Methodology and architecture for content collection

Distribution: Project internal

MedIEQ

Quality Labeling of Medical Web content using
Multilingual Information Extraction

National Centre for Scientific Research "Demokritos"
Teknillinen Korkeakoulu – Helsinki University of Technology
Universidad Nacional de Educacion a Distancia
Col.legi Oficial de Metges de Barcelona
Zentralstelle der deutschen Ärzteschaft zur Qualitätssicherung in der
Medizin
Vysoka Skola Ekonomicka V Praze
I-Sieve Technologies Ltd

2005107 **D6**

December 2006

Project ref. no.	2005107
Project acronym	MedIEQ
Project full title	Quality Labeling of Medical Web content using Multilingual Information Extraction

Security (distribution level)	Project internal
Contractual date of delivery	30 September 2006
Actual date of delivery	22 December 2006
Deliverable number	D6
Deliverable name	Methodology and architecture for content collection
Type	Report
Status & version	Final
Number of pages	34
WP contributing to the deliverable	WP5
WP / Task responsible	NCSR
Other contributors	UNED, WMA, AQUMED, UEP, I-sieve
Author(s)	K. Stamatakis, V. Karkaletsis, V. Metsis (NCSR), M. Ruzicka, M. Labský, V. Svátek (UEP), E.A. Cabrera, C.Muñoz, F. L. Ostenero V. Peinado (UNED), M. Pöllä, T. Honkela (TKK)
EC Project Officers	Artur Furtado
Keywords	Web content collection, crawling, spidering, content classification, link scoring, ML, heuristics, corpus formation.
Abstract (for dissemination)	This document proposes the methodology and architecture for the content collection in MedIEQ. Several tools, contributing to on-line content collection, constitute the Web content collection toolkit. The content collection methodology is designed on the principle of open architecture.

Table of contents

Executive summary

MedIEQ is an on-going EC-funded project aiming to:

- Provide a common vocabulary and machine readable schema for the quality labelling of health related web content and develop tools supporting the creation, maintenance and access of labelling data according to this schema;
- Specify a methodology for the content analysis of health web sites according to the MedIEQ schema and develop the tools that will implement this methodology;
- Integrate these technologies into a prototype labelling system in seven (7) languages (EN, ES, DE, CA, GR, FI, CZ) aiming to assist the labelling experts.

At the time of writing this report, the 1st version of the MedIEQ schema has been finalized (see Deliverable D4.1). This schema will allow the creation of machine readable content labels.

Work is now underway to develop applications to make use of such labels (generation, maintenance, validation against the MedIEQ schema).

Before that, other applications have to analyze the content of health websites and extract information related to the labelling criteria included in the MedIEQ schema. Two separate toolkits, handling the different levels of content analysis, have been scheduled: the Web Content Collection toolkit (WCC) and the Information Extraction toolkit (IET).

This report focuses on the Web Content Collection toolkit (WCC).

1. Introduction

The prototype MedIEQ labelling assisting system (also called AQUA, from Assisting Quality Assessment) consists of 5 subsystems or toolkits:
1. the label management toolkit (LAM),
2. the web content collection toolkit (WCC),
3. the information extraction toolkit (IET),
4. the multilingual resources management toolkit (MRM),
5. the monitor-update-alert toolkit (MUA).

LAM manages (generates/validates/modifies/compares) quality labels based on the schema proposed by MedIEQ (see Deliverable D4.1).
WCC identifies, classifies and collects on-line content relative to a number of machine readable quality criteria (proposed by the labelling agencies participating in the project) in seven languages (EN, ES, DE, CA, GR, FI, CZ).
IET analyses the web content collected by WCC and extracts attributes for MedIEQ compatible content labels.
MRM gives access to health-related multilingual resources (like MeSH, ICD or whatever being available); input from such resources is needed in specific parts of both the WCC and IET toolkits.
All data necessary to the different subsystems as well as to the overall AQUA system are stored in the MedIEQ repository.

Finally, MUA handles a few auxiliary but important jobs, like the configuration of monitoring tasks, the MedIEQ repository's entries updates, the alerts to labelling experts when important differences occur during monitoring existing quality labels.

This document focuses on WCC. The components participating in this toolkit are the following:
- Focused Crawler (identifying health related websites),
- Spider (navigating websites) with link-scoring and content-classification capabilities,
- Tools assisting the formation of corpora (to train and test classification algorithms)
- A mechanism producing trained classification/scoring models (to be used by the Spider).

Related work in online content collection is described in section 2. The WCC methodology, proposed by MedIEQ, with short descriptions of its components, is given in section 3. The proposed WCC architecture is described in section 4. Use cases are described in section 5. Section 6 provides further information and details on specification issues for the different tools participating in WCC. Section 7 gives our concluding remarks and describes the future steps. Finally, to assist the document reading due to the numerous acronyms employed, a Glossary is also found in the Appendix.

6

2. Related work

Web Content collection

MedIEQ opted to use "web content collection" instead of "focused crawling", which is often employed in the relevant literature, to describe the process of seeking the Web for content relevant to a pre-defined set of topics and store this content locally.

Web content collection in MedIEQ is divided in a number of separate, subsequent processes. Each such process necessitated the development of a set of individual tools and software components. While terms like "Crawler", "Spider", "Content classifier", "Link-scoring", etc., have been employed to identify some of them, a wider definition, which includes the aggregation of all individual tools and processes, seemed more appropriate.

Below, information on the related work in Focused Crawling and Link-Scoring (both falling under MedIEQ's "Web content collection") is presented.

Focused Crawling

A web crawler is a program which automatically traverses the web by downloading documents and following links from page to page. A general-purpose web crawler normally tries to gather as many pages as it can from a particular set of sites. In contrast, a Focused Crawler (the term "focused crawling" was introduced by Chakrabarti et al. in 1999 [10]) is a hypertext resource discovery system, which has the goal to selectively seek out pages that are relevant to a pre-defined set of topics. Rather than collecting and indexing all accessible web documents to be able to answer all possible ad-hoc queries, a focused crawler analyzes its crawl boundary to find the links that are likely to be most relevant for the crawl, and avoids irrelevant regions of the web. This leads to significant savings in hardware and network resources, and helps keep the crawl more up-to-date.

There is a substantial amount of work about the methodology of Web crawling. Many crawler architectures and prototypes (e.g. Mercator [24], PolyBot [36], UbiCrawler [6]) were proposed in the recent literature. The main focus of prior work lies on aspects like crawler scalability and throughput [24], distributed architecture [6], or implementational aspects in connection with particular programming languages (e.g. Java) [36]. However, these solutions do not address the demands of thematically focused Web retrieval applications. The idea of automatic categorization of Web data was addressed by many researchers [26, 16, 12]. The observation that the topic-specific information is on the Web always "a few clicks away" (i.e. the Web shows clear small-world behavior) [2, 3, 8]

has motivated several improvements to the general crawling scenario [33].

The first attempts to implement focused crawling were based on searching the Web using heuristic rules that would guide the choices of the crawler. These rules are usually based on keywords found near the link and in the rest of the page that contains it. The crawler performs a search strategy combined with the heuristic rules in order to follow successful paths leading to relevant pages. Such implementations are Fish-Search [19][20] and Shark-Search [23].

A similar idea of heuristic-based neighborhood exploration was proposed by Menczer in [27]. The recent paradigm of thematically focused Web exploration was intensively studied by Chakrabarti [9]. He considered aspects of hypertext categorization into hierarchical topic taxonomies [11, 12], distillation of thematical Web topics [13, 14], and the methodology of focused crawling [10] for Web a great extent [4].

