Uses: HTML::Entities;

Used by: Combine::Check_record;

10.2.11 MySQLhdb.pm

Uses: Combine::XWI; HTTP::Date; Encode; Combine::Config; Combine::selurl; Combine::Zebra;

Used by: combineExport; combineUtil; combineUtil; Combine::DataBase;

10.2.12 PosCheck_record.pm

Uses: Combine::LoadTermList; Combine::PosMatcher; Combine::Config;

Used by:

10.2.13 PosMatcher.pm

Uses: HTML::Entities;

Used by: Combine::PosCheck_record;

10.2.14 RobotRules.pm

Uses: Combine::Config; Combine::UA;

Used by: combine;

10.2.15 SD_SQL.pm

Uses: Combine::Config; Combine::selurl; DBI;

Used by: combineCtrl; combine; combineUtil;

10.2.16 UA.pm

Uses: Combine::Config; LWP::UserAgent; HTTP::Date; Digest::MD5;

Used by: combine; Combine::RobotRules;

10.2.17 XWI.pm

Uses: HTML::Entities;

Used by: combine; Combine::Check_record; Combine::MySQLhdb; Combine::XWI2XML;

10.2.18 XWI2XML.pm

Uses: Combine::XWI; Encode; Combine::Config; Compress::Zlib; MIME::Base64; Combine::CleanXML2CanDoc;

Used by: combineExport; Combine::Zebra;

10.2.19 Zebra.pm

Uses: Combine::XWI2XML; ZOOM;

Used by: Combine::MySQLhdb;

10.2.20 selurl.pm

Uses: URI; Combine::Config;

Used by: Combine::DataBase; Combine::MySQLhdb; Combine::SD_SQL;

10.3 External modules

These are the (non base) Perl modules Combine depend on. The modules marked with a '*' are not critical.

```
Alvis::Canonical
Alvis::Pipeline *
Compress::Zlib
Config::General
DBI
Data::Dumper *
Digest::MD5
Encode
Getopt::Long
HTML::Entities
HTML::Tidy *
HTML::TokeParser
HTTP::Date
HTTP::Status
Image::ExifTool
LWP::UserAgent
Lingua::Identify
Lingua::Stem
MIME::Base64
Net::hostent
URI
URI::Escape
URI::URL
XML::LibXML
XML::LibXSLT
ZOOM
```

A APPENDIX

A.1 Simple installation test

The following simple script is available in the `doc/InstallationTest.pl` file. It must be run as 'root' and tests that basic functions of the Combine installation works.

Basicly it creates and initializes a new jobname, crawls one specific test page and exports it as XML. This XML is then compared to a correct XML-record for that page.

A.1.1 InstallationTest.pl

```
use strict;
if ( $> != 0 ) {
    die("You have to run this test as root");
}

my $orec='';
while (<DATA>) { chop; $orec .= $_; }

$orec =~ s|<checkedDate>.*</checkedDate>||;
$orec =~ tr/\n\t //d;

my $olen=length($orec);
my $onodes=0;
while ( $orec =~ m/</g ) { $onodes++; }
print "ORIG Nodes=$onodes; Len=$olen\n";

our $jobname;
require './t/defs.pm';

system("combineINIT --jobname $jobname --topic /etc/combine/Topic_carnivor.txt >& /dev/null")

system("combine --jobname $jobname --harvest http://combine.it.lth.se/CombineTests/Installati
open(REC,"combineExport --jobname $jobname |");
my $rec='';
while (<REC>) { chop; $rec .= $_; }
close(REC);
$rec =~ s|<checkedDate>.*</checkedDate>||;
$rec =~ tr/\n\t //d;

my $len=length($rec);
my $nodes=0;
while ( $rec =~ m/</g ) { $nodes++; }
print "NEW Nodes=$nodes; Len=$len\n";

my $OK=0;

if ($onodes == $nodes) { print "Number of XML nodes match\n"; }
else { print "Number of XML nodes does NOT match\n"; $OK=1; }
```

```
if ($olen == $len) {
  print "Size of XML match\n";
} else {
  $orec =~  s|<originalDocument.*</originalDocument>||s;
  $rec =~  s|<originalDocument.*</originalDocument>||s;
  if (length($orec) == length($rec)) { print "Size of XML match (after removal of 'originalDo
  else { print "Size of XML does NOT match\n"; $OK=1; }
}

if (($OK == 0) && ($orec eq $rec)) { print "All tests OK\n"; }
else { print "There might be some problem with your Combine Installation\n"; }

__END__
<?xml version="1.0" encoding="UTF-8"?>
<documentCollection version="1.1" xmlns="http://alvis.info/enriched/">
<documentRecord id="17E4E1138D15C3C19866D0C563E7F35F">
<acquisition>
<acquisitionData>
<modifiedDate>2006-05-19 9:57:03</modifiedDate>
<checkedDate>2006-10-03 9:06:42</checkedDate>
<httpServer>Apache/1.3.29 (Debian GNU/Linux) PHP/4.3.3</httpServer>
<urls>
    <url>http://combine.it.lth.se/CombineTests/InstallationTest.html</url>
  </urls>
</acquisitionData>
<originalDocument mimeType="text/html" compression="gzip" encoding="base64" charSet="UTF-8">
H4sIAAAAAAAAA4WQsU7DMBCG9zzF4bmpBV2QcDKQVKJSKR2CEKObXBSrjm3sSyFvT0yCQGJgusG/
//u+E1flU1G9HrfwUD3u4fh8v98VwFLOXzYF52VVzg+b9Q3n2wPLE9FRr+NA2UyDFGnMdyaQ1FqS
sgYIA0FrPRS2PymDgs+hRPRIEozsMWNnHN+tbwKD2hpCQxkrpDfqYr0dAjgtDYUVlN4G9HIFB3RT
qMPAvns6Ipfi26Au09e5I61Gh78aCT+IR947qDvpA1I2UJvexg6+CJxsM0ad6/8kpkQiXB5XSWUC
BNsj/GGG4LBWrarhSw+0OiOIidZjmzGPeh15WL6ICS7zFUjT/AiuBXeRbwHj870/AeRYaTupAQAA
</originalDocument>
<canonicalDocument>
  <section>
    <section title="Installation test for Combine">
      <section>Installation test for Combine</section>
      <section>Contains some Carnivorous plant specific words like <ulink url="rel.html">Dros
<metaData>
    <meta name="title">Installation test for Combine</meta>
    <meta name="dc:format">text/html</meta>
    <meta name="dc:format">text/html; charset=iso-8859-1</meta>
    <meta name="dc:subject">Carnivorous plants</meta>
    <meta name="dc:subject">Drosera</meta>
    <meta name="dc:subject">Nepenthes</meta>
  </metaData>
<links>
    <outlinks>
      <link type="a">
```

```
        <anchorText>Drosera</anchorText>
        <location>http://combine.it.lth.se/CombineTests/rel.html</location>
      </link>
    </outlinks>
  </links>
<analysis>
<property name="topLevelDomain">se</property>
<property name="univ">1</property>
<property name="language">en</property>
<topic absoluteScore="1000" relativeScore="110526">
    <class>ALL</class>
  </topic>
<topic absoluteScore="375" relativeScore="41447">
    <class>CP.Drosera</class>
    <terms>drosera</terms>
  </topic>
<topic absoluteScore="375" relativeScore="41447">
    <class>CP.Nepenthes</class>
    <terms>nepenthe</terms>
  </topic>
<topic absoluteScore="250" relativeScore="27632">
    <class>CP</class>
    <terms>carnivorous plant</terms>
    <terms>carnivor</terms>
  </topic>
</analysis>
</acquisition>
</documentRecord>

</documentCollection>
```

