Focused Deep Web Entrance Crawling by Form Feature Classification

Lin Wang(✉), Ammar Hawbani, and Xingfu Wang

Computer Science and Technology, University of Science and Technology of China, Hefei 230022, Anhui, China
wangxfu@ustc.edu.cn, ammar12@gmail.com, xiaquhet@mail.ustc.edu.cn

Abstract. Currently, Most back-end web databases cannot be indexed by traditional hyperlink-based search engines due to their requirement of users' interactive queries via page form submission. In order to make hidden-Web information more easily accessible, this paper proposes a hierarchical classifier to locate domain-specific hidden Web entries at a large scale. The classifier is trained by appropriately selected page form features to get rid of non-relevant domains and non-searchable forms. Experiments conducted on eight different topics demonstrate that the technique can discover deep web interfaces accurately and efficiently.

Keywords: Deep web · Focused crawler · Searchable forms · HTML analysis · SVM classifier · Decision tree algorithm

1 Introduction

In recent years, the hidden web has been growing at a very fast pace. For a given domain of interest, there are many hidden-web sources needing to be integrated or searched[16]. Examples of applications that attempt to leverage these resources include: met searchers[3,18,23,24], hidden-web crawlers [9,20], online-database directories[1,17] and information integration systems [13,19,22].

A key requirement for these applications is the ability to locate the search entry, but accurately doing so at a large scale is a challenging problem due to the following difficulties:

- First, searchable forms are very sparsely distributed over the web. For example, a topic-focused best-first crawler [11] retrieves only 94 movie search forms after crawling 100,000 pages related to movies;
- In addition, the set of retrieved forms also includes many non-searchable forms that do not represent database queries such as forms for login, mailing list subscriptions, quote requests, and web-based email forms[8];
- Last but not least, the set of forms retrieved is also very heterogeneous it includes all searchable forms belong to distinct database domains with

L. Wang—Supported in part by the National Science Foundation under grant 61472382, 61272472 and 61232018

Y. Wang et al. (Eds.): BigCom 2015, LNCS 9196, pp. 79–87, 2015.
DOI: 10.1007/978-3-319-22047-5_7

different structure and textual features, making automatic method inefficient. For example, only 16% of the searchable forms retrieved by a form-focused crawler[6] are actually relevant.

In this paper, we present a new framework that addresses these challenges: firstly, we use a modified best-first crawler to just find domain-specific deep database entries; secondly, we use a hierarchical framework, utilizing page textual and html form features to guide our hidden page gathering process.

The remainder of the paper is organized as follows: in section 2 we give a brief overview of related work; in section 3, we present page form classifiers and describe the underlying framework in section 4; our experimental evaluation is discussed in section 5; we conclude in section 6.

2 Related Work

The huge growth of the deep web has motivated interest in the study of better crawlers, some important works include:

Ref.[15] introduces a page division method to distinguish traditional search engine interfaces from deep web interfaces and constructs topic-specific queries to obtain results for further conformation by analyzing the results.

Cope, et al.[14] use an automatic feature generator to depict candidate forms and a C4.5 decision tree to classify them. In their two testbeds – ANU collection and a random Web collection, they get an accuracy of more than 85% and 87% respectively.

Bergholz, et al.[10] describe a crawler which starts from the Publicly Indexable Web (PIW) to find entrance into the deep Web. This crawler is domain-specific and is initialized with pre-classified documents and relevant keywords.

In Ref.[8], they present a new adaptive focused crawling strategy for efficiently locating hidden Web entry points. Unfortunately, the ACHE framework they proposed cannot handle very sparse domains efficiently. Besides, the ACHE framework is complex and its overhead is large.

3 Two-Step Classifying Framework

In order to find domain-specific hidden Web entrance, we utilize two classifiers working in a hierarchical fashion to show directions for our crawler. The two classifiers are page text classifier and form feature classifier. Figure 1 shows the high-level architecture:

- First, given a URL we find its corresponding home page and check whether it is domain-specific using the page text classifier. Our crawler only digs into those sites containing domain-specific home pages. Previous researches[7,8] demonstrate that libsvm learning algorithm[12] can be used;
- Second, if a Web page is relevant, we extract searchable forms from it with the aid of the form feature classifier. According to Luciano, et al.[6] and Cope, et al.[14], a decision tree will be optimal in this case.

