

A Methodology to Retrieve, to Manage, to Classify and to Query Open Source Information

- Results of the OSILIA Project -

Stefan Scheer	ISIS - RMDS, Anti-Fraud Information Management
Ralf Steinberger	ISIS - RMDS, Anti-Fraud Information Management
Giovanni Valerio	Origin, Milan
Paul Henshaw	ISIS - Reliable Information Technologies

December 2000

T.P. 361
I – 21020 Ispra (VA)
Italy
Tel: + 39 0332 78{5683,6271}
Fax: + 39 0332 789098
Email: {stefan.scheer,ralf.steinberger}@jrc.it
URL: <http://www.jrc.org/isis/atia/>

Table of Contents

1. Introduction	6
1.1 Scope	6
1.2 Rationale	6
1.3 History and Objectives of the OSILIA project	7
2. The Methodology	8
2.1 General Approach	8
2.2 Detailed Approach	8
2.3 Using commercial services or software – an alternative	9
2.3.1 Commercial news clipping services	9
2.3.2 Commercial crawlers	9
2.4 Crawler	10
2.5 Converting	15
2.6 Cleaning	16
2.7 Comparing	18
2.8 Classification	18
2.9 Keyword Assignment	19
2.10 Querying and Displaying	20
3. The OSILIA project	21
3.1 Sources and Parameter Files	21
3.2 Databases (destinations)	24
3.3 Special Filter Mechanisms	24
3.4 Examples of Using GIST	25
4. Results and Discussion	27
4.1 Experiences Made	27
4.1.1 Performance	27
4.1.2 Quality of Downloaded and Modified Files	28
4.1.3 Evaluation of the Retrieval Results	28
4.2 Possible Improvement / Development and Further Applications	30
4.2.1 Avoiding multiple copies of the same document	30
4.2.2 Improving the extraction of the core article from the web page	31
4.2.3 Avoiding index files	31
4.2.4 Making use of the potential of meta news sites	32
4.2.5 Applying the techniques to further domains	32
5. References	32
6. Attachments	33

Abstract

The term *information overflow* points to both a positive and a negative facet of the electronic information age: the positive aspect is that almost unlimited amounts of information are nowadays available to the information-seeking person; the negative aspect is that it becomes increasingly difficult to find the relevant information in enormous information sources such as the internet.

The OSILIA project (*Open Sources Intelligence Library on Internet Abuse*) had the aim of trying to find a solution to this information retrieval problem by developing tools which identify, retrieve, classify and store documents for a specific domain fully automatically. The specific task of the exploratory project, which lasted from March to September 2000, was to identify online newspaper articles on the abuse of the internet and to classify the resulting documents into four domains. The result is a working prototype that is still downloading new documents every day. The tools and techniques that were developed for OSILIA are now ready to be applied to further application domains.

The introduction chapter of this JRC Technical Note outlines the scope and background of the project. The second chapter describes the application-independent methodology used in the system. The third chapter discusses the OSILIA-specific parameters and the decisions taken during the short project. Among other things, this chapter lists the approximately twenty German and English language news sites searched by the crawler (3.1) and the search word combinations used (the *query*, 3.3). The fourth chapter, finally, provides an evaluation of the performance of the prototype application (4.1) and points to future development (4.2).

The major software components that the JRC developed during the project period are:

- (1) a software agent which visits a specified set of online news sites every day and downloads all the new web pages which satisfy certain criteria, among which the condition that a certain combination of search words had to occur in the text. This part of the work is mainly described in sections 2.4 and 3.1;
- (2) software which cleans this first collection of potentially relevant documents in order to get rid of non-pertinent information such as advertising text, links to other news of the day, links to related articles, etc. (sections 2.6 and 3.3);
- (3) software which subjects the cleaned text to a more thorough filtering process and which classifies it into one or more of the user-specified classes (sections 2.6 to 2.8 and 3.3);
- (4) a database with a user interface which allows to search the remaining document collection using a variety of mechanisms and to view the results, using a traditional web browser (sections 2.10 and 3.4). The URL for the database with the documents downloaded up to September 2000 is <http://osilia.jrc.it> .

The results of the automatic procedure were compared to those of a manual newspaper clipping service that was carried out by the JRC's *Public Relations* office. The comparison showed that the crawler was rather thorough and did not miss any important articles. The major problems of the current software prototype are that it downloads multiple copies of the same article and that it retrieves a small number of web pages which are not relevant (4.1). Solutions to these problems and some further ideas of how to improve the performance and functionality of the software are listed in section 4.2.

1. Introduction

1.1 Scope

This report seeks to promote a certain methodology that has been worked out for the purpose of automating several interconnected tasks related to

- downloading relevant documents from publicly available sources on the internet
- cleaning and filtering these documents
- classifying these documents
- querying the collection of these documents.

This methodology is a spin-off from work that had been carried out in the context of the OSILIA project (*Open Sources Intelligence Library on Internet Abuse*). Therefore this report will present the methodology in chapter 2, followed by a detailed description of the OSILIA project and the necessary parameterisation of the software in chapter 3.

1.2 Rationale

The electronic age and the new medium internet caused a radical change to information and knowledge management. The availability of *open sources* created the new need for intelligent tools which help to find and manage large amounts of information. Internet search engines are getting better at helping users to satisfy ad-hoc information needs, but for many private persons, companies, government and other organisations, and particularly for intelligence departments, this ad-hoc information retrieval tool is not enough. The methodology presented here has the general aim of providing a means to search and collect open sources continually and automatically for application-specific documents and to make them available for later use.

The increasing deployment of computers and their intense interconnection also opens the doors to cyber-crime such as privacy abuse and damaging attacks caused by viruses, hacking, cracking, the destruction of information resident on computers and the obtention of unauthorised information during information exchange. It is therefore necessary to protect intellectual property, the privacy of private persons and the interests of modern type businesses like e-commerce.

The transnational character of the internet makes it particularly difficult to control and encourages an improper or non-respectable utilisation, such as the criminal usage of information deployment or the online offering of improper or criminal services. As recent events and public discussions showed, policy makers are behind in hindering such unlawful utilisation of online media. Therefore, they urgently need advice on the methods used and on the existing malicious applications so that they can prepare the ground for prosecution. In order to draw conclusions, policy makers need to know where, when, and in which context abuse or violation has taken place. They need to know about

- the time and place of such **events**,
- new **types** of hacking, cracking, abuse or similar methods,
- different ways of **reporting** on those subjects.

One of the ways to get information on these issues is to have a look at the internet itself. As a modern medium it makes any type of information publicly available; on the

other hand, the amount of information to search for or to control is too big, especially when regularly updated sources have to be taken into consideration. Policy makers therefore need automated systems that will assist them in ensuring the dependability of the information society and to guarantee a secure, safe and ethical use of online information exchange and services.

The methodology presented here should devise an automatic procedure which should be flexible enough so that it can be tuned to specific user needs. On the other hand, it was clearly intended to limit the methodology's scope to preparing and enabling policy making, and not to utilise it for operational prosecution.

Problems of the approach presented here are not only caused by the masses of available documents and their varying quality and reliability. For this reason, clear and specific mechanisms have to be invented that not only look at relevant web sites and that limit the downloads to those articles that really focus on what the user is interested in; the mechanism also has to strip the downloaded web pages from irrelevant information so that only the document itself remains.

It is a feature of many open sources that they are relatively short-lived. Web sites change often. This is particularly true for newspaper web sites, which are often only available for one or two days. For this reason, it is not enough to full-text index the documents found, as done by most commercial software. Instead, the texts must be downloaded and stored locally in order to keep the information.

1.3 History and Objectives of the OSILIA project

Due to the transnational character of the internet, it is very difficult to limit fraud on the internet. Recently, this issue is being discussed widely in the media and several national governments have made statements that the abuse of the internet is a serious issue which has to be dealt with. During a visit at the JRC, some members of the European Parliament also mentioned this issue to the director of the *Institute for Systems, Informatics and Safety* (ISIS), who then gave the impact for JRC scientists to work on this subject. The OSILIA project (*Open Sources Intelligence Library on Internet Abuse*) started in March 2000 and ended six months later.

Main objectives of the OSILIA project were to use agent technology to crawl online news sites regularly and to download and store documents covering cyber-crime-related issues such as hacking, viruses, denial-of-service attacks and paedophilia on the internet.

OSILIA is inherently based on those main procedures that are proposed by the general methodology: the potentially relevant texts which have been downloaded by the crawler (described in section 2.4) are then cleaned, filtered and classified automatically (sections 2.5 to 2.8). Furthermore, each of the remaining documents gets keywords assigned automatically (section 2.9) and both the keywords and the texts are then stored in a searchable database with a web interface (section 2.10).

The project goal was to show that the process of finding, retrieving, analysing, classifying and storing the documents can be automated. Due to the short time that was allocated to the project (six months), the main focus was to show the feasibility of the approach. Therefore, the performance of the system can only be indicative. Nevertheless a variety of experiences can be reported. A summary of the experiences made during phase 1 plus a list of potential extensions and improvements are discussed in chapter 4.

2. The Methodology

2.1 General Approach

The general methodology of the retrieval and classification process is an m-to-n mapping of open source information into categories of specific interest allowing the subsequent querying of that information, with

- n being the number of categories (data bases) to contain user-relevant information
- m being the number of web pages being searched for
- the mapping consisting of a number of subsequent procedures to manage (analyse and modify) the open source information.

Please note the following: *Open source information*, for this methodology, means web-based information and *categorising* documents means assigning them to one or more given classes (of a database), which are user-defined. The resulting document collection including the information on the classes, may then be searched, using web-based or other querying interfaces.

The suggested m-to-n mapping entails that:

- a source may not be obtained
- a source may be obtained, but not categorised
- a source may be modified prior to categorisation
- a source may be obtained and assigned to more than one categories.

In addition, basic principles are that

- it is not allowed to add information; only the cutting of information during the modification process should be allowed;
- meta data should be attached to each (modified) source, containing the source, the publication and download dates, as well as other information.

2.2 Detailed Approach

The automatic procedure, as it is presented in this methodology, consists of the following single steps:

- A *crawler* obtains documents which satisfy user-defined selection conditions.
- A *converting program* saves HTML page sources in a textual format.
- A *cleaning program* extracts the main text parts from the source page (thus taking out advertising and other irrelevant information).
- A *compare program* ensures that not more than one copy of the same source exists (thus excluding formally equal copies).
- A *categorising program* assigns each document to one or more classes if a sufficient number of search words was found; otherwise no assignment will be done.
- A *keyword assignment program* identifies the most significant words in each (categorised) document.
- A *client-server communicator* allows to query the database for user-relevant documents and to display them.