A.2 Example topic filter plug in

This example gives more details on how to write a topic filter Plug-In.

A.2.1 classifyPlugInTemplate.pm

```
#Template for writing a classify PlugIn for Combine
#See documentation at http://combine.it.lth.se/documentation/

package classifyPlugInTemplate; #Change to your own module name

use Combine::XWI; #Mandatory
use Combine::Config; #Optional if you want to use the Combine configuration system

#API:
#  a subroutine named 'classify' taking a XWI-object as in parameter
#    return values: 0/1
#        0: record fails to meet the classification criteria, ie ignore this record
```

```
#        1: record is OK and should be stored in the database, and links followed by the crawl
sub classify {
  my ($self,$xwi) = @_;

  #utility routines to extract information from the XWI-object
  #URL (can be several):
   # $xwi->url_rewind;
   # my $url_str="";
   # my $t;
   # while ($t = $xwi->url_get) { $url_str .= $t . ", "; }

  #Metadata:
   #  $xwi->meta_rewind;
   #  my ($name,$content);
   #  while (1) {
   #    ($name,$content) = $xwi->meta_get;
   #    last unless $name;
   #    next if ($name eq 'Rsummary');
   #    next if ($name =~ /^autoclass/);
   #    $meta .= $content . " ";
   #  }

  #Title:
   #  $title = $xwi->title;

  #Headings:
   #  $xwi->heading_rewind;
   #  my $this;
   #  while (1) {
   #    $this = $xwi->heading_get or last;
   #    $head .= $this . " ";
   #  }

  #Text:
   #  $this = $xwi->text;
   #  if ($this) {
   #    $text = $$this;
   #  }

################################
#Apply your classification algorithm here
#  assign $result a value (0/1)
################################

  #utility routines for saving detailed results (optional) in the database. These data may ap
  # in exported XML-records

  #Topic takes 4 parameters
```

```
  # $xwi->topic_add(topic_class_notation, topic_absolute_score, topic_normalized_score, topic
  #  topic_class_notation, topic_terms, and algorithm_id are strings
  #    max length topic_class_notation: 50, algorithm_id: 25
  #  topic_absolute_score, and topic_normalized_score are integers
  #  topic_normalized_score and topic_terms are optional and may be replaced with 0, '' respe

  #Analysis takes 2 parameters
  # $xwi->robot_add(name,value);
  # both are strings with max length name: 15, value: 20

    # return true (1) if you want to keep the record
    # otherwise return false (0)

  return $result;
}

1;
```