More recent methods use information related to the structure of the Web graph, in order to perform more efficient focused crawling. Some of these methods take advantage of the Topical Locality of the Web (the property of pages with similar topic being connected with hyperlinks [7]) and use it to guide the focused crawler [10]. Moreover, the "backlink" information (pages that link to a certain document), provided by search engines like Google or Altavista, can be used to generate a model of the Web-graph near a relevant page, such as in the case of Context Graphs [21].

The concepts of a thematically focused Web retrieval framework were studied by Menczer in [31]. The idea of focused crawling was used for a variety of Web retrieval scenarios, including exploration of user-specific topics of interest on the Web [9, 37], expert search within a well-defined set of Web sites (e.g. locating the name of the CEO within a given company site) [35], location of business information [32], or finding hidden-Web databases (pages that contain forms and are expected to be backed by databases with topic-specific contents) [5, 34]. The methodology of evaluating adaptive crawling strategies was addressed in the studies of [28].

The Crossmarc[1] focused crawler [40] exploits three distinct content discovery mechanisms (their start points being defined by humans): a) structured search: topic-based Web hierarchies (Web directories) are explored, b) free search: keywords, taken from Crossmarc's domain ontologies and lexicons, form sets of queries which are submitted to different search engines, c) similarity search: a set of "seed" pages is given and a "similar pages" search is conducted. URLs collected from all three mechanisms are finally merged, forming a domain-focused list.

There are also some methods that use reinforcement learning in order to deal with focused crawling. In [29], the crawling component is based on reinforcement learning, although some simplifying assumptions are made.

---

[1] http://www.iit.demokritos.gr/skel/crossmarc/

More specifically, in this approach the state space has been omitted, due to high dimensionality of the data. Therefore, the agent examines only the value of the possible actions to be taken, irrespective of the state of the environment. The actions are represented by the different hyperlinks that exist in a Web page, and the value of each action is estimated by a "bag-of-words" mapping of the keywords in the neighborhood of the hyperlink to a scalar value.

In InfoSpiders system [18], a multi-agent focused crawler, the process is initialized by a set of keywords and a set of root pages. Each agent starts with a root page and performs focused crawling by evaluating the link value and following the most promising links. Link value is assessed using a reinforcement learning method, using contextual words as input. Reward values are calculated online, by the reward that the agent receives when following a link. The user can provide relevance feedback to assist the learning process.

Finally, in [22] a focused crawler is described that consists of two interconnected regimes: ontology and crawling. The former is mainly done by the human engineer. He defines the crawling target in the form of instantiated ontology. The latter comprises the internet crawler. It interacts automatically with the data contained on the Web and retrieves them. Then it connects to the ontology to determine relevance.

In general, focused crawlers have been shown to provide better results for user-specific topics with substantially lower crawling overhead than exhaustive, unfocused engines [10, 35, 21]. However, prior work has not considered thematically focused crawl as an instrument for acquisition of aligned multi-lingual corpora for cross-language question answering and ontology learning applications.

One of the most robust and reliable solutions, which already has a long real-time running/testing period in the healthcare domain is MARVIN[2], the Crawling/Spidering application of HON (Health on the Net) foundation. MARVIN (Multi-Agent Retrieval Vagabond on Information Networks) searches the Web and selects only documents that are relevant to a specific and chosen domain. Document relevance is computed according to a formula that takes into consideration the number of words from a glossary of significant terms that MARVIN finds in the document, as well as their place in the document[3]. MARVIN stores selected documents in a database that users can then query. For more information on MARVIN see [41, 42].

Link Scoring

Experiences with a focused crawler including link analysis, linkage sociology (who links to who), sites inspection, semi-supervised learning etc, are described in [10]. Query refinement has been investigated also in

---

[2] http://www.hon.ch/Project/Marvin_project.html
[3] http://www.hon.ch/Project/Marvin_specificities.html

the context of ontologies and the Semantic Web (compare [38], [39]) resource discovery. The crawler aimed to identify the most promising crawl directions by periodical estimation of link-based hub scores of fetched Web pages, e.g. using the HITS [25] algorithm.

The similar idea of URL ordering on the crawl frontier by PageRank [7, 30] was proposed in [17]. It aims to gain efficiency by crawling "more important'' pages first. Various measures of importance for a page are introduced e.g. similarity to a driving query, number of pages pointing to this page (backlinks), pagerank and location (in a hierarchy).

A focused crawler who tries to learn the linkage structure is described in [15][35]. This involves looking for specific features in a page which makes it more likely that it links to a given topic. Such crawler implements link classifier based on these features. They may include page content, URL structure, the link annotation (anchor text), and text blocks in the neighborhood of the anchor tag.

Finally, Aggarwal et al. crawling approach (called intelligent crawling) [1] uses a combination of evidence (contents of in-linking pages, tokens in the URL, and contents of sibling pages), in order to rank the candidate hyperlinks by their level of interest and learns the relevant weight of these factors as it crawls. Making the assumption that the initial set of starting points can lead to all interesting pages, very central sites should be used as starting points for the crawl (e.g. Yahoo, Amazon, etc.). An analogous approach, based on keyword matching within link annotations, was addressed by Barabasi and Albert in [3].

The MedIEQ approach

Existing experience and previous initiatives have been considered already since MedIEQ was in its earliest steps. There were a number of solutions coming outside the MedIEQ consortium (e.g. MARVIN from HON). At the same time, the technical partners participating in MedIEQ had had important crawling/spidering experience from previous European projects (e.g. Crossmarc[4], Rainbow[5]) and also crawling/spidering software developed for the purposes of these projects.

MedIEQ approach, by design, is to semi-automate the labeling process and, therefore, assist the labeling expert while he/she is reviewing a health related site or document. Such a review was, and still is, a manual and time-consuming work. Since no previous approach was tackling the automation of the labeling process by providing machine-extracted information on quality criteria, MedIEQ partners decided to continue, expand and adapt the software tools they already owned. However, due to the open architecture design of WCC, other crawling/spidering techniques could also be exploited.

---

[4] http://www.iit.demokritos.gr/skel/crossmarc/
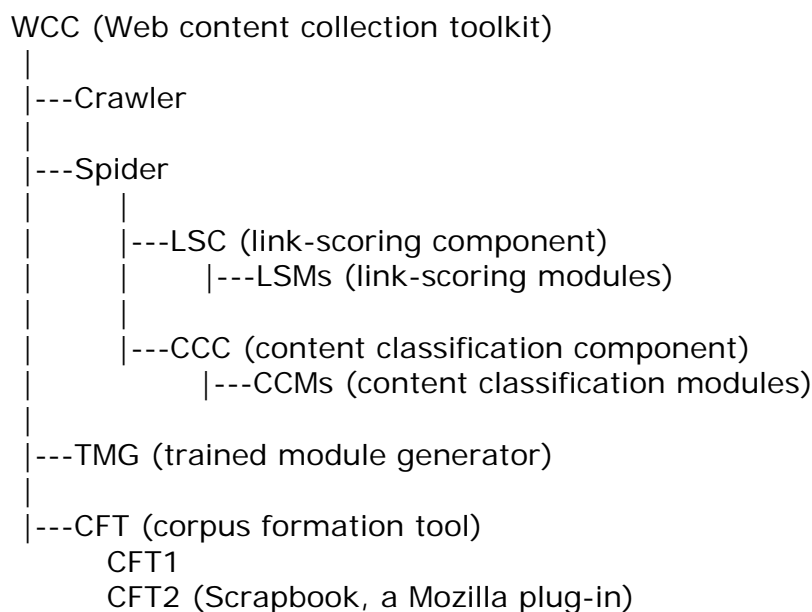[5] http://rainbow.vse.cz/descr.html

3. The content collection methodology

A set of tools/components for the collection of on-line content in different languages constitute the Web content collection (WCC) toolkit. The toolkit will provide user interfaces through which several actions will be possible.

