The WebCAT Framework
Automatic Generation of Meta-Data for Web Resources

Bruno Martins and Mário J. Silva
Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa
1749-016 Lisboa, Portugal

E-mail: bmartins@xldb.di.fc.ul.pt, mjs@di.fc.ul.pt

Abstract

Automated methods for resource annotation are a clear necessity, as the success of the Semantic Web depends on the availability of Web resources with meta-data conforming to known standards and ontologies. This paper describes the WebCAT framework for automatically generating RDF *descriptions of Web pages. We present a general view of the system and the algorithms involved, giving an emphasis to typical issues in processing Web data.*

1 Introduction

Although the Web is the largest information resource in the world it is currently not used to its full potential, as the design of tools effectively using the available resources remains very difficult. Its current infrastructure, based on HTML and designed for human consumption, presents problems to the development of automated tools. Content and graphical presentation are highly interlinked, as most HTML markup is actually used as graphical design elements. Markup errors are also common, and software operating on Web data has to tolerate all sorts of mistakes. The adoption of XHTML, a new version of HTML re-framed in terms of XML and CSS will hopefully amend these problems, by restoring the structural integrity of documents. However, poor HTML will be around for years, as the standard is well established.

The lack of a meta-data mechanism within the Web's initial architecture is also an important limitation. Given a resource, meta-data is an information set that describes it, facilitating both discovery and description. For instance, a meta-data element identifying a document's subject can be used as a keyword to index and retrieve the document, and also as a means of classifying its content. Dublin Core is a meta-data standard proposed for Web information. It lists 15 properties such as creator or title, which together represent the minimum information necessary to "facilitate discovery of document-like objects in a networked environment such as the Internet" [43]. While the elucidation of Web meta-data architectures is nowadays ongoing research, having a single universal standard remains an elusive goal.

The Semantic Web initiative aims at interoperability of meta-data as a means to develop the Web into a distributed system for knowledge representation and computing [4]. The Resource Description Framework (RDF) [22] was introduced as both the "lingua franca" and framework for producing a Semantic Web, providing agents with machine-understandable descriptions. XML provides the standard RDF syntax. Different schemas can be combined, promoting the reuse of standard elements. Adoption of RDF has nonetheless been slow [6], and the main barrier is a classic chicken-and-egg problem: people will not annotate data unless they perceive a value for their efforts, and meta-data will not be useful until a critical mass has been achieved.

Some recently proposed Semantic Web systems address the annotation of Web pages with ontology-derived semantic tags. However, such systems focus on manual and semi-automatic tooling to improve the productivity of a human annotator, rather than on fully automated methods. Humans can provide fine-grained semantic tags but this is an arduous and error-prone task, even with machine assistance. Text mining approaches can provide an alternative to automatically bootstrap the Semantic Web, and recent research efforts point in this direction [19, 15].

This paper describes WebCAT (acronym for Web Content Analysis Tool), an extensible framework to automatically extract meta-data and generate RDF descriptions from Web documents. By extensible, we mean designed in a modular way, with the expectation that third-party tools can

Research partially supported by FCT - Fundação para a Ciência e Tecnologia, under grants POSI/SRI/47071/2002 (project GREASE) and SFRH/BD/10757/2002 (FCT scholarship).Special thanks to Daniel Gomes and Marcirio Chaves for their comments on early drafts of this paper.

be easily integrated, and that the produced results can be enhanced in subsequent processing stages. The proposed framework provides a wealth of opportunities for the classification, analysis, and tagging of Web resources, both from research and practical standpoints. Example applications include Web agents, page scrapers, and crawlers. In the rest of this paper, we explain the process on which the information is extracted and represented in RDF, detailing also the algorithms and processing stages involved.

2 The WebCAT Framework

The main focus of WebCAT is on producing RDF descriptions from the Web resources provided as input. Whilst other formats exist, RDF is a generic open standard, whereas many alternatives are either proprietary or domain specific. Whenever possible, WebCAT expresses meta-data elements through specialization relationships of appropriate Dublin Core (DC) properties (a standard RDF schema [23]). This increases the potential for interoperability, as any application capable of processing DC will be able to process most of the properties produced by WebCAT. For the elements that DC fails to express, we use other appropriate schemas, again as much as possible reusing previous proposals. Figure 1 gives an architectural overview of WebCAT:

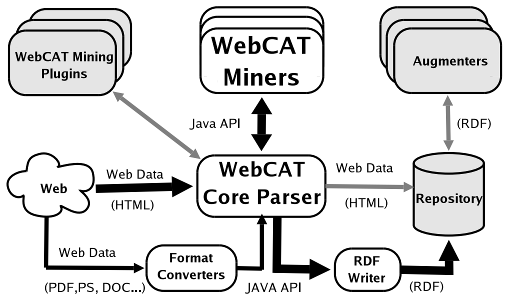

Figure 1. Overview of the WebCAT framework.