A.3 Default configuration files

A.3.1 Global

```
#@#Default configuration values Combine system

#Direct connection to Zebra indexing - for SearchEngine-in-a-box (default no connection)
#@#ZebraHost = NoDefaultValue
ZebraHost =

#Use a proxy server if this is defined (default no proxy)
#@#httpProxy = NoDefaultValue
httpProxy =

#Enable(1)/disable(0) automatic recycling of new links
AutoRecycleLinks = 1

#User agent handles redirects (1) or treat redirects as new links (0)
UserAgentFollowRedirects = 0

#Number of pages to process before restarting the harvester
HarvesterMaxMissions = 500

#Logging level (0 (least) - 10 (most))
Loglev = 10

#Enable(1)/disable(0) analysis of genre, language
doAnalyse = 1

#How long the summary should be. Use 0 to disable the summarization code
SummaryLength   = 0
```

```
#Store(1)/do not store(0) the raw HTML in the database
saveHTML = 1

#Use(1)/do not use(0) Tidy to clean the HTML before parsing it
useTidy = 1

#Use(1)/do not use(0) OAI record status keeping in SQL database
doOAI = 1

#Extract(1)/do not extract(0) links from plain text
extractLinksFromText = 1

#Enable(1)/disable(0) topic classification (focused crawling)
#Generated by combineINIT based on --topic parameter
doCheckRecord = 0

#Which topic classification PlugIn module algorithm to use
#Combine::Check_record and Combine::PosCheck_record included by default
#see classifyPlugInTemplate.pm and documentation to write your own
classifyPlugIn = Combine::Check_record

###Parameters for Std topic classification algorithm
###StdTitleWeight = 10 #
###StdMetaWeight = 4 #
###StdHeadingsWeight = 2 #
###StdCutoffRel = 10 #Class score must be above this % to be counted
###StdCutoffNorm = 0.2 #normalised cutoff for summed normalised score
###StdCutoffTot = 90 #non normalised cutoff for summed total score

###Parameters for Pos topic classification algorithm
###PosCutoffRel = 1 #Class score must be above this % to be counted
###PosCutoffNorm = 0.002 #normalised cutoff for summed normalised score
###PosCutoffTot = 1 #non normalised cutoff for summed total score

HarvestRetries                  = 5
SdqRetries                      = 5

#Maximum length of a URL; longer will be silently discarded
maxUrlLength = 250

#Time in seconds to wait for a server to respond
UAtimeout = 30

#If we have seen this page before use Get-If-Modified (1) or not (0)
UserAgentGetIfModifiedSince = 1

WaitIntervalExpirationGuaranteed = 315360000
```

```
WaitIntervalHarvesterLockNotFound = 2592000
WaitIntervalHarvesterLockNotModified = 2592000
WaitIntervalHarvesterLockRobotRules = 2592000
WaitIntervalHarvesterLockUnavailable = 86400
WaitIntervalRrdLockDefault = 86400
WaitIntervalRrdLockNotFound = 345600
WaitIntervalRrdLockSuccess = 345600

#Time in seconds after succesfull download before allowing a page to be downloaded again (aro
WaitIntervalHarvesterLockSuccess = 1000000

#Time in seconds to wait before making a new reschedule if a reschedule results in an empty re
WaitIntervalSchedulerGetJcf = 20

#Minimum time between accesses to the same host. Must be positive
WaitIntervalHost = 60

#Identifies MySQL database name, user and host
MySQLdatabase   = NoDefaultValue

#Base directory for configuration files; initialized by Config.pm
#@#baseConfigDir = /etc/combine

#Directory for job specific configuration files; taken from 'jobname'
#@#configDir = NoDefaultValue

<binext>
#Extensions of binary files
ps
jpg
jpeg
pdf
tif
tiff
mpg
mpeg
mov
wav
au
hqx
gz
z
tgz
exe
zip
sdd
doc
rtf
```

```
shar
mat
raw
wmz
arff
rar
</binext>

<converters>
#Configure which converters can be used to produce a XWI object
#Format:
#  1 line per entry
#  each entry consists of 3 ';' separated fields
#
#Entries are processed in order and the first match is executed
#  external converters have to be found via PATH and executable to be considered a match
#  the external converter command should take a filename as parameter and convert that file
#   the result should be comming on STDOUT
#
# mime-type   ;   External converter command ; Internal converter

text/html ; ; GuessHTML
#Check this
www/unknown ; ; GuessHTML
text/plain ; ; GuessText
text/x-tex ;  tth -g -w1 -r <  ; TeXHTML
application/x-tex ;  tth -g -w1 -r < ; TeXHTML
text/x-tex ; untex -a -e -giso ; TeXText
application/x-tex ; untex -a -e -giso ; TeXText
text/x-tex ;  ; TeX
application/x-tex ; ; TeX
application/pdf ; pdftohtml -i -noframes -nomerge -stdout ; HTML
application/pdf ; pstotext ; Text
application/postscript ; pstotext ; Text
application/msword ; antiword -t ; Text
application/vnd.ms-excel ; xlhtml -fw ; HTML
application/vnd.ms-powerpoint ; ppthtml ; HTML
application/rtf ; unrtf --nopict --html ; HTML
image/gif ; ; Image
image/jpeg ; ; Image
image/tiff ; ; Image
</converters>

<url>
  <exclude>
    #Exclude URLs or hostnames that matches these regular expressions
    #Malformed hostnames
    HOST: http:\/\/\.
```

```
    HOST: \@
  </exclude>
</url>
```

A.3.2 Job specific

```
#Please change
Operator-Email      = "YourEmailAdress@YourDomain"

#Password not used yet. (Please change)
Password    = "XxXxyYzZ"

<converters>
#Configure which converters can be used to produce a XWI object
#Format:
#  1 line per entry
#  each entry consists of 3 ';' separated fields
#
#Entries are processed in order and the first match is executed
#  external converters have to be found via PATH and executable to be considered a match
#  the external converter command should take a filename as parameter and convert that file
#   the result should be comming on STDOUT
#
# mime-type   ;   External converter command ; Internal converter

application/pdf ; MYpdftohtml -i -noframes -nomerge -stdout ; HTML
</converters>

<url>
#List of servernames that are aliases are in the file ./config_serveralias
#    (automatically updated by other programs)
#use one server per line
#example
#www.100topwetland.com  www.100wetland.com
#  means that www.100wetland.com is replaced by www.100topwetland.com during URL normalizatio
<serveralias>
<<include config_serveralias>>
</serveralias>

#use either URL or HOST: (obs ':') to match regular expressions to
# either the full URL or the HOST part of a URL.
<allow>
#Allow crawl of URLs or hostnames that matches these regular expressions
HOST: .*$
</allow>

<exclude>
#Exclude URLs or hostnames that matches these regular expressions
```

```
# default: CGI and maps
URL cgi-bin|htbin|cgi|\?|\.map$|_vti_

# default: binary files
URL \.exe$|\.zip$|\.tar$|\.tgz$|\.gz$|\.hqx$|\.sdd$|\.mat$|\.raw$
URL \.EXE$|\.ZIP$|\.TAR$|\.TGZ$|\.GZ$|\.HQX$|\.SDD$|\.MAT$|\.RAW$

# default: Unparsable documents
URL \.shar$|\.rmx$|\.rmd$|\.mdb$
URL \.SHAR$|\.RMX$|\.RMD$|\.MDB$

# default: images
URL \.gif$|\.jpg$|\.jpeg$|\.xpm$|\.tif$|\.tiff$|\.mpg$|\.mpeg$|\.mov$|\.wav$|\.au$|\.pcx$|\.x
URL \.GIF$|\.JPG$|\.JPEG$|\.XPM$|\.TIF$|\.TIFF$|\.MPG$|\.MPEG$|\.MOV$|\.WAV$|\.AU$|\.PCX$|\.X

# default: other binary formats
URL \.pdb$|\.class$|\.ica$|\.ram$|.wmz$|.arff$|.rar$|\.vo$|\.fig$
URL \.PDB$|\.CLASS$|\.ICA$|\.RAM$|.WMZ$|.ARFF$|.RAR$|\.VO$|\.FIG$

#more excludes in the file config_exclude (automatically updated by other programs)
<<include config_exclude>>
</exclude>
<sessionids>
#patterns to recognize and remove sessionids in URLs
sessionid
lsessionid
jsessionid
SID
PHPSESSID
SessionID
BV_SessionID
</sessionids>
#url is just a conatiner for all URL related configuration patterns
</url>
```