2.3 Using commercial services or software – an alternative

The main functionality of the tool set developed and put together by the JRC are the automatic crawling and gathering¹, the filtering, the classification and the storage of the documents. For some of these tasks, commercial services or commercial software exist. The reasons for developing an in-house approach will be presented in the following sections.

2.3.1 Commercial news clipping services

There are several companies which produce or collect information and sell it to interested parties. Two of them are the news agency *Reuters* and the *British BBC Monitoring* service². They offer the service of forwarding all of their news items which fulfil certain criteria to paying customers. BBC claims to cover news sources in one hundred languages and it is part of their service to translate all articles into English.

The approach of the US firm CYBERALERT³ is different in that it not only collects documents from a large number of English language news sites, newswires, etc., but it also monitors Usenet news groups. This means that specialised information which is not published in the conventional way is included in their list of sources. As a paid service, this firm gives access to the full-text index of all the sources they monitor so that their clients can search the database and download the full texts of the documents they are interested in. The major drawback of CYBERALERT is that it is limited to English language texts. The cost of CYBERALERT's service is a one-off setup fee of a few hundred US-Dollars and approximately 360 US-Dollars per search topic each month, where each search topic consists of a Boolean query.

This and other news-providing services are certainly a valuable source of information and could have been useful for the OSILIA project. However, in the long term, the cost will be considerable and the restriction to the English language is a serious drawback in a European environment.

2.3.2 Commercial crawlers

A crawler is a software agent which regularly and automatically visits web sites, searches them for potentially relevant documents and either downloads or full-text-indexes them. The crawler developed at the JRC (see section 2.4) goes to a pre-defined set of web sites, looks for a specific combination of search words and downloads the pages if they satisfy a set of user-defined conditions. The main conditions are that the web page must contain a combination of search words and that it is of one of the given formats.

Before developing our own crawling software, we considered purchasing a licence for commercial software products with similar functionality. Software considered by us was IBM's *Intelligent Miner for Text*⁴, Verity's *Agent Server*⁵, Excalibur's *Internet Spider*⁶

¹ For a sample list of online newspapers, see <http://www.gt.kth.se/publishing/news.html>

² See <http://www.monitor.bbc.co.uk/>

³ See <http://www.cyberalert.com/>

⁴ See <http://www.software.ibm.com/iminer/fortext/> for details

⁵ See <http://www.verity.com/products/agentserver/> for details

⁶ See <http://www.excalibur.com/> for details

and *Teleport Pro*⁷ by Tenmax. The first three of the mentioned products offer a much wider functionality and are therefore rather expensive. According to information by sales people over the telephone, the IBM product would cost a minimum of 25.000 Euro and Verity's product would cost us between 50.000 and 90.000 US-Dollars. It seems furthermore that the standard versions of both Verity's and Excalibur's software full-text index the retrieved texts, but they do not download them so that an even more expensive customised version of the software would have to be bought. Such an investment may be worthwhile if the tool were to be used a lot and regularly. However, as the goal of the OSILIA project was to show the feasibility of the application within very little time, we decided to check out the cheaper options, i.e. using the product by Tenmax, which costs only 40 US-Dollars.

It soon turned out that *Teleport Pro* did not meet our requirements because it did not allow Boolean queries to focus the downloaded files on the most relevant documents. For instance, it was not possible to specify within *Teleport Pro* that only documents containing both the words *paedophilia* and *internet* should be downloaded because documents containing only one of these words are of no interest. Therefore, it was decided to develop an in-house system, using public domain PERL components. This tool is described in section 2.4.

Through the development of our own crawler it would not only possible to ensure that all the required functionality would be covered, but also that it will be possible to amend the software to our future needs.⁸ The flexibility and the ease of usage of the parameter files, which allow users to adopt the system to their own needs, are certainly an asset and justify therefore the JRC's own development.

2.4 Crawler

The crawler is a PERL program that tries to retrieve and to download open sources following a list of user-defined parameters. The way to use the program is

```
[perl] crawler.pl parameter_file_path [project_name]
```

The parameter file contains the user-defined options. It is optional to specify a project name in the command line. Although it is not mandatory to create different parameter files for searching various web sites, the user might be advised to create a specific parameter file for each web domain that will be searched.

Parameter file:

The parameter file is a 'free format' text file that can contain both comments and white lines.

Any line beginning with the '#' symbol is ignored, as is any empty line:

The general format of a line is:

```
option=value
```

When the *value* is a list of strings, they have to be separated by spaces and/or semicolons:

⁷ See <http://www.tenmax.com/teleport/pro/> for details.

⁸ For example, it already turned out that the crawler, as it is, could somehow be re-used for similar tasks in JRC's IIMS and ConTraffic projects.

option=*value*{[;|s] *value*}*

Option names are not case-sensitive, so upper and lower case can be mixed to improve readability. Some option values, however, are case sensitive (i.e. the file path fragments of URLs). Options in square brackets are not mandatory. Options in single quotes have to be written as such.

List of options:

[‘Project’ = name]

default value: **crawler**

The project name is the base name for the log, status and index files (see below).

[‘Destination’ = path]

default: **./work**

The path of the folder which will contain the crawler results. It may be either absolute or relative to the application directory.

‘StartingUrl’ = URL {[;|s] URL}*

The list of starting URL’s is the list of URLs the crawling starts from.

[‘Exclusions’ = URL {[;|s] URL}*

default: *none*

The exclusions list may contain a list of URLs (either single pages or whole domains) which the crawler has to avoid visiting. Specifying a list of exclusions might become important in order not to download many pages which are of no interest.

[‘Query’ = Boolean query string | perl match string]

default: *none*

Although the query is optional, it is used to identify those pages which match it. The query may be written either as a Boolean query string (the AND, OR and NOT operators are allowed) or as a PERL regular expression (see PERL language specification).

[‘ApplyQuery’ = ‘bodyOnly’ | ‘headerOnly’ | ‘both’]

default: **both**

This option specifies whether the search must consider only the body of each page (in HTML marked up within <BODY> ... </BODY>), only the headers (in HTML marked up within <HEAD> ... </HEAD>), or the whole document. Excluding the header might become interesting as it allows to disregard, for example, meta information that has nothing to do with the text of the page.

[‘TrimTags’ = ‘all’ | ‘keepmeta’ | ‘no’]

default: **no**

With this option it is possible to search for information (for example, according to the query) considering only the pure text (TrimTags = all), the text and the meta tags (TrimTags = keepmeta), or all of the original html page (TrimTags = no). Keeping meta tags could be interesting because those tags often contain important information on the underlying text; information written inside, for example, anchor tags (<A> ...) is of much less importance.

```
['FileTypes' = ('text'|'image'|'audio'|'video') {( ';' | \s ) ( 'text' | 'image' | 'audio' | 'video') }*]
```

default: **text**

It can be specified which types of files the crawler is allowed to search and to download. This list is matched against the content/type attribute of each retrieved document.

```
['Roaming' = 'yes' | 'no' | 'directory' | 'site' | 'world']
```

default: **no**

This important option declares whether (**yes**) or not (**no**) the crawler is allowed to follow links outside the current page domain. Specifying **directory** means to limit the crawler's search area to pages whose URL's exactly start with what has been specified as a starting URL; specifying **site** is equivalent to **yes**, and, finally, with a specification like **world** is equivalent to **no**. Example: Suppose that we have www.mydomain.com/my-subdir given as a starting URL. The option **directory** tells the crawler to look only after sites which start with that URL. The option **site**, on the other hand, also allows looking for sites which start with www.mydomain.com/sub-dir2 (see also Figure 1 below).

```
['DownloadPages' = 'yes' | 'no']
```

default: **yes**

The crawler must know whether (**yes**) or not (**no**) it must store the text of relevant pages according to the destination option (see above). Specifying **no** means not to download a relevant page (only the index file will contain hints on whether a file is relevant).

```
['DownloadImages' = 'yes' | 'no']
```

default: **no**

Although it may sometimes be of interest to download inline images (images in the text – not those linked), it is basically not allowed to do this (see also Figure 1 below).

```
['SearchDepth' = integer number]
```

default: **0**

This rather useful option tells the crawler how many links it has to follow from the starting page(s). Specifying, for example, **0** means to download only the single starting page. Specifying **1** means to follow up to 1 link (to another page) that may be referred to in the HTML file.

['Timeout' = integer number]

default: **90**

The integer number states the number of seconds after which the crawler should give up waiting for a requested page. Having many pages to look for, it might be useful – for performance reasons – to limit the waiting time (for each page).

['Retries' = integer number]

default: **3**

Similarly to the timeout option it might be useful to limit the number of trials a (currently) not available page should be requested before giving up.

['AgentLifetime' = integer number ('h' | 'm' | 's')]

default: **10m**

This default value of 10 minutes tells the crawler how long its program is allowed to run. **h** stands for hour, **m** for minutes, and **s** for seconds.

['MaxFiles' = integer number]

default: **1024**

This option specifies how many files the crawler is allowed to download and to store. For obvious reasons it might be necessary to limit the number of files to be downloaded.

['MaxSize' = integer number]

default: **10**

Similarly to MaxFiles, with MaxSize it might be useful to limit the total disk space that should be allocated to the crawler's retrieval results.

['RetainOnlyLastVersion' = 'yes' | 'no']

default: **yes**

This option declares whether or not the crawler must substitute the stored pages with new versions. **No** means that each stored version of the same page will be given a different name. The naming principles for the downloaded files could become crucial the more (and probably equally written) files are going to be downloaded.

Index file

Since with each subsequent run, the crawler may be asked to retrieve the same documents over and over, the crawler creates a list of the already examined files along with their timestamp in the directory specified by the **Destination** option, to avoid downloading the same files every time.