Two main blocks can be identified: the parser and the miners. The parser performs low-level processing operations, receiving Web documents in either HTML or converted from other different file formats. WebCAT analyzes the textual contents, divides the text into atomic units, and extracts the hyper-links and available meta-data. The miners use the parsed information in more advanced processing operations, generating additional meta-data properties.

WebCAT has an extensible architecture, in practice meaning two things. First, WebCAT can be extended with new functionalities. New conversion modules may used for processing other types of documents, and customized mining plug-ins may be added or replaced. Second, the ability to directly access and manipulate RDF enables fine-tuned control and flexibility. The results can be re-analyzed later, as more powerful parsers become available, and the RDF descriptions can be augmented with additional information.

Augmenting is especially useful for algorithms requiring the simultaneous analysis of a large number of Web resources, instead of just specific documents. One can process a large number of documents previously scanned with WebCAT, in some way combine the available meta-data elements, and augment individual resources with the results. Besides processing RDF descriptions, highly demanding applications can use also the provided API directly.

3 The WebCAT Core Parser

Before advanced mining algorithms can be applied to Web data, a series of low-level processing operations are required. These are essentially related to scanning HTML in order to extract information from the documents.

First of all, information on the Web appears in a wide variety of formats and not only flavors of HTML [33]. As the availability of PDF files and Flash movies increases, it is important to take these formats into consideration. WebCAT uses wrappers over standard UNIX utilities (i.e. pdftotext) to convert documents into HTML. This allows a unified approach for the extraction of meta-data from different file formats. Addition of new formats should be straightforward, providing that suitable converters are available. However, a problem with these tools concerns their frequent failure in extracting the text, producing "garbage" as output instead of terminating with an error. More often yet, they extract only the text instead of producing HTML output. Wrappers can nonetheless be easily developed, using heuristics to avoid missconversions and give appropriate markup to text fragments (i.e. the first text line is also the title) [3].

An important aspect in processing HTML documents is the capability to deal with fuzzy, noisy, and irregular input, since real world HTML is usually broken from a specification point of view [36]. Problems typically found in HTML files include missing tags and tags whose nesting order is wrong, tags with no closing slash, tags missing quotes on attribute values, multiple ways to reference the same meta-data properties, or even binary files identified by Web servers as of the text/html MIME content type. WebCAT is similar to actual HTML browsers, in particular never throwing syntax errors and taking a best effort approach to solve markup problems. For example, overlapping tags are correctly restarted by the WebCAT core parser, and HTML like:

```
This is <B>bold, <I>bold italic, </b>
italic, </i>normal text.
```

is correctly interpreted as:

```
This is <b>bold, <i>bold italic, </i></b>
<i>italic, </i>normal text.
```

In the development of WebCAT, we started with a lexical analyzer generator similar to LEX [25]. However, we found it easier to write a fault tolerant parser from scratch, that scans the documents sequentially and keeps track of the markup elements found. This was also found advantageous over using available tools to parse HTML as a DOM tree, which were slow, harder to use and error prone.

3.1 Extracting Textual Information

The isolation of textual units (tokenization) from an input character stream is a common problem in text processing. It constitutes a very important aspect in the development of WebCAT, as deciding what constitutes a word from potentially large volumes of text is a non-trivial task [18].

The simplest solution is defining a word as a series of non-whitespace characters, followed by a series of whitespaces. However, this is not always true, as some non-whitespace characters (i.e. hyphens, dashes or the line-break) can be used to delimit words. Worse, the behavior of some characters is context-dependent. In the case of the hyphen character, a break is allowed when it is followed by letters, but not when followed by numbers (where it functions as the minus sign). The sentence bellow illustrates some tokenization problems, showing for instance how the characters ".", "," and "$" can be ambiguous.

```
|Mr.| |Andrew| |Smith||,| |your| |account|
|balance| |is| |$1,234.56|.|
```

Our tokenization procedure is based on the rules proposed by Linguateca to handle all corpora from the project AC/DC [38]. We also took a simple data-driven approach to the ambiguity problems addressed above, basing the behavior of the tokenizer on the "context pairs" formed by the character(s) before and after a given position in the text. For example, a word-tokenization table puts breaks between punctuation and letters, and between letters and whitespace, but not between consecutive letters or whitespaces.