A.4 SQL database

A.4.1 Create database

```
DROP DATABASE IF EXISTS $database;
CREATE DATABASE $database DEFAULT CHARACTER SET utf8;
USE $database;
```

A.4.2 Creating MySQL tables

```
All tables use UTF-8

Summary tables '^'=primary key, '*'=key:
```

```
TABLE hdb: recordid^, type, dates, server, title, ip, ...
TABLE links: recordid*, mynetlocid*, urlid*, netlocid*, linktype, anchor  (netlocid for urlid
TABLE meta: recordid*,  name, value
TABLE html: recordid^, html
TABLE analys: recordid*, name, value
TABLE topic: recordid*, notation*, absscore, relscore, terms, algorithm

(TABLE netlocalias: netlocid*, netlocstr^)
(TABLE urlalias: urlid*, urlstr^)
TABLE topichierarchy: node^, father*, notation*, caption, level
TABLE netlocs: netlocid^, netlocstr^, retries
TABLE urls: netlocid*, urlid^, urlstr^, path
TABLE urldb: netlocid*, urlid^, urllock, harvest*, retries, netloclock
TABLE newlinks urlid^, netlocid
TABLE recordurl: recordid*, urlid^, lastchecked, md5*, fingerprint*^
TABLE admin: status, queid, schedulealgorithm
TABLE log: pid, id, date, message
TABLE que: queid^, urlid, netlocid
TABLE robotrules: netlocid*, rule, expire
TABLE oai: recordid, md5^, date*, status
TABLE exports: host, port, last
```

A.4.3 Data tables

```
CREATE TABLE hdb (
  recordid int(11) NOT NULL default '0',
  type varchar(50) default NULL,
  title text,
  mdate timestamp NOT NULL,
  expiredate datetime default NULL,
  length int(11) default NULL,
  server varchar(50) default NULL,
  etag varchar(25) default NULL,
  nheadings int(11) default NULL,
  nlinks int(11) default NULL,
  headings mediumtext,
  ip mediumblob,
  PRIMARY KEY  (recordid)
) ENGINE=MyISAM AVG_ROW_LENGTH = 20000 MAX_ROWS = 10000000 DEFAULT CHARACTER SET=utf8;

CREATE TABLE html (
  recordid int(11) NOT NULL default '0',
  html mediumblob,
  PRIMARY KEY  (recordid)
) ENGINE=MyISAM AVG_ROW_LENGTH = 20000 MAX_ROWS = 10000000 DEFAULT CHARACTER SET=utf8;

CREATE TABLE links (
  recordid int(11) NOT NULL default '0',
```

```
  mynetlocid int(11) default NULL,
  urlid int(11) default NULL,
  netlocid int(11) default NULL,
  anchor text,
  linktype varchar(50) default NULL,
  KEY recordid (recordid),
  KEY urlid (urlid),
  KEY mynetlocid (mynetlocid),
  KEY netlocid (netlocid)
) ENGINE=MyISAM MAX_ROWS = 1000000000 DEFAULT CHARACTER SET=utf8;

CREATE TABLE meta (
  recordid int(11) NOT NULL default '0',
  name varchar(50) default NULL,
  value text,
  KEY recordid (recordid)
) ENGINE=MyISAM MAX_ROWS = 1000000000 DEFAULT CHARACTER SET=utf8;

CREATE TABLE analys (
  recordid int(11) NOT NULL default '0',
  name varchar(15) NOT NULL,
  value varchar(20),
  KEY recordid (recordid)
) ENGINE=MyISAM DEFAULT CHARACTER SET=utf8;

CREATE TABLE topic (
  recordid int(11) NOT NULL default '0',
  notation varchar(50) default NULL,
  abscore int(11) default NULL,
  relscore int(11) default NULL,
  terms text default NULL,
  algorithm varchar(25),
  KEY notation (notation),
  KEY recordid (recordid)
) ENGINE=MyISAM DEFAULT CHARACTER SET=utf8;
```