The index is a SDBM file, named **destination/project_index.db** with a valid **project**, or **destination/crawler_index.db** otherwise. To access the index file, it must be *tied* (PERL

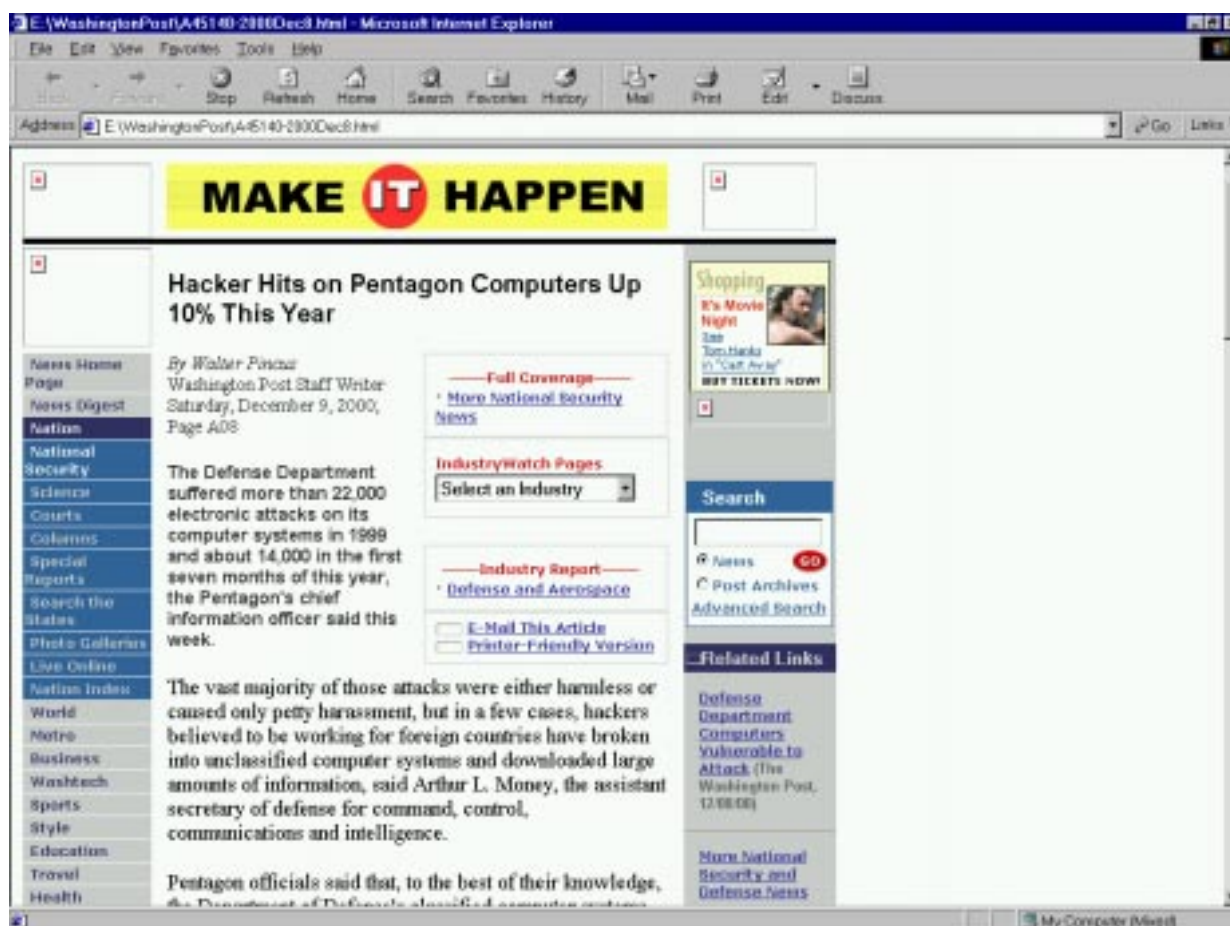


Figure 1: Re-displaying a *Washington Post* web page shows empty images (that have explicitly been excluded from the download). Normal buttons remain visible as long as they are no images that should be excluded. Buttons, of course, remain clickable if the linked page has been downloaded as well. Some images, however, can still be seen; this is due to the fact that the underlying link is a roaming link, i.e. that has a completely different URL. Roaming (in this specific case) is not allowed. Links of that kind are automatically not looked at without caring whether there is an image behind.

function *tie*) to a hash variable.⁹

Each SDBM entry is a couple **key, value**:

- the **key** is the plain URL (no fragment part, no query part) of the retrieved document eventually followed by a vertical bar '|' and a version sequence number (see below)
- the **value** is composed of four elements, separated by a vertical bar '|':
 - a number specifying if there are more versions of the same document

⁹ Usually this is done through a simple PERL script, for example:

```
use Fcntl;
use SDBM_File;
my %urlIndex;
my $destination = shift || die("Usage: [perl] dumpindex.pl destination_dir
[projectname]\n");
my $project = shift || 'crawler';
my $indexfile = "$destination/$project".'_index.db';
tie %urlIndex, SDBM_File, $indexfile, O_CREAT|O_RDWR, 0755;
while(($key,$value) = each %urlIndex) {
    print "$key\t$value\n";
}
untie %urlIndex;
```

- **1** means no other versions
 - **n** means there are other **n-1** versions of this document
 - In this latter case the key for the subsequent versions is **key|x** with **x** running from 2 to **n**
- a **1** if the document has been matched against the search query and found relevant, **0** otherwise
 - the timestamp of the retrieved document (as the number of seconds after Jan-1-1970)
 - the local file name for the retrieved document.

Log file

While the crawler works, it will write a report of its proceedings (new and updated files) into a log file in the directory specified by the **Destination** option. This allows a user interface to show what is going on and to know whether everything went fine.

Status file

While the crawler is working, it creates a single line status report file in the directory specified by the **Destination** option, specifying if it is running, whether it has completed the job, or whether its job was interrupted due to an error.

This file, too, aims at allowing a user interface to show what is going on and whether everything went fine.

Each file visited is tagged with one of the following values:

- **New**: the file is unknown to the system and its name will be stored
- **Updated**: the file is known to the system, its contents, however, have changed
- **No change**: the file is known to the system and changes have not been noticed.

2.5 Converting

The methodology proposes the usage of the **html2text**¹⁰ program. Basically it takes away the HTML tags, the header part together with all its meta tags¹¹, and inherent structures like tables, headers, etc.

The **html2text** program, as it is used here, takes all newly arrived HTML files, one by one, and puts the result into the sub-directory “converted”. The resulting files keep their main file names with the extension “txt”. Files of the xyz directory are converted as follows:

```
html2text -t xyz -oxyz\converted
```

The option **-t** assures that the converted files have a **.txt** extension. The option **-o** specifies the output directory.

¹⁰ see www.tenmax.com/tools/html2text

¹¹ If this information has not already been left out due to the “TrimTags” option in the parameter file (see 2.4).

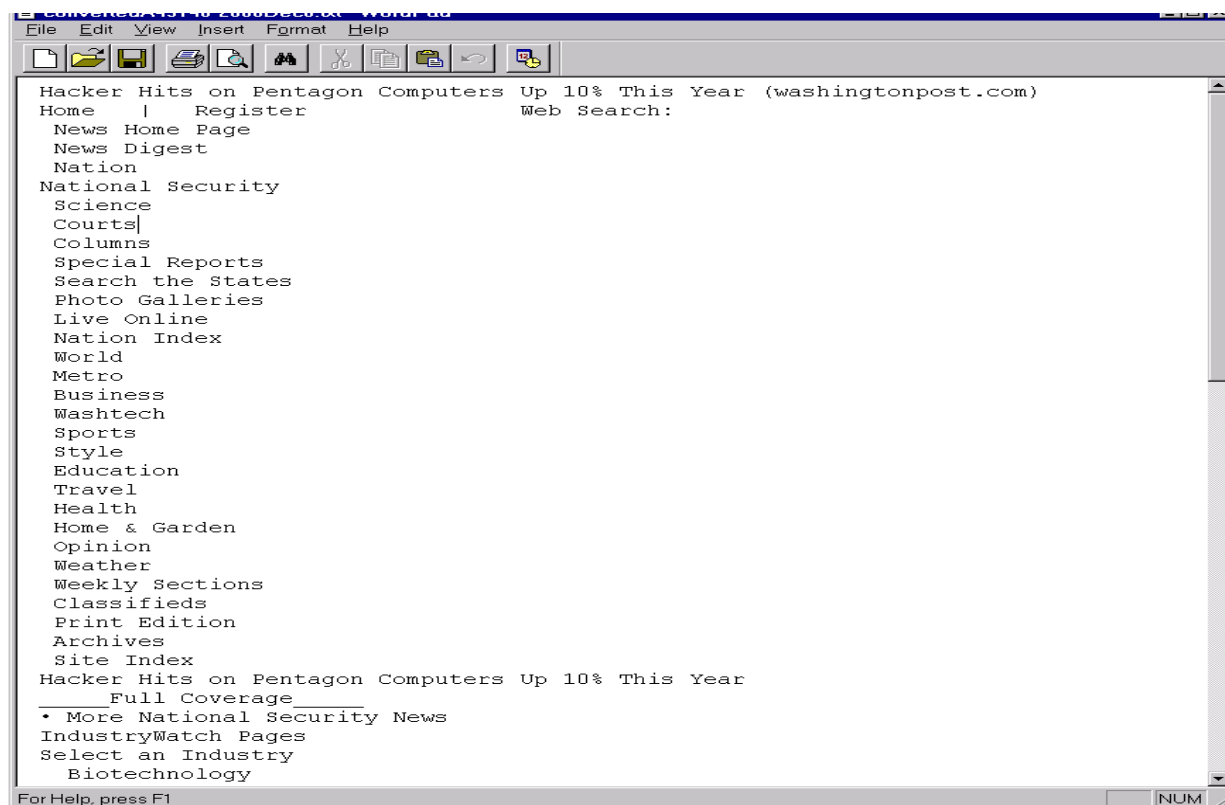


Figure 2: An HTML-text conversion (of the HTML file of Figure 1) reveals some of the difficulties encountered: the texts of the buttons remain as plain text.

As it can easily be conceived, these converted files contain all remaining plain texts even if the contexts of these texts have been lost (see Figure 2). Typically, texts on web pages are surrounded by links to other pages, header texts, etc., which will all remain after the *html2text* program has done its work.

In order to get the nucleus of a (news) page some more cleaning mechanisms have to be applied.

2.6 Cleaning

The cleaning program is a PERL program that takes two command-line parameters: 1. the (short) name of the news site¹², 2. the path that contains the files written in textual format. Consequently the cleaning program has to be called for each news site.

```
[perl] clean.pl news_site_short_name destination_path
```

The main task of the program is to strip a text of unrelated information so that only its main informative parts will remain. The idea behind this is that each converted file contains lines that

- are common or typical to many files (of that site)
- are of no use when extracting the main ingredients.

