A Tool for Web Links Prototyping

Inma Hernández, Hassan A. Sleiman, David Ruiz, Rafael Corchuelo
University of Seville
Seville, Spain
Email: {inmahernandez, hassansleiman, druiz, corchu}@us.es

***Abstract*—Crawlers for Virtual Integration processes must be efficient, given that VI process is online, which means that while the system is looking for the required information, the user is waiting for a response. Therefore, downloading a minimum number of irrelevant pages is mandatory in order to improve the crawler efficiency. Most crawlers need to download a page in order the determine its relevance, which results in a high number of irrelevant pages downloaded. We propose a tool that builds a set of prototype links for a given site, where each prototype represents links leading to pages containing a certain concept. These prototypes can then be used to classify pages before downloading them, just by analysing their URL. Therefore, they are the support for crawlers to navigate through sites downloading a minimum number of irrelevant pages while reducing bandwidth, making them suitable for VI systems.**

***Index Terms*—Web Crawling, Web Page Classification, Virtual Integration,**

I. Introduction

Virtual Integration aims at accessing web information in an automated manner, starting with a query, in which users express their interests, and obtaining information relevant to that query from the web. Automated access to the web requires a crawler, which is a tool able to navigate through web sites automatically, looking for relevant information. Traditional crawlers visit every link on every page, download their target, and check whether the page contains relevant information. This means that, even when a page is irrelevant, the crawler has to download it to realise it, which results in a large number of irrelevant pages downloaded.

Note that the VI process is online, which means that while the system is looking for the required information, the user is waiting for a response. Therefore, downloading a minimum number of irrelevant pages is mandatory in order to improve the crawler efficiency, which is a concern for several researchers [14].

We propose a classifier that helps crawlers to efficiently navigate through web sites. This classifier is able to determine if a web page is relevant or not by analysing exclusively its URL.

There are some crawling techniques that improve traditional crawlers efficiency by endowing the crawler with classification skills. For example, focused crawlers [1], [10], [13], [22], [24], [25] aim at finding pages belonging to one or more topics exclusively, so they are supported by a content based classifier that determines whether each page belongs to the topics, discarding the rest. Other crawlers include classifiers based on other features, like page structure [19], [20], [27]. Our classification proposal is different, since it is based on features that are not in the page to be classified, but in pages that link to it. Therefore, it is not necessary to download a page in order to classify it, which avoids downloading irrelevant pages, reducing the bandwidth and making it efficient and suitable for VI systems.

Other crawling techniques rely completely on the user to define the navigation patterns [2], [5], [8], [23], [29]. Instead, our proposal is automated and requires a minimum intervention from the user. Furthermore, our classifier is trained using an unlabelled training set of URLs, thus relieving the user from the tedious task of assigning a label to each page in the training set.

Our experience focuses on web sites that follow a certain navigational pattern, which is the most common pattern in the Web [19]. This pattern starts with a form page that allows issuing queries; then, after users submit a query, the system returns a hub, that is, a page containing an indexed list of answers to it, each of them containing just a brief description and a link to a page with more details. Note that the term "hub" is based on the hub and authority concepts introduced by Kleinberg [18].

Hubs in this kind of web sites are often defined by populating some patterns or scripts with data stored in a database [7]. This has two main implications: first, all hubs from the same web site usually share a common template with some fixed parts. Usually, this common parts are structured in the form of headers, footers and side bars containing navigational aids, copyright information and advertising [30], which frame the page areas that contain the information that varies from hub to hub. The second implication is that hubs and detail pages are generated on demand to respond to a user query. Similarly, URLs that point to each hub and detail page in the site are generated on demand as well by the same proceeding of filling a URL pattern with keywords or numbers that identify the generated page. Therefore, all URLs from a certain site can be expressed by a collection of URL patterns.

