

**SIXTH FRAMEWORK PROGRAMME:  
SCIENTIFIC SUPPORT TO POLICIES (SSP)**

**Area 3.6 "the protection of cultural heritage and associated conservation strategies"**



**SPECIFIC TARGETED RESEARCH PROJECT  
COINS**

***Deliverable D9: Numismatic Web Search Tool***

Project acronym: **COINS**  
Project full title: **Combat On-line Illegal Numismatic Sales**

Proposal/Contract no.: FP6-SSP5-044450  
Start date of project: 01/02/2007  
Date of delivery: 14/09/2008  
Status: **PUBLIC**

***Document description***

This document describes the Numismatic Web Search Tool, i.e. the tool developed within the COINS project to search the web for pages potentially containing coins for sale.

Responsible partner/author: PIN/Achille Felicetti, Maria Sifniotis

## **Executive summary**

The present report concerns the Reference Collections, created with data from the Fitzwilliam Museum and SAR. The collection comprises more than 1000 records, and is stored in a database accessible for research (via the management tool) or image recognition.

## Table of contents

Executive summary .....	2
1. Introduction: web image searching .....	4
2. Methodology .....	4
3. Search engine issues .....	5
4. Search Engine Implementation .....	8
6. Results of the COINS spider.....	10
References .....	11

## 1. Introduction: web image searching

Searching the web for images is still in its infancy. Quality of an engine's image results directly depends on the quality of the textual information associated with the images (e.g. filename, nearby text, 'alt' tags within etc). Advanced search abilities can display images according to colour properties (grayscale/colour) or size. In a few cases (Google, Exalead) the engines apply face recognition technology in order to identify likely images containing faces. Content-Based Image Retrieval (CBIR) search engines are few and mostly in research stages. Examples include WebSeek and IBM's QBIC. These engines consider the characteristic of the image itself-its shape and colours.

Without the ability to examine image content, searches must solely rely on textual metadata. Queries on a CBIR system, except from the image's characteristics, can be by example (an image is provided by the user as a search criteria), or by semantic retrieval (finding images of a specific subject).

Recent research by Google (Baluja & Jing, 2008) attempts to blend computer vision techniques with user feedback and preferences. Images are returned by their visual similarities by using local features descriptors. Results are encouraging, showing a large decrease in the number of out-of-context images returned. This also follows work done for the National Centre for Missing and Exploited Children (NCMEC). In this scenario, Google assists NCMEC in tracking child exploitation and search for patterns in images of abuse on the web (Baluja, 2008)

In the present work, we have been extending the toolset for the COINS project, introducing an image-searching plugin to an open source search engine, able to retrieve both HTML pages and candidate coin images.

## 2. Methodology

The COINS management tool is the 'glue' of interaction between the different solutions offered by COINS such as the web spider, semantic engine, image recognition tools, and the thesaurus application. It works by providing a friendly set of interfaces for the user to interact. The final version of the tool will be released as a web application and it will be written using W3C standards.

We expect the system to be used by two different user types: numismatics experts and law enforcement organizations. The numismatics experts would use it to define the coins reference collection, a set of selected coins to be used as a digital reference - an ideal coin collection for coherent sets (e.g. Roman Imperial coins) where optimal samples can be referred by humans or by machines for coin classification. This allows numismatists to build a complete coin knowledge base and thesaurus, by sharing and integrating their coin archives and their knowledge by exposing them to the semantic web.

The system makes available to law enforcement organizations a powerful tool to discover and identify stolen coins illegally sold on the internet basing on numismatists' information.

Figure 1 illustrates the design of our interconnected tools. In section A, the Coins spider has the role of retrieving and saving appropriate content on its database. The content includes the HTML file, images, and any other metadata created on-the-fly in index time. The mapping tool (section B) provides the interaction between the spider and the management tool firstly by translating the retrieved metadata in a semantically meaningful way and secondly by giving the ability to add

richer information to the content. Lastly, the Management tool (section C) among its other capabilities allows for navigational and semantic search of our data.