It is assumed that the converting program still has left “some structure”. For example, a header or a clickable text remains in one line and will not be mixed with subsequent text parts.

¹² This, however, presupposes hard-coded short names of sites.

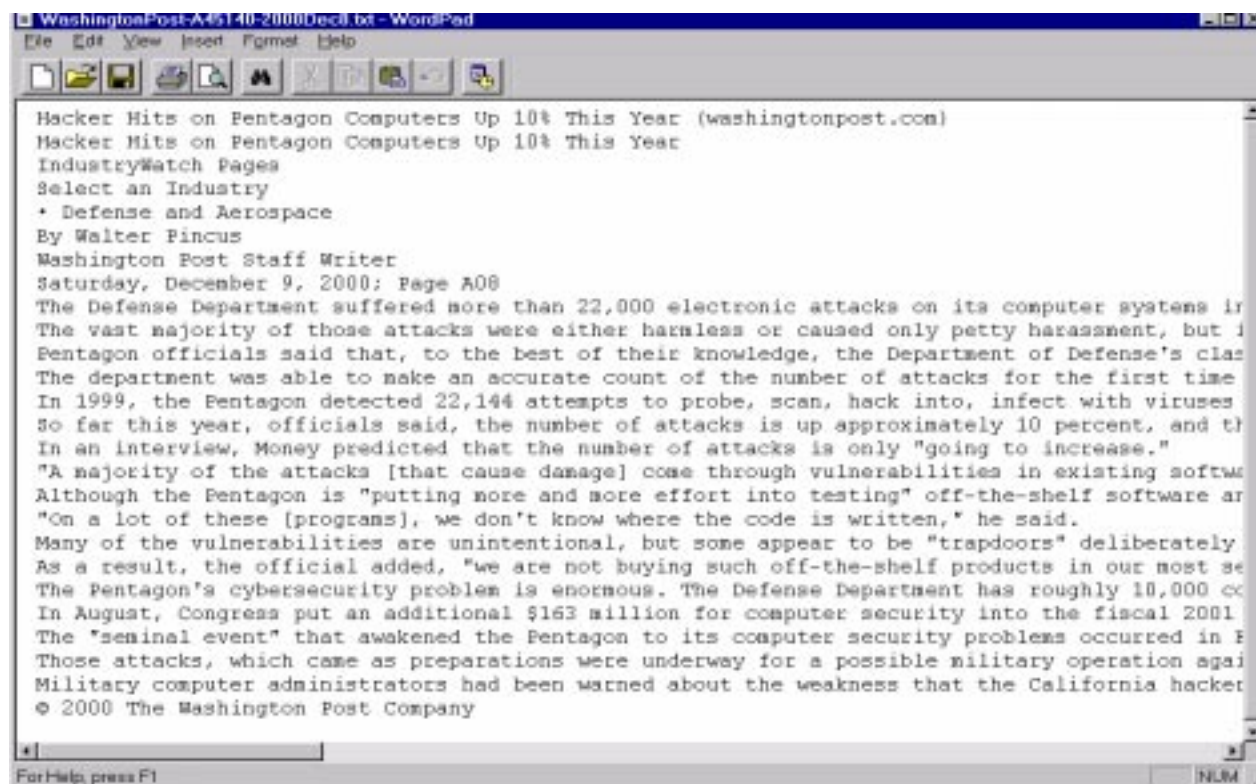


Figure 3: Knowing the structure of the file of Figure 2 and assuming that this structure is applied to all files of that type, a PERL program can be written with which the nucleus of the *Washington Post* article may approximately be obtained.

Therefore, in order to get rid of non-usable parts and to identify the main ingredients of a file, it has to be noted that:

- each file under investigation has to be checked separately,
- each line (of a file) has to be checked whether it matches the relevant search words specified for that site and responsible for exclusion,
- each line that does not match will be saved so that, at the end, a modified file will be reconstructed.

It is obvious that for news sites that reveal many different structures, or that are rather dynamically created, it will be increasingly difficult to find such “common” structures. As a result the cleaning mechanism may not be able to delete all unwanted pieces of text.

The way documents have to be cleaned in order to get the core information is news site-dependent; therefore there has to be an application-specific file containing the relevant mini-procedures for each news site. Procedures concern: (1) specifying patterns in lines that indicate the new end of the (modified) document (i.e. from which line onwards the text in the file has to be cut off), (2) specifying patterns in lines and how these should be substituted.

At the end, cleaned and subsequently modified files are re-written to their original file names.

In most cases, the cleaning program works well enough to delete most irrelevant information from the web page so that the text can be further analysed (see also Figure 3). However, another problem remains, which is that there are often multiple copies

of the same articles. For the sake of good performance, it is crucial to detect those identical files and to delete multiple copies.

2.7 Comparing

It was considered necessary to insert a *compare* program into the chain of programs which spots multiple copies of the same articles in the collection. The crawler often downloads several versions of the same article because it cannot check the regular update of web sites. The reasons for downloading multiple copies are:

- Web pages are sometimes organised as collecting pages, i.e., the creators of the site append new articles to previously existing ones. As the resulting page looks different from the one downloaded earlier on, the crawler will download a new version of the same article each time it visits the site.
- Some articles are kept on a site for a long time. In these cases, the web pages often contain the current date, which is updated automatically. If the crawler looks at such web pages, it will download the same articles again because the text of the page looks slightly different.
- News sites sometimes publish latest stories on the main page; those stories, however, may also be accessible at different places in more specific folders. In these cases, the crawler will download the same article again (if provided with sufficient search depth).

A PERL program has been written that basically compares two text files to see whether they are equal; the program terminates as soon as a single line has been found that does not have the equivalent representation in the other file. Empty lines count as well.

```
[perl] compare.pl source_path/(file_name | *) [*] [-d] [-h html_files_path]
```

The first file name must be preceded by a full path specification. If the second file name is given as “*”, the program will compare the first file of the specified directory with all other files of that directory. No second file name given will be substituted with “*”. To specify a path with a file name as “*” means comparing any file (of the directory) with any other remaining file of that directory.

Specifying ‘-d’ as third command line option means to automatically delete files that are equal to the first file name given.

Specifying ‘-h’ as fourth option followed by a path means to delete also the corresponding HTML files that should be found in the given path.

2.8 Classification

The classification is carried out by a PERL program (with no command line options) with the following main functions:

- (a) It reads the application-dependent information (from a separate file) that reveals files of no interest (to be discarded), key patterns, category counters and counting mechanism.
- (b) For each site it throws away files that do not contain any useful contents, for example index files that only serve as an overview and that list links to a variety of more detailed sites.

- (c) Each remaining file under investigation is checked for the existence of a variety of key patterns.
- (d) A scoring mechanism uses these key patterns and the weights assigned to each of them to calculate a score for each class to which the document could be assigned. This is done by adding the weight of a key pattern found to the class to which the key pattern belongs.
- (e) For a document to be assigned to any class: the score for this class has to pass the threshold set for this class.

Each file will be deleted from its original location: the files that passed the classification process will be saved elsewhere, the files that did not pass this process are of no further interest. As saving function the system call “copy” is actually used which automatically checks at the destination side whether the same file already exists. As destination various databases could be created.

It should be noted that this does not exclude the saving of files with the same content. The reason is that the download procedure (the “crawler”) has to invent somehow file names. This is not done in a deterministic way.

It should also be noted that this procedure does allow a file to be saved multiply: to get a file saved means that the relevant threshold has been reached. Since the numbers for the thresholds are collected simultaneously, it may, of course, happen that one and the same file ends up in more than one category.

For practical reasons the classification program has been split into two separate programs, one analysing English language files, the other analysing files written in German, thus having the syntax

[perl] classifyE.pl

or

[perl] classifyD.pl

2.9 Keyword Assignment

Retrieving new documents automatically and every day will typically lead to a very large collection of documents so that means should be provided to facilitate the retrieval of documents users are interested in. Assigning keywords to each text is one step towards this goal. Therefore, the JCR’s keyword identification tool was used to identify a ranked list of the most significant words of each text. The tool uses statistical information to identify outstanding terms of a text by comparing the word frequency of each text with a ‘normal’ or expected word frequency (as derived from general purpose reference corpora). Table 1 shows the automatically identified keywords for the text displayed in Figures 1 to 3. For detailed information on the JRC keyword identification tool, see [1].

These automatically derived keywords can be used for two purposes. Firstly, before reading a text found in the database, users can have a look at the text’s keyword list to get an idea of the approximate contents of this document and to decide beforehand whether or not the document is really useful. Secondly, it is possible to use the keyword lists for an advanced search function offered by the query interface, in addition to the usual full-text search (see Figure 4 in the attachments). Allowing the user to choose terms from the list of automatically identified keywords should help to increase the

Table 1. Keywords identified automatically for the text in Figure 1, plus their *keyness*.

pentagon	174.52
defense	166.56
hacker	107.52
computer	107.27
attack	56.96
system	49.97
software	46.65
unclassified	46.17
vulnerability	44.85
news	34.99
security	33.16
percent	32.81
intrusion	32.02
search	29.40
department	29.38
off-the-shelf	29.20
money	25.16

relevance of the retrieved documents because the keywords are a pre-selection of the most important words of the text.

2.10 Querying and Displaying

Once the crawler has been running for a few weeks, a rather important document collection is to be expected growing more and more every day. It is therefore clearly useful to provide a search interface that allows the users to search for specific documents or to browse the collection using a variety of criteria. Such criteria are, for instance:

- author
- document title
- publisher
- document language
- keywords
- subject classification (database)
- publication date
- downloading date (what's new), etc.

Similar requirements have been found in other projects using the Generic Information Server Toolkit (GIST) software¹³ [3]. Therefore it seemed to be adequate to use it for the methodology proposed. GIST provides facilities for the construction of information object repositories and managing them via the web. In order to build a GIST-based system, one first defines a data model, i.e. a description of the information objects to be managed. GIST uses the model to guide its behaviour when adding and

¹³ GIST is a tool kit for the rapid development of interactive web based information servers. GIST removes the technical barriers traditionally associated with creating interactive web sites. It has been specifically designed to allow user communities to share information and communicate more effectively without the need for a full-time technical web master.

manipulating the repository contents. Once the data model is defined, information objects can be inserted into the repository.

A data model for this methodology includes a description of “Document”-type objects based on the characteristics identified above; each document (file) is a separate object with a title, description, publisher and so on.