3.1 Hierarchy of tools and components participating in WCC

Below we can see the tree of tools and components participating in WCC toolkit.

```
WCC (Web content collection toolkit)
 |
 |---Crawler
 |
 |---Spider
 |       |
 |       |---LSC (link-scoring component)
 |       |       |---LSMs (link-scoring modules)
 |       |
 |       |---CCC (content classification component)
 |               |---CCMs (content classification modules)
 |
 |---TMG (trained module generator)
 |
 |---CFT (corpus formation tool)
        CFT1
        CFT2 (Scrapbook, a Mozilla plug-in)
```

The first two, the Crawler and the Spider, with their satellite components, the LSC and the CCC, are active components (frontline actors in web content collection), while the last two are auxiliary components (as CFT provides input to TMG and TMG outputs modules for CCC and LSC).

In 3.2, a step-by-step methodology for WCC is described, while in 3.3 there is a quick reference to all WCC tools and components. Details on their specifications can be found in section 6.

3.2 WCC methodology step-by-step

Here is a step-by-step description of the Web content collection methodology proposed in MedIEQ (step 1, below, can be omitted as content could be collected from already known sources):
1. Crawling: a user searches the web to identify on-line sources having relevant content (in our case, health related content). Such a search is performed by the Crawler (initial points are provided manually), exploiting existing search engines and Web directories.

11

2. <u>Spidering</u>: Health-related content sources (websites, documents, etc.), either known or identified by the Crawler, are explored: a) all internal links are scored (by LSC) and the most promising followed, b) every visited page's content is classified (by CCC) and fit pages are locally stored (they will be forwarded to IET toolkit). This double filtering, of irrelevant links, on the one hand, and of irrelevant pages, on the other hand, improves spidering speed and hardware cost.
3. <u>Training</u>: Before spidering, the generation of efficient link-scoring and content classification models is necessary. For this, known ML algorithms[6] are proposed and the TMG application is used. Before the models generation, one of the two proposed corpus formation tools (CFT) has assisted the user in the collection of the relevant examples to train/test the algorithm.

3.3 Tools and components quick reference

Short descriptions on the software tools and components of WCC are provided here.

Crawler

The Crawler (or Focused Crawler) searches the Web for health related content, which doesn't already have a quality label (at least not a label found in MedIEQ records). It is a meta-search-engine, exploiting results returned from known search engines and directory listings from known Web directories.

To set off crawling, the user provides two types of start points: sets of keywords and sets of URLs of Web directories. The more relevant to a given topic these start points are, the more focused the crawling will be.

On one hand, keywords are used to query the supported search engines. Their results are parsed and URLs are collected.

On the other hand, Web directories are explored (subtrees visited by the Crawler) and the contained URLs are collected.

The totalities of collected URLs from all sources are merged and a final URLs list is returned. Merging process minds to a) remove possible duplicates and b) ignore subpaths of URLs already in list. Finally, URLs having already a quality label (Crawler consults the MedIEQ repository for this) are also removed.

Spider

---

[6] The use of the WEKA platform (http://www.cs.waikato.ac.nz/ml/weka/) is adopted.

How does a spider fetch all web pages? The only way to collect links to new pages (URLs) is to scan already collected pages for hyperlinks that have not been collected yet. This is the basic principle of crawlers/spiders. They start from a given set of URLs, progressively fetch and scan them for new URLs and then fetch these pages in turn, in an endless cycle.

In this project we are more interested in kind of focused crawling (finding only relevant medical web sites/pages) and not in going through whole World Wide Web. We decided that the Spider will investigate only specific web sites collected by the Crawler (to ensure spidering only web sites from health domain) and it will follow only internal links (links pointing to new pages on the same site). Unlike general spiders this process is finite and potentially pending work is not so much hardware consuming.

In Spider's first version, we want to collect only static web pages, so the application omits any dynamically generated pages, hidden web etc. (these cases will be handled in future Spider versions).The first version of the Spider is simply fetching sites from the crawler one-by-one. Unreachable sites/pages are revisited in next run. This version is single thread (visiting one page at the time), and it doesn't implement more sophisticated features like "robot exclusion" (using robot.txt file mechanism) or dealing with difficult "spider traps".

To enhance spidering process it communicates with "Link scoring component" (LSC). This component gives a score to every link and returns values providing a sorted queue of unvisited links to the Spider. Spider continues by visiting the first (best scored) link from the queue and ignores links with score under given threshold.

LSC (link-scoring component)

The main purpose of LSC is to analyze link (link object including URL, surrounding text, link text, anchor etc.) and to forecast content of target page without really visiting it. Since we want to classify pages in already known categories (like contact pages, virtual consultation pages, etc.) LSC contains several "link scoring modules" dedicated to each category. LSC collects values from all modules and passes them back to Spider.

According to LSC results, Spider should filter out irrelevant pages (not visiting them, no local copy and no classification by CCC). This should improve spidering speed and hardware cost.

However, LSC's usefulness has to be examined by measuring the recall of the different LSMs. If we don't achieve a satisfactory recall for all LSMs (if many interesting links are missing), we retry with lower thresholds (where necessary). We re-measure the recalls and so on, until we get acceptable recall values. Obviously, if no satisfactory recall is ever achieved, the possibility to completely abandon the entire LSC will then be considered.

CCC (content classification component)

Given a page/document, CCC applies classification modules (CCM) in order to classify it. The CCC allows activating or deactivating different classification criteria and assigning thresholds to each category in the classification process (however, such rights are exclusive to the system administrator). For each classification criterion and language, different (alternative) classification modules (CCMs) may exist; those having best recall values are loaded and exploited: one module, the best, is selected per criterion and per language.

According to CCC results, irrelevant pages and also pages not needing any extraction (e.g. "target audience – child" positives) are filtered out (no local copy and no forward to Information Extraction toolkit) while accepted ones are redirected to the corresponding IE components, i.e. when a page is classified as "contact page", it is, eventually, sent to the contact-details-extraction component. This should improve IE speed and hardware cost.

CFT (corpus formation tool)

Two different applications are available for the formation of the needed training & testing corpora:

- A tool assisting corpora collection, developed within the MedIEQ consortium and called CFT1 and
- A Mozilla Firefox browser extension, developed initially at Murota Laboratory[7] allowing to store web pages and organize them in collections, called Scrapbook and, in this report, also CFT2.

CFT1

The CFT1 application searches for web pages having specific content (fit pages). CFT1 does not search the entire Web but looks only inside websites specified by the user. For that, it exploits the in-site search feature provided by Google search engine.

To set off CFT1, the user provides two types of start points: keywords (terms) and urls of websites to search-in. CFT1 returns urls pointing to fit webpages. The more specific these start points are, the more fit the returned webpages will be (a few initial tests help the user to narrow his/her search and obtain better results).

