Domain-Specific Web Site Identification: The CROSSMARC Focused Web Crawler

Konstantinos Stamatakis, Vangelis Karkaletsis, Georgios Paliouras
Institute of Informatics and Telecommunications, NCSR "Demokritos", GR-15310, Athens, Greece
{kstam, vangelis, paliourg}@iit.demokritos.gr

James Horlock, Claire Grover, James R. Curran, Shipra Dingare
School of Informatics, University of Edinburgh,
horlock@cstr.ed.ac.uk, jamesc@cogsci.ed.ac.uk,
{grover, sdingar1}@inf.ed.ac.uk

Abstract

This paper presents techniques for identifying domain specific web sites that have been implemented as part of the EC-funded R&D project, CROSSMARC. The project aims to develop technology for extracting interesting information from domain-specific web pages. It is therefore important for CROSSMARC to identify web sites in which interesting domain specific pages reside (focused web crawling). This is the role of the CROSSMARC web crawler.

1. Introduction

CROSSMARC is an EC-funded R&D project that aims to develop technology for information extraction from web pages in various languages, employing language technology methods as well as machine learning methods in order to facilitate technology porting to new domains. CROSSMARC also employs localisation methodologies and user modelling techniques in order to present the extraction results according to the user's personal preferences and constraints. The system is implemented with a multi-agent architecture in order to ensure a clear separation of responsibilities and to provide the system with clear interfaces and robust and intelligent information processing capabilities. The CROSSMARC architecture is composed of the following main processing stages:

- Collection of domain-specific web pages, involving two sub-stages: (a) *focused crawling* to identify web sites that are of relevance to the particular domain (e.g. retailers of electronic products), (b) *domain-specific spidering* of the retrieved web sites in order to identify web pages of interest (e.g. laptop product descriptions).
- Information Extraction from the domain-specific web pages. This involves Named Entity Recognition and Fact Extraction ([4], [5]).
- Data storage to store the extracted information (from descriptions in any of the project's languages) into a common database.
- Data presentation to present the extracted information to the end-user through a multilingual user interface, according to the user's language and preferences.

In this paper we present the CROSSMARC approach to focused web crawling. In Section 2 we outline related work and discuss the differences and similarities between web crawling and web site spidering. In Sections 3 and 4 we present the CROSSMARC implementation and the results of its evaluation in the domain of laptop offers from e-retailer sites.

2. Background

2.1. Related Work

The term 'focused crawling' was introduced by Chakrabarti et al. ([2]). The system described there starts with a seed set of representative pages and a topic hierarchy and tries to find more instances of interesting topics in the hierarchy by following the links in the seed pages. Pages are classified into topics, using a probabilistic text classifier.

Aggarwal et al. ([1]) present a significant improvement to the focused crawling approach, which they call intelligent crawling. In contrast to the focused crawling method, this uses a combination of evidence to rank the candidate hyperlinks by their level of interest and learns the relevant weight of these factors as it crawls. On the assumption that the initial set of starting points can lead to all interesting pages, very central sites should be used as starting points for the crawl (e.g. Yahoo!, etc.).

Another interesting approach to focused crawling is adopted by the InfoSpiders system (Menczer and Belew [6]), a multi-agent focused crawler. The process is initialized by a set of keywords and a set of root pages. Each agent starts with a root page and performs focused crawling by evaluating the link value and following the most promising links. Link value is assessed using a reinforcement learning method, using contextual words as input. Reward values are calculated online, by the reward that the agent receives when following a link. The user can provide relevance feedback to assist the learning process.

A different approach is proposed by Diligenti et al. ([3]). Their method of context focused crawling looks for the parents of a set of representative seed documents, up to a certain level and then builds text classifiers for each level. Then, starting from a central web node, it can predict how far it is from an interesting page, as in the Rennie and McCallum ([7]) method for spidering.

2.2. Focused crawling vs. site-specific spidering

In focused crawling, the aim is to adapt the behaviour of the search engine to the requirements of a user. The requirements of the user are expressed in one of three different ways: a query consisting of a set of keywords, a document that is representative of what the user is interested in, or a set of documents related to the user's interests. In the first two cases, the system either produces a query to a standard search engine, in order to construct a base set of documents that are potentially relevant to the topic, or starts the search from a set of user-specified central points on the Web, e.g. Yahoo!.