This technique actually works quite well and is also easy to implement. The table is a two-dimensional array of Boolean values, with each row and column representing a character, group of characters, character category or a change in HTML markup. One axis represents the context before the possible break, and the other axis represents context after it. Interpreting HTML markup and tokenization are tightly coupled, as tags are useful in detecting text boundaries. For instance, when separating sentences, both punctuation and HTML tags like <P>, <BR>, <DIV>, <CENTER>, <TABLE> and <TR> can be used as boundary marks.

We use the context-pairs approach for recognizing sentences and individual words. Rule based systems for identifying text boundaries have been described in the past. Grefenstette, for instance, reports 95% accuracy using a high speed regular expression method [18]. Mikheev reported error rates of 0.28 and 0.45% on the Brown and WSJ corpus, using rules like "when a period is preceded by an abbreviation and is followed by a lowercase word, proper name, comma or number, assign it as a sentence internal" [31]. This and similar rules were easily implemented with our context-pairs approach, which also achieved around 95% accuracy over the WSJ corpus.

Besides words and sentences, we additionally extract *n*-grams (sequences of *n* consecutive characters) of length 1 to 5 and collocations (sequences of *n* consecutive words forming a "logical" text block [42]). Occurrence statistics for all these features are also computed, and we keep track of the associated HTML markup information (i.e. occurrence in the title, bold typeface, italic typeface, headings). A motivation for extracting all these features is given by the promising results obtained in many text mining algorithms, when compared to only using frequently occurring words [9, 7]. For instance, character *n*-grams and term weighting heuristics based on HTML markup are successfully used by a WebCAT miner that guesses the text's language, described in a subsequent section of this paper.

3.2 Extracting Hyperlink Information

The Web is not just a text corpus, as it also contains hyperlinks between the pages. Linkage constitutes a rich human-oriented structure of information, which can be explored through appropriate mining algorithms [10]. Besides the textual contents, WebCAT extracts links, the corresponding anchor texts, image links, and the corresponding alternative textual descriptions. These operations are all relatively simple when compared to text tokenization, although they also raise interesting issues. An important aspect is converting links to a canonical form, as the same resource can be referenced through different equivalent URLs. The following operations are used:

1. Discard URLs not following the specification syntax

2. Convert URL host names lowercase, e.g. `http://WWW.TEST.PT` is converted to `http://www.test.pt`

3. Discard port number information if the value is the default for the protocol, e.g. `http://www.test.pt:80/` is converted to `http://www.test.pt`

4. Canonize file information containing relative paths, e.g. `http://www.test.pt/d1/..//` is converted to `http://www.test.pt/`

3.3 Extracting Meta-Data Information

Many Web pages already include meta-data information through the use of the HTML META tag and embedded Dublin Core. It is also nowadays possible to include a reference, or even embed RDF descriptions into the pages themselves. WebCAT considers the available RDF information, attaching it to the automatically generated descriptions. As for META tags, it currently handles standards like GeoTags [13], GeoURL [39], Dublin Core [43], the Robots Exclusion Protocol [24], and the HTTP-EQUIV properties.

Previous works suggest that meta-data usage is quite common in Web documents, but they also indicate a number of caveats attached to this conclusion [34]. Much of current meta-data usage can be attributed to the automatic generation of META tags by editing tools, and therefore they do not always correspond to accurate and comprehensive document descriptions. Moreover, meta-data information is not always provided in a standard machine readable form. For instance, different formats are used to express Dublin Core elements, and the same property can be expressed through different names and formats (i.e. Author or DC.Author). In handling META tags, WebCAT relies on heuristics and string processing operations, accounting for the most common cases and normalizing meta-data elements.

4 WebCAT Miners and Augmenters

In WebCAT, task specific modules are used to complement the information extracted by the core parser with additional meta-data. We call these modules miners, as they use combinations of learning algorithms and text analytics techniques to infer extra knowledge from the available data.