Additionally, CFT1 provides a learning mechanism: having manually classified (e.g. in pos|neg) a first set of returned pages, the user can then train a classifier. Therefore, having the classification model, the tool will, in future searches, auto-classify and store the incoming pages, limiting

---

[7] A member of the Chair of Human Resource Development in the Department of Human System Science at Graduate School of Decision Science and Technology, Tokyo Institute of Technology. Scrapbook is created by Gomita gomita.mail@gmail.com.

thus the results displayed to the user (the user has to classify only pages upon which the algorithm cannot decide).

CFT2 or the Scrapbook Firefox Mozilla plug-in

ScrapBook is a Firefox extension, which helps anyone to save Web pages and manage the collection. Key features are lightness, speed, accuracy and multi-language support. ScrapBook is available at: http://amb.vis.ne.jp/mozilla/scrapbook/.

Details on its features can be found at:
http://amb.vis.ne.jp/mozilla/scrapbook/feature1.php?lang=en
http://amb.vis.ne.jp/mozilla/scrapbook/feature2.php?lang=en
http://amb.vis.ne.jp/mozilla/scrapbook/feature3.php?lang=en

TMG (trained module generator)

TMG is a component for the generation of all classification modules needed in WCC: CCMs (content classification modules) and LSMs (link-scoring modules). What TMG takes as input, either in case of a CCM or of a LSM generation, is collections of content (ascii content, which is, for a CCM, collections of web pages and, for a LSM, collections of link objects). TMG uses different ML techniques to produce classification/scoring models.

First version of TMG uses ML techniques only. However, the possibility of allowing the user, in future versions, to select between ML and heuristics or probably use a combination of both techniques is examined.

15

4. The content collection architecture

A schema of the internal architecture of the WCC toolkit (dotted circle) as well as toolkit's position inside the overall MedIEQ system, is given here (Figure 1).
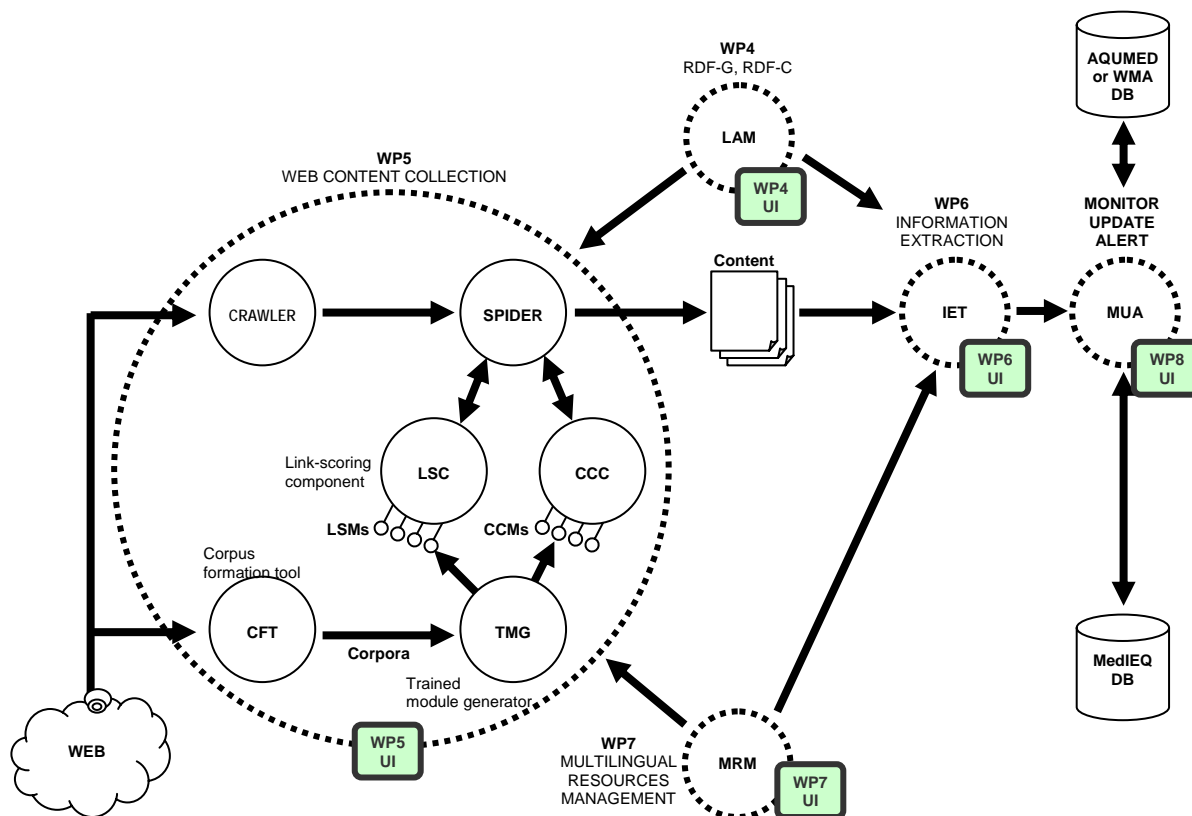

Figure 1: WCC toolkit's architecture and its position inside the overall MedIEQ system

Some quick explanations of the acronyms[8] used in the above figure:

Inside WCC

- LSC (link-scoring component)
  - LSMs (link-scoring modules)
- CCC (content classification component)
  - CCMs (content classification modules)
- TMG (trained module generator)
- CFT (corpus formation tool)

---

[8] Consult also the Glossary in Appendix.

Outside WCC

- LAM (Label Management Toolkit)
    - o   RDF-G (RDF label Generator)
    - o   RDF-C (RDF label Comparator)
- IET (Information Extraction Toolkit)
- MRM (Multilingual Resources Management Toolkit)
- MUA (Monitor Update Alert)

5. Users, user interfaces and use cases

5.1 Users & UIs

User types: the possible MedIEQ user types are either the labelling experts (quality specialists from AQUMED or WMA) or the system administrator. Therefore, two types of user interfaces (UIs) are foreseen, each addressed to a MedIEQ user type.

User interfaces: given that most today's users navigate the Web and feel familiar with Web interfaces (such as Web forms), the MedIEQ partners opted to design and develop the overall MedIEQ integrated system as a large Web application. This application is named AQUA system, an acronym standing for Assisting Quality Assessment system. Obviously, all toolkits and tools will run through web interfaces. This applies also in WCC.

Inside WCC, all components necessitate a UI. At the same time, every tool and component has an API (application programming interface).

However, concerning UIs, when looking at components specified actions in section 6, we see that some components will be exclusively accessed by the MedIEQ user administrator, others by the labelling experts only, while the rest by both user types. Our aim is that all specified actions will be handled through user friendly Web interfaces. Therefore, in many cases, some kind of aggregating interface is opted in order to simplify interactions between the users and the system (see AQUA details, in the forthcoming deliverable D12).

5.2 Use cases

Actors & interactions

An actor is a user; it's anyone who can use the MedIEQ system and/or its different MedIEQ toolkits and components. An actor can be either a human or a program (S/W). Below, we enumerate all possible actors:

Human actors

    1. Labelling expert (an expert from AQUMED or WMA)

    2. System administrator

S/W actors

    1. LAM (Label Management Toolkit, the WP4 toolkit)

2. WCC (Web Content Collectiuon, the WP5 toolkit)

3. IET (Information Extraction Toolkit, the WP6 toolkit)

4. MRM (Multilingual Resources Management Toolkit, the WP7 toolkit)

5. MUA (Monitor-Update-Alert, the WP8 toolkit)

6. MedIEQ Database

7. Cron jobs (services executing a single or multiple applications – according to some work-flow – on schedule)

Notice that we distinguish two interaction levels:

- High Interactions Level [-H-]: where all the above listed actors can directly interact with each other. This means that toolkits interact with other toolkits, the database and the cron jobs. Every toolkit has an API. The specified API actions are linked to interfaces.