On the other hand, in site-specific spidering, the spider navigates inside a web site, following best-scored-first links. Each web page visited is evaluated, in order to decide whether it is really relevant to the topic, and its hyperlinks are scored in order to decide whether they are likely to lead to useful pages. Thus, a score-sorted queue of hyperlinks is constructed, which guides the retrieval of new pages. The process stops according to user-specified resource constraints.

Apart from their similarities, the two tasks also have important differences, which stem from the fact that focused crawling is not restricted to the narrow boundaries of a web site, but should potentially be able to reach the whole world. While the site-specific spider starts from the top-page of a site following links and evaluating pages within the boundaries of the site, there are choices to be made concerning the starting point of the focused crawler. A central point on the Web such as a search engine hierarchy could be considered as a potential starting point, but it is likely that the crawler would wander endlessly before getting to the useful part of the hierarchy by chance. An alternative would be to identify a promising branch of the hierarchy to start from. Another possibility is to start from the documents returned by a query in a common search engine. A final approach would be to start with a set of documents provided by the user. For instance, one could ask Google to find pages that are similar to an initial dataset of domain-specific web pages.

3. The CROSSMARC Web Crawler

The CROSSMARC implementation comprises three distinct types of crawler, each being a realisation of the options outlined in the previous section:

- Version 1 exploits the topic-based web site hierarchies used by various search engines to return web sites within specific subparts of these hierarchies.
- Version 2 uses a given set of queries, exploiting the CROSSMARC domain ontologies and lexicons, submits them to a search engine, and returns the sites that contain the pages to retrieve.
- Version 3 takes a set of 'seed' pages and conducts a 'similar pages' search from advanced search engines such as Google. It then returns the sites that contain the pages.

For each version, the starting point is determined by the user and reasonable starting points are those likely to return a high proportion of relevant sites. For example, in Version 1, a reasonable starting point for the CROSSMARC domain of laptop offers might be the *Computers_and_Internet/Hardware/Notebook_Computers* branch of the Yahoo! hierarchy. In Version 2 a possible starting point might be a general query such as *notebook sale* and in Version 3 it would be a set of web pages containing laptop offers. Clearly, each type of crawler can be adapted to different search engines or web site hierarchies. However, because each web hierarchy or search engine provides different functionality and represents their hierarchical structure or query results in a different way, each one requires a slightly different implementation. In Version 1, the hierarchies that we have used are *directory.google.com, www.forthnet.gr, www.in.gr, dmoz.org, dir.yahoo.com, uk.dir.yahoo.com, it.dir.yahoo.com, fr.dir.yahoo.com, dir.lycos.com,* and *www.lycos.co.uk.* In Version 2, the search engines exploited are Google and Altavista, and, in Version 3, only Google. The front-end crawler script combines the results of the various crawler versions using the various search engines and hierarchies and returns all web sites found.

It must be noted that the crawling process also takes into account the languages involved in CROSSMARC and can be customized to new languages. Some of the starting points are language specific, such as the set of keywords in the queries and the list of web directories.

The list of web sites output from the crawler is filtered using a light version of the site-specific spidering tool implemented in CROSSMARC. The full version of the CROSSMARC spider (NEAC) has three components:

(a) *Site navigation*. This component traverses a web site, collecting information from each page visited, and forwarding part of the information collected to the "Page-Filtering" module and another part to the "Link-Scoring" module.

(b) *Page-filtering*. This component is responsible for deciding whether a page is an interesting one (e.g. contains laptop offers) and should be stored or not.

(c) *Link-scoring*. This component validates the links to be followed, in order to accelerate site navigation (only links scoring above a certain threshold are followed).

The light version of NEAC navigates the site until it finds an interesting web page. If it finds one, it considers the site as *fit* and stops navigating. If no such page is found the site is characterized as *unfit*. At the end of the process, only the fit web sites survive.

4. Experimental results

4.1. Crawler evaluation

It is important that the focused crawler should return as many interesting sites as possible. This initial set of sites may later be "reduced" by the spidering process. For reasons of efficiency, however, it is also important that it does not return too high a proportion of uninteresting sites since this would require the site-specific spidering component to perform a great deal of unnecessary processing. A balance between these competing requirements can be obtained by finding the optimal start points for each version of the crawler as well as the optimal combination of versions. In order to find these optimal settings, we performed an evaluation where we sought to measure the effectiveness of each version given different starting points and to discover how to maximize the overall effectiveness of the crawler by combining the various versions and starting points.