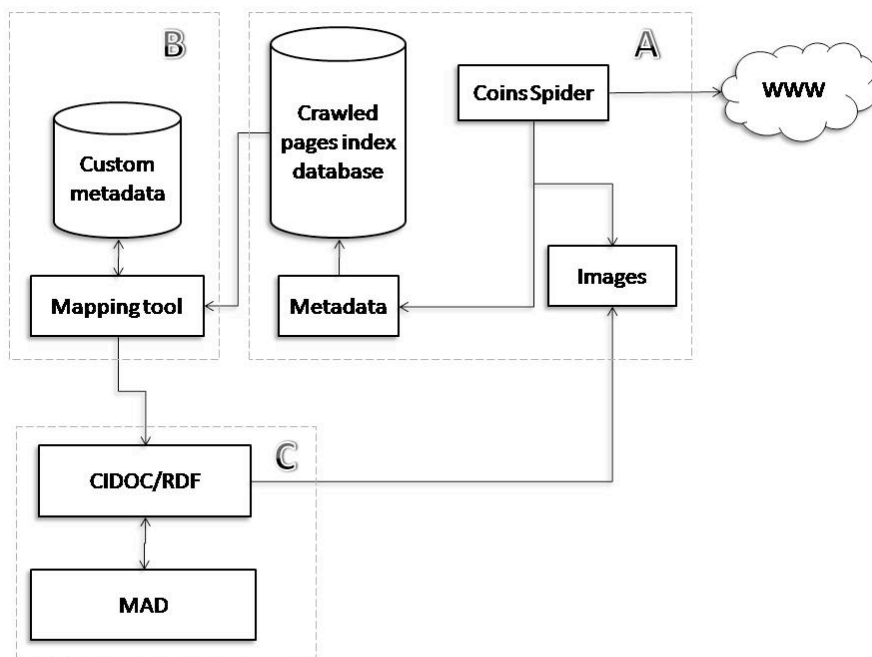


Figure 1. Diagram of tools and data flow

### 3. Search engine issues

When designing the search engine, a number of factors emerged, which influenced the design approach:

1. *Whole web crawling*: In order to compare candidate coin images with a list of known stolen ones, good candidate sites must first be found. It is a tremendous task to crawl the whole web and it may require a huge amount of hardware and man-hours. On the other hand, since traders want to be found by perspective buyers, they will use well known sites for the commerce.
2. *Open source approach*: One of the COINS project characteristics is its open source approach. This includes also the search engine.
3. *Image indexing and caching*: The search tool should be able to index and save images as well as text.
4. *Resource constraints*: effort should be placed to limit the image candidates even when indexing the web: the less ‘false candidates’ the better.
5. *Metadata retention*: any information related to a specific image should be kept as potential information to be used by the management tool
6. *Multilingual properties*: Not all pages are written in English. The search engine should take under consideration the existence of multilingual websites.
7. *Index updates*: How often should an index be updated? How feasible is it to remove or updated outdated content?

#### 3.1 Whole web crawling

It was clear from the beginning of the development that deep (extensive) web crawling in the count of millions of pages would be infeasible. Initial research was focused in identifying good candidate

web sites containing ancient coinage. After consultation with the heritage experts of the project crawling online auction websites is being tested. Consequently, the crawler begins with a set of starting URLs and fetches relevant pages back. Of course, more starting URLs can be added in the process. The right balance between a large number of hits and crawling extension may be obtained only experimentally.

### 3.2 Open source approach

In an effort to be as open about our code as possible, we have opted to use an open source search engine implementation and build our image searching capabilities on top of it. Four open search engines were evaluated for the purposes of the project; their summaries can be found in Table 1.

	<b>WebGlimpse</b>	<b>ht://Dig</b>	<b>SWISH-E</b>	<b>Nutch</b>
<b>Key Feature</b>	Suffix arrays	Simplicity	Metadata	Ranking Excerpts
<b>License</b>	Nonprofit use	GPL	GPL/LGPL	Apache
<b>Active</b>	No	No	Yes	Yes
<b>Crawling</b>	Local FS	Intranet	Intranet	All
<b>Caching</b>	No	No	No	Yes
<b>Clustering</b>	No	No	No	Yes
<b>Link Rank</b>	No	No	No	Yes

Table 1. Comparison of open source search engines

Evaluation results indicated Nutch (2008) as the best candidate for our case. It is a complete open source search engine, implemented in Java and can be fully customised by programming plugins for it. Nutch is based on Lucene, another open source project for indexing and searching documents. Research shows that Nutch achieves high performance (Cafarella & Cutting, 2004) often as good as commercial search engines (Benedict, 2004).