GIST supports the addition of information objects via HTML forms and XML files. XML files are more suitable for automated ingestion, so an XML description of each document must be generated.

The user interface for a GIST repository is produced using template files containing HTML text and embedded commands for formatting and control.

3. The OSILIA project

Necessary adaptations of the methodology (described in chapter 2) to specific applications, like for the OSILIA project, require the parameterisation of the following points:

- Type and number of sources and parameters given to the crawler
- Destinations
- Counting search words and classification thresholds.

The following three sections discuss these issues. Furthermore, we shall show specific OSILIA-related examples on database querying in section 3.4.

3.1 Sources and Parameter Files

It has been agreed, due to the short time available, to search English-written and German-written news sites for testing the methodology. In detail, these are:

- Apbnews (English): The site www.apbnews.com/newscenter/internetcrime regularly reports on internet crime related news. It is proposed to do a 2-step breadth-first search. It was not necessary to specify exclusions for the crawler (see section 2.4, page 11).
- Computernews (English): Under www.computerUser.com news out of the world of using computers are displayed. OSILIA does a relatively deep (3 steps) breadth-first search. A few exclusions have necessarily to be specified.
- Guardian (English): Being one of the major newspapers in the UK, the Guardian offers a daily updated web site under www.guardian.co.uk. This site, however, contains many links to advertisements and sites of no interest so that it was necessary to specify a huge list of exclusions. A breadth-first search is done with a depth of 2 steps.
- Newsbbc (English): The site “news.bbc.co.uk” reveals daily news that can be searched for in a breadth-first manner, however with a depth of 3. Three exclusions, in total, have been specified.
- Quicklinks(English): This American super site makes it possible to search for specific news; therefore it was plausible to give 4 starting URL’s to the parameter file each having the path www.qlinks.net/quicklinks/. The following file names have been given: “crypto.htm” (about security and encryption issues), “comcrim.htm”

(about computer crime), “content.htm” (about content regulation), “rating.htm” (about rating and filtering issues). These four URLs make it possible to have only a search depth of 1 be specified for a breadth-first search and to leave out any exclusion. Unfortunately, periodic downloads of the same articles can hardly be avoided because pages like “crypto.htm” append latest articles to previous ones¹⁴.

- Washington Post (English): The American newspaper publishes daily news updates under www.qlinks.net/quicklinks/crypto.htm. OSILIA needs a search depth of 4 inside a breadth-first search. It also needs a list of exclusions to be given to the system.
- Der Standard (German): It is a Austrian newspaper; its web site URL is “derstandard.at”. The parameter files contains a huge list of exclusions and a 2-step breadth-first search.
- Die Presse (German): Again an Austrian newspaper web site available under “194.158.136.155/textversion.taf”. It was necessary to specify 3 exclusions, a breadth-first search and a search depth of 2.
- Frankfurter Rundschau (German): This medium-sized German newspaper provides a web page under www.f-r.de/fr. In OSILIA it has been identified that some exclusions should be made; it has also been decided to do a breadth-first search with a depth of 2.
- Kurier (German): The Kurier is an Austrian newspaper that has a large circulation. The parameter file contains a small list of exclusions together with 2-step breadth-first search. The URL is “www2.kurier.at”.
- Neue Zürcher Zeitung (German): It is a Swiss newspaper widely known to an international public. The NZZ, as it is abbreviated, offers daily updated articles under www.nzz.ch/online. Some exclusions are necessary. The search is done in a breadth-first 2-step manner.
- Paperball (German): www.paperball.de is a German newspaper search engine which allows to find articles in a large variety of German-speaking newspapers. When the user enters a query string, the system displays articles of German newspapers whose contents match the query string. As this user-specific query string is automatically translated into a complex URL, we decided to use this facility by directly accessing the complex URL which Paperball produced on the basis of our first OSILIA query. The second part of the OSILIA query string became the query of the parameter file. Due to the fact that a Paperball query only shows 10 articles at a time, it was necessary to create more than one parameter file, each having a starting URL that obtains the first 10, then the second 10, and so on articles. To explain this in more detail: the starting URL
“www.paperball.de/service/paperball.fcgi?action=query&pg=detail&fmt=.&r=kriminal%2A+diebstahl+trojan%2A+missbrauch+p%E4dophil%2A&q=kriminal%2A+diebstahl+trojan%2A+missbrauch+p%E4dophil%2A&stq=51&d0=&d1=&what=german_web&rankedBy=date&categories=all&papers=all” says that the articles starting with the 51st article should be displayed, and that the query given to the Paperball search engine is “kriminal AND diebstahl AND trojan AND missbrauch AND pädophil”. Each Paperball parameter file needs a very large list of ex-

¹⁴ It is, of course, the responsibility of the page’s maintenance service to regularly “cut” those pages in order to make sure that they do not get too long.

clusions; on the other side, however, a simple breadth-first search with a depth of 1 is sufficient. Thus we get exactly those articles whose headings are displayed. We also get every time the index file (the overview page which points to the other newspapers) which, of course, is of no use.

- PC Welt (German): PC Welt is a weekly magazine reporting on any topic concerning the use of computers. The starting URL is www.pcwelt.de. It has been identified that 3 steps within a breadth-first search are optimal. A small list of exclusions was also necessary.
- Der Spiegel (German): It is a big weekly German magazine; its web site (www.spiegel.de) provides a lot of unrelated links; therefore it was necessary to set up a rather big list of exclusions. However, it was sufficient to specify 2 steps within a breadth-first search.
- Süddeutsche Zeitung (German): This is one of the major German newspaper which provides the service of also making available a text-only version, which is easier to deal with. The daily updated web site shows a summary of articles. The site that is used as starting URL is <http://www.sueddeutsche.de/aktuell/?section=showAll&myTM=text>. The search is done through a simple (depth of 1) breadth-first search. Therefore no exclusions are necessary.
- Tages Anzeiger (German): It is a Swiss newspaper that is widely known inside Switzerland. Its URL is "tagesanzeiger.ch/ta". The site is searched in a breadth-first manner and with a maximum of 2 steps. It was necessary to specify a large list of exclusions.
- Tagesspiegel (German): For the Berlin-based medium sized newspaper ("www.tagesspiegel.de") it was necessary to specify a large list of exclusions. We decided on a breadth-first search of a depth of 2.

The sites are queried with the following logical formula:

$$(a_1 \text{ OR } a_2 \text{ OR } \dots \text{ OR } a_m) \text{ AND } (b_1 \text{ OR } b_2 \text{ OR } \dots \text{ OR } b_n).$$

The idea behind this is to roughly filter web sites according to two areas of interest: computer, internet, or cyber-based issues on one hand, fraud, criminal, or abuse-based issues on the other hand. The first category of interest points at objects ("computers"), media ("internet") and their attributes ("digital"); the second category of interest points at attributes indicating the way how first-category attributes are used ("fraud", "criminal").

For the English language sites the a's (in the formula) are: 'internet', 'cyber', 'computer', 'electroni', 'digital', 'web', 'WWW'; the b's are 'theft', 'criminal', 'trojan', 'paedophil', 'abuse', 'intru', 'hack', 'spooft', 'virus', 'fraud'.

For the German language sites the a's are: 'internet', 'cyber', 'electroni', 'elektroni', 'digital'; the b's are 'diebstahl', 'kriminal', 'trojan', 'pädophil', 'missbrauch'.

In all parameter files the options were set in a way as not to allow the downloading of images and HTML headers.

3.2 Databases (destinations)

OSILIA is focussing on 'cyber crime'-related issues; therefore the following database records have been created:¹⁵

- (a) "Attacks on computer": This database contains sources that describe attacks on computer systems, networks, and infrastructures. Typical keywords for that category are: "virus" and "hacker".
- (b) "Attacks on internet": In this database sources are kept that report on attacks on the internet itself. Typical keywords are "trojan horse" and "hacking of web site".
- (c) "Computer-based fraud" (internet-related or not): Contrarily to the previous categories, this database keeps articles about fraud activities in which the use of computers is more or less necessary. Activities that will be searched for are "man-in-the-middle", "credit card stripping", or "attacks on encryption".
- (d) "Crime that happens to pass over the internet": This special category keeps sources which report on cases using the medium "internet" for criminal purposes. Typically, "paedophile" will be searched for as well as "bomb-making instructions".

3.3 Special Filter Mechanisms

As set out in the methodology part of this technical note, there are two locations with application-specific filters: (1) The program that cleans the text files (see 2.6) and (2) the program that classifies the texts (see 2.8).

For the first part, each news site that is being looked at has to read an own file that contains information according to which the file under investigation has to be "cut".

Example (quicklinks) of what a filtering process looks like:

Converted files seem to consist of lines containing patterns like "Find your next job", "Join Backflip", or "5 of 10" (indicating the page numbers). In these cases, because they are not followed by essential information, it seems plausible to cut off all those lines up to the end of the file. Lines starting with "TheStandard.com" will be cut off so that only the following text in the line will pass the filtering process. Lines that contain only patterns with "-"s are transformed into a new-line character. Finally, lines that contain "back to the top", "help" etc., are skipped completely.

For the second type of filters, according to the four types of destination (see 3.2) the OSILIA classification program contains four counters called "computer" counter, "internet" counter, "fraud" counter, and "crime" counter¹⁶.

1. First, for each news site, it has to be checked whether it makes sense to further analyse a certain file. Files, for example, that fulfil this pre-condition are files whose names start with "index" (for "Guardian"-type files), or with "ressort" (for files stemming from the domain "Der Standard").

2. Then, each file is checked to see whether any line matches a basic PERL regular expression.¹⁷

¹⁵ Recently a fifth category has been discussed focussing on "crimes that can be detected through the internet". Typically, fraud-detecting authorities try to work with the internet in order to find out fraud-related stakeholders.

¹⁶ When writing this Technical Note, the destinations have not yet been fully parameterised thus having the four counters hard-coded in *classifyE.pl* and in *classifyD.pl* (see 2.8), respectively.

Example (German language part):

The program basically checks whether at least one string (a word) from each of the following two blocks appears in the file (the '*' stands for any string; '-quoted strings do not contain blanks):

```
'computer*', 'internet*', 'web*', 'WWW', 'server*', 'cyber*', 'on-line*', 'online*',
'email*', 'e-mail*', 'net*', '*netz*', 'passwor*', 'kredit-karte*', 'kreditkarte*',
'*program*', 'firewall*', '*verschlüsselung*',
'*missbrauch*', '*mißbrauch*', '*angriff*', '*attaque*', 'hack*', '*krimin*', '*virus*',
'*viren*', 'wurm*', 'trojan*', 'pädoph*', 'pädo*', '*terror*', 'eindring*', 'kind*',
'*porno*'.
```

It is clear that, if a file should be considered further, it must completely match the PERL formula (In the example, a line must contain at least one search word out of the first part of the formula and one search word out of the second part).