The measures for evaluation were the standard measures of recall, precision, and f-score. The recall of the crawler is the ratio of fit sites retrieved by the crawler to all fit sites on the Web (where a fit site is one which contains at lease one fit, i.e. relevant, page). However, it is not possible to obtain a count of all fit sites on the Web so recall cannot be directly measured. The precision of the crawler is the ratio of fit sites returned by the crawler to all sites returned by the crawler. While it is in principle possible to measure precision by manually inspecting all the sites returned by the crawler, this is impractical given the large number of sites returned. Finally, f-score is a measure combining the measures of recall and precision defined as *2*Recall* Precision/(Recall+Precision).*

Although we cannot obtain exact figures for precision and recall, it is possible to *estimate* these measures. For this we needed to estimate how many of the sites returned by the crawler were fit and the number of fit sites on the Web. For the former we performed a manual inspection of a subset of the crawler output (150 pages). For the latter we had to make the assumption that all fit sites would be found within the combined output of all versions and start points of the crawler. The set of web sites returned by these steps is quite large and, assuming that it contains all the fit sites on the Web (as well as a large number of unfit sites), we estimated the total number of fit sites by manual examination of a random subset of 150 sites. Our main experiments were limited to English language sites in the laptop offer domain, and by the method outlined above we obtained an estimate of 993 such sites on the Web. This estimate is extremely conservative since the assumption that all fit sites are contained in the combined output of all versions of the crawler is clearly not a sound one. However, this appeared to be the only method available for obtaining any kind of estimate of the number of fit sites in the entire Web.

Table 1. Focused Crawler Evaluation

	Precision	*Recall*	*f-measure*
V1 general	32.0%	29.0%	30.4%
V1 narrow	52.3%	7.1%	12.5%
V2 general	21.3%	11.7%	15.1%
V2 man/mod	32.6%	49.5%	39.4%
V2 screen	41.0%	37.4%	39.1%
V3	38.0%	6.1%	10.5%

We performed a number of experiments, the results of which appear in Table 1. The first two rows show results for Version 1 with two sets of start points in the search engine hierarchies, general points relating to computer hardware retailers and more narrow points relating to notebooks and laptops. As the results indicate, the narrow hierarchy points adversely affect recall, while the general hierarchy points lead to a better balance between precision and recall. The next three rows show evaluation results for Version 2 with three kinds of queries as start points. The first set of queries are general ones using combinations of plural and singular versions of the keywords 'notebook', 'laptop' and 'sale'. The second set of queries consists of the combination of a number of manufacturer/model name pairs such as 'compaq presario' taken from the laptop domain ontology. The third set of queries are variations on a number of screen type specifications particular to laptops as opposed to desktops, e.g. '12.1" TFT'. As the results show, the more precise queries are more effective than the more general queries. The final row shows evaluation results for Version 3 where the 'seed pages' are a corpus of fit pages gathered as training and testing material for the Information Extraction component of the system. The results show very low recall.

The results of the experiments have helped us to determine the optimal settings for running the crawler. For English and the laptop domain we combine Version 1 with the more general start point and Version 2 with the two more specific sets of queries and we output the union of their results. On the test material used for the experiments this yields recall of 92.1%, precision of 45.2% and an f-measure of 60.64%. It may be possible to improve on this by using a more complex method of combining versions.

4.2. Spider evaluation

As described above, the site navigation module, while traversing a web site, collects information from each page visited and forwards part of the collected information to the page-filtering module and another part to the link-scoring module. Therefore, an efficient identification of interesting pages presupposes a well-tuned page-filtering module. The module is based on the use of machine learning methods for text classification. A variety of machine learning methods have been evaluated (Naïve Bayes, Nearest-Neighbour, J48, SMO, AdaBoost and LogitBoost). In order to obtain an unbiased estimate of the performance of the various learning methods, stratified ten-fold cross-validation was used to obtain the results. According to this methodology, the training set is split into ten equal-sized pieces maintaining the original distribution of the classes. Then, ten different training-test runs are performed, each of which uses one of the ten pieces for testing and the remaining nine for training. Average results over the ten runs are reported.