Furthermore, our hypothesis is that there is usually a correspondence between URL patterns and the concept contained in the pages with URLs following that pattern, so that we can classify web pages containing different concepts by means of the pattern matching their URL. Therefore, our classification technique consists on finding the different URL patterns that compose links in a given web site, that is, to build a set of prototypes for all URLs in the site. Then, we use these prototypes to classify links by template matching, that is, by

assigning each link to the class associated to its matching prototype. Furthermore, our technique is able as well to detect links belonging to the web site template, so that the crawler can process them adequately.

As a motivating example, in Figure 1 we show a hub page obtained after issuing a query on Amazon.com. If the user is interested in obtaining detailed information about products, for example, following only links marked in red means avoiding to download more than 50% of the hub linked pages, with the proportional reduction of used bandwidth. We notice that all links leading to pages containing information about the concept 'Product' have a similar pattern, which is shown in the Figure as well.

The rest of the article is structured as follows. Section II introduces the core definitions that will be used throughout the paper; Section III introduces the features that support our proposed classification tool; Section IV describes the tool design; Section V presents the related work in the web page classification area; finally, Section VI lists some of the conclusions drawn from the research and concludes the article.

II. Core Definitions

Next, we introduce some necessary concepts that will be used throughout the rest of the paper.

Hubset We define a hubset H as a set of hubs obtained from a particular site.

$$H = \{H_1, H_2, H_3, ..., H_n\}, n \geq 1 \tag{1}$$

We define a hub H_i as the set of links l_j it contains.

$$H_i = \{l_1, l_2, l_3, ..., l_m\}, m \geq 1 \tag{2}$$

Linkset For each hubset H, we define its linkset L as the set of links contained in H, that is, the set of links from every H_i in H.

$$L = \bigcup_{i=1}^{n} H_i \in H, n \geq 1 \tag{3}$$

Link We define a link l as a tuple that represents a URL, and is composed of

$$l = (S, A, P, N, V) \tag{4}$$

Where S is the schema of the URL, A is its authority or domain name, P is a sequence of path segments, N is a sequence of names of the parameters in the URL query string and V is the sequence of the former parameters values.

Links are obtained from URLs by means of a tokeniser, according to RFC 3986. Figure 2 show an schema of link tokenisation, using URL http://scholar.google.com/scholar?q=Web+crawling as an example. Note that RFC 3986 states that links also include an optional fragment preceded by a '#' symbol, which points to a specifical section inside a page. However, our goal is to build URL prototypes of URL that link to pages about a particular concept, and not specifical sections inside pages, therefore the fragment part of an URL is not useful for our purposes. Throughout this paper we will omit the fragment, although including it in the process will not affect the results.

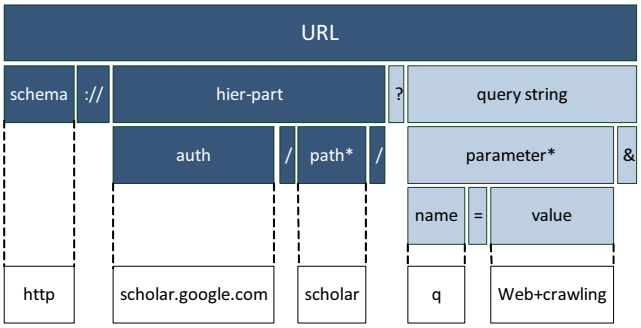


Fig. 2: RFC 3986 simplified URL schema with as an example of tokenisation

Prototype We define a prototype p as a link

$$p = (S, A, P, N, V) \tag{5}$$

where each element in P, N and V is either a literal or a wildcard, $\star$. In this case, a wildcard represents any sequence of characters (excluding separators '?', '/', '#', '=' and '&').