The filtering part for German language texts clearly shows the complexity encountered when a file has to be checked for search words: German allows multiple concatenation of words into a single compound word so that a word's sub-parts are difficult to identify.

3. A file that passes the previous rough type of filtering will then be scrutinised for more detailed search words. The conditions that follow are not exclusive; if the search word is found, appropriate counters will be augmented according to the relevance of the search word(s) matched.

Examples (German language part):

In the case that a line contains 'FBI', 'BKA', 'LKA', 'kriminalamt' each of the counters (see section 3.2) will be increased by one point.

The "attacks on computer" counter will be increased by one if either 'virus' or 'viren' is found.

In the same manner, the "attacks on the internet" counter will be increased by 1 if the word 'privacy' or 'Privatsphäre' is found. The same counter, however, will be increased by 3 if the word 'hoax' is detected as this word is rather specific thus contributing more to reach the threshold.

The "crime ... over the internet" counter will drastically be augmented if 'pädoph' is encountered; finding such an "important" word in a file is a strong indication that the underlying text is highly relevant.

4. The thresholds have been set to 5 for each file and each news site.

3.4 Examples of Using GIST

Some parameterisations to the general GIST mechanism had to be done in order to build the OSILIA repository. Much of the software and configuration data was reused from other projects (In particular the "KM" scheme used for the "KMeC" and "eConfidence" services formed the basis for the OSILIA model and user interface).

¹⁷ Similarly to the hard-coded counters, the matching formulae have not yet been parameterised so that the classification programs (*classifyE.pl* and *classifyD.pl*) still contain OSILIA-specific lines of code.

As described in 2.10, the OSILIA data model includes a description of “Document”-type objects; each article is a separate object with a title, description, publisher and so on.

The developers used PERL and Bourne shell scripts to generate XML files from the document text and associated keywords. An extract from an XML file is shown below. The file lists a “Guardian” article and its assigned keywords; the article has been stored in the database record “attacksOnComputer/EN”:

```
<Object>
  <obj_type>DOC</obj_type>
  <title><![CDATA[The Judas kiss of the love bug ]]></title>
  <abstract><![CDATA[<br><br>The Judas kiss of the love bug
Forget rioters - cyber-terrorists are the ones to smash capitalism
May 2K: special report
FBI - NIPC Advice & Warning on the virus
Sophos virus information
<br><br>Saturday May 6, 2000
For all the media space they claimed, the anarchists who marched
through
... remaining description text not shown ...
]]></abstract>
</Object>
<Data_file>
  <filename>
    <_entdata>/home/pslh/osilia/data/attacksOnComputer/EN/guardian-
0,3604,217864,00.txt</_entdata>
    <_entmime>text/plain</_entmime>
    <_entname>article.txt</_entname>
  </filename>
  <visible>Y</visible>
  <description>Full Text of article</description>
</Data_file>
<Language>
  <type>en</type>
</Language>
<Document>
  <publisher><![CDATA[guardian]]></publisher>
  <type>2</type>
</Document>
<class.code>attacksOnComputer</class.code>
<Keyword[0]>
  <word>anarchist</word>
</Keyword[0]>
<Keyword[1]>
  <word>virus</word>
</Keyword[1]>
<Keyword[2]>
  <word>cyberspace</word>
</Keyword[2]>
<Keyword[3]>
  <word>love</word>
</Keyword[3]>

... keywords 4 to 9 not shown ...

<Keyword[10]>
  <word>cause</word>
</Keyword[10]>
```

When developing the scripts to produce XML descriptions of the articles, the following issues were considered:

- Most of the article files still contained non-article text, particularly advertisements and navigation links. The scripts used some very simple heuristics to attempt to exclude such information and derive a probable article title (in Object.title).

- In some cases the article text was extremely lengthy, and exceeded the size limits imposed by the database engine for text fields. In order to avoid problems with storage and display, descriptions exceeding 32 kilobytes in length were truncated at the nearest (probable) paragraph boundary. HTML break tags
 were added to provide rudimentary paragraph separation. The full article text was added to each object as an attachment (the <Data_file> section of the XML file) for completeness.
- Language, publisher and subject classification information were derived from the name of the file. For example, the file:

.../data/attacksOnComputer/EN/computerUser-index021.txt

indicates an English article published by "Computer User" regarding "attacks on computers".

- The keywords for each article were extracted from its .kwt file and added as a repeating attribute. The system stores each keyword independently rather than as a single field so as to provide "browse by most frequently used keyword" facilities.
- The <![CDATA[data]]> construct was employed to ensure correct handling of special characters in longer text fields. This tag causes all characters between <![CDATA[and]]> to be interpreted literally.

A simple shell script was then developed to execute the GIST *add* program for each of the XML files.

Minor changes in the user interface were required to add support for keywords in the presentation, search and browse pages. Results of using the GIST for querying the database are shown in the figures in the attachment (see chapter 6).

4. Results and Discussion

4.1 Experiences Made

4.1.1 Performance

Although the performance of the whole system largely depends on the settings in the various parameter files, it can be said that, from a performance point of view, only one part needs further improvements: the *compare* program. It has turned out to be extremely time-consuming simply due to the fact that large amounts of files have to be compared with each other.

The problem of downloading multiple copies of almost identical text files is difficult to avoid. Even if the *compare* program has eliminated files due to their multiple existence it cannot be excluded that the same file will be downloaded again in the next round. Reasons for this are the following: (1) Parts of news sites are not updated regularly. For example, the pages about science of the German *Süddeutsche Zeitung* are updated once a week while the crawler accesses the site every day. (2) Often web pages will be kept as they are while only the daily time stamp (which may be written on top of the page) is exchanged. For the crawler this signifies a new page even if the contents are completely the same. (3) We have discovered web pages that "collect" information by simply appending new contents to existing ones. Thus "old" information

will continuously be downloaded because, obviously, it seems that the web page has changed.

An appropriate parameterisation of the cleaning and filtering mechanisms (see 2.6, 2.8 and 3.3) requires frequent user intervention due to (1) highly dynamic web pages (which continuously change their layout), (2) frequently changing web site URLs, meaning that the site would simply no longer be found. For the first argument, we might be confronted with new texts, new buttons etc.; as a consequence, documents will no longer be filtered or cleaned properly.

To summarise, it can be said that the whole process needs little, but permanent user intervention in order to keep up-to-date all the user-defined data written in the various parameter files. It even turned out that the *Starting URL* parameter has to be checked from time to time.

4.1.2 Quality of Downloaded and Modified Files

Obviously, the number of files downloaded depends on how detailed the various filters are written, and on the thresholds set. Due to high numbers of files obtained we had to tune those parameters appropriately.

The quality and relevance of downloaded files mainly depends on two criteria: the questions (1) whether only a single article has been downloaded or whether the page contains more than one article, and (2) whether the search words match the article itself or whether they match non-relevant text such as the text in advertising or in links to other locations). For some web sites, the whole procedure worked very well, thus providing mainly “good” documents; other web sites, however, needed permanent supervision in order to limit the number of “bad” downloads.

As the quality of modification depends on how to “capture” the structure of files, it will be necessary to put more effort into making use of those structures, especially the HTML structure of the downloaded web pages. Although this is not an impossible task, it may dramatically expand the underlying parameter files. In addition (as was said earlier on), continuous monitoring of web sites will be necessary.

4.1.3 Evaluation of the Retrieval Results

The results of OSILIA (with the current parameterisation) have been conferred for external review. A manual evaluation of the results has shown that 95% of the downloaded web pages were relevant, meaning that the articles covered the subject internet abuse in one way or another. The remaining 5% of web pages were presumably downloaded because the search words were found in advertising, in links to other articles of the day, or other text not pertaining to the news article as such, or because the search words did not carry the meaning expected or were unrelated to one another (e.g. “An internet report on child abuse in local authority homes” would score for “internet” and “abuse” but the one has nothing to do with the other).

Some of the web pages which were downloaded were so-called index files, i.e. overview pages which only contained pointers to the news of the day plus a two-line description of their contents. As we are not interested in these overview pages, but only in the articles themselves, these index pages were not considered relevant. Some pages were general discussions, for example of criminality, in which the internet aspect was of very minor importance. After taking out these index files and general articles, and the 5% of clearly irrelevant pages, about 75% of the texts of the initial document collec-

tion were left over, meaning that 75% of the downloaded and stored documents were single articles which definitely covered the field of internet abuse.

In these 75% of the documents which were clearly articles on internet abuse, we found many duplicates, or near-duplicates, of the same article, for reasons explained earlier in this chapter. On average, there were about two duplicates for each article. While for most articles no duplicates existed at all, for some articles there were a large number of copies. These usually came from a known subset of sites which keep the articles on line for a long time, only changing the related articles of the day, the time stamp, or other text outside the core newspaper article. After eliminating these multiple copies of the same article, about 25% of the initial document collection remained. This means that the performance will be three times better when a better mechanism to avoid duplicates will be in place.

Finally, about 5% of the downloaded articles were considered to be really innovative or important, meaning that they actually contained important new information. The discrepancy between the 5% of really important articles and the 25% relevant and distinct articles has to do with the fact that some events such as the striking of the I-love-you virus were discussed a lot and in many different newspapers, without there being new information from one newspaper to the other; they were also the subject of "update" articles every day or every week, much of which regurgitated the same information. This phenomenon is to be expected for popular front-cover events such as the striking of this particular virus. This specific problem will presumably be somewhat less important when searching for lower-profile news stories.

In parallel to finding internet abuse-related articles automatically, using a software agent, the JRC's Public Relations department was asked to carry out a traditional, manual newspaper clipping service covering the same area of interest. This parallel activity allowed a reliable evaluation of the performance of the automatic procedure. According to the same evaluator, the manual clipping service showed broad congruence between the two approaches: they caught essentially the same events, and the proportion of really relevant articles which added new information, compared to the generally relevant articles, is about the same, namely about 20% (equivalent to the 5% and the 25% in the automatic procedure). This result shows that the automatic procedure can reach the same performance as a manual service, while being, of course, much cheaper and less laborious. Furthermore, there are nowadays many open sources for news which do not exist in printed form, and these are usually not covered by traditional clipping services. Where the automatic procedure clearly is currently weaker, is the downloading of multiple copies of the same article.