A.4.4 Administrative tables

```
CREATE TABLE netlocalias (
  netlocid int(11),
  netlocstr varchar(150) NOT NULL,
  KEY netlocid (netlocid),
  PRIMARY KEY netlocstr (netlocstr)
) ENGINE=MyISAM DEFAULT CHARACTER SET=utf8;

CREATE TABLE urlalias (
  urlid int(11),
  urlstr tinytext,
  KEY urlid (urlid),
```

```
  PRIMARY KEY urlstr (urlstr(255))
) ENGINE=MyISAM DEFAULT CHARACTER SET=utf8;

   topichierarchy have to initialized manually


CREATE TABLE topichierarchy (
  node int(11) NOT NULL DEFAULT '0',
  father int(11) DEFAULT NULL,
  notation varchar(50) NOT NULL DEFAULT '',
  caption varchar(255) DEFAULT NULL,
  level int(11) DEFAULT NULL,
  PRIMARY KEY node (node),
  KEY father (father),
  KEY notation (notation)
) ENGINE=MyISAM DEFAULT CHARACTER SET=utf8;

CREATE TABLE netlocs (
  netlocid int(11) NOT NULL auto_increment,
  netlocstr varchar(150) NOT NULL,
  retries int(11) NOT NULL DEFAULT 0,
  PRIMARY KEY (netlocstr),
  UNIQUE INDEX netlockid (netlocid)
) ENGINE=MyISAM DEFAULT CHARACTER SET=utf8;

CREATE TABLE urls (
  netlocid int(11) NOT NULL DEFAULT '0',
  urlid int(11) NOT NULL auto_increment,
  urlstr tinytext,
  path tinytext,
  PRIMARY KEY urlstr (urlstr(255)),
  INDEX netlocid (netlocid),
  UNIQUE INDEX urlid (urlid)
) ENGINE=MyISAM MAX_ROWS = 1000000000 DEFAULT CHARACTER SET=utf8;

CREATE TABLE urldb (
  netlocid int(11) NOT NULL default '0',
  netloclock int(11) NOT NULL default '0',
  urlid int(11) NOT NULL default '0',
  urllock int(11) NOT NULL default '0',
  harvest tinyint(1) NOT NULL default '0',
  retries int(11) NOT NULL default '0',
  PRIMARY KEY  (urlid),
  KEY netlocid (netlocid),
  KEY harvest (harvest)
) ENGINE=MyISAM DEFAULT CHARACTER SET=utf8;

CREATE TABLE newlinks (
  urlid int(11) NOT NULL,
```

```
  netlocid int(11) NOT NULL,
  PRIMARY KEY  (urlid)
) ENGINE=MyISAM DEFAULT CHARACTER SET=utf8;

CREATE TABLE recordurl (
  recordid int(11) NOT NULL auto_increment,
  urlid int(11) NOT NULL default '0',
  lastchecked timestamp NOT NULL,
  md5 char(32),
  fingerprint char(50),
  KEY md5 (md5),
  KEY fingerprint (fingerprint),
  PRIMARY KEY (urlid),
  KEY recordid (recordid)
) ENGINE=MyISAM DEFAULT CHARACTER SET=utf8;

CREATE TABLE admin (
  status enum('closed','open','paused','stopped') default NULL,
  schedulealgorithm enum('default','bigdefault','advanced') default 'default',
  queid int(11) NOT NULL default '0'
) ENGINE=MEMORY DEFAULT CHARACTER SET=utf8;

   Initialise admin to 'open' status
INSERT INTO admin VALUES ('open','default',0)


CREATE TABLE log (
  pid int(11) NOT NULL default '0',
  id varchar(50) default NULL,
  date timestamp NOT NULL,
  message varchar(255) default NULL
) ENGINE=MyISAM DEFAULT CHARACTER SET=utf8;

CREATE TABLE que (
  netlocid int(11) NOT NULL default '0',
  urlid int(11) NOT NULL default '0',
  queid int(11) NOT NULL auto_increment,
  PRIMARY KEY  (queid)
) ENGINE=MEMORY DEFAULT CHARACTER SET=utf8;

CREATE TABLE robotrules (
  netlocid int(11) NOT NULL default '0',
  expire int(11) NOT NULL default '0',
  rule varchar(255) default '',
  KEY netlocid (netlocid)
) ENGINE=MyISAM DEFAULT CHARACTER SET=utf8;

CREATE TABLE oai (
  recordid int(11) NOT NULL default '0',
```

```
  md5 char(32),
  date timestamp,
  status enum('created', 'updated', 'deleted'),
  PRIMARY KEY (md5),
  KEY date (date)
) ENGINE=MyISAM DEFAULT CHARACTER SET=utf8;

CREATE TABLE exports (
  host varchar(30),
  port int,
  last timestamp DEFAULT '1999-12-31'
) ENGINE=MyISAM DEFAULT CHARACTER SET=utf8;
```

A.4.5 Create user dbuser with required priviligies

```
GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,CREATE TEMPORARY TABLES,
   ALTER,LOCK TABLES ON $database.* TO $dbuser;

GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,CREATE TEMPORARY TABLES,
   ALTER,LOCK TABLES ON $database.* TO $dbuser\@localhost;
```

A.5 Manual pages

A.5.1 combineCtrl

NAME combineCtrl - controls a Combine crawling job

SYNOPSIS combineCtrl <action> –jobname <name>
where action can be one of start, kill, load, recyclelinks, reharvest, stat, howmany, records, hosts, initMemoryTables, open, stop, pause, continue

OPTIONS AND ARGUMENTS jobname is used to find the appropriate configuration (mandatory)

Actions starting/killing crawlers

start

takes an optional switch `-harvesters n` where `n` is the number of crawler processes to start

kill

kills all active crawlers (and their associated combineRun monitors) for jobnam

Actions loading or recycling URLs for crawling

load

Read a list of URLs from STDIN (one per line) and schedules them for crawling

recyclelinks

Schedule all newly found (since last invocation of recyclelinks) links in crawled pages for crawling

reharvest

Schedules all pages in the database for crawling again (in order to check if they have changed)

Actions for controlling scheduling of URLs

open

opens database for URL scheduling (maybe after a stop)

stop

stops URL scheduling

pause

pauses URL scheduling

continue

continues URL scheduling after a pause

Misc actions

stat

prints out rudimentary status of the ready que (ie eligible now) of URLs to be crawled

howmany

prints out rudimentary status of all URLs to be crawled

records

prints out the number of ercords in the SQL database

hosts

prints out rudimentary status of all hosts that have URLs to be crawled

initMemoryTables

initializes the administrative MySQL tables that are kept in memory

DESCRIPTION Implements various control functionality to administer a crawling job, like starting and stoping crawlers, injecting URLs into the crawl que, scheduling newly found links for crawling, controlling scheduling, etc.

This is the preferred way of controling a crawl job.

EXAMPLES

`echo 'http://www.yourdomain.com/' | combineCtrl load -jobname aatest`

Seed the crawling job `aatest` with a URL

`combineCtrl start -jobname aatest -harvesters 3`

Start 3 crawling processes for job `aatest`

`combineCtrl recyclelinks -jobname aatest`

Schedule all new links crawling

`combineCtrl stat -jobname aatest`

See how many URLs that are eligible for crawling right now.