Common Path Links Let L be a linkset from a given site, $l = (S, A, P, N, V)$ be a link in L and $P(i)$ be the i-th path element of l, $P(i) \in P$. We define the set CPL as the set of all links l' in L having the same prefix as l up to (and excluding) $P(i)$, being $l' = \{S', A', P', N', V'\}$.

$$CPL_P(l, i) = \{l' \in L \cdot (S, A) = (S', A') \wedge P(j) = P'(j), j \in [1, i)\} \tag{6}$$

Similarly, we define CPL for the parameter names and values in l as the set of links l' in L having the same prefix as l ip to (and excluding) $N(i)$ or $V(i)$.

$$CPL_N(l, i) = \{l' \in L \cdot (S, A, P) = (S', A', P') \wedge N(j) = N'(j) \wedge V(k) = V'(k), j \in [1, i), k \in [1, i-1)\} \tag{7}$$

$$CPL_V(l, i) = \{l' \in L \cdot (S, A, P) = (S', A', P') \wedge N(j) = N'(j) \wedge V(k) = V'(k), j, k \in [1, i)\} \tag{8}$$

Recall that a prototype is nothing more than a link that includes some wildcard sections, so we can calculate the CPL (p,i) set of a prototype likewise.

Example As an example, consider p defined in Figure 3a. Prototype p represents all links starting with http://www.amazon.com/, followed by any combination of characters except separators, then the literal string /dp?ie=UTF-8&qid=, and another combination of characters at the end. Represented links are depicted in Figure 3c. If we check links in Amazon, we find out that links following this pattern are links to product detail pages.

Fig. 1: Example of Link Classification: Amazon.com hub page.

Let L be a linkset composed of links in Amazon, as shown in Figure 3b, and l_1 be the first link in Figure 3b. The $CPL_P(l_1, 1)$ set is shown in Figure 3d.

III. Classification Features

Next, we introduce the features that support building the set of prototypes that represent all links in a given site. We take a statistical approach to the problem of prototype building, and we base our technique in the definition of probabilistic features for each link and each token inside a link. First we give a formal definition of these features and later we illustrate their use by means of an example.

A. Features Definition

Link feature Let H be a hubset from a certain web site with size n, and l be a link $l \in H$. Probability F_L of a link in the context of H is defined as follows.

$$F_L(l) = \frac{|\{H_i \in H \cdot l \in H_i, i \in [1, n]\}|}{n} \tag{9}$$

In Equation 9, we must assure that the hubset is sufficiently large so that the probability estimation is statistically significant, so we require that $|H| \geq 30$, which is the usual threshold in statistical literature. F_L (l) takes values in the range $[1/n, 1]$. Links that appear more frequently in hubs from a hubset, have a higher F_L than those appearing just in a few of them, to the point that links with $F_L = 1$ appear in every single $H_i \in H$. At the other end of the distribution, links with F_L near to 0 never appear in any of H hubs.

As an example, Figure 4a shows the histogram of F_L values obtained from the 100 hubs in an e-commerce site (Amazon.com) an two academic sites (Microsoft Academic Search and TDG Scholar).

Tokens Features Let H be a hubset from a given site with size n, L its linkset, l a link of the form (S, A, P, N, V) in L and $X(i)$ be the i-th element of X, we define the feature value of $X(i)$ given as the following probability.

$$F_X(l, i) = \frac{|\{H_j \in H \cdot H_j \cap CPL_X(l, i+1) \neq \emptyset, j \in [1, n]\}|}{n} \tag{10}$$

These features values are in the same range as F_L, $[1/n, 1]$. Same as with F_L, path segments that appear more frequently in hubs from H have a higher F_P than those that only appears in URLs from some of the hubs.

Figure 4b shows the histogram of F_P, F_N and F_V values from the same hubsets and sites as defined for F_L values. It is noticeably similar to the F_L histogram presented earlier, with the majority of values around $1/n$, and just a small tail near 1.

Given that a prototype has the same signature as a link, both previous definitions III-A and III-A are applicable as well to prototypes. For the sake of simplicity, we assume that $F_P(p, i) = 1 \iff p = \{S, A, P, N, V\} \wedge P(i) = \star$ (similarly, with F_N and F_V).

B. Features Examples

Example Consider an experiment over Amazon.com, in which we issue 100 queries using the top 100 words in English language, discarding stop words. The result of this experiment is a hubset H composed of $n = 100$ hubs.