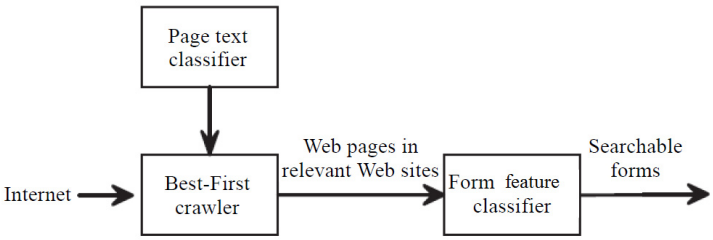

Fig. 1. The high-level architecture

The reason why we utilize classifiers in this hierarchical style is that the hierarchical structure leads to the merits of modularity. As a complex problem is broken down into simpler sub-parts, we can apply to each part a learning method that is best suited for the feature set of the partition, thus enabling the overall classification process to be not only accurate but also robust.

3.1 Form Feature Classification

A form is made up of structural and textual parts. Consider the famous Lucene Apache home page as an example, where we can find in-site search entry shown in Fig.2, which not only contains textual contents such as "sort", "Search", but also structural contents such as select elements, submission buttons.

Fig. 2. An illustration of form entry

```
<form method="get" action="/search" name="f" class="searchbox">
    <input type="text" name="query" value="" size="35">
    sort: <select name="mode">
        <option value="none"> time-biased relevance </option>
        <option value="pure"> relevancy </option>
        <option value="newestOnTop"> newest </option>
        <option value="oldestOnTop"> oldest </option>
    </select> & nbsp; <input type="submit" value="Search">
</form>
```

In order to identify whether a form is domain-specific searchable or not, we count 12 most distinguishable features N1 $\sim$ N12 (both structural and textual features included) to train the form classifier, depicted in Table 2.

3.2 Page Textual Feature Classification

To extract textual features from pages, some pre-processing steps are needed:

- First, all characters other than alphanumeric ones are replaced by a space character;
- Second, uppercase characters, if any, are converted to their lower case equivalents;
- Third, stop words, if any, are removed, using org.apache.lucene.analysis.StopAnalyzer;
- Fourth, each word in the remaining texts is stemmed, using org.apache.lucene.analysis.PorterStemmer;
- Finally, TFIDF[21] is used to transform each training example into its corresponding vector.

4 Modified Best-First Crawler

To improve the efficiency of our page gathering operation, we make modifications of the standard best-first version[10]. The detailed control flow of our crawler is displayed in Fig.3. It crawls within each domain-specific site until $depth \geq 3$ or the total number of pages $threshold \geq 100$ is visited. The reason why we set