SEE ALSO combine
Combine configuration documentation in */usr/share/doc/combine/*.

AUTHOR Anders Ardö, <anders.ardo@it.lth.se>

A.5.2 combine

NAME combine - main crawling machine in the Combine system

SYNOPSIS combine –jobname <name> –logname <id>

OPTIONS AND ARGUMENTS jobname is used to find the appropriate configuration (mandatory)
logname is used as identifier in the log (in MySQL table log)

DESCRIPTION Does crawling, parsing, optional topic-check and stores in MySQL database Normally started with the `combineCtrl` command. Briefly it get's an URL from the MySQL database, which acts as a common coordinator for a Combine job. The Webpage is fetched, provided it passes the robot exclusion protocoll. The HTML ic cleaned using `Tidy` and parsed into metadata, headings, text, links and link achors. Then it is stored (optionaly provided a topic-check is passed to keep the crawler focused) in the MySQL database in a structured form.
A simple workflow for a trivial crawl job might look like:

```
  Initialize database and configuration
combineINIT --jobname aatest
  Enter some seed URLs from a file with a list of URLs
combineCtrl  load --jobname aatest < seedURLs.txt
```

```
  Start 2 crawl processes
combineCtrl  start --jobname aatest --harvesters 2

  For some time occasionally schedule new links for crawling
combineCtrl recyclelinks --jobname aatest
  or look at the size of the ready que
combineCtrl stat --jobname aatest

  When satisfied kill the crawlers
combineCtrl kill --jobname aatest
  Export data records in a highly structured XML format
combineExport --jobname aatest
```

For more complex jobs you have to edit the job configuration file.

SEE ALSO combineINIT, combineCtrl
Combine configuration documentation in */usr/share/doc/combine/*.

AUTHOR Anders Ardö, <anders.ardo@it.lth.se>

COPYRIGHT AND LICENSE Copyright (C) 2005 Anders Ardö
This library is free software; you can redistribute it and/or modify it under the same terms as Perl itself, either Perl version 5.8.4 or, at your option, any later version of Perl 5 you may have available.
See the file LICENCE included in the distribution at http://combine.it.lth.se/

A.5.3 combineExport

NAME combineExport - export records in XML from Combine database

SYNOPSIS combineExport –jobname <name> [–profile alvis|dc|combine –charset utf8|isolatin –number <n> –recordid <n> –md5 <MD5> –pipehost <server> –pipeport <n> –incremental]

OPTIONS AND ARGUMENTS jobname is used to find the appropriate configuration (mandatory)

–profile

Three profiles: alvis, dc, and combine . alvis and combine are similar XML formats.

'alvis' profile format is defined by the Alvis enriched document format DTD. It uses charset UTF-8 per default.

'combine' is more compact with less redundancy.

'dc' is XML encoded Dublin Core data.

–charset

Selects a specific characterset from UTF-8, iso-latin-1 Overrides –profile settings.

–pipehost, –pipeport

Specifies the server-name and port to connect to and export data using the Alvis Pipeline. Exports incrementally, ie all changes since last call to combineExport with the same pipehost and pipeport.

–number

the max number of records to be exported

–recordid

Export just the one record with this recordid

–md5

Export just the one record with this MD5 checksum

–incremental

Exports incrementally, ie all changes since last call to combineExport using –incremental

–xsltscript

Generates records in Combine native format and converts them using this XSLT script before output. See example scripts in /etc/combine/*.xsl

DESCRIPTION

EXAMPLES

```
Export all records in Alvis XML-format to the file recs.xml
  combineExport --jobname atest > recs.xml

Export 10 records to STDOUT
  combineExport --jobname atest --number 10

Export all records in UTF-8 using Combine native format
  combineExport --jobname atest --profile combine --charset utf8 > Zebrarecs.xml

Incremental export of all changes from last call using localhost at port 6234 using the
default profile (Alvis)
  combineExport --jobname atest --pipehost localhost --pipeport 6234
```

SEE ALSO Combine configuration documentation in */usr/share/doc/combine/*.
Alvis XML schema (–profile alvis) at http://project.alvis.info/alvis_docs/enriched-document.xsd

AUTHOR Anders Ardö, <anders.ardo@it.lth.se>

A.5.4 combineRun

NAME combineRun - starts, monitors and restarts a combine harvesting process

SYNOPSIS combineRun <pidfile> <combine command to run>

DESCRIPTION Starts a program and monitors it in order to make sure there is alsways a copy running. If the program dies it will be restarted with the same parameters. Used by `combineCtrl` when starting combine crawling.

SEE ALSO combineCtrl

AUTHOR Anders Ardö, <anders.ardo@it.lth.se>

COPYRIGHT AND LICENSE Copyright (C) 2005 Anders Ardö

This library is free software; you can redistribute it and/or modify it under the same terms as Perl itself, either Perl version 5.8.4 or, at your option, any later version of Perl 5 you may have available.

See the file LICENCE included in the distribution at http://combine.it.lth.se/

A.5.5 combineUtil

NAME combineUtil - various operations on the Combine database

SYNOPSIS combineUtil <action> –jobname <name>

where action can be one of stats, termstat, classtat, sanity, all, serveralias, resetOAI, restoreSanity, deleteNetLoc, deletePath, deleteMD5, deleteRecordid, addAlias

OPTIONS AND ARGUMENTS jobname is used to find the appropriate configuration (mandatory)

Actions listing statistics

stats

Global statistics about the database

termstat

generates statistics about the terms from topic ontology matched in documents (can be long output)

classtat

generates statistics about the topic classes assigned to documents

Actions for sanity controlls

sanity

Performs various sanity checks on the database

restoreSanity

Deletes records which sanity checks finds insane

resetOAI

Removes all history (ie 'deleted' records) from the OAI table. This is done by removing the OAI table and recreating it from the existing database.

