A Tool for Link-Based Web Page Classification

Inma Hernández, Carlos R. Rivero, David Ruiz, and Rafael Corchuelo

Seville, Spain

{inmahernandez,carlosrivero,druiz,corchu}@us.es

Abstract. Virtual integration systems require a crawler to navigate through web sites automatically, looking for relevant information. This process is online, so whilst 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 to improve the crawler efficiency. Most crawlers need to download a page to determine its relevance, which results in a high number of irrelevant pages downloaded. In this paper, 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 by analysing exclusively its URL, minimising the number of irrelevant pages downloaded, improving crawling efficiency and reducing used bandwidth, making it suitable for virtual integration systems.

Keywords: Crawling, Web Page Classification, Virtual Integration.

1 Introduction

Virtual Integration aims at accessing web information in an automated manner, retrieving information relevant to a user 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 and check if it is relevant or not, which results in a large number of irrelevant pages downloaded.

Note that the Virtual Integration process is online, which means that whilst 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 to improve the crawler efficiency, which is a concern for several researchers [9,15,26].

There are some techniques that improve traditional crawlers efficiency by endowing the crawler with classification skills. For example, focused crawlers

⋆ 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).

find pages belonging to one or more topics exclusively, so they are supported by a content-based classifier that determines whether each page belongs to those topics [1,11,14,22,24]. Other crawlers include classifiers based on other features like page structure [19,20,27]. Finally, there are crawling techniques that rely completely on the user to define navigation patterns [2,5,8,23,29].

In this paper, we focus 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; then, after users submit a query, the system returns a hub, i.e., a page containing an indexed list of answers to it, each of which contains just a brief description and a link to a detail page. Note that the term "hub" is based on the hub and authority concepts introduced by Kleinberg [18].

In this kind of web sites, hubs are created by instantiating scripts with data stored in a database [7]. This means that all hubs from the same web site share a common template, usually 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. Similarly, URLs that point to each hub and detail page are generated as well by the same process of filling a URL pattern with keywords that identify the generated page. Therefore, all URLs from a certain site can be expressed by a collection of URL patterns.

We propose a classifier that helps crawlers to efficiently navigate through web sites, by determining if a web page is relevant by analysing exclusively its URL. Our classifier is different to existing proposals, 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 to classify it, which avoids downloading irrelevant pages, reducing the bandwidth and making it efficient and suitable for Virtual Integration systems. Moreover, our proposal is automated, requiring a minimum intervention from users. 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 training page.

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 or prototypes that compose links in a given web site. Then, we use these prototypes to classify links by template matching. Furthermore, our technique is able as well to detect links belonging to the Web site template.

The rest of the paper is structured as follows. Section 2 presents the related work in the web page classification area; Sections 3 and 4 introduce the core definitions that will be used throughout the paper; Section 5 describes the tool design; Section 6 presents the evaluation of our tool; finally, Section 7 lists some of the conclusions drawn from the research and concludes the paper.

2 Related Work

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, whilst others get them from neighbour pages.

Content-based classifiers ([17,25]) categorize a web page according to the words and sentences it contains. These kinds of classifiers group all pages within the same topic, assigning them the same class label. As for structure-based classifiers ([3,4,6,13,27,28]), 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 [10,21] which take both content and structural features into account.

All 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 [12], the headers preceding the anchor, the words in the URL address, or even a combination of them [16]. 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.

3 Core Definitions

In this Section, we introduce some preliminary concepts that will be used throughout the rest of the paper.

Hub. Each hub is defined by the set of links that it comprises, $H_i = \{l_1, l_2, l_3, ..., l_m\}$

Hubset. Set of hubs obtained from a particular site. $H = \{H_1, H_2, H_3, ..., H_n\}$

Linkset. Set of links that are comprised in a hubset H, $L = \bigcup_{i=1}^{n} H_i \in H$

Link. Tuple that represents a URL, $l = (S, A, P, N, V)$. Links are obtained from URLs by means of a tokeniser, according to RFC 3986, 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. For the sake of simplicity, throughout the paper we use the notation X to refer to any of the sequences P, N or V.