- Low Interactions Level [-L-]: within this level we group the interactions between components inside a toolkit. Tools and components inside a toolkit interact with each other according to the toolkit internal architecture.

Use case 1: Identify new health websites

Actors
[-H-] Labelling expert, WCC, MRM, Database
[-L-] Crawler
Interactions
[-H-] A labelling expert calls WCC to find new unlabelled websites (AQUMED scenario). This can also be done by creating a service that runs periodically (=a cron job).
[-H-] WCC asks from MRM the available linguistic resources.
[-H-] The labelling expert selects (defines) from the MRM interface a number of keywords to guide the unlabelled websites search.
[-L-] The Crawler runs (taking into account the user defined keywords, and other user configurations from its UI); by consulting the MedIEQ database it skips entries being previously identified.

Use case 2: Form a corpus

Actors
[-H-] Labelling expert, WCC, MRM
[-L-] CFT
Interactions
[-H-] A labelling expert calls WCC to form a training corpus for a given (existing or new) labelling criterion.
[-H-] WCC asks from MRM the available linguistic resources.
[-H-] The labelling expert selects (defines) from the MRM interface a number of keywords to guide the corpus formation task.

[-L-] The CFT looks for possible Crawler's output urls lists and prompts the labelling expert to select one such list.

[-L-] The CFT runs (taking into account the user defined urls list, the selected urls, the user provided keywords and other user configurations from its UI).

Use case 3: Generate a CCM

Actors

[-H-] System administrator, WCC

[-L-] TMG

Interactions

[-H-] The system administrator calls WCC; he wants to generate a content classification model for a given labelling criterion.

[-L-] WCC looks for available corpora (formed by the labelling expert using CFT, see above).

[-L-] The TMG component runs (taking into account the user specified corpora and other possible user configurations from its UI) and produces the required CCM module.

Use case 4: Generate a LSM

Actors

[-H-] System administrator, WCC, MRM

[-L-] TMG

Interactions

[-H-] The system administrator calls WCC; he wants to generate a link-scoring model for a given labelling criterion.

[-H-] WCC asks from MRM the available linguistic resources.

[-H-] The system administrator selects (defines) from the MRM interface a number of keywords to guide LSM generation task.

[-L-] The TMG component runs (taking into account the user specified keywords and other possible user configurations from its UI) and produces the required LSM module.

Use case 5: Collect on-line content

Actors

[-H-] Labelling expert, WCC

[-L-] Spider, CCC, LSC

Interactions

[-H-] The labelling expert calls WCC; he wants to collect (locally store) on-line health-related content.

[-L-] For this, WCC calls the Spider. The user has to specify a number of parameters like the web sources to explore, preferred languages of the content, the accepted content type(s), the labelling criteria etc.

[-L-] The Spider runs: a) internal (same host) links are collected and scored (LSC is called) and the most promising followed, b) every visited page's content is classified (CCC is called) and fit (to specified criteria) pages are locally stored.

20

6. Specifications of the WCC tools and components

Below, in the 'actions' paragraph of each tool, you see the possible actions provided to each user by the relevant UI. Note that an action's identifier starts by the first letter of the tool this action corresponds to (e.g. C for the Crawler), follows an A (which comes from Action) and ends with an auto-increment number (e.g. CA3). Exceptions are the actions of CCC, where the first letter is X, and the actions of CFT, where the first letter is F (as C has been used for Crawler's actions).

6.1 Detailed specifications

Crawler

a. Searches the Web starting from keywords and web directories specified by the user.

b. Has 3 search engine wrappers (for Google, Yahoo, MSN).

c. Supports several filtering options supported by search engines (language, file format, domain, date, etc.).

d. Queries to the selected search engines are constructed from the given keywords. Search results are processed and result urls collected.

e. Web directories are visited and contained urls collected.

f. Urls returned by all sources are cross-checked: duplicates and sub-paths are removed (e.g. from http://www.health.com/123/456/abc.html and http://www.health.com/123/index.html the first is removed).

g. Gives two options for output: only URLs OR URLs+data (data may be: title, keywords, description, last update, etc.). When URLs+data are selected:
- Every url is visited,
- Text is separated from structure (html, JavaScript, etc.),
- Text's language is identified,
- Content's encoding is detected,
- Content's encoding becomes utf-8 (if other),
- All data are extracted,
- Visited resources can be locally stored (if needed).

h. I/O: Crawler's input (keywords & URLs) and output (URLs) is always utf-8, therefore, the tool is language independent (content in any language is automatically supported).

21

Actions of the Crawler

(Group 1 - Actions provided to the labelling expert user through the expert interface)
CA1. Specify keywords and/or web directories URLs.
CA2. Select search engine to query.
CA3. Configure crawling through advanced search options (language, file types, date, host, etc.).
CA4. Output format selection: simple URLs list or URLs+data (data may be title, keywords, description, last update, etc.)

(Group 2 - Actions provided to the system administrator through the sysAdmin interface)
There is no such action.

Spider

a. Navigation: the spider navigates websites. Starting URLs may point either to the home page (e.g. http://www.myhealth.com/) or to a given area of a site (e.g. http://www.myhealth.com/prevention/).
Note that steps b, c and d (below) are performed for each visited URL.

b. Pre-processing: first, the Spider does some content pre-processing (4 steps):
- If content consists of text and structure (e.g. html, css, JavaScript, etc.), text is separated from structure,
- Text's language is identified (identified language values being kept),
- Content's encoding is detected,
- Content's encoding becomes utf-8 (if other).

c. Content classification
- The Spider calls a content classification component (CCC) consisting of several content classification modules (CCMs). Looking for content of different types (contact pages, virtual consultation pages, etc.), every specialized CCM gives to content (of every visited url) a corresponding probability. When this probability is over a threshold, content is locally stored (stored content will then be forwarded to IE tools).

d. Link scoring
- The Spider, from every page visited, collects the containing links (with some metadata on them, like alt text, anchor text, text blocks in the neighborhood, etc.)
- In-site links are examined while external links ignored.
- In order to score links, the Spider calls a link-scoring component (LSC, see below) which has several link-scoring modules (LSMs). As there is interest for links pointing to different types of content (contact pages, virtual consultation pages, etc.), every specialized LSM gives to every link object (link+metadata) a corresponding score.
- If score of at least one link scoring module reaches its threshold the link is added into the list of unvisited links (spidering queue).

e. Loop: steps b, c, d are repeated until the list of unvisited links is empty.
f. I/O: Spider's input (URLs lists) and output (stored web pages) is always utf-8, therefore, the tool is language independent (content in any language is automatically supported).

Actions of the Spider

(Group 1 - Actions provided to the labelling expert user through the expert interface)
SA1. Open/Load: Load the URLs collected by the Crawler or any URLs list by specifying its path (a URLs list is a txt file with one URL per line).
SA2. Edit URLs list: Add/remove/modify/save the contents of a URLs list.
SA3. Select: Select one or more URLs from the list (selected URLs to be spidered).
SA4. On/Off spider: start/pause/stop the spider.