Examples of miners already integrated within WebCAT include a content fingerprinting algorithm [5], an algorithm for detecting nepotistic hyperlinks (advertisements and other links that are present for reasons other than merit) [14], word stemmers [35] and other text analysis modules (i.e. named entity recognition) and document classification algorithms. The rest of this section presents three of these miners, detailing the particular algorithms used and their integration with the rest of the framework.

4.1 Language Classification

Language is important meta-data, as having documents with appropriate language annotations can be used to bootstrap more advanced processing modules (i.e. stemming, named entity recognition and/or topic labeling). A small percentage of documents already contain language meta-data but special care should be taken when using this information. Web authoring tools can leave the language meta-data tag set to the default value (usually "English"). Authors are also free to use whatever form they see fit to provide information in the META tag, not always standard or machine readable (e.g. "Portuguese", "Português", "pt" or "por").

Automatic language guessing systems have been described in the past, achieving very high accuracy. However, text from the Web is considerably different [1] (i.e more spelling errors, multilingual documents, small amounts of text), motivating the development of better approaches.

In WebCAT, we have a specific miner for automated language guessing, adding language labels to whole documents or text fragments [29]. The algorithm is similar to the proposal by Cavnar and Trenkle [9], already reported to be very precise and tolerant to errors. Essentially, the method is based on the statistical characterization of the texts in terms of the most frequent *n*-grams of characters that constitute them, comparing the document to classify with probabilistic models for the different languages. For instance the bi-gram "ão" occurs frequently on Portuguese documents, whereas it never occurs in English ones. We complemented the original idea with the more efficient similarity measure proposed by Lin [26], together with HTML markup heuristics to better handle Web data. In a controlled study, the algorithm presented a precision of about 91% in discriminating among 11 different languages [29], although problems were detected in recognizing similar variants of the same language (e.g. distinguishing Portuguese documents from Brazilian ones).

4.2 Named Entity Recognition

A cornerstone to the deployment of a semantic Web is the automatic generation of named entity (NE) annotations, with class and instance references to a semantic repository (i.e. an ontology). For example, the sentence:

```
Mary, currently a journalist in Lisbon, studied
with Jack during the nineties at Berkeley.
```

contains the following named entities:

```
[PERSON Mary], currently a journalist in
[LOCATION Lisbon], studied with [PERSON Jack]
during the nineties at [ORGANIZATION Berkeley].
```

Typical NE recognition systems consist of at least a tokenizer, NE datasets (gazetteers or ontologies) and NE extraction rules. The rules, which can be generated by hand (knowledge-based) or automatically (machine-learning), are the core of the system, combining the named entity datasets with elements such as capitalization and the surrounding text. The degree to which the datasets help in identifying named entities seems to vary. For instance Malouf found that gazetteers did not improve performance [27], whereas others have gained significant improvements using gazetteers and trigger phrases [8]. Mikheev et al. showed a

NE recognizer could perform well even without datasets for most classes, although this was not the case for geographical entities [32]. The same study also showed that simple matching of the input texts to previously generated lists performs reasonably well in this last problem.

The miner integrated in WebCAT uses information from a geographical ontology [11] as the main dataset, associating the found place references to the corresponding entries at the ontology. Instead of a machine learning technique requiring a large amount of hand-annotated training text, we use a simpler approach based on hand-coded rules. The considered entities are, for now, locations and organizations with a known geographical scope, together with numerical values, time and date expressions. We are also limiting the analysis to documents in Portuguese, Spanish, German and English (using the language guessing miner), although the approach could be extended to other languages. The collocations extracted by the core parser are matched against the feature names occurring in ontology, and they are marked with appropriate semantic tags. Regular expression rules are also used to identify named entities, combining the entries at the ontology with additional textual clues. Previous studies have shown that geographical references in text very often contain information about the types of places they are labeling, such as "city of A" or "district B" [21]. These rules should therefore provide a good means of classifying the found entities according to type. We also have more complex extraction rules, similar to the co-occurrence patterns proposed by Marti Hearst to generate taxonomies from text [20]. Examples of such patterns include expressions like "A, B, C and other cities" or "cities such as A, B, or C", where A, B and C are all recognized as names of cities. A similar approach has already been tested as the basis of a Web-scale extraction system [16]. Technically, our implementation uses a minimized deterministic finite state automaton. All rules are compiled into such an automaton, and since it is deterministic and minimized, the matching algorithm is extremely fast. Besides recognizing NEs, we also classify and assign them to entries at the ontology. Disambiguation is done through simple heuristics (i.e. references have one sense per discourse and highly populated areas are more often referenced) [28].