Prototype. Link $p = (S, A, P, N, V)$ that represents a URL pattern, where each element in P, N and V is either a literal or a wildcard, $\bigstar$. A wildcard represents any sequence of characters (excluding separators '?', '/', '#', '=' and '&')

Common Path Links. Let L be a linkset from a given site, l be a link in L and $X(i)$ be the i-th element of sequence X in l. We define the set $CPL_X(l, i)$ as the set of all links l' in L having the same prefix as l up to (and excluding) $X(i)$. Recall that a prototype is a link that includes some wildcard sections, so we can calculate the CPL set of a prototype likewise.

4 Classification Features

In this Section, 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.

4.1 Features Definition

Definition 1 (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.*

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

In Equation 1, we must assure that the hubset is sufficiently large so that the probability estimation is statistically significant, hence 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.

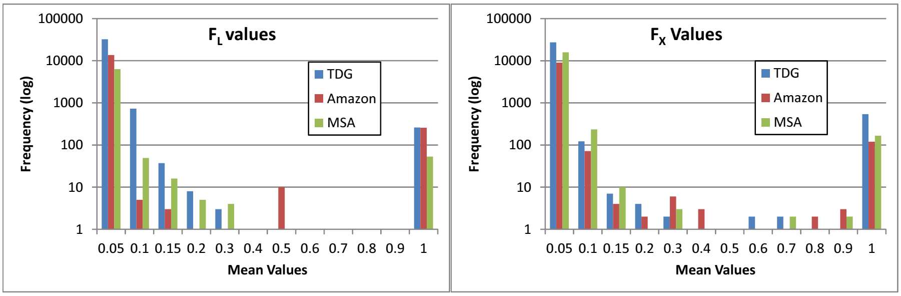
As an example, Figure 1a shows the histogram of F_L values obtained from 100 hubs in an e-commerce site (Amazon.com) an two academic sites (Microsoft Academic Search and TDG Scholar).

Definition 2 (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, Q) 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.*

$$F_X(l, i) = \frac{|\{H_j \in H \cdot H_j \cap CPL_X(l, i+1) \neq \emptyset, j \in [1, n]\}|}{n} \tag{2}$$

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 1b 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.



(a) F_L values histogram

(b) F_P, F_N, F_V values histogram

Fig. 1. F_P, F_N, F_V and F_L values histogram, from sites: Amazon.com, TDG Scholar and Microsoft Academic Search

Given that a prototype has the same signature as a link, both previous definitions 1 and 2 are applicable as well to prototypes. For the sake of simplicity, we assume that $F_P(p, i) = 1$ iff $p = \{S, A, P, N, V, Q\} \wedge P(i) = \bigstar$ (similarly, with F_N and F_V).

4.2 Features Examples

Example 1. 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 1.

All Amazon pages contain a navigation bar in their upper part, including links such as “Home” “Sign In” and “Help”. Examples of these links URLs are, respectively, links with ID 2, 3 and 4, and they are always present in every page from the site. Therefore, for any hubset extracted from Amazon, the probability of these URLs is always 1.

On the other side, there are links whose appearance depends on the specific 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 answers 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 $1/n$, i.e., 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 1a, 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 is a small but significant group of values around 1, i.e., 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.

Table 1. Values for feature F_L in Example 1

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

Let l_1 be the link with ID = 1 in previously defined H. After the experiment, we obtain the values for features F_P, F_N and F_V presented in Table 2a. As a comparison, in Table 2b, we show the values for features F_P, F_N and F_V for the prototype p that results when we replace the first path segment in l_1 ("Head-First-Java") with a wildcard.

Table 2. Values for features F_P, F_N and F_V for l_1 and p, in Example 1

	X(i)	Value
$F_P(l_1, 1)$	Head-First-Java	0.01
$F_P(l_1, 2)$	dp	0.01
$F_N(l_1, 1)$	ie	0.01
$F_N(l_1, 2)$	qid	0.01
$F_V(l_1, 1)$	UTF-8	0.01
$F_V(l_1, 2)$	123	0.01

(a) Values for l_1

	X(i)	Value
$F_P(p, 1)$	★	1
$F_P(p, 2)$	dp	0.98
$F_N(p, 1)$	ie	0.99
$F_N(p, 2)$	qid	0.99
$F_V(p, 1)$	UTF-8	0.99
$F_V(p, 2)$	123	0.01

(b) Values for p

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 near to 1 in Table 2b.