The F_L values calculated for some of the links in H are shown in Table I

All Amazon pages contain a navigation bar in the upper part of the page, including useful links such as “Home” “Sign In”

http	www.amazon.com	★	dp	ie	qid	UTF-8	★

a) Prototype p

LINK	S	A	P			N		V	
			1	2	3	1	2	1	2
l_1	http	www.amazon.com	Head-First-Java	dp	∅	ie	qid	UTF-8	130
l_2	http	www.amazon.com	Effective-Java	dp	∅	ie	qid	UTF-8	333
l_3	http	www.amazon.com	gp	site-directory	ref=topnav_sa	∅	∅	∅	∅
l_4	http	www.amazon.com	Head-First-Java	p-r	∅	ie	qid	UTF-8	130
l_5	http	www.amazon.com	Effective-Java	p-r	∅	ie	qid	UTF-8	333
l_6	http	www.amazon.com	Beginning-Programming	dp	∅	ie	qid	UTF-8	234
l_7	http	www.amazon.com	Sam-Teach-Java	dp	∅	ie	qid	UTF-8	123
l_8	http	www.amazon.com	Introduction-Java	dp	∅	ie	qid	UTF-8	130
l_6	http	www.amazon.com	Beginning-Programming	p-r	∅	ie	qid	UTF-8	234
...									
l_n	http	www.amazon.com	ref=gno_logo	∅	∅	∅	∅	∅	∅

b) Amazon Linkset

LINK	S	A	P		N		V	
			1	2	1	2	1	2
l_1	http	www.amazon.com	Head-First-Java	dp	ie	qid	UTF-8	130
l_2	http	www.amazon.com	Effective-Java	dp	ie	qid	UTF-8	333
l_3	http	www.amazon.com	Sam-Teach-Java	dp	ie	qid	UTF-8	123
l_4	http	www.amazon.com	Beginning-Programming	dp	ie	qid	UTF-8	234
...								
l_n	http	www.amazon.com	Introduction-Java	dp	ie	qid	UTF-8	234

c) Links in L matching prototype p

LINK	S	A	P		N		V	
			1	2	1	2	1	2
l_1	http	www.amazon.com	Head-First-Java	dp	ie	qid	UTF-8	130
l_2	http	www.amazon.com	Head-First-Java	p-r	ie	qid	UTF-8	130

d) $CPL_P(l_1,2)$

Fig. 3: Example of Prototype

and "Help". Examples of these links URLs are, respectively, links with ID 2, 3 and 4, and they are always present in almost any page we choose from the site. Therefore, for any hubset extracted from Amazon, the probability of these URLs is always 1 or near 1.

On the other side, there are links whose appearance depends on the specifical page being considered. For example, links to a page with detailed information about a product, just like links with ID 1, 5 and 6 in the example, only appear in hubs which are responses to certain queries. Therefore, its probability depends on the hubset, although we can assume that, for a random set of hubs, F_L value is rather low.

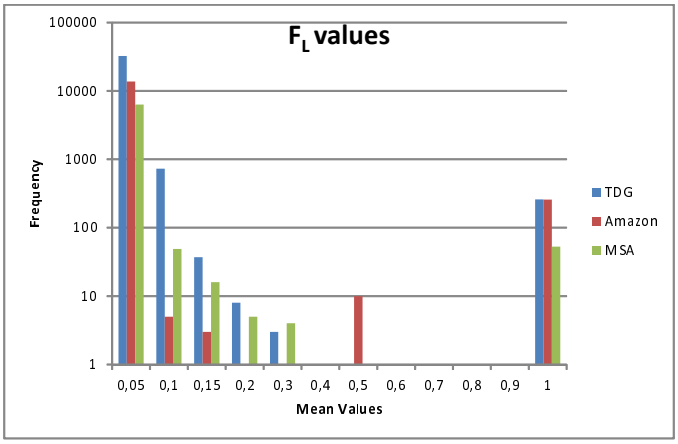
In general, our hypothesis is that for links whose F_L in a hubset H is not 1 (or near 1), it is in fact around $1/n$, that is, probability values are grouped around the two extremes of the distribution (0 and 1), and the number of links whose probability is in the middle of the distribution is very low. Back to Figure 4a, we observe that most values are grouped around 0.05, which means that most links just appear in a range of 1 to 5 hubs, approximately. We must note that there