Action all Does the statistics generation actions: stats, sanity, classtat, termstat

Actions for deleting records

deleteNetLoc

Deletes all records matching the ','-separated list of server net-locations (server-names optionally with port) in the switch –netlocstr. Net-locations can include SQL wild cards ('%').

deletePath

Deletes all records matching the ','-separated list of URl paths (excluding net-locations) in the switch –pathsubstrs. Paths can include SQL wild cards ('%').

deleteMD5

Delete the record which has the MD5 in switch –md5

deleteRecordid

Delete the record which has the recordid in switch –recordid

Actions for handling server aliases

serverAlias

Detect server aliases in the current database and do a 'addAlias' on each detected alias.

addAlias

Manually add a serveralias to the system. Requires switches –aliases and –preferred

DESCRIPTION Does various statistics generation as well as performing sanity checks on the database

EXAMPLES

```
combineUtil termstat -jobname aatest
```

Generate matched term statistics

SEE ALSO combine

Combine configuration documentation in */usr/share/doc/combine/*.

AUTHOR Anders Ardö, <anders.ardo@it.lth.se>

COPYRIGHT AND LICENSE Copyright (C) 2005 Anders Ardö

This library is free software; you can redistribute it and/or modify it under the same terms as Perl itself, either Perl version 5.8.4 or, at your option, any later version of Perl 5 you may have available.

See the file LICENCE included in the distribution at http://combine.it.lth.se/

A.5.6 Combine::FromHTML

NAME Combine::FromHTML.pm - HTML parser in combine package

AUTHOR Yong Cao <tsao@munin.ub2.lu.se> v0.06 1997-03-19 Anders Ardø 1998-07-18 added <AREA ... HREF=link ...> fixed <A ... HREF=link ...> regexp to be more general Anders Ardö 2002-09-20 added 'a' as a tag not to be replaced with space added removal of Cntrl-chars and some punctuation marks from IP added <style>...</style> as something to be removed before processing beefed up compression of sequences of blanks to include \240 (non-breakable space) changed 'remove head' before text extraction to handle multiline matching (which can be introduced by decoding html entities) added compress blanks and remove CRs to metadata-content Anders Ardö 2004-04 Changed extraction process dramatically

A.5.7 Combine::FromTeX

NAME Combine::FromTeX.pm - TeX parser in combine package

AUTHOR

```
Anders Ardø 2000-06-11
```

A.5.8 Combine::HTMLExtractor

NAME HTMLExtractor

DESCRIPTION Adopted from HTML::LinkExtractor - Extract links from an HTML document by D.H (PodMaster)

AUTHOR Anders Ardo D.H (PodMaster)

LICENSE Copyright (c) 2003 by D.H. (PodMaster). All rights reserved.

This module is free software; you can redistribute it and/or modify it under the same terms as Perl itself. The LICENSE file contains the full text of the license.

A.5.9 Combine::LoadTermList

NAME LoadTermList

DESCRIPTION This a module in the DESIRE automatic classification system. Copyright 1999.

LoadTermList - A class for loading and storing a stoplist with single words a termlist with classifications and weights

```
Subroutines:
  LoadStopWordList(StopWordListFileName)
     loads a list of stopwords, one per line, from
     the file StopWordListFileName.

  EraseStopWordList
     clears the stopword list

Subroutines:
 LoadTermList(TermListFileName) - loads TermClass from file
 LoadTermListStemmed(TermListFileName) - same plus stems terms

Input: A formatted term-list including weights and classifications
 Format:  <weight>: <term_reg_exp>=[<classification>, ]+
 weight can be a positive or negative number
 term_reg_exp can be words, phrases, boolean expressions (with @and
    as operator) on term_reg_exp or Perl regular expressions
```

AUTHOR Anders Ardö <Anders.Ardo@it.lth.se>

COPYRIGHT AND LICENSE Copyright (C) 2005,2006 Anders Ardö

This library is free software; you can redistribute it and/or modify it under the same terms as Perl itself, either Perl version 5.8.4 or, at your option, any later version of Perl 5 you may have available.

See the file LICENCE included in the distribution at http://combine.it.lth.se/

A.5.10 Combine::Matcher

NAME Matcher

DESCRIPTION This a module in the DESIRE automatic classification system. Copyright 1999. Modified in the ALVIS project. Copyright 2004

Exported routines: 1. Fetching text: These routines all extract texts from a document (either a Combine XWI datastructure or a WWW-page identified by a URL. They all return: $meta, $head, $text, $url, $title, $size $meta: Metadata from document $head: Important text from document $text: Plain text from document $url: URL of the document $title: HTML title of the document $size: The size of the document

```
Common input parameters:
     $DoStem: 1=do stemming; 0=no stemming
     $stoplist: object pointer to a LoadTermList object with a stoplist loaded
     $simple: 1=do simple loading; 0=advanced loading (might induce errors)
```

```
getTextXWI
    parameters: $xwi, $DoStem, $stoplist, $simple
      $xwi is a Combine XWI datastructure

getTextURL
   parameters: $url, $DoStem, $stoplist, $simple
      $url is the URL for the page to extract text from
```

2. Term matcher accepts a text as a (reference) parameter, matches each term in Term against text Matches are recorded in an associative array with class as key and summed weight as value. Match parameters: $text, $termlist $text: text to match against the termlist $termlist: object pointer to a LoadTermList object with a termlist loaded output: %score: an associative array with classifications as keys and scores as values

AUTHOR Anders Ardö <anders.ardo@it.lth.se>

COPYRIGHT AND LICENSE Copyright (C) 2005,2006 Anders Ardö

This library is free software; you can redistribute it and/or modify it under the same terms as Perl itself, either Perl version 5.8.4 or, at your option, any later version of Perl 5 you may have available.

See the file LICENCE included in the distribution at http://combine.it.lth.se/

A.5.11 Combine::PosMatcher

NAME PosMatcher

DESCRIPTION This a module in the DESIRE automatic classification system. Copyright 1999.