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, i.e., 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 to compose site URLs, i.e., they are relevant, so we keep them as literals.

5 Classification Tool

Based on the previous features, we implemented a link classifier, following the architecture in Figure 2. 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.

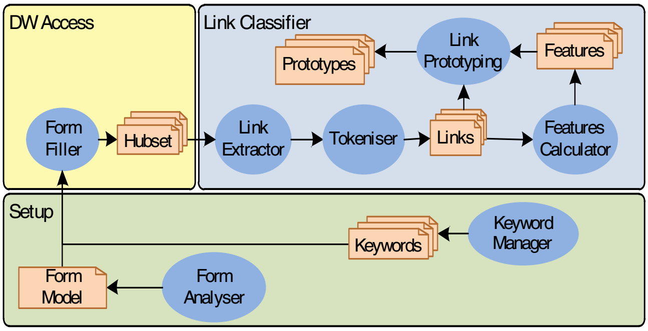
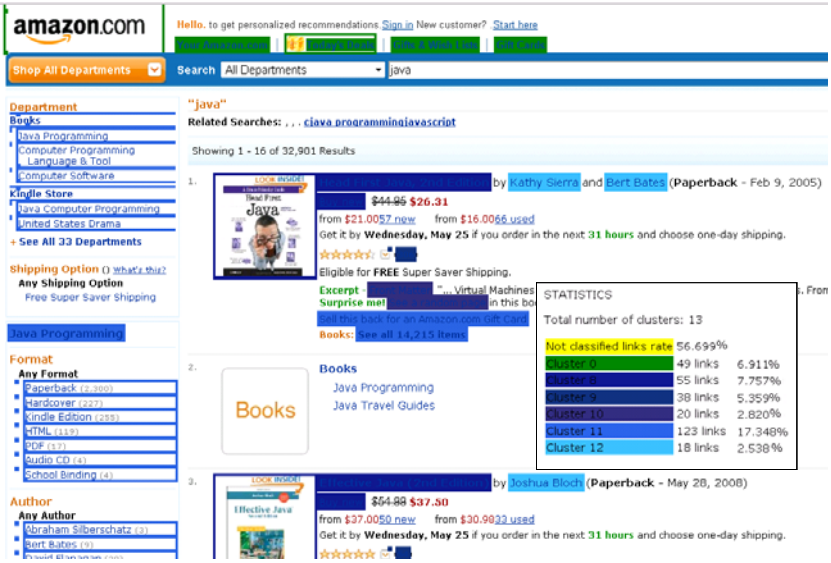


Fig. 2. Workflow of the architecture

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 4 are calculated, and used to build an ordered set of prototypes, where each prototype represents a different class of links, i.e., links leading to pages containing a different concept. Some prototypes may subsume other prototypes, i.e., 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 a higher priority to the most specifical prototype.

6 Evaluation

We developed a proof-of-concept application, based on the former architecture. An example of the classification results is presented in Figure 3. We observe that Cluster 0 represents the site template links, Cluster 8 products (`http://www.amazon.com/★/dp/★`), Cluster 9 product reviews (`http://www.amazon.com/★/product-reviews/★`) and Cluster 12 authors (`http://www.amazon.com/★/e/★`), amongst others.