State-of-the-art machine learning systems for recognizing named entities in newswire text achieve an F-score over 90%, although these approaches require a considerable amount of manual effort in creating a balanced and representative training set [37]. Our task is also harder, since we disambiguate NEs and associate them with entries at an ontology. Simple as it is, our approach for geographical NE recognition performs remarkably well, with initial studies indicating a precision/recall of about 0.89 and 0.68 [28]. Again, text from the Web is considerably different from newswire text, making our task harder.

4.3 Assigning Geographical Scopes

Finding automatic ways of attaching geographical scopes to Web resources is a challenging problem that is getting increasing attention [2]. It constitutes a very difficult classification task, leaving open challenges to current machine learning approaches. For instance, the amount of training data per parameter is so low that there are no repeatable phenomena to base probabilistic methods on, leading to the failure of typical methods.

A specific augmenter relies on the geographical named entities recognized for the documents and available in the `RDF` descriptions, together with a disambiguation algorithm that also builds on the geographical ontology [28]. Instead of the standard methodology of automatically inferring classifiers from a training set of documents, our approach uses carefully selected semantic features (the named entities), combining and disambiguating the available information through the ontological relationships that exist among the features (i.e. sub-region-of or adjacent-to). Also instead of just using the document's text, geographical references from linking documents are taken into consideration. Web resources are first processed by WebCAT in order to extract geographical references, which are then propagated through linkage to the neighboring documents. Finally, the individual `RDF` descriptions are augmented with the inferred geographical scope.

The disambiguation process sees the ontology of geographical features as a graph and takes its inspiration on the PageRank ranking algorithm [30]. The features and the ontological relationships between them can be seen as the nodes/vertexes of a graph, and the document occurrence frequency associated with each feature can be used as "importance" weights. A slightly modified version of PageRank is applied to this graph, computing a score for each feature at the geographical ontology. Finally, the highest scoring feature is selected as the scope for the document, and a corresponding semantic tag is added to the set of available meta-data elements. Initial evaluation studies indicated a very good potential for this method [28], showing comparable results to the Web-a-Where geo-tagging system [2].

5 WebCAT Application Scenario

WebCAT was incrementally developed over the past two years. It is currently one of the main components of a Portuguese search engine [41], providing a consistent view of Web data to the other software modules. Implemented in Java, it is available on-line at `webcat.sourceforge.net`.

The search engine crawler uses WebCAT as a hyper-link extractor. Documents are stored in a central repository for further processing, together with the extracted full-text and the provided RDF representation. Language meta-data is used to discard foreign documents, and other information is used to build specific indexes (i.e. an inverted index for the words, using anchor text and HTML markup information). A geographically-aware retrieval interface is currently being developed, making extensive use of meta-data mined through WebCAT. Finally, WebCAT was also used as one of the software modules providing data for a characterization study on the Portuguese Web [17].

Our experiences confirm the advantages and usefulness of using WebCAT, when compared to other existing tools and methods for extracting information from Web resources (i.e. SAX or DOM-like parsers, task-specific LEX/YACC parsers or regular expression methods). Table 1 gives some statistics from the crawl used for the characterization study of the Portuguese Web. The numbers provide anecdotal evidence on the volumes of data currently processed. Larger crawls have been recently performed, and WebCAT has also been used on the GOV and EuroGOV collections, as the search engine participated in TREC and WebCLEF [12, 40].

Collective Statistics	Value
Documents Analyzed	3235140
Downloaded Data	78 GB
Textual Data	8.8 GB
Links Seen (Excluding Internal Links)	243930
Web Sites Recognized	131864
Average Document Size	32.4 KB
Words Seen	1652645998
Distinct Words Seen	7880609
Statistics per Document	**Value**
Avg. Words per Document	438
Avg. Document Size	32.4 KB
Average Text Size	2.8 KB
Avg. Word Length	5 chars
Meta-Data Statistics	**Value**
% .PDF Docs	1.9%
% .DOC, .XLS, and .PPT Docs	0.7%
% Docs with description meta-tag	17%
% Docs with keywords meta-tag	18%
% Portuguese Docs	73%
% English Docs	17%
% Content Replicas	15.5%

Table 1. Statistics from the usage of WebCAT.