The page-filtering module was evaluated separately for each of the four languages that are used in CROSSMARC. The evaluation data for each language consisted of a number of web pages describing laptop product offerings and web pages that are "near-misses", i.e. pages that could be confused with the target pages.

Table 2 shows evaluation results of the algorithms evaluated on page-filtering for the English language (the results were similar for the other CROSSMARC languages).

Table 2. Page classification results for the English dataset

	Precision	*Recall*	*f-measure*
Naive Bayes	88.1%	89.3%	88.7%
Near.Neighbour	97.9%	96.0%	**96.9**%
Dec. Trees	99.0%	92.4%	95.6%
SVM (SMO)	96.4%	95.7%	96.1%
AdaBoost	97.1%	96.0%	96.5%
LogitBoost	97.7%	93.7%	95.7%

The most general conclusion of the experiments is that most learning methods are doing remarkably well (above the 90% mark) for all four datasets. The simplistic Naive Bayes classifier is an expected exception to this rule. The most surprising result is the high performance of the simple Nearest-Neighbour algorithm, with k=1, i.e., looking only at the class of the closest pre-classified neighbour. The Nearest-Neighbour classifier outperforms all other learning methods. Nevertheless, its difference from the SVM and the AdaBoost algorithms is insignificantly small in all experiments. Therefore, despite the appealing simplicity and the good results of the Nearest-Neighbour classifier, the best choice seems to be the SVM method, which is cheaper computationally than AdaBoost and faster in run time than a Nearest-Neighbour classifier.

5. Conclusion

The motivation for focused crawling comes from the poor performance of general-purpose search engines, which depend on the results of generic web crawlers. Moreover, the focused crawler output, a domain oriented list of web sites, still contains enough undesired entries to necessitate a closer look: light site spidering assures the domain relevance of a site. This is the approach implemented in CROSSMARC.

Note also that the described implementation of the focused crawling as a parasite to common web services, such as Google and Yahoo! reduces significantly the cost of its development and deployment.

Customization to new domains as well as to new languages is a crucial issue for CROSSMARC. The tools and methodologies developed facilitate such customization tasks and form a part of a platform for cross-lingual information management.

References

[1] C.C. Aggarwal, F. Al-Garawi, P.S. Yu, "Intelligent Crawling on the World Wide Web with Arbitrary Predicates", *Proceedings of the Tenth International World Wide Web Conference*, Hong Kong, May 2001, pp. 96-105.

[2] S. Chakrabarti, M.H. van den Berg, B.E. Dom. "Focused Crawling: a new approach to topic-specific web resource discovery", *Proceedings of the Eighth International World Wide Web Conference*, Toronto, Canada, 1999.

[3] M. Diligenti, F. Coetzee, S. Lawrence, C.L. Giles, M. Gori. "Focused Crawling using Context Graphs", *Proceedings of the 26th International Conference on Very Large Databases, VLDB 2000*, Cairo, Egypt, 2000, pp. 527–534.

[4] C. Grover, S. McDonald, V. Karkaletsis, D. Farmakiotou, G. Samaritakis, G. Petasis, M.T. Pazienza, M. Vindigni, F. Vichot and F. Wolinski. "Multilingual XML-Based Named Entity Recognition", *Proceedings of the International Conference on Language Resources and Evaluation (LREC-2002)*, Las Palmas, Spain, 2002, pp. 1060-1067.

[5] B. Hachey, C. Grover, V. Karkaletsis, A. Valarakos, M.T. Pazienza, M. Vindigni, E. Cartier and J. Coch. "Use of Ontologies for Cross-lingual Information Management in the Web", to appear in *Proceedings of the International Workshop on Ontologies and Information Extraction*, Bucarest, Romania, 2003.

[6] F. Menczer and R.K. Belew. "Adaptive Retrieval Agents: Internalizing Local Context and Scaling up to the Web". *Machine Learning, 39(2/3)*, 2000, pp. 203-242.

[7] J. Rennie and A. McCallum. "Using Reinforcement Learning to Spider the Web Efficiently", *Proceedings of the Sixteenth International Conference on Machine Learning (ICML-99)*, 1999, pp. 335-343.