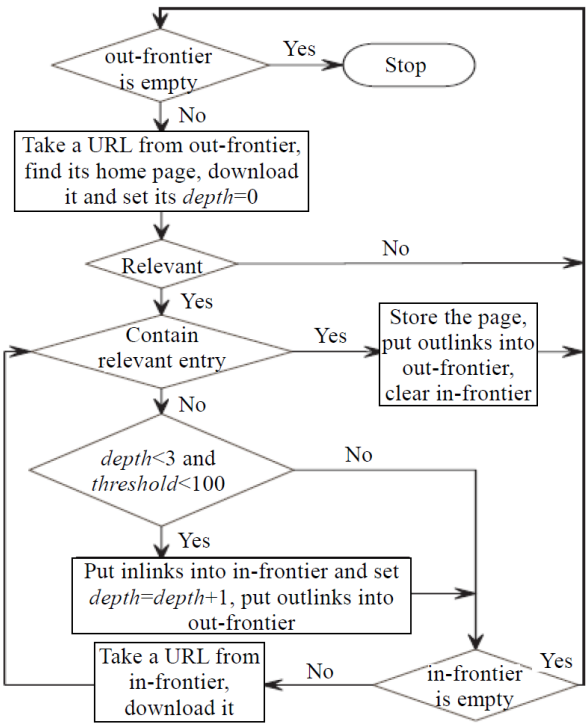

Fig. 3. The modified best-first crawler

depth < 3 is that Web databases tend to locate shallowly in their sites and the vast majority of them (approximately 94%) can be found at the top 3 levels[20]. Besides, in order to protect our crawler from getting trapped in some sites, we set a threshold for maximum pages to visit per site. Such variation enables the crawler to be searching in a more promising space.

5 Experiments

TEL-8 Query Interfaces[5] dataset is used to train our form classifier. The dataset contains 223 original interfaces extracted from eight representative domains, which are displayed in the first two column of Table 1.

Table 1. TEL-8 distributions of the eight domains

Domain	Sources	Positive	Negative	SVM	Overall
Airfare	20	116	316	0.961 9	0.90
Auto	28	251	356	0.946 3	0.88
Book	43	156	332	0.913 5	0.91
Rental	13	91	228	0.973 7	0.95
Hotel	34	170	272	0.994 1	0.94
Job	20	170	317	0.979 1	0.81
Movie	32	160	312	0.900 4	0.86

5.1 Train Form Feature Classifier

The form classifier is trained by decision tree algorithm. The training data are collected as follows: we extract 223 searchable forms from TEL-8 Query Interfaces as positive examples and manually gather 318 non-searchable forms as negative ones. From Table 2, we can deduce the following implications:

Searchable forms have plenty of checkboxes and 'option' items in selection lists.

No-Searchable forms have plenty of password tags and 'email' in input elements' name or value.

Two tools are used to construct the classification tree: R rpart algorithm[4] and Matlab fitctree function[2]. The decision tree generated by R is displayed in Fig.4, with the precision 0.949. And the fitctree in Matlab statistics toolbox reaches a precision of 0.917. They have similar results according to our experiments.

5.2 Train Page Textual Feature Classifier

To collect positive training samples for page textual feature classification, we apply a Python script to automatically fill out the query interface on the homepage of the online open directory project (http://dmoz.org/) and extract URLs

Table 2. Feature distributions of searchable and non-searchable forms

Feature	Number of	Searchable	Non-Searchable	Ratio
N1	**checkbox**	2.39	0.18	**13.04:1**
N2	**email-yes**	0.01	0.12	**1:13.13**
N3	file inputs	0.00	0.00	-
N4	hidden tags	4.45	1.63	2.72:1
N5	image inputs	0.36	0.21	1.73:1
N6	submission method-get	0.47	0.40	1.16:1
N7	**select option**	12.64	0.17	**74.23:1**
N8	**password tags**	0.02	0.10	**1:5.86**
N9	radio tags	0.48	0.10	4.76:1
N10	search-yes[1]	0.36	0.09	3.94:1
N11	text elements	3.00	1.01	2.97:1
N12	textarea elements	0.02	0.07	1:2.98

[1] Word 'search' within form tag or submission button

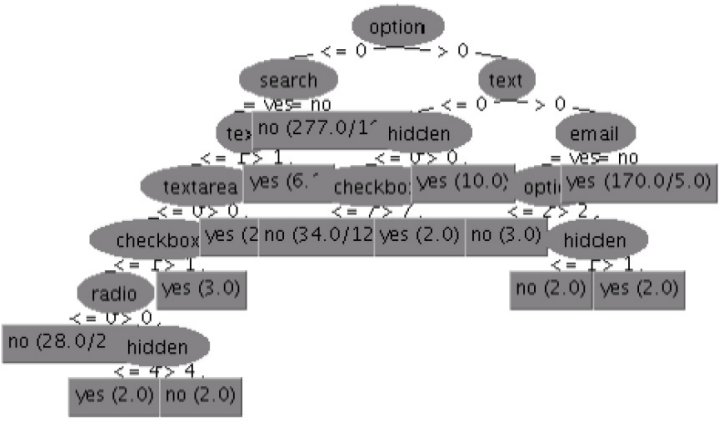

Fig. 4. The decision tree generated by R rpart algorithm

from the returned result pages. As for negative URL samples, we get them from the RDF dumps of DMOZ (http://rdf.dmoz.org/rdf/). DMOZ covers sixteen topics, among which we select 12 in our experiments, which are shown in Table 3,