(Group 2 - Actions provided to the system administrator through the sysAdmin interface)
SA5. Select link types: Select form a number of supported link types, which types of links to follow and which not, when navigating a website (for example, follow text links and links in swf graphics and ignore links in JavaScript, e.g. textlinks=1, swflinks=1, jslinks=0, etc.).
SA6. Select file types: Select from a number of supported extensions, which file extensions to accept (= to visit) and which not, when navigating a website (e.g. html=1, php=1, pdf=1, xml=0, txt=1, rdf=0, etc.).
SA7. Add/remove/modify the supported extensions.
SA8. On/Off LSC: Activate/deactivate the LSC.
SA9. On/Off CCC: Activate/deactivate the CCC.

LSC (link-scoring component)

a. LSC consists of several LSMs (link-scoring modules), being generated by TMG.

b. Some LSMs may be active while others inactive (skipped).

c. Active LSMs are called in sequences by LSC.

d. Each LSM has a "specialization": to forecast if content of target page correspond to a specific quality criterion without actually visit the page).

e. Each LSM has its own threshold (e.g. 0,3).

f. A LSM's input is a link-object and its output is a score value. When the score surpasses the LSM's threshold (e.g. score=0,6, threshold=0,3), the given link is appended in the unvisited (best-scored-first) links queue.

g. I/O: LSC's input is always utf-8 and its output is a number, therefore, the tool is language independent (content in any language is automatically supported).

Actions of the LSC

(Group 1 - Actions provided to the labelling expert user through the expert interface)
There is no such action.

(Group 2 - Actions provided to the system administrator through the sysAdmin interface)
LA1. On/Off LSC: Activate/deactivate LSC (or all LSMs).
LA2. On/Off LSMs: Activate/deactivate LSMs separately (e.g. activate the LSM which scores links to "virtual consultation" pages, deactivate the appropriate LSM for links to "contact" pages, etc.).
LA3. Adjust LSMs: Adjust the different threshold(s) above which a link is considered as promising in the different LS modules.
LA4. Select from a possible range of different LSMs of same purpose (e.g. select between three different LSMs for contact pages).
LA5. Add/load a new LSM.

CCC (content classification component)

a. CCC has access to several CCMs (content classification modules) being generated by the TMG (trained module generator, see details below).

b. Some CCMs may be active while others inactive (skipped).

c. CCMs are applied in sequences by CCC.

d. Each CCM has a "specialization": recognize content corresponding to a specific quality criterion.

e. Each CCM has its own threshold (e.g. 50/100).

f. CCC's input is a resource's content (text or text+structure, according to what fits best in every case) while its output is a series of probability values and/or binary values (1|0). Only when an answer from at least one CCM is positive and, at the same time, when further processing is needed (=IE), content is locally stored.

g. I/O: CCC's input is always utf-8 and its output is a number. This format is language independent. The CCC need to know the document language in order to select the appropriate CCM.

h. Given the vocabulary defined in each CCM, content is pre-processed and its features are extracted. Then, content is classified by executing a ML classifier.

Actions of the CCC

(Group 1 - Actions provided to the labelling expert user through the expert interface)
There is no such action.

(Group 2 - Actions provided to the system administrator through the sysAdmin interface)
XA1. On/Off CCC: Activate/deactivate CCC (or all CCMs).
XA2. On/Off CCMs: Activate/deactivate CCMs separately (e.g. activate the CCM which identifies "virtual consultation" pages; deactivate the appropriate CCM for "contact" pages, etc.).
XA3. Adjust CCMs: Adjust the different threshold(s) above which a page is considered as fit in the different CCMs.
XA4. Select from a possible range of different CCMs of same purpose (e.g. select between two different CCMs for contact pages).
XA5. Add/load a new CCM (output of TMG).

CFT (corpus formation tool)

Two different applications are available for the formation of the needed training & testing corpora:
- a corpus collection tool developed by NCSR, called CFT1 and
- a plug-in for Mozilla browser, called CFT2 or Scrapbook
(http://amb.vis.ne.jp/mozilla/scrapbook/).

CFT1

a. Current distribution comes with an AWT interface (a web interface, part of the AQUA interfaces, is being designed and is under development).

b. Has a built-in Google search engine wrapper, which, by taking some keywords from the user, searches inside given websites (specified by the user, e.g. Crawler's output) for specific content (e.g. contact pages).

c. Offers the possibility to train a content classifier and continue, with the help of the classifier, with subsequent in-site searches (=focused search).

d. Its user interface consists of:
- a content search/preview interface,
- a content annotation interface (allows characterization of content in given categories, e.g. pos|neg),
- a content storing interface (for storing & organization of content in collections/corpora).

e. I/O: CFT1's input (keywords) and output (sets of stored webpages) is always transformed to utf-8, therefore, the tool is language independent (content in any language is automatically supported).

Actions of the CFT1

(Group 1 - Actions provided to the labelling expert user through the expert interface)
FA1. Open/load a list of urls (Crawler's output or other).
FA2. Preview url in browser.
FA3. Edit urls list (add/remove/modify urls).
FA4. Select (check) urls to query.
FA5. Specify keywords to look for, inside selected hosts (e.g. search for keywords "contact", "address", and "e-mail" in http://www.who.org/).
FA6. Select whether to use or not a trained classifier.
FA7. Preview search result urls in a browser.
FA8. Edit result urls list (add/remove/modify urls).
FA9. Annotate urls: move a url into a classification set (e.g. pos or neg).
FA10. Locally store the content of the annotated urls.
FA11. Train a classifier from classification datasets (e.g. pos|neg data sets).

(Group 2 - Actions provided to the system administrator through the sysAdmin interface)
There is no such action.

CFT2 or the Scrapbook Mozilla plug-in

All information concerning the Scrapbook Mozilla extension is available on-line at: http://amb.vis.ne.jp/mozilla/scrapbook/ (see also section 3.3)

TMG (trained module generator)

a. TMG is run by the system administrator.

b. Outputs either trained content classification modules (CCMs) for the content classification component (CCC) or link-scoring modules (LSMs) for the link-scoring component (LSC). Both CCC and LSC are called by the Spider. TMG will perform both training & testing (the use of the WEKA platform is adopted).

c. What is necessary for TMG to run (i.e. what fields the TMG interface should provide to the user to specify values).
  1. type of module to generate: CCM or LSM,
  2. name of the module to be generated,
  3. user description for the module to generate,
  4. name of the corresponding criterion (when possible),
  5. number of classification categories,
  6. names of classification categories (e.g. pos, neg),
  7. paths to the training collections/corpora that correspond to the specified categories (e.g. for classification categories adult | child | professional, paths to directories containing the corresponding examples),

8. remove html tags or not,
9. remove stop words or not,
10. ML algorithm to use,
11. feature selection method to use,
12. paths to the testing collections/corpora as well as to the testing results file (e.g. an html table to be viewed from AQUA),
13. Output path: where to save the generated model (CCM or LSM).

d1. For the generation of a CCM, the training and testing corpora (e.g. pos | neg) are formed by the CFT.

d2. For the generation of a LSM, the link objects (i.e. link object = anchor + surrounding text + alt text + etc.) are collected and classified to the specified classification categories (e.g. pos | neg) manually by the user. Note that the development of a component assisting the formation of such corpora has to be examined (depending on the decision for the necessity or not of LSM).

e. Each model (either CCM or LSM) is associated to one criterion and one language (both indicated by the system administrator). A model consists of:
- The name of the criterion
- The number and name of the categories
- The language
- The trained model

h. I/O: TMG's input is always utf-8 and its output is a model. This format is language independent.