ID	l	$F_L(l)$
1	http://www.amazon.com/Head-First-Java/dp?ie=UTF8&qid=130	0.01
2	http://www.amazon.com/ref=gno_logo	0.99
3	http://www.amazon.com/Help/b/ref=topnav_help?ie=UTF8&node=508510	0.99
4	http://www.amazon.com/gp/yourstore/ref=pd_irl_gw?ie=UTF8&signIn=1	1.00
5	http://www.amazon.com/Effective-Java/dp?ie=UTF8&qid=130	0.01
6	http://www.amazon.com/Head-First-Java/product-reviews?ie=UTF8	0.03

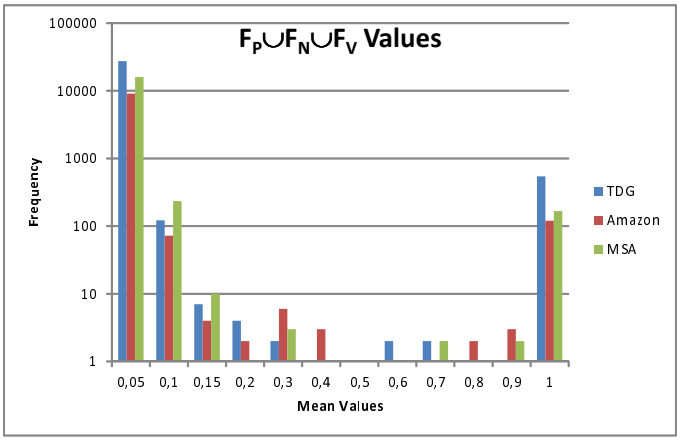
TABLE I: Values for feature F_L in Example III-B

is a small but significant group of values around 1, that is, the group of links that are present in every hub from the site. We can therefore conclude that links with $F_L = 1$ are those belonging to the site template. Hence, our technique allows us to detect the template of a given site, besides classifying its links according the concept contained in their targets.

Let l_1 be the link with ID = 1 in H defined previously. After the experiment, we obtained the values for features F_P, F_N and F_V presented in Table II. As a comparison, in Table III we show the values for features F_P, F_N and F_V for the



(a) F_L values histogram



(b) F_P, F_N, F_V values histogram

Fig. 4: F_P, F_N, F_V and F_L values histogram, from sites: Amazon.com, TDG Scholar and Microsoft Academic Search

l_1 = http://www.amazon.com/Head-First-Java/dp?ie=UTF-8&qid=123		
	i = 1	i = 2
$F_P(l_1, i)$	0.01	0.01
$F_N(l_1, i)$	0.01	0.01
$F_V(l_1, i)$	0.01	0.01

TABLE II: Values for feature F_P, F_N and F_V for link l_1 in Example III-B

prototype p that results when we replace the first path segment in l ("dp") with a wildcard. Based on the former example, we can extract some conclusions from the different values of F_P, F_N and F_V. For example, token "dp", with $F_P(p, 2) = 0.98$, is a fixed part of every link to Amazon product detail pages, and therefore, it is more frequent throughout the site than token "123", whose $F_V(p, 2)$ is near 0 as it is a parameter that identifies queries, and therefore, it is different for every issued query. As a result, its F_V value is 0.01, indicating that it just appears in links from a single hub. Similarly, parameter 1, with name "ie" and value "UTF-8", is also a fixed part in all Amazon links, so their F_N and F_V values respectively are

p = http://www.amazon.com/★/dp?ie=UTF-8&qid=123		
	i = 1	i = 2
$F_P(p, i)$	1	0.98
$F_N(p, i)$	0.99	0.99
$F_V(p, i)$	0.99	0.01