Nutch is roughly composed of three parts: fetcher, parser and indexer (Figure 2). The fetcher is responsible for retrieving web pages from the web as well as updating any renewed content. The parser navigates through the fetched HTML (or other types) pages and identifies tags and content available for indexing. Lastly, the indexer assembles indexable parts into the Nutch database.

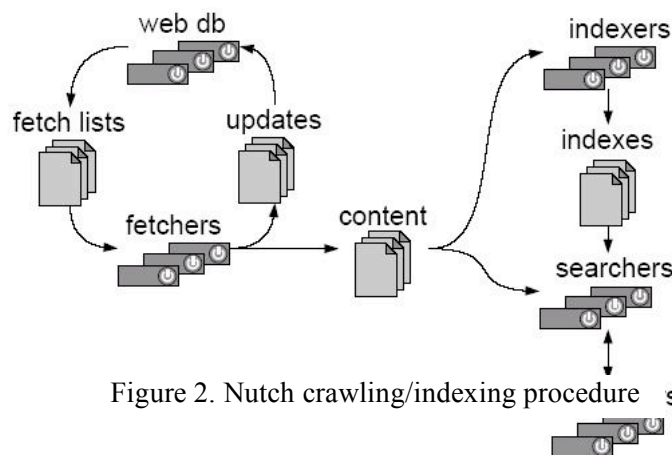


Figure 2. Nutch crawling/indexing procedure

Nutch can also scale up or down according to the user's needs. Its support for clustering uses Hadoop which implements Google's map/reduce computing paradigm and a Distributed File System (DFS).

Its use as a search engine is quite wide spread. Notable examples include Krugle (search for code, archives and technically-interesting content), Creative Commons (searches for Creative Commons licensed material) and the Internet Archive (searches National Library of Australia, the Bibliothèque Nationale de France, the Library of Congress, and the U.S. National Archives and Records Administration).

### **3.3 Image indexing and caching**

By default, Nutch is not an image indexer. It works as any text-based search engine. Work such as (Zhang, 2006) introduce an image plugin able to read JPG file formats. However it has been tested on a local filesystem scenario and not on the web.

The solution was to design an image indexing and caching extension to the parser component. The plugin inherits all the properties of the text-based Nutch parser and extends it for the inclusion of images. As a result, this also aids us in the case of web page updates.

### **3.4 Resource constraints**

An obvious solution to handling resource constraints is to limit the amount of pages the crawler is allowed to index. However, this approach may erroneously exclude positives. Additionally, it would be important to exclude automatic non-candidates such as banners and logos. The crawler is not aware of what it indexes; it does not know the content of a picture.

Taking these under consideration, we have used an approach where candidate URLs are fully indexed, but images with size less than a certain threshold (90x90 pixels - customisable) are automatically ignored and not cached. This manages to exclude not only banners and logos but also thumbnail pictures that are too small for image comparison.

### **3.5 Metadata retention**

By default, Nutch stores the whole textual content of a web page to its database. However, as the text is being retrieved, we can specify additional fields in the database that can dramatically improve the relationship with the management tool. Profiles of auction sites can be specified during the indexing and thus automatically include coin properties such as dimensions/mint type/etc.

### **3.6 Multilingual properties**

To extensively test for multilingual capabilities we are including non-English URLs of auction sites.

### **3.7 Index updates**

Updates to the index can be done as often or as infrequently wished. Different candidate URLs can be re-indexed in varying time intervals. For example, Auction site A can be queried for updates every 5 days while site B every one month. Initial experimentation has shown that a 25-day limit is good enough, since the majority of auction sites keep their ended listings data for at least one month.

## 4. Search Engine Implementation

### Case study: eBay

As mentioned in the introduction, a large proportion of illegal online trade occurs on the eBay site. As a result, the COINS spider uses eBay for testing purposes.