Apart from the technical performance of the system, it is interesting to ask what users actually get out of using such software. It is clear that there is a general need for such software, but the degree of usefulness may obviously differ from one application to the other. In the case of the OSILIA application, the insights gained during the document gathering process were rather surprising. The project had aimed to find out whether there was a substantial amount of well-validated open-source information available over the Internet on incidents of computer and internet abuse and crime; and the overall conclusion was "no": Such information was not available under the constraints given. The bulk of the information in the 5% of articles which were important or innovative was widely available from other sources, and most of what was not did not deal with specific incidents, but with vulnerabilities, defences, or general analysis. This means that, from the OSILIA *user* point of view, the successful technical development

led to the insight that a further collection of newspaper articles on internet abuse was *not* useful.

In evaluating this verdict, it should be remembered that it had been decided from the beginning that the project would look for newspaper and magazine articles, which could have a reasonable claim to authority, rather than looking in discussion groups and Usenet groups; and similarly the project had decided not to look to proprietary or private information. There is no doubt that more extensive information is available, but it is often hard to know how much confidence can be placed in it.

To summarise:

1. The results show that the method of identifying new newspaper articles on the internet automatically works very well. The OSILIA project was therefore able to provide an answer to the user's question, whether the internet could act as a valuable source of validated open-source information concerning cyber-crime and cyber-abuse. The fact that the answer to that question was negative is an indication to future users that the usability of such software may differ depending on the purpose of the information gathering.
2. The results also show that the major improvements which have to be made in order to increase the performance of the system concern two factors: firstly, duplicates have to be avoided as much as possible and, secondly, the process of extracting the core newspaper articles from the "noisy" web page has to be improved. For both problem areas, there are a number of promising solutions (see section 4.2).

4.2 Possible Improvement / Development and Further Applications

As we concluded in section 4.1.3, the main technical improvement should focus on avoiding multiple downloads of the same or extremely similar contents and on cleaning the downloaded web pages from irrelevant information such as publicity, related articles, other articles of the day, etc. Another type of improvement concerns the exclusion of so-called index files (overview pages containing mainly pointers to the full articles), for example by introducing a dynamic search depth (see section 2.4).

4.2.1 Avoiding multiple copies of the same document

Regarding the avoidance of multiple copies of the same article, the approach in the OSILIA project was to clean the downloaded web page as much as possible from textual information which does not pertain to the core article and then to compare the remaining text files with each other. Files which were completely identical, byte for byte, were deleted, but many other near-duplicates, differing often in only one or two lines, remained.

First experiments have shown that a comparison of the key word lists of articles (see section 2.9) provides a good measure of similarity of documents. Nearly identical files usually either have exactly the same keywords or they only differ by one or two keywords. It can be assumed, though, that two text files that have more than 60 or 70% of their keywords in common, are copies of each other. An overlap of 60% of the keywords should only be possible if two articles are mostly the same and only differ in a short introduction, or the like. This fact suggests that the keyword lists are a good means of identifying copies and near-copies of the same articles.

Table 2: Keywords (and their *keyness*) identified automatically for two texts that are about 70% identical. Common keywords are marked with bold face.

guideline	112.42	guideline	133.01
feds	61.50	card	62.50
penalty	57.67	crime	62.49
crime	56.45	sentence	51.78
card	46.54	theft	51.18
sentence	45.65	http	47.01
thief	42.06	zdnnet	47.01
case	35.85	zdn	47.01
loss	30.95	internet	43.51
computer	29.97	credit	42.93
theft	29.21	www	41.47
internet	28.45	thief	39.97
credit	27.47	trademark	39.19
hike	25.15	computer	35.04
		case	31.85
		identity	30.61
		copyright	29.63
		congress	29.48
		loss	28.38
		judge	24.07

Table 2 shows the keywords of two texts, written by the same author for two different newspapers, which only differ in the introduction. About 60% of the documents are identical. The fact that they have 11 keywords in common shows that keyword lists can indeed be used to identify text duplicates or near duplicates.

4.2.2 Improving the extraction of the core article from the web page

The second main area of improvement had to do with improving the extraction of the core news article from noisy web pages by deleting advertising and pointers to other articles. For this problem, the solution probably lies in the usage of the HTML document structure. For time restrictions in the first phase of the OSILIA project, we have not invested any time in making use of the HTML document structure to identify document titles, author and date information, etc., or to identify the HTML frames which contain the main information or the frames containing irrelevant information. Using the HTML document structure clearly has a lot of potential, so that this possibility will be followed up. However, the disadvantage of using the HTML structure is that the system will need more maintenance because it will be more prone to suffering from format changes made by the news providers. Each time the news providers change their format, the extraction program will need to be amended.

4.2.3 Avoiding index files

Index files, which are the overview pages that point to the individual articles, can be avoided by adding the typical names for index files to the exclusions list and is thus easily possible once the necessary effort of identifying the typical file names is put into it.

4.2.4 Making use of the potential of meta news sites

Another interesting point of further development could be the more in-depth investigation of meta news sites such as *Paperball* or *ComputerUser*. Such sites, by definition, provide pre-filtered documents so that a slightly different way of handling will be needed compared to “normal” sites. The challenge lies in how the handling of such sites can be integrated with the methodology proposed. On the other hand, when used efficiently, these sites can be very useful.

4.2.5 Applying the techniques to further domains

The methodology can easily be used for different application domains in which well-defined downloads of public sources are required on a regular basis. A first presentation¹⁸ of the results of OSILIA already ended with the offer for collaboration for another application. Application areas which are currently under discussion are: (1) gathering fraud-related news items for the intelligence department of the European Anti-Fraud Office OLAF, (2) gathering news items having to do with the proliferation of material which can be used to build ABC weapons, for JRC's nuclear safeguards activities, and (3) gathering news items in various fields for a political group at the European Parliament.

Applying the technology to one or more of these new application domains will certainly help the JRC-developed system to mature in the areas where it had not been possible to optimise the application in the course of the OSILIA project.

5. References

- [1] Steinberger, R.: *Language Engineering Technologies and their Use for TF-UCLAF*, JRC Technical Note No. I.99.83, pp. 5-9.
- [2] Steinberger, R., Hagman, J., Scheer S.: *Using Thesauri for Information Extraction and for the Visualisation of Multilingual Document Collections*. Proceedings of the Workshop on Ontologies and Lexical Knowledge Bases (OntoLex'2000). Sozopol, Bulgaria, September 2000.
- [3] Henshaw, P., Shiels, Ph.: *GIST (Generic Information Server Toolkit) Overview*, Special Publication No. I.00.137, December 2000.

¹⁸ On December 21, 2000, at the JRC.

6. Attachments

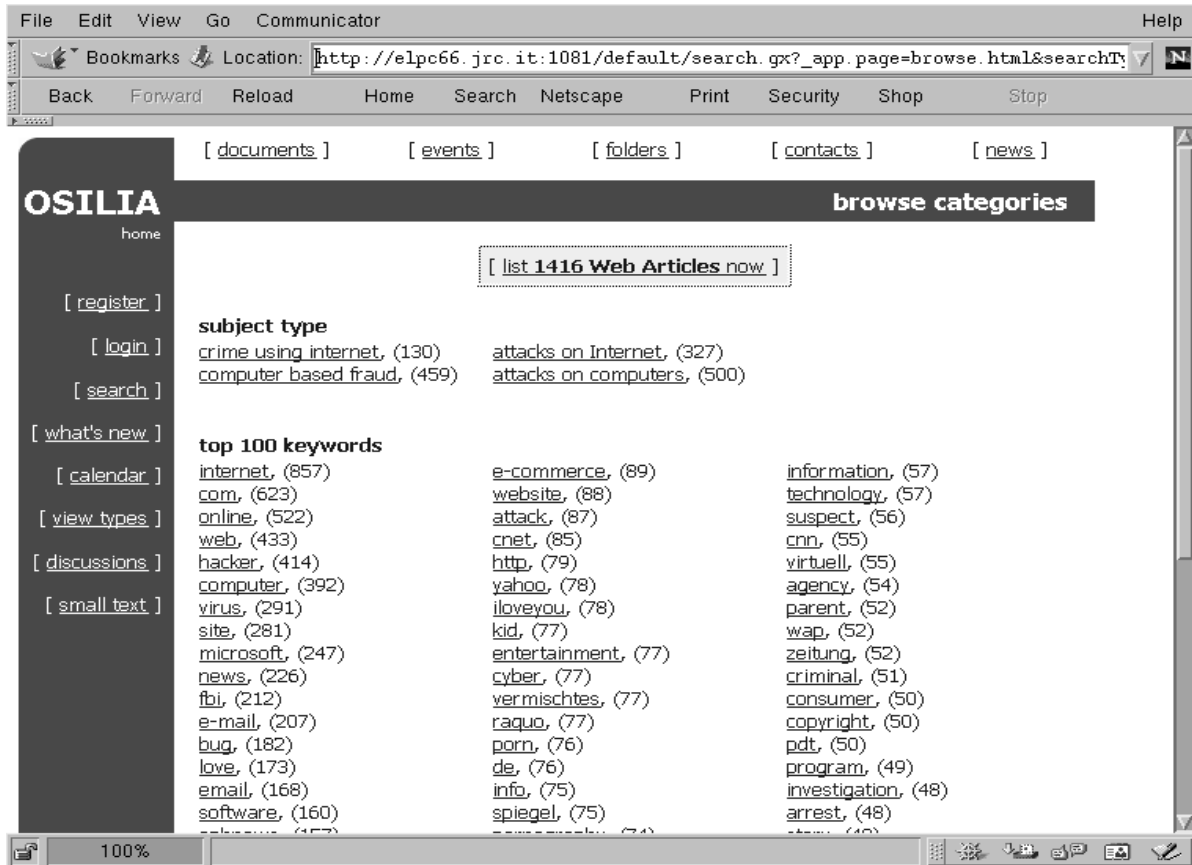


Figure 4: The document browse page summarises the frequency of the subject types and of the top 100 keywords. If the user clicks "computer-based fraud" at this point, the page is redisplayed with refined frequency data as it is shown in Figure 5.