TABLE III: Values for feature F_P, F_N and F_V for prototype p in Example III-B

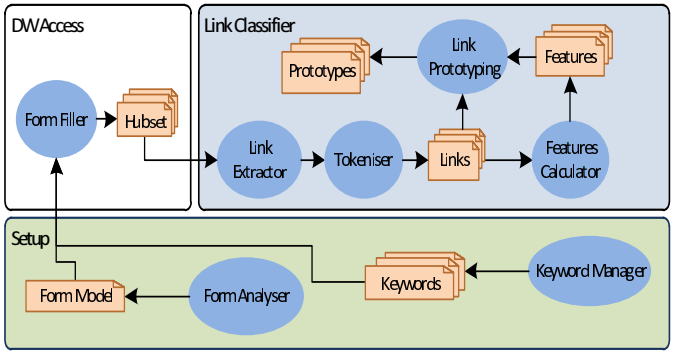


Fig. 5: Diagram of the architecture.

near to 1 in Table III. Our hypothesis regarding F_P, F_N and F_V values is the same exposed earlier for F_L values. In this case, the straightforward application is to build prototypes: tokens with a near-zero value are not relevant, so we can abstract over them and obtain a more general representation of all such segments in the form of a regular expression, that is, of a prototyping token. Meanwhile, tokens with a feature value significantly higher than the others (usually around 1) appear in most hubs, so they are part of the characteristic URL patterns used the site to compose URLs. In other words, they are relevant, and they are not grouped with others to form a prototype; instead, they stay as literals.

IV. Classification Tool

Based on the previous features, we implemented a link classifier, following the architecture in Figure 5. First, a training set is needed, composed by links from the site we wish to extract information from. For this purpose, we make use of the Form Analyser which analyses the forms to obtain a form model, and the Form Filler that uses this model to automatically fill in the form and retrieve the resulting hubs, composing a hubset. Our proposal is focused on keyword-based queries, hence the form filler only deals with forms that contain at least one text field. A Keyword Manager is responsible for finding a corpus of keywords to be used by the form filler, trying to obtain the maximum number of hubs as possible, minimising the keywords that yield no result.

Afterwards, all URLs from the retrieved hubset are extracted and tokenised. For each link, values of features F_P, F_N, F_V and F_L, as defined in section III are calculated, and used to build an ordered set of prototypes, where each prototype represents a different class of links, that is, links leading to pages containing a different concept. Empirical results show

that it is indeed necessary that prototypes keep an order, as we usually obtain some prototypes that subsume other prototypes, that is, a regular expression that is more general than other, and that matches all links matched as well by the latter. To avoid misclassifications, in cases like that we always give more priority to the most specifical prototype than to the most general one.

We developed a proof-of-concept application, obtaining promising evaluation values. Due to space limitations, we only include an example of the classification results in Figure 6, in which we observe Cluster 0 representing the site template links, Cluster 8 representing products, Cluster 9 representing product reviews and Cluster 12 representing authors, amongst others. A demo version of the application can be found in [16].

V. Related Work

A. Web page classification

Web page classification has been extensively researched, and several techniques have been applied with successful experimental results. In general, we catalogue classifiers according to the type and location of the classification features. There are three main trends in feature types: content-based, structure-based and hybrid classifiers. As for feature location, most approaches obtain features from the page to be classified, while others get them from neighbour pages.

Content-based classifiers ([17], [26]) categorize a web page according to the words and sentences it contains. This kind of classifiers group all pages within the same topic, assigning them the same class label. As for structure-based classifiers ([3], [4], [6], [12], [27] and [28]), the main feature used to classify pages is their physical organisation of contents, usually expressed in a tree-like data structure, like a DOM Tree. Also, there are hybrid approaches, [9] and [21], which take into account both content and structural features.

All the previous classifiers consider different kinds of features, but in most cases those features are extracted from the page to be classified, which requires downloading it previously. There are also classifiers that explore the possibility of classifying a web page by using features extracted from neighbour pages, instead of the page itself, being the neighbour of a page another page that has a link to the former, or, conversely, that is linked from it. All these proposals are content-based, and usually rely on features such as the link anchor text, the paragraph text surrounding the anchor [11], the headers preceding the anchor, the words in the URL address, or even a combination of these [15]. If the link is surrounded by a descriptive paragraph or the link itself contains descriptive words, it is possible to decide the page topic in advance of downloading it.