For this first case study we are using eBay US and specifically the ancient coins category. A study was made of four eBay sites, namely the French, UK, Italy and US ones for:

- The number of results in the ancient coins category
- The number of coin results in other categories.

Site name	Local search	Worldwide search
Italy	1686	6495
US	3383	7680
UK	935	6358
France	1388	4227

Table 2. Comparison of localised eBay sites

Table 2 shows the number of results retrieved in May 2008 for all items in the Ancient Coins category, both for a local and a worldwide search. The US site gives the maximum number of results, which justifies its choice for our primary case study.

Following this, we examined whether coins were found in other categories than the Ancient Coins one and whether these categories should be included in the crawl. Table 3 and 4 show the results of a simple keyword case; in table 4 queries were done in the local language. Results are included from Ancient coins and the other categories. It was found that a very low number of coins appeared in other categories than Ancient Coins, mostly in the Antiquities one.

Keyword:	Category:	Category:	%
Denarius	Coins	Other	
UK	144	3	2.1
US	308	4	1.3
Italy	427	3	0.7
France	17	0	0

Table 3: Comparing results across categories (A)

Keyword:	Category:	Category:	%
Roman coin	coins	other	
UK	413	70	16.9
US	1026	74	7.2
Italy	114	5	4.4
France	91	3	3.3

Table 4. Comparing results across categories (B)

It would seem that eBay users try to be as accurate as possible when assigning categories for their listings. These observations also include items that may belong in both categories and would be

thus retrieved from a crawl in just the Ancient Coins one. As a result the cost of crawling other categories for this case study was considered unnecessary.

Lastly, the COINS spider adheres to the good practices of web robots and fully respects robots.txt permissions before crawling a web page.

## **5. Overview of the COINS spider**

As mentioned above the COINS spider is based on Nutch. The crawler starts with the URL of eBay US Ancient Coins category and proceeds to fetch the links included. Aspects such as depth (how many links should be followed) and maximum number of links to be followed per page can be customised.

Experimentations with the eBay site provided a best depth of 3. This depth follows the top page, its links and the links on the fetched pages. More depth than this included a large number of erroneous entries while less did not give accurate results. The main reason behind this is the way eBay results are structured. While the specific category listing has its own domain (coins.listings.ebay.com) all referred items on eBay begin with cgi.ebay.com.

This was resolved by creating a profile set for the specific auction site. To illustrate, when the spider fetches a page from cgi.ebay.com it examines whether it belongs to the ancient coins category or not. If it does, it includes a customised tag in the database index that the specific page is a coin candidate. These profile sets can be extended to other candidate sites.

### **5.1 Image plugin**

The image plugin is an extension of the default Nutch HTML parser. Apart from the usual HTML tags it also checks for <img> ones. After the fetching cycle finishes, the indexing commences, where data from the web sites is actually indexed in the Nutch local database. The plugin works as follows:

1. Find all image tags in the HTML file
2. Check if the image URL has been examined before by looking at the hash table – if it has, ignore it.
3. For each new image URL:
  - a. Create a unique MD5 hash key composed by the URL of the image
  - b. Add the hash key to a unique hash table
  - c. Check if the image is valid and fits our dimension requirements
  - d. If it does, save the file with its MD5 hash key as its file name.
  - e. Include the MD5 hash file name as a tag inside our database index
  - f. Include a property as a coin candidate
4. Repeat for next image

This approach solves a number of problems. Firstly, with the hash table it avoids re-examining the same file over and over thus saving considerable bandwidth strain. Secondly, it automatically gets rid of small thumbnail images as well as logos and banners. Thirdly it provides a meaningful way to retrieve all image candidates at once when a user is searching the contents of the database. The plugin works with all popular image formats such as JPEG, GIF, PNG, TIFF and BMP.

## 5.2 The search tool

Once the crawling is complete, the user is able to interact with the results by accessing the actual search. The search runs as a Java servlet on an Apache Tomcat server. The user can search by keywords and results displayed include also the images related to the specific page. Another option is to view only the images that have been retrieved from the crawling and classified as possible coin candidates.