amazon.com
Hello. to get personalized recommendations. Sign in New customer? Start here
Your Amazon.com
Today's Deals
Gifts & Wish Lists
Gift Cards
Shop All Departments
Search
All Departments
java
Department
Books
Java Programming
Computer Programming Language & Tool
Computer Software
Kindle Store
Java Computer Programming
United States Drama
+ See All 33 Departments
Shipping Option () What's this?
Any Shipping Option
Free Super Saver Shipping
Java Programming
Format
Any Format
Paperback (2,300)
Hardcover (227)
Kindle Edition (255)
HTML (119)
PDF (17)
Audio CD (4)
School Binding (4)
Author
Any Author
Abraham Silberschatz (3)
Bert Bates (9)
"java"
Related Searches: , , , cjava programmingjavascript
Showing 1 - 16 of 32,901 Results
1.
Head First Java, 2nd Edition by Kathy Sierra and Bert Bates (Paperback - Feb 9, 2005)
Buy new $44.95 $26.31
from $21.0057 new from $16.0066 used
Get it by Wednesday, May 25 if you order in the next 31 hours and choose one-day shipping.
Eligible for FREE Super Saver Shipping.
Excerpt - Front Matter: "... Virtual Machines
Surprise me! See a random page in this bo
Sell this back for an Amazon.com Gift Card
Books: See all 14,215 items
2.
Books
Books
Java Programming
Java Travel Guides
3.
Effective Java (2nd Edition) by Joshua Bloch (Paperback - May 28, 2008)
$54.99 $37.50
from $37.0050 new from $30.9933 used
Get it by Wednesday, May 25 if you order in the next 31 hours and choose one-day shipping.

STATISTICS

Total number of clusters: 13

Not classified links rate	56.699%	
Cluster 0	49 links	6.911%
Cluster 8	55 links	7.757%
Cluster 9	38 links	5.359%
Cluster 10	20 links	2.820%
Cluster 11	123 links	17.348%
Cluster 12	18 links	2.538%

Fig. 3. Example of Link Classification: Amazon.com hub page

We performed an experiment to test our tool, evaluating the most relevant concepts on three different sites. The classification was evaluated by means of 10-fold cross evaluation, obtaining values for precision, recall and f1-measure in Table 3. For each measure, we show its mean value, as well as the confidence interval of 95%.

Table 3. Evaluation results

Site	Concept	Precision	Recall	F1-Measure
Amazon	Products	0.978 ± 0.022	0.703 ± 0.033	0.818 ± 0.011
	Reviews	0.978 ± 0.029	0.705 ± 0.031	0.819 ± 0.030
TDG Scholar	Authors	0.908 ± 0.005	0.761 ± 0.004	0.828 ± 0.014
Ms Academic	Papers	0.979 ± 0.003	0.864 ± 0.006	0.851 ± 0.023

We observe that recall values are always lower than precision. We have concluded that our proposal yields prototypes that are too specific, so our future work is focused on improving these results by means of post processing.

7 Conclusions

Our proposal classifies pages according to their URL format without downloading them beforehand, saving bandwidth and time. Parting from an unlabelled set of links, a set of prototypes is built, each of which represents 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 whilst downloading the minimum number of irrelevant pages. Besides, our classifier is able to detect the template of a web site, i.e., links that appear in every page in the site, and hence will most probably not lead to information related to that query.

There are some proposals that classify pages according to the text surrounding the link in the referring page. This 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.

References

1. Aggarwal, C.C., Al-Garawi, F., Yu, P.S.: On the design of a learning crawler for topical resource discovery. ACM Trans. Inf. Syst. 19(3), 286–309 (2001)
2. Anupam, V., Freire, J., Kumar, B., Lieuwen, D.F.: Automating web navigation with the webvcr. Comp. Netw. 33(1-6), 503–517 (2000)
3. Arasu, A., Garcia-Molina, H.: Extracting structured data from web pages. In: SIGMOD, pp. 337–348 (2003)
4. Bar-Yossef, Z., Rajagopalan, S.: Template detection via data mining and its applications. In: WWW, pp. 580–591 (2002)
5. Bertoli, C., Crescenzi, V., Merialdo, P.: Crawling programs for wrapper-based applications. In: IRI, pp. 160–165 (2008)
6. Blanco, L., Crescenzi, V., Merialdo, P.: Structure and semantics of Data-IntensiveWeb pages: An experimental study on their relationships. J. UCS 14(11), 1877–1892 (2008)
7. Blanco, L., Dalvi, N., Machanavajjhala, A.: Highly efficient algorithms for structural clustering of large websites. In: WWW 2011, pp. 437–446. ACM (2011)
8. Blythe, J., Kapoor, D., Knoblock, C.A., Lerman, K., Minton, S.: Information integration for the masses. J. UCS 14(11), 1811–1837 (2008)
9. Boldi, P., Codenotti, B., Santini, M., Vigna, S.: UbiCrawler: a scalable fully distributed web crawler. Softw., Pract. Exper. 34(8), 711–726 (2004)
10. Caverlee, J., Liu, L.: Qa-pagelet: Data preparation techniques for large-scale data analysis of the deep web. IEEE Trans. Knowl. Data Eng. 17(9), 1247–1262 (2005)