VI. Conclusions

Our proposal classifies pages according to their URL format without downloading them beforehand. Parting from an unlabelled set of links, a set of prototypes is built, each of one representing all links to pages containing a concept embodied in a particular web site. The resulting prototype set can be used by a crawler to improve its efficiency by selecting in each page only links leading to pages with concepts that are interesting for the user, reaching those pages while downloading the minimum number of irrelevant pages. Besides, our classifier is able to detect the template of a web site, that is, links that appear in every page in the site, and hence will most probably not lead to information related to that query.

Using features located on a page to classify it requires previous download, which results in wasted bandwidth and time. There are some proposals that classify pages according to the text surrounding the link in the referring page. This improves the crawlers efficiency, but it is not a general technique, given that not all links include in their surroundings words useful for classification. Our proposal classifies web pages depending on the link URL format, so it is not only efficient, but also generic and applicable in different domains. Besides, user supervision is kept to a minimum, given that the classifier is trained using an unlabelled set of links collected automatically.

These links are analysed and a set of prototypes is built, each of them representing all URLs that link to pages containing a different concept. Therefore users do not have to label large training sets, as it happens in supervised classifiers; instead they are only responsible for defining his or her interest, by picking the related prototypes. Note that users intervention is unavoidable, given that the relevancy criteria depends solely on them, but in our proposal it is kept to a minimum.

Traditional crawlers browse the whole sites, retrieving all pages, and spending a significant time and bandwidth while downloading them. Focused crawlers retrieve pages belonging to a topic more efficiently than traditional crawlers do, but still they are not a suitable solution for virtual integration systems, because a page has to be classified to know if the crawler must follow that path, and that requires the page to be downloaded in most cases.

The prototypes thus generated can later be used by a crawler to classify links following the template matching approach, that is, links are compared against prototypes, and assigned to the first class whose prototypes they match. If the link class is marked as relevant by the user, the links are followed by the crawlers; otherwise, they are ignored, making the crawler more efficient. As a result, we designed a link classifier that lays the foundations for an efficient crawler, able to access web pages automatically, while requiring as little intervention as possible from the users.

Acknowledgment

This paper was supported by the European Commission (FEDER), the Spanish and the Andalusian R&D&I programmes (grants TIN2007-64119, P07-TIC-2602, P08-TIC-4100, TIN2008-04718-E, TIN2010-21744, TIN2010-09809-E, TIN2010-10811-E, and TIN2010-09988-E).

Fig. 6: Example of Link Classification: Amazon.com hub page.

REFERENCES

[1] Charu C. Aggarwal, Fatima Al-Garawi, and Philip S. Yu. On the design of a learning crawler for topical resource discovery. *ACM Trans. Inf. Syst.*, 19(3):286–309, 2001.

[2] Vinod Anupam, Juliana Freire, Bharat Kumar, and Daniel F. Lieuwen. Automating web navigation with the webvcr. *Computer Networks*, 33(1-6):503–517, 2000.

[3] Arvind Arasu and Hector Garcia-Molina. Extracting structured data from web pages. In *SIGMOD Conference*, pages 337–348, 2003.

[4] Ziv Bar-Yossef and Sridhar Rajagopalan. Template detection via data mining and its applications. In *WWW*, pages 580–591, 2002.

[5] Claudio Bertoli, Valter Crescenzi, and Paolo Merialdo. Crawling programs for wrapper-based applications. In *Information Reuse and Integration*, pages 160–165, 2008.

[6] Lorenzo Blanco, Valter Crescenzi, and Paolo Merialdo. Structure and semantics of Data-IntensiveWeb pages: An experimental study on their relationships. *J. UCS*, 14(11):1877–1892, 2008.

[7] Lorenzo Blanco, Nilesh Dalvi, and Ashwin Machanavajjhala. Highly efficient algorithms for structural clustering of large websites. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 437–446, New York, NY, USA, 2011. ACM.