In DMOZ, each example looks like this:

```
<ExternalPage about=
      "http://www.airwise.com/airports/us/SLC/index.html">
  <d:Title> Salt Lake City Airport - airwise.com </d:Title>
  <d:Description> Information about the airport including
      airlines, ground transportation, parking, weather and
      airport news.
  </d:Description>
```

```
  <topic>Top/Regional/North_America/United_States/Utah/
      Localities/S/Salt_Lake_City/Transportation/Airports
  </topic>
</ExternalPage>
```

Table 3. Number of URLs in each DMOZ category

Arts	Business	Computers	Games	Health	Home
585 923	511 621	285 335	123 757	131 051	33 554
News	Recreation	Science	Shopping	Society	Sports
235 703	120 307	213 013	235 161	269 863	154 920

We use the content of 'd:Description' element and the Web page corresponding to the 'about' ExternalPage attribute to obtain a negative training page.

In order to be more representative, we derive URLs from each category according to its size. Since the 'Arts' category has the largest number of URLs, we get the most number of URLs from it. Excluded these URLs which cannot be downloaded, the number of positive and negative examples which we use to train a page classifier for each category is listed in the 3rd and 4th column of Table 1, where the precisions of the 8 trained SVM classifiers are also shown in the 5th column.

Given a Web page, we first obtain its corresponding plain texts by stripping away HTML tags. We will also strip embedded JavaScript code, style information (style sheets), as well as code inside php/asp tags (<?php ?> <%php ?> <% %>). After that, the pre-processing steps (see section 3.2) are needed in order to use these texts to train a SVM classifier. Five most frequent features obtained at this stage are presented in Table 4.

Table 4. Five most frequent textual features extracted

Category	Textual features (Feature: Frequency)									
Airfare	pm:	419	airline:	279	air:	124	am:	102	airway:	100
Auto	docum:	108	car:	105	leas:	84	search:	63	make:	56
Book	search:	130	title:	110	book:	95	author:	75	new:	72
Rental	pm:	402	option:	202	am:	168	airport:	144	car:	143
Hotel	hotel:	234	pm:	228	island:	151	new:	135	room:	84
Job	job:	207	new:	125	locat:	84	service:	82	island:	81
Movie	press:	211	book:	123	s:	109	video:	107	enter-tain:	107
Music	record:	456	music:	226	sub:	156	search:	97	new:	80

5.3 Overall Performance

We conduct eight scalable experiments with our deep Web interfaces gathering framework. For each topic, we extracts 50 seeds from the DMOZ as the starting set. We save those pages and their corresponding URLs if the following two conditions are satisfied at the same time. First, they are judged to be relevant by page text classifiers. Second, each page contains at least one searchable hidden Web entrance. Because Music Records databases are sparsely distributed, our best-first crawler only locates 50 hidden entrances for this topic category. For other categories, our crawler finds 100 entrances for each of them, among which five entries about Books category are listed below:

```
http://www.rwmilitarybooks.com/
http://www.artistsbooksonline.com/index.shtm
http://www.sfbc.com/
http://www.lns.cornell.edu/~seb/scouting-books.html
http://booksox.com/
```

At last, we manually verify whether the hidden Web entrances located by our crawler are what we want. The precisions of all these categories are shown in the last column of Table 1.

6 Conclusion

In this paper, a two-step framework is proposed to automatically identify domain-specific deep Web entrances. Eight scalable experimental results demonstrate that our method can find domain-specific hidden Web entrances accurately and efficiently. The average precision of the eight representative topic categories is 0.88.