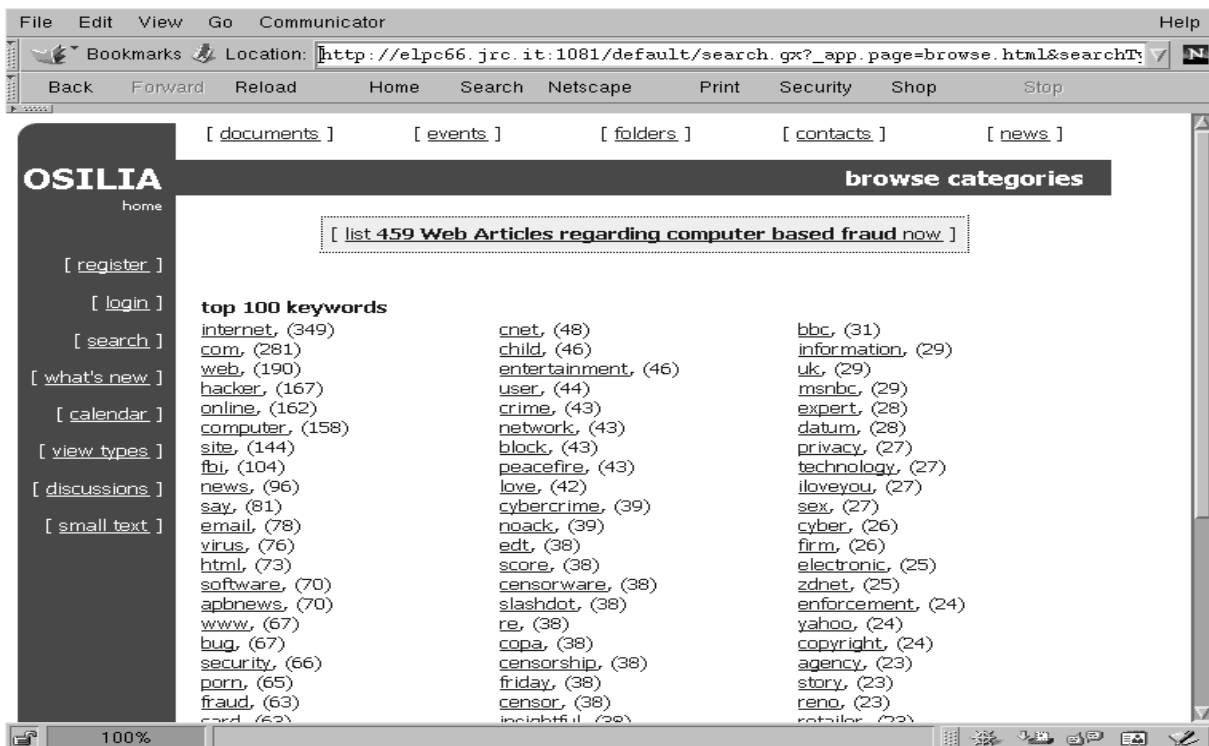


Figure 5: A more refined listing of top keywords for the documents of class "computer-based fraud".

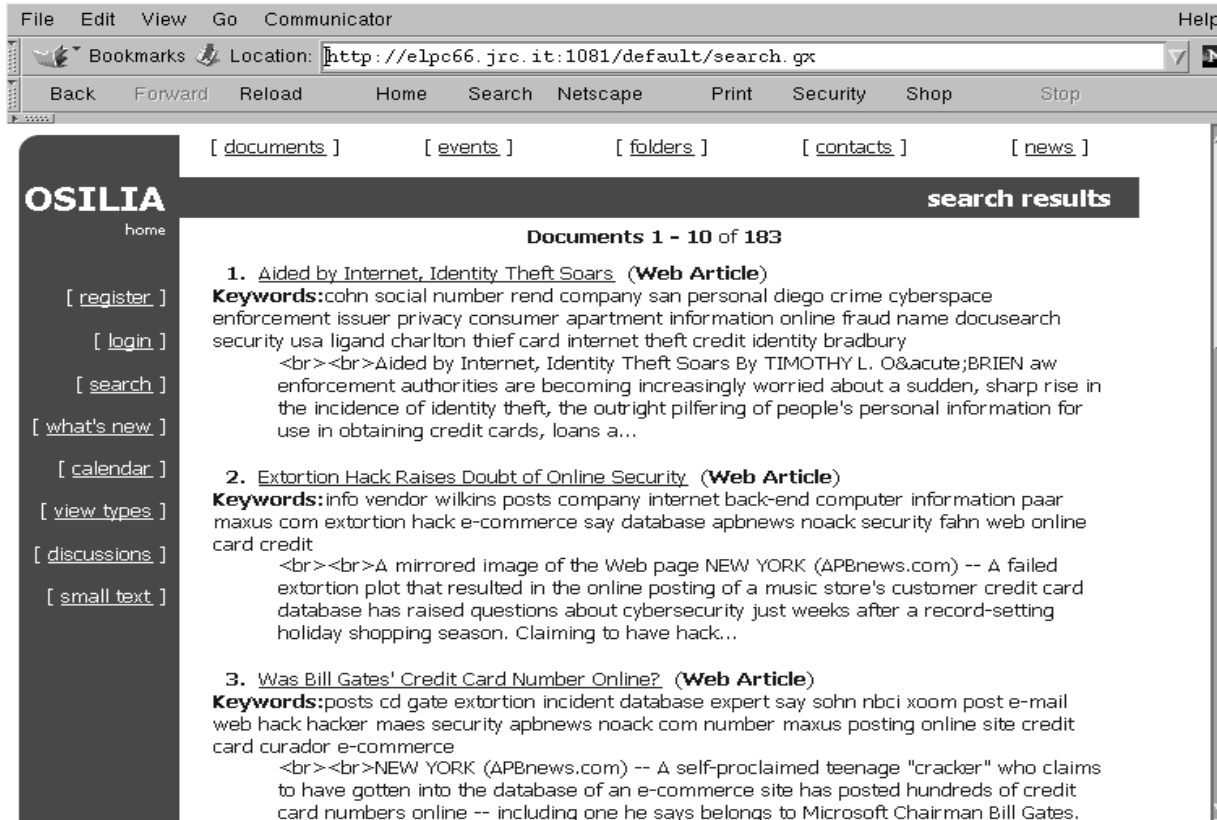


Figure 6: The user will be provided with this presentation of results when either using the search form (see Figure 8) or clicking the "list ... now" link displayed on the browse page.

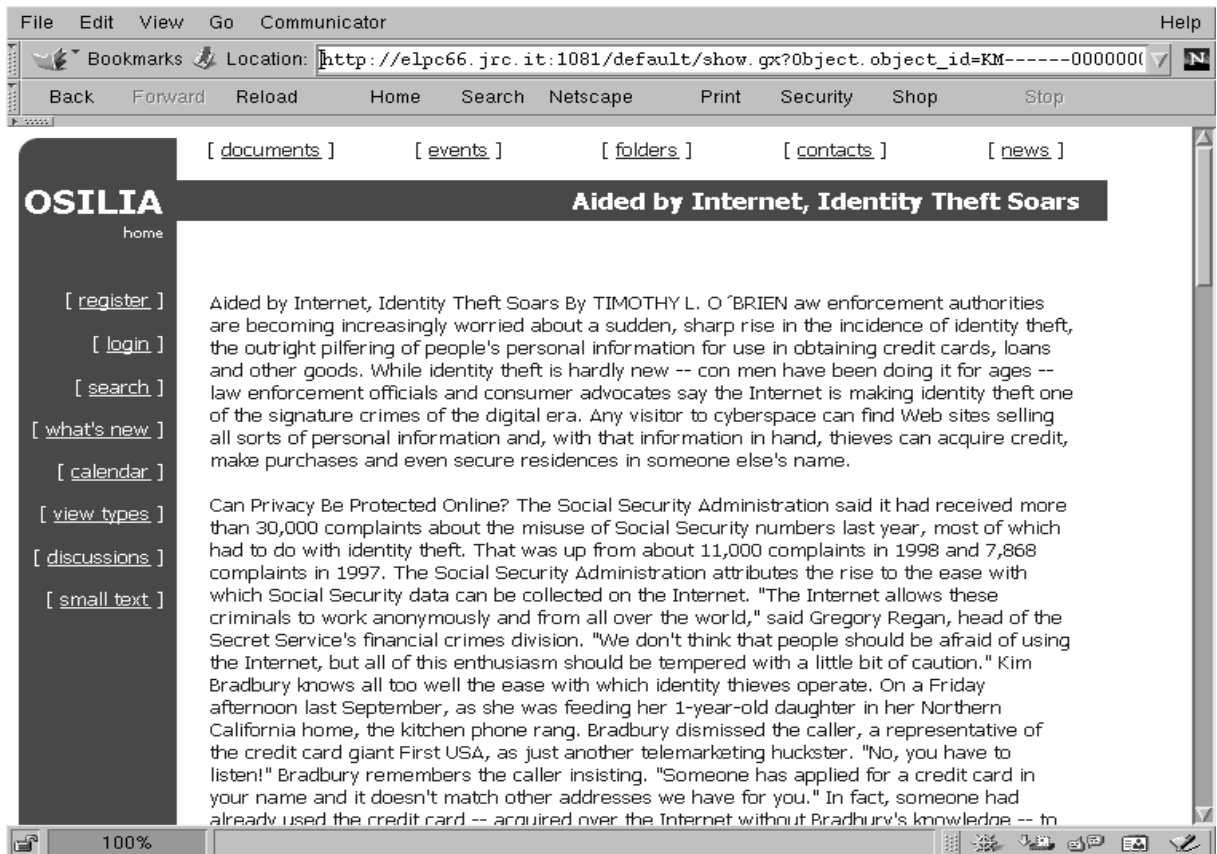


Figure 7: When the user selects an article from the search results page, the article is displayed as it is shown.

File Edit View Go Communicator Help

Bookmarks Location: http://elpc66.jrc.it:1081/default/search.gx?_app.page=search-ALL-top.html

Back Forward Reload Home Search Netscape Print Security Shop Stop

[documents] [events] [folders] [contacts] [news]

OSILIA

home

[register]
[login]
[search]
[what's new]
[calendar]
[view types]
[discussions]
[small text]

powered by **gist**

search for documents

title contains

publisher any publisher

document type Web Article (1416 items)

language English (870 items)

subject type computer based fraud (459)

keywords

free text "credit card"

[list all documents] [browse] [alphabetical search]

100%

Figure 8: In this form documents of the sub-type “web article” are searched for in the record “computer-based fraud”; documents must contain the text “credit card”.