[8] Jim Blythe, Dipsy Kapoor, Craig A. Knoblock, Kristina Lerman, and Steven Minton. Information integration for the masses. *J. UCS*, 14(11):1811–1837, 2008.

[9] James Caverlee and Ling Liu. Qa-pagelet: Data preparation techniques for large-scale data analysis of the deep web. *IEEE Trans. Knowl. Data Eng.*, 17(9):1247–1262, 2005.

[10] Soumen Chakrabarti. Focused web crawling. In *Encyclopedia of Database Systems*, pages 1147–1155. 2009.

[11] William W. Cohen. Improving a page classifier with anchor extraction and link analysis. In *NIPS*, pages 1481–1488, 2002.

[12] Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. RoadRunner: Towards automatic data extraction from large web sites. In *Very Large Data Bases*, pages 109–118, 2001.

[13] Guilherme T. de Assis, Alberto H. F. Laender, Marcos André Gonçalves, and Altigran Soares da Silva. Exploiting genre in focused crawling. In *String Processing and Information Retrieval*, pages 62–73, 2007.

[14] Jenny Edwards, Kevin S. McCurley, and John A. Tomlin. An adaptive model for optimizing performance of an incremental web crawler. In *WWW*, pages 106–113, 2001.

[15] Johannes Fürnkranz. Hyperlink ensembles: a case study in hypertext classification. *Information Fusion*, 3(4):299–312, 2002.

[16] Inma Hernández. Relc demo. http://www.tdg-seville.info/inmahernandez/Thesis+Demo, 2011.

[17] Andreas Hotho, Alexander Maedche, and Steffen Staab. Ontology-based text document clustering. *KI*, 16(4):48–54, 2002.

[18] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[19] Juliano Palmieri Lage, Altigran Soares da Silva, Paulo Braz Golgher, and Alberto H. F. Laender. Automatic generation of agents for collecting hidden web pages for data extraction. *Data Knowl. Eng.*, 49(2):177–196, 2004.

[20] Stephen W. Liddle, David W. Embley, Del T. Scott, and Sai Ho Yau. Extracting data behind web forms. In *ER (Workshops)*, pages 402–413, 2002.

[21] Alex Markov, Mark Last, and Abraham Kandel. The hybrid representation model for web document classification. *Int. J. Intell. Syst.*, 23(6):654–679, 2008.

[22] Sougata Mukherjea. Discovering and analyzing world wide web collections. *Knowl. Inf. Syst.*, 6(2):230–241, 2004.

[23] Alberto Pan, Juan Raposo, Manuel Álvarez, Justo Hidalgo, and Ángel Viña. Semi-automatic wrapper generation for commercial web sources. In *Engineering Information Systems in the Internet Context*, pages 265–283, 2002.

[24] Gautam Pant and Padmini Srinivasan. Link contexts in classifier-guided topical crawlers. *IEEE Trans. Knowl. Data Eng.*, 18(1):107–122, 2006.

[25] Ioannis Partalas, Georgios Paliouras, and Ioannis P. Vlahavas. Reinforcement learning with classifier selection for focused crawling. In *European Conference on Artificial Intelligence*, pages 759–760, 2008.

[26] Ali Selamat and Sigeru Omatu. Web page feature selection and classification using neural networks. *Inf. Sci.*, 158:69–88, 2004.

[27] Márcio L. A. Vidal, Altigran Soares da Silva, Edleno Silva de Moura, and João M. B. Cavalcanti. Structure-based crawling in the hidden web. *J. UCS*, 14(11):1857–1876, 2008.

[28] Karane Vieira, Altigran Soares da Silva, Nick Pinto, Edleno Silva de Moura, João M. B. Cavalcanti, and Juliana Freire. A fast and robust method for web page template detection and removal. In *CIKM*, pages 258–267, 2006.

[29] Yang Wang and Thomas Hornung. Deep web navigation by example. In *BIS (Workshops)*, pages 131–140, 2008.

[30] Lan Yi, Bing Liu, and Xiaoli Li. Eliminating noisy information in web pages for data mining. In *Knowledge Discovery and Data Mining*, pages 296–305, 2003.