Our application of the WebCAT framework constitutes a large scale effort of automatically collecting and generating meta-data for Web resources, in an attempt to bootstrap a Web scale semantic network. Improvements are still required in some of the mining algorithms integrated within the framework, and a thorough evaluation of their individual effectiveness is the main concern of our current activity.

SemTag/Seeker is the largest similar effort to date [15], reporting an experiment with 264 million Web pages (270Gb of dump data, 434 million semantic tags with accuracy around 82%). A cluster of 64 machines was used, processing about 200 docs/second (32 hours for processing the 264 million pages). The data on Table 1 was crawled with 9 machines, processing about 2 million docs/day (including fetching documents from the Web). Results for both systems are hardly comparable, and although ours appears to be slower, the experiences confirm the feasibility of using WebCAT for processing large collections.

6 Conclusions

While it is difficult to automate resource description, it would be impossible to describe everything on the Web manually. Automatic meta-data generation appears like an essential requisite for a widespread deployment of the Semantic Web. This paper presented WebCAT, a content analysis tool capable of automatically producing meta-data in `RDF` format, especially useful for Web based problem solvers and search agents. WebCAT provides a form of closed-loop learning, in which low-level information is extracted from the documents, a shallow level of understanding is provided through mining algorithms, and structured information becomes available for further learning. The paper provided some examples of the processing operations involved, namely discussing the important issues of how to deal with fuzzy and irregular input, and how to use mining algorithms to annotate Web pages. The modular architecture encourages the addition of new functionalities and there are many ideas for future enhancements.

References

[1] E. Amitay. Trends, fashions, patterns, norms, conventions... and hypertext too. *Journal of the American Society for Information Science and Technology*, 52(1):36–43, 2001.

[2] E. Amitay, N. Har'El, R. Sivan, and A. Soffer. Web-a-Where: Geotagging Web content. In *Proceedings of SIGIR-04, the 27th international conference on research and development in information retrieval*, 2004.

[3] K. Andersen. txt2html - Text to HTML converter. (Available online at `http://txt2html.sourceforge.net/`).

[4] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.

[5] A. Z. Broder. Some applications of Rabin's fingerprinting method. In *Sequences II: Methods in Communications, Security, and Computer Science*. Springer Verlag, 1993.

[6] M. H. Butler. Barriers to real world adoption of semantic Web technologies. Technical Report HPL-2002-333, HP Laboratories, 2002.

[7] M. F. Caropreso, S. Matwin, and F. Sebastiani. Statistical phrases in automated text categorization. Technical Report IEI-B4-07-2000, Centre National de la Recherche Scientifique, 2000.

[8] X. Carreras, L. Marques, and L. Padro. Named Entity Extraction using AdaBoost. In *Proceedings of CoNLL-2002, the 6th Conference on Natural Language Learning*, 2002.

[9] W. B. Cavnar and J. M. Trenkle. N-gram-based text categorization. In *Proceedings of SDAIR-94, the 3rd Symposium on Document Analysis and Information Retrieval*, 1994.

[10] S. Chakrabarti, B. E. Dom, S. R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, D. Gibson, and J. Kleinberg. Mining the Web's link structure. *Computer*, 32(8):60–67, 1999.

[11] M. Chaves, M. Silva, and B. Martins. A geographic knowledge base for text processing. In *Proceedings of SBBD-05, the 20th Brazilian Symposium on Databases*, 2005.

[12] N. Craswell and D. Hawking. Overview of the TREC-2002 Web track. In *Proceedings of TREC-2002, the 11th Text Retrieval Conference*, 2002.

[13] A. Daviel. Geo tags for HTML resource discovery, July 2003. (Available online at `http://geotags.com/geo/`).

[14] B. D. Davison. Recognizing nepotistic links on the Web. In *Proceedings of the AAAI-2000 workshop on Artificial Intelligence for Web Search*, 2000.

[15] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J. A. Tomlin, and J. Y. Zien. SemTag and Seeker: Bootstrapping the semantic Web via automated semantic annotation. In *Proceedings WWW-03, the 12th international conference on the World Wide Web*, 2003.

[16] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Web-scale information extraction in KnowItAll. In *Proceedings of WWW-04, the 13th international conference on World Wide Web*, 2004.

[17] D. Gomes and M. J. Silva. Characterizing a national community Web. *ACM Transactions on Internet Technology*, 5(2), May 2005.