References

1. Brightplanets searchable databases directory. http://www.completeplanet.com
2. Classification Trees and Regression Trees. http://cn.mathworks.com/help/stats/classification-trees-and-regression-trees.html
3. Google Base. http://base.google.com/
4. The R Project for Statistical Computing. http://www.r-project.org
5. The uiuc Web integration repository. http://metaquerier.cs.uiuc.edu/repository/
6. Barbosa, L., Freire, J.: Searching for hidden-web databases. In: WebDB, pp. 1–6 (2005)
7. Barbosa, L., Freire, J.: Combining classifiers to identify online databases. In: Proceedings of the 16th International Conference on World Wide Web, pp. 431–440. ACM (2012)
8. Barbosa, L., Freire, J.: An adaptive crawler for locating hidden-web entry points. In: Proceedings of the 16th International Conference on World Wide Web, pp. 441–450. ACM (2013)

9. Barbosa, L., Freire, J.: Siphoning hidden-web data through keyword-based interfaces. In: SBBD, pp. 309–321 (2014)
10. Bergholz, A., Childlovskii, B.: Crawling for domain-specific hidden web resources. In: Proceedings of the Fourth International Conference on Web Information Systems Engineering, WISE 2003, pp. 125–133. IEEE (2003)
11. Chakrabarti, S., Van den Berg, M., Dom, B.: Focused crawling: a new approach to topic-specific web resource discovery. Computer Networks **31**(11), 1623–1640 (1999)
12. Chang, C.C., Lin, C.J.: Libsvm: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology (TIST) **2**(3), 27 (2011)
13. Chang, K.C.C., He, B., Zhang, Z.: Toward large scale integration: building a metaquerier over databases on the web. In: CIDR, vol. 5, pp. 44–55 (2005)
14. Cope, J., Craswell, N., Hawking, D.: Automated discovery of search interfaces on the web. In: Proceedings of the 14th Australasian Database Conference, vol. 17, pp. 181–189. Australian Computer Society, Inc. (2003)
15. Du, X., Zheng, Y., Yan, Z.: Automate discovery of deep web interfaces. In: 2010 2nd International Conference on Information Science and Engineering (ICISE), pp. 3572–3575. IEEE (2010)
16. Fetterly, D., Manasse, M., Najork, M., Wiener, J.: A large-scale study of the evolution of web pages. In: Proceedings of the 12th International Conference on World Wide Web, pp. 669–678. ACM (2003)
17. Galperin, M.Y.: The molecular biology database collection: 2008 update. Nucleic Acids Research **36**(suppl 1), D2–D4 (2008)
18. Gravano, L., García-Molina, H., Tomasic, A.: Gloss: text-source discovery over the internet. ACM Transactions on Database Systems (TODS) **24**(2), 229–264 (1999)
19. He, H., Meng, W., Yu, C., Wu, Z.: Wise-integrator: An automatic integrator of web search interfaces for e-commerce. In: Proceedings of the 29th International Conference on Very Large Data Bases, vol. 29, pp. 357–368. VLDB Endowment (2013)
20. Raghavan, S., Garcia-Molina, H.: Crawling the hidden web (2014)
21. Torgo, L., Gama, J.: Regression by classification. In: Borges, D.L., Kaestner, C.A.A. (eds.) SBIA 1996. LNCS, vol. 1159, pp. 51–60. Springer, Heidelberg (1996)
22. Wu, W., Yu, C., Doan, A., Meng, W.: An interactive clustering-based approach to integrating source query interfaces on the deep web. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, pp. 95–106. ACM (2014)
23. Xu, J., Callan, J.: Effective retrieval with distributed collections. In: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 112–120. ACM (2008)
24. Yu, C., Liu, K.L., Meng, W., Wu, Z., Rishe, N.: A methodology to retrieve text documents from multiple databases. IEEE Transactions on Knowledge and Data Engineering **14**(6), 1347–1361 (2012)