Actions of the TMG

(Group 1 - Actions provided to the labelling expert user through the expert interface)
There is no such action.

(Group 2 - Actions provided to the system administrator through the sysAdmin interface)
TA1. Generate new modules; select what to generate: CCM or LSM.
TA2. Select the ML algorithm to use.
TA3. Define a number of other parameters (see above, in c), such as the module name & description, the necessary training, testing and results paths, the output path, etc.
TA4. Rename/delete a CCM or a LSM.

6.2 Communication issues

User-Crawler
Users can directly access and run the Crawler. The tool provides a UI through which all crawling parameters can be specified.

User-Spider

Users can directly access and run the Spider (in conjunction with IET, e.g. in monitoring tasks). The tool provides an API through which all spidering parameters can be specified.

Crawler-Spider

The Crawler always outputs lists of URLs in simple text format (.txt; text files containing one URL per line. Such text file paths as well as a number of other parameters for spidering, can be set through Spider's UI.

Spider-CCC

CCC is called by the Spider: the Spider accesses a web document, gets its content (ascii) and calls CCC to classify it. CCC gets either the path of a temporarily locally saved copy of the document or the content itself (as a string).

Spider-LSC

LSC is called by the Spider: the Spider identifies an in-site link, creates a link-object and calls LSC to score it.

CCC-CCMs

CCC exploits all available CCMs (those performing best, per criterion and per language). The user (system administrator) can deactivate a CCM. When giving a document's content to a classification model (CCM), an estimate (or a binary value, i.e 1|0) is returned: for high estimates or for positive values (and when necessary, e.g. IE follows), content is locally stored.

LSC-LSMs

LSC exploits all available LSMs (those performing best, per scoring task and per language). The user (system administrator) can deactivate a LSM. When giving a link object to a classification model (LSM), an estimate (or a binary value, i.e 1|0) is returned: for high estimates or for positive values, links are appended to the spidering queue.

TMG-CCM/LSM

The corpus (represented by paths pointing to collections of content corresponding to different classification/scoring categories), the language, the criterion and the ML technique to employ are introduced to generate a CCM or a LSM.

TMG-CFT

CFT forms the training/testing corpora for CCMs. Those corpora are collections of locally stored web pages/documents, according to categories specified (e.g. pos|neg), in separate directories in the file system. TMG user, through a browsing mechanism, indicates the desired collections directories to generate (train) or test a CCM.

WCC-IET

Considering that portions of the extraction work have to be done in spidering, the issue arising here is how to communicate the extracted values to the Information Extraction Toolkit (IET).

An idea here is to append, in an agreed structure, those values to every document/page which is forwarded from WCC to IET. Alternatively, the option of a surrogate file (e.g. a .dat or a .rdf file) which will contain all necessary metadata (one such file next to every forwarded document/page) has been examined. MedIEQ partners opted for the second solution, as, during a document/page's lifecycle, all components read-from and write-into this metadata file, leaving thus the original document/page untouched. The exact schema for this metadata file remains to be determined.

WCC-LAM
LAM manages and generates quality labels in the following formats: RDF/XML, N3, TURTLE. Therefore, any of the WCC components needing any label-contained information should be able to parse those formats.

WCC-MRM
Three tools from the WCC toolkit may need data handled by MRM: the Crawler, the Spider (in content classification, by CCC) and the TMG. These tools should support the formats in which MRM exports its data.

7. Concluding remarks

In this document we described the architecture of the Web Content Collection toolkit (WCC) and outlined a methodology for using it by different types of users in combination with other MedIEQ toolkits.

As we are currently implementing most of the WCC Toolkit's components, the presented architecture is still subject to change as we identify new challenges during both the development and testing stages.

The architecture should however be robust with respect to changes in the labelling schema, including the addition of more labelling criteria. In the proposed WCC methodology and architecture, handling future additional criteria is being anticipated. CFT (Corpus Formation Tool) and TMG (Trained Module Generator) enable the generation of new CCMs (Content Classification Modules) as well as of new LSMs (Link-Scoring Modules). At the same time, the proposed architecture facilitates the incorporation of such new modules and therefore enables system's support for any new labelling criteria which may be added in the future.

References

[1]. C. Aggarwal, F. Al-Garawi and P. Yu: Intelligent Crawling on the World Wide Web with Arbitrary Predicates. In Proceedings of the 10th International WWW Conference, pp. 96-105, Hong Kong, May 2001.

[2] R. Albert and A.L. Barabasi. Statistical Mechanics of Complex Networks. Reviews of Modern Physics, 74(01):47-97, 2002.

[3] R. Albert, H. Jeong, and A.L. Barabasi. Diameter of the World-Wide Web. Nature, 401:130-131, 1999.

[4] R.Baeza-Yates and B.Ribeiro-Neto. Modern Information Retrieval. Addison-Wesley, 1999.

[5] L. Barbosa and J. Freire. Searching for HiddenWeb Databases. 8th International Workshop on the Web and Databases, Baltimore, USA, pages 1-6, 2005.

[6] P. Boldi, B. Codenotti, M. Santini, and S. Vigna. UbiCrawler: a Scalable Fully Distributed Web Crawler. Software -Practice and Experience (SPE), 34(8):711- 726, 2004.

[7] S. Brin and L. Page. The Anatomy of a Large Scale Hyper-textual Web Search Engine. 7th International World Wide Web Conference (WWW), Brisbane, Australia, 1998.

[8] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J.L. Wiener. Graph Structure in the Web. Computer Networks, 33(1-6):309-320, 2000.

[9] S. Chakrabarti. Mining the Web: Discovering Knowledge from Web Data. Morgan Kaufmann, 2003.

[10] S. Chakrabarti, M. van den Berg, and B. Dom. Focused Crawling: a New Approach to Topic-Specific Web Resource Discovery. Computer Networks, 31(11-16):1623 -1640, 1999.

[11] S. Chakrabarti, B. Dom, R. Agrawal, and P. Raghavan. Scalable Feature Selection, Classification and Signature Generation for Organizing Large Text Databases into Hierarchical Topic Taxonomies. VLDB Journal, 7(3):163-178, 1998.

[12] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced Hypertext Categorization Using Hyperlinks. 1998 ACM SIGMOD International Conference on Management of Data, Seattle, USA, pages 307-318, 1998.

[13] S. Chakrabarti, B. Dom, R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, D. Gibson, and J. Kleinberg. Mining the Web's Link Structure. IEEE Computer, 32(8):60-67, 1999.

[14] S. Chakrabarti, M.M. Joshi, and V.B. Tawde. Enhanced Topic Distillation using Text, Markup Tags, and Hyperlinks. 24th International ACM Conference on Research and Development in Information Retrieval (SIGIR), New Orleans, USA, pages 208-216, 2001.

[15] Charu C. Aggarwal, Fatima Al-Garawi, and Philip S. Yu. Intelligent crawling on the world wide web with arbitrary predicates. In *World Wide Web*, pages 96-105, 2001.

[16] H. Chen and S. Dumais. Bringing Order to the Web: Automatically Categorizing Search Results. SIG CHI Conference on Human Factors in Computing Systems, Hague, Netherlands, pages 145-152, 2000.

[17] J. Cho, H. Garcia-Molina, and L. Page. Efficient Crawling through URL Ordering. 7th International World Wide Web Conference, Brisbane, Australia, pages 161-172, 1998.

[18] Filippo Menczer and Richard K. Belew. Adaptive Retrieval Agents: Internalizing Local Context and Scaling up to the Web. Machine Learning, 39(2/3):203-242, 2000.