[18] G. Grefenstette and P. Tapanainen. What is a word, what is a sentence? Problems of tokenization. In *Proceedings of COMPLEX-94, the 3rd International Conference on Computational Lexicography*, 1994.

[19] D. Gruhl, L. Chavet, D. Gibson, J. Meyer, P. Pattanayak, A. Tomkins, and J. Zien. How to build a webfountain: An architecture for very large-scale text analytics. *IBM Systems Journal - Utility Computing*, 43(1), 2004.

[20] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings ACL-92, the 14th International Conference on Computational Linguistics*, 1992.

[21] L. L. Hill, J. Frew, and Q. Zheng. Geographic names - The implementation of a gazetteer in a georeferenced digital library. *D-Lib Magazine*, 5(1), January 1999.

[22] G. Klyne and J. J. Carroll. Resource Description Framework (RDF): Concepts and abstract syntax, February 2004. W3C Recommendation.

[23] S. Kokkelink and R. Schwnzl. Expressing qualified Dublin Core in RDF/XML, 2002. (DCMI Proposed Recommendation - `http://dublincore.org/documents/2002/04/14/dcq-rdf-xml`).

[24] M. Koster. A method for web robots control, December 1996. (Internet Draft).

[25] M. E. Lesk and E. Schmidt. LEX: A lexical analyzer generator, 1979.

[26] D. Lin. An information-theoretic definition of similarity. In *Proceedings of ICML-98, the 15th International Conference on Machine Learning*, 1998.

[27] R. Malouf. Markov models for language-independent named entity recognition. In *Proceedings of CoNLL-2002, the 6th Conference on Natural Language Learning*, 2002.

[28] B. Martins and M. Silva. Categorizing web pages according to geographical scopes, 2005. (Submitted).

[29] B. Martins and M. Silva. Language identification in Web pages. In *Proceedings of ACM-SAC-DE-05, the Document Engineering Track of the 20th ACM Symposium on Applied Computing*, 2005.

[30] R. Mihalcea and P. Tarau. TextRank: Bringing order into texts. In *Proceedings of EMNLP-04, the 2004 Conference on Empirical Methods in Natural Language Processing*, 2004.

[31] A. Mikheev. Document centered approach to text normalization. In *Proceedings of SIGIR-00, the 23rd international conference on Research and development in information retrieval*, 2000.

[32] A. Mikheev, M. Moens, and C. Grover. Named entity recognition without gazetteers. In *Proceedings of EACL-99, the 9th Conference of the European Chapter of the Association for Computational Linguistics*, 1999.

[33] E. T. O'Neill, B. F. Lavoie, and R. Bennett. Trends in the evolution of the public web : 1998 - 2002. *D-Lib Magazine*, 9(4), April 2003.

[34] E. T. O'Neill, B. F. Lavoie, and P. D. McClain. Web characterization project : An analysis of metadata usage on the Web. *Journal of Library Administration*, 34(3/4), 2000.

[35] M. F. Porter. Snowball: A language for stemming algorithms, 2001.

[36] D. Raggett. Clean up your web pages with hp's HTML TIDY. In *Proceedings of WWW-98, the 7th International World Wide Web Conference*, 1998.

[37] T. K. Sang, E. F., and F. De Meulder. Introduction to the CoNLL-2003 shared task: Language-Independent Named Entity Recognition. In *Proceedings of CoNLL-2003, the 7th Conference on Natural Language Learning*, 2003.

[38] D. Santos and L. Sarmento. O projecto AC/DC: acesso a corpora / disponibilizacao de corpora. In A. Mendes and T. Freitas, editors, *Actas do XVIII Encontro da Associacao Portuguesa de Linguistica*, pages 705–717, October 2002.

[39] J. Schachter. GeoURL ICBM Address Server, 2002. (Available online at `http://www.geourl.org`).

[40] B. Sigurbjornsson, J. Kamps, and M. de Rijke. Blueprint of a cross-lingual web retrieval collection. In *Proceedings of the 5th Dutch-Belgian Information Retrieval Workshop*, 2005.

[41] M. J. Silva. The case for a Portuguese Web search engine. In *Proceedings of ICWI-2003, the 2003 IADIS International Conference on the WWW/Internet*, 2003.

[42] F. Smadja. Retrieving collocations from text: Xtract. *Computational Linguistics*, 19(1), 1993.

[43] S. Weibel. The state of the Dublin Core metadata initiative. *D-Lib Magazine*, 5(4), 1999.