11. Chakrabarti, S.: Focused web crawling. In: Encyclopedia of Database Systems, pp. 1147–1155 (2009)
12. Cohen, W.W.: Improving a page classifier with anchor extraction and link analysis. In: NIPS, pp. 1481–1488 (2002)
13. Crescenzi, V., Mecca, G., Merialdo, P.: RoadRunner: Towards automatic data extraction from large web sites. In: VLDB, pp. 109–118 (2001)
14. de Assis, G.T., Laender, A.H.F., Gonçalves, M.A., da Silva, A.S.: Exploiting Genre in Focused Crawling. In: Ziviani, N., Baeza-Yates, R. (eds.) SPIRE 2007. LNCS, vol. 4726, pp. 62–73. Springer, Heidelberg (2007)
15. Edwards, J., McCurley, K.S., Tomlin, J.A.: An adaptive model for optimizing performance of an incremental web crawler. In: WWW, pp. 106–113 (2001)
16. Fürnkranz, J.: Hyperlink ensembles: a case study in hypertext classification. Inf. Fusion 3(4), 299–312 (2002)
17. Hotho, A., Maedche, A., Staab, S.: Ontology-based text document clustering. In: KI, vol. 16(4), pp. 48–54 (2002)
18. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. J. ACM 46(5), 604–632 (1999)
19. Lage, J.P., da Silva, A.S., Golgher, P.B., Laender, A.H.F.: Automatic generation of agents for collecting hidden web pages for data extraction. Data Knowl. Eng. 49(2), 177–196 (2004)
20. Liddle, S.W., Embley, D.W., Scott, D.T., Yau, S.H.: Extracting Data Behind Web Forms. In: Olivé, À., Yoshikawa, M., Yu, E.S.K. (eds.) ER 2003. LNCS, vol. 2784, pp. 402–413. Springer, Heidelberg (2003)
21. Markov, A., Last, M., Kandel, A.: The hybrid representation model for web document classification. Int. J. Intell. Syst. 23(6), 654–679 (2008)
22. Mukherjea, S.: Discovering and analyzing world wide web collections. Knowl. Inf. Syst. 6(2), 230–241 (2004)
23. Pan, A., Raposo, J., Álvarez, M., Hidalgo, J., Viña, Á.: Semi-automatic wrapper generation for commercial web sources. In: EISIC, pp. 265–283 (2002)
24. Pant, G., Srinivasan, P.: Link contexts in classifier-guided topical crawlers. IEEE Trans. Knowl. Data Eng. 18(1), 107–122 (2006)
25. Selamat, A., Omatu, S.: Web page feature selection and classification using neural networks. Inf. Sci. 158, 69–88 (2004)
26. Shkapenyuk, V., Suel, T.: Design and implementation of a high-performance distributed web crawler. In: ICDE, pp. 357–368 (2002)
27. Vidal, M.L.A., da Silva, A.S., de Moura, E.S., Cavalcanti, J.M.B.: Structure-based crawling in the hidden web. J. UCS 14(11), 1857–1876 (2008)
28. Vieira, K., da Silva, A.S., Pinto, N., de Moura, E.S., Cavalcanti, J.M.B., Freire, J.: A fast and robust method for web page template detection and removal. In: CIKM, pp. 258–267 (2006)
29. Wang, Y., Hornung, T.: Deep web navigation by example. In: BIS (Workshops), pp. 131–140 (2008)
30. Yi, L., Liu, B., Li, X.: Eliminating noisy information in web pages for data mining. In: KDD, pp. 296–305 (2003)