[19] P. De Bra, G. Houben, Y. Kornatzky and R. Post. Information Retrieval in Distributed Hypertexts. In Proceedings of the 4th RIAO Conference, pp. 481-491, New York, 1994.

[20] P. De Bra and R. Post. Information Retrieval in the World-Wide Web: Making Client-based Searching Feasible. Computer Networks and ISDN Systems, 27(2):183-192, 1994.

[21] M. Diligenti, F. Coetzee, S. Lawrence, C.L. Giles, and M. Gori. Focused Crawling Using Context Graphs. 26th International Conference on Very Large Data Bases (VLDB), Cairo, Egypt, pages 527-534, 2000.

[22] M. Ehrig and A. Maedche. Ontology-focused crawling of web documents, 2003.

[23] M. Hersovici, M. Jacovi, Y.S. Maarek, D. Pelleg, M. Shtalhaim, and S. Ur. The Shark-Search algorithm. An Application: Tailored Web Site Mapping. 7th International World Wide Web Conference (WWW), Brisbane, Australia, pages 317-326, 1998.

[24] A. Heydon and M. Najork. Mercator: A Scalable, Extensible Web Crawler. World Wide Web, 2(4):219-229, 1999.

[25] J.M. Kleinberg. Authoritative Sources in a Hyperlinked Environment. Journal of the ACM, 46(5), 1999.

[26] H. Mase. Experiments on Automatic Web Page Categorization for IR System. Technical Report, Stanford University, 1998.

[27] F. Menczer. ARCCHNID: Adaptive Retrieval Agents Choosing Heuristic Neighborhoods. 14th International Conference on Machine Learning (ICML), Nashville,USA, pages 227-235, 1997.

[28] F. Menczer, G. Pant, and P. Srinivasan. Topical Web Crawlers: Evaluating Adaptive Algorithms. ACM Transactions on Internet Technology (TOIT), 4(4):378-419, 2004.

[29] A. McCallum, K. Nigam, J. Rennie and K. Seymore. Building Domain-Specific Search Engines with Machine Learning Techniques. In AAAI Spring Symposium on Intelligent Agents in Cyberspace, Stanford University, USA, March 1999.

[30] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web.Stanford, Digital Library Technologies, Working Paper, 1999-1020, 1998.

[31] G. Pant, S. Bradshaw, and F. Menczer. Search Engine -Crawler Symbiosis: Adapting to Community Interests. 7th European Conference on Research and Advanced Technology for Digital Libraries (ECDL), Trondheim, Norway, pages 221-232, 2003.

[32] G. Pant and F. Menczer. Topical Crawling for Business Intelligence. 7th European Conference on Research and Advanced Technology for Digital Libraries (ECDL), Trondheim, Norway, pages 233-244, 2003.

[33] G. Pant, P. Srinivasan, and F. Menczer. Crawling the Web. Web Dynamics - Adapting to Change in Content, Size, Topology and Use, pages 153-178, 2004.

[34] S. Raghavan and H. Garcia-Molina. Crawling the Hidden Web. 27th International Conference on Very Large Data Bases (VLDB), Roma, Italy, pages 129-138, 2001.

[35] J. Rennie and A. McCallum. Using Reinforcement Learning to Spider the Web Efficiently. 16th International Conference on Machine Learning (ICML), Bled, Slovenia, pages 335-343, 1999.

[36] V. Shkapenyuk and T. Suel. Design and Implementation of a High-Performance Distributed Web Crawler. 18th International Conference on Data Engineering (ICDE), San Jose, USA, pages 357-368, 2002.

[37] S. Sizov, M. Biwer, J. Graupmann, S. Siersdorfer, M. Theobald, G. Weikum, and P. Zimmer. The BINGO! System for Information Portal Generation and Expert Web Search. 1st Conference on Innovative Systems Research (CIDR), Asilomar, USA, 2003.

[38] N. Stojanovic. An approach for ontology-enhanced query refinement in information portals. In IEEE International Conference on Tools with Artificial Intelligence, pages 346-357, 2004.

[39] N. Stojanovic, R. Studer, and L. Stojanovic. An approach for step-by-step query refinement in the ontology-based information retrieval. In Proceedings of the IEEE/ACM Conference on Web Intelligence (WI), pages 120-137, 2004.

[40] K. Stamatakis, V. Karkaletsis, G. Paliouras, J. Horlock, C. Grover, J. Curran, S. Dingare, Domain Specific Web Site Identification: The CROSSMARC Focused Web Crawler, In Proceedings of the 2nd International Workshop on Web Document Analysis (WDA 2003), Edinburgh, UK, 2003.

[41] O. Baujard, V. Baujard, S. Aurel, C. Boyer and R. D. Appel, Trends in medical information retrieval on internet, Computers in Biology and Medicine, Volume 28, Issue 5, September 1998, pages 589-601.

[42] Gaudinat A, Ruch P, Joubert M, Uziel P, Strauss A, Thonnet M, Baud R, Spahni S, Weber P, Bonal J, Boyer C, Fieschi M, Geissbuhler A., Health search engine with e-document analysis for reliable search results, Int J Med Inform. 2006 Jan; 75(1):73-85.

APPENDIX: Glossary

TERM	ROLE	DESCRIPTION
API	IT term	Application Programming Interface
AQUA	The MedIEQ prototype system	Assisting Quality Assessment system (the MedIEQ system)
AQuMed	Project partner	Agency of Quality in Medicine
CCC	S/W component	Content classification component. Calls several CCMs.
CCM	S/W component	Content classification module
CFT	S/W component	Corpus formation tool
CFT1	S/W component	Corpus formation tool 1
CFT2	S/W component	Corpus formation tool 2 (Scrapbook, a Mozilla plug-in)
Crawler	S/W component	A tool that searches the web to discover interesting urls
HUG	Project partner	Geneva University Hospitals
HUT	Project partner	Helsinki University of Technology (currently TKK)
IET	S/W toolkit	Information extraction toolkit
i-sieve	Project partner	i-sieve Technologies Ltd.
LSC	S/W component	Link-scoring component. Calls several LSMs.
LSM	S/W component	Link-scoring module
MedIEQ	The Project	Quality labelling of medical web content using multilingual information extraction
ML	IT term	Machine learning
MRM	S/W toolkit	Multilingual resources management toolkit
MUA	S/W toolkit	Monitor-update-alert
NCSR	Project co-ordinator	National Centre for Scientific Research (NCSR) "Demokritos"
RDF	IT term	Resource Description Framework
RDF-CL	IT term	RDF Content Label
Scrapbook	S/W component	A Mozilla plug-in, proposed for use as a Corpus formation tool (CFT2)
Seal	IT term	A trustmark (see above)
Spider	S/W component	A tool that searches inside websites to discover interesting content
S/W	IT term	Software
TKK	Project partner	Helsinki University of Technology (ex HUT)
TMG	S/W component	Trained module generator
Trustmark	IT term	A visible sign that the content has a certificate. A visible sign means that there is a label with claims. A 'Claimmark' so to speak. The XHTML CSS icons being examples of such a "self-certification" sign.
UEP	Project partner	University of Economics in Prague
UI	IT term	User interface
UNED	Project partner	Universidad Nacional de Education a Distancia
URL	IT term	Uniform Resource Locator
URI	IT term	Universal Resource Identifier
WCC	S/W toolkit	Web content collection toolkit
WMA	Project partner	Web Medica Acreditada
WP	Project term	Work package