Exported routines: 1. Fetching text: These routines all extract texts from a document (either a Combine record, a Combine XWI datastructure or a WWW-page identified by a URL. They all return: $meta, $head, $text, $url, $title, $size $meta: Metadata from document $head: Important text from document $text: Plain text from document $url: URL of the document $title: HTML title of the document $size: The size of the document

```
   Common input parameters:
        $DoStem: 1=do stemming; 0=no stemming
        $stoplist: object pointer to a LoadTermList object with a stoplist loaded
        $simple: 1=do simple loading; 0=advanced loading (might induce errors)

 getTextMD5
    parameters: $md5, $hdb_top, $DoStem, $stoplist, $simple
      $md5 is a key into a Combine hdb-directory
      $hdb_top is the path to the top of the Combine hdb-directory

getTextXWI
    parameters: $xwi, $DoStem, $stoplist, $simple
      $xwi is a Combine XWI datastructure
```

```
getTextURL
   parameters: $url, $DoStem, $stoplist, $simple
      $url is the URL for the page to extract text from
```

2. Term matcher accepts a text as a (reference) parameter, matches each term in Term against text Matches are recorded in an associative array with class as key and summed weight as value. Match parameters: $text, $termlist $text: text to match against the termlist $termlist: object pointer to a LoadTermList object with a termlist loaded output: %score: an associative array with classifications as keys and scores as values

3. Heuristics: sum scores down the classification tree to the leafs cleanEiTree parameters: %res - an associative array from Match output: %res - same array

AUTHOR Anders Ardö, <anders.ardo@it.lth.se>

A.5.12 Combine::RobotRules

NAME RobotRules.pm

AUTHOR Anders Ardo version 1.0 2004-02-19

A.5.13 Combine::SD_SQL

NAME SD_SQL

DESCRIPTION Reimplementation of sd.pl SD.pm and SDQ.pm using MySQL contains both recyc and guard

Basic idea is to have a table (urldb) that contains most URLs ever inserted into the system together with a lock (the guard function) and a boolean harvest-flag. Also in this table is the host part together with its lock. URLs are selected from this table based on urllock, netloclock and harvest and inserted into a queue (table que). URLs from this queue are then given out to harvesters. The queue is implemented as: # The admin table can be used to generate sequence numbers like this: #mysql> update admin set queid=LAST_INSERT_ID(queid+1); # and used to extract the next URL from the queue #mysql> select host,url from que where queid=LAST_INSERT_ID(); # When the queue is empty it is filled from table urldb. Several different algorithms can be used to fill it (round-robin, most urls, longest time since harvest, ...). Since the harvest-flag and guard-lock are not updated until the actual harvest is done it is OK to delete the queue and regenerate it anytime.

########################## #Questions, ideas, TODOs, etc #Split table urldb into 2 tables - one for urls and one for hosts??? #Less efficient when filling que; more efficient when updating netloclock #Datastruktur TABLE hosts: create table hosts(host varchar(50) not null default '', netloclock int not null, retries int not null default 0, ant int not null default 0, primary key (host), key (ant), key (netloclock));

########## Handle to many retries?

```
algorithm takes an url from the host that was accessed longest ago
($hostid,$url)=SELECT host,url,id FROM hosts,urls WHERE
     hosts.hostlock < UNIX_TIMESTAMP()
     hosts.host=urls.host AND
     urls.urllock < UNIX_TIMESTAMP() AND
     urls.harvest=1 ORDER BY hostlock LIMIT 1;

algorithm takes an url from the host with most URLs
($hostid,$url)=SELECT host,url,id FROM hosts,urls WHERE
     hosts.hostlock < UNIX_TIMESTAMP()
     hosts.host=urls.host AND
     urls.urllock < UNIX_TIMESTAMP() AND
     urls.harvest=1 ORDER BY host.ant DESC LIMIT 1;

algorithm takes an url from any available host
($hostid,$url)=SELECT host,url,id FROM hosts,urls WHERE
     hosts.hostlock < UNIX_TIMESTAMP()
     hosts.host=urls.host AND
     urls.urllock < UNIX_TIMESTAMP() AND
     urls.harvest=1 LIMIT 1;
```

AUTHOR Anders Ardö <anders.ardo@it.lth.se>

A.5.14 Combine::XWI

NAME XWI.pm - class for interfacing to various web-index format translators

DESCRIPTION

```
2002-09-30 AA0
  added robot section in analogue with meta
```

AUTHOR Yong Cao <tsao@munin.ub2.lu.se> v0.05 1997-03-13

Anders Ardö, <anders.ardo@it.lth.se>

A.5.15 Combine::selurl

NAME selurl - Normalise and validate URIs for harvesting

INTRODUCTION Selurl selects and normalises URIs on basis of both general practice (hostname lowercasing, portnumber substsitution etc.) and Combine-specific handling (aplpying config_allow, config_exclude, config_serveralias and other relevant config settings).

The Config settings catered for currently are:

maxUrlLength - the maximum length of an unnormalised URL allow - Perl regular to identify allowed URLs exclude - Perl regular expressions to exclude URLs from harvesting serveralias - Aliases of server names sessionids - List sessionid markers to be removed

A selurl object can hold a single URL and has methods to obtain its subparts as defined in URI.pm, plus some methods to normalise and validate it in Combine context.

BUGS Currently, the only schemes supported are http, https and ftp. Others may or may not work correctly. For one thing, we assume the scheme has an internet hostname/port.

clone() will only return a copy of the real URI object, not a new selurl.

URI URI-escapes the strings fed into it by new() once. Existing percent signs in the input are left untouched, which implicates that:

(a) there is no risk of double-encoding; and

(b) if the original contained an inadvertent sequence that could be interpreted as an escape sequence, uri_unescape will not render the original input (e.g. url_with_%66_in_it goes whoop) If you know that the original has not yet been escaped and wish to safeguard potential percent signs, you'll have to escape them (and only them) once before you offer it to new().

A problem with URI is, that its object is not a hash we can piggyback our data on, so I had to resort to AUTOLOAD to emulate inheritance. I find this ugly, but well, this *is* Perl, so what'd you expect?