## 6. Results of the COINS spider

For testing purposes, the spider was run on a PowerMac G5 with 1GB of ram and a 2 Mbit ADSL connection. Threaded crawling requests to the eBay site are low in number, for politeness. After initial experimentations with URLs and depth of search, crawling for images commenced with a small number of top N to test the image plugin algorithm. Having verified that the plugin works in a robust way, the topN limitation was removed.

Through a period of a week, the US site was slowly crawled in order to fetch coins links and images. The initial results are encouraging. 13450 urls were fetched from eBay US. 10906 of them (81%) are actual ebay items (cgi.ebay.com links). 2544 of those links were identified as potential coins candidates – after pruning of duplicate URLs the number was reduced to almost half. The image spider also fetched 1279 prospective coin images related to the candidate coins URLs. Out of those images, 180 (14%) are not actual coins but pictures of coin books, busts of Caesar, etc (**Errore. L'origine riferimento non è stata trovata.**). Checking, parsing and saving of the images was complete in two hours.

We consider these results very encouraging since we have not utilised any image recognition techniques to reduce the number of non-coins images. Numerous of the fetched URLs were from completely different parts of the eBay site but due to the profile screening they were correctly not identified as coins candidates.

Search results for 'coincandidate:YES' in the Luke - Lucene Index Toolbox v 0.8.1 (2008-02-13).

#	Score	Doc. Id	anchor	boost	cache	coincandidate	content	digest	host	image_0_filename
0	0.0065	4		0.002887636	content content	YES		4cd8577b2f		img_564f74b
1	0.0065	6		0.0029258989	content content	YES		c33413a479		
2	0.0023	7		9.725624E-4	content content	YES		f9171ae091c		
3	0.0023	8		9.346012E-4	content content	YES		14dc13ad4e		
4	0.0023	9		9.729468E-4	content content	YES		2d407b1b23		
5	0.0078	13		0.0029837105	content content	YES		51c2ab74dal		
6	0.0023	14		9.5414155E-4	content content	YES		27aa09e7da		
7	0.0065	15		0.0029076594	content content	YES		88fe411d4be		
8	0.0023	16		9.7303727E-4	content content	YES		2714e6bc98		
9	0.0065	17		0.0029076594	content content	YES		e050b8f791		
10	0.0023	18		9.7303727E-4	content content	YES		df31c051afe		
11	0.0065	19		0.0029076594	content content	YES		a87fe52f8b7		
12	0.0023	20		9.7303727E-4	content content	YES		7b9995b3c3		
13	0.0033	44		0.0012522812	content content	YES		622be26792		
14	0.0023	97		9.211935E-4	content content	YES		7f2022fc856		
15	0.0023	98		9.691683E-4	content content	YES		c36d14708d		
16	0.0078	125		0.0031343077	content content	YES		5ce827b45cf		img_5f80c21
17	0.0078	126		0.002988445	content content	YES		c8e89958der		

Figure 5: Index analysis

## References

- Baluja, S. 2008. *Building software tools to find child victims*. Retrieved June 15, 2008, from <http://googleblog.blogspot.com/2008/04/building-software-tools-to-find-child.html>
- Baluja, S., & Jing, Y. 2008. PageRank for Product Image Search. *World Wide Web Conference 2008* (pp. 307-315). Beijing, China.: ACM.
- Benedict, L. 2004. *Comparison of Nutch and Google Search Engine Implementations on the Oregon State University Website*. Retrieved June 13, 2008, from Oregon University: [http://www.misterbot.fr/OSU\\_Queries.pdf](http://www.misterbot.fr/OSU_Queries.pdf)
- Cafarella, M., & Cutting, D. 2004. Building Nutch:Open. Source Search. *ACM Queue* , 2 (2), pp. 54-61.
- Nutch. 2008. *Nutch search engine homepage*. Retrieved June 15, 2008, from <http://lucene.apache.org/nutch>
- Zhang, Z., Rojas, C., Nasraoui, O., & Frigui, H. 2006.. SHOW AND TELL: A Seamlessly Integrated Tool For Searching with Image Content And Text. *ACM-SIGIR Open Source Information Retrieval workshop*. ACM.